

OPEN EDX & OAUTH2

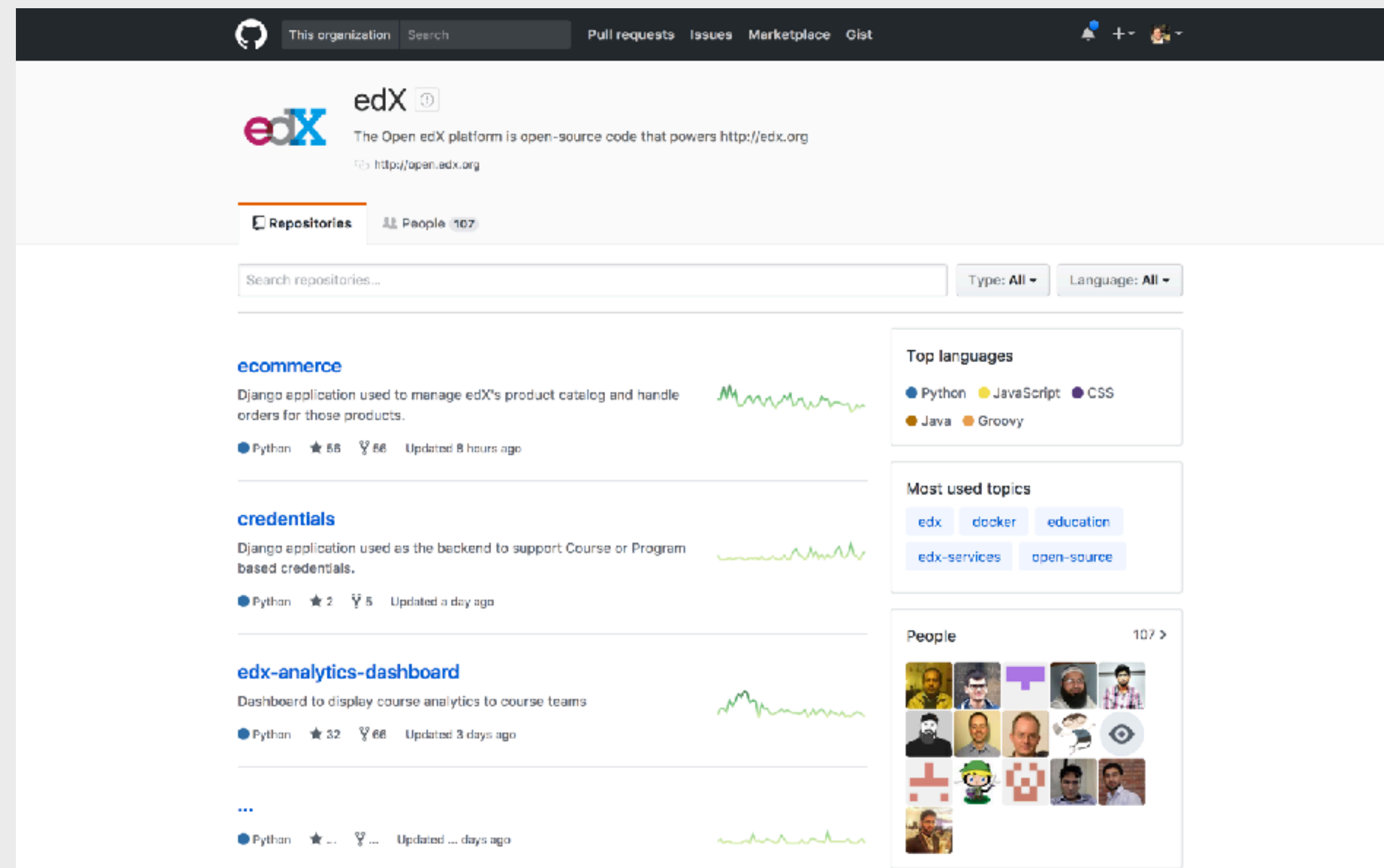
Scalable Extensions to the Platform

Miguel Amigot
CTO



WHY THIS IS INTERESTING

OPEN EDX IS INCREASINGLY MOVING TO MICROSERVICES



The screenshot displays the GitHub organization page for edX. At the top, the organization name 'edX' is shown with a description: 'The Open edX platform is open-source code that powers http://edx.org'. Below this, there are tabs for 'Repositories' and 'People (107)'. A search bar for repositories is present, along with filters for 'Type: All' and 'Language: All'. The main content area lists several repositories:

- ecommerce**: Django application used to manage edX's product catalog and handle orders for those products. It is written in Python, has 55 stars, 66 forks, and was updated 8 hours ago.
- credentials**: Django application used as the backend to support Course or Program based credentials. It is written in Python, has 2 stars, 5 forks, and was updated a day ago.
- edx-analytics-dashboard**: Dashboard to display course analytics to course teams. It is written in Python, has 32 stars, 66 forks, and was updated 3 days ago.

On the right side of the page, there are three summary boxes:

- Top languages**: Python, JavaScript, CSS, Java, Groovy.
- Most used topics**: edx, docker, education, edx-services, open-source.
- People**: A grid of 107 user avatars.

**CAN DEPLOY SEPARATE
WEBSITES AND SERVICES**

USE CASES

- 1** Insights and Ecommerce (already)
- 2** Customized admin dashboards
- 3** Instructor news feed?

INSIGHTS

The screenshot shows a web browser window with the URL 'edX Insights' and a user profile 'Miguel'. The page features the 'OPENedX INSIGHTS' logo and a 'LOGIN' button. A world map is displayed with several countries highlighted in blue, including the United States, Mexico, India, and China. Below the map, a dark banner contains the text 'Insights to help course teams improve courses.' The main content area is divided into three columns, each with a title and a description:

- Who are my students?**
Target your course to fit the backgrounds of your students.
↑ Enrollment
- What are students engaging with in my course?**
Improve frequently accessed content, or point students at underused resources.
📖 Engagement
- How well is my content supporting student learning?**
Adjust your course based on where students are struggling.
📊 Performance

**HOW DO WE HANDLE
USER ACCOUNTS?**

SINGLE SIGN-ON

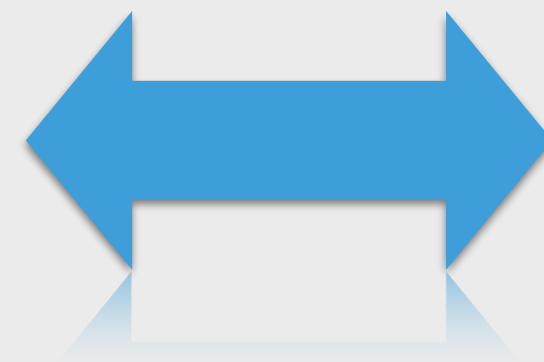
- 1 Use edx-platform's data
- 2 Referenced — but keep sessions
- 3 Single sign-on & single sign-out

HOW DOES THIS WORK?

OAUTH (MOSTLY)



Provider





Clients

OAUTH (SIMPLIFIED)

- 1 Register the client on edx/edx-platform (get an app client ID and a client secret)
- 2 Exchange these credentials on the client for access tokens and use these to get resources

/admin/oauth2/client/add/

Add client

User:	<input type="text"/> 
Name:	<input type="text"/>
Url:	<input type="text"/> <small>Your application's URL.</small>
Redirect uri:	<input type="text"/> <small>Your application's callback URL</small>
Client id:	<input type="text" value="594b2d279c8c48997e42"/>
Client secret:	<input type="text" value="576d85eb139536f39bceb44e3d355fc50"/>
Client type:	<input type="text" value="-----"/> 
Logout uri:	<input type="text"/> <small>Your application's logout URL</small>

[Save and add another](#)

[Save and continue editing](#)

[Save](#)

**BUT OAUTH DOESN'T
SAY WHO THE USER IS...**

**...SINCE THE ACCESS TOKEN
IS OPAQUE TO THE CLIENT**

oauth.net/articles/authentication

OAuth 2.0 is not an authentication protocol.

Much of the confusion comes from the fact that OAuth is used *inside* of authentication protocols, and developers will see the OAuth components and interact with the OAuth flow and assume that by simply using OAuth, they can accomplish user authentication. This turns out to be not only untrue, but also dangerous for service providers, developers, and end users.

This article is intended to help potential *identity providers* with the question of how to build an authentication and identity API using OAuth 2.0 as the base. Essentially, if you're saying "I have OAuth 2.0, and I need authentication and identity", then read on.

What is authentication?

Authentication in the context of a user accessing an application tells an application *who the current user is* and whether or not they're present. A full authentication protocol will probably also tell you a number of *attributes* about this user, such as a unique identifier, an email address, and what to call them when the application says "Good Morning". Authentication is all about the user and their presence with the application, and an internet-scale authentication protocol needs to be able to do this across network and security boundaries.

However, OAuth tells the application none of that. OAuth says absolutely nothing about the user, nor does it say how the user proved their presence or even if they're still there. As far as an OAuth client is concerned, it asked for a token, got a token, and eventually used that token to access some API. It doesn't know anything about who authorized the application or if there was even a user there at all. In fact, much of the point of OAuth is about giving this delegated access for use in situations where the *user is not present* on the connection between the client and the resource being accessed. This is great for client authorization, but it's really bad for authentication where the whole point is figuring out if the user is there or not (and who they are).

As an additional confounder to our topic, an OAuth process does usually include several kinds of authentication in its process: the resource owner authenticates to the authorization server in the authorization step, the client authenticates to the authorization server in the token endpoint, and there may be others. The existence of these authentication events within the OAuth protocol does not translate to the OAuth protocol itself being able to reliably convey authentication.

As it turns out, though, there are a handful of things that can be used along with OAuth to *create an*

SUPPLEMENT OAUTH WITH OPENID CONNECT

OPENID CONNECT

OAuth client IDs, client secrets and access tokens



A user identifier attached to each request

**EDX HAS AN OPENID
CONNECT AUTH BACKEND**

edx/auth-backends

```
1 """Django authentication backends.
2
3 For more information visit https://docs.djangoproject.com/en/dev/topics/auth/customizing/.
4 """
5 import json
6
7 from django.conf import settings
8 import django.dispatch
9 import six
10 from jwkest.jwk import KEYS
11 from social_core.backends.open_id_connect import OpenIdConnectAuth
12
13
14 # pylint: disable=abstract-method
15 class EdXOpenIdConnect(OpenIdConnectAuth):
16     """ OpenID Connect backend designed for use with the Open edX auth provider. """
17     name = 'edx-oidc'
18
19     ACCESS_TOKEN_METHOD = 'POST'
20     REDIRECT_STATE = False
21     ID_KEY = 'preferred_username'
22
23     # Store the token type to ensure that we use the correct authentication mechanism for future calls.
24     EXTRA_DATA = OpenIdConnectAuth.EXTRA_DATA + ['token_type']
25
26     DEFAULT_SCOPE = ['openid', 'profile', 'email'] + getattr(settings, 'EXTRA_SCOPE', [])
27
28     PROFILE_TO_DETAILS_KEY_MAP = {
29         'preferred_username': 'username',
30         'email': 'email',
31         'name': 'full_name',
32         'given_name': 'first_name',
33         'family_name': 'last_name',
34         'locale': 'language',
35         'user_tracking_id': 'user_tracking_id',
36     }
37
38     auth_complete_signal = django.dispatch.Signal(provides_args=[user, id_token])
```

**CALLING IT
FROM EACH CLIENT**

pip install edx-auth-backends

```
324 # module, which uses the PHP's date() style. Format details are
325 # described at http://www.php.net/date.
326 DATE_FORMAT = 'F d, Y'
327 TIME_FORMAT = 'g:i A'
328
329 ##### AUTHENTICATION
330 AUTH_USER_MODEL = 'core.User'
331
332 INSTALLED_APPS += ('social.apps.django_app.default',)
333
334 # Allow authentication via edX OAuth2/OpenID Connect
335 AUTHENTICATION_BACKENDS = (
336     'auth_backends.backends.EdXOpenIdConnect',
337     'django.contrib.auth.backends.ModelBackend',
338 )
339
340 # Set to true if using SSL and running behind a proxy
341 SOCIAL_AUTH_REDIRECT_IS_HTTPS = False
342
343 SOCIAL_AUTH_ADMIN_USER_SEARCH_FIELDS = ['username', 'email']
344
345 SOCIAL_AUTH_PIPELINE = (
346     'social.pipeline.social_auth.social_details',
347     'social.pipeline.social_auth.social_uid',
348     'social.pipeline.social_auth.auth_allowed',
349     'social.pipeline.social_auth.social_user',
350
351     # By default python-social-auth will simply create a new user/username if the username
352     # from the provider conflicts with an existing username in this system. This custom pipeline function
353     # loads existing users instead of creating new ones.
354     'auth_backends.pipeline.get_user_if_exists',
355     'social.pipeline.user.get_username',
356     'social.pipeline.user.create_user',
357     'social.pipeline.social_auth.associate_user',
358     'social.pipeline.social_auth.load_extra_data',
359     'social.pipeline.user.user_details'
360 )
361
362 SOCIAL_AUTH_USER_FIELDS = ['username', 'email', 'first_name', 'last_name']
363
364 # Always raise auth exceptions so that they are properly logged. Otherwise, the PSA middleware will redirect to an
365 # auth error page and attempt to display the error message to the user (via Django's message framework). We do not
366 # want the user to see the message; but, we do want our downstream exception handlers to log the message.
367 SOCIAL_AUTH_RAISE_EXCEPTIONS = True
368
```

settings/base.py

```
358     social.pipeline.social_auth.load_extra_data ,
359     'social.pipeline.user.user_details'
360 )
361
362 SOCIAL_AUTH_USER_FIELDS = ['username', 'email', 'first_name', 'last_name']
363
364 # Always raise auth exceptions so that they are properly logged. Otherwise, the PSA middleware will redirect to an
365 # auth error page and attempt to display the error message to the user (via Django's message framework). We do not
366 # want the user to see the message; but, we do want our downstream exception handlers to log the message.
367 SOCIAL_AUTH_RAISE_EXCEPTIONS = True
368
369 # Set these to the correct values for your OAuth2/OpenID Connect provider
370 SOCIAL_AUTH_EDX_OIDC_KEY = None
371 SOCIAL_AUTH_EDX_OIDC_SECRET = None
372 SOCIAL_AUTH_EDX_OIDC_URL_ROOT = None
373
374 # This value should be the same as SOCIAL_AUTH_EDX_OIDC_SECRET
375 SOCIAL_AUTH_EDX_OIDC_ID_TOKEN_DECRYPTION_KEY = None
376
377 # Enables a special view that, when accessed, creates and logs in a new user.
378 # This should NOT be enabled for production deployments!
379 ENABLE_AUTO_AUTH = False
380
381 # Prefix for auto auth usernames. This value MUST be set in order for auto-auth to function. If it were not set
382 # we would be unable to automatically remove all auto-auth users.
383 AUTO_AUTH_USERNAME_PREFIX = 'AUTO_AUTH_'
384
385 # Maximum time (in seconds) before course permissions expire and need to be refreshed
386 COURSE_PERMISSIONS_TIMEOUT = 900
387
388 LOGIN_REDIRECT_URL = '/courses/'
389
390 # Determines if course permissions should be checked before rendering course views.
391 ENABLE_COURSE_PERMISSIONS = True
392
393 # What scopes and claims should be used to get courses
394 EXTRA_SCOPE = ['permissions', 'course_staff']
395 COURSE_PERMISSIONS_CLAIMS = ['staff_courses']
```

Login and Logout URLs

```
33 # using namespace. Once python-social-auth is updated to fix that, remove the namespace arg.
34 url('', include('social.apps.django_app.urls', namespace='social')),
35 url(r'^accounts/login/$',
36     RedirectView.as_view(url=reverse_lazy('social:begin', args=['edx-oidc']), permanent=False, query_string=True),
37     name='login'),
38 url(r'^accounts/logout/$', views.logout, name='logout'),
39 url(r'^accounts/logout_then_login/$', views.logout_then_login, name='logout_then_login'),
40 url(r'^test/auto_auth/$', views.AutoAuth.as_view(), name='auto_auth'),
41 url(r'^announcements/', include('pinax.announcements.urls', namespace='pinax_announcements')),
42 ]
43
```

MAIN POINTS

- 1 EdX is moving to microservices...
and auth is obviously ready
- 2 Built on popular standards: OAuth
and OpenID Connect*
- 3 Easy to build separately scalable
services with user auth

QUESTIONS?

miguel@ibleducation.com
@miguelamigot