
# Crash Course: Advanced Topics in Apigee Edge

# The Team

apigee


# Deep Dive 3-legged OAuth 2.0

Alex Koo – Apigee Principal Architect
Diego Zuluaga – Apigee Principal Architect

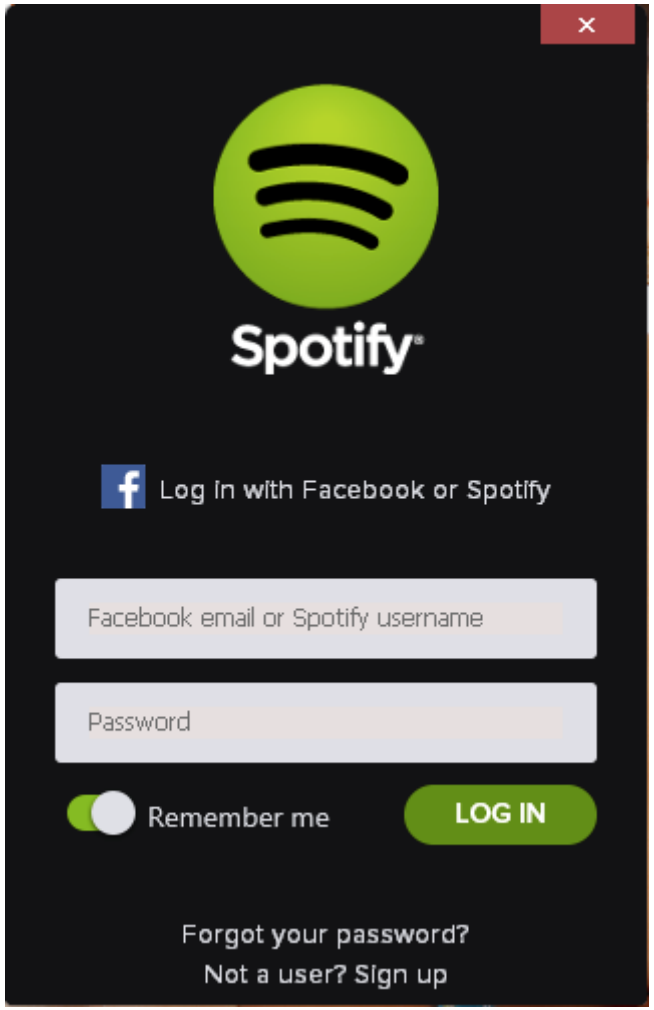
# So, what's the use case for it?

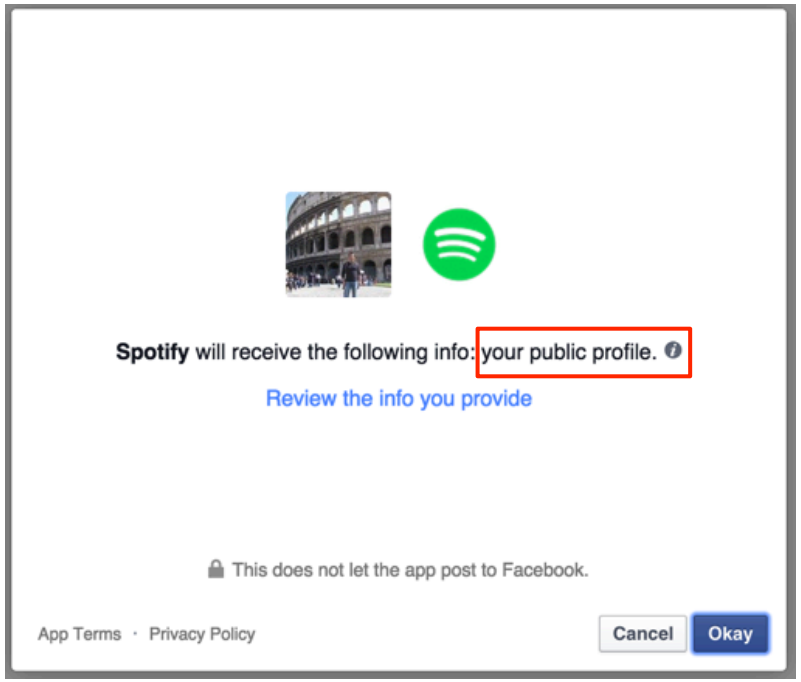

# do I want to give access to these these *resources* to someone else?

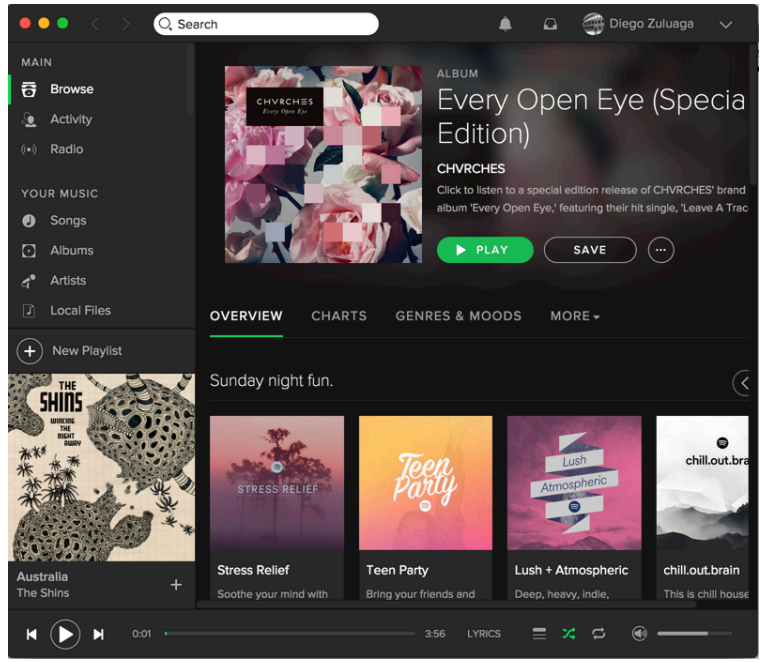


apigee

# Essentially: How to authorize external applications to access your *resources*




apigee

# OAuth Basic Concepts

- **OAuth 2.0** is a protocol that allows clients to grant access to server resources to another entity without sharing credentials
- Client IDs and Secrets are used to identify and authenticate applications (application's consumer key and consumer secret)
- Tokens are issued to allow access to specific resources for a specified period of time and may be revoked by the user that granted permission or by the server that issued the token

# OAuth Basic Concepts

- We can use scopes to limit the access for a given token, granting permission only for the operations that are necessary
- Five different grant types specify the different authentication usage scenarios OAuth supports
- We must protect tokens, and OAuth 2.0 requires that all API traffic be sent via SSL

# Access Tokens

**Access Tokens** allow access to a protected resource for a specific application to perform only certain actions for a limited period of time.

Identification info from the requesting application **(client ID and secret)**

\+

Resource owner credentials (if needed)

\+

Optional information about what the application wants to do with the resource **(scope)**


=

Access Token and (optional) refresh token

In Apigee, access tokens are opaque strings with no encoded meaning. Access tokens are passed as bearer tokens in an Authorization header.

# Refresh Tokens

**Refresh Tokens**, if provided, represent a limited right to reauthorize the granted access by obtaining new access tokens.

Identification info from the requesting application **(client ID and secret)**


+

Refresh token

+

Optional information about what the application wants to do with the resource **(scope)**


=

Access Token

# Scopes

**Scopes** identify what an application can do with the resources to which it is requesting access. Scope names are defined by the authorization server and are associated with information that enables decisions on whether a given API request is allowed or not.

Apigee associates scope names to be matched with a combination of API resource path and verb. So, for example:

**Scope 1: "READ"**

- GET /photos
- GET /photos/{id}

**Scope 2: "UPDATE"**

- GET /photos
- GET /photos/{id}
- POST /photos
- PUT /photos/{id}

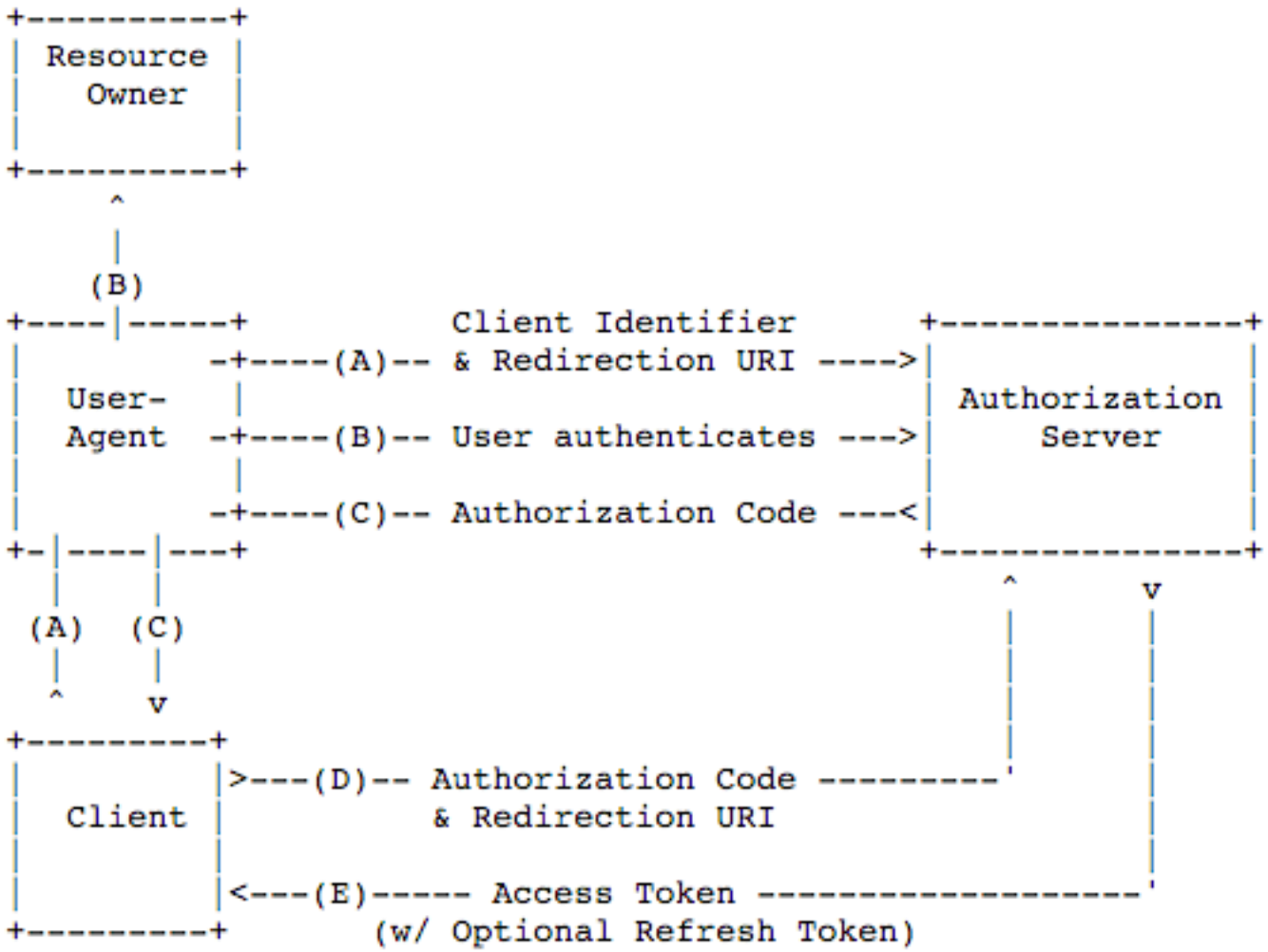
When an application requests an access token, the scope names are optional.

# OAuth 2.0 Grant Types

An **OAuth Grant** is a credential representing the resource owner's authorization. More often than not, we tend to think of grants in terms of the process used to obtain an access token.

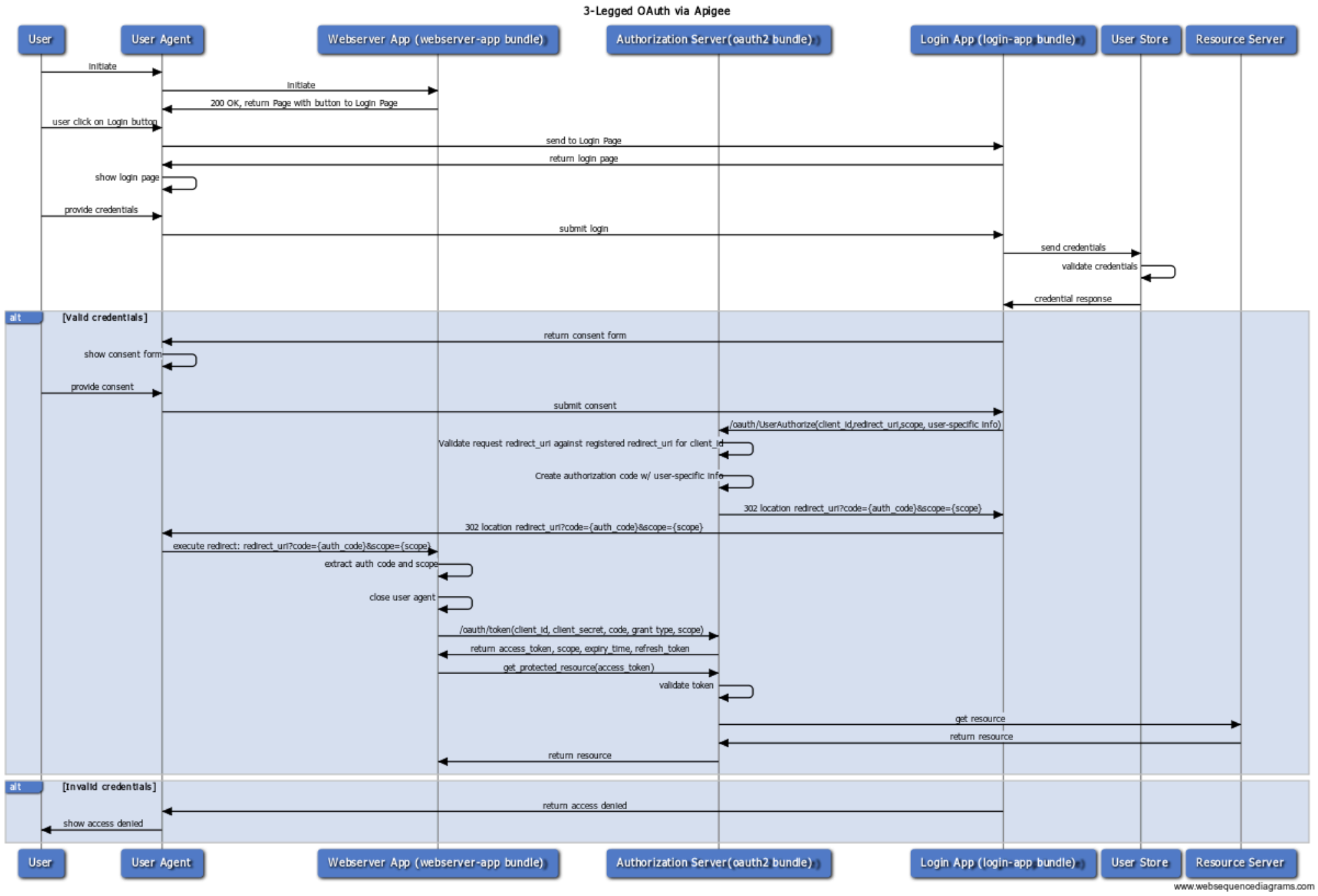
| Grant Type | Typical Use Case | Complex? |
|---|---|---|
| **No specific resource owner is involved** | | |
| Client Credentials | Business system interactions, where resources being operated on are owned by the partner, not a particular user | No |
| **A specific resource owner is involved** | | |
| Resource Owner Password Credentials | Resources are owned by a particular user and the requesting application is **trusted** | A bit |
| **Authorization Code** | **Resources are owned by a particular user and the requesting application is untrusted** | **Very** |
| Implicit | Resources are owned by a particular user, and the requesting application is an **untrusted browser-based app written in a scripting language such as JavaScript** | Very, and potentially insecure as well |
| Refresh | For generating a new access token. Refresh tokens have longer TTLs than access tokens. | No |

# OAuth 2.0 Auth Code Grant Type





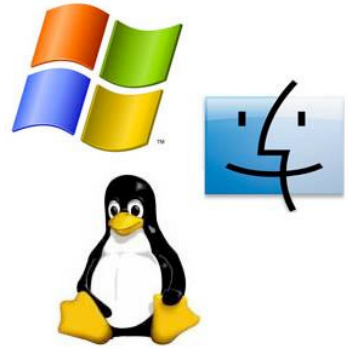

 





# Before we get started…



1. Open an Account in Cloud9 - it's free…

   https://c9.io

2. Start by cloning this Apigee Samples Repo

   ```
   git clone https://github.com/apigee/api-platform-samples.git
   ```

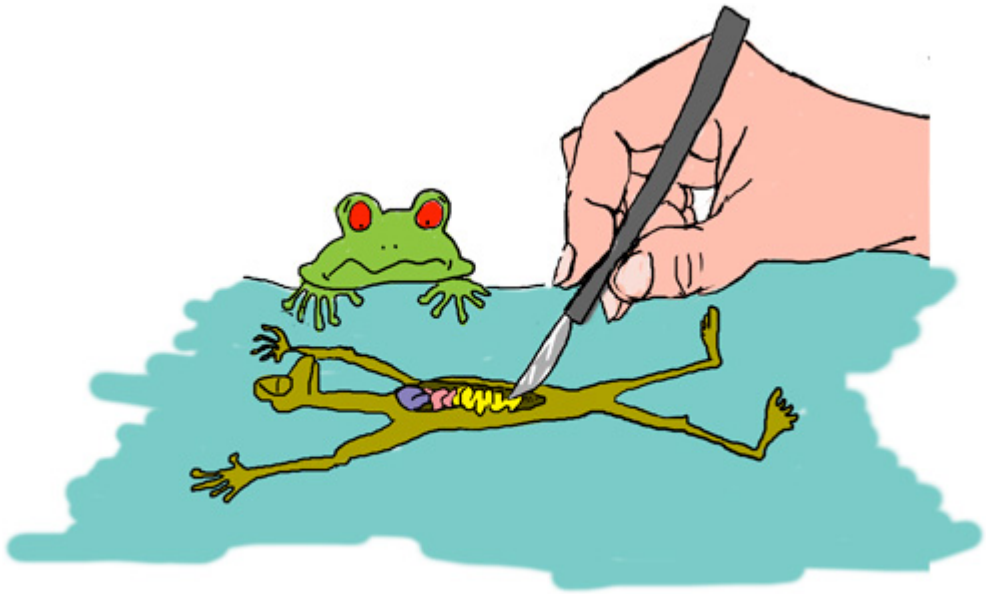
   

   – https://github.com/apigee/api-platform-samples
     - Login App
     - Third-party App
     - OAuth 2.0 API
     - User Authentication/Management Endpoint

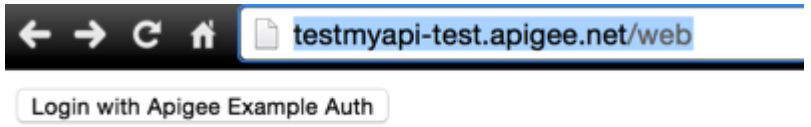
3. Install apigeetool

   ```
   npm install apigeetool -g
   ```

# Let’s dissect our API Proxy Bundles



# webserver-app bundle



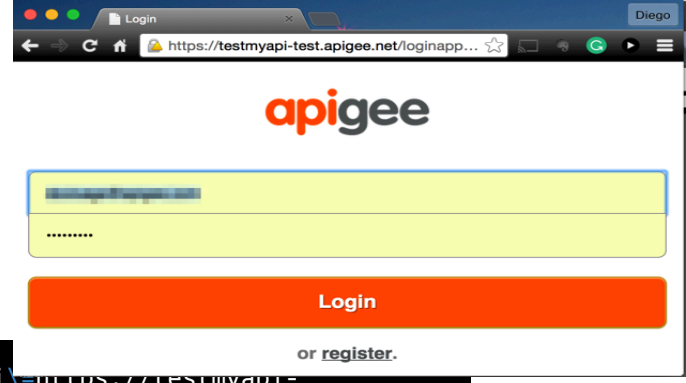

- Represents the third-party web app
- Link or button to the login page

```
$ curl http://testmyapi-test.apigee.net/web -v
> GET /web HTTP/1.1
> Host: testmyapi-test.apigee.net
< HTTP/1.1 200 OK
      <!DOCTYPE html>
<html>
   <head>
      <script>
         var BASEURL="https://testmyapi-test.apigee.net";
         var REDIRECT="https://testmyapi-test.apigee.net/web/callback";
         var CLIENT_ID="VXNYaci4FGfKfEERy5KhXHeIln2pONDr";


         function login()
         {
         window.location.href=BASEURL+'/loginapp/login?apikey='+CLIENT_ID+'&redirect_uri='+REDIRECT
+'&scope=order&state=123';
         }
      </script>
   </head>
   <body>
      <input type="button" value="Login with Apigee Example Auth" onclick="login()" />
   </body>
</html>
```

# login-app bundle

- Represents the login-app - login page:

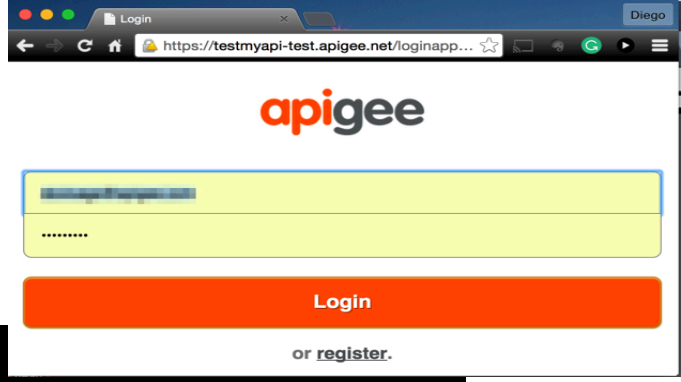



```
curl https://testmyapi-test.apigee.net/loginapp/login\?apikey\=VXNYaci4FGfKfEERy5KhXHeIln2pONDr\&redirect_uri\=https://testmyapi-
test.apigee.net/web/callback\&scope\=order\&state\=123 -v
> GET
< HTTP/1.1 200 OK
< set-cookie: sid=s%3AiHGIrOYTOfGwncJNV03Typkeb6rYAB6V.8GGzrvr4JTHZV6l%2FUo2oKqBgCHuNGbrvE8uulbXvjW8; Path=/; Expires=Mon, 05 Oct 2015
01:45:46 GMT; HttpOnly
<!DOCTYPE html>
<html>
    <head>
        <title>Login</title>
        <link rel="stylesheet" type="text/css" href="/loginapp/stylesheets/global.css" >
        <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">
    </head>
    <body>
        <form id="login" name="login" method="post">
            <h1><img src="/loginapp/images/apigee_logo_md.png" alt="Apigee" /></h1>


            <label for="username" class="noshow">Username</label>
            <input id="username" name="username" type="text" placeholder="Email address" required />
            <label for="password" class="noshow">Password</label>
            <input id="password" name="password" type="password" placeholder="Password" required />
            <input name="submit" type="submit" value="Login" />
            <p class="intro">or <a href="/loginapp/register?
apikey=VXNYaci4FGfKfEERy5KhXHeIln2pONDr&amp;state=123&amp;scope=order&amp;redirect_uri=https%3A%2F%2Ftestmyapi-test.apigee.net%2Fweb
%2Fcallback">register</a>.</p>
        </form>
    </body>
* Connection #0 to host testmyapi-test.apigee.net left intact
</html>
```

# login-app bundle

- Represents the login-app - submit credentials:




```
$ curl 'https://testmyapi-test.apigee.net/loginapp/login?
apikey=VXNYaci4FGfKfEERy5KhXHeIln2pONDr&state=123&scope=order&redirect_uri=https%3A%2F%2Ftestmyapi-test.apigee.net%2Fweb
%2Fcallback&app=oauth2-app' -H 'Cookie: __lc.visitor_id.3296802=S1436475468.3abaac467f; sid=s%3ARL8HY7b7IqporrtwlLUi8-
E5uX4YkAY4.yxUe2oPoukTxjwhoHdhz%2B8k9A9ghfsu7B%2Ft2rWuF8Og' -H 'Content-Type: application/x-www-form-urlencoded' --data
'username=dzuluaga%40apigee.com&password=apigee123&submit=Login' --compressed -v
> POST /loginapp/login?apikey=VXNYaci4FGfKfEERy5KhXHeIln2pONDr&state=123&scope=order&redirect_uri=https%3A%2F%2Ftestmyapi-
test.apigee.net%2Fweb%2Fcallback&app=oauth2-app HTTP/1.1
> Host: testmyapi-test.apigee.net
> User-Agent: curl/7.42.1
> Accept: */*
> Accept-Encoding: deflate, gzip
> Cookie: __lc.visitor_id.3296802=S1436475468.3abaac467f; sid=s%3ARL8HY7b7IqporrtwlLUi8-E5uX4YkAY4.yxUe2oPoukTxjwhoHdhz
%2B8k9A9ghfsu7B%2Ft2rWuF8Og
> Content-Type: application/x-www-form-urlencoded
> Content-Length: 62
>
* upload completely sent off: 62 out of 62 bytes
< HTTP/1.1 302 Found
< X-Powered-By: Express
< Location: /loginapp/consent?apikey=VXNYaci4FGfKfEERy5KhXHeIln2pONDr&app=oauth2-app&state=123&scope=order&redirect_uri=https%3A%2F
%2Ftestmyapi-test.apigee.net%2Fweb%2Fcallback
< set-cookie: sid=s%3AbigsdjFYyAfyuFg7Jk-HgcVkojwLzKI9.5B3q8Pq23EVv3ffNSX5yqok77XyV6ZCRgCfCdIWwbzc; Path=/; Expires=Mon, 05 Oct 2015
04:41:30 GMT; HttpOnly
```

# user-mgmt-v1 bundle - User Store

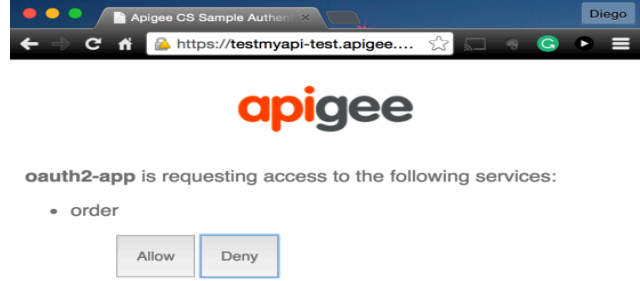

- Serves as the credential validation endpoint

```
$ curl https://testmyapi-test.apigee.net/v1/users/authenticate \
-X POST -d '{"username": "dzuluaga@apigee.com", "password": "apigee123" }' \
-H 'Content-Type:application/json' -v

< HTTP/1.1 403 Forbidden
< Content-Type: application/json
< Content-Length: 85
< Connection: keep-alive
<
* Connection #0 to host testmyapi-test.apigee.net left intact
{"status":"failure", "message":"Authentication failed for user
dzuluaga@apigee.com."}%
```

# login-app bundle

- Represents the login-app - consent page: “Allow” decision=yes



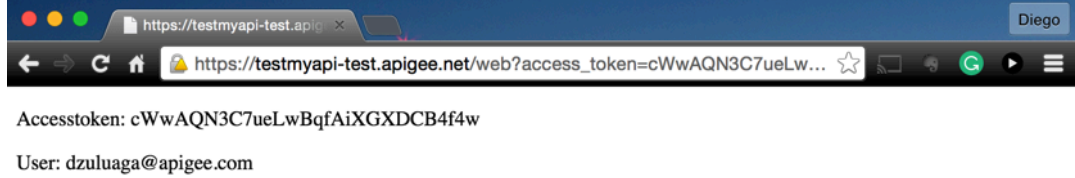

```
curl 'https://testmyapi-test.apigee.net/loginapp/consent?
apikey=VXNYaci4FGfKfEERy5KhXHeIln2pONDr&app=oauth2-app&state=123&scope=order&redirect_uri=https%3A
%2F%2Ftestmyapi-test.apigee.net%2Fweb%2Fcallback' -H 'Cookie: __lc.visitor_id.
3296802=S1436475468.3abaac467f; sid=s%3AMYwxTt148YagDN-htNbRv9UppUml9cYR.
9rL7bNV3p93TAamgLk3wVTVAnpOdvuzkLzhligHGnaw'-H 'Content-Type: application/x-www-form-urle
data 'decision=yes' --compressed -v

> Content-Type: application/x-www-form-urlencoded
> Content-Length: 12
>
* upload completely sent off: 12 out of 12 bytes
< HTTP/1.1 302 Found
< X-Powered-By: Express
< Location: https://testmyapi-test.apigee.net/web/callback?scope=&code=ylkMuj5l
< Date: Mon, 05 10 2015 04:53:08 GMT
< Content-Length: 0
< Connection: keep-alive
```

Get cookie from previous request

Authorization code

# webserverapp bundle




- Send authorization code to oauth2 bundle to obtain an access token
- Uses the secret

```
$ curl 'https://testmyapi-test.apigee.net/web/callback?scope=&code=LwhCoj7P'
-v
> GET /web/callback?scope=&code=LwhCoj7P HTTP/1.1
< HTTP/1.1 302 Redirect
< Location: https://testmyapi-test.apigee.net/web?
access_token=GOocdfQI40xNhZGUTn4uhIcwGYAS
< Content-Length: 0
< Connection: keep-alive
```

# Additional Challenges
# Q&A

# Use tokens from a Third Party Provider e.g. Google or Facebook

**Use case:**

- I want use tokens from Google or Facebook to access user resources
- I want leverage Apigee API Management capabilities. E.g. Traffic management, analytics, big data, etc.

**API proxy Sample**

- Apigee Tutorial
  http://apigee.com/docs/api-services/content/use-third-party-oauth-system
- Music Access - API Proxy Sample
  https://github.com/dzuluaga/apigee-tutorials/tree/master/apiproxies/musicapi-oauth-delegated-authentication
- Google OAuth 2.0 Playground
  https://developers.google.com/oauthplayground/

# How to Reuse Refresh Token?

```
<OAuthV2 enabled="true" continueOnError="false" async="false" name="AccessTokenRefresh">
    <Operation>RefreshAccessToken</Operation>
    <ReuseRefreshToken>true</ReuseRefreshToken>
    <GenerateResponse enabled="false"/>
</OAuthV2>
```