

COMS E6998: Microservices and Cloud Applications

Lecture 6: OAuth2, FB, Twitter, Cloud/AWS Security, 12 Factor Applications

Dr. Donald F. Ferguson
dff9@columbia.edu

Contents

Contents

- Questions, Comments, Discussion?
- (Web/Cloud) Security Concepts 101
 - Concepts
 - HTTPS
 - Certificates, DNS, ...
- Authentication and Registration
 - SeekaTV demos: Email, Facebook, Twitter
 - Email registration/login
 - Social network registration/login
 - OAuth2 Overview; Facebook, Twitter
- Cloud Security Concepts and AWS Realization
 - Overview: Identity, Role, Resource, Policy
 - Application level versus AWS security management
 - Services and subsystems security
- Project 4: Simple Security
- 12 Factor Application

Questions?
Comments?
Discussion

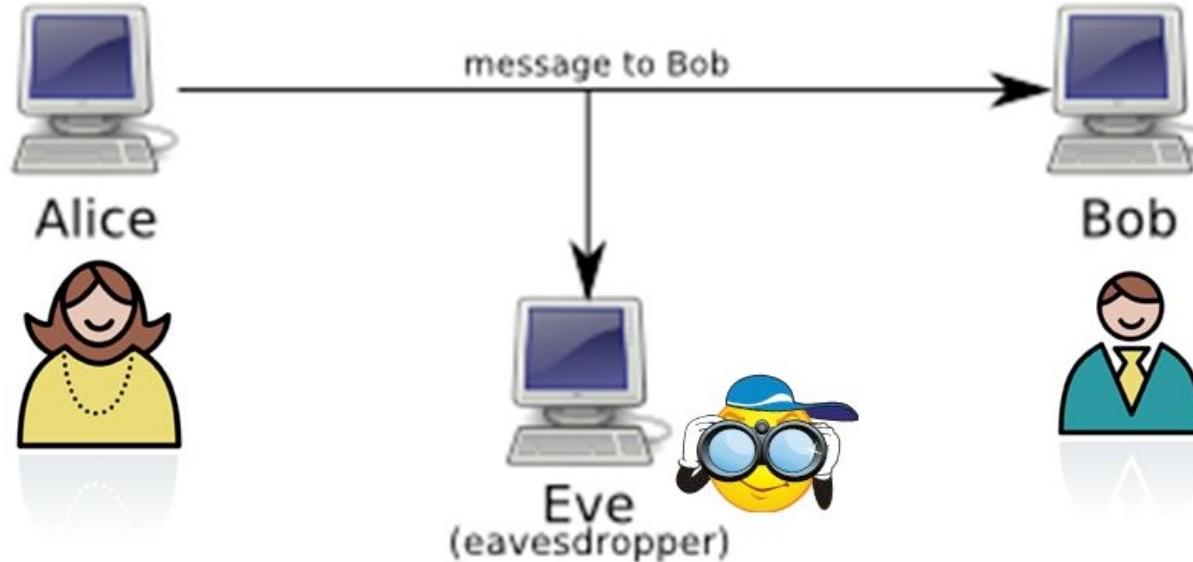
Web/Cloud Security 101

TLS/HTTPS
Certificates
DNS
CloudFront

(Some) Security Facets

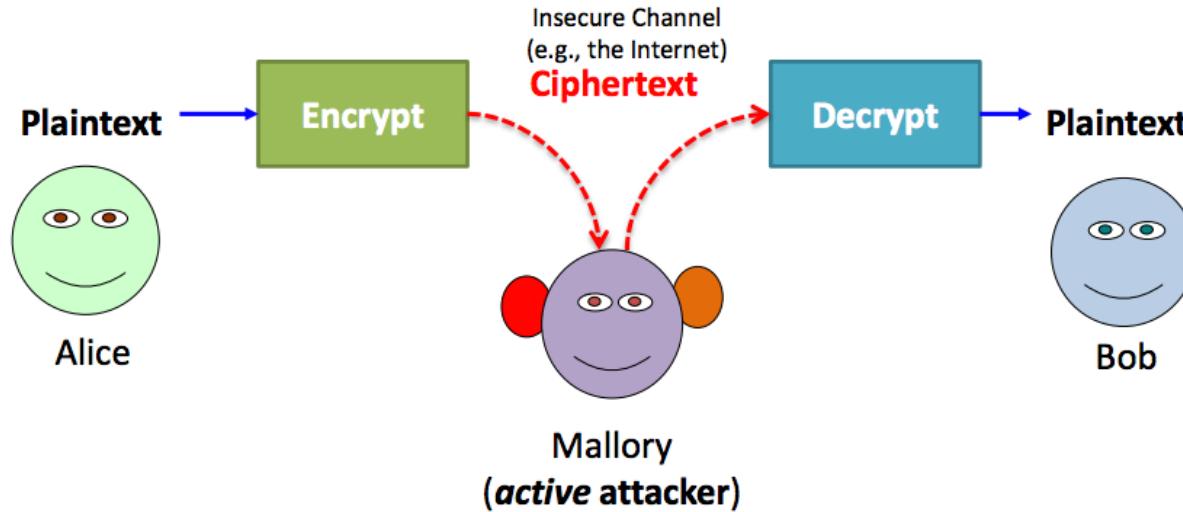
- Authentication: “(...) is the act of confirming the truth of an attribute of a single piece of data claimed true by an entity. In contrast with identification, which refers to the act of stating or otherwise indicating a claim purportedly attesting to a person or thing's identity, authentication is the process of actually confirming that identity.”
- Authorization:
 - “The process of [authorization](#) is distinct from that of authentication. Whereas authentication is the process of verifying that ‘you are who you say you are’, authorization is the process of verifying that ‘you are permitted to do what you are trying to do’.
 - “**Authorization** is the function of specifying access rights/privileges to resources related to [information security](#) and [computer security](#) in general and to [access control](#) in particular ^[1]. More formally, “to authorize” is to define an access policy.”
- (Communication) Security
 - “**Transport Layer Security (TLS)** and its predecessor, **Secure Sockets Layer (SSL)**, both frequently referred to as “SSL”, are [cryptographic protocols](#) that provide [communications security](#) over a [computer network](#).
 - “The connection is *private* (or *secure*) because [symmetric cryptography](#) is used to encrypt the data transmitted.”
 - The identity of the communicating parties can be *authenticated* using [public-key cryptography](#).
 - The connection ensures *integrity* because each message transmitted includes a message integrity check using a [message authentication code](#) to prevent undetected loss or alteration of the data during transmission.
 - **HTTPS** (also called **HTTP over Transport Layer Security [TLS]**, ^[1] **HTTP over SSL**, ^[2] and **HTTP Secure**^{[3][4]}) is a [communications protocol](#) for [secure communication](#) over a [computer network](#) which is widely used on the [Internet](#). HTTPS consists of communication over [Hypertext Transfer Protocol](#) (HTTP) within a connection encrypted by [Transport Layer Security](#), or its predecessor, Secure Sockets Layer. The main motivation for HTTPS is [authentication](#) of the visited [website](#) and protection of the [privacy](#) and [integrity](#) of the exchanged data.

Alice – Bob – Eve



- Eve observes the unencrypted communication.
- Probably to do something malicious with the observed information, e.g.
 - Steal credit card numbers.
 - Steal login IDs and passwords.

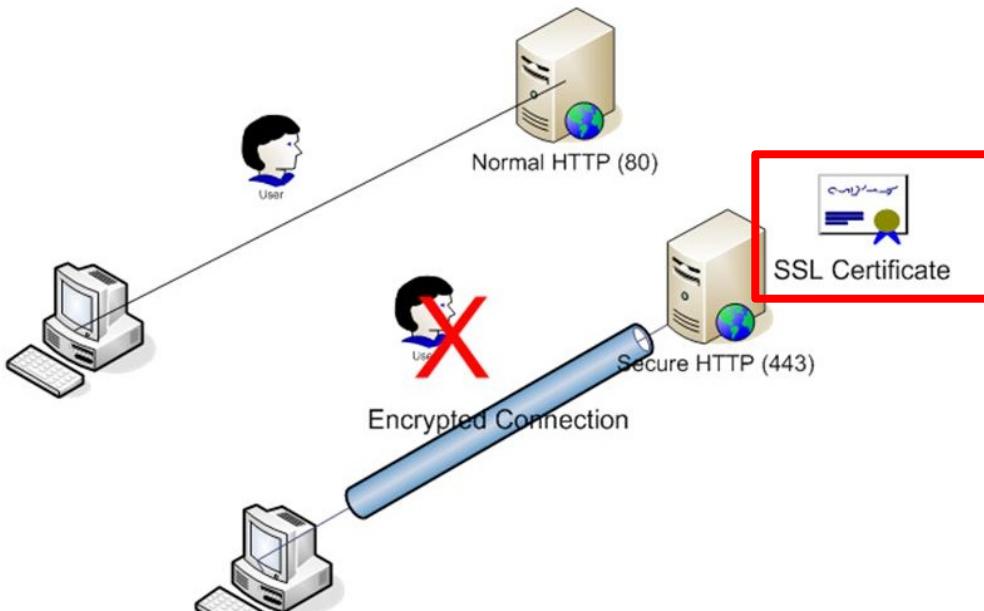
Alice – Bob – Mallory



- Mallory intercepts the communication, including any security set up, e.g.
 - Keys
 - BASIC-AUTH
- Makes Bob think she is Alice and Alice think she is Bob to steal information or send evil messages (e.g. transfer money).

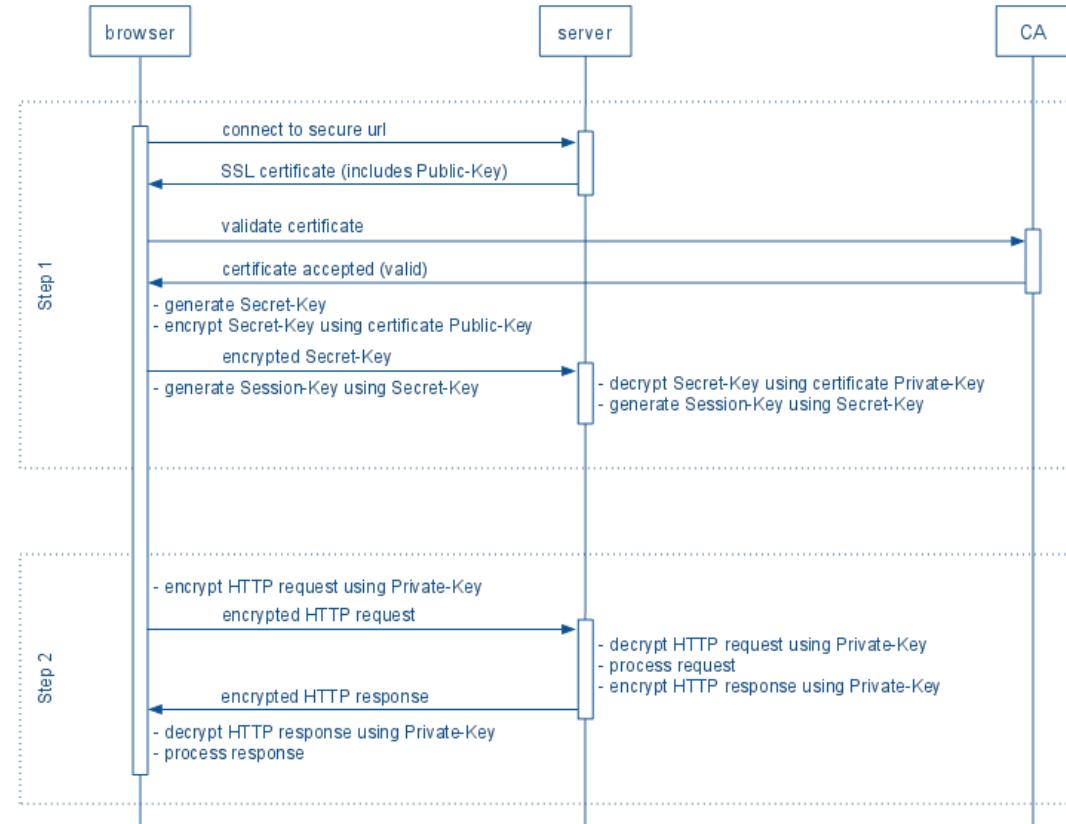
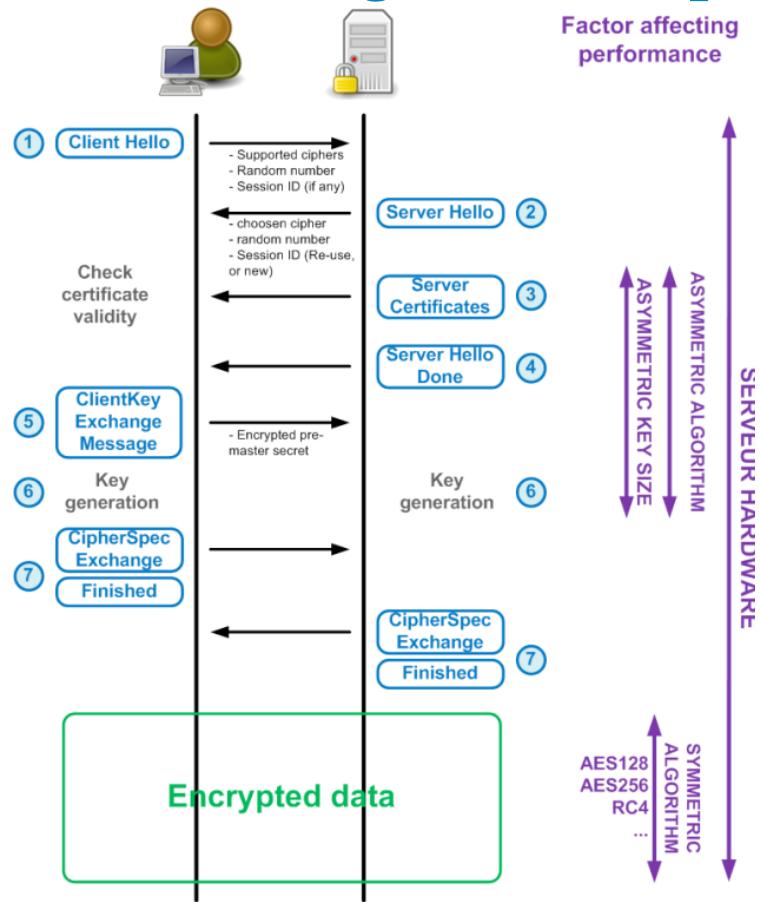
HTTPS

HTTP vs HTTPS



- Using
 - The SSL Certificate
 - PKI cryptography
- Bob can *prove* identity to Alice.
- Alice can send messages to Bob that only Bob can decrypt.
- This allows a
 - Key exchange
 - To set up a secure channel
- But, Alice has not proven identity to Bob.

Two Diagrams Explaining HTTPS/TLS



HTTPS Example

The screenshot shows a web browser window for Bank of America. The address bar indicates a secure connection to 'Bank of America Corporation [US]' at <https://www.bankofamerica.com>. The developer tools are open, specifically the 'Security' panel, which displays a 'Security overview' for the main origin. The overview states that the page is secure (valid HTTPS) and lists a valid certificate issued by Symantec Class 3 EV SSL CA - G3. It also notes that all resources on the page are served securely. A warning is present about obsolete connection settings, stating that the connection uses TLS 1.2, RSA, and AES_256_GCM. The 'Secure Sign-in' form on the left is highlighted with a red box, and a 'Get started' button is visible below it.

Open a checking account

Get extra protection with a debit chip card when used at chip-enabled terminals or ATMs.

[Get started](#)

- I am connected to the owner of *bankofamerica.com* →
- I can send the site a shared secret, and be sure no one else can see it.
- This works for REST because REST is still HTTP(S).

Certificate

The screenshot shows a web browser with the Bank of America homepage open at <https://www.bankofamerica.com>. A detailed certificate analysis is overlaid on the page, showing the certificate chain and individual certificate details.

Certificate Chain:

- VeriSign Class 3 Public Primary Certification Authority - G5
- Symantec Class 3 EV SSL CA - G3
- www.bankofamerica.com

www.bankofamerica.com Certificate Details:

- Issued by: Symantec Class 3 EV SSL CA - G3
- Expires: Thursday, July 26, 2018 at 7:59:59 PM Eastern
- This certificate is valid

Details:

Subject Name	
Inc. Country	US
Inc. State/Province	Delaware
Business Category	Private Organization
Serial Number	2927442
Country	US
Postal Code	60603
State/Province	Illinois
Locality	Chicago
Street Address	135 S La Salle St
Organization	Bank of America Corporation
Organizational Unit	eComm Network Infrastructure
Common Name	www.bankofamerica.com

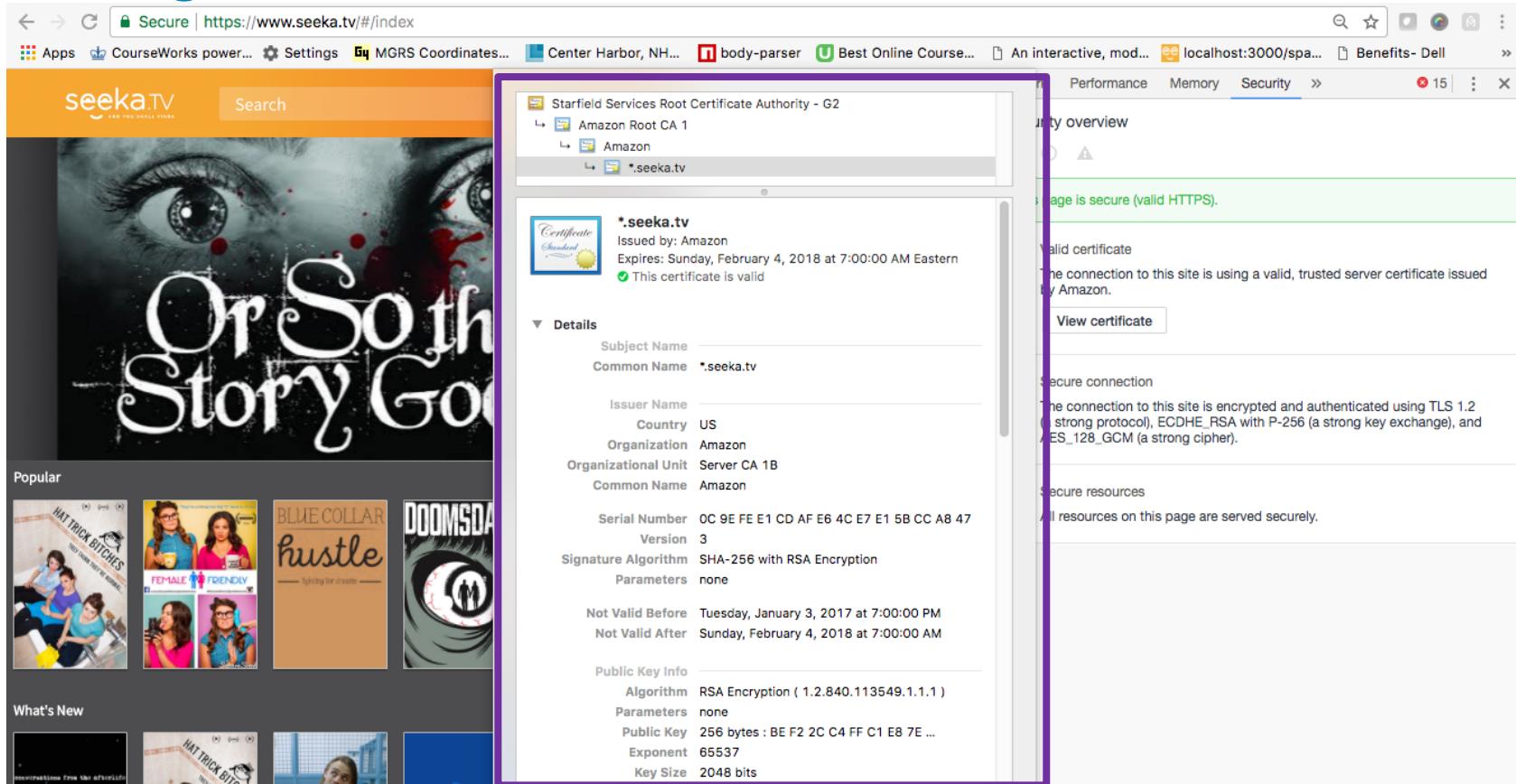
Issuer Name:

Country	US
Organization	Symantec Corporation
Organizational Unit	Symantec Trust Network
Common Name	Symantec Class 3 EV SSL CA - G3

Browser Security Overview:

- Security overview
- This page is secure (valid HTTPS).
- Valid certificate
- The connection to this site is using a valid, trusted server certificate issued by Symantec Class 3 EV SSL CA - G3.
- [View certificate](#)
- Secure resources
- All resources on this page are served securely.
- Obsolete connection settings
- The connection to this site uses TLS 1.2 (a strong protocol), RSA (an obsolete key exchange), and AES_256_GCM (a strong cipher).

Using Your Certificate



The screenshot shows a web browser window with the URL <https://www.seeka.tv/#/index>. The main content area displays the homepage of seeka.TV, featuring a large eye and the text "Or So the Story Go". Below this are sections for "Popular" and "What's New" with various thumbnail images. A certificate details dialog box is overlaid on the page, showing the following information:

Starfield Services Root Certificate Authority - G2

- Amazon Root CA 1
- Amazon
- *.seeka.tv

***.seeka.tv**

Issued by: Amazon
Expires: Sunday, February 4, 2018 at 7:00:00 AM Eastern
This certificate is valid

Details

Subject Name	-----
Common Name	*.seeka.tv
Issuer Name	-----
Country	US
Organization	Amazon
Organizational Unit	Server CA 1B
Common Name	Amazon
Serial Number	0C 9E FE E1 CD AF E6 4C E7 E1 5B CC A8 47
Version	3
Signature Algorithm	SHA-256 with RSA Encryption
Parameters	none
Not Valid Before	Tuesday, January 3, 2017 at 7:00:00 PM
Not Valid After	Sunday, February 4, 2018 at 7:00:00 AM
Public Key Info	-----
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	none
Public Key	256 bytes : BE F2 2C C4 FF C1 E8 7E ...
Exponent	65537
Key Size	2048 bits

The browser's security tab shows the following details:

- Validity overview: The page is secure (valid HTTPS).
- Valid certificate: The connection to this site is using a valid, trusted server certificate issued by Amazon.
- View certificate: A button to view the full certificate details.
- Secure connection: The connection is encrypted and authenticated using TLS 1.2 (a strong protocol), ECDHE_RSA with P-256 (a strong key exchange), and AES_128_GCM (a strong cipher).
- Secure resources: All resources on this page are served securely.

Using Your Certificate

CloudFront Distributions > E222FFEHQ8PIDH



General Origins Behaviors Error Pages Restrictions Invalidations Tags

Edit

Distribution ID E222FFEHQ8PIDH
ARN arn:aws:cloudfront:██distribution/E222FFEHQ8PIDH
Log Prefix -
Delivery Method Web
Cookie Logging Off
Distribution Status Deployed
Comment Preview
Price Class Use Only US, Canada and Europe
AWS WAF Web ACL Dev Seeka TV
State Enabled

Alternate Domain Names (CNAMEs) preview.seeka.tv
SSL Certificate *.seeka.tv (811acf0██-ead71ece6abe)
Domain Name d2vjg1f5uddyw3.cloudfront.net
Custom SSL Client Support Only Clients that Support Server Name Indication (SNI)
Security Policy TLS1

Supported HTTP Versions HTTP/2, HTTP/1.1, HTTP/1.0
IPv6 Enabled
Default Root Object index.html
Last Modified 2017-07-10 14:44 UTC-4
Log Bucket -

Using Your Certificate

Request a certificate Import a certificate Actions ▾

Viewing 1 to 1 of 1 certificates

	Name ▾	Domain name ▾	Additional names	Status ▾	Type ▾	In use? ▾
<input type="checkbox"/>		*.seeka.tv		Issued	Amazon Issued	Yes

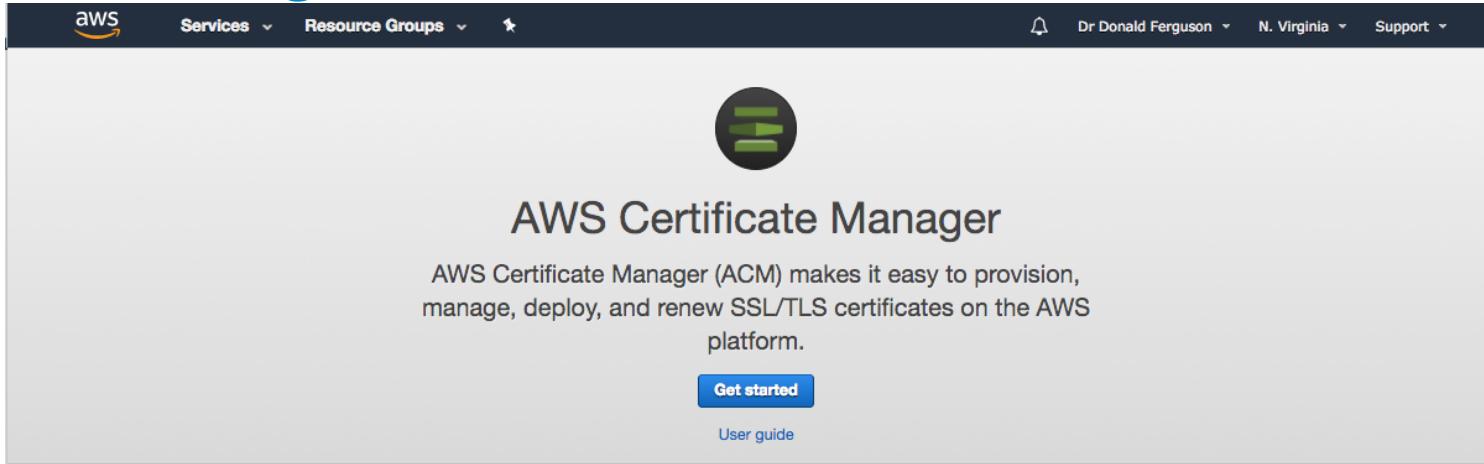
Status

Status Issued
Detailed status The certificate was issued at 2017-01-04T15:16:26UTC

Details

Type	Amazon Issued	Requested at	2017-01-04T15:07:01UTC
In use?	Yes	Issued at	2017-01-04T15:16:26UTC
Domain name	*.seeka.tv	Not before	2017-01-04T00:00:00UTC
Number of additional names	0	Not after	2018-02-04T12:00:00UTC
Identifier	811acf02-5f1e-4a73-8ead71ece6abe	Public key info	RSA 2048-bit
Serial number	0c:9e:fe:e1:0f:5f:04:73:15:b:cc:a8:47:eb:3c	Signature algorithm	SHA256WITHRSA
Associated resources	arn:aws:cloudfront:██:distribution/E222FFEHQ8PIDH, arn:aws:cloudfront:██:distribution/E2W9S9P9B47C0G,	ARN	arn:aws:acm:us-east-1:54██:certificate/811acf02-5f1e-4cee-a832-ead71ece6abe

Obtaining a Certificate



The screenshot shows the AWS Certificate Manager (ACM) console. At the top, there is a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, and a star icon. On the right side of the navigation bar are 'Dr Donald Ferguson', 'N. Virginia', and 'Support' dropdowns, along with a bell icon. The main content area features a large circular icon with three horizontal bars, followed by the text 'AWS Certificate Manager'. Below this, a description reads: 'AWS Certificate Manager (ACM) makes it easy to provision, manage, deploy, and renew SSL/TLS certificates on the AWS platform.' There are two buttons: a blue 'Get started' button and a smaller 'User guide' link. Below this section, there are three main features displayed as cards:

- Provision certificates**: Shows an icon of a computer monitor with a key and a plus sign. Description: 'Provide the name of your site, establish your identity, and let ACM do the rest. ACM manages renewal of SSL/TLS certificates issued by Amazon for you.'
- Deploy SSL/TLS-based sites and applications**: Shows an icon of a person silhouette with a lock and a gear. Description: 'Create an Elastic Load Balancer or Amazon CloudFront distribution and use ACM-provided or imported certificates with SSL/TLS to securely identify your site.'
- Manage certificates**: Shows an icon of a computer monitor with a cloud and a cursor. Description: 'See all of your ACM-provided and imported certificates in one place in the AWS Management Console. Automate management tasks by using the ACM API, SDK, or CLI.'

You can use AWS Certificate Manager certificates with other AWS Services.

Choose **Import a certificate** to import an existing certificate instead of requesting a new one. [Learn more.](#)

 **Import a certificate**

Add domain names

Type the fully qualified domain name of the site you want to secure with an SSL/TLS certificate (for example, www.example.com). Use an asterisk (*) to request a wildcard certificate to protect several sites in the same domain. For example: *.example.com protects www.example.com, site.example.com and images.example.com.

Domain name*

Remove

donald-ferguson.net

Add another name to this certificate

You can add additional names to this certificate. For example, if you're requesting a certificate for "www.example.com", you might want to add the name "example.com" so that customers can reach your site by either name.
[Learn more.](#)

***At least one domain name is required**

Cancel

Review and request

Review and request

After you request the certificate, email will be sent to the registered owner of each domain name below. The domain owner or an authorized representative can validate control of the domain and approve the certificate by following the instructions in the body of the email. After all of the domains are validated, the certificate will be issued.

Domain name

The name you want to secure with an SSL/TLS certificate.

Domain name donald-ferguson.net

Cancel

Previous

Confirm and request

Request a certificate

Step 1: Add domain names

Step 2: Review and request

Step 3: Validation



Request in progress

A certificate request with a status of Pending validation has been created. Further action is needed to complete the validation and approval of the certificate.

Validation



We will send email to the registered owner of each domain listed below. To validate control of the domain, the owner of the domain or an authorized representative must go to the Amazon certificate approval website and approve the request. Further instructions are provided in the body of the email.

▼ **donald-ferguson.net**

webmaster@donald-ferguson.net
postmaster@donald-ferguson.net
hostmaster@donald-ferguson.net
admin@donald-ferguson.net
DONFF2@AOL.COM
administrator@donald-ferguson.net

If you or an authorized representative did not receive the email we sent, or if you want to learn more, click the help icon (?) above.

Continue

[Lookup](#)

Showing results for: DONALD-FERGUSON.NET

Original Query: donald-ferguson.net

Contact Information

Registrant Contact

Name: DONALD FERGUSON
Organization:
Mailing Address: [REDACTED]
ROA [REDACTED] 39 US
Phone: +1.91454 [REDACTED]
Ext:
Fax: +1.914 [REDACTED]
Fax Ext:
Email: [REDACTED]@AOL.COM

Admin Contact

Name: DONALD FERGUSON
Organization:
Mailing Address: [REDACTED]
ROA [REDACTED] 39 US
Phone: +1.91454 [REDACTED]
Ext:
Fax: +1.914 [REDACTED]
Fax Ext:
Email: [REDACTED]@AOL.COM

Tech Contact

Name: DONALD FERGUSON
Organization:
Mailing Address: [REDACTED]
ROA [REDACTED] 39 US
Phone: +1.91454 [REDACTED]
Ext:
Fax: +1.914 [REDACTED]
Fax Ext:
Email: [REDACTED]@AOL.COM

[Submit WHOIS](#)
[WHOIS](#)

[WHOIS](#)

Certificate approval for donald-ferguson.net



 Amazon Certificates (no-reply@certificates.amazon.com)

Sun, Oct 22, 2017 1:49 pm

To: you [Details](#) 



Greetings from Amazon Web Services,

We received a request to issue an SSL/TLS certificate for **donald-ferguson.net**.

Verify that the following domain, AWS account ID, and certificate identifier correspond to a request from you or someone in your organization.

Domain: **donald-ferguson.net**

AWS account ID: 8**██████████**0

AWS Region name: **us-east-1**

Certificate identifier: **41128081-██████████87-811f2810fd38**

To approve this request, go to [Amazon Certificate Approvals](https://us-east-1.certificates.amazon.com/approvals?code=ecbea72d-6710-4e59-9ec4-80c4439b1cd0&context=61b77bcd-0c95-4fc7-82ed-825fc610a764-75732d656173742d31) (<https://us-east-1.certificates.amazon.com/approvals?code=ecbea72d-6710-4e59-9ec4-80c4439b1cd0&context=61b77bcd-0c95-4fc7-82ed-825fc610a764-75732d656173742d31>) and follow the instructions on the page.



Amazon Web Services (AWS) has received a request to issue an SSL certificate for donald-ferguson.net. You are listed as one of the authorized representatives for this domain name. Your authorization is required prior to issuing this certificate.

Verify that the domain name, AWS account ID, and certificate identifier below correspond to a request from you or a person authorized to request certificates for this domain name.

Domain name	donald-ferguson.net
AWS account number	8327-2025-5830
AWS Region	us-east-1
Certificate identifier	4112808 [REDACTED] 11f2810fd38

Review the information presented above and click **I Approve** only if you recognize the request and the account requesting it. By clicking **I Approve**, you authorize Amazon to request a certificate for the above domain name.

Success

Request a certificate Import a certificate Actions ▾

Viewing 1 to 1 of 1 certificates

	Name ▾	Domain name ▾	Additional names	Status ▾	Type ▾	In use? ▾
<input type="checkbox"/>		donald-ferguson.net		Issued	Amazon Issued	No

Status

Status Issued
Detailed status The certificate was issued at 2017-10-22T17:51:30UTC

Details

Type	Amazon Issued	Requested at	2017-10-22T17:48:55UTC
In use?	No	Issued at	2017-10-22T17:51:30UTC
Domain name	donald-ferguson.net	Not before	2017-10-22T00:00:00UTC
Number of additional names	0	Not after	2018-11-22T12:00:00UTC
Identifier	41128081-[REDACTED]-811f2810fd38	Public key info	RSA 2048-bit
Serial number	08:d8:2-[REDACTED]:f9:d9:55:bb:1b	Signature algorithm	SHA256WITHRSA
		ARN	arn:aws:acm:us-east-1:83272025-[REDACTED]-20ce-4088-ab87-811f2810fd38

Tags

CloudFront Distribution

CloudFront Distributions > E2ZFLMK3IJGTRR

General **Origins** **Behaviors** **Error Pages** **Restrictions** **Invalidations** **Tags**

Edit

Distribution ID	E2ZFLMK3IJGTRR
ARN	arn:aws:cloudfront::832[REDACTED]ZFLMK3IJGTRR
Log Prefix	-
Delivery Method	Web
Cookie Logging	Off
Distribution Status	InProgress
Comment	-
Price Class	Use All Edge Locations (Best Performance)
AWS WAF Web ACL	-
State	Enabled
Alternate Domain Names (CNAMEs)	
SSL Certificate	donald-ferguson.net (4[REDACTED]11f2810fd38)
Domain Name	d2und4t9a1xcv.j.cloudfront.net
Custom SSL Client Support	Only Clients that Support Server Name Indication (SNI)
Security Policy	TLSv1.1_2016
Supported HTTP Versions	HTTP/2, HTTP/1.1, HTTP/1.0
IPv6	Enabled
Default Root Object	-
Last Modified	2017-10-22 16:54 UTC-4
Log Bucket	-

CloudFront Distribution

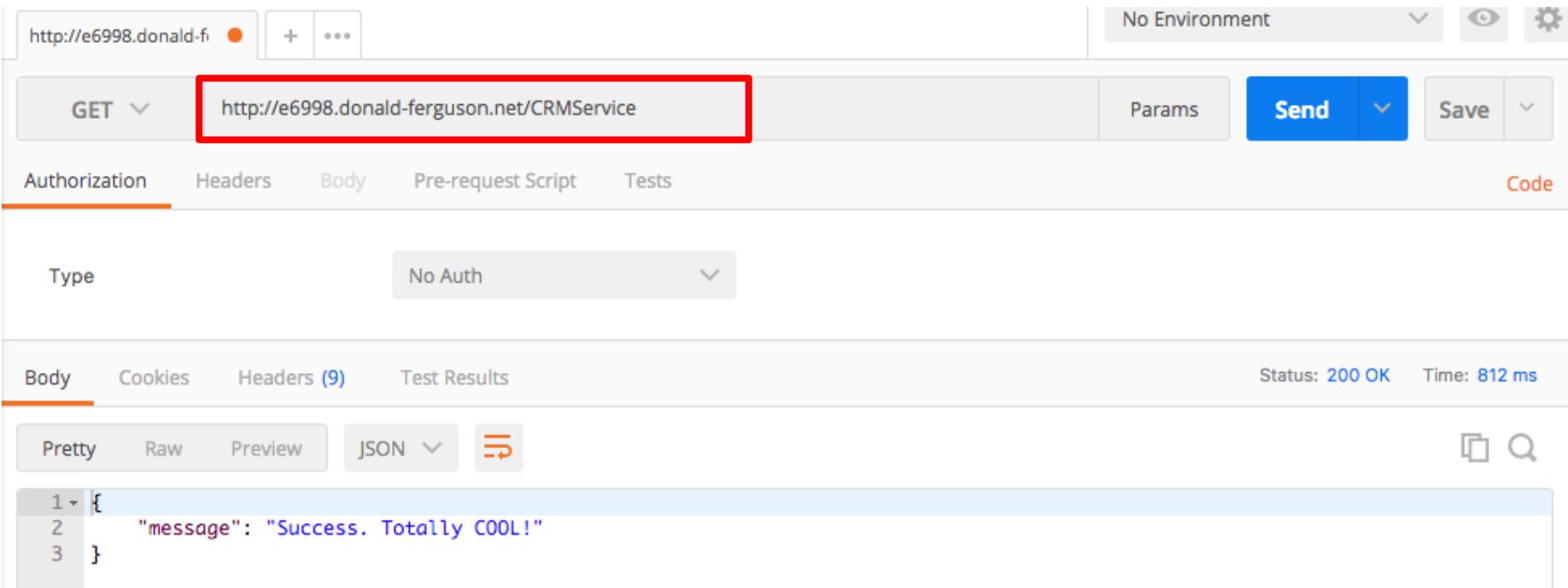
CloudFront Distributions > E2ZFLMK3IJGTRR

	Origin Domain Name and Path	Origin ID	Origin Type	Origin Access Identifier	Origin Protocol Policy	HTTPS Port	HTTP Port
<input type="checkbox"/>	dff-columbia.s3-website-us-east-1.amazonaws.com	S3-dff-columbia	Custom Origin	-	HTTP Only	443	80
<input type="checkbox"/>	fh26i8s0aa.execute-api.us-east-1.amazonaws.com/prod/CRMService	Custom-fh26i8s0aa.execute-api.us-east-1.amazonaws.com	Custom Origin	-	HTTP Only	443	80
<input type="checkbox"/>	elasticbeanstalk-us-west-2-832720255830.s3.amazonaws.com	S3-elasticbeanstalk-us-west-2-832720255830	S3 Origin	-	-	-	-

CloudFront Distributions > E2ZFLMK3IJGTRR

	Precedence	Path Pattern	Origin	Viewer Protocol Policy	Forwarded Query Strings	Trusted Signers
<input type="checkbox"/>	0	/CRMService	Custom-fh26i8s0aa.execute-api.us-east-1.amazonaws.com	HTTP and HTTPS	No	-
<input type="checkbox"/>	1	Default (*)	S3-dff-columbia	HTTP and HTTPS	No	-

Behaviors



The screenshot shows the Postman application interface. At the top, there is a header bar with a URL field containing `http://e6998.donald-f...`, a red 'Delete' button, a '+' button, a '...' button, and a dropdown for 'No Environment'. To the right are buttons for 'Send', 'Save', and settings. Below the header, the main interface shows a 'GET' request with the URL `http://e6998.donald-ferguson.net/CRMService` highlighted with a red box. To the right of the URL are buttons for 'Params', 'Send', and 'Save'. Below the URL, tabs for 'Authorization', 'Headers', 'Body', 'Pre-request Script', and 'Tests' are visible, with 'Authorization' being the active tab. The 'Body' tab is also active. On the right, a 'Code' button is visible. Under the 'Body' tab, a 'Type' dropdown is set to 'No Auth'. The 'Headers' tab shows '(9)' entries. The 'Test Results' tab shows a status of 'Status: 200 OK' and a time of '812 ms'. Below these tabs, there are buttons for 'Pretty', 'Raw', 'Preview', and 'JSON' (with a dropdown arrow). To the right of these buttons are icons for copy, search, and refresh. The 'Preview' section displays a JSON response with the following content:

```
1 [ {  
2   "message": "Success. Totally COOL!"  
3 } ]
```

Behavior

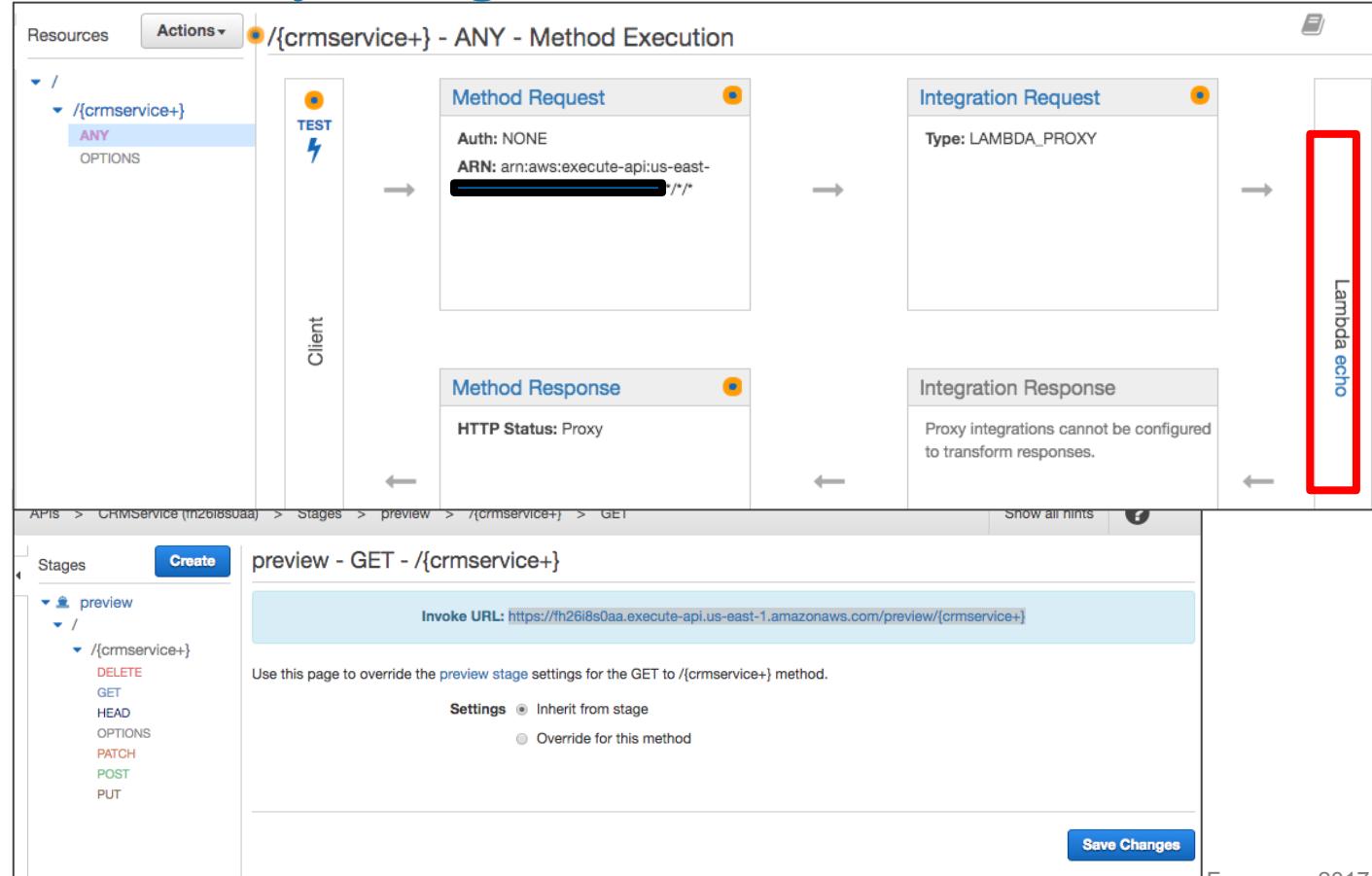
Edit Behavior

Cache Behavior Settings

Path Pattern	/CRMService	i
Origin	Custom-fh26i8s0aa.execute-api.us-east-1.amazonaws.com	i
Viewer Protocol Policy	<input checked="" type="radio"/> HTTP and HTTPS <input type="radio"/> Redirect HTTP to HTTPS <input type="radio"/> HTTPS Only	i
Allowed HTTP Methods	<input checked="" type="radio"/> GET, HEAD <input type="radio"/> GET, HEAD, OPTIONS <input type="radio"/> GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE	i
Cached HTTP Methods	GET, HEAD (Cached by default)	i
Cache Based on Selected Request Headers	None (Improves Caching) ▼	i
Learn More		
Object Caching	<input type="radio"/> Use Origin Cache Headers <input checked="" type="radio"/> Customize	i
Learn More		

Invoke application
REST APIs via API Gateway

API Gateway Stage



Implementation

Lambda > Functions > echo

ARN - arn:aws:lambda:us-east-1:832720255830:function:echo

echo

Qualifiers ▾ Actions ▾ GatewayTest ▾ **Test**

Configuration Triggers Monitoring

▼ Function code

Code entry type: Edit code inline Runtime: Node.js 6.10 Handler: index.handler

```
index.js
14  *
15  * To scan a DynamoDB table, make a GET request with the TableName as a
16  * query string parameter. To put, update, or delete an item, make a POST,
17  * PUT, or DELETE request respectively, passing in the payload to the
18  * DynamoDB API as a JSON body.
19  */
20  exports.handler = (event, context, callback) => {
21      console.log('Received event:', JSON.stringify(event, null, 2));
22
23      const done = (err, res) => callback(null, {
24          statusCode: err ? '400' : '200',
25          body: err ? err.message : JSON.stringify(res),
26          headers: {
27              'Content-Type': 'application/json',
28          },
29      });
30
31      done(null, { message: "Success. Totally COOL!"});
32  }
```

Behaviors

The screenshot shows the Postman application interface. At the top, there is a header with 'GET' and a URL input field containing 'http://e6998.donald-ferguson.net/testFBLogin1.html'. The URL field is highlighted with a red box. To the right of the URL are buttons for 'Params', 'Send', 'Save', and a dropdown. Below the header, there are tabs for 'Authorization', 'Headers', 'Body', 'Pre-request Script', and 'Tests'. The 'Authorization' tab is selected and highlighted with an orange underline. To the right of these tabs is a 'Code' button. The main content area shows a dropdown for 'Type' with 'No Auth' selected. Below this, there are tabs for 'Body', 'Cookies', 'Headers (11)', and 'Test Results'. The 'Body' tab is selected and highlighted with an orange underline. To the right of these tabs are 'Status: 200 OK' and 'Time: 425 ms'. The 'Body' section contains a code editor with line numbers 1 through 15. The code is an HTML document with AngularJS and Facebook API integration. The code editor has a 'Pretty' button, a 'Raw' button, a 'Preview' button, and an 'HTML' dropdown. To the right of the code editor are two icons: a copy icon and a search icon.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>Facebook Login JavaScript Example</title>
5  <meta charset="UTF-8">
6  </head>
7  <body>
8  <div ng-app="apiController" ng-controller="myCtrl">
9  <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
10 <script src="/apiController.js"></script>
11 <script>
12
13
14 // This is called with the results from from FB.getLoginStatus().
15 function statusChangeCallback(response) {
```

Behavior

Edit Behavior

Default Cache Behavior Settings

Path Pattern	Default (*)	i
Origin	S3-dff-columbia	i
Viewer Protocol Policy	<input checked="" type="radio"/> HTTP and HTTPS <input type="radio"/> Redirect HTTP to HTTPS <input type="radio"/> HTTPS Only	i
Allowed HTTP Methods	<input type="radio"/> GET, HEAD <input type="radio"/> GET, HEAD, OPTIONS <input checked="" type="radio"/> GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE	i
Cached HTTP Methods	GET, HEAD (Cached by default) <input type="checkbox"/> OPTIONS	i
Cache Based on Selected Request Headers	None (Improves Caching) ▾ Learn More	i
Object Caching	<input checked="" type="radio"/> Use Origin Cache Headers <input type="radio"/> Customize Learn More	i

Static content in S3

Implementation

Amazon S3 > dff-columbia

Overview Properties Permissions Management

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

Upload + Create folder More

Name	Last modified	Size	Storage class
apiController.js	Oct 22, 2017 4:40:49 PM	3.0 KB	Standard
error.html	Oct 22, 2017 4:41:12 PM	4.6 KB	Standard
facebook_login.png	Oct 22, 2017 4:41:12 PM	4664	Standard
old-index.html	Oct 22, 2017 4:40:49 PM	3.0 KB	Standard
testFBLogin1.html	Oct 22, 2017 4:41:12 PM	4.6 KB	Standard
testFBLogin3.html	Oct 6, 2016 8:51:51 AM	2.9 KB	Standard

Amazon S3 > dff-columbia

testFBLogin1.html Latest version ▾

Overview Properties Permissions

Open Download Download as Make public Copy path

Owner donff2

Last modified Oct 22, 2017 4:41:12 PM

Etag de354210290c5ae17549cd866d64920a

Storage class Standard

Server side encryption None

Size 4664

Link <https://s3.amazonaws.com/dff-columbia/testFBLogin1.html>

All of this to securely get to ...

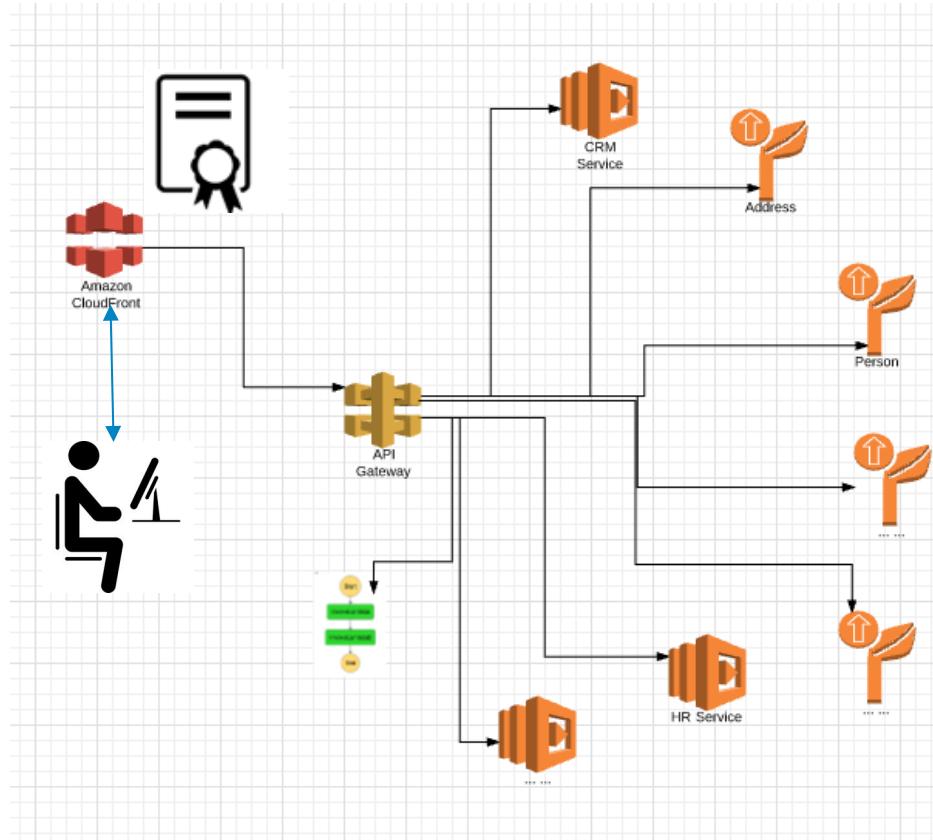


The screenshot shows a web browser window with the following details:

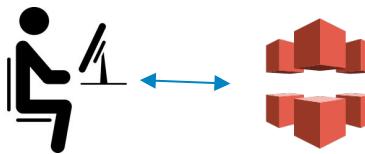
- Address Bar:** Secure | <https://e6998.donald-ferguson.net/testFBLogin1.html>
- Toolbar:** Apps, CourseWorks power..., Settings, MGRS Coordinates..., Center Harbor, NH..., body-parser, Best Online Course..., An interactive, mod...
- Content Area:**
 - Section Header:** **COMSE6998-014: Microservice and Cloud Applications**
 - Text:** This is without a doubt the totally coolest course on the Columbia syllabus, taught by an adjunct professor who is too cool for school. In this lecture you learn:
 - List:** 1. How to login to an application with Facebook.
2. General concepts around OAuth2
3. How this all fits into a larger security model.
 - Facebook Login Button:** A large blue button with the Facebook logo and the text "Continue with Facebook".
 - Text:** Log In with Facebook
 - Input Fields:** Full Name: {{name}}, email: {{email}}, security token: {{secret}}

What Did We Set Up?

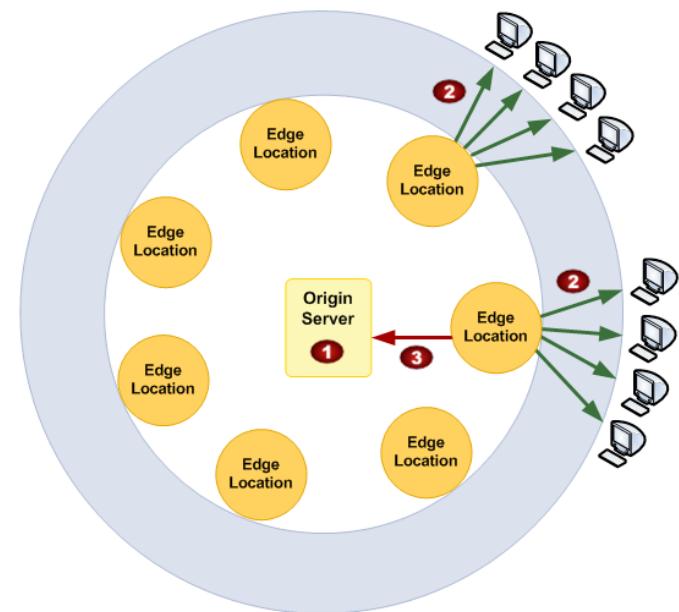
- We
 - Provide ownership of a domain.
 - Obtained a certificate.
 - Installed the certificate.
- Alice
 - Whether a person or REST client.
 - Can *authenticate* Bob.
 - And *authenticate* to Bob if,
 - They share a secret
 - Or trust someone else.



In Reality



The client – CloudFront connection is distributed and extremely local with file and REST API caching.



CDN Benefits: Global CloudFront Network



© 2013 Amazon.com, Inc. and its affiliates. All rights reserved. May not be copied, modified or distributed in whole or in part without the express consent of Amazon.com, Inc.

Once Again, ...

- This was a very AWS specific explanation for simplicity.
- There are lots of technology choices for individual elements.
- Any realistic, non-trivial application solution requires
 - HTTPS
 - Obtaining and installing certificates.
 - Something implementing a single site image for multiple systems, and perhaps multiple data centers.
 - Caching and request routing.
 - API management and optimization.

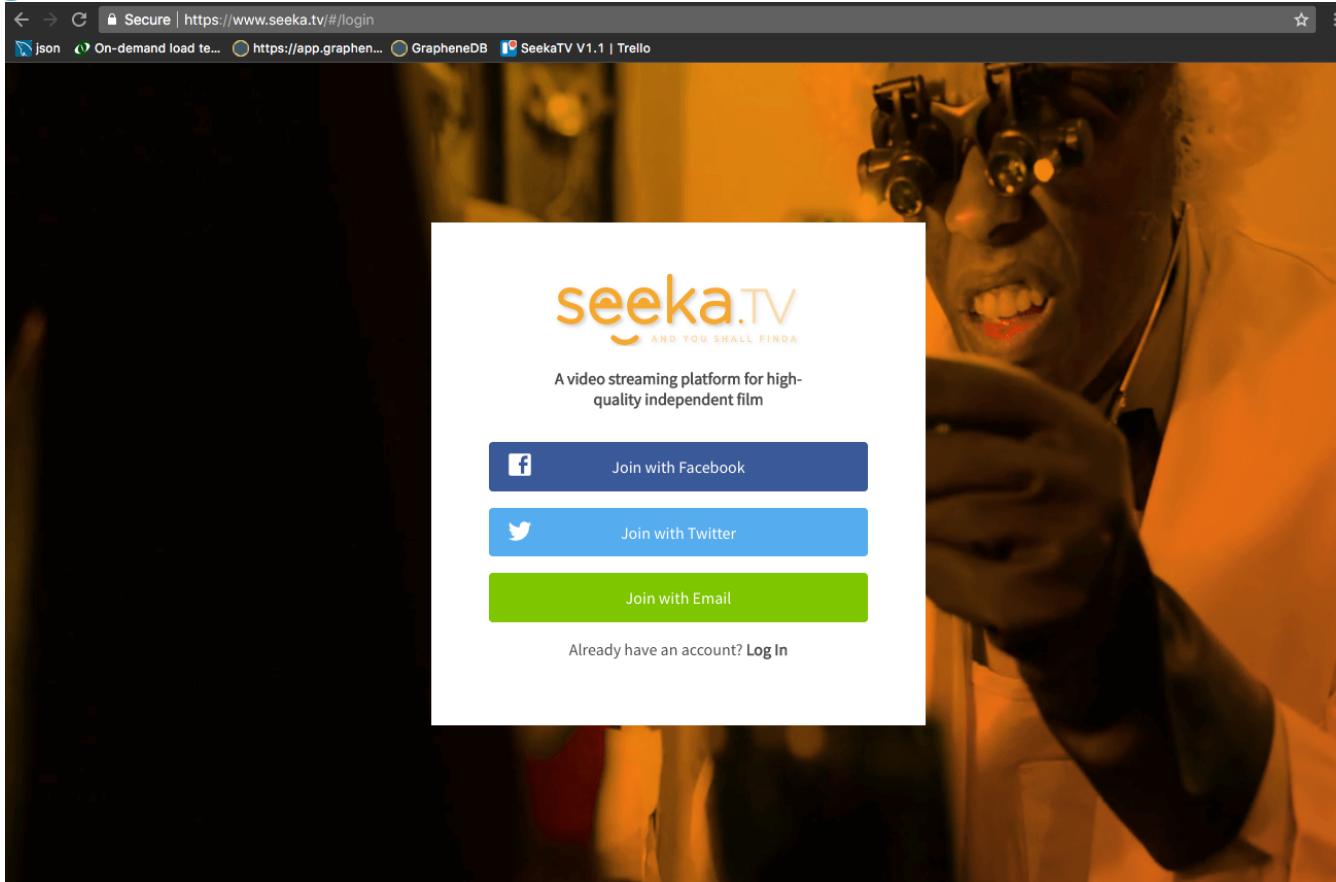
Seeka TV Demo

Email, FB, Twitter

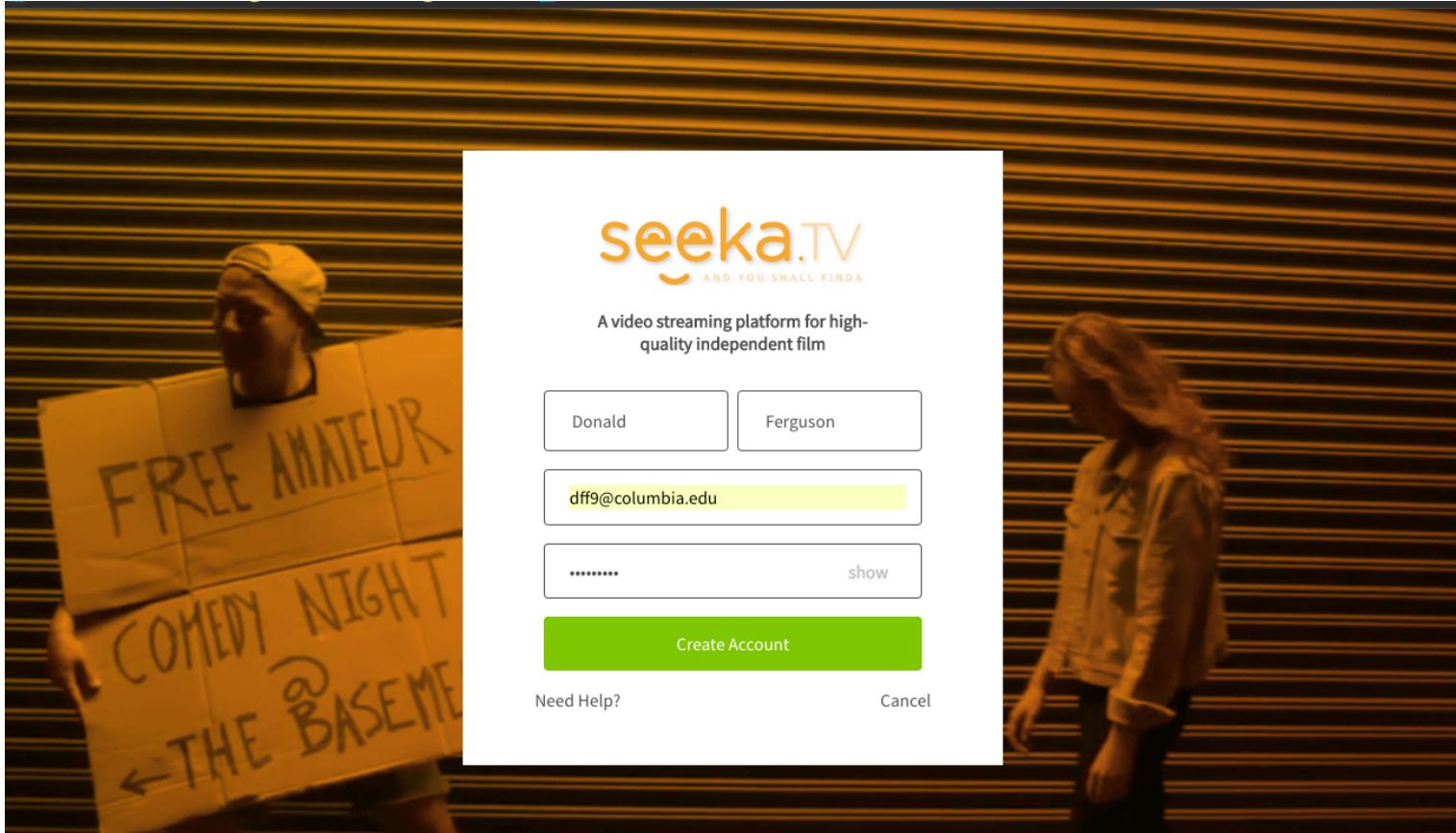
OAuth2

Demo

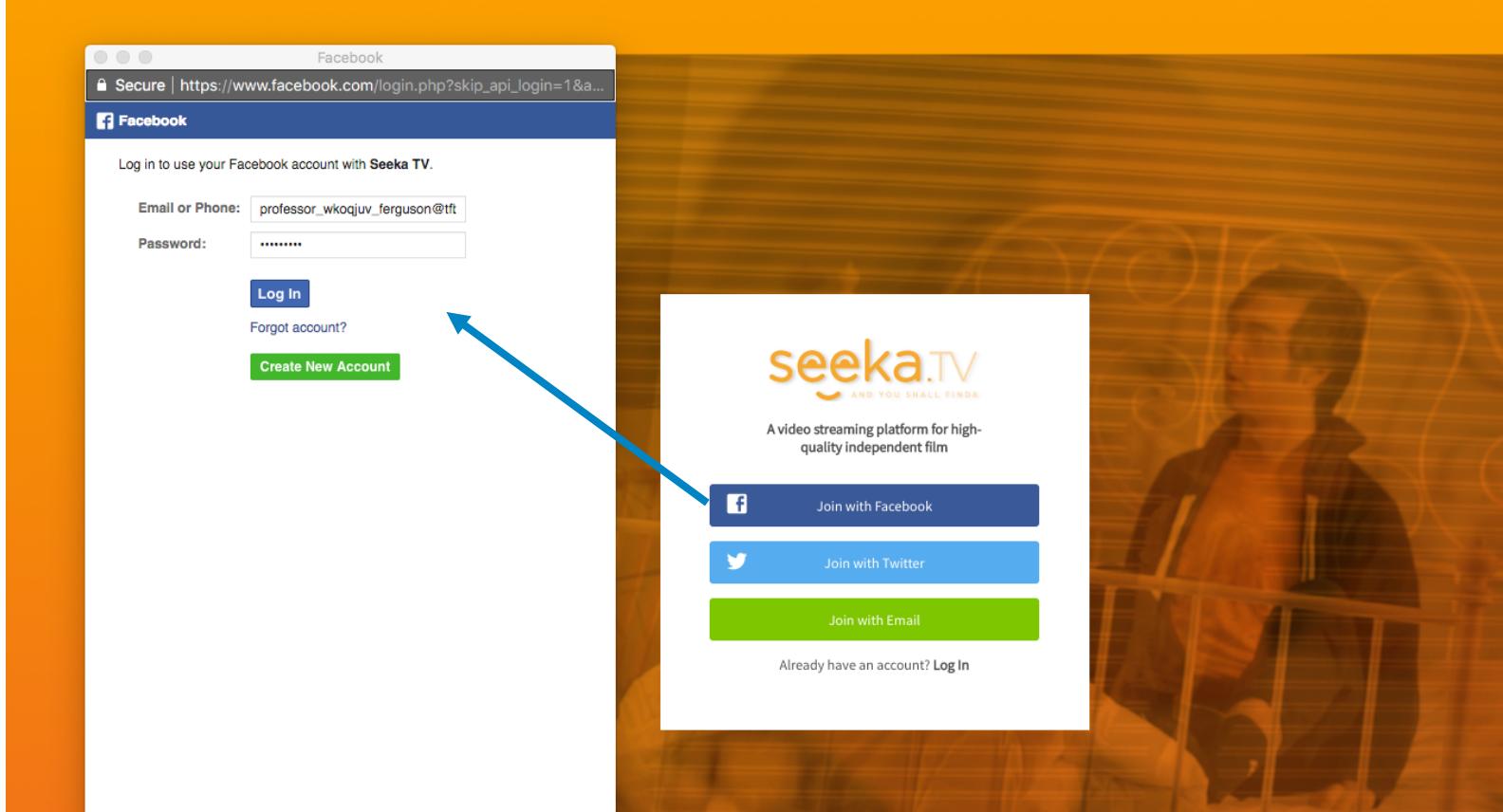
Registration



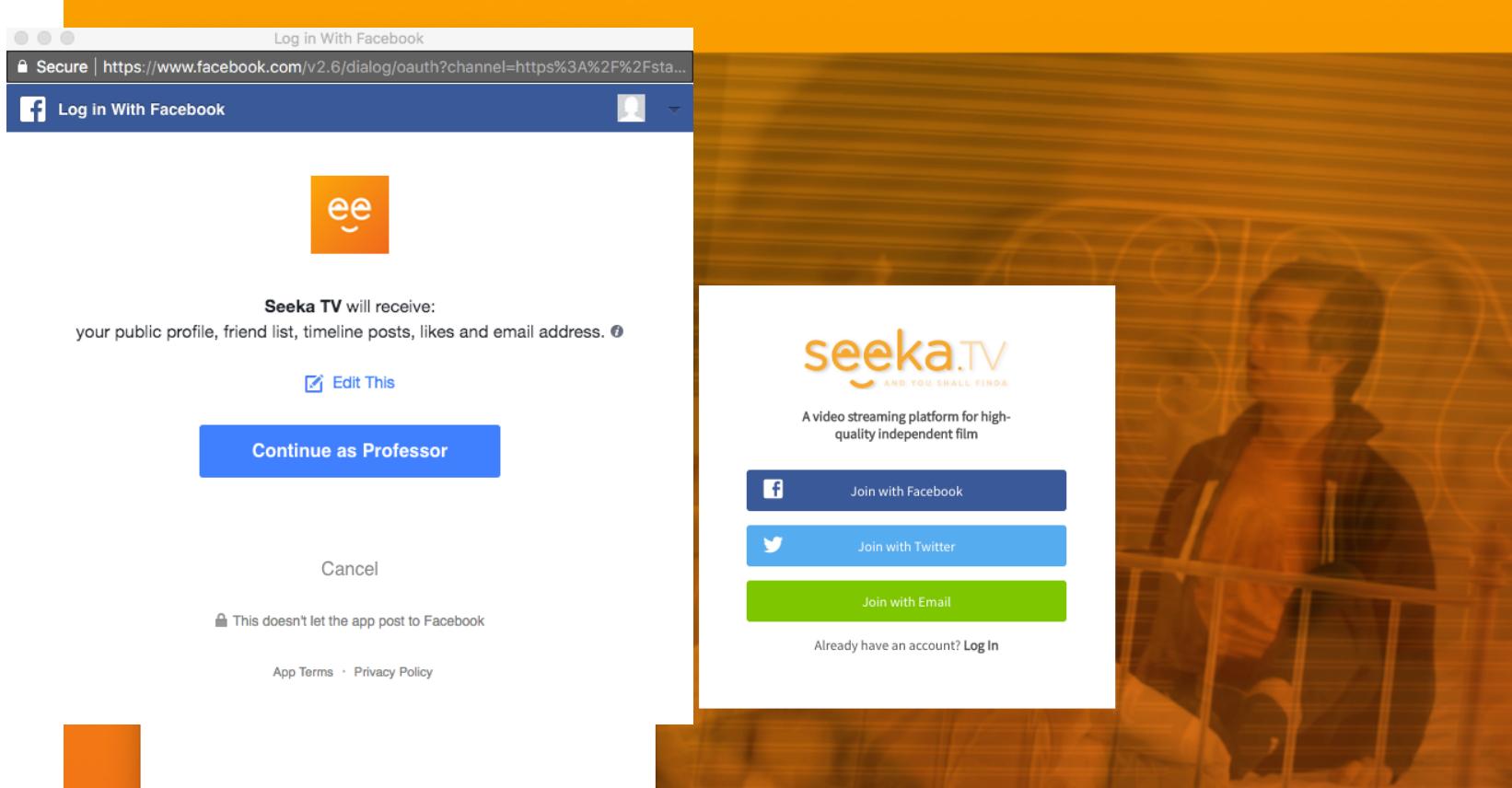
Registration



Registration



Registration



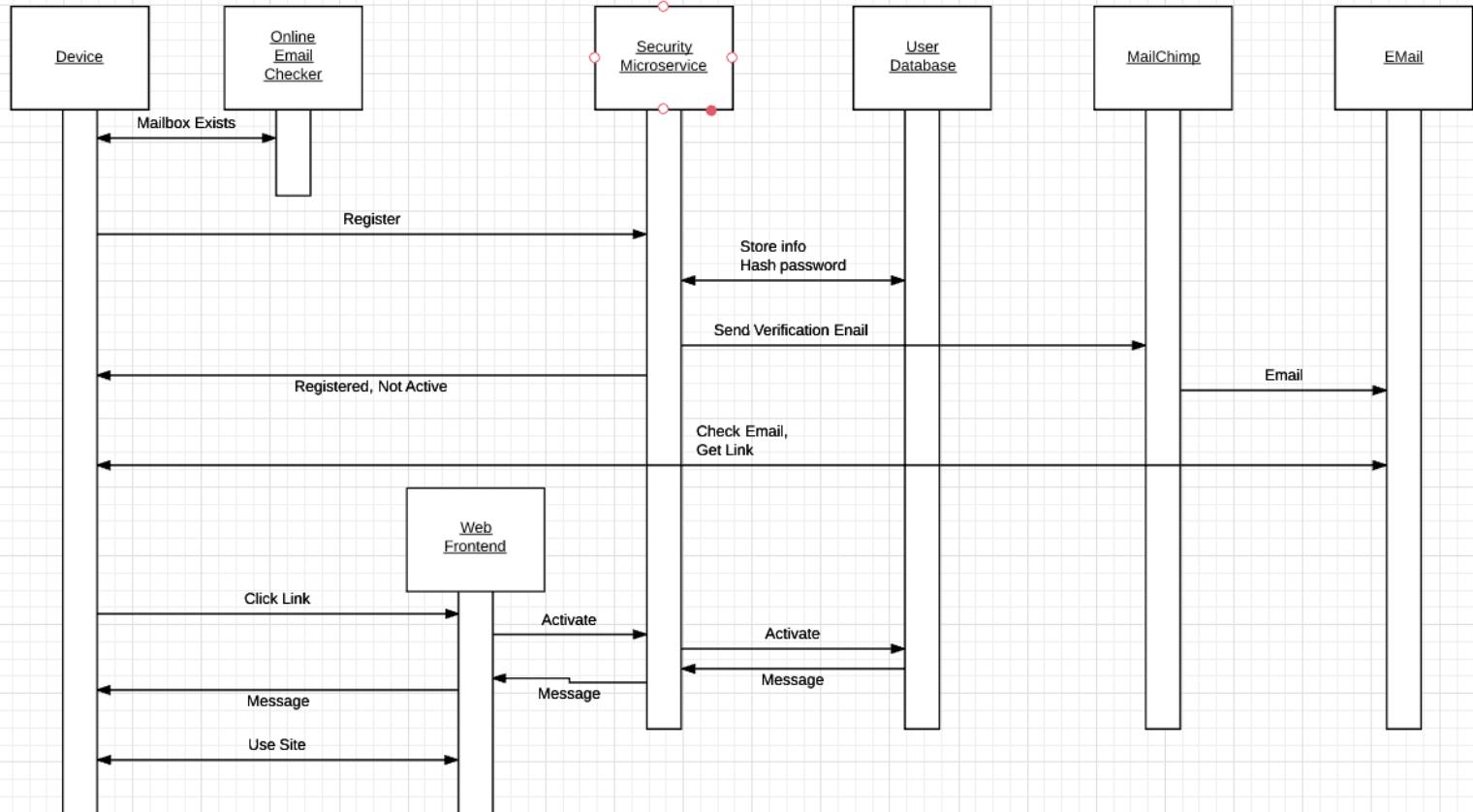
Pros and Cons of Social Media Logon

(<https://econsultancy.com/blog/61911-the-pros-and-cons-of-a-facebook-login-on-ecommerce-sites>)

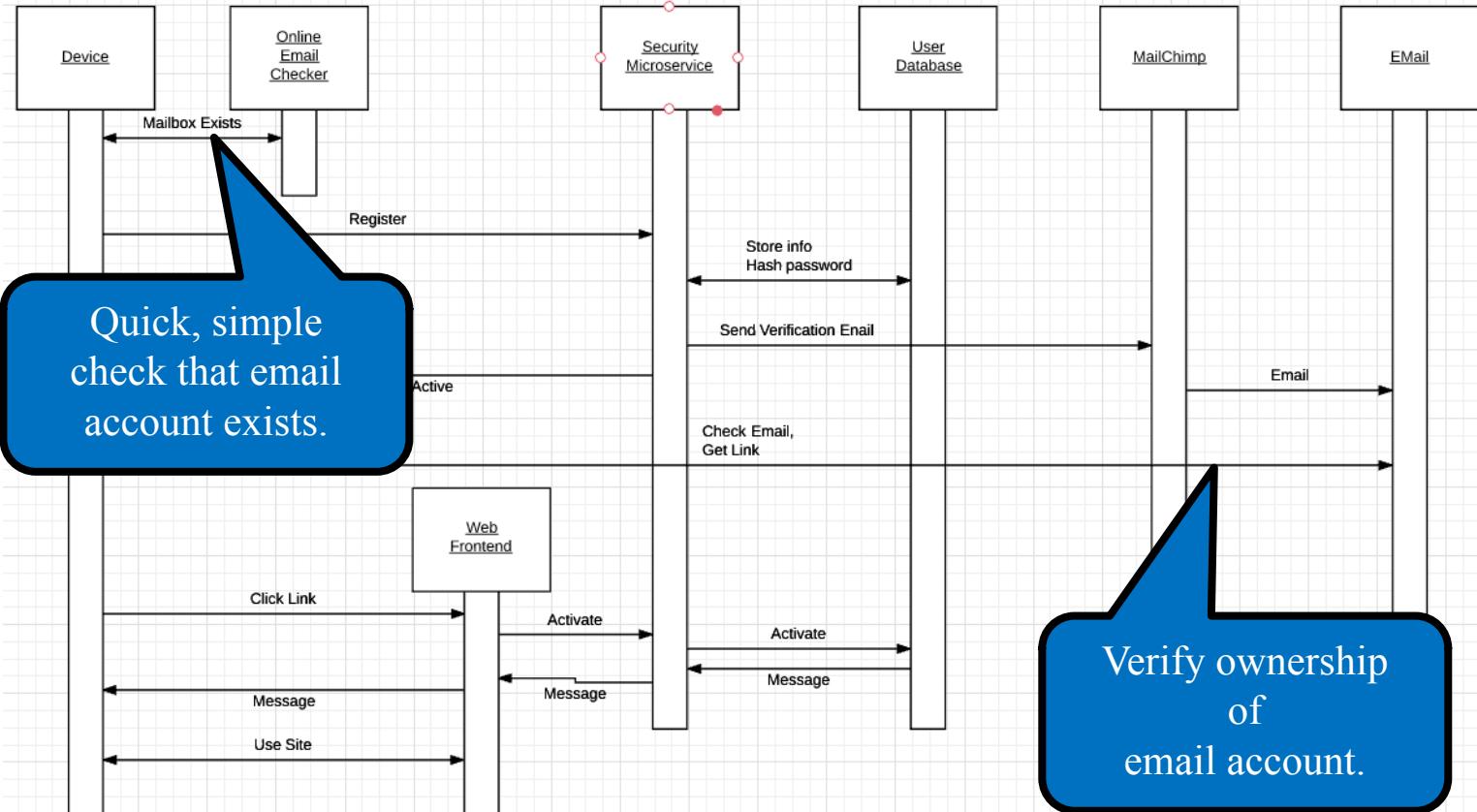
- Pros
 - One password for many services: Some people hate making up passwords
 - Familiarity: People trust Facebook and may not want to give you info.
 - Added convenience drives business: “Forced registration is a major cause of checkout abandonment, with [a quarter \(26%\) of respondents in a recent Econsultancy survey](#) stating that being forced to register would cause them to abandon a purchase.”
 - Sharing: Simplifies customers sharing their use of your site → drives awareness.
 - Access to profile data: Do not require users to enter a new profile.
 - Your site does not have to have code for password reset, locked account, etc.
- Cons
 - Some users don’t want everything to be connected.
 - Some people don’t use Facebook.
 - Muddying your brand image.

Email

Email Registration

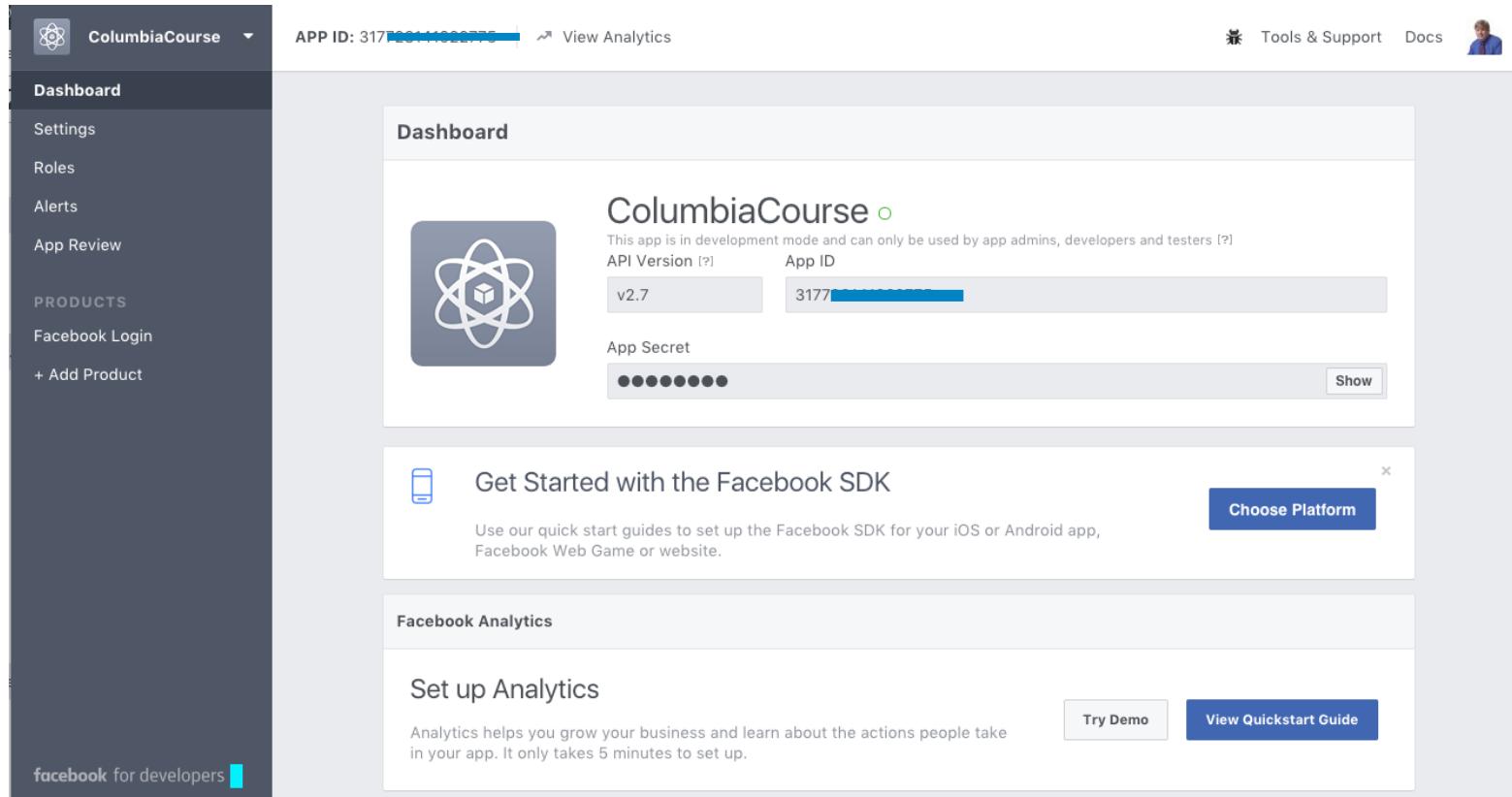


Email Registration



Social Media Registration (Facebook)

Define Application



The screenshot shows the Facebook App Dashboard for the application "ColumbiaCourse". The left sidebar contains navigation links: Dashboard, Settings, Roles, Alerts, App Review, PRODUCTS (Facebook Login, + Add Product), and a "facebook for developers" link. The main content area displays the app's details: App ID (3177), API Version (v2.7), App ID (3177), and App Secret (redacted). A "Get Started with the Facebook SDK" callout is present, along with "Facebook Analytics" and "Set up Analytics" sections.

APP ID: 3177 [View Analytics](#)

Tools & Support Docs 

Dashboard

ColumbiaCourse 

This app is in development mode and can only be used by app admins, developers and testers [?]

API Version [\[?\]](#) App ID

v2.7 3177

App Secret  [Show](#)

Get Started with the Facebook SDK  [Choose Platform](#)

Use our quick start guides to set up the Facebook SDK for your iOS or Android app, Facebook Web Game or website.

Facebook Analytics

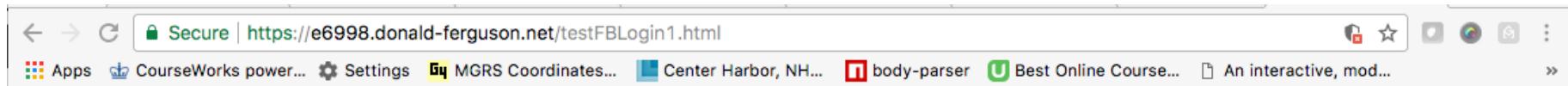
Set up Analytics

Analytics helps you grow your business and learn about the actions people take in your app. It only takes 5 minutes to set up.

[Try Demo](#) [View Quickstart Guide](#)

facebook for developers

Web Page



A screenshot of a web browser window. The address bar shows a secure connection to <https://e6998.donald-ferguson.net/testFBLogin1.html>. The page content is visible below the browser's header.

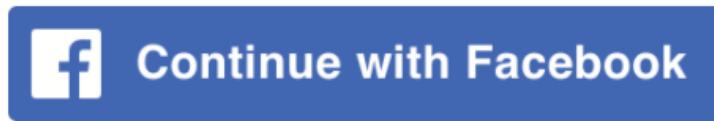
Secure | https://e6998.donald-ferguson.net/testFBLogin1.html

Apps CourseWorks power... Settings MGRS Coordinates... Center Harbor, NH... body-parser Best Online Course... An interactive, mod...

COMSE6998-014: Microservice and Cloud Applications

This is without a doubt the totally coolest course on the Columbia syllabus, taught by an adjunct professor who is too cool for school. In this lecture you learn:

1. How to login to an application with Facebook.
2. General concepts around OAuth2
3. How this all fits into a larger security model.



Log In with Facebook

Full Name: {{name}}

email: {{email}}

security token: {{secret}}

Code (I)

```
<div ng-app="apiController" ng-controller="myCtrl">
  <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
  <script src="/apiController.js"></script>
  <script>

    // This function is called when someone finishes with the Login
    // Button. See the onlogin handler attached to it in the sample
    // code below.
    function checkLoginState() {
      FB.getLoginStatus(function(response) {
        statusChangeCallback(response);
      });
    }

    window.fbAsyncInit = function() {
      FB.init({
        appId      : '3177████████',
        cookie     : true,  // enable cookies to allow the server to access
                           // the session
        xfbml     : true, // parse social plugins on this page
        version   : 'v2.5' // use graph api version 2.5
      });

      // Now that we've initialized the JavaScript SDK, we call
      // FB.getLoginStatus(). This function gets the state of the
      // person visiting this page and can return one of three states to
      // the callback you provide. They can be:
      //
      // 1. Logged into your app ('connected')
      // 2. Logged into Facebook, but not your app ('not_authorized')
      // 3. Not logged into Facebook and can't tell if they are logged into
      //    your app or not.
      //
      // These three cases are handled in the callback function.

      FB.getLoginStatus(function(response) {
        statusChangeCallback(response);
      });
    };

    // Load the SDK asynchronously
    (function(d, s, id) {
      var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js";
      fjs.parentNode.insertBefore(js, fjs);
    })(document, 'script', 'facebook-jssdk'));
  </script>
```

Code (II)

```
<h1 style="text-align: center;">COMSE6998-014: Microservice and Cloud Applications</h1>
<p>
  This is without a doubt the totally coolest course on the Columbia syllabus, taught by an adjunct professor who is too cool for school. In this lecture you
  learn:
  <ol>
    <li>How to login to an application with Facebook.
    <li>General concepts around OAuth2
    <li>How this all fits into a larger security model.
  </ol>
</p>

<!--
<fb:login-button scope="public_profile,email" onlogin="checkLoginState();">
</fb:login-button>
-->
<div ng-click="login()">
  <br>
  <span>Log In with Facebook</span>
</div>

<div>
  <p>
    Full Name: {{name}}<br>
    email: {{email}}<br>
    security token: {{secret}}<br>
  </p>
</div>
</div>
</body>
</html>
```

Facebook Login

COMSE6998-014: Microservice and Cloud Applications

This is without a doubt the totally coolest course on the Columbia syllabus, taught by an adjunct professor who is too cool for school. In this lecture you learn:

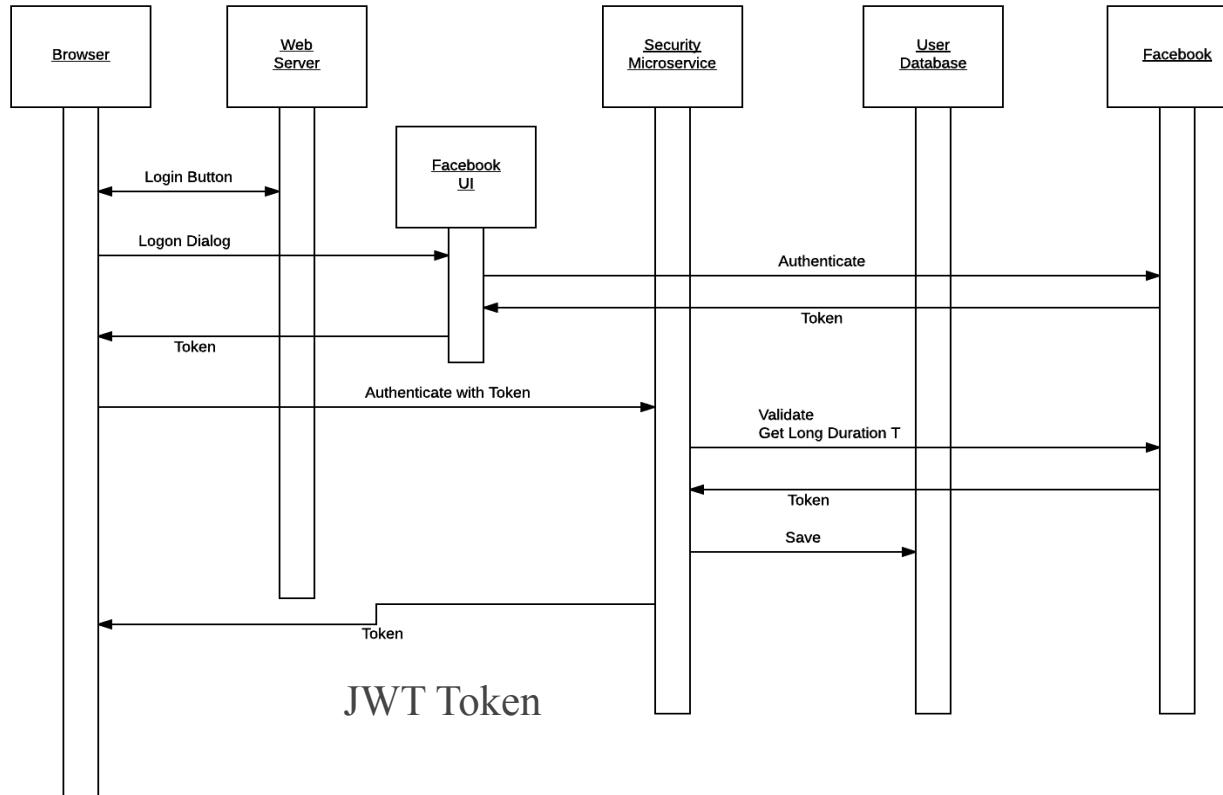
1. How to login to an application with Facebook.
2. General concepts around OAuth2
3. How this all fits into a larger security model.

 Continue with Facebook

Log In with Facebook

Full Name: Professor Ferguson
email: professor_rncgpb_ferguson@tfbnw.net
security token:
EAAEgZBNkI0dcBAC05drNhSJKfC6b1owCaqUmoW3MsC2NFfHnTE4DioHT1jR28kPTvZCh63W

Facebook Login



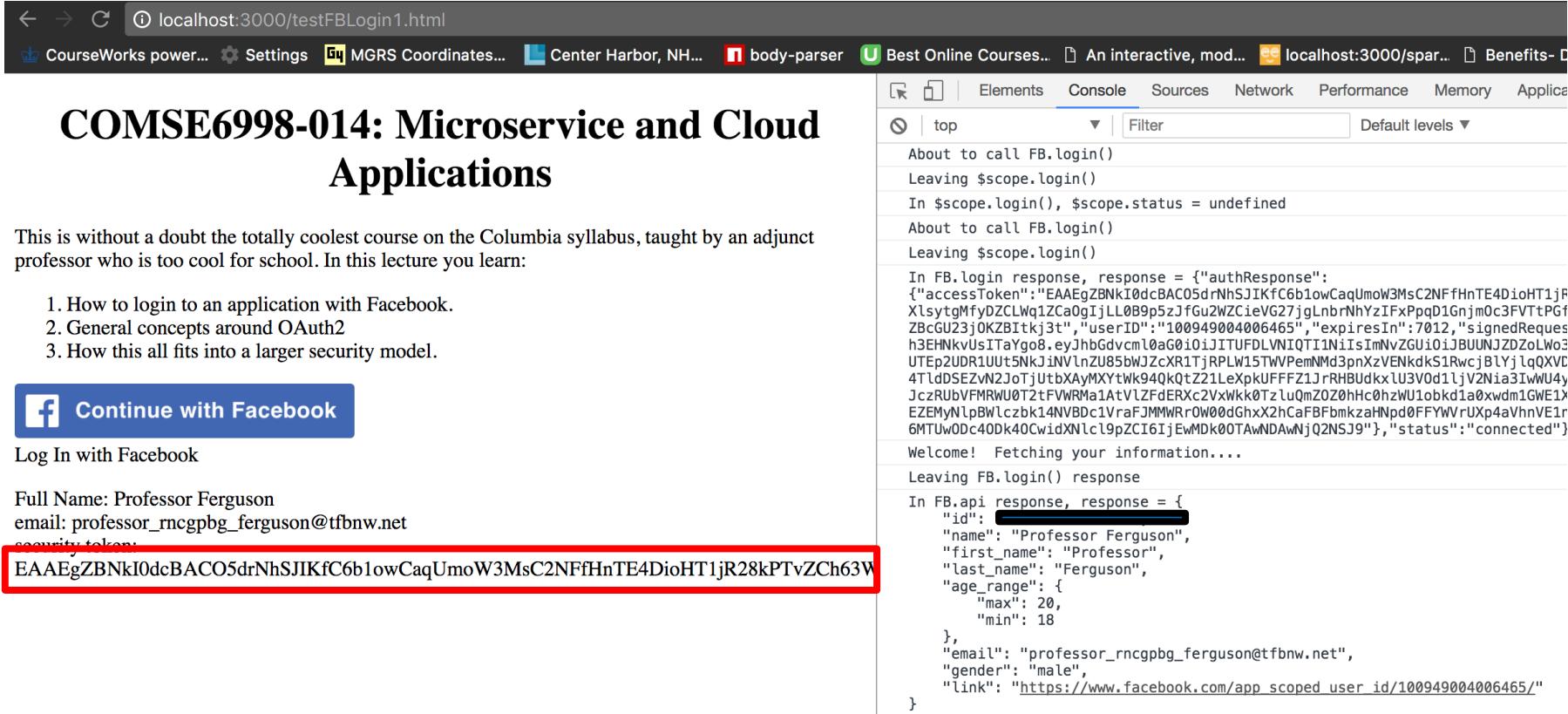
Validate Token

```
public class FBConnector {  
  
    private static final Version FACEBOOK_APP_VERSION = Version.VERSION_2_6;  
  
    private static final org.slf4j.Logger logger = LoggerFactory.getLogger(FBConnector.class);  
  
    public static Gson prettyGson = new GsonBuilder().setPrettyPrinting().create();  
  
    public static boolean validateUIDAndToken(String accessToken, String uid) {  
        boolean authenticated = false;  
        User theUser = null;  
  
        try {  
            theUser = FBConnector.fetchUserObjectAsJsonObject(accessToken, uid);  
  
            if (theUser != null) {  
                authenticated = true;  
            }  
        } catch (Exception e) {  
            System.out.println("Error = " + e.toString());  
            e.printStackTrace();  
        }  
  
        return authenticated;  
    }  
}
```

Validate Token

```
public static User fetchUserObjectAsJsonObject(String accessToken, String uid) {  
    User theUser = null;  
  
    try {  
        logger.debug("In FB Connector.");  
  
        DefaultFacebookClient facebookClient23 = new DefaultFacebookClient(accessToken, FACEBOOK_APP_VERSION);  
  
        logger.debug("Got Facebook client");  
        logger.debug("Calling FB for ID = " + uid);  
  
        // Make the API call  
        theUser = facebookClient23.fetchObject(uid, User.class);  
  
        // logger.debug("Back from FB call, result = " +  
        // prettyGson.toJson(theUser));  
        logger.debug("Last name = " + theUser.getLastName());  
  
        // Based on the FB App ID and access token, we sometimes get the  
        // last_name and first_name,  
        // and sometimes just the full name, e.g. "Hermione Grainger."  
        if ((theUser.getLastName() == null) || (theUser.getFirstName() == null)) {  
            theUser = updateNames(theUser);  
        }  
    } catch (Exception e) {  
        System.out.println("Error = " + e.toString());  
        e.printStackTrace();  
        theUser = null;  
    }  
  
    return theUser;  
}
```

Facebook Token



The screenshot shows a web browser window with the URL `localhost:3000/testFBLogin1.html`. The page content is as follows:

COMSE6998-014: Microservice and Cloud Applications

This is without a doubt the totally coolest course on the Columbia syllabus, taught by an adjunct professor who is too cool for school. In this lecture you learn:

1. How to login to an application with Facebook.
2. General concepts around OAuth2
3. How this all fits into a larger security model.

Below this, there is a button labeled "Continue with Facebook" with a Facebook icon. To its left, the text "Log In with Facebook" is visible. Further down, there is a "Full Name: Professor Ferguson" field, an "email: professor_rncgpb_ferguson@tfbnw.net" field, and a "security token:" field containing the value `EAAEgZBNkI0dcBAC05drNhSJKfC6b1owCaqUmoW3MsC2NFfHnTE4DioHT1jR28kPTvZCh63W`, which is highlighted with a red box.

The browser's developer tools are open, specifically the "Console" tab. The console output shows the following log entries:

```
About to call FB.login()
Leaving $scope.login()
In $scope.login(), $scope.status = undefined
About to call FB.login()
Leaving $scope.login()
In FB.login response, response = {"authResponse": {"accessToken": "EAAEgZBNkI0dcBAC05drNhSJKfC6b1owCaqUmoW3MsC2NFfHnTE4DioHT1jR28kPTvZCh63W", "user": {"id": "100000000000000", "name": "Professor Ferguson", "first_name": "Professor", "last_name": "Ferguson", "age": 30, "email": "professor_rncgpb_ferguson@tfbnw.net", "gender": "male", "link": "https://www.facebook.com/app_scoped_user_id/100000000000000/"}}, "status": "connected"}
Welcome! Fetching your information...
Leaving FB.login() response
In FB.api response, response = {
  "id": "100000000000000",
  "name": "Professor Ferguson",
  "first_name": "Professor",
  "last_name": "Ferguson",
  "age": 30,
  "email": "professor_rncgpb_ferguson@tfbnw.net",
  "gender": "male",
  "link": "https://www.facebook.com/app_scoped_user_id/100000000000000/"}}
```

Graph API Explorer Demo

facebook for developers

Products Docs Tools & Support News Videos

Search

Create App

Graph API Explorer

Application: [?] Graph API Explorer ▾

Access Token: [Get Token ▾](#)

 [GET](#) → /v2.10 /me?fields=birthday,devices,name  [Submit](#)

Learn more about the Graph API syntax

Node: me

+ Search for a field

1 Debug Message (Show)

```
{  
  "name": "Professor Ferguson",  
  "id": [REDACTED]  
}
```

OAuth

Twitter and Others

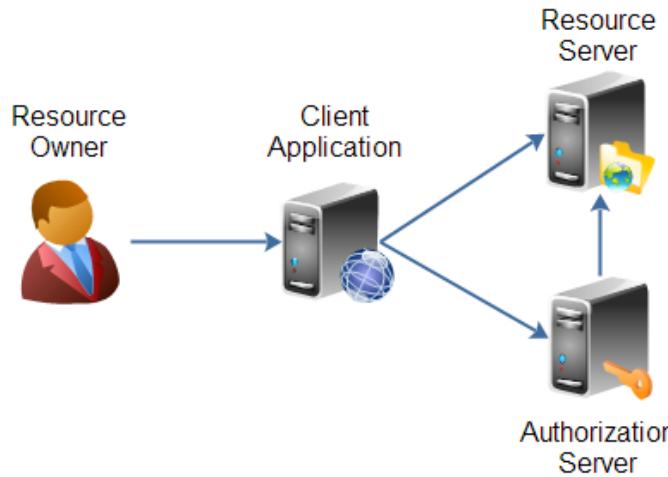
Overview (<http://tutorials.jenkov.com/oauth2/index.html>)

- Resource Owner
 - Controls *access* to the “data.”
 - Facebook **user**, LinkedIn **user**, ...
- Resource Server
 - The website that holds/manages info.
 - Facebook, LinkedIn, ...
 - And provides access API.
- Client Application
 - “The product you implemented.”
 - Wants to read/update
 - “On your behalf”
 - The data the data that the Resource Server maintains, e.g. posts, status, tweets, ...
- Authorization Server
 - Grants/rejects authorization
 - Based on Resource Owner decisions.
 - Usually (logically) the same as Resource Server.

OAuth 2.0 defines the following roles of users and applications:

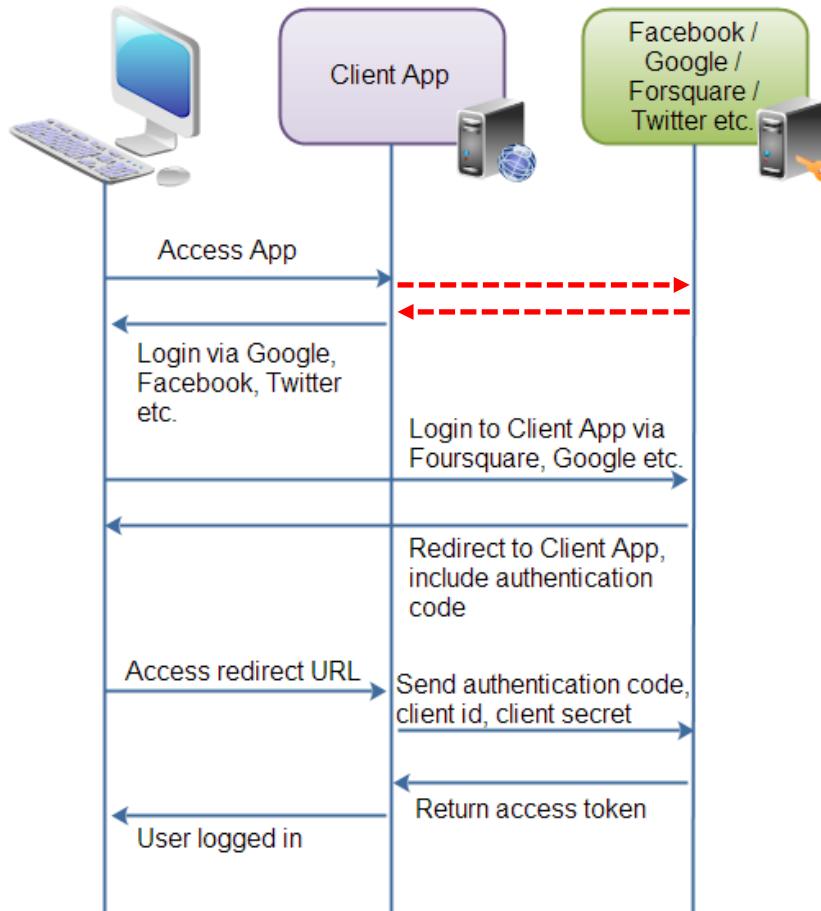
- Resource Owner
- Resource Server
- Client Application
- Authorization Server

These roles are illustrated in this diagram:



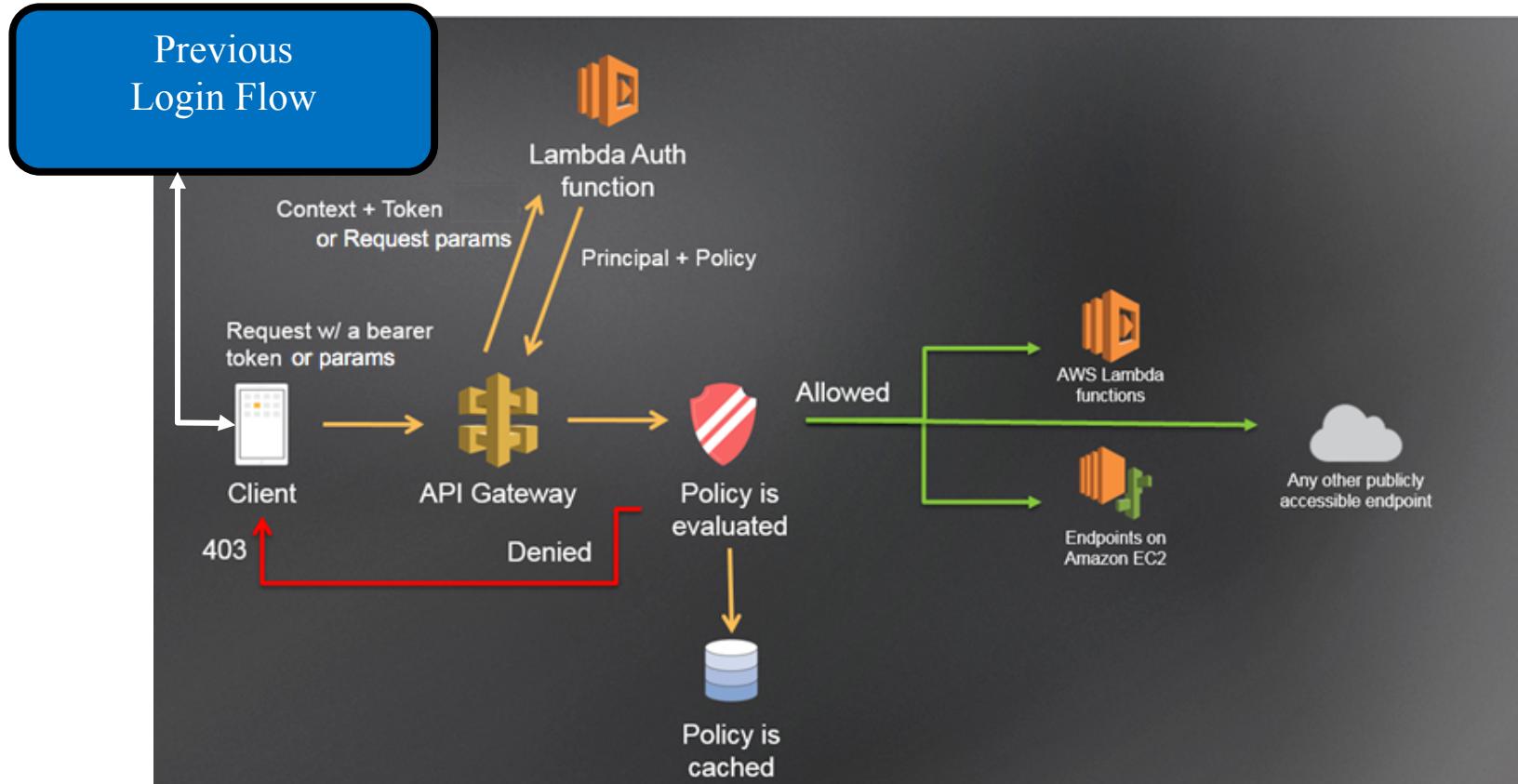
Roles/Flows

- User Clicks “Logon with XXX”
 - Redirect user to XXX
 - With Client App application ID.
 - And permissions app requests.
 - **Code MAY have to call Resource Server to get a token to include in redirect URL.**
- Browser redirected to XXX
 - Logon on to XXX prompt.
 - Followed by “do you grant permission to ...”
 - Clicking button drives a redirect back to Client App. URL contains a temporary token.
- User/Browser
 - Redirected to Client App URL with token.
 - Client App calls XXX API
 - Obtains access token.
 - Returns to User.
- Client App can use access token on API calls.



Authorization

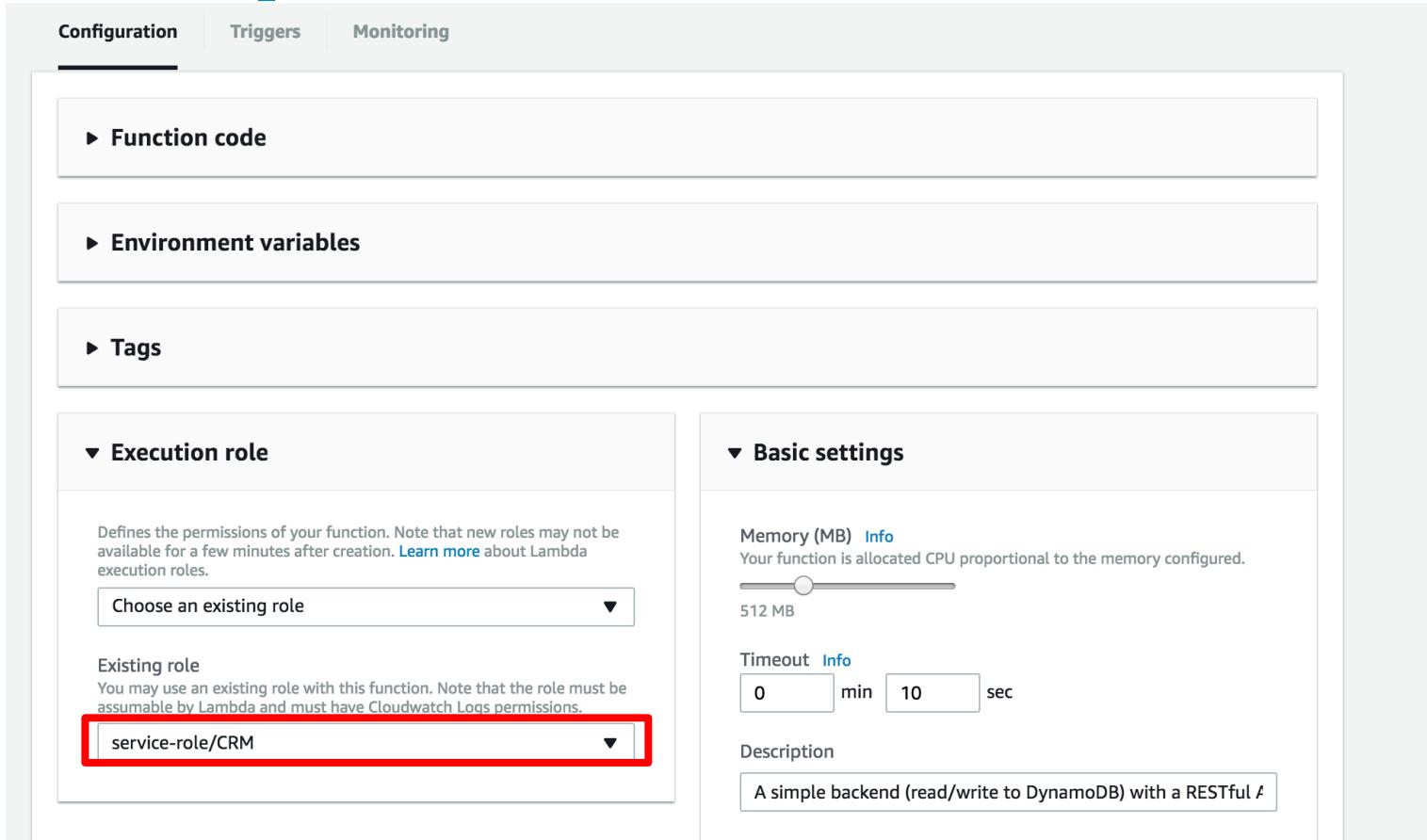
Custom Authorizer



Authorization

- 1st check occurs at perimeter in API GW Authorization
 - Based on (bearer token) JWT token
 - Is this role authorized to invoke this URL?
 - Implementation
 - Lambda function that ...
 - Checks an authorization table (role, URL)
 - Or the rights are encoded in the token.
- All down stream microservices
 - Receive the JWT token and perform their own authorization decisions.
 - Run with a set of AWS roles and permissions that determine if AWS allows access, which may include user role.

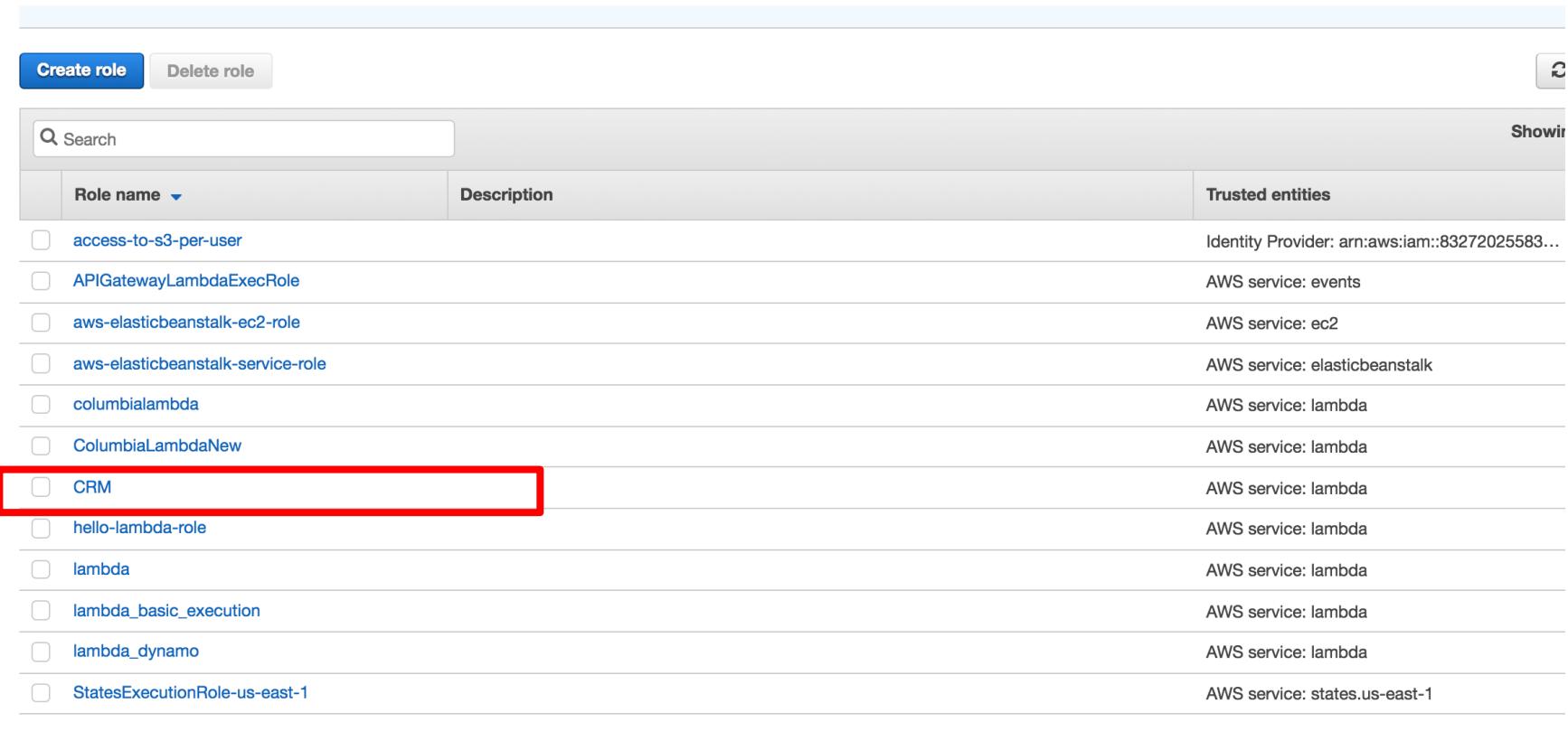
An Example – Lambda Function Role



The screenshot shows the AWS Lambda Function Configuration page. The top navigation bar includes tabs for Configuration (which is selected), Triggers, and Monitoring. The main content area is divided into several sections:

- Function code**
- Environment variables**
- Tags**
- Execution role** (with a description and a dropdown menu)
 - Choose an existing role
 - Existing role (with a dropdown menu showing "service-role/CRM", which is highlighted with a red box)
- Basic settings** (with sliders for Memory and Timeout)
 - Memory (MB) Info: A slider set to 512 MB.
 - Timeout Info: Set to 0 min 10 sec.
 - Description: A simple backend (read/write to DynamoDB) with a RESTful API.

Role/Policies



The screenshot shows the AWS IAM Roles list page. At the top, there are buttons for 'Create role' and 'Delete role'. Below that is a search bar with the placeholder 'Search'. On the right, there is a 'Showing' label and a refresh icon. The main table has three columns: 'Role name', 'Description', and 'Trusted entities'. The 'Role name' column contains a dropdown arrow. The 'Description' and 'Trusted entities' columns are aligned to the right. The 'CRM' role is highlighted with a red box. The table lists the following roles:

Role name	Description	Trusted entities
<input type="checkbox"/> access-to-s3-per-user		Identity Provider: arn:aws:iam::83272025583...
<input type="checkbox"/> APIGatewayLambdaExecRole		AWS service: events
<input type="checkbox"/> aws-elasticbeanstalk-ec2-role		AWS service: ec2
<input type="checkbox"/> aws-elasticbeanstalk-service-role		AWS service: elasticbeanstalk
<input type="checkbox"/> columbialambda		AWS service: lambda
<input type="checkbox"/> ColumbiaLambdaNew		AWS service: lambda
<input type="checkbox"/> CRM		AWS service: lambda
<input type="checkbox"/> hello-lambda-role		AWS service: lambda
<input type="checkbox"/> lambda		AWS service: lambda
<input type="checkbox"/> lambda_basic_execution		AWS service: lambda
<input type="checkbox"/> lambda_dynamo		AWS service: lambda
<input type="checkbox"/> StatesExecutionRole-us-east-1		AWS service: states.us-east-1

Role/Policy

Role ARN: arn:aws:iam::00070005820:role/service-role/CRM

Role description: [Edit](#)

Instance Profile ARNs:

Path: /service-role/

Creation time: 2017-09-26 13:09 EDT

Permissions [Trust relationships](#) [Access Advisor](#) [Revoke sessions](#)

Attach policy Attached policies: 2

Policy name	Policy type	Actions
AWSLambdaBasicExecutionRole-6ac3e99f-01ee-47fa-8422-d9294b9802cb	Managed policy	X
AWSLambdaMicroserviceExecutionRole-b5736af1-9b96-4112-bf75-8f2c7dcf460c	Managed policy	X

[Policy summary](#) [{ } JSON](#) [Edit policy](#) [Simulate policy](#)

Filter

Service	Access level	Resource	Request condition
DynamoDB	Limited: Read, Write	TableName = All	None

[Allow \(1 of 108 services\)](#) [Show remaining 108](#)

[+ Add inline policy](#)