
Installing, Configuring, and Running the Open edX Platform: Ironwood Release

Release

Mar 21, 2019

1	General Information	3
1.1	Read Me	3
1.2	Getting Help	3
1.3	Other edX Resources	4
1.4	edX Browser Support	9
2	Open edX Platform Releases	11
2.1	Open edX Ironwood Release	11
2.2	Open edX Hawthorn Release	13
2.3	Open edX Ginkgo Release	16
2.4	Open edX Ficus Release	18
2.5	Open edX Eucalyptus Release	20
2.6	Open edX Dogwood Release	22
2.7	Open edX Cypress Release	25
2.8	Open edX Birch Release	26
3	Installing and Starting the Open edX Platform	29
3.1	Open edX Platform Installation Options	29
3.2	Installation Prerequisites	31
3.3	Installing and Updating Devstack	32
3.4	Installing and Starting Analytics Devstack	33
3.5	Starting the Open edX Developer Stack	34
3.6	Troubleshooting Devstack	36
4	Configuring the Open edX Platform	41
4.1	Guidelines for Updating the Open edX Platform	41
4.2	Configuring Open edX Sites	41
4.3	Changing the Appearance of Open edX Sites	43
4.4	Adding Custom Fields to the Registration Page	51
4.5	Specifying Allowed Registration Email Patterns	52
4.6	Adding the CourseTalk Widget	53
4.7	Enabling Open edX Search	54
4.8	Enabling Badging	57
4.9	Enabling Course Certificates	67
4.10	Enabling Self-Paced Courses	73
4.11	Enabling Public Course Content	73
4.12	Enabling Custom Courses	76

4.13	Enabling Custom Course Settings	76
4.14	Enabling Discussion Notifications	77
4.15	Enabling Entrance Exams	77
4.16	Configuring Open Response Assessments	78
4.17	Enabling Course Prerequisites	79
4.18	Enabling Course and Video Licensing	80
4.19	Configuring Transcript Behavior	81
4.20	Configuring an edX Instance as an LTI Tool Provider	82
4.21	Enabling Social Sharing of Courses and Certificates	87
4.22	Configuring a Password Policy	89
4.23	Enabling Third Party Authentication	90
4.24	Enabling Timed Exams	116
4.25	Enabling the User Retirement Feature	117
4.26	Setting Up the YouTube API Key	125
4.27	Installing an XBlock	127
4.28	Enabling a CDN for Course Assets	127
5	Adding edX Insights for Course Teams	129
5.1	Options for Installing edX Insights	129
5.2	Using Elastic MapReduce on AWS	135
6	Adding E-Commerce to the Open edX Platform	143
6.1	Install and Start the E-Commerce Service	143
6.2	Comprehensive Theming	148
6.3	Manage Static Assets	152
6.4	Create E-Commerce Products	153
6.5	Enable the E-Commerce Service Receipt Page	166
6.6	Manage Orders	166
6.7	Test Your E-Commerce Application	167
6.8	Additional E-Commerce Features	172
6.9	Internationalization	176
7	Setting Up the Open edX Mobile Applications	179
7.1	Accessing the Source Code	179
7.2	Configuring Mobile Application Features	179
7.3	Configuring Video Modules for Mobile	181
7.4	Enabling Push Notifications	182
8	Index of Open EdX Feature Flags	183
9	Glossary	187
9.1	A	187
9.2	C	188
9.3	D	190
9.4	E	191
9.5	F	192
9.6	G	192
9.7	H	193
9.8	I	193
9.9	K	193
9.10	L	193
9.11	M	194
9.12	N	195
9.13	O	195
9.14	P	195

9.15	Q	196
9.16	R	196
9.17	S	197
9.18	T	198
9.19	U	198
9.20	V	198
9.21	W	199
9.22	XYZ	199

This guide provides instructions for using your own instance of the Open edX platform and associated applications.

This document applies to *Ironwood*, which is the most recent release of the Open edX platform. This document also contains instructions for installing Open edX releases.

You can install any release of the Open edX platform. For most situations, edX recommends that you install the most recent release. However, if your organization or project is currently running an earlier release, or your software requires a specific release, you may want to install an earlier release.

1.1 Read Me

The *Installing and Configuring the Open edX Platform* documentation is created using [RST](#) files and [Sphinx](#). As a member of the community, you can help update and revise this documentation project on GitHub.

https://github.com/edx/edx-documentation/tree/master/en_us/install_operations/source

The edX documentation team welcomes contributions from Open edX community members. You can find guidelines for how to [contribute to edX Documentation](#) in the GitHub [edx/edx-documentation](#) repository.

1.2 Getting Help

This section describes resources for getting help if you need technical assistance during the installation process or after your site is running.

1.2.1 Where to Find Help

There are a number of places to get help, including mailing lists and real-time chat. Please choose an appropriate venue for your question. This helps ensure that you get good prompt advice, and keeps discussion focused. For details of your options, see the [Getting Help](#) page.

1.2.2 Information to Provide When Asking for Help

Keep in mind when asking for help that you have much more knowledge of your situation than others do. You need to provide that context so that people can understand your problem and provide good help. To make it easier for others to help you, please provide the following types of information.

- **What you are trying to do?** What version of the code are you installing, and why? What is your larger goal?

- **What you have done?** What instructions were you following? What commands did you run? Did you deviate from the instructions at any point, even a little bit? Full output of those commands, even if it seems overwhelming, can be very useful.
- **What has gone wrong?** Are there errors in log files? Did you get specific failures in the terminal? Provide full details about the undesired results you saw.

While working through your problem, helpers often need additional information. Stay involved in the discussion, and try to give them what they need. They know best what information they need to solve the problem.

1.3 Other edX Resources

Learners, course teams, researchers, developers: the edX community includes groups with a range of reasons for using the platform and objectives to accomplish. To help members of each group learn about what edX offers, reach goals, and solve problems, edX provides a variety of information resources.

To help you find what you need, browse the edX offerings in the following categories.

- *Resources for edx.org Learners*
- *The edX Partner Portal*
- *The Open edX Portal*
- *System Status*
- *Resources for edx.org Course Teams*
- *Resources for Researchers*
- *Resources for Developers*
- *Resources for Open edX*

All members of the edX community are encouraged to make use of the resources described in this preface. We welcome your feedback on these edX information resources. Contact the edX documentation team at docs@edx.org.

1.3.1 Resources for edx.org Learners

Documentation

The [edX Help Center for Learners](#) includes topics to help you understand how to use the edX learning management system. The Help Center is also available when you select **Help** while you are in a course, and from your edX dashboard.

In a Course

If you have a question about something you encounter in an edX course, try these options for getting an answer.

Note: If you find an error or mistake in a course, contact the course staff by adding a post in the [course discussions](#).

- Check the **Course** page in the course. Course teams use this page to post updates about the course, which can include explanations about course content, reminders about when graded assignments are due, or announcements for upcoming events or milestones.
- Look for an “Introduction”, “Overview”, or “Welcome” section in the course content. In the first section in the course, course teams often include general information about how the course works and what you can expect, and also what they expect from you, in the first section in the course.
- Participate in the [course discussions](#). Other learners might be able to answer your question, or might have the same question themselves. If you encounter an unfamiliar word, phrase, or abbreviation, such as “finger exercise” or “board work”, search for it on the **Discussion** page, or post a question about it yourself. Your comments and questions give the course team useful feedback for improving the course.
- Investigate other resources. Some courses have a [wiki](#), which can be a good source of information. Outside of the course, a course-specific Facebook page or Twitter feed might be available for learners to share information.

Resources on the edx.org Website

To help you get started with the edX learning experience, edX offers a course (of course!). You can find the edX [Demo](#) course on the edx.org website.

When you are working in an edX course, you can select **Help** to access a help center with [frequently asked questions](#) and answers.

If you still have questions or suggestions, you can contact the [edX Support](#) team for help.

For opportunities to meet others who are interested in edX courses, check the edX Global Community [meetup](#) group.

1.3.2 The edX Partner Portal

The [edX Partner Portal](#) is the destination for partners to learn, connect, and collaborate with one another. Partners can explore rich resources and share success stories and best practices while staying up-to-date with important news and updates.

To use the edX Partner Portal, you must register and request verification as an edX partner. If you are an edX partner and have not used the edX Partner Portal, follow these steps.

1. Visit [partners.edx.org](#), and select **Create New Account**.
2. Select **Request Partner Access**, then fill in your personal details.
3. Select **Create New Account**. You will receive a confirmation email with your account access within 24 hours.

After you create an account, you can sign up to receive email updates about edX releases, news from the product team, and other announcements. For more information, see [Release Announcements by Email](#).

Course Team Support in the edX Partner Portal

EdX partner course teams can get technical support in the [edX Partner Portal](#). To access technical support, submit a support ticket, or review any support tickets you have created, go to [partners.edx.org](#) and select **Course Staff Support** at the top of the page. This option is available on every page in the Partner Portal.

1.3.3 The Open edX Portal

The [Open edX Portal](#) is the destination for learning about hosting an Open edX instance, extending the edX platform, and contributing to Open edX. In addition, the Open edX Portal provides product announcements and other community resources.

All users can view content on the Open edX Portal without creating an account and logging in.

To comment on blog posts or the edX roadmap, or subscribe to email updates, you must create an account and log in. If you do not have an account, follow these steps.

1. Visit open.edx.org/user/register.
2. Fill in your personal details.
3. Select **Create New Account**. You are then logged in to the [Open edX Portal](#).

Release Announcements by Email

To receive and share product and release announcements by email, you can subscribe to announcements on one of the edX portal sites.

1. Create an account on the [Open edX Portal](#) or the [edX Partner Portal](#) as described above.
2. Select **Community** and then **Announcements**.
3. Under **Subscriptions**, select the different types of announcements that you want to receive through email. You might need to scroll down to see these options.
4. Select **Save**.

You will now receive email messages when new announcements of the types you selected are posted.

1.3.4 System Status

For system-related notifications from the edX operations team, including outages and the status of error reports. On [Twitter](#), you can follow [@edxstatus](#).

Current system status and the uptime percentages for edX servers, along with the Twitter feed, are published on the [edX Status](#) web page.

1.3.5 Resources for edx.org Course Teams

Course teams include faculty, instructional designers, course staff, discussion moderators, and others who contribute to the creation and delivery of courses on [edx.org](#) or [edX Edge](#).

The edX Course Creator Series

The courses in the edX Course Creator Series provide foundational knowledge about using the edX platform to deliver educational experiences. These courses are available on [edx.org](#).

- *edX101: Overview of Creating a Course*
- *StudioX: Creating a Course with edX Studio*
- *BlendedX: Blended Learning with edX*
- *VideoX: Creating Video for the edX Platform*

edX101: Overview of Creating a Course

The [edX101](#) course is designed to provide a high-level overview of the course creation and delivery process using Studio and the edX LMS. It also highlights the extensive capabilities of the edX platform.

StudioX: Creating a Course with edX Studio

After you complete edX101, [StudioX](#) provides more detail about using Studio to create a course, add different types of content, and configure your course to provide an optimal online learning experience.

BlendedX: Blended Learning with edX

In [BlendedX](#) you explore ways to blend educational technology with traditional classroom learning to improve educational outcomes.

VideoX: Creating Video for the edX Platform

[VideoX](#) presents strategies for creating videos for course content and course marketing. The course provides step-by-step instructions for every stage of video creation, and includes links to exemplary sample videos created by edX partner institutions.

Documentation

Documentation for course teams is available from the docs.edx.org web page.

- [Building and Running an edX Course](#) is a comprehensive guide with concepts and procedures to help you build a course in Studio and then use the Learning Management System (LMS) to run a course.

You can access this guide by selecting **Help** in Studio or from the instructor dashboard in the LMS.

- [Using edX Insights](#) describes the metrics, visualizations, and downloadable .csv files that course teams can use to gain information about student background and activity.

These guides open in your web browser. The left side of each page includes a **Search docs** field and links to the contents of that guide. To open or save a PDF version, select **v: latest** at the lower right of the page, then select **PDF**.

Note: If you use the Safari browser, be aware that it does not support the search feature for the HTML versions of the edX guides. This is a known limitation.

Email

To receive and share information by email, course team members can:

- Subscribe to announcements and other new topics in the edX Partner Portal or the Open edX Portal. For information about how to subscribe, see [Release Announcements through the Open edX Portal](#).
- Join the [openedx-studio](#) Google group to ask questions and participate in discussions with peers at other edX partner organizations and edX staffers.

Wikis and Web Sites

The edX product team maintains public product roadmaps on *the Open edX Portal* and *the edX Partner Portal*.

The [edX Partner Support](#) site for edX partners hosts discussions that are monitored by edX staff.

1.3.6 Resources for Researchers

At each partner institution, the data czar is the primary point of contact for information about edX data. To set up a data czar for your institution, contact your edX partner manager.

Data for the courses on edx.org and edX Edge is available to the data czars at our partner institutions, and then used by database experts, statisticians, educational investigators, and others for educational research.

Resources are also available for members of the Open edX community who are collecting data about courses running on their sites and conducting research projects.

Documentation

The [edX Research Guide](#) is available on the docs.edx.org web page. Although it is written primarily for data czars and researchers at partner institutions, this guide can also be a useful reference for members of the Open edX community.

The *edX Research Guide* opens in your web browser, with a **Search docs** field and links to sections and topics on the left side of each page. To open or save a PDF version, select **v: latest** at the lower right of the page, and then select **PDF**.

Note: If you use the Safari browser, be aware that it does not support the search feature for the HTML versions of the edX guides. This is a known limitation.

Discussion Forums and Email

Researchers, edX data czars, and members of the global edX data and analytics community can post and discuss questions in our public research forum: the [openedx-analytics](#) Google group.

The edX partner portal also offers community [forums](#), including a Research and Analytics topic, for discussions among edX partners.

Important: Please do not post sensitive data to public forums.

Data czars who have questions that involve sensitive data, or that are institution specific, can send them by email to data.support@edx.org with a copy to your edX partner manager.

Wikis

The edX Analytics team maintains the [Open edX Analytics](#) wiki, which includes links to periodic release notes and other resources for researchers.

The [edx-tools](#) wiki lists publicly shared tools for working with the edX platform, including scripts for data analysis and reporting.

1.3.7 Resources for Developers

Software engineers, system administrators, and translators work on extending and localizing the code for the edX platform.

Documentation

Documentation for developers is available from the [edX Developer Documentation](#) landing page.

GitHub

These are the main edX repositories on GitHub.

- The [edx/edx-platform](#) repo contains the code for the edX platform.
- The [edx/edx-analytics-dashboard](#) repo contains the code for edX Insights.
- The [edx/configuration](#) repo contains scripts to set up and operate the edX platform.

Additional repositories are used for other projects. Our contributor agreement, contributor guidelines and coding conventions, and other resources are available in these repositories.

Getting Help

The [Getting Help](#) page in the Open edX Portal lists different ways that you can ask, and get answers to, questions.

Wikis and Web Sites

The [Open edX Portal](#) is the entry point for new contributors.

The edX Engineering team maintains an [open Confluence wiki](#), which provides insights into the plans, projects, and questions that the edX Open Source team is working on with the community.

The [edx-tools](#) wiki lists publicly shared tools for working with the edX platform, including scripts and helper utilities.

1.3.8 Resources for Open edX

Hosting providers, platform extenders, core contributors, and course staff all use Open edX. EdX provides release-specific documentation, as well as the latest version of all guides, for Open edX users. See the [Open edX documentation](#) page for a list of the documentation that is available.

1.4 edX Browser Support

The edX platform runs on the following browsers.

- Chrome
- Safari
- Firefox
- Microsoft Edge and Microsoft Internet Explorer 11

The edX platform is routinely tested and verified on the current version and the previous version of each of these browsers. We generally encourage the use of, and fully support only, the latest version.

Note: If you use the Safari browser, be aware that it does not support the search feature for the guides on docs.edx.org. This is a known limitation.

Open edX Platform Releases

The following sections provide information about releases of the Open edX platform.

2.1 Open edX Ironwood Release

This section describes how to install the Open edX Ironwood release.

- *What's Included in Ironwood*
- *What Is the Ironwood Git Tag?*
- *Installing the Ironwood Release*
- *Upgrading from the Hawthorn Release*
- *Upgrading to a Subsequent Ironwood Release*

2.1.1 What's Included in Ironwood

The Open edX Ironwood release contains several new features for learners, course teams, and developers. For more information, see the [Open edX Release Notes](#).

2.1.2 What Is the Ironwood Git Tag?

A git tag identifies the version of Open edX code that is the Ironwood release. About two dozen repositories are tagged as part of an Open edX release. Many other repositories are installed as dependencies of those repositories. You can find the most up-to-date git tag for the current Open edX release on the [Open edX Releases Wiki page](#).

2.1.3 Installing the Ironwood Release

You can install the Open edX Ironwood release using either *Devstack* or the [Open edX Native Installation](#) instructions. Ironwood releases have git tag names like `open-release/Ironwood.1`. The available names are detailed on the [Open edX Named Releases](#) page.

2.1.4 Upgrading from the Hawthorn Release

The recommended approach to upgrading an existing installation of the Open edX Hawthorn release to the Ironwood release is to make a fresh installation of the Hawthorn release on a new machine, and move your data and settings to it.

To move and upgrade your Hawthorn data onto an Ironwood installation, follow these steps.

1. Be sure that your Hawthorn installation is on the latest `open-release/hawthorn.master`. This ensures that your database is fully migrated and ready for upgrade to Hawthorn.
2. Stop all services on the Hawthorn machine.
3. Dump the data on the Hawthorn machine. Here's an example script that will dump the MySQL and Mongo databases into a single `.tgz` file. The script will prompt for the MySQL and Mongo passwords as needed.

```
#!/bin/bash
MYSQL_CONN="-uroot -p"
echo "Reading MySQL database names..."
mysql ${MYSQL_CONN} -ANe "SELECT schema_name FROM information_schema.
↳schemata WHERE schema_name NOT IN ('mysql','information_schema',
↳'performance_schema')" > /tmp/db.txt
DBS="--databases $(cat /tmp/db.txt)"
NOW="$(date +%Y%m%dT%H%M%S)"
SQL_FILE="mysql-data-${NOW}.sql"
echo "Dumping MySQL structures..."
mysqldump ${MYSQL_CONN} --add-drop-database --no-data ${DBS} > ${SQL_FILE}
echo "Dumping MySQL data..."
# If there is table data you don't need, add --ignore-table=tablename
mysqldump ${MYSQL_CONN} --no-create-info ${DBS} >> ${SQL_FILE}

for db in edxapp cs_comment_service_development; do
    echo "Dumping Mongo db ${db}..."
    mongodump -u admin -p -h localhost --authenticationDatabase admin -d $
↳{db} --out mongo-dump-${NOW}
done

tar -czf openedx-data-${NOW}.tgz ${SQL_FILE} mongo-dump-${NOW}
```

4. Copy the `.tgz` data file to the Ironwood machine.
5. Stop all services on the Ironwood machine.
6. Restore the Hawthorn data into the Ironwood machine. As an example, you might use the following commands.

```
$ tar -xvf openedx-data-20170811T154750.tgz
$ mysql -uroot -p < mysql-data-20170811T154750.sql
$ mongorestore -u admin -p -h localhost --authenticationDatabase admin --
↳drop -d edxapp mongo-dump-20170811T154750/edxapp
$ mongorestore -u admin -p -h localhost --authenticationDatabase admin --
↳drop -d cs_comment_service mongo-dump-20170811T154750/cs_comment_
↳service_development
```

7. To migrate data from Hawthorn to Ironwood, you need to drop the database tables used by djcelery. These tables should be empty in your Hawthorn data, so it is safe to drop them. The edx-platform application has a management command to check that they are empty and drop them:

```
$ sudo su - -s /bin/bash edxapp
edxapp@xyz:~$ . edxapp_env
edxapp@xyz:~$ cd edx-platform/
edxapp@xyz:~/edx-platform$ python manage.py lms drop_djcelery_tables --
↪ settings=aws
```

8. Run the Ironwood migrations, which will update your Hawthorn data to be valid for Ironwood:

```
$ /edx/app/edx_ansible/edx_ansible/util/install/sandbox.sh --tags migrate
```

9. Copy your configuration files from the Hawthorn machine to the Ironwood machine.
10. Restart all services.

2.1.5 Upgrading to a Subsequent Ironwood Release

Occasionally, we release updates to Ironwood. For example, the second release of Ironwood will be `open-release/ironwood.2`. The steps to upgrade differ based on your original installation method.

Upgrading a Docker Installation

Devstack is installed using Docker. To upgrade from one Ironwood release to another, follow the instructions in *Updating Devstack*.

Upgrading a Native Installation

If you installed Open edX using the [Open edX Native Installation](#), you can upgrade from one Ironwood release to another by re-running those steps using your desired Ironwood tag as the new value for `OPENEDX_RELEASE`.

2.2 Open edX Hawthorn Release

This section describes how to install the Open edX Hawthorn release.

- *What's Included in Hawthorn*
- *What Is the Hawthorn Git Tag?*
- *Installing the Hawthorn Release*
- *Upgrading from the Ginkgo Release*
- *Upgrading to a Subsequent Hawthorn Release*

2.2.1 What's Included in Hawthorn

The Open edX Hawthorn release contains several new features for learners, course teams, and developers. For more information, see the [Open edX Release Notes](#).

User Retirement Feature

The Hawthorn release also includes the new user retirement feature, which is a set of APIs and tooling that enables Open edX instances to retire registered users. There have been many changes to privacy laws (for example, GDPR or the European Union General Data Protection Regulation) intended to change the way that businesses think about and handle Personally Identifiable Information (PII) data. The user retirement feature is a step toward enabling Open edX to support some of the key updates in privacy laws. For more information, see [retirement](#).

2.2.2 What Is the Hawthorn Git Tag?

A git tag identifies the version of Open edX code that is the Hawthorn release. About two dozen repositories are tagged as part of an Open edX release. Many other repositories are installed as dependencies of those repositories. You can find the most up-to-date git tag for the current Open edX release on the [Open edX Releases Wiki page](#).

2.2.3 Installing the Hawthorn Release

You can install the Open edX Hawthorn release using either *Devstack* or the the [Open edX Native Installation](#) instructions.

Hawthorn releases have git tag names like `open-release/hawthorn.1`. The available names are detailed on the [Open edX Releases Wiki page](#).

2.2.4 Upgrading from the Ginkgo Release

The recommended approach to upgrading an existing installation of the Open edX Ginkgo release to the Hawthorn release is to make a fresh installation of the Hawthorn release on a new machine, and move your data and settings to it.

To move and upgrade your Ginkgo data onto a Hawthorn installation, follow these steps.

1. Be sure that your Ginkgo installation is on the latest `open-release/ginkgo.master`. This ensures that your database is fully migrated and ready for upgrade to Hawthorn.
2. Stop all services on the Ginkgo machine.
3. Dump the data on the Ginkgo machine. Here's an example script that will dump the MySQL and Mongo databases into a single `.tgz` file. The script will prompt for the MySQL and Mongo passwords as needed.

```
#!/bin/bash
MYSQL_CONN="-uroot -p"
echo "Reading MySQL database names..."
mysql ${MYSQL_CONN} -ANe "SELECT schema_name FROM information_schema.
↪schemata WHERE schema_name NOT IN ('mysql','information_schema',
↪'performance_schema')" > /tmp/db.txt
DBS="--databases $(cat /tmp/db.txt)"
NOW="$(date +%Y%m%dT%H%M%S)"
SQL_FILE="mysql-data-${NOW}.sql"
echo "Dumping MySQL structures..."
mysqldump ${MYSQL_CONN} --add-drop-database --no-data ${DBS} > ${SQL_FILE}
echo "Dumping MySQL data..."
# If there is table data you don't need, add --ignore-table=tablename
mysqldump ${MYSQL_CONN} --no-create-info ${DBS} >> ${SQL_FILE}

for db in edxapp cs_comment_service_development; do
    echo "Dumping Mongo db ${db}..."
```

```
mongodump -u admin -p -h localhost --authenticationDatabase admin -d $
↪ {db} --out mongo-dump- $\{NOW\}$ 
done
tar -czf openedx-data- $\{NOW\}$ .tgz  $\{SQL\_FILE\}$  mongo-dump- $\{NOW\}$ 
```

4. Copy the .tgz data file to the Hawthorn machine.
5. Stop all services on the Hawthorn machine.
6. Restore the Ginkgo data into the Hawthorn machine. As an example, you might use the following commands.

```
$ tar -xvf openedx-data-20170811T154750.tgz
$ mysql -uroot -p < mysql-data-20170811T154750.sql
$ mongorestore -u admin -p -h localhost --authenticationDatabase admin --
↪ drop -d edxapp mongo-dump-20170811T154750/edxapp
$ mongorestore -u admin -p -h localhost --authenticationDatabase admin --
↪ drop -d cs_comment_service mongo-dump-20170811T154750/cs_comment_
↪ service_development
```

7. To migrate data from Ginkgo to Hawthorn, you need to drop the database tables used by dcelery. These tables should be empty in your Ginkgo data, so it is safe to drop them. The edx-platform application has a management command to check that they are empty and drop them:

```
$ sudo su - -s /bin/bash edxapp
edxapp@xyz:~$ . edxapp_env
edxapp@xyz:~$ cd edx-platform/
edxapp@xyz:~/edx-platform$ python manage.py lms drop_djcelery_tables --
↪ settings=aws
```

8. Run the Hawthorn migrations, which will update your Ginkgo data to be valid for Ginkgo:

```
$ /edx/app/edx_ansible/edx_ansible/util/install/sandbox.sh --tags migrate
```

9. Copy your configuration files from the Ginkgo machine to the Hawthorn machine.
10. Restart all services.

>> ANYTHING ABOUT DJANGO 1.11?

2.2.5 Upgrading to a Subsequent Hawthorn Release

Occasionally, we release updates to Hawthorn. For example, the second release of Hawthorn will be `open-release/hawthorn.2`. The steps to upgrade differ based on your original installation method.

Upgrading a Docker Installation

Devstack is installed using Docker. To upgrade from one Hawthorn release to another, follow the instructions in *Updating Devstack*.

Upgrading a Native Installation

If you installed Open edX using the [Open edX Native Installation](#), you can upgrade from one Hawthorn release to another by re-running those steps using your desired Hawthorn tag as the new value for `OPENEDX_RELEASE`.

2.3 Open edX Ginkgo Release

This section describes the Open edX Ginkgo release.

- *What's Included in Ginkgo*
- *What Is the Ginkgo Git Tag?*
- *Upgrading from the Ficus Release*
- *Upgrading to a Subsequent Ginkgo Release*

2.3.1 What's Included in Ginkgo

The Open edX Ginkgo release contains several new features for learners, course teams, and developers. For more information, see [Open edX Ginkgo Release](#).

2.3.2 What Is the Ginkgo Git Tag?

A git tag identifies the version of Open edX code that is the Ginkgo release. You can find the most up-to-date git tag for the current Open edX release on the [Open edX Releases Wiki page](#).

The following Open edX git repositories have the Ginkgo git tag:

- edx-platform
- configuration
- course_discovery
- cs_comments_service
- xqueue
- ecommerce
- ecommerce-worker
- edx-analytics-configuration
- edx-analytics-dashboard
- edx-analytics-data-api
- edx-analytics-pipeline
- edx-certificates
- edx-custom-al ly-rules
- edx-demo-course
- edx-documentation
- edx-notes-api
- edx-ui-toolkit
- notifier
- ux-pattern-library

2.3.3 Upgrading from the Ficus Release

One approach to upgrading an existing installation of the Open edX Ficus release to the Ginkgo release is to make a fresh installation of the Ginkgo release on a new machine, and move your data and settings to it.

To move and upgrade your Ficus data onto a Ginkgo installation, follow these steps.

1. Be sure that your Ficus installation is on Ficus.4. The Ficus.4 release provided required database migrations beyond what was in Ficus.3. The only version of Ficus that will upgrade to Ginkgo successfully is Ficus.4.
2. Stop all services on the Ficus machine.
3. Dump the data on the Ficus machine. Here's an example script that will dump the MySQL and Mongo databases into a single .tgz file. The script will prompt for the MySQL and Mongo passwords as needed.

```
#!/bin/bash
MYSQL_CONN="-uroot -p"
echo "Reading MySQL database names..."
mysql ${MYSQL_CONN} -ANe "SELECT schema_name FROM information_schema.
↳ schemata WHERE schema_name NOT IN ('mysql','information_schema',
↳ 'performance_schema')" > /tmp/db.txt
DBS="--databases $(cat /tmp/db.txt)"
NOW="$(date +%Y%m%dT%H%M%S)"
SQL_FILE="mysql-data-{$NOW}.sql"
echo "Dumping MySQL structures..."
mysqldump ${MYSQL_CONN} --add-drop-database --no-data {$DBS} > {$SQL_FILE}
echo "Dumping MySQL data..."
# If there is table data you don't need, add --ignore-table=tablename
mysqldump ${MYSQL_CONN} --no-create-info {$DBS} >> {$SQL_FILE}

for db in edxapp cs_comment_service_development; do
    echo "Dumping Mongo db {$db}..."
    mongodump -u admin -p -h localhost --authenticationDatabase admin -d {$
↳ {$db} --out mongo-dump-{$NOW}
done
tar -czf openedx-data-{$NOW}.tgz {$SQL_FILE} mongo-dump-{$NOW}
```

4. Copy the .tgz data file to the Ginkgo machine.
5. Stop all services on the Ginkgo machine.
6. Restore the Ficus data into the Ginkgo machine. As an example, you might use the following commands.

```
$ tar -xvf openedx-data-20170811T154750.tgz
$ mysql -uroot -p < mysql-data-20170811T154750.sql
$ mongorestore -u admin -p -h localhost --authenticationDatabase admin --
↳ drop -d edxapp mongo-dump-20170811T154750/edxapp
$ mongorestore -u admin -p -h localhost --authenticationDatabase admin --
↳ drop -d cs_comment_service mongo-dump-20170811T154750/cs_comment_
↳ service_development
```

7. To migrate data from Ficus to Ginkgo, you need to drop the database tables used by djcelery. These tables should be empty in your Ficus data, so it is safe to drop them. The edx-platform application has a management command to check that they are empty and drop them:

```
$ sudo su - -s /bin/bash edxapp
edxapp@xyz:~$ . edxapp_env
edxapp@xyz:~$ cd edx-platform/
edxapp@xyz:~/edx-platform$ python manage.py lms drop_djcelery_tables --
↳ settings=aws
```

8. Run the Ginkgo migration script, which will update your Ficus data to be valid for Ginkgo:

```
$ /edx/app/edx_ansible/edx_ansible/util/install/sandbox.sh --tags migrate
```

9. Copy your configuration files from the Ficus machine to the Ginkgo machine.
10. Restart all services.

Upgrading Django Oscar

The Ginkgo release of Open edX upgrades Django Oscar to version 1.4. If you have an existing installation of Open edX with the E-Commerce service, and your `orders` table is larger than a million rows, there is an additional step to upgrade your Django Oscar installation.

The migration includes a change to the `guest_email` column in the `orders` table. This change is applied automatically. If your `orders` table is larger than a million rows, this migration may lock the table for an extended amount of time. The E-Commerce service does not normally use the `guest_email` column. If your installation does not use this column, you can avoid the table lock by using the `--fake` argument in migrating the `orders` table, running the `migrate` command in the following form.

```
./manage.py migrate orders 0013 --fake
```

2.3.4 Upgrading to a Subsequent Ginkgo Release

Occasionally, we release updates to Ginkgo. For example, the second release of Ginkgo will be `open-release/ginkgo.2`. The steps to upgrade differ based on your original installation method.

Upgrading a Vagrant Installation

Devstack and Fullstack are installed using Vagrant. To upgrade from one Ginkgo release to another, run the following commands in the host operating system:

```
$ export OPENEDX_RELEASE=open-release/ginkgo.2
$ vagrant provision
```

Upgrading a Native Installation

If you installed Open edX using the [Open edX Native Installation](#), you can upgrade from one Ginkgo release to another by re-running those steps using your desired Ginkgo tag as the new value for `OPENEDX_RELEASE`.

2.4 Open edX Ficus Release

This section describes the Open edX Ficus release. Note that edX no longer supports the Ficus releases.

- *What's Included in Ficus*
- *What Is the Ficus Git Tag?*

- *Upgrading to a Subsequent Ficus Release*

2.4.1 What's Included in Ficus

The Open edX Ficus release contains several new features for learners, course teams, and developers. For more information, see [Open edX Ficus Release](#).

2.4.2 What Is the Ficus Git Tag?

A git tag identifies the version of Open edX code that is the Ficus release. You can find the most up-to-date git tag for the current Open edX release on the [Open edX Releases Wiki](#) page.

The following Open edX git repositories have the Ficus git tag:

- edx-platform
- configuration
- cs_comments_service
- xqueue
- ecommerce
- ecommerce-worker
- edx-analytics-configuration
- edx-analytics-dashboard
- edx-analytics-data-api
- edx-analytics-pipeline
- edx-certificates
- edx-custom-all-y-rules
- edx-demo-course
- edx-documentation
- edx-notes-api
- edx-ui-toolkit
- notifier
- programs
- ux-pattern-library

2.4.3 Upgrading to a Subsequent Ficus Release

Occasionally, we release updates to Ficus. For example, the second release of Ficus is `open-release/ficus.2`. The steps to upgrade differ based on your original installation method.

Upgrading a Vagrant Installation

Devstack and Fullstack are installed using Vagrant. To upgrade from one Ficus release to another, run the following commands in the host operating system:

```
$ export OPENEDX_RELEASE=open-release/ficus.2
$ vagrant provision
```

Upgrading a Native Installation

If you installed Open edX using the [Open edX Native Installation](#), you can upgrade from one Ficus release to another by re-running those steps using your desired Ficus tag as the new value for `OPENEDX_RELEASE`.

2.5 Open edX Eucalyptus Release

This section describes the Open edX Eucalyptus release. Note that edX no longer supports the Eucalyptus release.

- *What's Included in Eucalyptus*
- *What Is the Eucalyptus Git Tag?*
- *Upgrading from Dogwood to Eucalyptus*
- *Upgrading to a Subsequent Eucalyptus Release*

2.5.1 What's Included in Eucalyptus

The Open edX Eucalyptus release contains several new features for learners, course teams, and developers. For more information, see [Open edX Eucalyptus Release](#).

2.5.2 What Is the Eucalyptus Git Tag?

A Git tag identifies the version of Open edX code that is the Eucalyptus release. You can find the most up-to-date Git tag for the current Open edX release on the [Open edX Releases Wiki](#) page.

The following Open edX Git repositories have the Eucalyptus Git tag.

- edx-platform
- configuration
- cs_comments_service
- xqueue
- XBlock
- notifier
- edx-ora2
- edx-documentation
- edx-certificates

- edx-analytics-data-api-client
- edx-analytics-configuration
- edx-analytics-dashboard
- edx-analytics-data-api
- edx-analytics-pipeline

2.5.3 Upgrading from Dogwood to Eucalyptus

You can upgrade an Open edX instance that is running the Dogwood release to the Eucalyptus release. EdX provides the `upgrade.sh` script if you have a simple Dogwood installation and want to upgrade it automatically. If you have a more complex or customized installation, you may need to upgrade manually.

The `upgrade.sh` script is in the edX configuration repository on GitHub.

Note: The upgrade script is only for upgrading instances running the Dogwood release. If your instance is running a release prior to the Dogwood release, follow the instructions to upgrade it to each intervening release, and then upgrade from Dogwood to Eucalyptus.

Caution: Before upgrading your Open edX instance, back up all data and configuration files. Then verify that you can restore your Open edX instance from the backup files.

On the computer or virtual machine that is running the Dogwood release of Open edX, run the upgrade script for your type of installation.

1. Download the script.

```
$ export OPENEDX_RELEASE=open-release/eucalyptus.1
$ curl -OL https://raw.githubusercontent.com/edx/configuration/$OPENEDX_RELEASE/util/vagrant/
↪upgrade.sh
```

2. Run the script.

- For devstack, run `bash upgrade.sh -c devstack`.
- For fullstack, run `bash upgrade.sh -c fullstack`.

You can find the most up-to-date Git tag for the current Open edX release on the [Open edX Releases Wiki page](#).

You can also run `bash upgrade.sh -h` to see which other options the script accepts.

The script creates a temporary directory in which it upgrades Open edX, then cleans up extra files and directories when it finishes running.

After upgrading Open edX to the Eucalyptus release, start the LMS and Studio and verify that course content and data was migrated correctly.

2.5.4 Upgrading to a Subsequent Eucalyptus Release

Occasionally, we release updates to Eucalyptus. For example, the second release of Eucalyptus is `open-release/eucalyptus.2`. The steps to upgrade differ based on your original installation method.

Upgrading a Vagrant Installation

Devstack and Fullstack for Eucalyptus are installed using Vagrant. To upgrade to a Eucalyptus point release, follow these steps in the host operating system.

```
$ export OPENEDX_RELEASE=open-release/eucalyptus.2
$ vagrant provision
```

Upgrading a Native Installation

If you installed Open edX using the [Open edX Native 12.04 Installation](#), re-run those steps using your desired Eucalyptus tag as the new value for `OPENEDX_RELEASE`.

2.6 Open edX Dogwood Release

This section describes the Open edX Dogwood release. Note that edX no longer supports the Dogwood release.

- *What's Included in Dogwood*
- *What Is the Dogwood Git Tag?*
- *Upgrading from Cypress to Dogwood*
- *Upgrading to a Dogwood Point Release*

Note: Now that the Open edX Eucalyptus release is available, edX no longer supports the Dogwood release.

2.6.1 What's Included in Dogwood

The Open edX Dogwood release contains several new features for learners, course teams, and developers. See the release notes for the [Open edX Dogwood Release](#) for more details.

2.6.2 What Is the Dogwood Git Tag?

A Git tag identifies the version of Open edX code that is the Dogwood release. You can find the most up-to-date Git tag for the current Open edX release on the [Open edX Releases Wiki page](#).

The following Open edX Git repositories have the Dogwood Git tag.

- edx-platform
- configuration
- cs_comments_service
- xqueue
- XBlock
- notifier
- edx-ora2

- `edx-documentation`
- `edx-certificates`
- `edx-analytics-data-api-client`
- `edx-analytics-configuration`
- `edx-analytics-dashboard`
- `edx-analytics-data-api`
- `edx-analytics-pipeline`

2.6.3 Upgrading from Cypress to Dogwood

You can upgrade an Open edX instance that is running the Cypress release to the Dogwood release. EdX provides the `migrate.sh` script if you have a simple Cypress installation and want to upgrade it automatically. If you have a more complex or customized installation, you may need to upgrade manually.

Automatic Upgrading

The `migrate.sh` script is in the edX configuration repository on GitHub.

Note: The upgrade scripts provided are verified only for upgrading instances running the Cypress release. If you are running any other version of the Open edX Platform, the upgrade scripts might not work.

Caution: Before upgrading your Open edX instance, back up all data and configuration files. Then verify that you can restore your Open edX instance from the backup files.

On the computer or virtual machine that is running the Cypress release of Open edX, run the upgrade script for your type of installation.

- For devstack, run `./migrate.sh -c devstack -t named-release/dogwood`.
- For fullstack, run `./migrate.sh -c fullstack -t named-release/dogwood`.

You can find the most up-to-date Git tag for the current Open edX release on the [Open edX Releases Wiki page](#).

You can also run `./migrate.sh -h` to see which other options the script accepts.

The script creates a temporary directory in which it upgrades Open edX, then cleans up extra files and directories when it finishes running.

After upgrading Open edX to the Dogwood release, start the LMS and Studio and verify that course content and data was migrated correctly.

Upgrade Process Overview

This is an overview of what happens during an upgrade from Cypress to Dogwood. The `migrate.sh` script implements this process. You may need to understand this process if your installation is customized in some way, or if you need to diagnose problems during the upgrade.

Upgrading Cypress to Dogwood is more involved than most Open edX release upgrades.

- Dogwood upgrades the Django framework from version 1.4 to 1.8, which changed the database migration tool from South to Django. When you upgrade from Cypress to Dogwood, it is important to take special care with the database migrations.
- Dogwood upgrades Python from 2.7.3 to 2.7.10. This means virtualenvs have to be recreated.

The upgrade from Cypress to Dogwood includes these steps.

1. Applies a [forum migration described on the Open edX wiki](#) to support teams discussion filtering.
2. Updates edx-platform to the `release-2015-11-09` tag. This is the last released version that used Django 1.4.
3. Recreates the virtualenvs to use Python 2.7.10 instead of 2.7.3.
4. Migrates the database. This makes the database current with the last 1.4 code.
5. Uninstalls South so that it does not interfere with the new Django migrations.
6. Updates edx-platform to the `dogwood-first-18` tag. This is the first version of the code that used Django 1.8.
7. Applies all the initial Django migrations. This gets your database ready to use the new Django 1.8 migration mechanism.
8. Updates edx-platform to Dogwood.
9. Runs Django database migrations.
10. Runs two management commands to update records in the database: `generate_course_overview` and `post_cohort_membership_fix`.
11. Runs the forum migration again. This step processes any discussion topics created during the running of the script.

Similar steps are followed to upgrade other repositories such as xqueue.

2.6.4 Upgrading to a Dogwood Point Release

Occasionally, we release updates to Dogwood. The first of these is named Dogwood.1, then Dogwood.2, and so on. The steps differ based on your original installation method. You will need to know the name of the Dogwood tag you want to install, for example `named-release/dogwood.2`.

Upgrading a Vagrant Installation

Devstack and Fullstack are installed using Vagrant. To upgrade to a Dogwood point release, follow these steps in the host operating system.

```
$ export OPENEDX_RELEASE={desired-dogwood-tag}
$ vagrant provision
```

Upgrading a Native Installation

If you installed Open edX using the [Open edX Native 12.04 Installation](#), re-run those steps using your desired Dogwood tag as the new value for `OPENEDX_RELEASE`.

2.7 Open edX Cypress Release

This section describes the Open edX Cypress release. Note that edX no longer supports the Cypress release.

- *What's Included in Cypress*
- *What is the Cypress Git Tag?*
- *Upgrading from Birch to Cypress*

2.7.1 What's Included in Cypress

The Open edX Cypress release contains several new features for learners, course teams, and developers. See the Open edX Release Notes for more details.

Note: There are several new features in the Cypress release that are available, but not enabled by default in new installations. For details, see the following topics.

- *Enabling Open edX Search*
 - *Enabling Badging*
 - *Enabling Custom Courses*
 - *Enabling Course and Video Licensing*
-

2.7.2 What is the Cypress Git Tag?

The Git tag for the Cypress release is named `-release/cypress`. You use this tag to identify the version of Open edX code that is the Cypress release.

The following Open edX Git repositories have the Git tag named `-release/cypress`.

- `edx-platform`
- `configuration`
- `cs_comments_service`
- `notifier`
- `edx-certificates`
- `xqueue`
- `edx-documentation`
- `edx-ora2`
- `XBlock`

2.7.3 Upgrading from Birch to Cypress

You can upgrade an Open edX instance that is running the Birch release to the Cypress release, by using the `migrate.sh` script in the configuration repository, [available here](#).

Note: The upgrade scripts provided are verified only for upgrading instances running the Birch release. If you are running any other version of the Open edX Platform, the upgrade scripts might not work.

Caution: Before upgrading your Open edX instance, back up all data and configuration files. Then verify that you can restore your Open edX instance from the backup files.

On the computer or virtual machine that is running the Birch release of Open edX, run the upgrade script for your type of installation:

- For devstack, run `./migrate.sh -c devstack`.
- For fullstack, run `./migrate.sh -c fullstack`.
- You can also run `./migrate.sh -h` to see which other options the script accepts.

The script creates a temporary directory in which it upgrades Open edX, then cleans up extra files and directories when it finishes running.

After upgrading Open edX to the Cypress release, start the LMS and Studio and verify that course content and data was migrated correctly.

2.8 Open edX Birch Release

This section describes how to install the Open edX Birch release.

- *What's Included in Birch*
- *What is the Birch Git Tag?*
- *Upgrading from Aspen to Birch*

Note: EdX no longer supports the Birch release.

2.8.1 What's Included in Birch

The Open edX Birch release contains several new features for students, course teams, and developers. See the Open edX Release Notes for more details.

Note: There are several new features in the Birch release that are available, but not configured in new installations. For details, see the following topics.

- *Enabling Course Prerequisites*
 - *Enabling Entrance Exams*
-

2.8.2 What is the Birch Git Tag?

The Git tag for the Birch release is **named-release/birch.2**. You use this tag to identify the version of Open edX code that is the Birch release.

The following Open edX Git repositories have the Git tag **named-release/birch.2**:

- edx-platform
- configuration
- cs_comments_service
- notifier
- edx-certificates
- xqueue
- edx-documentation
- edx-ora2
- XBlock

2.8.3 Upgrading from Aspen to Birch

You can upgrade your Open edX instance that is running the Aspen release to the Birch release, using the `migrate.sh` script in the configuration repository, [available here](#).

Note: The upgrade scripts provided are verified only for upgrading instances running the Aspen release. If you are running any other version of the Open edX Platform, the upgrade scripts might not work.

Caution: Before upgrading your Open edX instance, back up all data and configuration files. Then verify that you can restore your Open edX instance from the backup files.

On the computer or virtual machine running the Aspen release of Open edX, run the upgrade script for your type of installation:

- For devstack, run `./migrate.sh -t named-release/birch.2 -c devstack`.
- For fullstack, run `./migrate.sh -t named-release/birch.2 -c fullstack`.
- You can also run `./migrate.sh -h` to see which other options the script accepts.

The script creates a temporary directory in which it upgrades Open edX, then cleans up extra files and directories when it finishes running.

After upgrading Open edX to the Birch release, run the edX Platform and verify that course content and data was migrated correctly.

Installing and Starting the Open edX Platform

The following sections provide information about how to install and start the latest version of the Open edX platform.

3.1 Open edX Platform Installation Options

This section describes Open edX installation options and the components that each option includes. There are two development environment installation options, which install the Open edX software using Docker. If you prefer, you can install into an Ubuntu machine of your own using the Native installation.

- *Open edX Platform on Docker*
- *Native Installation*
- *Software Components*

3.1.1 Open edX Platform on Docker

You can install the Open edX developer stack (**Devstack**) or the Open edX analytics developer stack (**Analytics Devstack**).

- Devstack is a set of Docker containers designed for local development. For more information, see *Open edX Devstack*.
- Analytics Devstack is a modified version of the Devstack installation that allows you to run Open edX Analytics. For more information, see *Open edX Analytics Devstack*.

You can run Devstack or Analytics Devstack on Linux or macOS. See the [Docker](#) downloads page for information about the operating systems and architectures on which you can run Docker.

Devstack using [Docker for Windows](#) has not been tested and it is not supported.

Open edX Devstack

Devstack is a deployment of the Open edX platform within a set of Docker containers designed for local development. Running the Open edX platform locally allows you to discover and fix system configuration issues early in development.

Devstack simplifies certain production settings to make development more convenient. For example, `nginx` and `gunicorn` are disabled in Devstack; Devstack uses Django's `runserver` instead.

Note: Because of the large number of dependencies needed to develop extensions to Open edX Insights, a separate development environment is available to support Analytics development. For more information, see *Installing and Starting Analytics Devstack*.

For more information about Docker, see the [Docker documentation](#).

Open edX Analytics Devstack

Some users might want to develop Analytics features on their instance of the Open edX platform. Because of the large number of dependencies needed to develop extensions to Analytics, edX has created a separate developer stack, known as Analytics Devstack. We strongly recommend that you install the Analytics Devstack instead of adding Analytics extensions to an instance of devstack.

Analytics Devstack is a modified version of the *Open edX developer stack*. This development environment provides all of the services and tools needed to modify the Open edX Analytics Pipeline, Data API, and Insights projects.

3.1.2 Native Installation

The Native installation installs the Open edX software on your own Ubuntu 16.04 machine in a production-like configuration. Details are at the [Open edX Native Installation](#) page on the edX wiki.

3.1.3 Software Components

A Devstack installation includes the following Open edX components:

- The Learning Management System (LMS)
- Open edX Studio
- Discussion Forums
- Open Response Assessments (ORA)
- E-Commerce
- Credentials
- Notes
- Course Discovery
- XQueue
- Open edX Search
- A demonstration Open edX course

Analytics Devstack also includes the following Open edX components:

- Open edX Analytics Data API
- Open edX Insights
- The components needed to run the Open edX Analytics Pipeline. This is the primary extract, transform, and load (ETL) tool that extracts and analyzes data from the other Open edX services.

3.2 Installation Prerequisites

To use Devstack or Analytics Devstack successfully, you must meet the following knowledge and software requirements.

3.2.1 Knowledge Prerequisites

Devstack and Analytics Devstack require an understanding of the following items.

- Basic terminal usage. If you are using a Mac computer, see [Introduction to the Mac OS X Command Line](#).
- Diagnosing and fixing failures may involve many different technologies and skills. It will help to know these things.
 - The basics of how Python web applications are built, installed, and deployed.
 - How to manage a Linux system, including supervisor.
 - The basics of configuration management and automation. We use [Ansible](#) to automate the installation process.

3.2.2 Software Prerequisites

Devstack and Analytics Devstack require the following software.

- `make`
- [Docker](#) 17.06 CE or later. We recommend Docker Stable, but Docker Edge should work as well.

You must have `docker-compose` in your path. On macOS, installing Docker for Mac takes care of this requirement.

Allocate Sufficient Resources to Docker

Since a Docker-based Devstack runs many containers, you should configure Docker with a sufficient amount of resources. We find that configuring Docker for Mac with a minimum of 2 CPUs and 6GB of memory works well. For more information, see the [Docker documentation](#).

Use overlay2 on Linux

If you are using Linux, use the `overlay2` storage driver, kernel version 4.0+ and *not* `overlay`. To check which storage driver your `docker-daemon` uses, run the following command.

```
docker info | grep -i 'storage driver'
```

3.3 Installing and Updating Devstack

This section provides information about how to install and update the Open edX developer stack (Devstack).

- *Installation Prerequisites for Devstack*
- *Install Devstack*
- *Updating Devstack*

3.3.1 Installation Prerequisites for Devstack

Before you install Devstack, make sure that you have met the *installation prerequisites*.

3.3.2 Install Devstack

To install Devstack, follow these steps.

1. Decide which branch you will be working with. “master” is the latest code in the repositories, changed daily. Open edX releases are more stable, for example, Hawthorn.
2. Check out a local copy of the `edx/devstack` repository from `https://github.com/edx/devstack`.

```
git clone https://github.com/edx/devstack
```

3. Navigate to the `devstack` directory

```
cd devstack
```

4. If you are not using the master branch, check out the branch you want.

```
git checkout open-release/hawthorn.master
```

5. If you are not using the master branch, define an environment variable for the Open edX version you are using, such as `hawthorn.master` or `zebrawood.rc1`. Note that unlike a server install, the value of the `OPENEDX_RELEASE` variable should not use the `open-release/` prefix.

```
export OPENEDX_RELEASE=hawthorn.master
```

6. Run `make dev.checkout` to check out the correct branch in the local checkout of each service repository.

```
make dev.checkout
```

7. Clone the Open edX service repositories. The Docker Compose file mounts a host volume for each service’s executing code. The host directory defaults to be a sibling of the `/devstack` directory. For example, if you clone the `edx/devstack` repository to `~/workspace/devstack`, host volumes will be expected in `~/workspace/course-discovery`, `~/workspace/ecommerce`, etc. You can clone these repositories with the following command.

```
make dev.clone
```

To customize where the local repositories are found, set the `DEVSTACK_WORKSPACE` environment variable.

- (macOS only) Share the cloned service directories in Docker, using **Docker -> Preferences -> File Sharing** in the Docker menu.
- Run the provision command to configure the various services with superusers (for development without the auth service) and tenants (for multi-tenancy).

Note: When you run the provision command, databases for `ecommerce` and `edxapp` will be dropped and recreated.

Use the following default provision command.

```
make dev.provision
```

The default username and password for the superusers are both `edx`. You can access the services directly using Django admin at the `/admin/` path, or log in using single sign-on at `/login/`.

When you have completed these steps, see *Starting the Open edX Developer Stack*.

For help with running Devstack, see *Troubleshooting Devstack*.

3.3.3 Updating Devstack

EdX publishes new images for Open edX services frequently. After you have installed and started Devstack, you can update your Devstack installation to use the most up-to-date versions of the Devstack images by running the following sequence of commands.

```
make down
make pull
make dev.up
```

This stops any running Devstack containers, pulls the latest images, and then starts all of the Devstack containers.

3.4 Installing and Starting Analytics Devstack

This section provides information about how to install and start the Open edX analytics developer stack (Analytics Devstack).

- *Installation Prerequisites for Analytics Devstack*
- *Install Analytics Devstack*

3.4.1 Installation Prerequisites for Analytics Devstack

Before you install Analytics Devstack, make sure that you have met the *installation prerequisites*.

3.4.2 Install Analytics Devstack

To install Analytics Devstack, follow these steps.

1. Follow steps 1 through 6 in *Install Devstack*.

2. Pull the relevant Docker images by running the following commands.

```
make pull
make pull.analytics_pipeline
```

3. Configure the Analytics Devstack by running the following provision command.

```
make dev.provision.analytics_pipeline
```

4. Start the analytics service by running the following command.

```
make dev.up.analytics_pipeline
```

This command mounts the repositories under the `DEVSTACK_WORKSPACE` directory. Note that it may take up to 60 seconds for Hadoop services to start.

1. Access the analytics pipeline shell by running the following command.

```
make analytics-pipeline-shell
```

Viewing Logs

To see logs from containers running in detached mode, run the following command.

```
make logs
```

As an alternative, you can use Docker's Kitematic feature, which is available from the **Docker for Mac** menu.

To view the logs of a specific service container, run a `make <service>-logs` command. For example, to access the logs for Hadoop's namenode, run this command.

```
make namenode-logs
```

For additional information, see *Starting the Open edX Developer Stack*.

For help with running the Analytics Devstack, see *Troubleshooting Devstack*.

3.5 Starting the Open edX Developer Stack

After you have installed Devstack within Docker, you can start Open edX services. You start the edX services with the following command. This command mounts the repositories under the `DEVSTACK_WORKSPACE` directory.

```
make dev.up
```

3.5.1 Connecting to Services

After it starts, each service is accessible at `localhost` on a specific port. The table below provides links to the homepage of each service. Since some services are not meant to be user-facing, the homepage may be the API root.

Service	URL
Catalog/Discovery	http://localhost:18381/api-docs/
Credentials	http://localhost:18150/api/v2/
E-Commerce/Otto	http://localhost:18130/dashboard/
LMS	http://localhost:18000/
Notes/edx-notes-api	http://localhost:18120/api/v1/
Studio/CMS	http://localhost:18010/

After the services have started, you can get shell access to one of the services with a `make <service>-shell` command. For example, to access the Catalog/Course Discovery Service, run this command.

```
make discovery-shell
```

3.5.2 Viewing Logs

To see logs from containers running in detached mode, run the following command.

```
make logs
```

As an alternative, you can use Docker's Kitematic feature, which is available from the **Docker for Mac** menu.

To view the logs of a specific service container, run a `make <service>-logs` command. For example, to access the logs for the Ecommerce service, run this command.

```
make ecommerce-logs
```

3.5.3 Restarting Services

To restart a particular service, run a `docker-compose restart <service>` command. For example, to restart the Studio service, run the following command.

```
docker-compose restart studio
```

3.5.4 Restarting Devstack

To reset your environment and start provisioning from scratch, run the following command.

```
make destroy
```

For information on all the available make commands, run the following command.

```
make help
```

3.5.5 Default Accounts

When you install an Open edX system, the following user accounts are created by default.

Account	Description
staff@example.com	An LMS and Studio user with course creation and editing permissions. This user is a course team member with the Admin role, which gives rights to work with the demonstration course in Studio, the LMS, and Insights.
verified@example.com	A student account that you can use to access the LMS for testing verified certificates.
audit@example.com	A student account that you can use to access the LMS for testing course auditing.
honor@example.com	A student account that you can use to access the LMS for testing honor code certificates.

The default password for all of these accounts is `edx`.

3.6 Troubleshooting Devstack

If you are having trouble running Devstack on Docker, here are some troubleshooting tips.

3.6.1 Check the logs

If a container stops unexpectedly, you can look at its logs for clues:

```
docker-compose logs lms
```

3.6.2 Update the code and images

Make sure you have the latest code and Docker images.

Pull the latest Docker images by running the following command from the `devstack` directory.

```
make pull
```

Pull the latest Docker Compose configuration and provisioning scripts by running the following command from the `devstack` directory.

```
git pull
```

The images are built from the master branches of the application repositories (for example, `edx-platform`, `ecommerce`, etc.). Make sure you are using the latest code from the master branches, or have rebased your branches on master.

3.6.3 Clean the containers

Sometimes containers end up in strange states and need to be rebuilt. Run `make down` to remove all containers and networks. This will **not** remove your data volumes.

3.6.4 Reset your local installation

Sometimes you just aren't sure what's wrong, if you would like to hit the reset button run `make dev.reset`.

Running this command will perform the following steps:

- Bring down all containers.
- Reset all git repositories to the HEAD of master.
- Pull new images for all services.
- Compile static assets for all services.
- Run migrations for all services.

3.6.5 Start over

If you want to completely start over, run `make destroy`. This command removes all containers, networks, **and** data volumes.

3.6.6 Resetting a database

In case you botched a migration or just want to start with a clean database.

1. Open up the mysql shell and drop the database for the desired service, using the following commands.:

```
make mysql-shell
mysql
DROP DATABASE (insert database here)
```

2. From your `devstack` directory, run the provision script for the service. The provision script should handle populating data such as Oauth clients and Open edX users, and running migrations.

```
./provision-(service_name)
```

3.6.7 File ownership change

If you notice that the ownership of some or all files have changed and you need to enter your root password when editing a file, you might have pulled changes to the remote repository from within a container. When you run `git pull`, `git` changes the owner of the files that you pull to the user that runs that command. Within a container, that is the root user, so `git` operations should be run outside of the container.

To fix this situation, change the owner back to yourself outside of the container by running the following command.

```
$ sudo chown <user>:<group> -R .
```

3.6.8 Running LMS commands within a container

Most of the `paver` commands require a settings flag. If the flag is omitted, the flag defaults to `devstack`, which is the settings flag for vagrant-based `devstack` instances. If you run into issues running `paver` commands in a docker container, you should append the `devstack_docker` flag. For example:

```
$ paver update_assets --settings=devstack_docker
```

3.6.9 Resource busy or locked

While running `make static` within the ecommerce container, you could get an error saying:

```
Error: Error: EBUSY: resource busy or locked, rmdir '/edx/app/ecommerce/ecommerce/
↳ecommerce/static/build/'
```

To fix this, remove the directory manually outside of the container and run the command again.

3.6.10 No space left on device

If you see the error `no space left on device` on a Mac, Docker has run out of space in its `Docker.qcow2` file.

Here is an example of this error while running `make pull`.

```
...
32d52c166025: Extracting [=====>]
1.598 GB/1.598 GB ERROR: failed to register layer: Error processing tar
file(exit status 1): write /edx/app/edxapp/edx-platform/.git/objects/pack/
pack-4ff9873be2ca8ab77d4b0b302249676a37b3cd4b.pack: no space left on device
make: *** [pull] Error 1
```

To address this error, first try the following command to clean up dangling images.

```
docker image prune -f # (This is very safe, so try this first.)
```

If you are still seeing issues, you can try cleaning up dangling volumes.

Warning: In most cases this removes only volumes that you no longer need, but this is not a guarantee.

```
docker volume prune -f # (Be careful, this will remove your persistent data!)
```

3.6.11 Memory Limit

While provisioning, some have seen the following error:

```
...
Build failed running pavelib.assets.update_assets: Subprocess return code: 137
```

This error is an indication that your docker process died during execution. Most likely, this error is due to running out of memory. Try increasing the memory allocated to Docker.

3.6.12 Docker is using lots of CPU time when it should be idle

On the Mac, this often manifests as the `hyperkit` process using a high percentage of available CPU resources. To identify the container(s) responsible for the CPU usage:

```
make stats
```

Once you've identified a container using too much CPU time, check its logs. For example, run the following command to check the LMS logs.

```
make lms-logs
```

The most common culprit is an infinite restart loop where an error during service startup causes the process to exit, but we've configured `docker-compose` to immediately try starting it again (so the container will stay running long enough for you to use a shell to investigate and fix the problem). Make sure the set of packages installed in the container matches what your current code branch expects; you may need to rerun `pip` on a requirements file or pull new container images that already have the required package versions installed.

Support

File any issues you encounter in JIRA under the [PLAT](#) (Platform) project.

Configuring the Open edX Platform

The following sections provide information about Open edX Platform configuration options.

4.1 Guidelines for Updating the Open edX Platform

When you update the Open edX Platform, you should not change configuration files on a running server. Doing so can result in unpredictable problems.

If you need to change settings on a running server, take the following steps.

1. Provision a new server that matches the running server.
2. Make configuration changes on the new server.
3. Start the new server.
4. Reroute traffic from the old server to the new server.
5. Decommission the old server.

4.2 Configuring Open edX Sites

By default, an Open edX installation has one site for users to interact with. You can configure multiple sites within your Open edX installation. A site presents Open edX courses and content in an individual way. If you set up multiple sites, you can configure them independently of each other. For example, you can assign a different theme to each site and specify which courses are available on each site.

You host each site on a separate domain or subdomain from your Open edX installation. You configure the domain name for a site in the Django admin site when you create it. For example, you might configure one site with the domain name `university.edu` and another site with the domain name `advancedplacement.edu`. Or you might configure one site with the domain name `arts.myuniversity.edu` and another site with the domain name `sciences.myuniversity.edu`.

4.2.1 Create an Open edX Site

To create an Open edX site, follow these steps.

1. Sign in to the Django administration console for your base URL. For example, `http://{your_URL}/admin`.
2. Select **Sites** to open the `http://{your_URL}/admin/sites/site` page.
3. Enter the domain name for the site. This is the domain name in the URL for the site. For example, `myuniversity.edu`.

To make a site available on a non-default port that is entered as part of the URL, the domain name must include the port number. For example, if an LMS site is available at `my-site.localhost:8000`, the domain name for the site must be `my-site.localhost:8000`.

4. Select **Save**.

4.2.2 Configuring Sites Independently

You can set configuration properties independently for individual sites. The values that you define for individual sites override the default values that are present in the `cms.env.json` or `lms.env.json` files. For example, you can set the `PLATFORM_NAME` property to a different value for each of your sites to indicate that the sites present courses for different organizations or audiences.

Configuring a Site

To set configuration properties for a site, follow these steps.

1. Sign in to the Django administration console for your base URL. For example, `http://{your_URL}/admin`.
2. Select **Site Configurations**.
3. Select **Add site configuration**.
4. From the **Site** menu, select the site you want to configure.
5. Enter configuration properties in the **Values** field. Structure all properties in valid JavaScript Object Notation (JSON) format.

The following example shows a set of configuration properties for a site.

```
{
  "course_email_from_addr": "courses@onlineu.edu",
  "university": "Online University",
  "PLATFORM_NAME": "Online University",
  "email_from_address": "courses@onlineu.edu",
  "payment_support_email": "payments@onlineu.edu",
  "SITE_NAME": "onlineu.edu",
  "site_domain": "onlineu.edu",
  "SESSION_COOKIE_DOMAIN": "onlineu.edu"
}
```

Note: To make courses site-specific, you set the `course_org_filter` property to an organization identifier. Only that organization's courses are available from the site.

6. When you are ready for the configuration settings to take effect, select **Enabled**.

The configuration properties that you set do not affect the site until you select **Enabled**. If needed, you can return to the **Site configurations** screen for this site to enable the configuration properties later.

7. Select **Save**.

Site Configuration Reference

An example of the properties that you define to configure a site follows.

```
{
  "ECOMMERCE_API_URL": "https://my-site.sandbox.edx.org/api/v2",
  "ECOMMERCE_PUBLIC_URL_ROOT": "https://my-site.sandbox.edx.org",
  "ECOMMERCE_API_SIGNING_KEY": "ecommerce-secret",
  "COURSE_CATALOG_VISIBILITY_PERMISSION": "see_in_catalog",
  "COURSE_ABOUT_VISIBILITY_PERMISSION": "see_about_page",
  "ENABLE_COMBINED_LOGIN_REGISTRATION": true,
  "ENABLE_PAID_COURSE_REGISTRATION": true,
  "course_email_template_name": "my-site",
  "course_email_from_addr": "my-site@example.com",
  "ALLOW_AUTOMATED_SIGNUPS": true,
  "domain_prefix": "my-site",
  "university": "Education Programs",
  "PLATFORM_NAME": "Education Programs",
  "platform_name": "Education Programs",
  "show_only_org_on_student_dashboard": true,
  "email_from_address": "my-site@example.com",
  "payment_support_email": "payments@example.com",
  "SITE_NAME": "my-site.sandbox.edx.org",
  "site_domain": "my-site.sandbox.edx.org",
  "SESSION_COOKIE_DOMAIN": "my-site.sandbox.edx.org",
  "course_org_filter": "MyOrgX",
  "course_index_overlay_text": "<img src='/static/my-site/images/400x103.png' width=
↵ '400' height='103' />",
  "homepage_overlay_html": "<img src='/static/my-site/images/400x103.png' width='400'
↵ height='103' />",
  "payment_email_signature": "Education Programs<br>The Digital Programs Team<br>my-
↵ site@example.com<br>101 Example Street<br>Example State"
}
```

For more information about themes, see [Changing Themes for an Open edX Site](#).

4.3 Changing the Appearance of Open edX Sites

This section describes how you can customize your Open edX sites to change the way they look.

4.3.1 Changing Themes for an Open edX Site

The theme for a website defines the appearance of its user interface (UI): the logo, the color scheme, and the links in the page headers and footers are examples of different aspects of an Open edX site that are defined by its theme.

Open edX provides a default theme that is defined by page templates, CSS styling, and assets such as images that are provided in the Open edX code. You can change the appearance of the following parts of an Open edX site.

- The Studio UI, which is used by course teams.
- The learning management system (LMS) UI, which is used by learners and course teams.
- The UI of the E-commerce service, which is used by course offering and order managers.

The topics in this section describe how you can change the way an Open edX site looks, without changing how it works.

Themes Overview

After you configure one or more sites for your Open edX installation, including their domain and platform names, you can set up a theme to use for each site.

To override the files that constitute the default Open edX theme, you create replacements for one or more of those files, place them in file paths that are constructed and named in parallel to the default file locations, configure your Open edX instance to use the files in your theme's directories instead of the default locations, and then compile the theme.

EdX first looks for files in your theme directories, and uses any file that matches the exact file path and file name of a default UI file. If matching files are not found, then Open edX looks in the default location.

For more information about Open edX sites, see [Configuring Open edX Sites](#).

Root Directories for Theme Files

Themes are located outside of the Open edX source directories, in any location you like. For example, you might make a directory called `/my-open-edx-themes`.

Within that directory, you create a separate directory for each Open edX repository that you want to create a theme for, such as `edx-platform` and `ecommerce`.

Within each of those directories, you create another directory and name it for your theme, such as `my-theme`. You can create a number of themes, each with their own name. Within each theme directory, you create directories and files to parallel the structure in the corresponding Open edX repository.

After you create these directories, you might have a structure like this one.

```
my-open-edx-themes
├── ecommerce
│   └── my-theme
├── edx-platform
│   └── my-theme
│       ├── cms
│       └── lms
```

You must give the files that you create for a theme the same relative file paths and file names as the default files that they override. Different root directories for the relative paths apply to Studio, the LMS, and the E-commerce service.

- For Studio and the LMS, relative file paths are from the root directory of the local clone of the `edx/edx-platform` repository in your installation directory.
- For the E-commerce service, relative file paths are from the `ecommerce` directory of the local clone of the `edx/ecommerce` repository in your installation directory.

For example, the root directory for the relative file paths of your theme files might be at one of the following file paths.

- For the LMS UI or Studio UI, `/edx/app/edxapp/edx-platform`.
- For the UI of the E-commerce service, `/edx/app/ecommerce/ecommerce/ecommerce`.

The following subdirectories hold the UI files that you can override.

- `static` holds media files such as images and styling resources such as Syntactically Awesome Style Sheet (Sass) files that produce Cascading Style Sheet (CSS) files.
- `templates` holds Python web application page templates that produce the HTML for UI pages.

Creating a Theme

To create a theme, you add a theme directory to the installation-wide themes directory that you added when you enabled theming for your Open edX installation. Then, you copy default UI files into your theme directory and update them to change the appearance of the Open edX site or sites that will use that theme.

- [Understanding Which UI Files to Customize](#)
- [Example File Path for a Theme File](#)
- [Naming a Theme Directory](#)

For more information about the themes directory, see [Enabling and Applying Themes](#).

Understanding Which UI Files to Customize

You can customize the default images, Sass files, and web application template files for the Open edX components in your installation.

- To replace an image, you can override any of the images in the `static/images` directories for the components you are theming.
- For the LMS and Studio, you can customize any of the Sass files in the `static/sass` directories.
- For the E-commerce service, you can customize any of the Sass files in the `static/sass/partials` directory.

Note: Do not customize Sass files in the `static/sass/base` directory.

- You can override any of the HTML templates in the `lms/templates` or `cms/templates` directories for the components that you want to apply a theme to.

UI files for the LMS are stored in the directories shown in the following example theme directory. You can examine the default UI files in the source repository of the component that you want to apply the theme to.

```
/my-open-edx-themes/edx-platform/my-theme/lms/static/images  
/my-open-edx-themes/edx-platform/my-theme/lms/static/sass  
/my-open-edx-themes/edx-platform/my-theme/lms/templates
```

Example File Path for a Theme File

The default Open edX theme includes an image file named `logo.png` that appears in the header of most LMS pages. The file path of that image in the `edx-platform` repository is `lms/static/images/logo.png`.

The following example shows an absolute file path of the LMS logo image in a theme directory. The file path after `/my-open-edx-themes/edx-platform/my-theme/` matches the relative file path of that image in the default directory for the LMS UI.

```
/my-open-edx-themes/edx-platform/my-theme/lms/static/images/logo.png
```

Naming a Theme Directory

The name of the directory that you create to hold your versions of the image, theme, and Sass styling files identifies the theme. As a result, if you want to create a theme named `my-theme`, the name of the directory within your installation-wide themes directory must be `my-theme`.

Note: If you add more than one site-specific theme directory, make sure that each of the directory names is unique. The Open edX installation looks for a theme with a certain name, and selects the first one that matches.

Because the UI files for the LMS and Studio are located together in the `edx-platform` repository, you can create one theme that applies to both the LMS and Studio. If you want to create a theme for the E-commerce service, you must add a separate theme directory for its files.

The theme directory holds the UI files that override the corresponding default files.

For example, if the themes directory for your site is `/my-open-edx-themes`, the files in the following example create a theme named `my-theme`.

```
/my-open-edx-themes/edx-platform/my-theme/lms/static/images/logo.png
/my-open-edx-themes/edx-platform/my-theme/lms/static/sass/partial/base/_variables.
↪ scss
/my-open-edx-themes/edx-platform/my-theme/lms/templates/navigation.html
/my-open-edx-themes/edx-platform/my-theme/cms/static/images/studio-logo.png
/my-open-edx-themes/edx-platform/my-theme/cms/static/images/logo.png
/my-open-edx-themes/edx-platform/my-theme/cms/templates/login.html
```

Because the theme directory includes UI files in both the `lms` and `cms` subdirectories, you can apply the theme to both the LMS and Studio.

Note: After you create or make changes to a theme, you must update the theme. Updating a theme compiles Sass files to create the CSS files that style your UI. For more information, see [Compiling a Theme](#).

Enabling and Applying Themes

You must enable the use of themes for your Open edX installation before you can apply themes to your Open edX sites. If your installation has only one site, you apply a theme to that default site.

For more information about Open edX sites, see [Configuring Open edX Sites](#).

- [Enable Themes](#)
- [Example Settings for Comprehensive Theme](#)
- [Apply a Theme to a Site](#)

Enable Themes

To enable the use of themes for your Open edX installation, follow these steps.

1. Create an installation-wide themes directory to hold the customized UI files for all of the themes that you create. This directory will hold subdirectories for each theme. Your Open edX installation will look in the directory to find the themes you apply to sites.

You can create this directory at any location on a file system that is accessible to your Open edX installation. For example, you might place it at the root of the file system in a directory named `/my-open-edx-themes`.

2. Set the file permissions on the themes directory, and all of its subdirectories, to enable read+write permissions for the Ubuntu user.

For example, to allow the devstack `edxapp` user for the LMS and Studio to read and write files in the themes directory, you might use the following commands.

```
sudo chown -R edxapp:edxapp /my-open-edx-themes
sudo chmod -R u+rw /my-open-edx-themes
```

On fullstack and native installations the Ubuntu user is `www-data`.

3. For each Open edX component that you want to theme, set the `ENABLE_COMPREHENSIVE_THEMING` configuration property to `true`.

The specific method that you use to configure Open edX components depends on the type of environment you are using. For example, you can set the configuration property in the following files.

- For the LMS, you edit `/edx/app/edxapp/lms.env.json` to set `"ENABLE_COMPREHENSIVE_THEMING": true`.
- For Studio, you edit `/edx/app/edxapp/cms.env.json` to set `"ENABLE_COMPREHENSIVE_THEMING": true`.
- For the E-commerce service, you edit `/edx/etc/ecommerce.yml` to set `ENABLE_COMPREHENSIVE_THEMING: true`.

If any of these files do not exist, you can add them to define this configuration setting.

4. For each Open edX component that you want to apply a theme to, add the absolute path of the themes directory to the `COMPREHENSIVE_THEME_DIRS` configuration property.

The specific method that you use to configure Open edX components depends on the type of environment you are using. For example, you can set the configuration property in the following files.

- For Studio, add the path to `COMPREHENSIVE_THEME_DIRS` in `/edx/app/edxapp/cms.env.json`.

```
"COMPREHENSIVE_THEME_DIRS": [
  "/my-open-edx-themes/edx-platform"
],
```

- For the LMS, add the path to `COMPREHENSIVE_THEME_DIRS` in `/edx/app/edxapp/lms.env.json`.

```
"COMPREHENSIVE_THEME_DIRS": [
  "/my-open-edx-themes/edx-platform"
],
```

- For the E-commerce service, add the path to `COMPREHENSIVE_THEME_DIRS` in `/edx/etc/ecommerce.yml`.

```
COMPREHENSIVE_THEME_DIRS: ["/my-open-edx-themes/ecommerce"]
```

5. Restart all servers.

Note: You can create more than one themes directory for your Open edX installation. To use multiple themes directories, include the path to each directory in the `COMPREHENSIVE_THEME_DIRS` configuration property. The following example shows the configuration for multiple themes directories.

```
"COMPREHENSIVE_THEME_DIRS": [  
  "/my-open-edx-themes/edx-platform",  
  "/my-other-open-edx-themes/edx-platform"  
],
```

Example Settings for Comprehensive Theme

For the following file structure:

```
edx  
├── my-themes  
│   ├── my-theme-red  
│   │   ├── cms  
│   │   ├── lms  
│   │   │   ├── static  
│   │   │   └── templates
```

set these in `lms.env.json` and `cms.env.json`:

```
"COMPREHENSIVE_THEME_DIRS": [  
  "/edx/my-themes",  
],  
"THEME_NAME": "my-theme-red"
```

Apply a Theme to a Site

To apply a theme to an Open edX site, follow these steps.

1. Make sure that you have enabled theming for your Open edX installation and that you have configured an installation-wide themes directory. For more information, see [Enabling and Applying Themes](#).
2. Make sure that you have created a theme and that you know the identifier of the theme. The identifier of a theme is the name of the directory for that theme, within your installation-wide themes directory. For more information, see [Creating a Theme](#).
3. Sign in to the Django administration console for your base URL. For example, `http://{your_URL}/admin`.
4. Select **Site themes**.
5. Select **Add site theme**.
6. From the **Site** menu, select the site you want to apply a theme to.

7. In the **Theme dir name** field, enter the identifier of the theme.
8. Select **Save**.

Compiling a Theme

To update a theme, you compile the Sass files to create the CSS files that style your UI when you apply the theme.

- *Update a Theme for the LMS or Studio*
- *Update a Theme for the E-commerce Service*

Update a Theme for the LMS or Studio

To update a theme for Studio or the LMS, follow these steps.

1. Log in to the Open edX machine as the `edxapp` user.
2. Change to the `/edx/app/edxapp/edx-platform` directory.
3. Execute the `paver update_assets` command to update all themes.

If you want to update specific themes, use the arguments described in the following table.

Argument	Description
<code>--theme-dirs</code>	Provide a space-separated list of the theme directories that you want to update. Only files in the theme directories that you include are updated.
<code>--themes</code>	Provide a space-separated list of the themes that you want to update. Only the themes that you include are updated.

Update a Theme for the E-commerce Service

For the E-commerce service, commands are available for you to update all themes at once, or to update only the themes you specify.

To update a theme for the E-commerce service, follow these steps.

1. Log in to the server for the E-commerce service as the `ecommerce` user.
2. Change to the `/edx/app/ecommerce/ecommerce` directory.
3. To update all themes, execute one of these commands.
 - `make migrate`
 - `python manage.py update_assets`
4. To specify a theme or set of themes to update, or to include optional arguments, execute `python manage.py update_assets` with the options described in the following table.

Argument	Description
<code>--settings</code>	Provide the name of a Django settings module in Python package syntax. For example, <code>--settings=ecommerce.settings.production</code> .
<code>--themes</code>	Provide a space-separated list of the themes that you want to update. Only the themes that you include are updated.
<code>--output-style</code>	Defines the coding style for the compiled CSS files. Possible values are <code>nested</code> , <code>expanded</code> , <code>compact</code> , and <code>compressed</code> . The default value is <code>nested</code> .
<code>--skip-system</code>	Disables Sass file compilation for the default Sass files provided in the Open edX software. Use this option if you have only updated the Sass files in your theme.
<code>--skip-collect</code>	Only compile the Sass files and do not deploy the resulting CSS files.
<code>--enable-source-comments</code>	Include the location of the source file as comments in the resulting CSS files. Enabling this argument can be useful when you are testing a theme.

In addition, an [example theme](#) is available for review.

4.3.2 Styling Drag and Drop Problems

You can customize the appearance of drag and drop problems in your Open edX site using custom Cascading Style Sheet (CSS) files. The style that you configure applies to all drag and drop problems in all courses. For more information about drag and drop problems, see [Drag and Drop Problem](#).

The following two methods apply CSS styling to drag and drop problems.

- You can apply CSS styles to drag and drop problems by creating a theme for your site and updating the Synchronously Awesome Style Sheet (Sass) files that produce the CSS files. For more information, see [Creating a Theme](#).
- You can apply CSS styles to drag and drop problems by adding a Python programming language module that includes a CSS file to your Open edX site. For more information, see the instructions in this section.

Note: This section provides information about styling the drag and drop problem type that was added to the edX platform in 2016. This drag and drop problem type replaced an earlier drag and drop problem type. You should use the latest drag and drop problem type for all new course problems.

Note: Course teams can also style the background and text colors of the draggable items in an individual drag and drop problem, without adding CSS files or configuring a Python module. For more information, see [Changing the Visual Style of a Drag and Drop Problem](#).

To customize the style of drag and drop problems by adding a CSS file in a Python module, follow these steps.

1. Create a custom CSS style sheet that applies styles to the drag and drop problem user interface. You can base your style sheet on the [example CSS file for drag and drop problems](#) that is included in the drag and drop problem module.
2. Create a Python module that includes your custom CSS style sheet. For example, the following Python module files include a CSS style sheet.

```
./my_drag_and_drop_style
./my_drag_and_drop_style/css
./my_drag_and_drop_style/css/my_drag_and_drop_style.css
```



```
./my_drag_and_drop_style/__init__.py
./setup.py
```

For more information about creating and installing Python modules, see the documentation for the Python programming language.

3. Install your Python module on the LMS server as the `edxapp` user.

```
pip install /path/to/my/module
```

4. Edit the `lms.env.json` file for your LMS server. Add the `drag-and-drop-v2` object to the `XBLOCK_SETTINGS` object. Include the content shown in the following example.

```
"XBLOCK_SETTINGS":{
  "drag-and-drop-v2": {
    "theme": {
      "package": "my_drag_and_drop_style",
      "locations": ["css/my_drag_and_drop_style.css"]
    }
  }
}
```

Enter the name of your Python module in the value of the `package` object. The value in the example above is `my_drag_and_drop_style`.

Enter the path to your CSS style sheet file in the value of the `locations` object. The value in the example above is `css/my_drag_and_drop_style.css`. The path must be relative to your Python module installation directory. You can include more than one path in the `locations` array, separated by commas.

5. Restart the LMS.

4.4 Adding Custom Fields to the Registration Page

This topic describes how to add custom fields to the registration page for your instance of Open edX.

- [Overview](#)
- [Add Custom Fields to the Registration Page](#)

4.4.1 Overview

By default, the registration page for each instance of Open edX has fields that ask for information such as a user's name, country, and highest level of education completed. You can add custom fields to the registration page for your own Open edX instance. These fields can be different types, including text entry fields and drop-down lists.

4.4.2 Add Custom Fields to the Registration Page

Before you add a custom field to the registration page, you must make sure that the combined sign-in and registration form is enabled for your Open edX instance. To do this, open the `lms.env.json` and `cms.env.json` files, and set the `ENABLE_COMBINED_LOGIN_REGISTRATION` feature flag to `True`. These files are located one level above the `edx-platform` directory.

To add custom fields to the registration page, follow these steps.

1. Start and sign in to your instance of Open edX.
2. Use Python to create a Django form that contains the fields that you want to add to the page, and then create a Django model to store the information from the form.

For more information about how to create Django forms, see [Django Forms on the Django website](#).

3. In the `lms.env.json` file, add the app for your model to the `ADDL_INSTALLED_APPS` array.
4. In the `lms.env.json` file, set the `REGISTRATION_EXTENSION_FORM` setting to the path of the Django form that you just created, as a dot-separated Python string.

For example, if your form is named “ExampleExtensionForm” and is located at “path/to/the_form.py”, the value of the setting is `path.to.the_form.ExampleExtensionForm`.

5. Run database migrations.
6. Restart the LMS and verify that the new fields appear on your registration form.

Note: For an example app for custom forms, see the OpenCraft [custom_form_app](#) page in [GitHub](#).

4.5 Specifying Allowed Registration Email Patterns

This topic describes how to restrict registration on your site by specifying which email address patterns are allowed in registration emails.

- [Overview](#)
- [Configure Allowed Email Patterns](#)

4.5.1 Overview

By default, all email addresses are accepted when learners register for an account on your Open edX site. You have the option of restricting registrations to learners who use an allowed email address pattern. Doing so can be useful in cases where you want to allow only learners who are members of a school, organization, or corporation to register and access your courses.

Note: Configuring your site using the procedure below only restricts registration to learners whose email addresses match the specified patterns. It does not hide courses from any learners, or prevent access to pages on your site that can be accessed without registration.

4.5.2 Configure Allowed Email Patterns

To specify the email patterns that are allowed for registration, follow these steps.

1. Locate the `lms.env.json` and `cms.env.json` files, which are located one level above the `edx-platform` directory. You make the same changes to both files.
2. In the `lms.env.json` and `cms.env.json` files add the `REGISTRATION_EMAIL_PATTERNS_ALLOWED` setting.

```
"REGISTRATION_EMAIL_PATTERNS_ALLOWED": null
```

If the value for this setting is `null`, there are no restrictions, and all email addresses are accepted for registration.

3. Use one or more Python regular expressions to specify the email domains that allowed email addresses must match.

The following example allows email addresses using the pattern `example.com` or `any.example.com` to register. It also allows `school.tld` addresses, but only if those addresses have a `.` before the `@` symbol.

```
"REGISTRATION_EMAIL_PATTERNS_ALLOWED" = [  
    "^.*@(.*\\.\\.)?example\\.\\.com$",  
    "^(\\w+\\.\\.\\w+)@school\\.\\.tld$" ]
```

4. Save the `lms.env.json` and `cms.env.json` files.
5. Restart your `edxapp` instances.

4.6 Adding the CourseTalk Widget

This topic describes how to add [CourseTalk](#) widgets to your instance of Open edX. When you add the CourseTalk widget, it is visible for all courses.

- [Overview](#)
- [Add the CourseTalk Widget](#)

4.6.1 Overview

The CourseTalk widget allows learners to write reviews of your course and see reviews that other learners have written. Learners can write reviews on the course **Course** page, and the reviews are visible both in the course, and on the course **About** page. Only learners who are enrolled in the course can leave reviews of a course.



4.6.2 Add the CourseTalk Widget

To add the CourseTalk widget, follow these steps.

1. Sign in to the Django administration console for your base URL. For example, go to `http://{your_URL}/admin`.
2. In the navigation pane, locate **Coursetalk**, and then select **Course talk widget configurations**.
3. On the **Select course talk widget configuration to change** page, select **Add course talk widget configuration**.
4. On the **Add course talk widget configuration** page, select the **Enabled** check box, enter a value in the **Platform key** field, and then select **Save**.

Note: You can use any text that you want as your platform key. EdX recommends that you use your domain name, or part of your domain name.

5. In the LMS, open the **Course** page for any of your Open edX courses. Verify that the **Reviews** link in the sidebar opens the CourseTalk widget.
6. Sign out of your instance of Open edX and view the About page for any course.
7. In the right pane, verify that the CourseTalk widget appears below the course information panel.

4.7 Enabling Open edX Search

You can add a search feature to your Open edX site so that prospective learners can find courses more easily. When a learner searches for a key word or words, the search feature returns a list of the courses that are currently open for enrollment and that match the entered key words.

This section describes how to enable search in your instance of Open edX.

- [Overview](#)
- [Search Engines and edX Search](#)
- [EdX Search Requirements](#)
- [Install edX Search](#)
- [Enable Indexing](#)
- [Supported Flags](#)

4.7.1 Overview

EdX Search is a Django application that provides access to search services from within edX Platform applications. Searching is accomplished by creating an index of documents, and then searching within that index for matching information.

When you install the Open edX devstack, edX search is enabled by default. You must enable this feature to use it with Open edX fullstack.

Note: Before proceeding, review [Guidelines for Updating the Open edX Platform](#).

4.7.2 Search Engines and edX Search

By default, edX Search uses MockSearchEngine for testing and ElasticSearch Engine for production. You can configure edX Search to use a different search engine.

MockSearchEngine

MockSearchEngine is a simple implementation using a JSON file for index storage. It has no specific requirements, but it does not scale well and should only be used for testing.

ElasticSearchEngine

ElasticSearchEngine is a ElasticSearch back-end implementation. It uses same ElasticSearch version that is already part of Open edX Platform. The current version is v1.5.2. The ElasticSearch Python client is the latest 1.x version.

4.7.3 EdX Search Requirements

EdX Search requires the following applications.

- Django (edX Platform version)
- pyMongo (edX Platform version)
- pytz
- Django elasticsearch (0.4.5)

4.7.4 Install edX Search

EdX Search is included in Open edX Platform GitHub requirements and is installed automatically when you install the Open edX Platform.

4.7.5 Enable Indexing

You enable course indexing by setting the `ENABLE_COURSEWARE_INDEX` flag.

You enable library indexing by setting the `ENABLE_LIBRARY_INDEX` flag.

Indexing is done from Studio as a Celery task. Every publish event triggers the reindex procedure.

You can also reindex the course manually through the **Reindex** button in the **Course Overview** page.

Note: The search feature only returns results for courses that are currently open for enrollment. Course teams can set the enrollment start and end dates for their courses in Studio. For more information, see [Guidelines for Start and End Dates](#) in the *Building and Running an Open edX Course* guide.

Which Data Gets Indexed

Which data gets indexed is determined by the module `index_dictionary()` function implementation. Modules supporting this method are `Sequence`, `Vertical`, `Video`, and `HTML Block`. You can add support to any module type.

Course metadata, including the name, description, and start and end dates are also indexed.

4.7.6 Supported Flags

The following flags are supported in the CMS and LMS applications.

CMS

- `ENABLE_COURSEWARE_INDEX`: Enables and disables courseware content and course info indexing.
- `ENABLE_LIBRARY_INDEX`: Enables and disables library content indexing.
- `SEARCH_ENGINE`: Sets the search engine to use. There are two predefined values.
 - `"search.elastic.ElasticSearchEngine"`
 - `"search.tests.mock_search_engine.MockSearchEngine"`
- `ELASTIC_FIELD_MAPPINGS`: Sets any additional field mappings that elastic search should be aware of. For example, the following code includes the course start date.

```
ELASTIC_FIELD_MAPPINGS = {
  "start_date": {
    "type": "date"
  }
}
```

LMS

- `ENABLE_COURSEWARE_SEARCH`: Enables and disables Courseware Search feature (in course searching).
- `ENABLE_DASHBOARD_SEARCH`: Enables and disables Dashboard Search feature (in enrolled courses searching).
- `ENABLE_COURSE_DISCOVERY`: Enables and disables Course Discovery feature (over courses searching and facet filtering).
- `COURSE_DISCOVERY_FILTERS`: If provided, overrides the list of facets that are used in the Course Discovery feature to filter the results. By default, all facets will be displayed. The list of available facets includes:
 - Course organization: "org"
 - Course type: "modes"
 - Course language: "language"
- `SEARCH_ENGINE`: Sets the search engine to use. The following values are predefined.
 - "search.elastic.ElasticSearchEngine"
 - "search.tests.mock_search_engine.MockSearchEngine"
- `SEARCH_INITIALIZER`: Used to set custom SearchInitializer. SearchInitializer provides an extension to achieve masquerade and other presearch environmental settings.
 - default: SearchInitializer
 - LMS implementation: `lms.lib.courseware_search.lms_search_initializer.LmsSearchInitializer`
- `SEARCH_RESULT_PROCESSOR`: Used to set custom SearchResultProcessor. SearchResultProcessor does post processing and data manipulation on a result set returned by SearchEngine.
 - default: SearchResultProcessor
 - LMS implementation: `lms.lib.courseware_search.lms_result_processor.LmsSearchResultProcessor`
- `SEARCH_FILTER_GENERATOR`: Used to set custom SearchFilterGenerator. SearchFilterGenerator sets filters defined by current active user. Basic implementation sets only course start date filter.
 - default: SearchFilterGenerator
 - LMS implementation: `lms.lib.courseware_search.lms_filter_generator.LmsSearchFilterGenerator`

4.8 Enabling Badging

This topic describes how to enable and configure badging in your instance of Open edX.

Note: Before proceeding, make sure you have read *Guidelines for Updating the Open edX Platform*.

- [Overview](#)
- [Prerequisites](#)

- *Enable Badges in Studio and the Learning Management System*
- *Enable Badges Within Each Course*
- *Configure Badges for Your Open edX Instance*
- *Create Course Completion Badges for Your Open edX Instance*
- *Create Course Event Badges for Your Open edX Instance*
- *Creating New Badges for Your Open edX Instance*

4.8.1 Overview

Badges provide a way for learners to share their course achievements. For courses that have course completion badges enabled, learners receive a badge at the same time that they receive a course certificate, and have the option of sharing their badges to a badging site such as Mozilla Backpack. For more information, see *Course Completion Badges*.

In addition to badges that are awarded for completing single courses, you can also issue badges for various cross-course events. For example, you can create badges to be awarded when any of the following events occur.

- A learner enrolls in a certain number of courses.
- A learner receives a completion certificate for a certain number of courses.
- A learner receives a completion certificate for every course in a specified list of courses.

For more information, see *Course Event Badges*.

By default, Open edX supports Open Badges (<http://openbadges.org/>), an open standard originally developed by the Mozilla Foundation. Open Badges provides a badge generator called Badgr Server, which is used by default in Open edX.

You can use a badge generator other than Badgr Server. For information, see *Specify a Badge Generator Other Than Badgr Server*.

For information about creating custom badges, see *Create New Badges for Your Open edX Instance*.

4.8.2 Prerequisites

Setting up the badges feature on your instance of Open edX involves performing the set up and configuration tasks that are described in the topics below.

- *Make Sure Certificates Are Enabled*
- *Specify a Badge Generator*

Make Sure Certificates Are Enabled

Badge generation depends on certificate generation. Badges for course completion are automatically generated when a course certificate is generated for a learner. Make sure certificates are enabled on your Open edX instance. For information, see *Enabling Course Certificates*.

Specify a Badge Generator

By default, Open edX uses Badgr Server as the badge generator. If you do not use Badgr Server, you can configure a different badge generator.

- To use Badgr Server, see *Install Badgr Server* and *Specify a Badge Issuer for Your Organization*.
- To use a different badge generator, see *Specify a Badge Generator Other Than Badgr Server*.

Install Badgr Server

Badgr Server provides an API for issuing Open Badges. Follow the instructions at <https://github.com/concentricsky/badgr-server> to install and run Badgr Server.

Important: You must install Badgr Server at a publicly accessible IP address, to allow the Open edX LMS and services such as Mozilla Backpack to contact Badgr Server.

If you do not use Badgr Server, you can configure a different badge generator. See *Specify a Badge Generator Other Than Badgr Server*.

Specify a Badge Issuer for Your Organization

Note: This step is required only if you use Badgr Server for your badge generator.

If you are using Badgr Server, log in to your installation of Badgr Server and add an issuer of Open Badges for your organization.

For more information about issuing Open Badges, see the [Issuing Badges](#) topic on the Mozilla wiki.

Specify a Badge Generator Other Than Badgr Server

By default, Open edX uses Badgr Server as the badge generator. To use Badgr Server, see *Install Badgr Server* and *Specify a Badge Issuer for Your Organization*.

To specify a different badge generator, follow these steps.

Note: Because the services and software used for badge generation can differ significantly, the steps described in this topic are intended as guidelines and not as exact procedures.

1. Add the `BadgingBackend` object as a subclass to `lms/djangoapps/badges/backends/base.py`.

A `BadgingBackend` object must include the `award` function

```
award(badge_class, user_id, evidence_url=None)
```

where

- `badge_class` is the definition of the badge class for which the `BadgeAssertion` is created,
- `user_id` is an ID for the user that the `BadgeAssertion` is to be created for, and
- `evidence_url` is an optional URL for the location where the badge evidence can be viewed.

The `award` function is responsible for the server awarding a badge to a learner. It is also responsible for all checks leading up to the awarding of the badge, including making sure the badge specification does not already exist on the target before creating a new badge specification.

The `award` function should either return a `BadgeAssertion` object or raise an exception if the award cannot be given. By default, a base badge generation object called `BadgeBackend` exists, and contains a dummy version of the `award` function.

```
class BadgeBackend(object):
    """
    Defines the interface for badge generators.
    """
    __metaclass__ = ABCMeta

    @abstractmethod
    def award(self, badge_class, user, evidence_url=None):
        """
        Create a badge assertion for the user using this badge generator.
        """
```

2. In the `BadgeAssertion` object that is generated by the `award` function, make sure the `backend` value is the name of the `BadgingBackend` subclass.

```
class BadgeAssertion(models.Model):
    """
    Tracks badges on our side of the badge baking transaction
    """
    user = models.ForeignKey(User)
    badge_class = models.ForeignKey(BadgeClass)
    data = JSONField()
    backend = models.CharField(max_length=50)
    image_url = models.URLField()
    assertion_url = models.URLField()
```

3. In the `lms.env.json` and `cms.env.json` files, set the value of `BADGING_BACKEND` as a dot-separated python path specification to the object that you use to create and assign badges.
4. When you have finished modifying your configuration files for the badge generator, restart the Studio and Learning Management System processes so that the updated environment configurations are loaded.

4.8.3 Enable Badges in Studio and the Learning Management System

To enable badges, you modify the `lms.env.json` and `cms.env.json` files, which are located one level above the `edx-platform` directory.

1. In the `lms.env.json` and `cms.env.json` files, in the `FEATURES` section, set the value of `ENABLE_OPENBADGES` to `True`.

```
# Enable OpenBadge support.
'ENABLE_OPENBADGES': True,
```

2. In `lms.env.json`, set the values for the following parameters.
 - `BADGR_API_TOKEN`: A string containing the API token for the Badgr superuser account. Obtain the token from the `/v1/user/auth-token` page while logged in to the API as the superuser.
 - `BADGR_BASE_URL`: A string containing the base URL for Badgr Server. The Badgr Server must be installed at a publicly accessible IP address.

- `BADGR_ISSUER_SLUG`: A string that is the slug for the Badgr issuer. The slug can be obtained from the URL of the Badgr Server page that displays the issuer. For example, in the URL `http://exampleserver.com/issuer/test-issuer`, the issuer slug is `test-issuer`.

```
##### Badgr OpenBadges generation #####  
  
BADGR_API_TOKEN = None  
# Do not add the trailing slash here.  
BADGR_BASE_URL = "http://localhost:8005"  
BADGR_ISSUER_SLUG = "test-issuer"
```

3. Save the `lms.env.json` and `cms.env.json` files.
4. Run database migrations.
5. Restart the Studio and Learning Management System processes so that the updated environment configurations are loaded.

4.8.4 Enable Badges Within Each Course

The ability to issue course completion badges is enabled by default for all courses, but can be turned off or on again using an advanced setting in Studio. For information, see [Enable or Disable Badges for Your Course](#) in *Building and Running an Open edX Course*.

Note: The course-level setting to enable badges does not affect badges that are issued for completing or enrolling in multiple courses.

4.8.5 Configure Badges for Your Open edX Instance

You can configure several types of badges to issue to learners in your Open edX instance.

- *Course Completion Badges*
- *Course Event Badges*

In addition, you can use the badging framework to create custom badges. For information, see [Creating New Badges for Your Open edX Instance](#).

Course Completion Badges

For courses that have badges enabled, learners receive a course completion badge at the same time as they receive a course certificate, and have the option of sharing their badges to a badging site such as Mozilla Backpack.

You can configure a different course completion badge for each course mode that you support (for example, “professional”, “advanced”, or “basic”). You can also specify a different badge image for each of the badges that you configure. For information, see [Enable Badges Within Each Course](#) and [Create Course Completion Badges for Your Open edX Instance](#).

Course Event Badges

Course event badges are awarded across courses, and can be awarded when any of the following events occur.

- A learner enrolls in a certain number of courses.
- A learner receives a completion certificate for a certain number of courses.
- A learner receives a completion certificate for every course in a specified list of courses.

You can customize these course event badges with your parameters and badge images. For information, see *Create Course Event Badges for Your Open edX Instance*.

4.8.6 Create Course Completion Badges for Your Open edX Instance

Important: Default images are supplied in Open edX for course completion badges. Be sure to replace these default badge images with your organization’s own badge images before any badges are issued. When the first badge is issued for a given course, badge images are uploaded to Badgr Server. All badges issued in future for this course will use this uploaded original badge image, even if you subsequently change badge images in the Django Administration badge image configuration.

1. Access the Django Administration website for your instance of Open edX. To do this, go to `https://<host name of your Open edX instance>/admin`. For example, this might be `https://YourOrganization.org/admin`.
2. Select **Site Administration > Badges > Course complete image configurations**, and then define a course complete image configuration for each course mode on your platform for which you want to issue badges upon course completion. Examples of course modes are “professional” or “advanced” or “basic”.
3. For each course complete badge image configuration, set these parameters.
 - Mode: The course mode for which the badge image should be used.
 - Icon: The badge image to use for the specified course mode.

Important: Be sure to replace the default badge images with your organization’s own badge images before any badges are issued.

4. Optionally, you can define a badge image that will be used as the default badge image for any course modes that do not have an explicitly specified badge image.

To do so, in the course complete image configuration that references the image you want to use as a default, select the **Default** checkbox. After you save the configuration, this badge image is used for any course completion badge configurations that do not have a badge image explicitly specified.

Note: You can specify only one default badge image.

5. Save each configuration parameter and exit the Django Administration website.

4.8.7 Create Course Event Badges for Your Open edX Instance

Open edX provides several customizable course event badges that can be awarded when any of the following events occur.

- A learner enrolls in a certain number of courses.
- A learner receives a completion certificate for a certain number of courses.
- A learner receives a completion certificate for every course in a specified list of courses.

Before course event badges can be awarded, you must customize them with your parameters and badge images. To customize any of the course event badges, follow these steps.

Note: You can also use the badging framework to create custom badges. For information, see [Creating New Badges for Your Open edX Instance](#).

1. Access the Django Administration website for your instance of Open edX. To do this, go to `https://<host name of your Open edX instance>/admin`. For example, this might be `https://YourOrganization.org/admin`
2. Select **Site Administration > Badges > Badge Classes**.
3. Add a badge class for each course event for which you want to issue badges. Examples of course events might be enrolling in five courses, or completing three required courses.
4. For each badge class, set the following parameters.
 - Slug: A unique identifier that you choose to identify the badge class. This identifier can contain only numbers, lowercase letters, underscores, or hyphens. The slug, combined with the Issuing Component value, uniquely identifies a badge.
 - Issuing Component: Identifies the part of the platform that is issuing the badge. This identifier can contain only numbers, lowercase letters, underscores, or hyphens. For the three customizable course event badges that are included in the Open edX platform, the value for **Issuing Component** must be `openedx__course` (with two underscores). For *course completion badges* that are included in the Open edX platform, the issuing component value should be empty.

For new badge types that you create, specify an **Issuing Component** value that identifies the software component responsible for issuing the badge. For example, if badges are issued by the course management component, you might define **Issuing Component** as `platform__course`; if badges are issued based on activity in course discussions, you might define **Issuing Component** as `platform__discussions`.

- Display name: The human readable badge name that is used when badges are shown to learners, for example, in the Accomplishments view of learners' profiles.
- Course ID: This value should be blank for course event type badges, as they are not associated with a single course.
- Description: A description of this badge.
- Criteria: A description of the criteria for awarding this badge.
- Mode: The course mode for the course associated with this badge, if applicable.
- Image: The badge image to use for this badge. Badge images should be square .png files less than 250KB in size.

An example of a badge class configuration might have the following values.

- slug: `enrolled_three`
- issuing_component: `openedx__course`
- display_name: `Enroll in Three Courses`
- description: `Enrolled in three courses`

- `criteria`: A learner must enroll in three courses to receive this badge
 - `image`: `triple_enrollment_badge_image.png`
5. When you have finished defining the badge class, select **Save**.
 6. Next, you create a new course event badge configuration that defines all of the course event badges that you want to issue. Select **Site Administration > Badges > Course event badges configurations > Add course event badges configuration**.

Important: You can create more than one course event badge configuration, but you can only mark one configuration as **Enabled**. Only the most recently activated course event badge configuration is used.

7. Within the new course event badge configuration, set the following parameters.
 - **Courses completed:** Define badges to be awarded for completing a certain number of courses, or completion of specific courses. Define one badge per line. On each line, enter the number of courses that must be completed, followed by a comma and then the slug of the badge class to associate with this badge.

For example, to configure two badges, one that is awarded when a learner completes 3 courses, and another that is awarded when a learner completes 8 courses, you add two lines to the **Courses completed** field.

`3, completed_three 8, completed_eight`

where `completed_three` and `completed_eight` are badge slugs that you previously defined in badge classes.
 - **Courses enrolled:** Define badges to be awarded for enrolling in a certain number of courses, or enrolling in specific courses. Define one badge per line. On each line, enter the number of courses that must be enrolled in, followed by a comma and then the slug of the badge class to associate with this badge.

For example, to configure a badge that is awarded when a learner enrolls in 5 courses, you add this definition.

`5, enrolled_five`

where `enrolled_five` is a badge slug that you previously defined in a badge class.
 - **Course groups:** Define badges to be awarded for completing a list of specific courses. Define one badge per line. On each line, enter the slug of the badge class, a comma, then the list of course keys.

For example, to configure a badge that is awarded when a learner completes the 3 prerequisite courses in a series, you add this definition.

`prereq_computerscience_badge_slug, course1_identifier, course2_identifier, course3_identifier`

where `prereq_computerscience_badge_slug` is a badge slug that you previously defined in a badge class, and `course1_identifier`, `course2_identifier`, and `course3_identifier` are the Course IDs for the three courses that must be completed for this badge.
8. When you have finished defining badges in the configuration, select **Save**.
9. To activate this configuration, select **Enabled** at the top of the configuration page.

Important: You can create more than one course event badge configuration, but you can only mark one configuration as **Enabled**. Only the most recently activated course event badge configuration is used.

4.8.8 Creating New Badges for Your Open edX Instance

In addition to using the default customizable badges that are provided with the Open edX platform, you can design new badges that are generated when a particular XBlock-related or course-related action occurs.

Before you create new badges, you should understand the following concepts.

- **Badge Class** - The specification of the badge that is to be awarded. Parameters for badge classes are described in Step 4 of the *Create Course Event Badges for Your Open edX Instance* topic.
- **Slug** - This field in the `BadgeClass` model uniquely identifies the badge class.
- **Issuing Component** - This field in the `BadgeClass` model identifies the part of the software that is issuing the badge. For example, the name of the XBlock where the occurrence of some event triggers the awarding of a badge.

For the customizable *course event badges* that are included with the Open edX platform, the value for `issuing_component` must be `openedx__course` (with two underscores). For *course completion badges* that are included in the Open edX platform, the issuing component value should be empty.

For new badge types that you create, specify an **Issuing Component** value that identifies the software component responsible for issuing the badge. For example, if badges are issued by the course management component, you might define **Issuing Component** as `platform__course`; if badges are issued based on activity in course discussions, you might define **Issuing Component** as `platform__discussions`.

- The combination of badge slug, issuing component, and optionally, course ID uniquely identifies an awarded badge.
- **Award method** - The `award` function on the `Badge Class` instantiates the badge generator and calls the badge generator's `award` function, which is responsible for awarding a badge to a learner, as well as for performing all checks leading up to the awarding of the badge. On the `Badge Class`, the `award` function either returns a `BadgeAssertion` object or raises an exception if the award cannot be given.
- **Badge Assertion** - A specific generated instance of a badge that a learner has been awarded. Badge assertions contain data about the learner who earned the badge, and the date and time that the badge was awarded. Multiple badge assertions can be awarded for a specific `BadgeClass`, including to the same learner. You can get all assertions for a particular learner for a specific `BadgeClass` using the `get_for_user` method, using `user` as an argument.

For instructions for creating new badges, see *Create New Badges for Your Open edX Instance*.

Create New Badges for Your Open edX Instance

To create new badges for use within your Open edX platform, follow these steps.

1. Create a `Badge Class` specifying the details of the badge you want to award. You can do this using the `get_badge_class` method from the `badges` app, or create a badge class in Django Admin. For Django Admin instructions, see steps 1-5 in *Create Course Event Badges for Your Open edX Instance*.

`BadgeClass.get_badge_class` creates the requested badge class if one does not already exist that has the same combination of `slug` and `issuing_component`. Badge classes are uniquely identified by a combination of their `slug` and `issuing_component` fields, and optionally also the `course_id` field if a badge is associated with an individual course.

If a badge class already exists with the same combination of `slug` and `issuing_component` that is in the request, the existing badge is returned. No new badge class is created, and no changes are made to the values of the existing badge class.

2. In the XBlock or component where badges are to be awarded based on some event occurring, add declarations to the `badging` and `user` services.

The following example illustrates creating a badge class and awarding it from an XBlock.

```
from xblock import XBlock
from xblock.fragment import Fragment
import pkg_resources

@XBlock.wants('badging')
@XBlock.wants('user')
class AwardBlock(XBlock):
    """
    A Block that awards a badge when a learner visits it.
    """
    def award_badge(self, user_service, badge_service):
        user = user_service.get_current_user()
        badge_class = badge_service.get_badge_class(
            slug='general_award', issuing_component='my_org__award_block',
            description="A shiny badge, given to anyone who finds it!",
            criteria="Visit a page with an award block.",
            # This attribute not available in all runtimes,
            # but if we have both of these services, it's a safe bet we're in the_
↪LMS.
            course_id=self.runtime.course_id,
            # The path to this file should be somewhere relative to your XBlock's_
↪package.
            image_file_handle=pkg_resources.get_resource_stream(__name__, 'badge_
↪images/award.png')
            # Badge image should be a square PNG file less than 250KB in size.
        )
        # Award the badge.
        if not badge_class.get_for_user(user):
            badge_class.award(user)

    def student_view(self, context=None):
        """
        Displayed to the learner when they visit.
        """
        # If the user and badge services are not present, we cannot award the badge.
        # If they are, we are ready to award one.
        user_service = self.runtime.service(self, 'user')
        badge_service = self.runtime.service(self, 'badging')
        if user_service and badge_service:
            self.award_badge(user_service, badge_service)
        return Fragment(u"<div><p>You just earned a badge!</p></div>")
```

Get Information about Badge Assertions

Depending on the type of badge you have created and are awarding, you might want to limit the number of times that a learner can receive a badge. You can find whether a specific learner has already received a particular badge in either of the following ways.

- Use `badge_class.assertions_for_user` with `user` as an argument, to return a list of all assertions that the specified user has received for the badge class.
- Use the Badges API GET method to return a list of assertions for a particular username. For example, use `GET /api/badges/v1/assertions/user/{username}/`.

For more information and additional parameters, see [Supported Badges API Endpoint](#).

Supported Badges API Endpoint

The Badges API supports the endpoint `GET /api/badges/v1/assertions/user/{username}/`. You can use the following query parameters with this endpoint.

Note: All of these query parameters are optional.

Parameters	Description
slug	If used, filters by the badge class identified by this slug. Unless <code>issuing component</code> is also specified, assumes a null/empty <code>issuing component</code> .
issuing component	If used, also requires <code>slug</code> to be specified. Filters by the badge class.
courseid	If used, returns only badge assertions that were awarded as part of the specific course.

For example, to get a list of badge assertions issued for a badge with an `issuing_component` value of `openedx__course` and a `slug` value of `enroll_in_three_courses`, the query would be

```
{openedx domain}/api/badges/v1/assertions/user/?issuing_component=openedx__course&
↪slug=enroll_in_three_courses
```

where `<openedx_instance>` is the site URL for your Open edX instance.

Possible query results are as follows.

- 200 on success, with a list of badge assertions.
- 403 if a user who does not have permission to masquerade as another user specifies a username other than their own.
- 404 if the specified user, badge class, or course does not exist.

4.9 Enabling Course Certificates

This topic describes how to enable and configure course certificates in your instance of Open edX.

For information about configuring program certificates, refer to documentation in the `edx/credentials` GitHub repository.

- *Course Certificates Overview*
- *Enable Course Certificates in Studio and the LMS*
- *Configuring Course Certificates in Studio*
- *Configure Course Certificates for Your Open edX Instance*
- *Customize Certificate Templates For Your Organization*
- *Enable Certificates in Additional Languages*
- *Display Hours of Effort on Course Certificates*
- *Generate Certificates For a Course*

4.9.1 Course Certificates Overview

Organizations and course teams can choose to generate certificates for learners who pass a course. Learners can view, print, or share their certificates.

For additional information about certificates, see [Setting Up Certificates in Studio](#) in the *Building and Running an Open edX Course* guide or [Print a Web Certificate](#) in the *Open edX Learner's Guide*.

To enable course certificates on your instance of Open edX, you must enable a feature flag in both Studio and the Learning Management System and complete the configuration tasks described in this topic.

Note: Before proceeding, review [Guidelines for Updating the Open edX Platform](#).

4.9.2 Enable Course Certificates in Studio and the LMS

To enable certificates, you modify the `lms.env.json` and `cms.env.json` files, which are located one level above the `edx-platform` directory.

1. In the `lms.env.json` and `cms.env.json` files, set the value of `CERTIFICATES_HTML_VIEW` within the `FEATURES` object to `true`.

```
"FEATURES": {  
  ...  
  'CERTIFICATES_HTML_VIEW': true,  
  ...  
}
```

2. Save the `lms.env.json` and `cms.env.json` files.
3. If it does not exist already, create the folder `/tmp/certificates` owned by the user and group `www-data`. Depending on your configuration, this folder might not survive reboots, and so might need to be created by a script.
4. Run database migrations.

4.9.3 Configuring Course Certificates in Studio

Within Studio, course team members with the Admin role can create and edit a certificate configuration that is used to generate certificates for their course, including adding signatories and images for organization logo and signature images for signatories. For details, [Setting Up Certificates in Studio](#) in *Building and Running an Open edX Course*.

4.9.4 Configure Course Certificates for Your Open edX Instance

1. Access the LMS Django Administration website for your instance of Open edX. To do this, go to `https://<host name of your Open edX instance>/admin`. For example, this might be `https://courses.YourOrganization.com/admin`.
2. Under **Site Administration** > **Certificates**, add an HTML View Configuration, and select **Enabled**.
3. Set the following certificates-related parameters for your Open edX instance.
 - `platform_name`
 - `company_about_url`

- `company_privacy_url`
- `company_tos_url`
- `company_verified_certificate_url`
- `logo_src`
- `logo_url`

For each course mode for which you want to offer certificates (such as “honor” or “verified”), define these parameters.

- `certificate_type`
- `certificate_title`
- `document_body_class_append`.

Make sure the mode name matches your course mode name exactly. An example follows.

For more information about course modes, also called enrollment modes or enrollment tracks, see [enrollment track](#).

```
{
  "default": {
    "accomplishment_class_append": "accomplishment-certificate",
    "platform_name": "YourPlatformName",
    "company_about_url": "http://www.YourOrganization.com/about-us",
    "company_privacy_url": "http://www.YourOrganization.com/our-privacy-policy
↪",
    "company_tos_url": "http://www.YourOrganization.com/our-terms-service",
    "company_verified_certificate_url": "http://www.YourOrganization.com/
↪about_verified_certificates",
    "logo_src": "/static/certificates/images/our_logo.svg",
    "logo_url": "www.YourOrganization.com"
  },
  "honor": {
    "certificate_type": "honor",
    "certificate_title": "Honor Certificate",
    "document_body_class_append": "is-honorcode"
  },
  "verified": {
    "certificate_type": "verified",
    "certificate_title": "Verified Certificate",
    "document_body_class_append": "is-idverified"
  },
  "base": {
    "certificate_type": "base",
    "certificate_title": "Certificate of Achievement",
    "document_body_class_append": "is-base"
  },
  "distinguished": {
    "certificate_type": "distinguished",
    "certificate_title": "Distinguished Certificate of Achievement",
    "document_body_class_append": "is-distinguished"
  }
}
```

4. Save the configuration parameters.
5. Restart the Studio and LMS processes to load the updated environment configurations.

Discontinue Audit Track Certificates

Organizations that offer certificates to audit track learners who pass a course can discontinue generation of this type of certificate. For example, your organization makes a strategic decision to offer certificates only to learners who select an enrollment track other than “audit”. Learners can continue to audit courses, but they no longer receive certificates.

For more information about course tracks, also called enrollment modes or enrollment tracks, see *enrollment track*.

An outline of the steps you might take if your organization decides to stop offering certificates for learners in the audit track follows.

1. Stop advertising audit track certificates for new courses.
2. Identify running courses that offer an audit track certificate and, for those courses, determine the course end date that is furthest in the future.
3. Select a cutoff date for generating audit track certificates that is after the last course end date identified in step 2.
4. Set `AUDIT_CERT_CUTOFF_DATE` to a date in YYYY-MM-DD format. Specifying this date ensures that certificates are not generated for audit track learners in any course after the specified date.

The `AUDIT_CERT_CUTOFF_DATE` feature flag affects only the generation of audit certificates. Learners who audit courses continue to receive grades, which are shown on the course **Progress** page.

4.9.5 Customize Certificate Templates For Your Organization

Set up the templates for certificates that your organization will issue. Base templates are included, but you must ensure that they are customized for your organization. For example, you can change the images that appear on certificates for each course mode that your organization supports, as well as fonts and colors that are used on certificates.

To issue certificates in more than one language, see *Enable Certificates in Additional Languages*.

To display hours of effort on certificates, see *Display Hours of Effort on Course Certificates*.

Assets for HTML certificates exist in the following locations.

- `lms/templates/certificates` - this folder contains `.html` files for certificates. The file `valid.html` is an example of a certificate file. Files with names that start with an underscore, such as `_certificate_footer.html`, are partial files that can be referenced in the main certificate `.html` files.
- `lms/static/certificates` - subfolders of this folder contain assets used in creating certificates, such as images, fonts, and sass/css files.

Note: The organization logo on a certificate is uploaded in Studio. For details, see *Setting Up Certificates in Studio in Building and Running an Open edX Course*.

4.9.6 Enable Certificates in Additional Languages

You can configure course certificates to render in a specific language.

- *Configure Course Certificates in Additional Languages*

Configure Course Certificates in Additional Languages

To enable generating course certificates in languages other than the default language of your platform, follow these steps.

Note: Base certificate templates already exist for English and Spanish. If you want a course certificate that is different from the default certificate for the organization or language, create a new certificate template.

1. Add the language in which you want to generate certificates to `EDXAPP_CERTIFICATE_TEMPLATE_LANGUAGES` (`edx/configuration/playbooks/roles/edxapp/defaults/main.yml`), where the key is the language code and the value is the name of the language.

For example, `'fr': 'français'`.

2. In the LMS Django Administration site for your instance of Open edX, under **Site Administration > Certificates > Certificate templates**, add a certificate template for each additional language in which you want to generate certificates.
3. In each certificate template, modify the configuration parameters as required to apply the template either to all course runs in an organization, or to a single course run.

Parameter	To apply the template to all course runs in the organization	To apply the template to a specific course run
Language	Select the language that you want the certificate to be generated in.	Select the language that you want the certificate to be generated in.
Organization ID	Enter the ID for the organization whose courses should use this certificate template.	Select None.
Course Key	Leave empty.	Enter the course key for the course run which should use this certificate template.
Mode	(Optional) Specify the course mode for which certificates will be generated using this template. If no mode is specified, this template is used for all course modes.	(Optional) Specify the course mode for which certificates will be generated using this template. If no mode is specified, this template is used for all course modes.
Is Active	Select this checkbox to make this template active.	Select this checkbox to make this template active.

Note: If more than one certificate template would apply to a course run, the most specific (lowest level) template is used. For example, if you define a certificate template for all courses in an organization and another template for a specific course run, the template for the course run is used.

4. Save each certificate template.
5. **ONLY** if you are enabling additional language certificates for a specific course run, add a certificate generation course setting in LMS Django Administration, under **Site Administration > Certificates**.
6. In the **Course key** field of the certificate generation course setting, specify the course run for which you are enabling language specific certificate templates.
7. Select **Language specific templates enabled**, and save the configuration.

4.9.7 Display Hours of Effort on Course Certificates

To display hours of effort for a course run on a course certificate, follow these steps.

1. Log in to the Discovery service Django Administration site for your instance of Open edX. To do this, go to `https://<discovery.host name of your Open edX instance>/admin`. For example, this might be `https://discovery.YourOrganization.com/admin`.
2. Under **Course Metadata > Course Runs**, locate the course run, and make sure there are values for the following attributes.
 - Max effort
 - Weeks to complete
3. Log in to the LMS Django Administration site for your instance of Open edX. To do this, go to `https://<courses.host name of your Open edX instance>/admin`. For example, this might be `https://courses.YourOrganization.com/admin`.
4. Under **Site Administration > Certificates**, add or edit a certificate generation course setting.
5. Select **Yes** for **Include hours of effort** and save the configuration.
6. Under **Site Administration > Certificates**, add or edit a certificate template.
7. In the certificate template, ensure that a `div` element exists that includes the context variable `hours_of_effort`.
8. Save your edits to the certificate template.

4.9.8 Generate Certificates For a Course

To generate certificates for a course, run the `manage.py` script with the following settings. When the script finishes running, grades are calculated for learners who are enrolled in the course, and certificates are generated for eligible learners.

1. Obtain the course ID for the course for which you are generating certificates. When you view course content in your browser, the course ID appears as part of the URL. For example, in the URL `http://www.edx.org/course/course-v1:edx+demoX_Demo_2015`, the course ID is `course-v1:edx+demoX_Demo_2015`. For some courses, the course ID contains slashes. For example, `edx/DemoX/Demo_2014`.
2. Run `manage.py` with the following settings, replacing `{CourseID}` with the actual course ID. Do not include beginning or trailing slashes.

```
./manage.py lms --settings=aws ungenerated_certs -c {CourseID}
```

For example,

```
./manage.py lms --settings=aws ungenerated_certs -c course-v1:edx+demoX_Demo_2015.
```

Note: If the LMS is running on a server that does not have https support (such as a locally run fullstack for testing) you will need to use the `--insecure` flag so that the certificate generation service contacts the LMS on http instead of on https.

3. View the certificate generation status for a course using `gen_cert_report`. An example follows.

```
./manage.py lms --settings=aws gen_cert_report -c course-v1:edx+demoX_Demo_2015.
```

4.10 Enabling Self-Paced Courses

This topic describes how to enable the self-paced courses feature in your instance of Open edX.

- [Overview](#)
- [Enable Self-Paced Courses in the Learning Management System](#)

4.10.1 Overview

By default, courses are instructor-paced. These courses run on a schedule, typically four to eight weeks, with new content released each week and set assignment due dates. You can configure your instance of Open edX so that it enables self-paced courses. A self-paced course releases content all at once and is available to complete for three to twelve months after the start date. In a self-paced course, there are no due dates other than the course end date.

To enable self-paced courses on your instance of Open edX, you must enable the self-paced course feature in the Learning Management System. Then, course teams are able to set up a course as either instructor-paced or self-paced in Studio.

For information about how course teams set course pacing, see the [Set the Course Run Schedule and Pacing in Studio](#) topic in the *Building and Running an Open edX Course* guide.

Note: Before proceeding, review [Guidelines for Updating the Open edX Platform](#).

4.10.2 Enable Self-Paced Courses in the Learning Management System

To enable self-paced courses, follow these steps to edit the configurations, using the Django administration console for your Open edX LMS.

1. Log in to the Django administration console for the LMS.
2. In the **Self_Paced** section, locate **Self paced configurations**, and then select **Add**.
3. Select the **Enabled** and **Enable course home page improvements** checkboxes.
4. Select **Save**.

4.11 Enabling Public Course Content

By default, learners must create an Open edX account, be signed in to the LMS, and enroll in a course before they can see the course content. The *Public Course Content* feature gives you the option to make either a course outline or course content available to anyone, regardless of whether they have registered for an Open edX account or enrolled in the course. You can decide which courses, and which parts of those courses, that you want to make public. For example, you can:

- **Make just the course outline public.** The LMS displays the course outline without any links to internal course pages, giving potential learners an overview of what they will see when they enroll.
- **Make the entire course public.** Anyone visiting your course outline can follow links to visit internal course pages, and freely navigate HTML and Video course content and handouts.

- **Show different content blocks to public learners vs. enrolled learners.** You can create content tailored to the public view, while still supporting the needs of your enrolled audit and paid learners.

4.11.1 Public Course Content and Search Engines

In addition to making your course content visible to people who already use your site, the public course content feature also allows Google and other search crawlers to index your public course contents. People who are searching for information about the topics you teach can then find your course through their normal search behaviors. This can increase the visibility of your organization's courses and boost enrollments for genuinely interested learners.

4.11.2 Enable Public Course Content in the Admin

The public course content feature is enabled in the Django LMS Admin with the `seo.enable_anonymous_courseware_access` waffle flag. You can use this flag in two different ways:

- Create a normal waffle flag to enable this flag for all courses. For more information, see the [Waffle documentation](#).



LMS Administration

Home > Waffle > Flags > Add flag

Add flag

Name:
The human/computer readable name.

Everyone:
Flip this flag on (Yes) or off (No) for everyone, overriding all other settings. Leave as Unknown to use normally.

- Create a **Waffle flag course override** with the ID of a course to enable this flag for just that course.

The screenshot shows the LMS Administration interface. At the top, there is a blue header with the text 'LMS Administration'. Below the header is a breadcrumb trail: 'Home > Waffle_Utils > Waffle flag course overrides > Add Waffle flag course override'. The main content area is titled 'Add Waffle flag course override'. Below the title, there is a text input field for 'Waffle flag' containing the value 'seo.enable_anonymous_courseware_access'. Below that is a text input field for 'Course id' containing the value 'course-v1:edX+DemoX+Demo_Course'. Below that is a dropdown menu for 'Override choice' with 'Force On' selected. At the bottom, there is a checkbox labeled 'Enabled' which is checked.

4.11.3 Set Visibility for a Course in Studio

After you set the waffle flag to enable public course content in the Django LMS Admin, you set the visibility for the course content and its About page in Studio Advanced Settings.

Set Course Content Visibility

1. View your course in Studio, and navigate to the **Advanced Settings** page.
2. Locate the **Course Visibility For Unenrolled Learners** setting.

By default, this is set to "private", which ensures that only enrolled learners can access your course content.

If you change this to "public_outline", then your course outline, but not the other course content, will be visible to everyone.

If you change this to "public", then all of your course content, including the course outline, will be visible to everyone.

Set Course About Page Visibility

If you want your course to be crawled by Google and other search engines, you should also ensure that your course's About page is visible, and that it is shown in the course catalog. Without these settings, only people with a link to your course outline or specific content blocks will be able to find your course.

1. View your course in Studio, and navigate to the **Advanced Settings** page.
2. Locate the **Course Visibility in Catalog** setting and set it to "both".

4.11.4 Feature Limitations

The public course content feature is currently subject to some limitations, including the following.

- Anonymous or unenrolled learners are not members of any cohort, and so they only see course blocks that are not assigned to a content group. If you want to restrict public access to selected course blocks, you need create content groups for your private content, and ensure that your enrolled learners are members of a cohort that can see this content group.
- Only HTML components, Video components, and course handouts have a “public” view. Unenrolled learners will see a message requesting that they enroll to see more complex content types, such as discussion forums, problem blocks, randomized content blocks, exams, Open Response Assessment, and other XBlocks.
- Unenrolled learners will not see course completion or progress updates as they proceed through the course, and Open edX doesn’t remember where they left off if they leave the course and come back.
- The edX mobile apps do not support public course content.

4.12 Enabling Custom Courses

To enable designated users to create custom courses (CCX) on your instance of Open edX, you must configure the `server-vars.yml` file in the edX platform.

Note: Before proceeding, review *Guidelines for Updating the Open edX Platform*.

1. Stop the LMS server.
2. Create or update the file `/edx/app/edx_ansible/server-vars.yml` to include the CCX feature flag.

```
EDXAPP_FEATURES:  
  CUSTOM_COURSES_EDX: true
```

3. Run the command `/edx/bin/update`.

```
sudo /edx/bin/update edx-platform <your-branch-name>
```

4. Restart the LMS server.

4.13 Enabling Custom Course Settings

To enable course developers to add custom fields to a course on your instance of Open edX, you must configure the `cms.env.json` file in the edX platform.

Note: Before proceeding, review *Guidelines for Updating the Open edX Platform*.

1. Stop the LMS server.
2. Create or update the file `cms.env.json` to include the custom course settings feature flag.

```
'ENABLE_OTHER_COURSE_SETTINGS': true,
```

3. Restart the LMS server.

4.14 Enabling Discussion Notifications

You can set up notifications that learners receive the first time that anyone adds a response to a discussion post that they have made.

For more information, including the text of the discussion notification message, see [Automatic Email Messages from the Open edX Platform and Receiving Notifications](#).

- [Requirements for Discussion Notifications](#)
- [Enable Discussion Notifications](#)

4.14.1 Requirements for Discussion Notifications

To create discussion notifications for your instance of the Open edX platform, you need the following items.

- edX Automated Communication Engine (ACE). For more information about how to install and configure ACE, see [edX Automated Communication Engine](#).
- A third party email client, such as Salthru. For information about how to configure your email client, see the documentation for the client.

4.14.2 Enable Discussion Notifications

After you have set up ACE and the third party email client, you enable discussion notifications in the Django administration console for your instance of Open edX.

1. Sign in to the LMS Django administration console for your base URL. For example, `http://{your_URL}/admin`.
2. On the **Site Administration** page, locate **Site Configuration**.
3. In the **Site Configuration** section, next to **Site configurations**, select **Change**.
4. On the **Change site configuration** page, locate the **Values** field, and then add the following value.

```
{  
  "enable_forum_notifications":true  
}
```

5. Select **Save**.

4.15 Enabling Entrance Exams

This topic describes how to enable entrance exams in your instance of Open edX.

- [Overview](#)
- [Configure the Milestones Application](#)
- [Enable Entrance Exams in Studio and the Learning Management System](#)

4.15.1 Overview

Course teams can create an entrance exam for the course. Learners must pass the entrance exam before participating in the course.

To enable this feature on your instance of Open edX, you must enable entrance exams in Studio and the Learning Management System.

For information about entrance exams, see the *Building and Running an Open edX Course* and *Open edX Learner's* guides.

Note: Before proceeding, review *Guidelines for Updating the Open edX Platform*.

4.15.2 Configure the Milestones Application

1. Set the value of `MILESTONES_APP` in the `lms.env.json` and `cms.env.json` files to `True`.

```
# Milestones application flag
'MILESTONES_APP': True,
```

2. Save the `lms.env.json` and `cms.env.json` files.
3. Run database migrations.

4.15.3 Enable Entrance Exams in Studio and the Learning Management System

To enable entrance exams, you modify the `lms.env.json` and `cms.env.json` files, which are located one level above the `edx-platform` directory.

1. Set the value of `ENTRANCE_EXAMS` in the `lms.env.json` and `cms.env.json` files to `True`.

```
# Entrance exams feature flag
'ENTRANCE_EXAMS': True,
```

2. Save the `lms.env.json` and `cms.env.json` files.

4.16 Configuring Open Response Assessments

You can change the default configuration for the Open Response Assessment (ORA2) application. You can change the default file storage system or change the default set of files that learners are prohibited from submitting.

4.16.1 Configuring ORA2 to Upload Files to Alternative Storage Systems

By default, the Open Response Assessment (ORA2) application stores files that learners upload in an Amazon S3 bucket.

You can configure ORA2 to store files in an alternate system. To have learners' files stored in a system other than Amazon S3, follow these steps.

1. In the ORA-2 repository, implement the `BaseBackend` class defined in the `base.py` file.

For example, the `S3.py` file in the same directory is an implementation of `BaseBackend` for Amazon S3. You must implement the equivalent class for the storage system you intend to use.

2. Configure ORA2 to use your alternative storage system by modifying the value of `backend_setting` in `init file` to point to your implementation of BaseBackend.
3. Add code to instantiate the new implementation to the `get_backend()` function in the `init.py` file.
4. Configure ORA2 to use the alternative storage system by modifying the value of `ORA2_FILEUPLOAD_BACKEND` in the Django settings to point to your implementation of BaseBackend.

4.16.2 Prohibiting Submission of Specified File Types

Course teams can configure open response assessments so that learners can upload files along with their text responses. During the peer review stage of the assessment, other learners download the submitted file and read the response.

To protect learners from exposure to files with malicious content, the ORA2 application uses a “blacklist” to identify a set of file types that learners are not permitted to upload.

To add or remove file types from the blacklist, follow these steps.

1. In the ORA-2 repository, use an editor to open the `submission_mixin.py` file.
2. Locate the `FILE_EXT_BLACK_LIST` parameter in the file. By default, this parameter lists the following file types.

```
FILE_EXT_BLACK_LIST = [  
    'exe', 'msi', 'app', 'dmg', 'com', 'pif', 'application', 'gadget',  
    'msp', 'scr', 'hta', 'cpl', 'msc', 'jar', 'bat', 'cmd', 'vb', 'vbs',  
    'jse', 'ws', 'wsf', 'wsc', 'wsh', 'scf', 'lnk', 'inf', 'reg', 'ps1',  
    'ps1xml', 'ps2', 'ps2xml', 'psc1', 'psc2', 'msh', 'msh1', 'msh2', 'mshxml',  
    'msh1xml', 'msh2xml', 'action', 'apk', 'app', 'bin', 'command', 'csh',  
    'ins', 'inx', 'ipa', 'isu', 'job', 'mst', 'osx', 'out', 'paf', 'prg',  
    'rgs', 'run', 'sct', 'shb', 'shs', 'u3p', 'vbscript', 'vbe', 'workflow',  
    'htm', 'html',  
]
```

3. Add or remove values from this list.
4. Save your changes to `submission_mixin.py`.
5. Restart the Studio (CMS) and Learning Management System (LMS) processes so that your updates are loaded.

For more information and examples of how course teams might set up an open response assessment, see [Introduction to Open Response Assessments](#) in the *Building and Running an Open edX Course* guide.

4.17 Enabling Course Prerequisites

This topic describes how to enable course prerequisites in your instance of Open edX.

- [Overview](#)
- [Configure the Milestones Application](#)
- [Enable Prerequisite Courses in Studio and the Learning Management System](#)

4.17.1 Overview

Course teams can set prerequisites for a course. Learners must complete the prerequisite courses before participating in the course.

To use this feature on your instance of Open edX, you must configure the Milestones application, then enable prerequisites in Studio and the Learning Management System.

For information about prerequisites, see the *Building and Running an Open edX Course* and *Open edX Learner's* guides.

Note: Before proceeding, review *Guidelines for Updating the Open edX Platform*.

4.17.2 Configure the Milestones Application

1. Set the value of `MILESTONES_APP` in the `lms.env.json` and `cms.env.json` files to `True`.

```
# Milestones application flag
'MILESTONES_APP': True,
```

2. Save the `lms.env.json` and `cms.env.json` files.
3. Run database migrations.

4.17.3 Enable Prerequisite Courses in Studio and the Learning Management System

To enable prerequisite courses, you modify the `lms.env.json` and `cms.env.json` files, which are located one level above the `edx-platform` directory.

1. Set the value of `ENABLE_PREREQUISITE_COURSES` in the `lms.env.json` and `cms.env.json` files to `true`.

```
# Prerequisite courses feature flag
'ENABLE_PREREQUISITE_COURSES': true,
```

2. Save the `lms.env.json` and `cms.env.json` files.

4.18 Enabling Course and Video Licensing

This topic describes how to enable licensing in your instance of Open edX.

- [Overview](#)
- [Enable Licensing in Studio](#)

4.18.1 Overview

Course teams can specify licensing options for course content as well as for each video in a course.

Course teams can select one of the following license options.

- All Rights Reserved
- Creative Commons

By specifying the license, course teams communicate to learners whether and how they can reuse course content.

To enable this feature on your instance of Open edX, you must enable licensing in both Studio and the Learning Management System.

Note: Before proceeding, review *Guidelines for Updating the Open edX Platform*.

4.18.2 Enable Licensing in Studio

To enable licensing, you modify the `lms.env.json` and `cms.env.json` files, which are located one level above the `edx-platform` directory.

1. In the `lms.env.json` and `cms.env.json` files, in the `FEATURES` dictionary, add `'LICENSING': True`:

```
FEATURES = {  
    'LICENSING': True,  
    . . .
```

2. Save the `lms.env.json` and `cms.env.json` files.

4.19 Configuring Transcript Behavior

As a best practice, transcripts should always be provided, so that course videos meet accessibility requirements. Video transcripts are displayed in the language selected by the learner in the video player if they are available in that language. Otherwise, by default video transcripts fall back to an English language transcript. In cases where no transcript is available, you can configure Open edX so that the video player does not display the caption and transcript buttons.

You can configure the default transcript behavior using the `FALLBACK_TO_ENGLISH_TRANSCRIPTS` feature flag. By default, this feature flag is set to `TRUE`. If you set it to `FALSE`, then the video transcript will not fall back to an English language transcript and if no transcript is available, the caption and transcript buttons are not displayed in the video player.

4.19.1 Configuring the Transcript Feature Flag

Note: Before proceeding, review *Guidelines for Updating the Open edX Platform*.

To set the `FALLBACK_TO_ENGLISH_TRANSCRIPTS` feature flag, you modify the `lms.env.json` and `cms.env.json` files, which are located one level above the `edx-platform` directory.

1. In the `lms.env.json` and `cms.env.json` files, in the `FEATURES` dictionary, change the value of `FALLBACK_TO_ENGLISH_TRANSCRIPTS` to `FALSE`.
2. Save the `lms.env.json` and `cms.env.json` files.

4.20 Configuring an edX Instance as an LTI Tool Provider

You can configure your edX instance to be a learning tool interoperability (LTI) provider to other systems and applications. You can use this LTI capability to present content from an edX course in any application that is configured to be a consumer of that content. After you enable your edX instance as an LTI tool provider and configure credentials for the tool consumers, course teams can reuse course content from the edX instance in contexts other than the edX LMS.

4.20.1 Enable LTI Provider Functionality

LTI provider functionality is provided in the `lti_provider` app, located in `edx-platform/lms/djangoapps/lti_provider`.

By default, the `lti_provider` app is not used by edX installations. To enable this functionality throughout the platform, follow these steps.

1. In the `edx/app/edxapp/lms.env.json` file, edit the file so that it includes the following line in the `features` section.

```
"FEATURES" : {  
  ...  
  "ENABLE_LTI_PROVIDER": true  
}
```

2. Save the `edx/app/edxapp/lms.env.json` file.
3. Run database migrations.
4. Restart the LMS server.

To verify that the LTI provider functionality is enabled, you can check for the presence of the following database tables.

```
lti_provider_gradedassignment  
lti_provider_lticonsumer  
lti_provider_ltiuser  
lti_provider_outcomeservice
```

If these tables are not present, check that the migrations have run properly.

4.20.2 Configuring Credentials for a Tool Consumer

For each external learning management system or application (external LMS) that you want to allow access to your edX instances as an LTI tool consumer, you create OAuth1 credentials, and then configure your edX instance to allow access. Each external LMS that you *configure as a tool consumer* must have separate credentials.

After you complete the configuration of a tool consumer on your edX system, you can add the consumer credentials to your external LMS. For examples of how course teams might set up a course on an external LMS as a consumer of edX course content, see [Using Open edX as an LTI Tool Provider](#) in the *Building and Running an edX Course* guide.

Configure the Tool Consumer

To configure an LTI tool consumer to have access to your Open edX installation, follow these steps.

1. Sign in to the Django administration console for your base URL. For example, `http://{your_URL}/admin`.

2. In the **LTI Provider** section, next to **LTI Consumers** select **Add**.
3. Enter the following information.
 - **Consumer Name:** An identifying name for the tool consumer.
 - **Consumer Key:** The console generates a unique key value for this tool consumer. Alternatively, you can use an external application to generate the key, and then enter it here.
 - **Consumer Secret:** The console generates a unique secret value for this tool consumer. Alternatively, you can use an external application to generate the secret, and then enter it here.

Important: Do not supply a value for the **Instance guid** field. The tool consumer generates and supplies a globally unique identifier.

4. Select **Save** at the bottom of the page.

4.20.3 Define an Interval for Grade Aggregation (Optional)

When an external LMS links to problem components in a graded edX subsection, the edX system grades the answers to those problems, and then transfers the grades back to the external LMS.

- If the link is to an individual problem component on the edX system, the edX system returns the grade for each learner immediately.
- If the link is to a unit or subsection, you can configure an interval of time for the edX system to delay before returning the grades. The edX system aggregates all of the problems in the unit or subsection that the learner answers during that interval. Aggregating grades can reduce the number of notification messages that learners receive.

By default, the edX system aggregates grades for units and subsections every 15 minutes.

To change the interval for returning aggregated grades, follow these steps.

1. In `edx/app/edxapp/lms.env.json`, change the value for the following parameter.

```
LTI_AGGREGATE_SCORE_PASSBACK_DELAY = 15 * 60
```

You specify a time value in seconds.

2. Save the `/lms/envs/common.py` file.
3. Restart the Learning Management System processes so that the updated environment configurations are loaded.

4.20.4 Options for LTI Authentication and User Provisioning

When you use your Open edX system as an LTI tool provider, data is collected by the Open edX system for all learner activity. Each learner has a user account on the Open edX system that is linked to the user account on the tool consumer system, so that activity, grades, and state can be passed from one system to the other.

The Open edX system supports these user authentication flows for LTI.

- *Anonymous User Authentication*
- *Open edX User Authentication*

Anonymous User Authentication

The first time a learner encounters an Open edX resource in a course, the Open edX content is immediately launched by a POST to the URL. Without requiring any action from the learner, the Open edX system creates a user account and provisions it with a system-generated username, and links it to the tool consumer user account for that learner. Learners never interact with the Open edX system directly.

This authentication flow presents a virtually seamless experience that significantly reduces user error. The Open edX system passes learner data to the tool consumer with no subsequent reconciliation of data between the systems.

After you *configure your edX instance as an LTI tool provider*, no further configuration is needed on your Open edX system for this user authentication flow.

Open edX User Authentication

The first time a learner encounters an Open edX resource in a course, he is prompted to either sign in with existing credentials or create a user account. The Open edX system creates a user account and provisions it with the supplied values, and links it to the tool consumer user account for that learner. The POST to the URL then delivers the Open edX resource in the tool consumer. After the initial sign in or account creation step, learners do not interact with the Open edX system directly.

In this authentication flow, learners knowingly establish or use credentials on the Open edX system. This flow provides a smooth learner experience that can also satisfy legal requirements or privacy concerns.

After you configure your edX instance as an LTI tool provider, you can *configure Open edX user authentication* between your Open edX system and the tool consumer.

4.20.5 Configuring Open edX User Authentication for LTI

Every learner who accesses content on an Open edX system must have a user account. The Open edX system uses the accounts to collect data for learner interactions with course content.

After you *configure your edX instance as an LTI tool provider*, you can configure Open edX user authentication between your Open edX installation and an LTI tool consumer.

For more information about the authentication flows that are available, see *Options for LTI Authentication and User Provisioning*.

- *Configure Open edX User Authentication for LTI*
- *Test LTI Authentication*

Configure Open edX User Authentication for LTI

To configure Open edX user authentication between your Open edX installation and an LTI tool consumer, follow these steps.

Note: A consumer key and secret are required. The Django administration console provides a hexadecimal string for the secret, but does not provide a hexadecimal string for the key. You must use an external tool to generate the key.

1. Sign in to the Django administration console for your base URL. For example, `http://{your_URL}/admin`.

2. In the **Third_Party_Auth** section, next to **Provider Configuration (LTI)** select **Add**.
3. Select **Enabled**.
4. Enter the **Name** of your Open edX system, as you want it to appear on the registration page that is presented to learners who access Open edX content from this LTI tool consumer.
5. To customize the registration process for learners, you can make selections for these optional fields.
 - **Skip Registration Form:** If you select this option, users are not asked to confirm any user account data that is supplied for them by the LTI tool consumer (name, email address, and so on).
By default, this option is cleared and learners review a registration form with the account details supplied by the tool consumer.
 - **Skip Email Verification:** If you select this option, users are not required to confirm their email addresses, and their accounts are activated immediately upon registration.
By default, this option is cleared and learners receive an email message and must select a link in that message to activate their user accounts.
6. Enter the following information.
 - **Lti consumer key:** Enter the hexadecimal string of the key.
 - **Lti consumer secret:** The system generates a hexadecimal string value for this field. Alternatively, you can replace it with a secret generated by an external tool.
7. Optionally, change the default value for the **Lti max timestamp age**.
8. Select **Save** at the bottom of the page.

Test LTI Authentication

To verify the sign in process for an LTI provider configuration, follow these steps.

1. Have the LTI consumer key and secret for the LTI provider configuration available. For example, use the Django administration console to open the **Change Provider Configuration (LTI)** page.
2. Use a separate browser window or tab to open the **IMS LTI 1.1 Consumer Launch** page.
3. As the **Launch URL**, enter your base URL followed by `/auth/login/lti/`. For example, `http://{your_URL}/auth/login/lti/`.
4. Copy the **Lti consumer key** value, and then on the **IMS LTI 1.1 Consumer Launch** page paste it in as the **Key**.
5. Copy the **Lti consumer secret** value, and then on the **IMS LTI 1.1 Consumer Launch** page paste it in as the **Secret**.
6. Optionally, change the default values in the **Launch Data** section of the **IMS LTI 1.1 Consumer Launch** page to match the set of values that the tool consumer is configured to supply.
7. To test the workflow for a learner who does not yet have a user account on your Open edX system, follow these steps.
 - Use a separate browser window or tab to make sure that you are signed out of your Open edX LMS.
 - On the **IMS LTI 1.1 Consumer Launch** page, select **Recompute Launch Data** and then select **Press to Launch**.

The page that is configured for delivery to an unauthenticated user loads at the bottom of the page. In the example that follows, the registration page appears (that is, it was not configured to be skipped) and the learner is prompted to complete required fields.

IMS LTI 1.1 Consumer Launch

This is a very simple reference implementation of the LMS side (i.e. consumer) for IMS LTI 1.1.

[Toggle Resource and Launch Data](#)

LTI Resource


Launch URL:

Key:

Secret:

Launch Data

[toggle debug data](#)

 REGISTER Sign in

We couldn't create your account.

- Please enter your Email.
- Please select your Country.

You've successfully signed into [redacted]. We just need a little more information before you start learning with edX.

Email *

Full name *

Needed for any certificates you may earn

Public username *

The name that will identify you in your courses - (cannot be changed later)

Country *

Gender Year of birth

Highest level of education completed

Mailing address

Tell us why you're interested in edX

I agree to the edX [Terms of Service and Honor Code](#). *

* Required field

8. To test the workflow for a learner who already has a user account on your Open edX system, follow these steps.
 - Use a separate browser window or tab to sign in to your Open edX LMS.
 - On the **IMS LTI 1.1 Consumer Launch** page, select **Recompute Launch Data** and then select **Press to Launch**.

Your Open edX user account is linked to the LTI provider configuration, and your learner dashboard on the Open edX site loads at the bottom of the page. To unlink your user accounts, select the arrow next to your username, and then select **Account**. In the **Connected Accounts** section, select **Unlink** next to the LTI provider configuration name.

For more information and examples of how course teams might set up a course on an external LMS as a consumer of edX course content, see [Using Open edX as an LTI Tool Provider](#) in the *Building and Running an edX Course* guide.

4.21 Enabling Social Sharing of Courses and Certificates

This section describes how to configure Open edX so that learners can share their certificates, and so course teams can enable learners to share their courses on social media.

- [Overview](#)
- [Configure Social Sharing](#)
- [Enable Custom Course URLs](#)

4.21.1 Overview

You can enable learners to share courses and certificates that they earn on social media sites such as Facebook and Twitter.

To use this feature on your instance of Open edX, you must configure social sharing settings.

Optionally, you can also enable course teams to set custom URLs for social sharing. If a course team sets a custom course URL, posts to the social sharing site can include a link back to that URL. If you do not enable custom course URLs, a link to the course About page in the LMS is used.

Note: Before proceeding, review [Guidelines for Updating the Open edX Platform](#).

4.21.2 Configure Social Sharing

To enable social sharing icons for courses, you modify the `lms.env.json` file, which is located one level above the `edx-platform` directory.

1. In the `lms.env.json` file, modify the `SOCIAL_SHARING_SETTINGS` dictionary as needed.

```
SOCIAL_SHARING_SETTINGS = {
  'CUSTOM_COURSE_URLS': True,
  'DASHBOARD_FACEBOOK': True,
  'CERTIFICATE_FACEBOOK': True,
  'CERTIFICATE_FACEBOOK_TEXT': None,
  'CERTIFICATE_TWITTER': True,
```

```
'CERTIFICATE_TWITTER_TEXT': None,  
'DASHBOARD_TWITTER': True,  
'DASHBOARD_TWITTER_TEXT': None  
}
```

- (a) For each social sharing icon that you want to enable, set the value of the setting to `True`.
 - (b) If you set `DASHBOARD_TWITTER` or `CERTIFICATE_TWITTER` to `True`, you can also specify default text that learners will see in the Twitter sharing dialog and that can be included in their tweet. Set the default text in the `DASHBOARD_TWITTER_TEXT` and `CERTIFICATE_TWITTER_TEXT` values. Learners can edit this text before they select the **Share with Twitter** button in the LMS.
 - (c) If you set `CUSTOM_COURSE_URLS` to `True`, you must *Enable Custom Course URLs*.
2. Configure the `SOCIAL_MEDIA_FOOTER_NAMES` array in the `lms.env.json` file to set the order of links you want learners to see in the footer.

```
SOCIAL_MEDIA_FOOTER_NAMES = [  
    "facebook",  
    "twitter",  
    "youtube",  
    "linkedin",  
    "google_plus",  
    "reddit",  
]
```

3. Configure the `SOCIAL_MEDIA_FOOTER_DISPLAY` dictionary in the `lms.env.json` file to define how you want social media icons to be displayed. For each social media icon you enable, you define a `title`, `icon`, and `action`.

```
"facebook": {  
    "title": _("Facebook"),  
    "icon": "fa-facebook-square",  
    "action": _("Like {platform_name} on Facebook")  
},  
"twitter": {  
    "title": _("Twitter"),  
    "icon": "fa-twitter",  
    "action": _("Follow {platform_name} on Twitter")  
},  
"linkedin": {  
    "title": _("LinkedIn"),  
    "icon": "fa-linkedin-square",  
    "action": _("Follow {platform_name} on LinkedIn")  
}  
}
```

4. Save the `lms.env.json` file.

4.21.3 Enable Custom Course URLs

In addition to enabling the social sharing icons, you can allow course teams to provide a custom URL for social sharing sites to link back to.

You must set the `CUSTOM_COURSE_URLS` parameter to `True` in both the `lms.env.json` and `cms.env.json` files. In the `cms.env.json` file, this parameter is the only social sharing setting.

```
SOCIAL_SHARING_SETTINGS = {  
    'CUSTOM_COURSE_URLS': True  
}
```

When finished, save the `lms.env.json` and `cms.env.json` files.

Set a Custom URL for a Course

When you enable custom course URLs in your instance of Open edX, course teams can then set custom URLs for their courses.

In Studio **Advanced Settings**, the course team specifies the custom course URL in the **Social Media Sharing URL** setting.

This URL is provided to the social sharing site for linking back to a course location. This URL is used only if you have enabled custom URLs in your instance of Open edX.

4.22 Configuring a Password Policy

This topic describes how to configure a password policy in your instance of Open edX.

- [Overview](#)
- [Creating a Password Validator](#)
- [Configuring a Password Validator](#)

4.22.1 Overview

By default, Open edX imposes a minimal password complexity policy for all users who log in to the LMS or Studio. Under the default password complexity policy, passwords must contain 2 to 75 characters and cannot be similar to the user's username or email address.

You can substitute your own password policy for the default policy. To configure a password policy in replacement of the default password policy, follow these steps.

1. Create or import a new password validator. This is a Python class that defines how a password is validated. For details about writing a password validator class, see [Creating a Password Validator](#).
2. Add your password validator to the list in the `AUTH_PASSWORD_VALIDATORS` configuration key in the `lms.env.json` configuration file. For details, see [Configuring a Password Validator](#).

4.22.2 Creating a Password Validator

An Open edX password validator is a Python class that specifies how user passwords are validated. You can use whatever criteria you choose to establish a password policy for your Open edX instance. You can create your own custom password validator, or import one or more password validators from `password_policy_validators` in the `edx-platform` repository on GitHub. Those password validators include minimum length, maximum length, user attribute similarity, minimum alphabetic, minimum numeric, minimum uppercase, minimum lowercase, minimum punctuation, and minimum symbols. For more information, see also the [Django password validation documentation](#).

4.22.3 Configuring a Password Validator

To configure your Open edX instance to use a particular password validator, add your password validator to the list in the `AUTH_PASSWORD_VALIDATORS` configuration key in the `lms.env.json` configuration file. For example, to add a password validator named `MyPasswordValidator`, add a line like this to the `lms.env.json` configuration file.

```
"AUTH_PASSWORD_VALIDATORS": [  
  {  
    "NAME": "path.to.file.MyPasswordValidatorClass",  
  },  
]
```

4.23 Enabling Third Party Authentication

To enhance sign in options for your users, you can enable third party authentication, sometimes also called “third party auth”, “single sign on”, or “SSO”, between organizational authentication systems and the sites you define for your implementation of the edX platform. After you enable third party authentication, users can register and sign in to your Open edX site with their campus or organizational credentials.

4.23.1 Supported Identity Providers

In an exchange of authentication and authorization data, an identity provider securely asserts the identity and access rights of a set of users. Your Open edX site is the service provider that allows the users access on the basis of credentials sent by an identity provider.

For example, your Open edX site hosts the courses of three different organizations. When you configure the Open edX site to be a service provider, and configure each of the three organizations to be identity providers, you permit learners who have valid user credentials at any of those organizations to access the Open edX site.

You can enable third party authentication between your Open edX site and many types of identity providers. The Open edX platform provides support for three types of identity providers.

Supported Identity Providers

The Open edX platform has integrated support for the following providers.

- **OAuth** based providers (OAuth2 and the older OAuth v1). Google, Facebook, LinkedIn, and Azure Active Directory are available by default. Any other [OAuth backends supported by python-social-auth v0.2.12](#) can be enabled by changing a configuration setting. People in the Open edX community sometimes use “third party auth” to refer to Google or Facebook integration. Single sign on, or “SSO”, and “third party auth” are largely interchangeable terms for the purposes of this document.
- **Security Assertion Markup Language (SAML) version 2.0**, or Shibboleth. SAML is an SSO standard mostly used by universities and corporations. Shibboleth is the name of a particular implementation of SAML, commonly used by higher education institutions. People in the Open edX community sometimes use “SSO” to refer to SAML or Shibboleth. “SSO” and “Third Party Auth” are largely interchangeable terms for the purposes of this doc. For more information, see [Integrating with a SAML Identity Provider](#).
- **LTI**. Users can use Learning Tools Interoperability® (LTI®) to authenticate.

Provisionally Supported Identity Providers

The Open edX platform also includes limited support for the following SSO providers.

- OpenID
- Apache-hosted Shibboleth
- SSL client certificates
- Central Authentication Service (CAS)

These providers are part of the `external_auth` app, tend to be older and less robustly tested, and have a much more limited feature set. These providers are included in the source code but are not officially supported.

Integrating Identity Providers

Regardless of the standard that the identity provider you want to integrate with uses, you begin by *enabling the third party authentication feature* for your site.

For example, your Open edX site hosts the courses of three different organizations. When you configure the Open edX site to be a service provider, and configure each of the three organizations to be identity providers, you permit learners who have valid user credentials at any of those organizations to access the Open edX site.

If you are using *edX as an LTI tool provider* to a external learning management system or application, you can set up an authentication workflow between your Open edX site and the system that is the LTI tool consumer. For more information, see *Options for LTI Authentication and User Provisioning* and *Configuring Open edX User Authentication for LTI*.

4.23.2 SSO Behavior

The following behavior applies to all three types of provider (OAuth, SAML, and LTI).

New Learner Registration

- When a user signs in by using single sign on (SSO) for the first time, their account is not normally created automatically. Instead, the user information sent by the provider is used to pre-fill the registration form. The user can then edit any of the information before finalizing the creation of their account. The user could also cancel the registration process at this point, and no account would be created for them.
- The user information that is passed from the external authentication provider to Open edX varies depending on the provider. For example, Google, Facebook, a university's SAML provider, and a corporate SAML provider may all provide different user information. Some providers may pass the user's full name, first name, last name, username, email address, external user ID, and more. Other providers may pass only an opaque "user token" that can be used to permanently and consistently identify that user, but which is not considered personal information and does not correspond to any other public identifier.
- After a user submits the registration form, their user account is created and is automatically "linked" to the external provider. For more information, see *Linking Accounts*.
- When a user creates an account by using SSO, the password field on the registration form is hidden. User accounts created by using SSO will have a **random and highly secure password** assigned to their account. The user will not know (or need to know) this password. However, the user can always use the "reset password" feature to change their password, if they would also like to be able to use a traditional password-based sign in method.

Important: No matter which type of sign in method is used, a full and independent Open edX user account is always created for the new user, with a copy of the user’s information. As a result, if the external provider updates the user’s information (such as name or email address), that information will **not** be automatically updated in the user’s Open edX account. In other words, the use of the external account as a reference that provides user details is a one- time event, not an ongoing connection.

Linking Accounts

- To be able to sign in by using an external provider such as Google, the user’s Open edX account must be “linked” to that provider. For example, if a user’s account is linked to Google, the user can click the **Login with Google** button, and will be automatically signed in to their Open edX account.
- User accounts can be linked to zero, one, or many external providers.
- Any provider can be linked or unlinked from an account at any time.
- If an external provider is used to register a new account, the newly created account will automatically be linked to that provider.
- If a user tries to sign in by using an external provider that is not yet linked to any Open edX account, the user will be given the following options.
 - Sign in to an existing account (using a password), which will then link the new provider to that existing account.
 - Create a new account.

4.23.3 Integrating Third Party Authentication in Open edX

For the Open edX platform, you complete two steps to integrate third party authentication.

1. Enable the third party authentication feature.
2. Set up a provider.

Enable the Third Party Authentication Feature

By default, third party authentication is not enabled on Open edX installations. To enable this feature, follow these steps.

1. In the `edx/app/edxapp/lms.env.json` file, edit the file so that it includes the following line in the features section.

```
"FEATURES" : {  
  ...  
  "ENABLE_COMBINED_LOGIN_REGISTRATION": true,  
  "ENABLE_THIRD_PARTY_AUTH": true  
}
```

2. Save the `edx/app/edxapp/lms.env.json` file.

Set Up a Third Party Authentication Provider

You can set up an OAuth or SAML provider. For information about the differences between the two providers, see *Supported Identity Providers*.

Integrating with an OAuth2 Identity Provider

Open edX has specific instructions for Google, Facebook, LinkedIn, and Azure Active Directory. For more information about how to set up one or more of these integrations, see *Common OAuth2 Providers*.

The system also supports integrations with alternative OAuth2 providers. For more information, see *Additional OAuth2 Providers (Advanced)*.

Common OAuth2 Providers

Integrating with the most common OAuth2 IdPs has several steps.

1. *Register the Open edX instance with the provider.*
2. *Configure Open edX.*
3. *Add the provider configuration.*

Register the Open edX Instance

The most common OAuth2 providers are Google, Facebook, LinkedIn, and Azure Active Directory.

- *Register the Open edX Instance with Google*
- *Register the Open edX Instance with Facebook*
- *Register the Open edX Instance with LinkedIn*
- *Register the Open edX Instance with Azure Active Directory*

Register the Open edX Instance with Google

The following instructions describe how to configure Google as a third party authentication provider so that users can use Google accounts (which includes Google Apps accounts) to sign in. These are based on the official Google instructions.

1. Obtain credentials to access the Google API. To do this, follow the [official Google instructions](#) to go to the [Google Developers Console](#), create a new project, and enable the Google+ API service.
2. In the Google Developers Console, select **API Manager**, and then select **OAuth Consent Screen**.
3. For **Product name shown to users** enter the name of your Open edX instance (for example, “Example Academy Online”).
4. Select **Save**.
5. Select the **Credentials** tab, select **Create credentials**, and then select **OAuth client ID**.
 - For **Application type**, select **Web application**.

- Leave the **Authorized JavaScript origins** field blank.
 - For **Authorized redirect URIs**, enter `<Open edX instance domain>/auth/complete/google-oauth2/`. For example, for devstack, enter `http://localhost:8000/auth/complete/google-oauth2/`.
 - Select **Create** to finish creating the credentials.
6. After you obtain the credentials, note the client ID and the client secret.

Register the Open edX Instance with Facebook

To create the app in the Facebook developer portal, follow these steps.

1. Sign in to Facebook, then go to the [Facebook for Developers](#) page.
2. Select **Add a New App**, and then select **Website**.
3. Enter a name for the app, and then select **Create New Facebook App ID**.
4. Enter your information in the rest of the fields in the app creation process.
5. Under **Quick Start for Website**, select **Skip Quick Start**.
You are now at the developer console page for the new Facebook app.
6. Select **Settings**, and note the **App ID** and **App Secret**.
7. On the **Settings** page, select **Add Platform**, and then select **Website**.
8. For **Site URL**, enter the URL of your Open edX instance (for example, `<http://localhost:8000/>` for devstack).
9. In the **App Domains** field, enter the domain name from this URL (for example, `localhost`).
10. Select **Save Changes**.
11. Under **Products -> Facebook Login -> Settings -> Authorized redirect URIs**, enter `<Open edX instance domain>/auth/complete/facebook/`. For example, for devstack, enter `http://localhost:8000/auth/complete/facebook/`.
12. Select **Save Changes**.

Register the Open edX Instance with LinkedIn

To create the LinkedIn app, follow these steps.

1. Go to the [LinkedIn Developers](#) page.
2. Click **Create Application**, enter your information in the form, and then submit the form.
3. On the page that opens with information about the app, note the **Client ID** and **Client Secret**.
4. In the **OAuth 2.0** section, for **Authorized Redirect URL**, enter `<Open edX instance domain>/auth/complete/linkedin-oauth2/`. For example, for devstack, enter `http://localhost:8000/auth/complete/linkedin-oauth2/`.
5. Select **Update** to save your information.

Register the Open edX Instance with Azure Active Directory

You can use Azure Active Directory to allow users with Microsoft Office 365 Business accounts to sign in to Open edX. Note that this feature currently does not work with other types of Microsoft accounts (such as “@live.com” or “@hotmail.com” accounts).

1. If you do not have a Microsoft account, create one on the [Microsoft sign in page](#).
2. If you do not have an Azure subscription, create one on the [Azure account creation page](#).

Note: You must enter a credit card on this page, but if you do not create any virtual machines or other services besides Azure AD, you will not be charged.

3. Go to the [Azure sign in page](#).
4. Click **New**, locate **Active Directory**, and then select **Create**.
5. Enter a name, domain name, and country.
6. Create the new application.
 - (a) Find the new Active Directory in the portal, select **Applications**, select **Add**, and then select **Add an application my organization is developing**.
 - (b) Enter a name for the app, and then select **Web Application**.
 - (c) For **Sign-on URL**, enter <LMS URI>/auth/complete/azuread-oauth2/. For example, you might enter `http://localhost:8000/auth/complete/azuread-oauth2/`.
 - (d) For **App ID URL**, enter <LMS URI>/sign in. For example, you might enter `http://localhost:8000/sign in`.
 - (e) Finish creating the new app.
7. In the portal, locate your Azure AD application, click **Configure**, and then locate and make a note of the client ID. For example, the client ID may be `fe3c3868-0faa-44ee-a1bf-1110aeab1a65`.
8. In the **Keys** section, select a two-year duration, and then select **Save** to create a secret key. Note the value of the key. For example, the key value may be `abcdef12341yHlmOrR8D3v1V1cD2VtL7k9xk9DSB8vw=`.
9. In the **Permissions to other applications** section, locate the **Delegated Permissions** option for Windows Azure Active Directory, and then select **Sign in and read user profile**.
10. Verify the Azure AD domain name. To do this, follow these steps.
 - (a) In the portal, locate the new Active Directory.
 - (b) Select **Domains**, select **Add**, and then add the root domain you want to use (for example, `edx.org`). Make sure that you add the root domain first, and then follow the TXT record verification process.
 - (c) (optional) After the domain has been verified, add subdomains (for example, `courses.edx.org`). Subdomains also request verification, but do not need it.
11. In the Active Directory, select **Applications**, and then select the application that you created.
12. Enable **multi-tenant** support.

Configure Open edX

Configuring Open edX is very similar for Google, Facebook, LinkedIn, and Azure.

1. In the `lms.env.json` file, change the value of `FEATURES > ENABLE_THIRD_PARTY_AUTH` to `true` (it is `false` by default).
2. If necessary, make sure that the correct backend is specified.

- If you are using Google, Facebook, LinkedIn, or Active Directory, open the `lms.env.json` file and look for the `THIRD_PARTY_AUTH_BACKENDS` list. By default, the file does not contain this list.

If the `lms.env.json` file does not contain the `THIRD_PARTY_AUTH_BACKENDS` list, you do not have to complete any additional steps.

If the `lms.env.json` file contains the `THIRD_PARTY_AUTH_BACKENDS` list, add the backend for the applicable IdP to the list.

- For Google, add `"social.backends.google.GoogleOAuth2"`.
- For Facebook, add `"social.backends.facebook.FacebookOAuth2"`.
- For LinkedIn, add `"social.backends.linkedin.LinkedinOAuth2"`.
- **For Azure Active Directory, add** `"social.backends.azuread.AzureADOAuth2"`.
- If you are using a custom backend, add the applicable OAuth2 provider to the `THIRD_PARTY_AUTH_BACKENDS` list in the `lms.env.json` file. If the file does not contain the `THIRD_PARTY_AUTH_BACKENDS` list, create the list, and then add the OAuth2 provider.

For more information, see the [AWS template file](#) file in GitHub.

3. In the `lms.auth.json` file, add the client secret. To do this, create the `SOCIAL_AUTH_OAUTH_SECRETS` key if the key does not already exist, and then add the appropriate value for your IdP.

Note: If you are using Ansible, set the `EDXAPP_SOCIAL_AUTH_OAUTH_SECRETS` variable.

- For Google, add the following value.

```
"SOCIAL_AUTH_OAUTH_SECRETS": { "google-oauth2":  
  "abcdef123456789101112131" }
```

- For Facebook, add the following value.

```
"SOCIAL_AUTH_OAUTH_SECRETS": {  
  "facebook": "98765432181bbe3a2596efa8ba7abcde"  
}
```

- For LinkedIn, add the following value.

```
"SOCIAL_AUTH_OAUTH_SECRETS": { "linkedin-oauth2": "4D3Cb2aB1C0dEFGH" }
```

- For Azure, add the following value.

```
"SOCIAL_AUTH_OAUTH_SECRETS": { "azuread-oauth2":  
  "abcdef12341yHlmOrR8D3v1V1cD2VtL7k9xk9DSB8vw=" }
```

4. Restart the LMS server so that it will use the new settings.

Add the Provider Configuration

1. Go to `<LMS_URI>/admin/third_party_auth/oauth2providerconfig/`. For example, on `devstack`, go to `http://localhost:8000/admin/third_party_auth/oauth2providerconfig/`.

2. Select **Add Provider Configuration (OAuth)**.
3. Make sure that **Enabled** is selected.
4. Make sure that **Visible** is selected.
5. For **Icon Class**, enter the appropriate value.
 - For Google, enter `fa-google-plus`.
 - For Facebook, enter `fa-facebook`.
 - For LinkedIn, enter `fa-linkedin`.
 - For Azure, leave the field blank.
6. For **Name**, enter the appropriate value.
 - For Google, enter `Google`.
 - For Facebook, enter `Facebook`.
 - For LinkedIn, enter `LinkedIn`.
 - For Azure, enter `Microsoft`.
7. For **Backend Name**, select the appropriate value.
 - For Google, select **google-oauth2**.
 - For Facebook, select **facebook**.
 - For LinkedIn, select **linkedin-oauth2**.
 - For Azure, select **azuread-oauth2**.

Note: If the value does not appear in the list, either the `ENABLE_THIRD_PARTY_AUTH` setting or the `THIRD_PARTY_AUTH_BACKENDS` setting is not configured correctly.

8. For **Client ID**, enter the client ID that you noted earlier.
9. Leave **Client Secret** blank. Open edX sets the secret through `lms.auth.json`, which is more secure.
10. (Optional) If you want Facebook or LinkedIn to provide the user's email address during registration, enter the following code into **Other settings**.

For Facebook, use this code.

```
{
  "SCOPE": ["email"],
  "PROFILE_EXTRA_PARAMS": {
    "fields": "id, name, email"
  }
}
```

For LinkedIn, use this code.

```
{ "SCOPE": ["r_basicprofile", "r_emailaddress"], "FIELD_SELECTORS":
["email-address"] }
```

11. Select **Save**.

Users who have an account with the IdP that you have configured can now sign in.

Note: For Google only, if you see the following error message, the `SOCIAL_AUTH_OAUTH_SECRETS` setting is not correct.

```
'unicode' object has no attribute 'get'
```

To resolve this problem, make sure that this setting does not appear multiple times in the `lms.auth.json` file.

Additional OAuth2 Providers (Advanced)

You can add any other third party authentication provider that supports the OAuth2 standard to the Open edX platform. To do this, follow these steps.

Note: OAuth1 providers are also supported and can be configured using these same steps. However, OAuth2 is preferred over OAuth1 wherever possible.

1. In `lms.env.json`, change the value of `FEATURES > ENABLE_THIRD_PARTY_AUTH` to `true` (it is `false` by default).
2. Install the `python-social-auth` authentication backend specific to that provider, and determine the python module path of the backend.

- If the provider is a [python-social-auth supported backend](#), the backend is already installed.

To determine the python module path of the backend, locate the backend in the [list of python-social-auth backends](#), open the file for the backend, and locate the name of the class. The python module path is of the format `social.backends.<file name>.<class name>`.

For example, for GitHub, the file is `github.py` and the class in that file is `GithubOAuth2`. The backend module path is therefore `social.backends.github.GithubOAuth2`.

- If the provider is not a `python-social-auth` supported backend, you must create a new Python package that includes the code required to implement the backend. Your python package must contain a module with a class that subclasses `social.backends.oauth.BaseOAuth2`. For more information, see the [python-social-auth documentation](#), or see the code of the fully supported backends (such as Google or Facebook) for examples.
3. Enable the `python-social-auth` authentication backend specific to that provider:
 - (a) In the `THIRD_PARTY_AUTH_BACKENDS` setting in `lms.env.json`, add the full path of the module to the list.
 - (b) (optional) Set the value of `THIRD_PARTY_AUTH_BACKENDS` to match [the default value in the `aws.py` file](#), and then add any additional backends you need.
 4. Obtain a client ID and client secret from the provider.
 5. Add the client secret to `lms.auth.json`. To do this, create a new key called `SOCIAL_AUTH_OAUTH_SECRETS` if it doesn't already exist, and then add the backend name to that key as follows.

```
"SOCIAL_AUTH_OAUTH_SECRETS": { "backend-name": "secret" }
```

If you are using Ansible, the variable to set is called `EDXAPP_SOCIAL_AUTH_OAUTH_SECRETS`.

6. Restart the LMS server so that it will use the new settings.

7. Go to `<LMS_URI>/admin/third_party_auth/oauth2providerconfig/`. For example, on devstack, go to `http://localhost:8000/admin/third_party_auth/oauth2providerconfig/`.
8. Select **Add Provider Configuration (OAuth)**.
9. Make sure that **Enabled** is selected.
10. Make sure that **Visible** is selected.
11. For **Icon Class**, select one of the following options.
 - Use a generic icon by entering `fa-sign-in`.
 - Use a relevant Font Awesome icon.
 - Upload an SVG icon using the **Icon Image** field.
12. For **Name**, enter the name of the provider.
13. For **Backend Name**, select the backend name from the list. (If it does not appear in the list, either the `ENABLE_THIRD_PARTY_AUTH` setting or the `THIRD_PARTY_AUTH_BACKENDS` setting is not configured correctly.)
14. For **Client ID**, enter the client ID that you noted earlier.
15. Leave **Client Secret** blank. Open edX sets the secret through `lms.auth.json`, which is more secure.
16. Select **Save**.

Users can now sign in using this OAuth2 provider.

Integrating with a SAML Identity Provider

You can integrate your Open edX site with federated identity solutions that use the Security Assertion Markup Language, version 2.0 (SAML 2.0) standard. An example is Shibboleth, a single sign on system that is used by many educational institutions.

- *Exchange Metadata*
- *Add and Enable a SAML Identity Provider*
- *Configure the SAML Identity Provider*
- *Test an Enabled SAML Provider*

Exchange Metadata

SAML metadata is an XML file that contains the information necessary for secure interactions between identity providers and security providers. You send the URL of your metadata file, created when you *configured your Open edX site as a SAML service provider*, to each identity provider that you want to add. Similarly, you obtain the metadata URLs from identity providers before you add and enable them for your installation.

Add and Enable a SAML Identity Provider

To add and enable a SAML 2.0 identity provider, follow these steps.

1. Log in to the Django administration console for your base URL. For example, `http://{your_URL}/admin`.

2. In the **Third_Party_Auth** section, next to **Provider Configuration (SAML IdPs)** select **Add**.

Note: If you want to change the configuration of an existing provider, next to **Provider Configuration (SAML IdPs)** select **Change**, and then select **Update** for the provider that you want to configure.

3. Enter the following information for the provider.
 - **Icon class:** Specifies a [Font Awesome](#) image for the button that users will select to access the sign in page for this IdP. The **fa-sign-in** icon is used by default. For university or institutional providers, a suggested alternative is **fa-university**.
 - **Name:** The name of the IdP as you want it to appear on the sign in page.
 - **Secondary:** Select this option to include the IdP in an intermediary list of providers that users access from a **Use my institution/campus credentials** button on the sign in page.
 - **Backend name:** The default, **tpa-saml**, is optimized for use with Open edX and works with most SAML providers. Select a different option only if you have added a custom backend that provides additional functionality.
 - **Site:** Select the site that you are configuring to use this IdP. Each IdP can only belong to one site at a time. For more information about sites in Open edX, see [Configuring Open edX Sites](#).
 - **IdP slug:** A short, unique name to identify this IdP in the URL. The slug becomes part of a URL, so the value that you enter cannot include spaces.
 - **Entity ID:** The URI that identifies the IdP. This ID must match the value specified in the metadata XML file.
 - **Metadata source:** The URL of the XML file that contains this provider's metadata.
4. Specify your selections for any of the other, optional configuration options. For more information about these options, see [Configure the SAML Identity Provider](#).
5. When you are ready to enable the provider, select **Enabled** at the top of the page. Alternatively, save your configuration settings and enable the provider at another time.
6. Select **Save** or one of the other save options at the bottom of the page.

Next, you can [test an enabled provider](#).

Configure the SAML Identity Provider

To customize the registration process for IdP, you make selections for these optional fields on the **Add Provider Configuration (SAML IdP)** page.

- **Skip Registration Form:** If you select this option, users are not asked to confirm the user account data supplied for them by the IdP (name, email address, and so on). Select this option only for providers that are known to provide accurate user information.

By default, users review a registration form with the supplied account details.

- **Skip Email Verification:** If you select this option, users are not required to confirm their email addresses, and their accounts are activated immediately upon registration.

By default, users receive an email message and must select a link in that message to activate their user accounts.

- **User ID Attribute:** Required. This value is used to associate the user's edX account with the campus account. It is not displayed to users.

By default, uses `userid,urn:oid:0.9.2342.19200300.100.1.1`.

- **Optional user attributes:** You can indicate specific URN values for the following user attributes.

By default, the registration form includes all of the following attributes if they are sent by the IdP.

- **Full Name Attribute:** `commonName,urn:oid:2.5.4.3`
- **First Name Attribute:** `givenName,urn:oid:2.5.4.42`
- **Last Name Attribute:** `surname,urn:oid:2.5.4.4`
- **Username Hint Attribute:** `userid,urn:oid:0.9.2342.19200300.100.1.1`
- **Email Attribute:** `mail,urn:oid:0.9.2342.19200300.100.1.3`

If the identity provider sends a value that you do not want to be included on the the registration form, you can enter a value such as “DISABLED” or “IGNORE” in that field.

Test an Enabled SAML Provider

To verify the sign in process for an IdP that you have enabled, follow these steps.

1. On the Django administration console, in the **Third_Party_Auth** section, select **Provider Configuration (SAML IdPs)**.
2. Check the icon in the **Metadata ready** column for the IdP. After the provider’s metadata is fetched successfully from the URL that you provided as the metadata source, a check mark in a green circle appears and the provider is ready for use immediately.

You might need to wait 30-60 seconds for the task to complete, and then refresh this page.

If the check mark does not appear, make sure that celery is configured correctly and is running. You can also manually trigger an update by running the management command `./manage.py lms saml --pull --settings=aws` on fullstack or `./manage.py lms saml --pull --settings=devstack` on devstack.

3. For additional information about the data fetched from the IdP, on the Django administration console select **SAML Provider Data**, and then select the provider. The page that opens reports data fetched from the metadata source URL and the date and time it was fetched.
4. To verify that users can use the IdP for sign in, go to the sign in page for your LMS. The page should include the institutional sign in button.

Sign in

Email

username@domain.com

The email address you used to register with Sandboxylon 5

Password

[Forgot password?](#)

Remember me

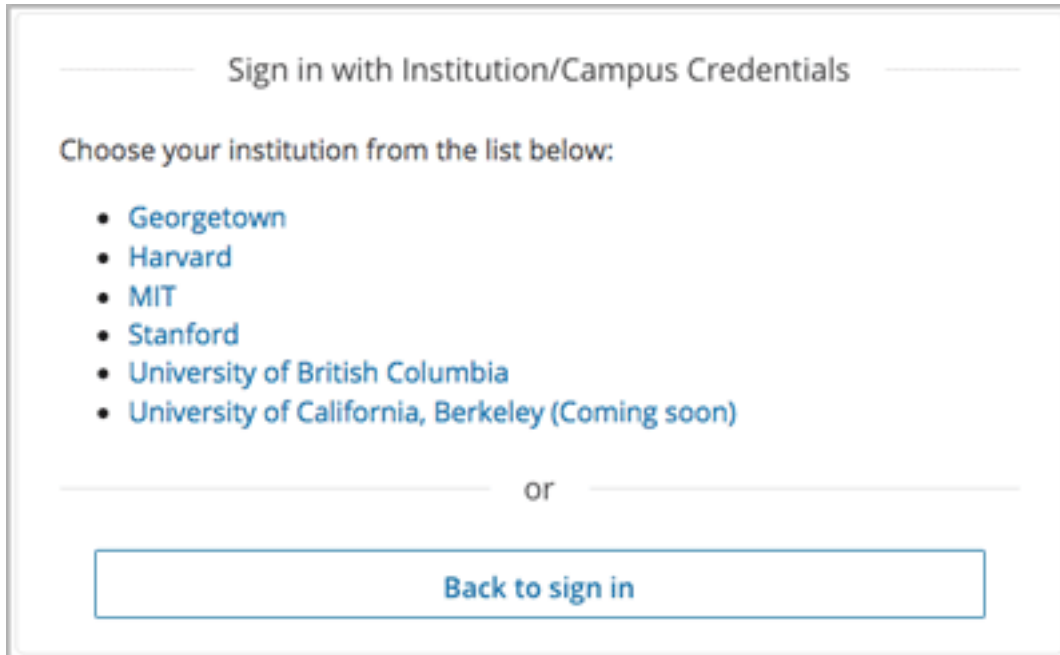
Sign in

or sign in with

[Facebook](#) [Google](#)

[Use my institution/campus credentials](#)

5. Select **Use my institutional/campus credentials**. The list of providers that appears should include the IdP that you enabled.



4.23.4 Configuring your Open edX Site as a SAML Service Provider

The first step in configuring your Open edX site to act as a SAML SP is to create a credential key pair to ensure secure data transfers with identity providers. To complete the configuration procedure, you configure your Open edX site as a SAML SP, which creates your metadata XML file.

- *Generate Public and Private Keys*
- *Add Keys to the LMS Configuration File*
- *Configure your Open edX Site as a Service Provider*
- *Ensure that the SAML Authentication Backend is Loaded*

After you complete this configuration, you can share your metadata file with SAML identity providers, and configure them to assert identity and access rights for users to your installation. For more information, see *Integrating with a SAML Identity Provider*.

Generate Public and Private Keys

To generate the keys for your Open edX installation, follow these steps.

1. On your local computer or on the server, open Terminal or a Command Prompt and run the following command.

```
openssl req -new -x509 -days 3652 -nodes -out saml.crt -keyout saml.key
```
2. Provide information at each prompt.

Two files, `saml.crt` and `saml.key`, are created in the directory where you ran the command.

Add Keys to the LMS Configuration File

Note: Configuration settings added to the `lms.auth.json` file are reset to their default values when you use Ansible to update `edx-platform`.

To configure an Open edX site with your public and private SAML keys, follow these steps.

1. Open the `edx/app/edxapp/lms.auth.json` file in your text editor.
2. Edit the file to add a section for the SAML keys. For example, `# SAML KEYS`.
3. In the new section, add the `SOCIAL_AUTH_SAML_SP_PUBLIC_CERT` parameter followed by a colon (:), a space, and the YAML literal style indicator (!).

```
# SAML KEYS
SOCIAL_AUTH_SAML_SP_PUBLIC_CERT: |
```

4. Open the `saml.crt` file, copy its entire contents, and then paste them onto the line after the `SOCIAL_AUTH_SAML_SP_PUBLIC_CERT` parameter.

```
# SAML KEYS
SOCIAL_AUTH_SAML_SP_PUBLIC_CERT: |
-----BEGIN CERTIFICATE-----
SWP6P/ClypaYkmS...
...j9+hjvbBf3szk=
-----END CERTIFICATE-----
```

5. Add the `SOCIAL_AUTH_SAML_SP_PRIVATE_KEY` parameter followed by a colon (:), a space, and the YAML literal style indicator (!).

```
# SAML KEYS
SOCIAL_AUTH_SAML_SP_PUBLIC_CERT: |
-----BEGIN CERTIFICATE-----
SWP6P/ClypaYkmS...
...j9+hjvbBf3szk=
-----END CERTIFICATE-----
SOCIAL_AUTH_SAML_SP_PRIVATE_KEY: |
```

6. Open the `saml.key` file, copy its entire contents, and then paste them onto the line after the `SOCIAL_AUTH_SAML_SP_PRIVATE_KEY` parameter.

```
# SAML KEYS
SOCIAL_AUTH_SAML_SP_PUBLIC_CERT: |
-----BEGIN CERTIFICATE-----
SWP6P/ClypaYkmS...
...j9+hjvbBf3szk=
-----END CERTIFICATE-----
SOCIAL_AUTH_SAML_SP_PRIVATE_KEY: |
-----BEGIN RSA PRIVATE KEY-----
WlicmlkZN+FtM5h...
...s/psgLDn38Q==
-----END RSA PRIVATE KEY-----
```

7. Save and close the `lms.auth.json` file.

Configure your Open edX Site as a Service Provider

To configure your Open edX site as a SAML service provider, follow these steps.

1. Sign in to the Django administration console for your base URL. For example, `http://{your_URL}/admin`.
2. In the **Third_Party_Auth** section, next to **SAML Configuration** select **Add**.

Note: If you want to change the configuration of an existing service provider, next to **SAML Configuration** select **Change**, and then select **Update** for the provider that you want to configure.

3. Select **Enabled**.
4. Enter the following information.
 - **Entity ID:** Enter a URI for the server. To ensure that this value uniquely identifies your site, the naming convention that edX recommends is to include the server's domain name. For example, `http://saml.mydomain.com/`.
 - **Site:** Specify the site that you are configuring to be a SAML service provider. There can only be one SAML Service Provider per site. For more information about Sites in Open edX, see *Configuring Open edX Sites*.
 - **Organization Info:** Use the format in the example that follows to specify a language and locale code and identifying information for your installation.

```
{
  "en-US": {
    "url": "http://www.mydomain.com",
    "displayname": "{Complete Name}",
    "name": "{Short Name}"
  }
}
```

- **Other config str:** Define the security settings for the IdP metadata files. For more information about the security settings, see the *Python SAML Toolkit*. An example follows.

```
{
  "SECURITY_CONFIG": {
    "signMetadata": false,
    "metadataCacheDuration": ""
  }
}
```

1. Optionally, you can save your public and private keys in the Django administration console. Because this procedure saves your credentials in the database, edX recommends that you use the `lms.auth.json` file instead. For more information, see *Add Keys to the LMS Configuration File*.
2. Select **Save**. You can direct identity providers to `{your LMS URL}/auth/saml/metadata.xml` for your metadata file.

Ensure that the SAML Authentication Backend is Loaded

By default, SAML is included as an approved data format for identity providers. The default configuration of the `/edx/app/edxapp/lms.env.json` file does not explicitly include the `THIRD_PARTY_AUTH_BACKENDS` setting.

If you have customized this file and added the `THIRD_PARTY_AUTH_BACKENDS` setting to it, you might need to verify that the `third_party_auth.saml.SAMLAuthBackend` `python-social-auth` backend class is specified for it. That backend is required before you can add SAML IdPs.

To verify that the SAML authentication backend is loaded on a devstack or fullstack installation, review the `/edx/app/edxapp/lms.env.json` file.

4.23.5 Advanced Third Party Authentication Features

The Open edX Third Party Authentication feature offers many advanced configuration options and integration points that can be used to customize the sign in experience for learners using specific third party providers. None of the following features are required for a typical installation, but they may be useful.

Skip Registration Form

When you configure an OAuth2 or SAML provider in the Django admin, there is a boolean **Skip registration form** option for that provider. Normally, when a new user signs in with an external provider, the information sent to Open edX is only used to pre-fill the registration form. When **Skip registration form** is enabled for a provider, the information sent to Open edX is also used to pre-fill the registration form, but then the form is automatically submitted before the user has a chance to see it. If the registration succeeds, the user will immediately be signed in to their newly created account and can start learning right away. However, if there is any error, such as no email address was provided automatically (since email addresses are required) or the email address of the student conflicts with another existing Open edX user account, then the pre-filled registration form will still be displayed to the student, along with an error message. The student can then fix the error and submit the form to complete the registration process.

Skip Email Verification

When you configure an OAuth2 or SAML provider in the Django admin, there is a boolean **Skip email verification** for that provider. Normally, all users are required to verify their email address by clicking a link in the verification email that gets automatically sent to the user upon registration. When this option is selected, Open edX trusts that the email address provided by the external authentication provider is correct.

If a user registering with a provider that has this option enabled, and the user's email address matches the email address that was sent to Open edX by the external provider, the user's email address will be marked as verified, and no verification email will be required. (However, the email address sent by the external provider is only used to pre-fill the registration form, and the user has a chance to edit the email address on the registration form before creating their account. If they edit their email address so that it no longer matches the email address sent by the external provider, the user will still be required to verify their email address as usual.)

Hinted Sign In

When you link to Open edX, you can create “hinted links” that prompt the user to sign in using a specific provider. For example, the following link represents a typical link to an example course.

```
https://courses.example.com/courses/course-v1:OrganizationX+Course101x+1T2016/  
↪courseware/3eddbb5919544c229d34b3175debc6d6/f9900289d2d0474096d20d23a1eed81/
```

The following link represents a hinted link to the same course. At the end of the URL, `?tpa_hint=oa2-google-oauth2` has been added.

```
https://courses.example.com/courses/course-v1:OrganizationX+Course101x+1T2016/  
↪courseware/3eddbb5919544c229d34b3175debc6d6/f9900289d2d0474096d20d23a1eed81/?tpa_  
↪hint=oa2-google-oauth2
```


When users select the typical link, they go to the sign in page. When they select the hinted link, they receive a “Would you like to sign in using your Google credentials?” prompt that includes a large button to use Google to sign in, as well as a small button to use other sign in options.

The intended use case of this feature is for organizations that wish to provide a link from an on-premise system (or a course in another LMS like Canvas) to a course in Open edX. In that case, the organizations will want students to sign in using the organizations’ SAML providers. By using hinted links, the sign in and/or registration process will be simpler for students, as they won’t have to select a sign in provider or enter a password.

To create a hinted link for an OAuth2 provider, append `?tpa_hint=oa2-providername` to any Open edX URL, where `providername` is the “Backend Name” value from the “Provider Configuration (OAuth)” section of the Open edX Django admin.

To create a hinted link for a SAML provider, append `?tpa_hint=saml-idpslug` to any Open edX URL, where `idpslug` is the “Idp slug” value from the “Provider Configuration (SAML IdP)” section of the Open edX Django admin.

Auto-enrollment

Open edX has a feature that allows instructors, marketing teams, and others to create links that automatically enroll students who click the link into a specific course.

If you send users to `{LMS base URL}/account/finish_auth` and include `course_id`, `enrollment_action`, and optional `course_mode` and `email_opt_in` parameters, the system auto-enrolls the user in the specified course.

For example, the following URL would auto-enroll the user who clicks this link into the edX Demo course (that course’s ID, `course-v1:edX+DemoX+Demo_Course`, has been url-encoded as the value of the `course_id` parameter). If the user is not signed in, they are auto-enrolled after they sign in or register.

```
https://courses.example.com/account/finish_auth?course_id=course-v1%3AedX%2BDemoX%2BDemo_Course&enrollment_action=enroll&email_opt_in=false
```

You can include the following parameters.

- `enrollment_action`: This is required so that the auto-enrollment behavior is triggered. It must be set to either `enroll` (for free courses) or `add_to_cart` (for paid courses). If `add_to_cart` is used, learners are directed to the e-commerce basket page rather than being instantly enrolled.
- `course_id`: This is required. The ID of the course to enroll the user in, or add to cart, etc. The value must be URL-encoded.
- `course_mode`: One of `audit`, `honor`, `verified`, etc. This parameter applies only if `enrollment_action` is `enroll`. If `course_mode` is not specified, and the course has more than one mode, learners are sent to the track selection page, where they can choose which track of the course to enroll in. If `honor` or `audit` is specified, learners are instantly enrolled. If any other mode is specified, learners are sent to the payment/verification flow.
- `email_opt_in`: Can be `true` or `false`. If `true`, this indicates that the user has consented to receiving marketing emails from Open edX and the course partner.

The auto-enrollment feature can be combined with *hinted sign in*, if you create a URL such as the following example.

```
https://courses.example.com/account/finish_auth?course_id=course-v1%3AedX%2BDemoX%2BDemo_Course&enrollment_action=enroll&email_opt_in=false&tpa_hint=oa2-facebook
```

Secondary Providers

When you configure an OAuth2 or SAML provider in the Django admin, there is a boolean option to mark the provider as a **Secondary** provider. Normally, third party sign in providers are displayed on the Open edX sign in and register pages. However, secondary providers are not displayed directly on the sign in or register pages. Instead, users see a button that says “Use my institution/campus credentials”. Clicking that will bring up a separate menu that lists all the secondary providers.

The intended use case of this feature is to keep the sign in/register page from being cluttered with too many buttons.

SAML Attribute Requirements

When you integrate Open edX with a SAML provider, you can allow only some users to sign in based on some criteria. For example, organizations may not want alumni or guest users to be able to sign in to Open edX using their SAML provider, even though those users have valid sign in credentials for the organization.

Users can be filtered based on `eduPersonEntitlement` attributes (supported out of the box), or other attributes (requires custom code). For details on how this can be set up, refer to [this edx-code mailing list post](#).

4.23.6 Eliminating PII From Third-Party Authentication

Open edX sites and Open edX Edge do not require any personally identifiable information (PII) about learners during third-party authentication. PII is information that can be used to reveal an individual’s identity, such as a name. Your identity provider (IdP) service can send only non-personal identifiers to create Open edX site accounts for learners. If you configure third-party authentication in this way, the Open edX site never receives PII from your organization.

- *Creating Open edX Learner Accounts Without Transmitting PII*
- *Minimizing PII in Account Profiles*
- *PII Considerations with SAML*

Note: The types of information that constitute PII and requirements for handling it depend on the laws and regulations that apply to your organization. The information in this documentation is intended to explain how Open edX sites can be configured to handle learner data. You cannot rely on this documentation as a source of legal guidance.

Creating Open edX Learner Accounts Without Transmitting PII

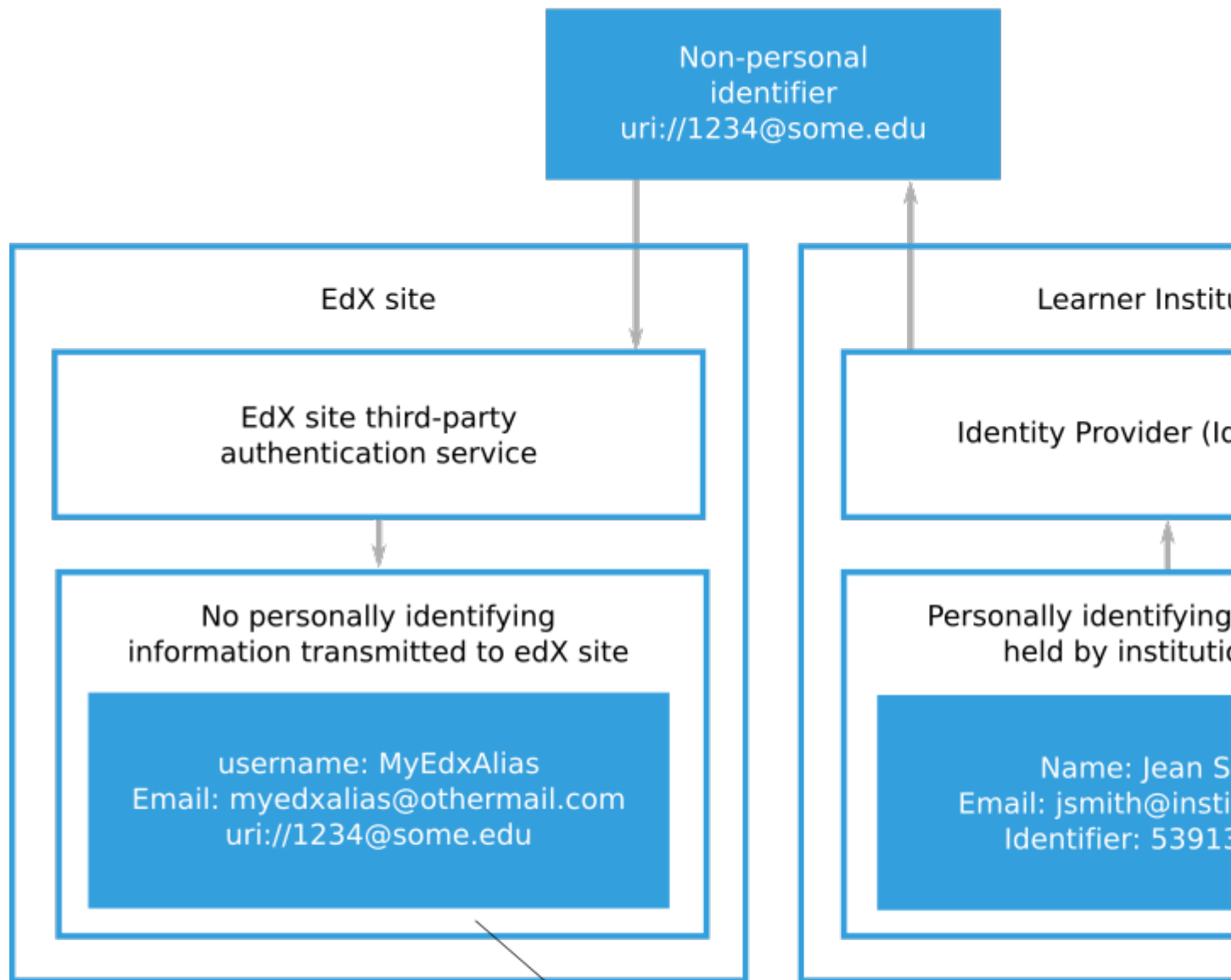
After you have configured a third-party authentication profile for an Open edX site, the site’s sign-in page will display a button for users to access your IdP service. Learners will access your IdP service and authenticate using their organization credentials. Your IdP service will direct learners back to the Open edX site’s sign-in screen with an authentication token.

The IdP authentication token includes an identifying string for a learner. This identifier serves as a link between the identifying information that your organization maintains for a learner and the Open edX account for that learner. The identifying string does not need to include any PII for the learner. The identifying string is sometimes referred to as an opaque identifier because it does not make the identity of the learner visible.

When a learner uses third-party authentication to sign in to an Open edX site for the first time, the Open edX site creates a learner account. The new Open edX learner account is permanently associated with the identifying string included in the IdP authentication token. Learners will be prompted to create profiles by entering Open edX usernames, email

addresses, and other information in their accounts. Learners can take steps to minimize the PII in their profiles. For more information, see *Minimizing PII in Account Profiles*.

The following diagram shows how an IdP can direct a learner to an Open edX site to create a learner account, without transmitting any PII.

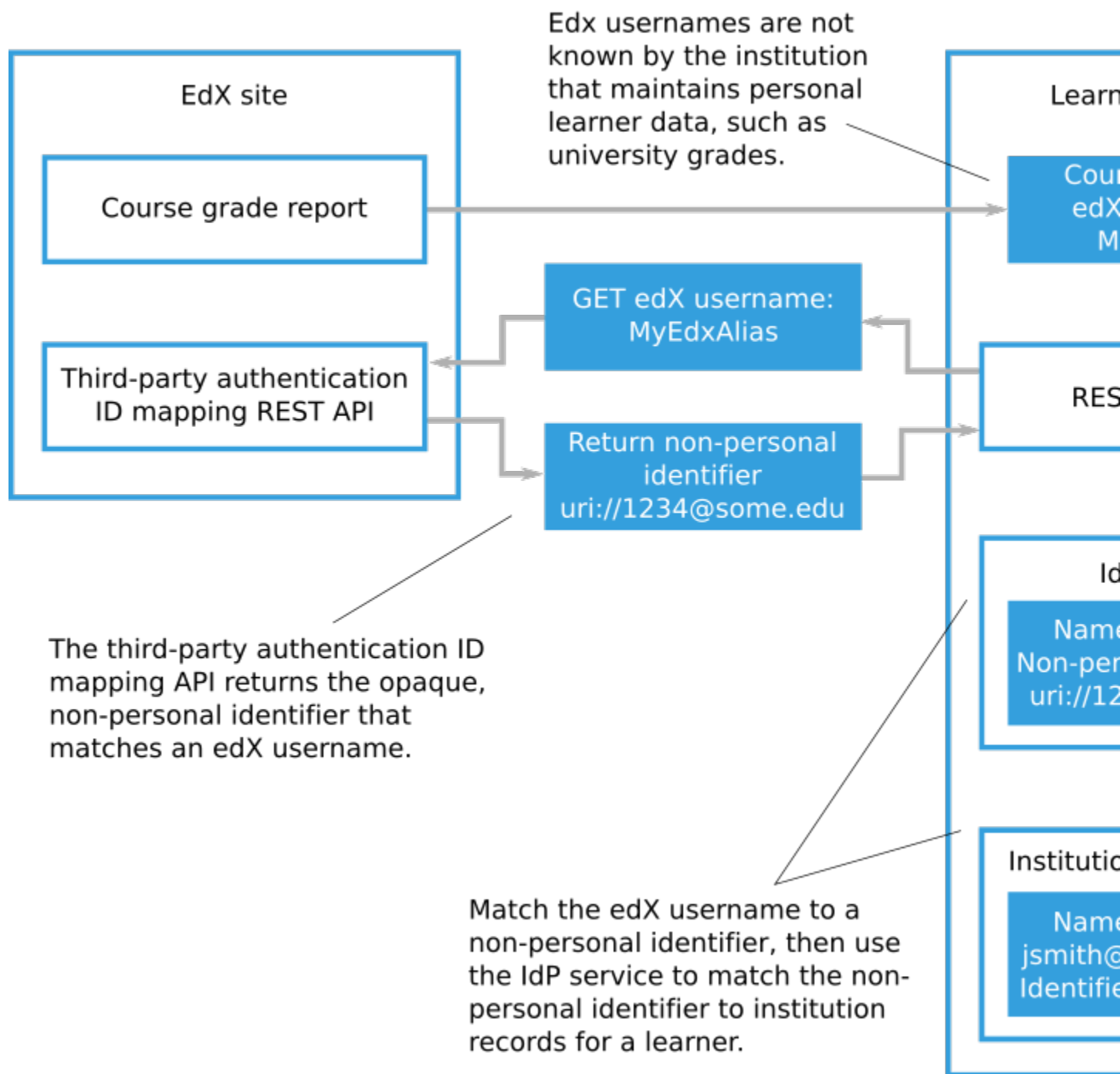


Learners can enter non-identifying information in their edX profiles, after the IdP directs them to the edX site. The opaque identifier sent by the IdP remains associated with the edX learner profile.

You can download a report containing grades for all learners in the Open edX courses that you run. The report includes the Open edX username for each learner enrolled in the course, but the usernames may not correspond to

learner records that your organization maintains. If you need to match the scores for Open edX site learners to the learner records that your organization maintains, you can use the third- party authentication ID mapping REST API to retrieve the user ID SAML attribute and matching Open edX username for a learner. For more information about grade reports for Open edX courses that you run, see [Generate a Grade Report for All Learners in a Course](#).

The following diagram shows how an organization that uses third-party authentication can match non-personally identifying Open edX learner usernames with the records that the organization holds for those learners.



Minimizing PII in Account Profiles

When your IdP directs a learner to an Open edX site for the first time, the learner enters information to create an Open edX site account. The basic information required for an Open edX site account is an email address, full name, public username, password, and country. Learners may also provide additional personal details such as gender, year of birth, and educational background. While course teams have access to full registration information for learners enrolled in their courses, only public usernames are used to identify learners in course discussions and other public-facing course interactions.

To minimize PII stored on an Open edX site, learners can limit the information in their Open edX account profiles to the basic information required for an Open edX site account. Additionally, learners may use random or nondescript public usernames and create non-identifying email addresses to receive course updates.

If you want to avoid transmitting PII for the Open edX learner accounts that use third-party authentication, you should not include personally identifying information in the authentication token. The only piece of information that is required in the authentication token is the user ID, which should not be personally identifying.

For more information about configuring the information in a third-party authentication token, see [Configure the SAML Identity Provider](#).

PII Considerations with SAML

Organizations that use SAML to integrate their IdPs with Open edX need to understand what personal information is provided to Open edX when users sign in using SAML.

When a user uses SAML or Shibboleth to sign in to Open edX, the IdP authoritatively asserts pieces of information about the user.

Each piece of information is called a SAML attribute. Each attribute has a name and a formal identifier that is called an object identifier (OID) uniform resource name (URN).

The following table lists some common attributes.

Name	OID URN	Example Value
User ID	urn:oid:0.9.2342.19200300.100.1.1	123456789
Full Name	urn:oid:2.5.4.3	Donna Noble
First Name	urn:oid:2.5.4.42	Donna
Last Name	urn:oid:2.5.4.4	Noble
Email	urn:oid:0.9.2342.19200300.100.1.3	dnoble@school.edu
eduPersonPrincipalName	urn:oid:1.3.6.1.4.1.5923.1.1.1.6	dnoble@school.edu
eduPersonEntitlement	urn:oid:1.3.6.1.4.1.5923.1.1.1.7	urn:mace:school.edu:confocalMicroscope

For more information about `eduPersonPrincipalName` and `eduPersonEntitlement`, see [eduPersonPrincipalName and eduPersonEntitlement on the eduPerson Object Class Specification page](#).

When a new user uses SAML to sign in to Open edX, the IdP sends information about the user to Open edX in the form of these attributes. These attributes can be used to pre-fill the registration form.

For example, when a user signs in, their server may provide the following information.

```
"urn:oid:0.9.2342.19200300.100.1.1: 123, urn:oid:2.5.4.3: John Smith,
urn:oid:0.9.2342.19200300.100.1.3: jsmith@school.edu"
```

Open edX saves the user ID because it is required to match future sign-in attempts to the correct user account. The full name (John Smith) and email (jsmith@school.edu) is pre-filled on the registration form, but if the user edits those, Open edX saves only the new changed version. (For example, if the user changes the email to jsmith@example.com, Open edX would record the user's email as jsmith@example.com and would not have any record of the jsmith@school.edu address.)

By default, Open edX expects the standard **User ID** field (urn:oid:0.9.2342.19200300.100.1.1) and uses that as the unique external ID. Open edX can accept any set of attributes that a SAML IdP sends, as long as one of the attributes is a unique, permanent identifier for the user being authenticated. The URN or OID of the unique identifier must be specified in the Open edX “Provider Configuration (SAML IdPs)” profile for the organization (part of the Django administration console) in the **User ID Attribute** field.

If you want to configure Open edX to always store all of the attributes that the external IdP sends for each user, you can enable that with a setting, as follows.

1. Sign in to the Django administration console for your base URL. For example, `http://{your_URL}/admin`.
2. Go to the **SAML Configuration** page, and then select **Add SAML Configuration**. (Do not go to the **SAML Provider Configuration** page.)
3. In the **Other Config Str** section, add the following code.

```
"EXTRA_DATA": [ "attributes" ]
```

This setting causes all the attributes to be saved in the `social_auth_usersocialauth` table's `extra_data` column for every new SAML user. This data is only accessible by accessing the database directly or by going to the **User social auths** page of the Open edX LMS Django administration console.

For organizations that want to avoid sending any personally identifiable information to Open edX during the SAML sign in or registration process, we recommend that the organization configure their SAML IdP to only send a single attribute: **a unique, permanent, opaque user identifier**. This should be a value that uniquely identifies any learner or staff member, but is different from their organization ID or any other identifier they may have.

For reporting and analytics purposes when using an opaque user identifier, the organization can use the *Third Party Auth ID Mapping API* to convert Open edX user IDs found in reports or analytics back to these opaque organization user identifiers. Organization partners can then convert each opaque user identifier back to the official learner ID.

4.23.7 Third Party Auth ID Mapping API

Open edX has an API that can be used to retrieve the association between Open edX user IDs and the unique user identifiers sent by the external platform.

When an Open edX course is used in an on-premise environment, and SAML, LTI, or other SSO is enabled to allow learners to authenticate to edX using their on-premise credentials, normally an on-premise identity provider (IdP) sends an identifier. This identifier is stored in edX and linked to the edX account. Later, when learners return from an on-premise IdP, edX can authenticate the learner using their edX account.

Because learners may create Open edX accounts using an alternative identity, instructors may have difficulty identifying Open edX learners in their courses. Grade records obtained from Open edX will only contain Open edX learner IDs, which would be unknown to the on-premise system.

This API can be used for mapping between the edX user ID and the ID provided by the IdP. This API provides information that allows instructors or course support staff to figure out the identity of the on-premise learners who are using the edX course. It also allows grade information coming from Open edX to be appropriately associated to on-premise learners for uploading to an on-premise learning management system (LMS) or learner information system (SIS).

This API is intended to be consumed by on-premise middleware, which combines the information from an on-premise system. The middleware communicates with the premise system to get information about the course and enrollment, then uses that information to control instructor access. The instructor can then perform tasks such as obtaining the edX learner's on-premise identity, uploading grades back to their LMS or SIS, and making groups based on on-premise activities.

This API also helps with the issue where learners enter incorrect email addresses and cannot activate their accounts, and then cannot authenticate with Open edX through SSO because of the inactive accounts. With this API, on-premise course support staff or instructors can provide the Open edX operator with the learner's real edX username instead of the ID that the IdP passed over to Open edX, so that the Open edX operator does not have to manually query the tables to figure out the mapping between the source system ID and the Open edX username.

Using the Third Party Auth ID Mapping API

To use the Third Party Auth ID Mapping API, follow these steps.

1. Have one or more third party auth providers enabled, and have learners who have link their accounts to one or more of those providers.
2. Set up an OAuth2 client. To do this, open the Django administration panel, select **OAuth2**, select **Client**, select **Add Client**, and then enter the following information.
 - (a) For **User**, create a dedicated user for this API.
 - (b) For **URL**, **Redirect URL**, enter a dummy value such as `http://localhost/`. The URL is not used for this API.
 - (c) For **Client Type**, specify **Confidential**.
 - (d) Leave other fields with the default values.
3. Give the new client permission to this API. To do this, go to **Third_Party_Auth**, select **Provider API Permissions**, and then select **Add Provider API Permission**.
4. Select the OAuth2 client that you just created, and select an external authentication provider. This will give the new client access to query that provider mapping.

The API is now available at `{LMS base URL}/api/third_party_auth/v0/providers/{provider_id}/users`. The API client must use OAuth2 to authenticate before that endpoint will work. The `{provider_id}` value uses the same format as described in *Hinted Sign In*.

To test the API, follow these steps.

1. Use client credentials (from the django admin) to get the access token, as follows.

```
curl --data "client_id=CLIENT_ID&client_secret=CLIENT_SECRET&grant_type=client_credentials" http://localhost:8000/oauth2/access_token

Returns: {"access_token": "c1efde84445b2f256e1c80886b3f6d46339b84ee",
↪ "token_type": "Bearer", "expires_in": 31535999, "scope": ""}
```

2. Use the access token to issue requests to the API, as in the following examples.

```
curl --header "Authorization: Bearer ACCESS_TOKEN" http://localhost:8000/↪ api/third_party_auth/v0/providers/{provider_id}/users

curl --header "Authorization: Bearer ACCESS_TOKEN" http://localhost:8000/↪ api/third_party_auth/v0/providers/{provider_id}/users?↪ username=USERNAME1,USERNAME2
```

```
curl --header "Authorization: Bearer ACCESS_TOKEN" http://localhost:8000/
↪api/third_party_auth/v0/providers/{provider_id}/users?
↪username=USERNAME1&username=USERNAME2

curl --header "Authorization: Bearer ACCESS_TOKEN" http://localhost:8000/
↪api/third_party_auth/v0/providers/{provider_id}/users?remote_id=REMOTE_
↪ID1,REMOTE_ID2

curl --header "Authorization: Bearer ACCESS_TOKEN" http://localhost:8000/
↪api/third_party_auth/v0/providers/{provider_id}/users?remote_id=REMOTE_
↪ID1&remote_id=REMOTE_ID2
```

The following example shows a return value from the API.

```
{
  "page": 1,
  "page_size": 200,
  "count": 8,
  "results": [
    {"username": "USERNAME1", "remote_id": "REMOTE_ID1"},
    {"username": "USERNAME2", "remote_id": "REMOTE_ID2"},
  ]
}
```

This section includes information for teams involved in identity management at Open edX installations, including development operations (DevOps) and information technology (IT).

4.24 Enabling Timed Exams

This topic describes how to enable the timed exams feature in your instance of Open edX.

- [Overview](#)
- [Enable Timed Exams in Studio and the Learning Management System](#)

4.24.1 Overview

Course teams can configure course subsections to limit the amount of time that learners have to complete problems in that subsection.

To use this feature on your instance of Open edX, you must enable the timed exams feature in Studio and the Learning Management System.

For information about how course teams set up timed exams, see the [Offering Timed Exams](#) topic in *Building and Running an Open edX Course*. For information about the learner experience, see [Taking a Timed Exam](#) in the *Open edX Learner's Guide*.

Note: Before proceeding, review [Guidelines for Updating the Open edX Platform](#).

4.24.2 Enable Timed Exams in Studio and the Learning Management System

To enable timed exams, you modify the `lms.env.json` and `cms.env.json` files, which are located one level above the `edx-platform` directory.

1. Set the value of `ENABLE_SPECIAL_EXAMS` in the `lms.env.json` and `cms.env.json` files to `true`.

```
# Timed exams feature flag
'ENABLE_SPECIAL_EXAMS': true,
```

2. Save the `lms.env.json` and `cms.env.json` files.
3. Restart the Studio (CMS) and Learning Management System (LMS) processes so that your updates are loaded.

4.25 Enabling the User Retirement Feature

There have been many changes to privacy laws (for example, GDPR or the European Union General Data Protection Regulation) intended to change the way that businesses think about and handle Personally Identifiable Information (PII).

As a step toward enabling Open edX to support some of the key updates in privacy laws, edX has implemented APIs and tooling that enable Open edX instances to retire registered users. When you implement this user retirement feature, your Open edX instance can automatically erase PII for a given user from systems that are internal to Open edX (for example, the LMS, forums, credentials, and other independently deployable applications (IDAs)), as well as external systems, such as third-party marketing services.

This section is intended not only for instructing Open edX admins to perform the basic setup, but also to offer some insight into the implementation of the user retirement feature in order to help the Open edX community build additional APIs and states that meet their special needs. Custom code, plugins, packages, or XBlocks in your Open edX instance might store PII, but this feature will not magically find and clean up that PII. You may need to create your own custom code to include PII that is not covered by the user retirement feature.

4.25.1 Implementation Overview

In the Open edX platform, the user experience is enabled by several services, such as LMS, Studio, ecommerce, credentials, discovery, and more. Personally Identifiable Identification (PII) about a user can exist in many of these services. As a consequence, to remove a user's PII, you must be able to request each service containing PII to remove, delete, or unlink the data for that user in that service.

In the user retirement feature, a centralized process (the *driver* scripts) orchestrates all of these requests. For information about how to configure the driver scripts, see [Setting Up the User Retirement Driver Scripts](#).

The User Retirement Workflow

The user retirement workflow is a configurable pipeline of building-block APIs. These APIs are used to:

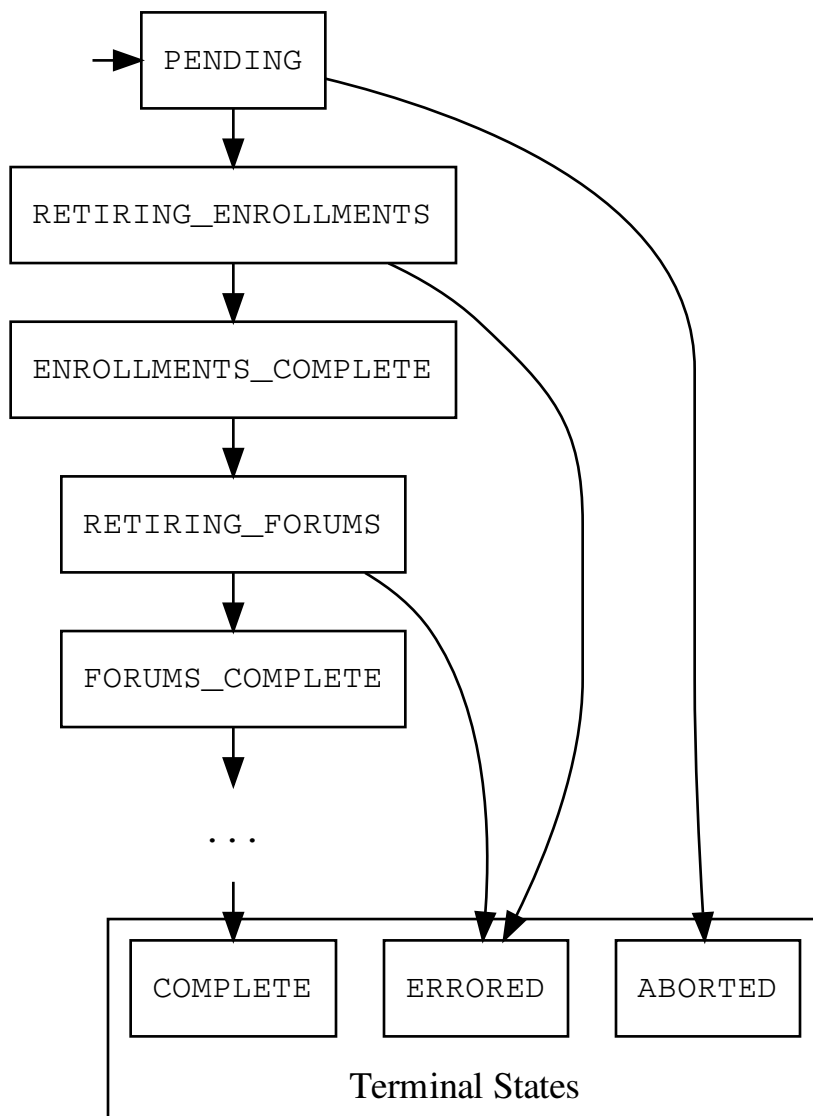
- “Forget” a retired user's PII
- Prevent a retired user from logging back in
- Prevent re-use of the username or email address of a retired user

Depending on which third parties a given Open edX instance integrates with, the user retirement process may need to call out to external services or to generate reports for later processing. Any such reports must subsequently be destroyed.

Configurability and adaptability were design goals from the beginning, so this user retirement tooling should be able to accommodate a wide range of Open edX sites and custom use cases.

The workflow is designed to be linear and rerunnable, allowing recovery and continuation in cases where a particular stage fails. Each user who has requested retirement will be individually processed through this workflow, so multiple users could be in the same state simultaneously. The LMS is the authoritative source of information about the state of each user in the retirement process, and the arbiter of state progressions, using the `UserRetirementStatus` model and associated APIs. The LMS also holds a table of the states themselves (the `RetirementState` model), rather than hard-coding the states. This was done because we cannot predict all the possible states required by all members of the Open edX community.

This example state diagram outlines the pathways users follow throughout the workflow:



Unless an error occurs internal to the user retirement tooling, a user's retirement state should always land in one of the terminal states. At that point, either their entry should be cleaned up from the `UserRetirementStatus` table or, if the state is `ERRORED`, the administrator needs to examine the error and resolve it. For more information, see [Recovering from `ERRORED`](#).

The User Experience

From the learner's perspective, the vast majority of this process is obscured. The Account page contains a new section titled **Delete My Account**. In this section, a learner may click the **Delete My Account** button and enter their password to confirm their request. Subsequently, all of the learner's browser sessions are logged off, and they become locked out of their account.

An informational email is immediately sent to the learner to confirm the deletion of their account. After this email is sent, the learner has a limited amount of time (defined by the `--cool_off_days` argument described in [Setting Up the User Retirement Driver Scripts](#)) to contact the site administrators and rescind their request.

At this point, the learner's account has been deactivated, but *not* retired. An entry in the `UserRetirementStatus` table is added, and their state set to `PENDING`.

By default, the **Delete My Account** section is visible and the button is enabled, allowing account deletions to queue up. The `ENABLE_ACCOUNT_DELETION` feature in django settings toggles the visibility of this section. See [Django Settings](#).

Third Party Auth

Learners who registered using social authentication must first unlink their LMS account from their third-party account. For those learners, the **Delete My Account** button will be disabled until they do so; meanwhile, they will be instructed to follow the procedure in this help center article: [How do I link or unlink my edX account to a social media account?](#).

4.25.2 Setting Up User Retirement in the LMS

This section describes how to set up and configure the user retirement feature in the Open edX LMS.

Django Settings

The following Django settings control the behavior of the user retirement feature. Note that some of these settings values are lambda functions rather than standard string literals. This is intentional; it is a pattern for defining *derived* settings specific to Open edX. Read more about it in [openedx/core/lib/derived.py](#).

Setting Name	Default	Description
RE-TIRED_USERNAME_PREFIX	'retired__user_'	The prefix part of hashed usernames. Used in RETIRED_USERNAME_FMT.
RE-TIRED_EMAIL_PREFIX	'retired__user_'	The prefix part of hashed emails. Used in RETIRED_EMAIL_FMT.
RE-TIRED_EMAIL_DOMAIN	'retired.invalid'	The domain part of hashed emails. Used in RETIRED_EMAIL_FMT.
RE-TIRED_USERNAME_FMT	lambda settings: RETIRED_USERNAME_PREFIX + '{}'	The username field for a retired user gets transformed into this format, where {} is replaced with the hash of their username.
RE-TIRED_EMAIL_FMT	lambda settings: RETIRED_EMAIL_PREFIX + '{}@' + settings. RETIRED_EMAIL_DOMAIN	The email field for a retired user gets transformed into this format, where {} is replaced with the hash of their email.
RE-TIRED_USER_SALTS	None	A list of salts used for hashing usernames and emails. Only the last item in this list is used as a salt for all new retirements, but historical salts are preserved in order to guarantee that all hashed usernames and emails can still be checked. The default value MUST be overridden!
RETIREMENT_SERVICE_WORKER_USERNAME	'RETIREMENT_SERVICE_USER'	The username of the retirement service worker.
RETIREMENT_STATES	See <code>lms/envs/common.py</code> in the <code>RETIREMENT_STATES</code> setting	A list that defines the name and order of states for the retirement workflow. See <i>Retirement States</i> for details.
FEATURES['ENABLE_ACCOUNT_DELETION']	True	Whether to display the “Delete My Account” section the account settings page.

Retirement States

The state of each user’s retirement is stored in the LMS database, and the state list itself is also separately stored in the database. We expect the list of states will be variable over time and across different Open edX installations, so it is the responsibility of the administrator to populate the state list.

The default states are defined in `lms/envs/common.py` in the `RETIREMENT_STATES` setting. There must be, at minimum, a `PENDING` state at the beginning, and `COMPLETED`, `ERRORED`, and `ABORTED` states at the end of the list. Also, for every `RETIRING_foo` state, there must be a corresponding `foo_COMPLETE` state.

Override these states if you need to add any states. Typically, these settings are set in `lms.envs.json`.

After you have defined any custom states, populate the states table with the following management command:

```
$ ./manage.py lms --settings=<your-settings> populate_retirement_states

All states removed and new states added. Differences:
  Added: set([u'RETIRING_ENROLLMENTS', u'RETIRING_LMS', u'LMS_MISC_COMPLETE', u
↔ 'RETIRING_LMS_MISC', u'ENROLLMENTS_COMPLETE', u'LMS_COMPLETE'])
  Removed: set([])
  Remaining: set([u'ERRORED', u'PENDING', u'ABORTED', u'COMPLETE'])
States updated successfully. Current states:
PENDING (step 1)
RETIRING_ENROLLMENTS (step 11)
```

```

ENROLLMENTS_COMPLETE (step 21)
RETIRING_LMS_MISC (step 31)
LMS_MISC_COMPLETE (step 41)
RETIRING_LMS (step 51)
LMS_COMPLETE (step 61)
ERRORED (step 71)
ABORTED (step 81)
COMPLETE (step 91)

```

In this example, some states specified in settings were already present, so they were listed under `Remaining` and were not re-added. The command output also prints the `Current states`; this represents all the states in the states table. The `populate_retirement_states` command is idempotent, and always attempts to make the states table reflect the `RETIREMENT_STATES` list in settings.

Retirement Service User

The user retirement driver scripts authenticate with the LMS and IDAs as the retirement service user with oauth client credentials. Therefore, to use the driver scripts, you must create a retirement service user, and generate a DOT application and client credentials, as in the following command.

```

app_name=retirement
user_name=retirement_service_worker
./manage.py lms --settings=<your-settings> manage_user $user_name $user_name@example.
↪com --staff --superuser
./manage.py lms --settings=<your-settings> create_dot_application $app_name $user_name

```

..note:: The client credentials (client ID and client secret) will be printed to the terminal, so take this opportunity to copy them for future reference. You will use these credentials to configure the driver scripts. For more information, see *Setting Up the User Retirement Driver Scripts*.

The retirement service user needs permission to perform retirement tasks, and that is done by specifying the `RETIREMENT_SERVICE_WORKER_USERNAME` variable in Django settings:

```

RETIREMENT_SERVICE_WORKER_USERNAME = 'retirement_service_worker'

```

Django Admin

The Django admin interface contains the following models under `USER_API` that relate to user retirement.

Name	URI	Description
Retirement States	/admin/user_api/retirementstate/	Represents the table of states defined in <code>RETIREMENT_STATES</code> and populated with <code>populate_retirement_states</code> .
User Retirement Requests	/admin/user_api/userretirementrequest/	Represents the table that tracks the user IDs of every learner who has ever requested account deletion. This table is primarily used for internal bookkeeping, and normally isn't useful for administrators.
User Retirement Statuses	/admin/user_api/userretirementstatus/	Model for managing the retirement state for each individual learner.

In special cases where you may need to manually intervene with the pipeline, you can use the User Retirement Statuses management page to change the state for an individual user. For more information about how to handle these cases, see *Handling Special Cases*.

4.25.3 Setting Up the User Retirement Driver Scripts

Tubular ([edx/tubular on github](#)) is a repository of Python 3 scripts designed to plug into various automation tooling. Included in Tubular are two scripts intended to drive the user retirement workflow.

scripts/get_learners_to_retire.py Generates a list of users that are ready for immediate retirement. Users are “ready” after a certain number of days spent in the `PENDING` state, specified by the `--cool_off_days` argument. Produces an output intended for consumption by Jenkins in order to spawn separate downstream builds for each user.

scripts/retire_one_learner.py Retires the user specified by the `--username` argument.

These two scripts share a required `--config_file` argument, which specifies the driver configuration file for your environment (for example, production). This configuration file is a YAML file that contains LMS auth secrets, API URLs, and retirement pipeline stages specific to that environment. Here is an example of a driver configuration file.

```
client_id: <client ID for the retirement service user>
client_secret: <client secret for the retirement service user>

base_urls:
  lms: https://courses.example.com/
  ecommerce: https://ecommerce.example.com/
  credentials: https://credentials.example.com/

retirement_pipeline:
  - ['RETIRING_EMAIL_LISTS', 'EMAIL_LISTS_COMPLETE', 'LMS', 'retirement_retire_
↪mailings']
  - ['RETIRING_ENROLLMENTS', 'ENROLLMENTS_COMPLETE', 'LMS', 'retirement_unenroll']
  - ['RETIRING_LMS_MISC', 'LMS_MISC_COMPLETE', 'LMS', 'retirement_lms_retire_misc']
  - ['RETIRING_LMS', 'LMS_COMPLETE', 'LMS', 'retirement_lms_retire']
```

The `client_id` and `client_secret` keys contain the oauth credentials. These credentials are simply copied from the output of the `create_dot_application` management command described in *Retirement Service User*.

The `base_urls` section in the configuration file defines the mappings of IDA to base URLs used by the scripts to construct API URLs. Only the LMS is mandatory here, but if any of your pipeline states contain API calls to other services, those services must also be present in the `base_urls` section.

The `retirement_pipeline` section defines the steps, state names, and order of execution for each environment. Each item is a list in the form of:

1. Start state name
2. End state name
3. IDA to call against (LMS, ECOMMERCE, or CREDENTIALS currently)
4. Method name to call in Tubular’s `edx_api.py`

For example: `['RETIRING_CREDENTIALS', 'CREDENTIALS_COMPLETE', 'CREDENTIALS', 'retire_learner']` will set the user’s state to `RETIRING_CREDENTIALS`, call a pre-instantiated `retire_learner` method in the `CredentialsApi`, then set the user’s state to `CREDENTIALS_COMPLETE`.

Examples

The following are some examples of how to use the driver scripts.

Set Up Environment

Set up your execution environment.

```
git clone https://github.com/edx/tubular.git
cd tubular
virtualenv --python=`which python3` venv
source venv/bin/activate
```

List of Targeted Learners

Generate a list of learners that are ready for retirement (those learners who have selected and confirmed account deletion and have been in the PENDING state for the time specified `cool_off_days`).

```
mkdir learners_to_retire
scripts/get_learners_to_retire.py \
  --config_file=path/to/config.yml \
  --output_dir=learners_to_retire \
  --cool_off_days=5
```

Run Retirement Script

After running these commands, the `learners_to_retire` directory contains several INI files, each containing a single line in the form of `USERNAME =<username-of-learner>`. Iterate over these files while executing the `retire_one_learner.py` script on each learner with a command like the following.

```
scripts/retire_one_learner.py \
  --config_file=path/to/config.yml \
  --username=<username-of-learner-to-retire>
```

Using the Driver Scripts in an Automated Framework

At edX, we call the user retirement scripts from [Jenkins](#) jobs on one of our internal Jenkins services. The user retirement driver scripts are intended to be agnostic about which automation framework you use, but they were only fully tested from Jenkins.

For more information about how we execute these scripts at edX, see the following wiki articles:

- [User Retirement Jenkins Implementation](#)
- [How to: retirement Jenkins jobs development and testing](#)

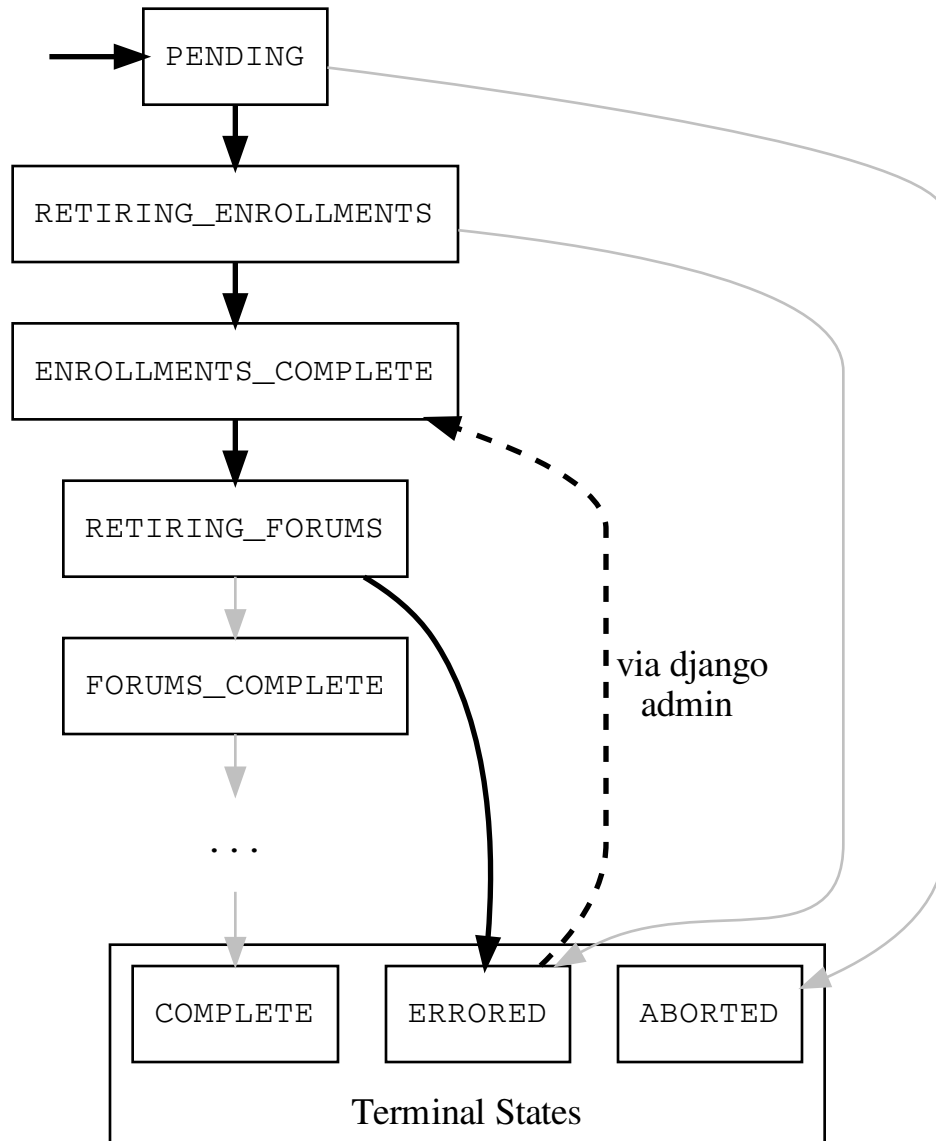
And check out the Groovy DSL files we use to seed these jobs:

- `platform/jobs/RetirementJobs.groovy` in `edx/jenkins-job-dsl`
- `platform/jobs/RetirementJobEdxTriggers.groovy` in `edx/jenkins-job-dsl`

4.25.4 Handling Special Cases

Recovering from ERRORED

If a retirement API indicates failure (4xx or 5xx status code), the driver immediately sets the user’s state to ERRORED. To debug this error state, check the `responses` field in the user’s row in `user_api_userretirementstatus` (User Retirement Status) for any relevant logging. Once the issue is resolved, you need to manually set the user’s `current_state` to the state immediately prior to the state which should be re-tried. You can do this using the Django admin. In this example, a user retirement errored during forums retirement, so we manually reset their state from ERRORED to ENROLLMENTS_COMPLETE.



Now, the user retirement driver scripts will automatically resume this user’s retirement the next time they are executed.

Rerunning some or all states

If you decide you want to rerun all retirements from the beginning, set `current_state` to `PENDING` for all retirements with `current_state == COMPLETE`. This would be useful in the case where a new stage in the user retirement workflow is added after running all retirements (but before the retirement queue is cleaned up), and you want to run all the retirements through the new stage. Or, perhaps you were developing a stage/API that didn't work correctly but still indicated success, so the pipeline progressed all users into `COMPLETED`. Retirement APIs are designed to be idempotent, so this should be a no-op for stages already run for a given user.

Cancelling a retirement

Users who have recently requested account deletion but are still in the `PENDING` retirement state may request to rescind their account deletion by emailing or otherwise contacting the administrators directly. `edx-platform` offers a Django management command that administrators can invoke manually to cancel a retirement, given the user's email address. It restores a given user's login capabilities and removes them from all retirement queues. The syntax is as follows:

```
$ ./manage.py lms --settings=<your-settings> cancel_user_retirement_request <email-of-  
↪user-to-cancel-retirement>
```

Keep in mind, this will only work for users which have not had their retirement states advance beyond `PENDING`. Additionally, the user will need to reset their password in order to restore access to their account.

4.26 Setting Up the YouTube API Key

This topic describes how to set the YouTube API key for your instance of Open edX.

- [Overview](#)
- [Get a YouTube API Key](#)
- [Install the YouTube API Key in Open edX](#)

4.26.1 Overview

If you intend for courses on your Open edX instance to include videos that are hosted on YouTube, you must get a YouTube API key and set the key in the Open edX Platform.

The Open edX Platform uses the [YouTube Data API v3](#), which requires that the application uses an API key.

4.26.2 Get a YouTube API Key

To get the YouTube API key, follow YouTube's [instructions for obtaining authorization credentials](#). YouTube provides two different options for API keys: server keys or browser keys. You should use a **browser key** for Open edX.

Note: Before proceeding, review [Guidelines for Updating the Open edX Platform](#).

4.26.3 Install the YouTube API Key in Open edX

After you obtain a YouTube API key, you must install that key into your Open edX installation. There are two different ways you can do this.

- *Option 1: Ansible (recommended)*
- *Option 2: JSON files*

Option 1: Ansible (recommended)

Ansible is the automation system used for installing and updating Open edX. If you set your YouTube API key in Ansible's configuration file, then Ansible will make sure that the YouTube API key remains in place when you update Open edX.

To set your YouTube API key in Ansible's configuration file, follow these steps.

1. Find the **configuration** repository on your Open edX server. If you are running devstack or fullstack, the directory is `/edx/app/edx_ansible/edx_ansible`.
2. In that repository, open the `playbooks/roles/edxapp/defaults/main.yml` file in a text editor.
3. Find the line for the YouTube API key.

```
EDXAPP_YOUTUBE_API_KEY: "PUT_YOUR_API_KEY_HERE"
```

Replace `PUT_YOUR_API_KEY_HERE` with your YouTube API key. Ensure that the YouTube API key is within by quotation marks.

4. Save and close the file.
5. Run Ansible so that it applies your YouTube API key to your Open edX installation.

For example, if you are running the Open edX Cypress release, run the following command.

```
/edx/bin/update edx-platform named-release/cypress
```

Option 2: JSON files

Ansible outputs information to several JSON files used by Open edX. If you prefer not to edit the Ansible configuration, you can edit these files directly.

However, every time you update Open edX, your edits will be overwritten by Ansible. As a result, we recommend setting your YouTube API key in Ansible's configuration instead.

To set your YouTube API key by editing JSON files, complete the following steps.

1. Find the **edx-platform** repository on your Open edX server. If you are running devstack or fullstack, the directory is `/edx/app/edxapp/edx-platform`.
2. In the directory *above* your repository, there should be several JSON files, including `lms.auth.json` and `cms.auth.json`. If you are running devstack or fullstack, the directory is `/edx/app/edxapp`.
3. Open the `lms.auth.json` file in your text editor.
4. Find the line for the YouTube API key.

```
"YOUTUBE_API_KEY": "PUT_YOUR_API_KEY_HERE",
```

Replace `PUT_YOUR_API_KEY_HERE` with your YouTube API key. Verify that the YouTube API key is between the quotation marks.

5. Save and close the file.
6. Open the `cms.auth.json` file and make the same change. If that line does not exist in this file, create it.
7. Save and close the file.

4.27 Installing an XBlock

The XBlock framework allows developers to expand the Open edX platform by building different learning experiences and deploying them as XBlocks. Before course teams can use an XBlock in courses running on an instance of the Open edX platform, both of the following tasks must be completed.

- A system administrator installs the XBlock in the instance of the Open edX platform.
- Course teams enable the XBlock in the specific courses that will use it.

To install an XBlock, follow these steps.

1. Obtain the GitHub location and commit, or PyPi package name and version, for the XBlock.
2. Run `pip` along with either the GitHub link to the XBlock or the PyPi package name.

An example of the GitHub link that installs the Oppia XBlock follows.

```
pip install git+https://github.com/oppia/xblock.  
↪git@9f6b95b7eb7dbabb96b77198a3202604f96adf65#egg=oppia-xblock==0.0.0
```

An example of the PyPi package name that installs the Peer Instruction XBlock follows.

```
pip install ubcpi-xblock==0.4.4
```

The course teams that want to include components that use the XBlock can then enable the XBlock for their courses. To do so, they add the name specified in the XBlock's `setup.py` file to each course's advanced module list. For more information, see [Enabling Additional Exercises and Tools](#).

4.28 Enabling a CDN for Course Assets

By default, all course assets are served directly from your Open edX instance. For courses with large enrollments, or courses with large assets, such as high-quality images, PDFs, or video files, this can increase not only the load on the instance but also the time it takes for learners to load the courses on their computers and mobile devices.

You can configure your Open edX instance to serve assets from a content delivery network (CDN) instead. Using a CDN offloads the work required to deliver assets from your Open edX instance.

Note: Whether you need a CDN depends on the type, and amount, of content in your courses. Not all installations need a CDN for their course assets.

- [Overview](#)
- [Guidelines for CDN Configuration](#)

- *Enable the CDN*

4.28.1 Overview

A CDN, or content delivery network, is a service that places servers all around the world, and keeps copies of content on those servers. Instead of serving files from one location, a CDN serves them from whichever server is closest to the end user. This results in faster download speeds and, ultimately, faster page loads.

4.28.2 Guidelines for CDN Configuration

When you configure a CDN for use with your Open edX instance, you should identify your Open edX instance as the origin server.

Choose the cache expiration carefully: you want content to be cached long enough to keep the load on your Open edX instance low, but not so long that changes made to course assets are not realized within a reasonable amount of time. As a starting point, edX recommends a cache expiration period of one hour.

4.28.3 Enable the CDN

After you configure your CDN, follow these steps.

1. Sign in to the Django administration console for your base URL. For example, `http://{your_URL}/admin`.
2. In the **Static_Replace** section, next to **Asset base url configs**, select **Add**.
3. Select **Enabled**.
4. Enter the hostname for your CDN provider.

For example, if you were using CloudFront, this would look something like `d37djvu3ytnwxt.cloudfront.net`.

Be sure not to include the scheme (`http://` or `https://`) when you specify the hostname.
5. Select **Save**.

Adding edX Insights for Course Teams

The following topics provide information about installing, configuring, and running [edX Insights](#) and its dependencies in a production environment.

5.1 Options for Installing edX Insights

This topic is intended for those who are interested in running [edX Insights](#) and its dependencies in a production environment. This topic does not provide complete installation procedures for edX Insights. It presents the introductory material that is available now.

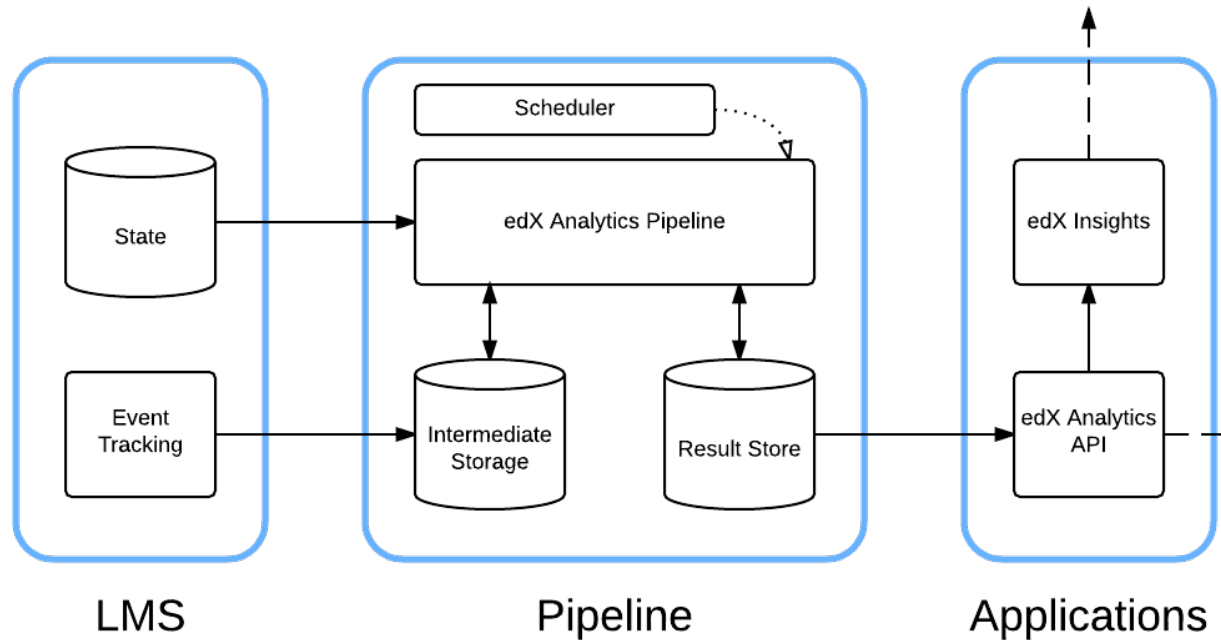
- *Overview*
- *What You Should Know Before You Start*
- *Planning Your Deployment*
- *Example Deployments*

5.1.1 Overview

Course teams use edX Insights to access data gathered from active courses. In edX Insights, course teams can display charts, summary statistics, and data tables.

The Learning Management System (LMS) gathers data about learner activity. This data is aggregated by the edX Analytics Pipeline. The aggregated data is exposed by the edX Analytics Data API. EdX Insights reads the data from the edX Analytics Data API and presents the data to course team members.

Architecture Diagram



Components

- *LMS*
- *edX Analytics Pipeline*
- *Scheduler*
- *edX Analytics Data API*
- *edX Insights*

LMS

The LMS records learner actions in tracking log files. The standard `logrotate` utility periodically compresses and copies these files into a file system that can be read by the edX Analytics Pipeline. The LMS also captures a lot of information in a MySQL database. The edX Analytics Pipeline connects directly to this database to extract information about learners.

edX Analytics Pipeline

The edX Analytics Pipeline reads the MySQL database used by the LMS as well as the tracking log files produced by the LMS. The data is processed and the resulting summary data is published to the result store. The result store is a MySQL database.

Requirements

- Hadoop version 1.0.3 or higher
- Hive version 0.11.0.2 or higher
- Sqoop version 1.4.5
- Python 2.7
- Either Debian version 6.0 or higher, or Ubuntu version 12.04 or higher
- A MySQL server version 5.6 or higher

Scheduler

The Scheduler schedules the execution of data computation tasks. Data computation tasks are run by the edX Analytics Pipeline. Data computation tasks are used to update parts of the result store.

edX Analytics Data API

The [ref:opendataapi:edX Analytics Data API](#)<edX Data Analytics API Overview> provides an HTTP interface for accessing data in the result store. Typically, the data in the result store is updated periodically by the edX Analytics Pipeline.

Requirements

Python 2.7

edX Insights

EdX Insights uses the edX Analytics Data API to present data to users. Users access the data using a supported web browser. EdX Insights communicates directly with the LMS to authenticate users, authorize users, and read course structure information.

Requirements

Python 2.7

5.1.2 What You Should Know Before You Start

To install edX Insights and deploy the edX Analytics Pipeline, you must understand the following concepts.

- Basic terminal usage.
- How the LMS has been deployed and configured.
- Basic computer network terminology.
- YAML file format.

If you plan to use Amazon Web Services, an understanding of AWS terminology is also required.

5.1.3 Planning Your Deployment

All edX Analytics services are designed to be relocatable. This means that they do not require a particular configuration of virtual servers. You are free to choose how the services should be distributed among the resources you have available.

Hadoop

Most of the computation performed by the edX Analytics Pipeline is implemented as Map Reduce jobs that must be executed by a Hadoop cluster. You can scale your Hadoop cluster based on your current and projected data sizes. Hadoop clusters can be scaled vertically and horizontally as your data grows. For very small installations of Open edX, a single virtual server should be sufficiently powerful to process your data.

Amazon's [Elastic MapReduce](#) (EMR) service offers preconfigured Hadoop clusters. If you are able to use Amazon Web Services (AWS), use of this service is recommended. Proper installation and configuration of Hadoop can be time consuming. For more information, see [Using Elastic MapReduce on AWS](#).

Additionally, vendors such as Cloudera and MapR offer simplified Hadoop administration experiences.

Hadoop is a distributed system that consists of several different services. It is worth noting that they are Java services and require a non-trivial amount of memory to run. The high memory requirement might prevent you from running all services on the same virtual server if it does not have enough memory available.

edX Applications

The edX Analytics Data API responds to a small number of requests every time a page is loaded in edX Insights. Small installations can probably host both services on the same virtual server. Larger installations should consider hosting the services on more than one virtual server. A load balancer is recommended for each service that requires more than one virtual server.

Result Store

The results of computations performed by the edX Analytics Pipeline are stored in a MySQL database. Even small installations should use a different MySQL server than the one used by the LMS. The edX Analytics Pipeline's write patterns to the result store are more I/O intensive than usual. Placing both databases on the same server can degrade the performance of the LMS.

Scheduler

Scheduling executions of the edX Analytics Pipeline can be accomplished in many different ways. Any tool that can periodically execute shell commands should work. The simplest tool that can perform this task is [cron](#). [Jenkins](#) is also a good candidate.

5.1.4 Example Deployments

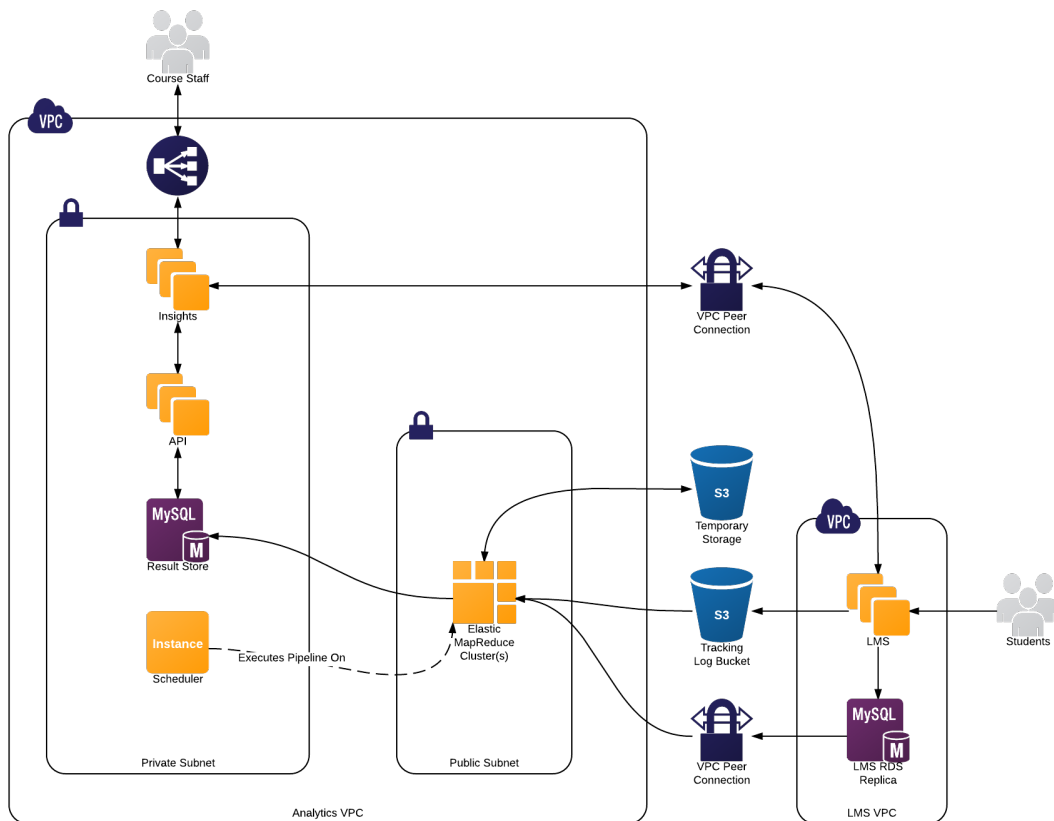
- *Small Scale Using Elastic MapReduce*
- *Large Scale Using Elastic MapReduce*
- *Large Scale Without Using Elastic MapReduce*

Small Scale Using Elastic MapReduce

A small deployment might consist of a single master node and a single core node. The Scheduler is deployed to the master node and periodically executes the edX Analytics Data Pipeline on this server. Additionally, the edX Analytics API, edX Insights and result store are deployed to the master node. These services run continuously. For more information, see *Using Elastic MapReduce on AWS*.

Large Scale Using Elastic MapReduce

A large scale deployment consists of a single master node, several core nodes, and many task nodes deployed into a public subnet of a Virtual Private Cloud.



The edX Analytics API and edX Insights are each deployed into an auto-scaling group behind an Elastic Load Balancer which terminates SSL connections and distributes the load among the application servers. The application servers are deployed into a private subnet of the Virtual Private Cloud. A single virtual server is deployed into a private subnet to host the Scheduler. The Relational Database Service is used to deploy a MySQL server into a private subnet. The MySQL database will be used as the result store. For more information, see [Using Elastic MapReduce on AWS](#).

Large Scale Without Using Elastic MapReduce

A large deployment that does not use Elastic MapReduce requires additional configuration steps to establish a properly configured environment. A Hadoop cluster is deployed. The master node is considered the node that is running the Job Tracker service for Hadoop 1.X deployments or the Resource Manager service for Hadoop 2.X deployments. Hive and Sqoop are deployed to the master node. Several servers are deployed outside of the Hadoop cluster that host the remainder of the infrastructure. The edX Analytics API and edX Insights services are each deployed to at least one server. The Scheduler is deployed to another server. A MySQL database is deployed to a server that is configured to host a relational database.

5.2 Using Elastic MapReduce on AWS

This topic provides an overview of the components and services that you set up and configure in a deployment that uses Amazon EMR. Work to prepare complete installation procedures for edX Insights is in progress.

- *Virtual Private Cloud*
- *Identity and Access Management*
- *SSH Credentials*
- *Elastic Compute Cloud*
- *Relational Database Service*
- *Scheduler Service*
- *Simple Storage Solution Buckets*
- *MySQL Connector Library*
- *Cluster Configuration File*
- *EMR Cluster*

For more information about AWS, including detailed procedures, see the [AWS Documentation](#).

Important: The tasks described by this topic rely on the use of third-party services and software. Because the services and software are subject to change by their owners, the information provided here is intended as a guideline and not as exact procedures.

5.2.1 Virtual Private Cloud

Create a Virtual Private Cloud (VPC) with at least one public subnet. A limitation of EMR running in a VPC is that clusters must be deployed into public subnets of VPCs. An existing VPC can be used if one is already available for use.

EdX recommends that you configure the VPC to have at least one private subnet in addition to the required public subnet. A private subnet is not required. For more information, see the [example configuration scenario](#) in the *Amazon Virtual Private Cloud User Guide* published by AWS.

An example configuration that includes only a [single public subnet](#) can also be found in the *Amazon Virtual Private Cloud User Guide*.

Other Considerations

- To take advantage of price fluctuations in the [spot pricing market](#), you can also deploy several public subnets in different availability zones.
- Consider that the clusters deployed using EMR will need network connectivity to a read replica of the LMS database. EdX recommends that you use the same VPC as the LMS if possible.
- While it is possible to run all of these services outside of a VPC, edX does not recommend doing so.

5.2.2 Identity and Access Management

For Amazon Identity and Access Management (IAM), create an IAM role for use by the EMR cluster. Assign all of the Elastic Compute Cloud (EC2) nodes in the cluster to this role. An option is to consider copying the contents of the [default IAM roles for Amazon EMR](#) used by AWS. The command `aws emr create-default-roles` facilitates this task. For more information, see [default IAM roles for Amazon EMR](#).

Also, you need to create a role named `emr` with the policy `AmazonElasticMapReduceforEC2Role` for the EC2 instance profile.

Make sure that an IAM user with administrative privileges is available for use.

5.2.3 SSH Credentials

Generate a secure SSH key that you use to access all AWS resources. For example, run the following command.

```
ssh-keygen -t rsa -C "your_email@example.com"
```

Upload the public key from the SSH key pair to AWS and assign a key pair name to it.

5.2.4 Elastic Compute Cloud

Ensure that at least one Amazon Elastic Compute Cloud (EC2) instance is available to host the various services. Depending on the scale of your deployment, additional instances might be necessary.

Make sure to use the secure key pair name to deploy all of the EC2 instances.

5.2.5 Relational Database Service

Deploy a MySQL 5.6 Relational Database Service (RDS) instance. This RDS instance is the result store.

Connectivity

Ensure that there is connectivity between the EC2 instance that hosts the edX Analytics API and your RDS instance.

Also, ensure that instances deployed into the public subnet of the VPC can connect to your RDS instance.

Users

Create at least the following two users on the RDS instance.

- The “pipeline” user must have permission to create databases, tables, and indexes, and issue arbitrary Data Manipulation Language (DML) commands.
- The “api” user must have read-only access.

Configure passwords for both of these users.

Other Considerations

Configure the RDS instance to use “utf8_bin” collation by default for columns in all databases and tables.

5.2.6 Scheduler Service

Establish an SSH connection to the EC2 instance within the VPC that will run the scheduler service. Then, issue the following commands from a shell running on that instance.

1. Check out the sources files from `edx-analytics-configuration`.

```
git clone https://github.com/edx/edx-analytics-configuration.git
```

2. Configure the shell to use the AWS credentials of the administrative AWS user.

```
export AWS_ACCESS_KEY_ID=<access key ID goes here>
export AWS_SECRET_ACCESS_KEY=<secret access key goes here>
```

3. Install the `AWS Command Line Interface`.

```
pip install awscli
```

5.2.7 Simple Storage Solution Buckets

Create a Simple Storage Solution (S3) bucket to hold all Hadoop logs from the EMR cluster.

```
aws s3 mb s3://<your logging bucket name here>
```

Then, create an S3 bucket to hold secure configuration files and initialization scripts.

```
aws s3 mb s3://<your configuration bucket name here>
```

5.2.8 MySQL Connector Library

Download the `MySQL connector library` from Oracle. After the download is complete, you then upload it to S3.

```
aws s3 cp /tmp/mysql-connector-java-5.1.*.tar.gz s3://<your configuration_
↳bucket name here>/
```

The `edx-analytics-configuration/batch/bootstrap/install-sqoop` script references a specific version of the `MySQL connector library`. Update this `install-sqoop` script to point to the correct version of the library in the S3 bucket. You must update the script before you continue.

Then, upload the contents of the `edx-analytics-configuration/batch/bootstrap/` directory into your configuration bucket.

```
aws s3 sync edx-analytics-configuration/batch/bootstrap/ s3://<your_
↳configuration bucket name here>/
```

5.2.9 Cluster Configuration File

Create a cluster configuration file to specify the parameters for the EMR cluster. Review the parameters that follow, and change them to specify your desired configuration. For example, review the `core` and `task` mappings and change the values for `bidprice` and `type` to meet your needs.

Then, save this file to a temporary location such as `/tmp/cluster.yml`.

```

{
  name: <your cluster name here>,
  keypair_name: <your keypair name here>,
  vpc_subnet_id: <your VPC public subnet ID here>,
  log_uri: "s3://<your logging bucket name here>",
  instance_groups: {
    master: {
      num_instances: 1,
      type: m3.xlarge,
      market: ON_DEMAND,
    },
    core: {
      num_instances: 2,
      type: m3.xlarge,
      market: SPOT,
      bidprice: 0.8
    },
    task: {
      num_instances: 1,
      type: m3.xlarge,
      market: SPOT,
      bidprice: 0.8
    }
  },
  release_label: emr-4.7.2,
  applications: [ {name: Hadoop}, {name: Hive}, {name: Sqoop-Sandbox},
↪{name: Ganglia} ],
  steps: [
    {
      type: script,
      name: Install MySQL connector for Sqoop,
      step_args: [ "s3://<your-analytics-packages-bucket>/install-sqoop",
↪"s3://<your-analytics-packages-bucket>" ],
      # You might want to set this to CANCEL_AND_WAIT while debugging step_
↪failures.
      action_on_failure: TERMINATE_JOB_FLOW
    }
  ],
  configurations: [
    {
      classification: mapred-site,
      properties:
      {
        mapreduce.framework.name: 'yarn',
        mapreduce.jobtracker.retiredjobs.cache.size: '50',
        mapreduce.reduce.shuffle.input.buffer.percent: '0.20',
      }
    },
    {
      classification: yarn-site,
      properties:
      {
        yarn.resourcemanager.max-completed-applications: '5'
      }
    }
  ],
  user_info: []
}

```


You might find you need to update Hadoop instance types and container sizes. In particular, if you encounter jobs that are running out of physical memory, you might want to choose a larger instance. If your instance is a good size but being underutilized, you might want to explicitly define larger values in the “mapred-site” configuration than would be provided by default in the instance size you are using. Here is an example of settings we use with an m3.2xlarge instance type.

```
{
  classification: mapred-site,
  properties:
  {
    mapreduce.framework.name: 'yarn',
    mapreduce.jobtracker.retiredjobs.cache.size: '50',
    mapreduce.reduce.shuffle.input.buffer.percent: '0.20',
    mapreduce.map.java.opts: '-Xmx2458m',
    mapreduce.reduce.java.opts: '-Xmx4916m',
    mapreduce.map.memory.mb: '3072',
    mapreduce.reduce.memory.mb: '6144'
  }
}
```

5.2.10 EMR Cluster

Deploy the EMR cluster.

```
EXTRA_VARS="@/tmp/cluster.yml" make provision.emr
```

Example Output

```
pip install -q -r requirements.txt

ansible-playbook --connection local -i 'localhost,' batch/provision.yml -e "
↳$EXTRA_VARS"

PLAY [Provision cluster]_
↳*****

TASK: [provision EMR cluster]_
↳*****
changed: [localhost]

TASK: [add master to group]_
↳*****
ok: [localhost]

TASK: [display master IP address]_
↳*****
ok: [localhost] => {
  "msg": "10.0.1.236"
}

TASK: [display job flow ID]_
↳*****
ok: [localhost] => {
  "msg": "j-29UUJVM8P1NPY"
}
```

```
PLAY [Configure SSH access to cluster]_
↳*****

TASK: [user | debug var=user_info]_
↳*****
ok: [10.0.1.236] => {
  "item": "",
  "user_info": []
}

TASK: [user | create the edxadmin group]_
↳*****
changed: [10.0.1.236]

TASK: [user | ensure sudoers.d is read]_
↳*****
changed: [10.0.1.236]

TASK: [user | grant full sudo access to the edxadmin group]_
↳*****
changed: [10.0.1.236]

TASK: [user | create the users]_
↳*****
skipping: [10.0.1.236]

TASK: [user | create .ssh directory]_
↳*****
skipping: [10.0.1.236]

TASK: [user | assign admin role to admin users]_
↳*****
skipping: [10.0.1.236]

TASK: [user | copy github key[s] to .ssh/authorized_keys]_
↳*****
skipping: [10.0.1.236]

TASK: [user | create bashrc file for normal users]_
↳*****
skipping: [10.0.1.236]

TASK: [user | create .profile for all users]_
↳*****
skipping: [10.0.1.236]

TASK: [user | modify shell for restricted users]_
↳*****
skipping: [10.0.1.236]

TASK: [user | create bashrc file for restricted users]_
↳*****
skipping: [10.0.1.236]

TASK: [user | create sudoers file from template]_
↳*****
changed: [10.0.1.236]
```

```
TASK: [user | change home directory ownership to root for restricted users]_
↪***
skipping: [10.0.1.236]

TASK: [user | create ~/bin directory]_
↪*****

skipping: [10.0.1.236]

TASK: [user | create allowed command links]_
↪*****

skipping: [10.0.1.236]

PLAY RECAP_
↪*****
10.0.1.236      : ok=0    changed=4    unreachable=0    failed=0
localhost     : ok=4    changed=1    unreachable=0    failed=0
```

Additional Tasks

To complete the EMR configuration, additional configuration and automation procedures are required, including scheduling jobs and automating log duplication. For more information, see the [edX Analytics Installation](#) wiki page.

Work to prepare complete installation procedures for edX Insights is in progress. This section presents the introductory, overview material that is available now.

Adding E-Commerce to the Open edX Platform

EdX uses a Django application called `ecommerce` to provide the platform with ecommerce functionality. This [E-Commerce service](#) extends [Oscar](#), an open source Django ecommerce framework, to manage the edX product catalog and handle orders for those products. The following sections describe how to install and use the E-Commerce service with the Open edX platform.

To complete the procedures that this section describes, you use both the E-Commerce Service's Django administration site and the E-Commerce Administration Tool (CAT). The CAT is a web app that is included with the E-Commerce service and enables you to configure and manage products that are associated with the courses and programs on your instance of the Open edX learning management system (LMS).

In addition to these required steps, you can add optional features to the E-Commerce service for your instance of the Open edX platform. For more information, see [Additional E-Commerce Features](#).

6.1 Install and Start the E-Commerce Service

To install and start the edX E-Commerce service, you must complete the following steps.

- *Set Up a Virtual Environment*
- *Run Migrations*
- *Configure edX OpenID Connect (OIDC)*
- *Configure a Site, Partner, and Site Configuration*
- *Start the Server*
- *Switch from ShoppingCart to E-Commerce*
- *Development Outside Devstack*

6.1.1 Set Up a Virtual Environment

1. Create or activate a Python virtual environment. You execute all of the commands described in this section within the virtualenv (unless otherwise noted).

For more information, see [Virtual Environments](#).

2. Run the following command to install dependencies.

```
$ make requirements
```

3. (Optional) Create settings overrides that you do not commit to the repository. To do this, create a file named `ecommerce/settings/private.py`. The `ecommerce/settings/local.py` file reads the values in this file, but Git ignores the file.

6.1.2 Run Migrations

To set up the `ecommerce` database, you must run migrations.

Note: Local installations use SQLite by default. If you use another database backend, make sure you update your settings and create the database, if necessary, before you run migrations.

1. To run migrations, execute the following command.

```
$ make migrate
```

When you run migrations, the E-Commerce service adds a default site to your installation.

6.1.3 Configure edX OpenID Connect (OIDC)

The E-Commerce service relies on the edX [OpenID Connect \(OIDC\)](#) authentication provider for login. OIDC is built on top of OAuth 2.0. Currently, the LMS serves as the authentication provider.

To configure the E-Commerce service to work with OIDC, complete the following procedures.

- *Create and Register a Client*
- *Designate the Client as Trusted*

Create and Register a Client

To create and register a new OIDC client, follow these steps.

1. *Start Devstack.*
2. In your browser, go to `http://127.0.0.1:8000/admin/oauth2/client/`.
3. Select **Add client**.
4. Leave the **User** field blank.
5. For **Client Name**, enter `E-Commerce Service`.
6. For **URL**, enter `http://localhost:8002/`.

7. For **Redirect URL**, enter `http://127.0.0.1:8002/complete/edx-oidc/`. This is the OIDC client endpoint.

The system automatically generates values in the **Client ID** and **Client Secret** fields.

8. For **Client Type**, select **Confidential (Web applications)**.

9. Select **Save**.

Designate the Client as Trusted

After you create your client, designate it as trusted. Trusted clients bypass the user consent form that usually appears after the system validates the user’s credentials.

To designate your client as trusted, follow these steps.

1. In your browser, go to `http://127.0.0.1:8000/admin/oauth2_provider/trustedclient/add/`.
2. In the **OAuth 2.0 clients** list, select the redirect URL for the client that you just created.
3. Select **Save**.

6.1.4 Configure a Site, Partner, and Site Configuration

To finish creating and configuring your OIDC client, you must configure a partner, site, and site configuration for the E-Commerce service to use. The site that you configure is the default site that the E-Commerce service adds when you run migrations. You must update this default site to match the domain that you will use to access the E-Commerce service. You must also set up a site configuration that contains an `oauth_settings` JSON field that stores your OIDC client’s settings, as follows.

Setting	Description	Value
SOCIAL_AUTH_EDX_OIDC_CLIENT_KEY	OAuth 2.0 client key	The Client ID field in the <i>Create and Register a Client</i> section.
SOCIAL_AUTH_EDX_OIDC_CLIENT_SECRET	OAuth 2.0 client secret	The value from the Client Secret field in the <i>Create and Register a Client</i> section.
SOCIAL_AUTH_EDX_OIDC_AUTH_URL	OAuth 2.0 authentication URL	For example, <code>http://127.0.0.1:8000/oauth2</code> .
SOCIAL_AUTH_EDX_OIDC_ID_TOKEN_DECRYPTION_KEY	OIDC ID token decryption key, used to validate the ID token	The same value as SOCIAL_AUTH_EDX_OIDC_SECRET
SOCIAL_AUTH_EDX_OIDC_ID_TOKEN_ISSUER	OIDC ID token issuer	For example, <code>http://127.0.0.1:8000/oauth2</code> .
SOCIAL_AUTH_EDX_OIDC_USER_LOGOUT_URL	User logout URL	For example, <code>http://127.0.0.1:8000/logout</code> .

To configure your default site, partner, and site configuration, use the appropriate settings module for your environment (`ecommerce.settings.devstack` for Devstack, `ecommerce.settings.production` for Fullstack) to run the following Django management command. This command updates the default site and creates a new partner and site configuration with the specified options.

```
$ sudo su ecommerce
$ python manage.py create_or_update_site --site-id=1 --site-
↪domain=localhost:8002 --partner-code=edX --partner-name='Open edX' --lms-
↪url-root=http://localhost:8000 --theme-scss-path=sass/themes/edx.scss --
↪payment-processors=cybersource,paypal --client-id=[change to OIDC client_
↪ID] --client-secret=[change to OIDC client secret]
```

Note: The `--lms-url-root` option must start with the desired protocol (e.g. `http://`).

Add Another Site, Partner, and Site Configuration

If you want to add more sites, partners, and site configurations, you can use the `create_or_update_site` command. The following options are available for this command.

Option	Required	Description	Example
<code>--site-id</code>	No	Database ID of a site you want to update.	<code>--site-id=1</code>
<code>--site-domain</code>	Yes	Domain by which you will access the E-Commerce service.	<code>--site-domain=ecommerce.example.com</code>
<code>--site-name</code>	No	Name of the E-Commerce site.	<code>--site-name='Example E-Commerce'</code>
<code>--partner-code</code>	Yes	Short code of the partner.	<code>--partner-code=edX</code>
<code>--partner-name</code>	No	Name of the partner.	<code>--partner-name='Open edX'</code>
<code>--lms-url-root</code>	Yes	URL root of the Open edX LMS instance.	<code>--lms-url-root=https://example.com</code>
<code>--theme-scss-path</code>	No	STATIC_ROOT relative path of the site's SCSS file.	<code>--theme-scss-path=sass/themes/edx.scss</code>
<code>--payment-processors</code>	No	Comma-delimited list of payment processors used on the site.	<code>--payment-processors=cyberpaypal</code>
<code>--client-id</code>	Yes	OIDC client ID.	<code>--client-id=ecommerce-key</code>
<code>--client-secret</code>	Yes	OIDC client secret.	<code>--client-secret=ecommerce-secret</code>
<code>--from-email</code>	Yes	Address from which email messages are sent.	<code>--from-email=notification@example.com</code>
<code>--enable-enrollment-codes</code>	No	Indication that specifies whether enrollment codes for seats can be created.	<code>--enable-enrollment-codes</code>
<code>--payment-support-email</code>	No	Email address displayed to user for payment support.	<code>--payment-support-email@example.com</code>
<code>--payment-support-url</code>	No	URL displayed to user for payment support.	<code>--payment-support-url=http://example.com/support</code>

To add another site, use the appropriate settings module for your environment (`ecommerce.settings`).

devstack for Devstack, `ecommerce.settings.production` for Fullstack) to run the following Django management command. This command creates a new site, partner, and site configuration with the options that you specify.

```
$ sudo su ecommerce
$ python manage.py create_or_update_site --site-domain=[change me] --partner-
↪code=[change me] --partner-name=[change me] --lms-url-root=[change me] --
↪client-id=[OIDC client ID] --client-secret=[OIDC client secret] --from-
↪email=[from email]
```

6.1.5 Start the Server

To complete the installation and start the E-Commerce service, follow these steps.

Note: Local installations use SQLite by default. If you use another database backend, make sure you update your settings and create the database, if necessary, before you run migrations.

1. (Devstack only) If you are using devstack, switch to the `ecommerce` user and use the `ecommerce.settings.devstack` settings module to run the following commands.

```
$ sudo su ecommerce
$ make serve
```

2. To run the server, execute the following command.

```
$ python manage.py runserver 8002
```

Note: If you use a different port, make sure you update the OIDC client by using the LMS Django Administration site. For more information about configuring the OIDC client, see [Configure edX OpenID Connect \(OIDC\)](#).

6.1.6 Switch from ShoppingCart to E-Commerce

Note: The ShoppingCart service was deprecated in the Dogwood release of Open edX. Ecommerce-related tasks are now handled by the E-Commerce service.

If you are upgrading from an earlier version of Open edX, follow these steps to use the E-Commerce service for commerce-related tasks instead of ShoppingCart.

1. Sign in to the LMS Administration Django site for your base URL. For example, `http://{your_URL}/admin`.
2. In the **Commerce** section, next to **Commerce configuration**, select **Add**.
3. Select **Enabled**.
4. Select **Checkout on ecommerce service**.
5. (Optional) In the **Single course checkout page** field, override the default path value of `/basket/single-item/` with your own path value.

Important: If you override the default path value, you must also change all of the code that relies on that path.

6. Set the **Cache Time To Live** value in seconds.
7. Select the site for which you want to enable the E-Commerce service.
8. Select **Save**.

6.1.7 Development Outside Devstack

If you are running the LMS in `devstack` but would prefer to run the E-Commerce service on your host, set up a reverse port-forward. This reverse port-forward allows the LMS process inside your devstack to use `127.0.0.1:8002` to make calls to the E-Commerce service running on your host. This simplifies LMS URL configuration.

To set up a reverse port-forward, execute the following command when you SSH into your devstack. Make sure that you run this command on the VM host, not the guest.

```
$ vagrant ssh -- -R 8002:127.0.0.1:8002
```

6.2 Comprehensive Theming

Any application, including Otto, can be loosely divided into two parts:

- the user interface (“how it looks”), and
- the application logic (“how it works”).

Considerations of the Otto user interface include, for example, how the products are laid out on the page, how the selectors look, how the checkout button is labelled, what sort of fonts and colors are used to display the text, and so on. Considerations of Otto’s application logic includes how Otto adjusts product price based on discount coupons, and how it records that information to be displayed in the future.

Theming consists of changing the user interface without changing the application logic. When you set up an E-Commerce website for use with your Open edX site, you probably want to use your organization’s own logo, modify the color scheme, change links in the header and footer for SEO (search engine optimization) purposes, and so on.

However, although the user interface might look different, the application logic must remain the same so that Otto continues to work properly. A well- designed theme preserves the general layout and structure of the user interface, so that users of the website still find it familiar and easy to use. Be careful about making sweeping changes to the user interface without warning: your users will be very confused!

The default Open edX theme is named “Comprehensive Theming”. You can disable Comprehensive Theming by setting `ENABLE_COMPREHENSIVE_THEMING` to `False`, as shown in this example, then applying your custom theme.

```
ENABLE_COMPREHENSIVE_THEMING = False
```

- *Theme Structure*
- *Enabling a Theme*
- *Disabling a Theme*
- *Troubleshooting*

6.2.1 Theme Structure

From a technical perspective, theming consists of overriding core templates, static assets, and Sass with themed versions of those resources.

Every theme must conform to a directory structure that mirrors the Otto directory structure.

```

my-theme
├── README.rst
├── static
│   ├── images
│   │   └── logo.png
│   ├── sass
│   │   ├── partials
│   │   │   └── utilities
│   │   │       └── _variables.scss
│   └── templates
│       ├── oscar
│       │   ├── dashboard
│       │   └── index.html
│       └── 404.html

```

Templates

Any template included in `ecommerce/templates` directory can be “themed”. However, make sure not to override class names or ID values of HTML elements within a template, as these are used by JavaScript or CSS. Overriding these names and values can cause unwanted behavior.

Static Assets

Any static asset included in `ecommerce/static` can be overridden except for the CSS files in the `ecommerce/static/css` directory. CSS styles can be overridden via Sass overrides explained below.

Caution: Theme names must be unique. The names of static assets or directories must not be same as the theme’s name, otherwise static assets will not work correctly.

Sass

Sass overrides are a little different from static asset or template overrides. There are two types of styles included in `ecommerce/static/sass`:

- base
- partials

Caution: Styles present in `ecommerce/static/sass/base` should not be overridden as overriding these could result in an unexpected behavior.

Any styles included in `ecommerce/static/sass/partials` can be overridden. Styles included in this directory contain variable definitions that are used by main Sass files. Elements of the user interface such as header/footer, background, fonts, and so on, can be updated in this directory.

6.2.2 Enabling a Theme

To enable a theme, you must first install your theme onto the same server that is running Otto. If you are using `devstack` or `fullstack` to run Otto, you must be sure that the theme is present on the Vagrant virtual machine. It is up to you where to install the theme on the server, but a good default location is `/edx/app/ecommerce/ecommerce/themes`.

Note: All themes must reside in the same physical directory.

In order for Otto to use the installed themes, you must specify the location of the theme directory in Django settings by defining `COMPREHENSIVE_THEME_DIRS` in your settings file, as shown in the example, where `/edx/app/ecommerce/ecommerce/themes` is the path to where you have installed the themes on your server.

```
COMPREHENSIVE_THEME_DIRS = ["/edx/app/ecommerce/ecommerce/themes", ]
```

You can list all theme directories using this setting.

After you install a theme, you associate it with your site by adding appropriate entries to the following tables.

- Site
- Site Themes

For local `devstack`, if the Otto server is running at `localhost:8002` you can enable a `my-theme` by following these steps.

1. Add a new site with the domain `localhost:8002` and the name “Otto My Theme”.
2. Add a site theme with the theme dir name `my-theme`, selecting `localhost:8002` from the site dropdown.

The Otto server can now be started, and you should see that `my-theme` has been applied. If you have overridden Sass styles and you are not seeing those overrides, then you need to compile Sass files as described in [Compiling Theme Sass](#).

6.2.3 Disabling a Theme

A theme can be disabled by removing its corresponding `Site Theme` entry using `django admin`.

Creating or Updating Site and SiteTheme

If you have already set up `COMPREHENSIVE_THEME_DIRS`, you can use the management command for adding `Site` and `SiteTheme` directly from the terminal.

```
python manage.py create_or_update_site_theme --site-domain=localhost:8002 --site-  
↪name=localhost:8002 --site-theme=my-theme
```

The `create_or_update_site_theme` command accepts the following optional arguments, listed below with examples.

- `settings`: The settings file to use. The default file is `ecommerce.settings.devstack`.

```
python manage.py create_or_update_site_theme --settings=ecommerce.settings.production
```

- **site-id:** The ID of the site that you want to update.

```
# update domain of the site with id 1 and add a new theme
# `my-theme` for this site
python manage.py create_or_update_site_theme --site-id=1 --site-domain=my-theme.
↳localhost:8002 --site-name=my-theme.localhost:8002 --site-theme=my-theme
```

- **site-domain:** The domain of the site to be created.

```
python manage.py create_or_update_site_theme --site-domain=localhost:8002 --site-
↳theme=my-theme
```

- **site-name:** The name of the site to be created. The default setting is ''.

```
python manage.py create_or_update_site_theme --site-domain=localhost:8002 --site-
↳name=localhost:8002 --site-theme=my-theme
```

- **site-theme:** The theme dir for the new theme.

```
python manage.py create_or_update_site_theme --site-domain=localhost:8002 --site-
↳name=localhost:8002 --site-theme=my-theme
```

Compiling Theme Sass

You use the management command `update_assets` to compile and collect themed Sass.

```
python manage.py update_assets
```

The `update_assets` command accepts the following optional arguments, listed below with examples.

- **settings:** The settings file to use. The default file is `ecommerce.settings.devstack`.

```
python manage.py update_assets --settings=ecommerce.settings.production
```

- **themes:** The space-separated list of themes to compile Sass for. Possible options are `all` for all themes, `no` to skip Sass compilation for themes. The default option is `all`.

```
# compile Sass for all themes
python manage.py update_assets --theme=all

# compile Sass for only given themes, useful for situations if you have
# installed a new theme and want to compile Sass for just this theme

python manage.py update_assets --themes my-theme second-theme third-theme

# skip Sass compilation for themes, useful for testing changes to system
# Sass, keeping theme styles unchanged

python manage.py update_assets --theme=no
```

- **output-style:** The coding style for compiled CSS files. Possible options are `nested`, `expanded`, `compact` and `compressed`. The default option is `nested`.

```
python manage.py update_assets --output-style='compressed'
```

- `skip-system`: This flag disables system Sass compilation.

```
# useful in cases where you have updated theme Sass, and system Sass is  
# unchanged.
```

```
python manage.py update_assets --skip-system
```

- `enable-source-comments`: This flag enables source comments in generated CSS files.

```
python manage.py update_assets --enable-source-comments
```

- `skip-collect`: Use this flag to skip the `collectstatic` call after Sass compilation.

```
# useful if you just want to compile Sass, and call ``collectstatic`` later,  
# possibly by a script
```

```
python manage.py update_assets --skip-collect
```

6.2.4 Troubleshooting

If you have gone through the preceding procedures and you are not seeing theme overrides, check the following areas.

- `COMPREHENSIVE_THEME_DIRS` must contain the path for the directory containing themes. For example, if your theme is `/edx/app/ecommerce/ecommerce/themes/my-theme` then the correct value for `COMPREHENSIVE_THEME_DIRS` is `['/edx/app/ecommerce/ecommerce/themes']`.
- The domain name for site is the name that users will put in the browser to access the site, and includes the port number. For example, if Otto is running on `localhost:8002` then the value for domain should be `localhost:8002`.
- The theme dir name is the name of the directory of your theme. For example, for our ongoing example, `my-theme` is the correct theme dir name.

6.3 Manage Static Assets

After you *configure a partner and at least one site* for the E-Commerce system to use, you must compile all static assets and move them to the correct location to be served. The edX E-Commerce service uses `django-compressor` and `RequireJS` to manage static assets.

- `django-compressor` compiles and minifies CSS and JavaScript files, and names files to facilitate cache busting after new file deployment.
- `RequireJS` manages JavaScript dependencies.

Note: The static file directories are set up so that `make static` reads the build output directory of `r.js` before it checks for assets in the `ecommerce/static/` directory. EdX does not recommend that you run `make static` locally. If you run `make static` or `r.js` locally, make sure you delete the `ecommerce/static/build` folder or that you run `make static` before you continue with development. If you do not run `make static` again, `django-compressor` ignores all changes that you make to static files.

6.3.1 Compile and Move Static Assets

To compile and move your static assets and deploy your E-Commerce service, execute the following command locally or on another server.

```
$ make static
```

If you create new pages that have RequireJS dependencies, remember to add your new JavaScript modules to the RequireJS build file for the project. This is the `build.js` file.

6.4 Create E-Commerce Products

After you *configure a partner and at least one site* for the E-Commerce service to use, and you have compiled and moved your static assets, you can create products. For more information, see the following topics.

6.4.1 Creating Products Overview

The edX platform offers several types of products. You create these products in E-Commerce web pages.

- Course seats represent an *enrollment track*. Each course seat has an associated set of attributes, such as price and certificate availability. The edX code uses course seats to determine how a given enrollment should be handled. For more information, see *Create Course Seats*.
- Coupons allow you to offer learners a discount, either percentage or fixed amount, on a course enrollment. For more information, see *Create and Manage Coupons*.
- Enrollment codes allow users to purchase bulk enrollments for a course. For more information, see *Enable and Create Enrollment Codes*.
- Programs are collections of related courses. Learners can enroll in and purchase courses separately, or you can configure programs to allow one-click purchasing of all courses in a program. For more information, see *About Programs*.

Start the E-Commerce Service

Before you can create a product, you must start the E-Commerce service on your site. Follow these steps to start the E-Commerce service.

1. In the ecommerce and LMS configuration files (`/edx/etc/ecommerce.yml` and `/edx/app/edxapp/lms.auth.json`, respectively), verify the following settings.

Note: If you are using `devstack`, these values are set correctly for you. However, edX recommends that you verify these values.

- The `EDX_API_KEY` value in the LMS file must be the same as the `EDX_API_KEY` value in the ecommerce file. If the values differ, change the value in the LMS file to match the ecommerce file.
 - The `ECOMMERCE_API_SIGNING_KEY` value in the LMS file must be the same as the `JWT_SECRET_KEY` value in the ecommerce file. If the values differ, change the value in the LMS file to match the ecommerce file.
2. On `devstack`, start the E-Commerce server on port 8002, and start the LMS on port 8000.

6.4.2 Create Course Seats

A course seat represents an enrollment track, sometimes called an enrollment mode. For information about the enrollment tracks that edX offers, see *enrollment track*.

You create course seats by creating a course on the **Create New Course** page in the E-Commerce Course Administration tool, which is located at `http://localhost:8002/courses/`. After you create a course, the E-Commerce service creates the course seats that are associated with that course.

To create a course seat, follow these steps.

1. Start the E-Commerce Service on your site. For details, see *Start the E-Commerce Service*.
2. In a browser on your E-Commerce server, go to `http://localhost:8002/courses/` to access the E-Commerce Course Administration tool.
3. On the **Courses** page, select **Add New Course**.
4. On the **Create New Course** page, enter the following information for your course.
 - Course ID
 - Course Name
 - Course Type
 - Course Seats
 - Bulk *Enrollment Code* Yes/No

For **Course Type**, select a course type and the options for that course type.

- If you select **Free (Audit)**, you must specify whether you want to allow honor code learners to earn an honor code certificate. To do this, select **Yes** under **Include Honor Seat**.
- If you select **Verified**, you must add the following information.
 - **Price (in USD)**
 - **Upgrade Deadline**
 - **Verification Deadline**
 - **Include Honor Seat**: This option grants honor code certificates to learners who successfully complete the course.
- If you select **Professional Education**, you must add the following information.
 - **Price (in USD)**
 - **ID Verification Required?**
 - **Upgrade Deadline**
 - **Verification Deadline**: This option is required if you select **Yes** for **ID Verification Required?**
- If you select **Credit**, you must add the following information.
 - **Price (in USD)**: The price for a verified certificate.
 - **Upgrade Deadline**
 - **Credit Provider**
 - **Price (USD)**: The price for course credit.
 - **Credit Hours**
 - **Upgrade Deadline**

- **Verification Deadline**
- **Include Honor Seat:** This option grants honor code certificates to learners who successfully complete the course.

5. Select **Create Course**.

6.4.3 Create and Manage Coupons

This topic covers how to create and distribute coupons and their associated coupon codes. You can use coupons to provide discounted or free course enrollments, also called “course seats”, to your learners.

- *Create a Coupon*
- *Download Coupon Information*
- *Edit a Coupon*
- *Deactivate a Coupon*
- *Distribute Coupon Codes to Learners*
- *View the Invoice for a Coupon*

Create a Coupon

To create a coupon, you create one or more coupon codes that learners use to receive their discounts. You then use your email system to distribute the discount or enrollment codes for that coupon.

Creating a coupon code has several steps.

- *Enter Basic Information*
- *Specify the Coupon Code Type*
- *Specify Courses*
- *Specify Usage Limitations*
- *Specify Invoicing Options*
- *Finish and Review Coupon*

You create coupons and their associated discount or enrollment codes on the **Create New Coupon** page in the E-Commerce Coupon Administration tool, which is located at <http://localhost:8002/coupons/>. In the tool, you enter basic information and select the options for your coupon.

Enter Basic Information

Each coupon requires some basic information. To enter basic information for your coupon, follow these steps.

1. Start the E-Commerce Service on your site. For details, see *Start the E-Commerce Service*.
2. In a browser on your E-Commerce server, go to <http://localhost:8002/coupons/> to access the E-Commerce Coupon Administration tool.

3. On the **Coupon Codes** page, select **Create Coupon**.
4. On the **Create New Coupon** page, enter the following information.
 - **Coupon Name:** The name you want to give the coupon, such as “January 15% Promotion”. The name must have fewer than 30 characters.
 - **Category:** A list of possible classifications for your coupon, such as “Course Promotion” and “Financial Assistance”. Categories can help you keep track of your coupons.
 - **Valid from** and **Valid until:** The dates and times when the discount or enrollment code is valid for use. The time zone is set to Universal Coordinated Time (UTC).
 - **Email Domains:** Optional. A list of comma separated domains. If specified, only users registered with an email address that matches can use the coupon. If null, any user can use the coupon.
 - **Discount Value:** The discount that you want to apply to the course fee, specified as a percentage between 1% and 99% or a fixed amount in U.S. dollars.
 - **Client:** The name of the organization that you create the codes for. Note that the current system cannot automatically send this organization an invoice for the codes you create. For more information, see [Specify Invoicing Options](#) and [View the Invoice for a Coupon](#).
 - **Note** (optional): Any additional information that you want to add to your coupon, such as why the coupon was created. The note is visible on the coupon page in the coupon administration tool and in the .csv file for the coupon. It is not visible to learners.
 - **Tax Deducted Source (TDS):** This field is not yet active.

After you specify basic information for your coupon, you must specify the coupon code type.

Specify the Coupon Code Type

In addition to entering basic information, you must specify a coupon code type. The coupon code type is either “enrollment code” or “discount code”.

- An enrollment code covers the entire fee for a course seat.
- A discount code offers between 1% and 99% or a fixed dollar amount off the fee for a course seat.

A “course seat” is a single course enrollment in a specific course track.

To specify the coupon code type, follow these steps.

1. For **Code Type**, select **Enrollment Code** if you want the coupon code to cover the entire course fee, or select **Discount Code** if you want to provide a discount for the course.
2. If you selected **Enrollment Code**, locate the **Number of Codes** field, and then enter the number of enrollment codes that you want to create.

If you selected **Discount Code**, the following fields are visible.

- **Discount per Code** (required): Enter the percent or U.S. dollar amount of the discount that you want to offer, then select **Percent** or **Fixed**. Do not add a percent sign or a dollar sign.
- **Code** (optional): If you want to specify your own discount code, such as SCHOLAR15, enter the code that you want in this field. This code can have 1 to 16 characters.

If you want the system to generate a discount code for you, leave this field empty, and then enter the number of discount codes that you want to create in the **Number of Codes** field.

After you complete this step, you must specify the courses that you want your coupon to apply to.

Specify Courses

In addition to specifying your coupon's code type, you must specify the courses for your coupon. Your coupon can apply to a single course or to multiple courses.

Note: If you want your coupon to apply to multiple courses, you must use the edX Course Catalog API. The Course Catalog API is in beta and is not documented or fully supported.

To specify the courses for your coupon, follow these steps.

1. On the **Create New Coupon** page, select **Single Course** or **Multiple Courses**.
2. Specify the courses for your coupon.

If you selected **Single Course**, follow these steps.

- (a) For **Course ID**, enter the ID of the course that you want. To find the course ID, open the course administration tool at `http://localhost:8002/courses`, select your course name in the list of courses, and then locate the **Course ID** field on the page for the course.
- (b) For **Seat Type**, select the type of seat for the coupon. A "seat" is a single course enrollment in a specific course track. For example, the seat type might be "verified" or "professional". The options for this field appear after you enter a valid value in the **Course ID** field.

If you selected **Multiple Courses**, follow these steps.

- (a) In the **Valid for** field, enter a query to retrieve the courses that you want. For more information about creating a query, see [Create a Query for Multiple Courses](#).

To see a preview of the results that your query will return, enter your query in the **Valid for** field, and then select **Preview**. A dialog box opens that lists the courses in your query results.
- (b) For **Seat Types**, select **Non-credit** or **Credit**.

If you select **Non-credit**, you must also select **Verified**, **Professional**, or both.

After you complete these steps, you must *specify usage limitations for your coupon*.

Create a Query for Multiple Courses

Note: To create a query, you must use the edX Course Catalog API. The Course Catalog API is in beta and is not documented or fully supported.

The coupon administration tool uses queries to return a catalog of courses. The coupons that you create apply to each course in that catalog.

These queries use the following syntax.

```
field_name:search_terms
```

Your query can contain operators such as quotation marks ("), asterisks (*), and hyphens (-).

For example, your query might resemble one of the following queries.

```
org:(Company OR University)
org:(-Company OR -University)
```

```
start:[2016-01-01 TO 2016-12-31]
```

```
number:6.002x*
```

For more information about queries, including syntax and operators, see [Query string syntax](#).

The following table lists the field names that you can use in your query, along with a description for each field name.

Field Name	Description
announcement	The date the course is announced to the public, in YYYY-MM-DD format.
end	The course run end date, in YYYY-MM-DD format.
enrollment_end	The enrollment end date, in YYYY-MM-DD format.
enrollment_start	The enrollment start date, in YYYY-MM-DD format.
key	The course run key, sometimes also called the course ID.
language	The language in which the course is administered.
max_effort	The estimated maximum number of hours necessary to complete the course.
min_effort	The estimated minimum number of hours necessary to complete the course.
number	The course number (for example, 6.002x).
org	The organization associated with the course (for example, MITx).
pricing_type	The pricing for the course run. Options are <code>instructor_paced</code> or <code>self_paced</code> .
start	The course run start date, in YYYY-MM-DD format.
title	The course title.

Specify Usage Limitations

In addition to specifying courses for your coupon, you must specify usage limitations. Usage limitations control whether one or more customers can use the coupon, as well as the number of times each customer can use the coupon.

To specify usage limitations, follow these steps.

1. On the **Create New Coupon** page, locate **Usage Limitations**.
2. Select one of the following options.

- **Can be used once by one customer**

If you select this option, the **Number of Codes** field is visible. In this field, specify the number of individual discount or enrollment codes you want to create. The value must be between 1 and 1000. Make sure that you create enough discount or enrollment codes so that each person receives one code.

- **Can be used once by multiple customers or**

- **Can be used multiple times by multiple customers**

If you select one of these options, the **Maximum Number of Uses** field is visible. In this field, specify the number of times customers can use the coupon code.

For example, if you want to create a single coupon code that is available for use by 10 different customers, and each customer can use the code only one time, select **Can be used once by multiple customers**, 1 for **Number of Codes**, and 10 for **Maximum Number of Uses**.

If you want to create a coupon code that is available for 10 uses, whether by one customer or multiple customers, select **Can be used multiple times by multiple customers**, 1 for **Number of Codes**, and 10 for **Maximum Number of Uses**.

After you specify usage limitations, you must specify invoicing options for your coupon.

Specify Invoicing Options

In addition to setting usage limitations for your coupon, you must specify invoicing options. You can send an invoice when you create the coupon or after one or more customers have redeemed the coupon. The invoice can be for the total amount or for part of the total amount.

To specify the way you want to invoice your client, follow these steps.

1. On the **Create New Coupon** page, locate **Invoice Type**.
2. Select one of the following options.
 - **Already invoiced:** Select this option if you have already sent an invoice to your client.
If you select this option, you must also enter information in the following fields.
 - **Invoice Number:** Your internal invoice number.
 - **Invoice Amount:** The amount that you have already invoiced the client.
 - **Payment Date:** The date when payment is due from the client.
 - **Invoice after redemption:** Select this option if you will send an invoice to your client after one or more coupon codes have been redeemed.
 - **N/A:** Select this option if neither of the other options applies to your situation.

Finish and Review Coupon

- After you have *entered all the basic coupon information* and specified the options that you want, select **Create Coupon**.

When you select **Create Coupon**, the system generates one or more discount or enrollment codes as well as the URLs where users can redeem these codes.

When the system has finished generating the coupon, a page for the coupon opens. This page lists the information for your coupon, including all discount or enrollment codes for the coupon, the coupon's status, URLs where users can redeem the codes, dates the coupon is valid, and the course that the coupon applies to. You can also *download a .csv file with the coupon information* from this page.

Download Coupon Information

After you create a coupon, you can download a .csv file that lists the information that you entered when you created the coupon. The .csv file also includes additional information, such as the discount or enrollment codes that are associated with your coupon and the system-generated URL where a user can redeem each code.

1. In your browser, go to `http://localhost:8002/coupons/` to open the coupon administration tool.
2. On the **Coupon Codes** page, locate the coupon that you want in the table, and then select the name of the coupon. The page for the coupon opens.
3. On the page for the coupon, select **Download**. Your .csv file begins downloading automatically.

The .csv file for your coupon lists the following information.

Coupon Name	The name of the coupon.
Code	The 16-digit alphanumeric discount or enrollment code. The .csv file lists at least one entry for each code. The system creates an additional entry for the code each time the status of the code changes (for example, when the status changes from Active to Redeemed).
Maximum Coupon Usage	The number of times that the discount code or enrollment code that is associated with the coupon can be used. For single-use coupons, this value is 1. For multi-use coupons, this is the value that you specified in the Maximum Number of Uses field.
Redemption Count	The number of times the coupon has been redeemed. The initial value is 0, and the value is incremented each time that a discount code or enrollment code for the coupon is redeemed.
Coupon Type	The type of coupon code associated with this coupon. Possible values are Enrollment and Discount .
URL	The URL where the user redeems the coupon code.
Catalog Query (for multi-course coupons only)	The query that was used to determine which courses the coupon applies to.
Course Seat Types (for multi-course coupons only)	The seat type that the coupon applies to. For example, the seat type might be “verified” or “professional”. For more information about seat types, see <i>Specify Courses</i> .
(for single-course coupons only) Course ID	The ID of the course that the coupon applies to.
(for single-course coupons only) Organization	The organization that provides the course.
Client	The organization that purchased the coupon codes.
Category	The value that you selected for the Category field when you created the coupon.
Note	The text, if any, that you entered in the Note field when you created the coupon.
Price	The regular fee for the course.
Email Domains	The email domains allowed to use this coupon.
Invoiced Amount	The text, if any, that you entered in the Total to Invoice to Client field when you created the coupon.
Discount Percentage	The percent discount, if any, that you specified when you created the coupon.
Discount Amount	The dollar amount discount, if any, that you specified when you created the coupon.
Status	The status of the coupon. Possible values are Active , Redeemed , or Expired .
Order Number	The order number associated with the redemption of the enrollment or discount code.
Redeemed By Username	The username of the customer who redeemed the enrollment or discount code.
(for multi-course coupons only) Redeemed for Course ID	The course ID of the course for which the coupon was redeemed.
Created By	The username of the user who created the coupon.
Create Date	The date the coupon was created.
Coupon Start Date	The first date the coupon can be used.
Coupon Expiry Date	The last date the coupon can be used.

Edit a Coupon

You edit a coupon by using the coupon administration tool.

Note: You cannot edit the following fields.

- **Code Type**
 - **Course ID**
 - **Single course** or **Multiple courses**
 - **Seat Type**
 - **Usage Limitations**
 - **Number of Codes** or **Maximum Number of Uses**
-

1. In your browser, go to `http://localhost:8002/coupons/` to open the coupon administration tool.
2. On the **Coupon Codes** page, locate the coupon that you want in the table, and then select the name of the coupon. The page for the coupon opens.
3. On the page for the coupon, select **Edit Coupon**.
4. Make the changes that you want.
5. Select **Save Changes**.

Deactivate a Coupon

To deactivate a coupon, change the **Valid from** and **Valid until** date fields so that both dates are in the past. For more information, see *Edit a Coupon*.

Distribute Coupon Codes to Learners

You can distribute both discount codes and enrollment codes to learners in two ways.

- You provide a coupon code that learners enter on the **Checkout** page for the verified or professional certificate track. If you do this, edX recommends that you also provide the URL for the course About page to make signing up for the course easier.
- You provide a URL for an offer landing page. This automatically generated page presents information about the course, lets the learner know that the coupon code has been applied, and provides the opportunity for the learner to enroll. Learners can access this URL if they do not have an edX account or they are not signed in. However, learners must sign in or create an edX account to redeem the coupon and enroll in the course.

A URL for an offer landing page has the following format.

```
http://localhost:8002/coupons/offer/?code=#####
```

Note: If the coupon code is a discount code, the learner must pay any balance due before enrolling in the course for a verified or professional certificate.

To distribute the coupon code or URL to learners, you determine the coupon code or the URL for the learner to use, and then you create and send an email that includes the coupon code or the URL. For suggestions for email message text, see *Example Email Messages*.

Find a Coupon Code or URL

You can find coupon codes, whether discount codes or enrollment codes, and their associated URLs in two places: on the page for the coupon in the coupon administration tool, and in a downloadable .csv file. You can use either option to find the coupon code or URL for your learners.

Find a Code or URL on the Coupon Page

To find a coupon code or URL on the page for the coupon in the coupon administration tool, follow these steps.

1. In your browser, go to `http://localhost:8002/coupons/` to open the coupon administration tool.
2. On the **Coupon Codes** page, locate the coupon that you want in the table, and then select the name of the coupon. The page for the coupon opens.
3. On the page for the coupon, locate the table under **Codes**.
4. In the table, locate the information that you want.
 - For a coupon code that the learner will enter on the **Checkout** page, use the value in the **Code** column.
 - For an offer landing page, use the URL in the **Redemption URL** column.

Find a Code or URL in a Downloaded File

To find a coupon code or URL in the .csv file for a coupon, follow these steps.

1. *Download a .csv file* that lists the information for your coupon, and then open the .csv file.
2. In the .csv file, locate the information that you want.
 - For a coupon code that the learner will enter on the **Checkout** page, use the value in the **Code** column.
 - For an offer landing page, use the URL in the **URL** column.

Send an Email Message

After you *locate the coupon code or URL* that you want to use, you use your email system to provide that information in a message to potential learners.

Note: When you send a coupon code for a learner to use on the **Checkout** page, edX recommends that you include the About page URL for the course as well as the coupon code to help the learner enroll more easily.

Example Email Messages

You can use the following email messages as examples of the communication that you send to your learners.

If Learners Enter a Coupon Code on the Checkout Page

Dear learner,

This message includes a discount <or an enrollment> code for edX101: Overview of Creating an edX Course. For more information about the course, see <https://www.edx.org/course/overview-creating-edx-course-edx-edx101>.

To redeem this code, sign up for a verified <or professional> certificate, and then enter the following coupon code in the ****Coupon Code**** field on the ****Checkout**** page:


```
ZDPC3AQV3732RQT5
```

```
We look forward to learning with you!
```

```
The edX101 course team
```

If Learners Visit an Offer Landing Page

```
Dear learner,
```

```
This message includes a discount <or an enrollment> code for edX101: Overview of Creating an edX Course. To redeem this code and enroll in the course, visit the following URL:
```

```
http://localhost:8002/coupons/offer/?code=ZDPC3AQV3732RQT5
```

```
We look forward to learning with you!
```

```
The edX101 course team
```

View the Invoice for a Coupon

When you create a coupon, the E-Commerce service generates and fulfills an order. The Invoice Payment Processor module in the service then records the fulfilled order. Because the Invoice Payment Processor module assumes that you have sent or will send an invoice to the client who purchased the coupon, the module also records the order in the Invoice table in the Django administration panel so that you can view and reconcile the order.

To view your coupon invoices in the Invoice table, go to `http://localhost:8002/admin/invoice/invoice/`. The table lists all of the invoices for your coupons, along with information such as the client name and the invoice status.

For more information about the way that the E-Commerce service manages orders, see [Manage Orders](#).

6.4.4 Enable and Create Enrollment Codes

Enrollment codes allow users to purchase bulk enrollments for a course.

Note: Enrollment codes used for bulk enrollments, as described in this topic, are not related to the “enrollment code” coupon code type that is referred to in the [Create and Manage Coupons](#) topic.

Enable Enrollment Codes

Before you create enrollment codes that can be used for bulk enrollments for a course, you must enable enrollment codes in the E-Commerce service and in individual courses.

1. To enable enrollment codes in the E-Commerce service, run the site configuration command together with the following option.

```
``--enable-enrollment-codes=True``
```

For more information, see *Add Another Site, Partner, and Site Configuration*.

2. To enable enrollment codes in individual courses, follow these steps.
 - (a) Follow step 1 through step 5 in *Create Course Seats* to access and select options on the **Add New Course** page. Do not select **Create Course** after you complete step 5.
 - (b) Select **Include Enrollment Code**.
 - (c) Select **Create Course**.

After you select **Create Course**, enrollment codes are enabled for that course.

Create Enrollment Codes

1. Go to the enrollment page for the course.
2. On the enrollment page, select **Buy enrollment**. Do not select **Enroll in the course**.

The basket page opens.
3. For **Number of Enrollment Codes**, enter the number of enrollment codes that you want. Each enrollment code is for one course seat.
4. Select **Purchase**.

After you select **Purchase**, you receive an e-mail message that contains a link to a .csv file. The .csv file lists the enrollment codes.

6.4.5 About Programs

Programs are collections of related courses that you make available on your marketing site. Each program is of a particular type.

The cost for a program is the sum of the cost for each of its courses. You can change the cost for a program by creating a program offer, which is a discount on the program price of either a percentage or fixed amount. For more details, see *Create Program Offers*.

Create Program Types

You add program types in the Discovery Service Django administration site for your Open edX instance.

To add a program, follow these steps.

1. Sign into the Discovery Service Django Administration site for your Open edX instance. For example, `https://<discovery-baseURL>/admin/`, or `localhost:18381` if you are testing locally.
2. In the **Course Metadata** section, select **Program types**.
3. Select **Add Program Type**.
4. On the **Add program type** page, specify a name for the new program type, and select the seat types that are applicable to programs of this type.
5. Optionally, add a program logo image and a slug for this program type for use on the marketing site.
6. When you have finished entering information for the program type, select one of the **Save** options: **Save**, **Save and add another**, or **Save and continue editing**.

You can now specify this program type when you create new programs.

Create Programs

You add programs and specify the courses that are in each program in the Discovery Service Django administration site for your Open edX instance.

To add a program, follow these steps.

1. Sign into the Discovery Service Django Administration site for your Open edX instance. For example, `https://<discovery-baseURL>/admin/`, or `localhost:18381` if you are testing locally.
2. In the **Course Metadata** section, select **Programs**.
3. Select **Add Program**.

On the **Add Program** page, a UUID is assigned to the new program.

4. Enter information for the new program. Required fields, for example **Title**, **Status** and **Type**, have boldface names.
 - In the **Courses** field, specify the courses that are part of the program. Names of current courses are automatically matched as you continue to type. To add a course that does not currently exist, click the plus sign (+) next to the field to create a new course.
 - To allow learners to purchase upgrades to the verified track for all the courses in the program with one click, select **One click purchase enabled**.
5. When you have finished entering information for the program, select one of the **Save** options: **Save**, **Save and add another**, or **Save and continue editing**.

6.4.6 Create Program Offers

Program offers are discounts, either percentage or fixed amount discounts, that apply to a specific program. When a program offer is active for a program, the program's price appears discounted both on the program's purchase button and on the e-commerce checkout page.

You create program offers on the **Create Program Offer** page in the E-Commerce Program Offers Administration tool, which is located at `http://localhost:8002/programs/offers`.

Note: Each program can be associated with only one program offer. To offer a new discount, edit the existing program offer for your program.

To create a program offer, follow these steps.

1. Start the E-Commerce Service on your site. For details, see *Start the E-Commerce Service*.
2. Obtain the Program UUID for the program for which you are creating an offer. Find your program's UUID in the Discovery Service Django administration site, under **Course Metadata > Programs**.
3. In a browser on your E-Commerce server, go to `http://localhost:8002/programs/offers/` to access the E-Commerce Program Offers Administration tool.
4. On the **Program Offers** page, select **Create Program Offer**.
5. On the **Create Program Offer** page, enter the following information for your program offer.
 - Program UUID
 - Start Date
 - End Date

- Discount Type - either percentage or absolute.
- Discount Value - the value of the discount based on the discount type.

Note: To ensure that your program discount is reflected even when only some, not all, of a program's courses are in a learner's basket for checkout, you must select the **Enable Partial Program Offer** setting in the E-Commerce Service Django Administration site, under **Core > Site configurations**.

1. Select **Create Program Offer**.

6.5 Enable the E-Commerce Service Receipt Page

The E-Commerce service includes a receipt page that you can display to users after their orders are complete.

- *Enable the Receipt Page*

6.5.1 Enable the Receipt Page

To enable the default receipt page for the E-Commerce service, follow these steps.

1. Sign in to the LMS Django administration console for your base URL. For example, `http://{your_URL}/admin`.
2. On the **Site Administration** page, locate **Site Configuration**.
3. In the **Site Configuration** section, next to **Site configurations**, select **Change**.
4. On the **Change site configuration** page, locate the **Values** field, and then add the following JSON format value.

```
{
  "ECOMMERCE_RECEIPT_PAGE_URL": "/checkout/receipt/?order_number="
}
```

5. Select **Save**.

6.6 Manage Orders

EdX has created a framework that manages order placement and fulfillment for digital products. Most of the products that edX supports involve modifications to enrollments.

The edX framework allows modules that fulfill enrollment-related products to interact with the [edX Enrollment API](#), which is a part of the LMS. This process can be either synchronous or asynchronous.

6.6.1 Place an Order

To place an order, you need the following information.

- Order number
- User

- Basket
- Shipping method
- Shipping charge
- Billing address
- Order total

To place an order, use the `handle_order_placement()` method that `EdxOrderPlacementMixin` provides. If you modify this code, make sure that the code collects payment before it creates order objects.

6.6.2 Fulfill Orders

To fulfill orders, emit a `post_checkout` signal. An internal fulfillment API then delegates fulfillment of individual order items to the appropriate fulfillment modules.

6.6.3 About Fulfillment Modules

The E-Commerce base fulfillment module has the following interface.

```
fulfill_product(product)
revoke_line(line)
```

Every `ProductType` has a corresponding module that extends this interface and fulfills order items of that particular `ProductType`. To fulfill an order, the system gives each fulfillment module configured in `settings (_oscar.py)` an opportunity to fulfill order lines.

- The `fulfill_product` method fulfills the order. For example, `fulfill_product` might enroll a learner in a course or upgrade the learner to a verified certificate).
- The `revoke_line` method revokes a specific order line. For example, `revoke_line` might unenroll learners from courses or downgrade a learner from a verified seat.

6.6.4 Recover from a Fulfillment Error

If a fulfillment operation fails, the E-Commerce service assigns the order a status indicating the reason for the failure. If you have enabled asynchronous order fulfillment, the service tries again to fulfill the order. You can also manually retry fulfillment of a failed order from the [Oscar](#) order dashboard.

6.7 Test Your E-Commerce Application

To test new applications that you develop for the E-Commerce service, you create and run tests for the Open edX platform first, and then you run a set of tests that are specific to E-Commerce.

For more information about tests for the Open edX platform, see [Testing Open edX Features](#) in the *Open edX Developer's Guide*.

6.7.1 Tests for E-Commerce

When you develop E-Commerce applications, you must run a pre-packaged set of unit tests, Python tests, and E-Commerce acceptance tests.

Run E-Commerce Unit Tests

The E-Commerce unit tests include migrations, the unit test suite, and quality checks. You can run the full unit test, or save time for a local test by disabling the migrations. (You can also run the quality checks independent of the unit test suite.)

- To run the full unit test, including migrations and quality checks, use the following command.

```
$ make validate
```

Note: If numerous unit tests fail with an `OfflineGenerationError` message, run the following command, then try to run unit tests again.

```
$ DJANGO_SETTINGS_MODULE=ecommerce.settings.test make static
```

- To run unit tests with quality checks but without migrations, run the following command.

```
$ DISABLE_MIGRATIONS=1 make validate
```

Note: We recommend that you only run tests without migrations when you run the tests locally.

- To validate code quality independently, run the following command.

```
$ make quality
```

Python Unit Tests

When you create E-Commerce tests, use the `TestCase` class in `ecommerce/tests/testcases.py` to ensure every test has `Site` and `Partner` objects configured. This will help you test any code that relies on these models, which are used for multi-tenancy.

- To run all Python unit tests and quality checks, run the following command.

```
$ make validate_python
```

- To run all Python unit tests and quality checks *in parallel*, run the following command.

```
$ make fast_validate_python
```

- To run the Python unit tests in a specific file, such as `ecommerce/courses/tests/test_utils.py`, run the following command and substitute the desired file path.

```
$ DISABLE_ACCEPTANCE_TESTS=True ./manage.py test
ecommerce.courses.tests.test_utils --settings=ecommerce.settings.test
--with-ignore-docstrings --logging-level=DEBUG
```

Setting the `DISABLE_MIGRATIONS` variable significantly decreases the time needed to run tests by creating the test database directly from Django model definitions as opposed to applying the defined migrations.

```
$ DISABLE_MIGRATIONS=1 DISABLE_ACCEPTANCE_TESTS=True ./manage.py test
ecommerce.courses.tests.test_utils --settings=ecommerce.settings.test
--with-ignore-docstrings --logging-level=DEBUG
```

JavaScript Unit Tests

The E-Commerce project uses [Jasmine](#) for JavaScript unit testing. To create these tests, place your tests in the `ecommerce/static/js/test/specs` directory, and add a `_spec` suffix. For example, your test name may be `ecommerce/static/js/test/specs/course_list_view_spec.js`.

All JavaScript code must adhere to standards outlined in the [edX JavaScript Style Guide](#). These standards are enforced using [ESLint](#).

- To run all JavaScript unit tests and linting checks, run the following command.

```
$ make validate_js
```

Run E-Commerce Acceptance Tests

To run specific acceptance tests for the E-Commerce service, you must complete the following procedures.

- *Configure the LMS*
- *Configure E-Commerce*
- *Configure Acceptance Tests*
- *Run Acceptance Tests*

Configure the LMS

To configure the LMS, follow these steps.

1. Verify that the following settings in `/edx/app/edxapp/lms.env.json` are correct.

```
"ECOMMERCE_API_URL": "http://localhost:8002/api/v2/"
"ECOMMERCE_PUBLIC_URL_ROOT": "http://localhost:8002"
"JWT_ISSUER": "http://127.0.0.1:8000/oauth2" // Must match the E-Commerce JWT_
↔ISSUER setting
"OAUTH_ENFORCE_SECURE": false
"OAUTH_OIDC_ISSUER": "http://127.0.0.1:8000/oauth2"
```

2. Verify that the following settings in `/edx/app/edxapp/lms.auth.json` are correct.

```
"ECOMMERCE_API_SIGNING_KEY": "insecure-secret-key" // Must match the E-Commerce_
↔JWT_SECRET_KEY setting
"EDX_API_KEY": "PUT_YOUR_API_KEY_HERE" // Must match the E-Commerce EDX_API_KEY_
↔setting
```

3. Verify that an LMS account with staff and superuser permissions exists.

By default, most devstack and fullstack LMS instances include a user account that has staff permissions. This account has the username `staff`, the email address `staff@example.com`, and the password `edx`. Run the following commands to grant the account superuser permissions.

```
cd ~/edx-platform
./manage.py lms set_superuser staff --settings=devstack
```

4. Navigate to the OAuth2 Clients section of the Django administration console (e.g. <http://localhost:8000/admin/oauth2/client/>). Sign in using the superuser account you created earlier.

Verify that an OAuth2 client with the following attributes exists. If one does not already exist, *create a new OAuth 2 client*.

The client ID and secret must match the values of the E-Commerce `SOCIAL_AUTH_EDX_OIDC_KEY` and `SOCIAL_AUTH_EDX_OIDC_SECRET` settings, respectively.

```
Name: ecommerce
URL: http://localhost:8002/
Redirect URI: http://localhost:8002/complete/edx-oidc/
Client ID: 'ecommerce-key'
Client Secret: 'ecommerce-secret'
Client Type: Confidential (Web applications)
Logout url: http://localhost:8002/logout/
```

5. Navigate to the `Edx_Oauth2_Provider Trusted clients` section of the Django administration console (http://localhost:8000/admin/edx_oauth2_provider/trustedclient/).
Verify that the OAuth2 client referred to above is designated as a trusted client (look for the Redirect URI). If this isn't already the case, add the client created above as a new trusted client.
6. In the Django administration panel, create a new access token for the superuser account. Set the client to the OAuth2 client referred to above. Make note of this token; it is required to run the acceptance tests.
7. Make sure that the LMS instance that you will use for testing has at least two courses that learners can enroll in. By default, most LMS instances include the edX demonstration course. Use Studio to create a second course.

Configure E-Commerce

You use the E-Commerce Course Administration Tool (“ECAT”) to finish configuring the two courses in your LMS instance.

1. Become the `ecommerce` user.

```
sudo su ecommerce
```

2. Verify that the following keys in `/edx/etc/ecommerce.yml` match your settings in `/edx/app/edxapp/lms.env.json` and `/edx/app/edxapp/lms.auth.json`.
 - One of the `JWT_AUTH:JWT_ISSUERS` entries should match `/edx/app/edxapp/lms.env.json:JWT_ISSUER`. The default value is `http://127.0.0.1:8000/oauth2`.
 - `JWT_AUTH:JWT_SECRET_KEY` should match `/edx/app/edxapp/lms.auth.json:ECOMMERCE_API_SIGNING_KEY`. The default value is `lms-secret`.
 - `EDX_API_KEY` should match `/edx/app/edxapp/lms.auth.json:EDX_API_KEY`. The default value is `PUT_YOUR_API_KEY_HERE`.
3. Set up the E-Commerce environment.

```
cd /edx/app/ecommerce/ecommerce
source ../ecommerce_env
source ../venvs/ecommerce/bin/activate
make requirements
make static
make migrate
```


4. Create a new site linking to the LMS.

```
./manage.py create_or_update_site \  
  --site-id=1 \  
  --site-domain=localhost:8002 \  
  --partner-code=edX \  
  --partner-name='Open edX' \  
  --lms-url-root=http://localhost:8000 \  
  --theme-scss-path=sass/themes/edx.scss \  
  --payment-processors=cybersource,paypal \  
  --client-id=ecommerce-key \  
  --client-secret=ecommerce-secret \  
  --from-email staff@example.com
```

5. Start the E-Commerce Django development server.

```
./manage.py runserver 0.0.0.0:8002
```

6. Get the course key from the LMS by navigating to a course and examining its URL. The course key should look something like `course-v1:edX+DemoX+Demo_Course`.
7. Navigate to the E-Commerce Courses page (<http://localhost:8002/courses/>) to add the two test courses that are on your LMS instance to E-Commerce. Configure one course as a “Free (Audit)” course, and the second as a “Verified” course.
8. To configure the “Verified” course, click **Add New Course**. Leave the default values in all fields, with the exception of the following fields.

```
Course ID: The course key from the LMS  
Course Name: Use any name  
Course Type: Verified  
Include Honor Seat: No
```

9. Navigate to the learner dashboard (e.g. <http://localhost:8000/dashboard>) and confirm that the course you added in E-Commerce now has a green “Upgrade to Verified” badge, which you can select.
10. In the ECAT, add the second course on your LMS instance to E-Commerce. Specify its course type as Free (Audit).
11. To test integration with external payment processors, update the contents of the `PAYMENT_PROCESSOR_CONFIG` dictionary found in the settings with valid credentials. To override the default values for development, create a private settings module, `private.py`, and set `PAYMENT_PROCESSOR_CONFIG` inside the module.

Note: If you created a `private.py` file to create settings overrides when you *set up your virtual environment*, you can use that same `private.py` file.

Configure Acceptance Tests

You configure acceptance tests by using the settings in the `e2e/config.py` file. You can use the default values for most settings in this file. However, for the following settings, you must specify values using environment variables.

Variable	Description
ACCESS_TOKEN	The OAuth2 access token used to authenticate requests.
ECOMMERCE_URL_ROOT	The URL root for the E-Commerce service.
LMS_URL_ROOT	The URL root for the LMS.
LMS_USERNAME	A username for any current LMS user, to use during testing.
LMS_EMAIL	The email address used to sign in to the LMS.
LMS_PASSWORD	The password used to sign in to the LMS.

If you test PayPal integration, you must also specify values for the following settings by using environment variables.

Variable	Description
PAYPAL_EMAIL	The email address used to sign in to PayPal during payment.
PAYPAL_PASSWORD	The password used to sign in to PayPal during payment.

Run Acceptance Tests

Run all acceptance tests by executing `make e2e`.

To run a specific test, execute the following command.

```
$ nosetests -v <path/to/the/test/module>
```

The acceptance tests rely on the *environment variables that you have configured*. For example, when you run the acceptance tests against local instances of E-Commerce and the LMS, you might run the following command, replacing values between angle brackets (<>) with your own values.

```
$ ECOMMERCE_URL_ROOT="http://localhost:8002" LMS_URL_ROOT="http://127.0.0.1:8000" LMS_
↪USERNAME="<username>" LMS_EMAIL="<email address>" LMS_PASSWORD="<password>" ACCESS_
↪TOKEN="<access token>" LMS_HTTPS="False" LMS_AUTO_AUTH="True" PAYPAL_EMAIL="<email_
↪address>" PAYPAL_PASSWORD="<password>" ENABLE_CYBERSOURCE_TESTS="False" VERIFIED_
↪COURSE_ID="<course ID>" make e2e
```

When you run the acceptance tests against a production-like staging environment, you might run the following command.

```
$ ECOMMERCE_URL_ROOT="https://ecommerce.stage.edx.org" LMS_URL_ROOT="https://courses.
↪stage.edx.org" LMS_USERNAME="<username>" LMS_EMAIL="<email address>" LMS_PASSWORD="
↪<password>" ACCESS_TOKEN="<access token>" LMS_HTTPS="True" LMS_AUTO_AUTH="False"
↪PAYPAL_EMAIL="<email address>" PAYPAL_PASSWORD="<password>" BASIC_AUTH_USERNAME="
↪<username>" BASIC_AUTH_PASSWORD="<password>" HONOR_COURSE_ID="<course ID>" VERIFIED_
↪COURSE_ID="<course ID>" make e2e
```

6.8 Additional E-Commerce Features

After you install the basic features of the E-Commerce service on your instance of the Open edX platform, you can enable or install additional features. You can set up E-Commerce to send email, switch payment processors, or track event data if one of your usual processors is unavailable.

6.8.1 Sending Notifications

The edX E-Commerce service uses the [Communications API](#) that is part of [Oscar](#) to create and send notifications in the form of email messages. To send notifications, you must set up notifications, create one or more email messages, and then send the email messages.

Set Up Notifications

1. Enable the E-Commerce service to send notifications. To do this, change the value of the `ENABLE_NOTIFICATIONS` feature flag to `True`.
2. Define communication type codes to refer to particular types of notification. For example, you might define a communication type code named `COURSE_SEAT_PURCHASED` to correspond to the purchase of a course seat.

Create an Email Message

The E-Commerce service can send both HTML and plain text email messages. To create an email message, create the following three files in the `ecommerce/ecommerce/templates/customer/emails/` folder.

- An HTML template that extends `email_base.html` and includes the body of the email.
- A plain text file that contains the email's subject line.
- A plain text file that contains the body of the email.

Use the following convention to name these files.

```
commtype_{communication type code}_body.html
```

For example, if the communication type code is `course_seat_purchased`, the three files would have the following names.

- `commtype_course_seat_purchased_body.html`
- `commtype_course_seat_purchased_body.txt`
- `commtype_course_seat_purchased_subject.txt`

Note: To add a custom email body, override `block body` in the `email_base.html` file. To add a custom footer, override `block footer` in the `email_base.html` file.

Send Email Messages

To send email messages, use the `send_notification(user, commtype_code, context)` method. This method is implemented in `ecommerce/ecommerce/notifications/notifications.py`.

6.8.2 Changing Payment Processors

Payment processors sometimes experience temporary outages. When these outages occur, you can use Waffle switches to disable the faulty payment processor or processors, then re-enable them after the outage is over.

The names of these switches use prefixes that are the value of the `PAYMENT_PROCESSOR_SWITCH_PREFIX` setting. By default, this value is `payment_processor_active_`. The following table lists valid switches and the payment processors they control.

Payment Processor	Switch Name	Default Value
PayPal	payment_processor_active_paypal	True
CyberSource	payment_processor_active_cybersource	True

In the unlikely event that all payment processors are disabled, the LMS will display an informative error message explaining why payment is not currently possible.

6.8.3 Apple Pay

Apple Pay support is available when you use the CyberSource processor. Apple Pay allows learners to check out quickly without having to manually fill out the payment form. If you are not familiar with Apple Pay, take a moment to read the following documents to understand the user flow and necessary configuration.

- [Apple Pay JS](#)
- [CyberSource Simple Order API](#)

Apple Pay is available only to learners using Safari on the following platforms:

- iOS 10+ on devices with a Secure Element
- macOS 10.12+. The user must have an iPhone, Apple Watch, or a MacBook Pro with Touch ID that can authorize the payment.

An exhaustive list of devices that support Apple Pay is available on [Wikipedia](#).

Note: The Apple Pay button is not displayed to users with incompatible hardware and software.

Settings

Apple Pay is configured via the `PAYMENT_PROCESSOR_CONFIG` dictionary in settings. The following keys are required.

Name	Purpose
apple_pay_merchant_identifier	Merchant identifier created at the Apple Developer portal
apple_pay_country_code	Two-letter ISO 3166 country code for your business/merchant account
apple_pay_merchant_id_domain_association	Domain verification text obtained from the Apple Developer portal
apple_pay_merchant_id_certificate_path	Filesystem path to the merchant identity certificate (used to authenticate with Apple to start sessions). That file should be kept in a secure location that is only accessible by administrators and the application's service user.

6.8.4 Tracking Data

The E-Commerce service uses [Segment](#) to collect business intelligence data.

To emit events to your Segment project, specify your Segment project's API key as the value of the `SEGMENT_KEY` setting.

6.8.5 Gating E-Commerce Service Features

You can release new E-Commerce service features and functionality behind a feature gate. This project uses the [Waffle](#) library for feature gating.

Types of Feature Gates

Waffle supports the following types of feature gates.

- **Flag:** This gate allows you to enable a feature for specific users, groups, users who meet certain criteria (such as authenticated users or staff), or a certain percentage of visitors.
- **Switch:** This gate is a Boolean that turns a feature on or off for all users.
- **Sample:** This gate allows you to define the probability with which a given feature will be on.

For more information about creating or updating features and feature gates, see the [Waffle documentation](#).

Feature Gates

Waffle offers the following feature gates.

Name	Type	Purpose
user_enrollments_on_dashboard	Switch	Display a user's current enrollments on the dashboard user detail page.
publish_course_modes_to_lms	Switch	Publish prices and SKUs to the LMS after every course modification.
async_order_fulfillment	Sample	Specify what percentage of orders are fulfilled asynchronously.
ENABLE_CREDIT_APP	Switch	Enable the credit checkout page, from which learners can purchase credit in a course.
ENABLE_NOTIFICATIONS	Switch	Enable email notifications for a variety of user actions, such as when an order is placed.
PAYPAL_RETRY_ATTEMPTS	Switch	Enable users to retry unsuccessful PayPal payments.

Enable a Feature Permanently

If you want to make a feature permanent, remove its feature gate from relevant code and tests, and delete the gate from the database.

6.8.6 Maintaining the E-Commerce Service

Most of the time, you do not have to perform maintenance on the E-Commerce service. However, E-Commerce creates **basket** objects to track products that users want to purchase before users place an order. As more baskets and orders are created, the baskets table can grow large. Depending on your database backend, a large table can become difficult to manage and migrate. After an order is placed, you can delete the corresponding basket from the baskets table.

To delete one or more baskets, follow these steps.

Note: Baskets that contain products but that are not used to create orders, such as when a user adds a product to a basket but does not complete the order process, are not deleted. These baskets provide records that users intended to purchase a product.

1. To display the number of baskets that you can delete, run the following command.

```
$ ./manage.py delete_ordered_baskets
```

2. To delete all the baskets that appear after you run the command in step 1, use the `--commit` option.

```
$ ./manage.py delete_ordered_baskets --commit
```

6.9 Internationalization

Follow the [internationalization coding guidelines](#) in the edX Developer's Guide when developing new features.

Languages are enabled in the settings file, for example in `ecommerce/settings/base.py`

```
LANGUAGES = (
    ('en', _('English')),
    ('es', _('Spanish')),
    ('es-419', _('Spanish (Latin American)')),
)
```

More details can be found in the [Django documentation for Languages](#).

6.9.1 E-Commerce Language Negotiation

Language negotiation for E-Commerce is handled by [Django's Locale Middleware](#). Django's Language Negotiation rules, in priority order, are as follows.

1. Language prefix is not enabled by default for the E-Commerce application.
2. `LANGUAGE_SESSION_KEY` is not used.
3. `LANGUAGE_COOKIE_NAME` is used to negotiate language. A user's language is set in their account settings in `edx-platform` which sets the language cookie. The language cookie name that is set for the E-Commerce language should be the same as the language cookie name set in the `edx-platform` repo. This can be configured in `ecommerce/settings/base.py`.

```
LANGUAGE_COOKIE_NAME = 'openedx-language-preference'
```

4. The `Accept-Language` HTTP header is used.
5. `LANGUAGE_CODE` is used as the last resort. This can be set in `ecommerce/settings/base.py`

6.9.2 PayPal Language Negotiation

To enable localization for PayPal, follow these steps.

1. Sign in to the LMS Django administration console for your base URL. For example, `http://{your_URL}/admin`.
2. On the **Site Administration** page, locate **Waffle**.
3. In the **Waffle** section, next to **Switches**, select **Add**.
4. On the **Add switch** page, locate the **Name** field, and then add `create_and_set_webprofile`.
5. On the **Add switch** page, select the **Active** checkbox.

6. Select **Save**.

A language code for E-Commerce will be taken from a cookie as described in *E-Commerce Language Negotiation*. When the language code is fetched from the cookie, only the base language is used. For example, `es-419` resolves to `es`. PayPal requires a country code. To get the country code, we use the language code to map it to a country. For example, the language code `es` will map to the country code `MX` when it is sent to PayPal. To add your language for PayPal, look up [PayPal's country to language mapping](#) and add it to `PAYPAL_LOCALES` in `ecommerce/extensions/payment/constants.py`.

```
PAYPAL_LOCALES = {
    'zh': 'CN',
    'fr': 'FR',
    'en': 'US',
    'es': 'MX',
}
```

If a language fetched from the cookie cannot be found in `PAYPAL_LOCALES`, the `LANGUAGE_CODE` is used. If the `LANGUAGE_CODE` does not exist in `PAYPAL_LOCALES`, PayPal will use its own language negotiation.

Setting Up the Open edX Mobile Applications

This section is intended for those who are building Open edX mobile applications and customizing an Open edX installation to support their use.

- *Accessing the Source Code*
- *Configuring Mobile Application Features*
- *Configuring Video Modules for Mobile*
- *Enabling Push Notifications*

7.1 Accessing the Source Code

There are currently two edX mobile applications, one for iOS and one for Android. You can find the source code and additional documentation for each application in the following repositories.

- iOS: <http://github.com/edx/edx-app-ios>
- Android: <http://github.com/edx/edx-app-android>

7.2 Configuring Mobile Application Features

For the mobile API and authentication to work correctly with the edX mobile applications, you enable them in the `edx/app/edxapp/lms.env.json` file. You then complete the setup for authentication by creating OAuth clients and adding their IDs to the custom configuration file of each mobile application.

7.2.1 Enable Mobile Application Features

Note: Configuration settings added to the `lms.env.json` file are reset to their default values when you use Ansible to update `edx-platform`.

To enable the mobile application features, follow these steps.

1. In the `edx/app/edxapp/lms.env.json` file, add the following lines to the features section.

```
"FEATURES" : {
  ...
  "ENABLE_MOBILE_REST_API": true,
  "ENABLE_OAUTH2_PROVIDER": true,
  "ENABLE_COMBINED_LOGIN_REGISTRATION": true
}
```

If you are running in a non-SSL environment, you can set `"OAUTH_ENFORCE_SECURE": false`. This configuration setting should never be set to `false` in anything other than a development environment.

1. Save the `edx/app/edxapp/lms.env.json` file.
2. Restart the server.

7.2.2 Create the OAuth Clients

You create an OAuth client ID and secret that are specific to your installation for use by the mobile applications. The procedure that follows assumes that you create a different client for each of the edX mobile applications.

Before you can create OAuth clients, a superuser must exist on the server. For more information, see [edX Managing the Full Stack](#).

To create your OAuth clients, follow these steps.

1. Log in to the Django administration console for your base URL. For example, `http://{your_URL}/admin`.
2. In the **Oauth2** section, select **Clients**.
3. Select **Add client**. A dialog box opens with the **Client id** and **Client secret** populated for the new client.
4. Enter a **Url** and **Redirect Url** for the first mobile application. For example, `https://{your_URL}/api/mobile/{version}/?app=ios`. While the console requires values for these fields, they are not used by the edX mobile applications. You can enter the same value in both fields.
5. For the **Client type**, select **Public**.
6. Optionally, enter a **User** and an identifying **Name**.
7. Select **Save and add another**.
8. Repeat steps 4-6 for the second mobile application, and then select **Save**.

7.2.3 Configure the Applications with OAuth Client IDs

The procedure that follows assumes that you have a different OAuth client ID for each edX mobile application.

To configure each edX mobile application with its OAuth client ID, you add a setting to its custom configuration `.yaml` file. For information about how to set up custom configuration directories and files, see the GitHub repositories for [iOS](#) and [Android](#).

1. Obtain the OAuth client ID for the first mobile application from your Django administration console. For more information, see *Create the OAuth Clients*.
2. In your custom GitHub configuration directory, edit the `.yaml` file for the first mobile application. For example, for the edX mobile application for iOS, edit your `ios.yaml` file.
3. Add the following line to the `.yaml` file.

```
OAuth_CLIENT_ID: '{client_id_for_ios_app}'
```

4. Save the file.
5. Repeat steps 1-4 for the second mobile application.

7.2.4 Configure OAuth Token Expiration

When OAuth tokens expire, learners who use the mobile apps to access your site must log in again.

The `lms/envs/common.py` file includes the default configuration settings for the number of days before OAuth tokens expire for confidential clients and public clients. Instead of modifying the defaults in `lms/envs/common.py`, add the configuration settings to the `lms.env.json` file and set values that will override the default settings.

To configure the number of days before OAuth tokens expire, follow these steps.

1. In the `edx/app/edxapp/lms.env.json` file, add the following lines to specify the number of days that OAuth tokens remain valid for confidential or public clients.

Note: Make sure you add these lines at the top level of the JSON dictionary, and not inside any other declarations.

```
"OAuth_EXPIRE_CONFIDENTIAL_CLIENT_DAYS" : 365  
"OAuth_EXPIRE_PUBLIC_CLIENT_DAYS" : 30
```

2. Save the `lms.env.json` file, then restart the `edxapp` app.

The values that you defined in `lms.env.json` override the default values defined in `lms/envs/common.py`.

7.3 Configuring Video Modules for Mobile

Course videos must be specifically prepared to ensure that they are in mobile accessible formats. Video modules in mobile-available courses should have low resolution encodings that are readily accessible by mobile devices.

After you obtain a low resolution encoding for a video file and post it online, your course team can add its location to the video module that includes the video in the course structure.

- To configure a video module in edX Studio, you edit the video component. On the **Advanced** tab, in the **Video File URLs** field, enter the URL to the mobile-targeted video as the first URL in the list. For more information, see *Working with Video Components* in *Building and Running an Open edX Course*.
- To configure a video module by editing the course XML, enter the URL to the mobile-targeted video as the first URL in the list of `html5_sources`. For more information, see *Video Components* in the *edX Open Learning XML Guide*.

7.4 Enabling Push Notifications

You enable push notifications on the server and then configure each of the edX mobile applications. Currently, you can use Parse for notification delivery.

The procedures that follow assume that you have obtained an application ID, REST API key, and client key from Parse.

7.4.1 Enable Push Notification on the Server

To enable the push notification feature, follow these steps.

1. In the `edx/app/edxapp/cms.auth.json` file, add the following lines.

```
PARSE_KEYS = {  
    "APPLICATION_ID": "{app_id}",  
    "REST_API_KEY": "{API_key}"  
}
```

2. Save the `edx/app/edxapp/cms.auth.json` file.
3. Restart the server.
4. Log in to the Django administration console for your Studio URL. For example, `http://studio.{your_URL}/admin`.
5. In the **Contentstore** section, select **Push notification configs**.
6. Select **Add push notification config**.
7. Verify that **Enabled** is selected, and then select **Save**.

7.4.2 Configure Push Notification on the Clients

The procedure that follows assumes that you use a single set of Parse credentials for both of the edX mobile applications.

You add push notification settings to the applicable custom configuration `.yaml` file. For information about how to set up custom configuration directories and files, see the GitHub repositories for [iOS](#) and [Android](#).

1. In your custom GitHub configuration directory, edit the `.yaml` file that stores configuration settings that are common to both of the edX mobile applications. For example, edit your `shared.yaml` file.
2. Add the following lines to the `.yaml` file.

```
PARSE:  
  NOTIFICATIONS_ENABLED: true  
  APPLICATION_ID: {your application id}  
  CLIENT_KEY: {your client key}  
  
PUSH_NOTIFICATIONS: true
```

3. Save the file.

Index of Open EdX Feature Flags

The following list includes feature flags that are available in the Open edX platform. For more information about feature flags, see [edX Feature Flags](#) in the Open edX wiki.

Name	Description	Default Value
ACCESS_REQUIRE_STAFF_FOR_COURSE	Supported	FALSE
ADVANCED_SECURITY	Supported	TRUE
ALLOW_COURSE_STAFF_GRADE_DOWNLOADS	Supported	FALSE
AUDIT_CERT_CUTOFF_DATE	Supported	None
AUTH_USE_CAS	Supported	TRUE
AUTH_USE_CERTIFICATES	Supported	FALSE
AUTH_USE_OPENID	Supported	FALSE
AUTH_USE_OPENID_PROVIDER	Supported	FALSE
AUTH_USE_SHIB	Supported	FALSE
AUTOMATIC_AUTH_FOR_TESTING	Development	FALSE
AUTOMATIC_VERIFY_STUDENT_IDENTITY_FOR_TESTING	Development	FALSE
CERTIFICATES_HTML_VIEW	Supported	FALSE
CERTIFICATES_INSTRUCTOR_GENERATION	Supported	FALSE
CUSTOM_COURSES_EDX	Supported	FALSE
DEBUG_LEVEL	Deprecated	0
DISABLE_LOGIN_BUTTON	Supported	FALSE
DISPLAY_ANALYTICS_DEMOGRAPHICS	Deprecated	TRUE
DISPLAY_ANALYTICS_ENROLLMENTS	Deprecated	TRUE
DISPLAY_DEBUG_INFO_TO_STAFF	Supported	TRUE
EMBARGO	Supported	FALSE
ENABLE_CORS_HEADERS	Supported	FALSE
ENABLE_COSMETIC_DISPLAY_PRICE	Supported	FALSE
ENABLE_COURSE_DISCOVERY	Supported	FALSE
ENABLE_COURSEWARE_SEARCH	Supported	FALSE
ENABLE_CREATOR_GROUP	Supported	FALSE

Continued on next page

Table 8.1 – continued from previous page

Name	Description	Default Value
ENABLE_CREDIT_API	Supported	FALSE
ENABLE_DASHBOARD_SEARCH	Supported	FALSE
ENABLE_DISABLING_XBLOCK_TYPES	Supported	TRUE
ENABLE_DISCUSSION_EMAIL_DIGEST	Supported	FALSE
ENABLE_DISCUSSION_HOME_PANEL	Supported	FALSE
ENABLE_DISCUSSION_SERVICE	Supported	TRUE
ENABLE_DJANGO_ADMIN_SITE	Supported	TRUE
ENABLE_EDXNOTES	Supported	FALSE
ENABLE_FEEDBACK_SUBMISSION	Supported	FALSE
ENABLE_HTML_XBLOCK_STUDENT_VIEW_DATA	Supported	FALSE
ENABLE_INSTRUCTOR_ANALYTICS	Deprecated	FALSE
ENABLE_INSTRUCTOR_BACKGROUND_TASKS	Supported	TRUE
ENABLE_INSTRUCTOR_EMAIL	Supported	TRUE
ENABLE_INSTRUCTOR_LEGACY_DASHBOARD	Deprecated	FALSE
ENABLE_LMS_MIGRATION	Deprecated	FALSE
ENABLE_MANUAL_GIT_RELOAD	Supported	FALSE
ENABLE_MAX_FAILED_LOGIN_ATTEMPTS	Supported	TRUE
ENABLE_MAX_SCORE_CACHE	Supported	TRUE
ENABLE_MKTG_SITE	Supported	FALSE
ENABLE_MOBILE_REST_API	Supported	FALSE
ENABLE_OAUTH2_PROVIDER	Supported	FALSE
ENABLE_ONLOAD_BEACON	Supported	FALSE
ENABLE_OPENBADGES	Supported	FALSE
ENABLE_PAID_COURSE_REGISTRATION	Supported	FALSE
ENABLE_PREREQUISITE_COURSES	Supported	FALSE
ENABLE_PSYCHOMETRICS	Deprecated	FALSE
ENABLE_RENDER_XBLOCK_API	Removed	FALSE
ENABLE_S3_GRADE_DOWNLOADS	Supported	FALSE
ENABLE_SHOPPING_CART	Deprecated	FALSE
ENABLE_SPECIAL_EXAMS	Supported	False
ENABLE_STUDENT_HISTORY_VIEW	Supported	TRUE
ENABLE_STUDENT_NOTES	Supported	TRUE
ENABLE_TEAMS	Supported	FALSE
ENABLE_TEXTBOOK	Supported	TRUE
ENABLE_THIRD_PARTY_AUTH	Supported	FALSE
ENABLE_VIDEO_BEACON	Supported	FALSE
ENABLE_VIDEO BUMPER	Supported	FALSE
ENABLE_AUTOADVANCE_VIDEOS	Supported	FALSE
ENABLE_XBLOCK_VIEW_ENDPOINT	Supported	FALSE
ENFORCE_PASSWORD_POLICY	Supported	TRUE
ENTRANCE_EXAMS	Supported	FALSE
FALLBACK_TO_ENGLISH_TRANSCRIPTS	Supported	TRUE
FORCE_UNIVERSITY_DOMAIN	Deprecated	FALSE
INDIVIDUAL_DUE_DATES	Supported	FALSE
LICENSING	Supported	FALSE
LOG_POSTPAY_CALLBACKS	Supported	TRUE
MAX_ENROLLMENT_INSTR_BUTTONS	Supported	200
MILESTONES_APP	Supported	FALSE

Continued on next page

Table 8.1 – continued from previous page

Name	Description	Default Value
MODE_CREATION_FOR_TESTING	Development	FALSE
PREVENT_CONCURRENT_LOGINS	Supported	TRUE
REQUIRE_COURSE_EMAIL_AUTH	Supported	TRUE
RUN_AS_ANALYTICS_SERVER_ENABLED	Deprecated	FALSE
SEGMENT_IO_LMS	Deprecated	FALSE
SHIB_DISABLE_TOS	Supported	FALSE
SHOW BUMPER_PERIODICITY	Supported	7 * 24 * 3600
STORE_BILLING_INFO	Supported	FALSE
SUBDOMAIN_BRANDING	Deprecated	FALSE
SUBDOMAIN_COURSE_LISTINGS	Deprecated	FALSE
USE_DJANGO_PIPELINE	Supported	TRUE
USE_MICROSITES	Supported	FALSE

A - C - D - E - F - G - H - I - K - L - M - N - O - P - R - S - T - V - W - XYZ

Note: Most of the links to documentation provided in this glossary are to the [Building and Running an edX Course](#) guide, for edX partners. Many of the same topics are available in the Open edX version of this guide, [Building and Running an Open edX Course: Hawthorn Release](#).

9.1 A

AAC

Advanced audio coding (AAC) is an audio coding standard for digital audio compression. AAC is the standard format for YouTube.

A/B test

See [Content Experiment](#).

About page

The course page that provides potential learners with a course summary, prerequisites, a course video and image, and important dates.

accessible label

In a problem component, you use special formatting to identify the specific question that learners will answer by selecting options or entering text or numeric responses.

This text is referred to as the accessible label because screen readers read all of the text that you supply for the problem and then repeat the text that is identified with this formatting immediately before reading the answer choices for the problem. This text is also used by reports and Insights to identify each problem.

All problems require accessible labels.

For more information, see [The Simple Editor](#).

advanced editor

An OLX (open learning XML) editor in a problem component that allows you to create and edit any type of problem. For more information, see [The Advanced Editor](#).

Amazon Web Services (AWS)

A third-party file hosting site where course teams can store course assets, such as problem files and videos. If videos are posted on both YouTube and AWS, the AWS version of the video serves as a backup in case the YouTube video does not play.

assignment type

The category of graded student work, such as homework, exams, and exercises. For more information, see [Establishing a Grading Policy For Your Course](#).

9.2 C

CAPA problem

A CAPA (computer assisted personalized approach) problem refers to any of the problem types that are implemented in the edX platform by the `capa_module` XBlock. Examples range from text input, drag and drop, and math expression input problem types to circuit schematic builder, custom JavaScript, and chemical equation problem types.

Other assessment methods are also available, and implemented using other XBlocks. An open response assessment is an example of a non-CAPA problem type.

certificate

A document issued to an enrolled learner who successfully completes a course with the required passing grade. Not all edX courses offer certificates, and not all learners enroll as certificate candidates.

For information about setting up certificates for your course, see [Setting Up Certificates in Studio](#).

chapter

See [Section](#).

checkbox problem

A problem that prompts learners to select one or more options from a list of possible answers. For more information, see [Checkbox Problem](#).

chemical equation response problem

A problem that allows learners to enter chemical equations as answers. For more information, see [Chemical Equation Problem](#).

circuit schematic builder problem

A problem that allows learners to construct a schematic answer (such as an electronics circuit) on an interactive grid. For more information, see [Circuit Schematic Builder Problem](#).

closed captions

The spoken part of the transcript for a video file, which is overlaid on the video as it plays. To show or hide closed captions, you select the **CC** icon. You can move closed captions to different areas on the video screen by dragging and dropping them.

For more information, see [Watching Videos on the edX Video Player](#).

codec

A portmanteau of “code” and “decode”. A computer program that can encode or decode a data stream.

cohort

A group of learners who participate in a class together. Learners who are in the same cohort can communicate and share experiences in private discussions.

Cohorts are an optional feature of courses on the edX platform. For information about how you enable the cohort feature, set up cohorts, and assign learners to them, see [Using Cohorts in Your Courses](#).

component

The part of a unit that contains your actual course content. A unit can contain one or more components. For more information, see [Developing Course Components](#).

content experiment

You can define alternative course content to be delivered to different, randomly assigned groups of learners. Also known as A/B or split testing, you use content experiments to compare the performance of learners who have been exposed to different versions of the content. For more information, see [Overview of Content Experiments](#).

content library

See [Library](#).

content-specific discussion topic

A category within the course discussion that appears at a defined point in the course to encourage questions and conversations. To add a content-specific discussion topic to your course, you add a discussion component to a unit. Learners cannot contribute to a content-specific discussion topic until the release date of the section that contains it. Content-specific discussion topics can be divided by cohort, so that learners only see and respond to posts and responses by other members of the cohort that they are in.

For more information, see [Working with Discussion Components](#). For information about making content-specific discussion topics divided by cohort, see [Setting up Discussions in Courses with Cohorts](#).

course catalog

The page that lists all courses offered in the edX learning management system.

course handouts

Course handouts are files you make available to learners on the **Home** page. For more information, see [Adding Course Updates and Handouts](#).

course mode

See [enrollment track](#).

course navigation pane

The navigation frame that appears at one side of the **Course** page in the LMS. The course navigation pane shows the sections in the course. When you select a section, the section expands to show subsections. When you select a subsection, the first unit in that subsection appears on the course page.

See also [Unit Navigation Bar](#).

Course page

The page that opens first when learners access your course. On the **Course** page, learners can view the course outline and directly access the course, either by clicking a specific section or subsection on the outline, or by clicking the **Start Course** button (**Resume Course** if the learner has previously accessed the course).

The latest course update, such as a course welcome message, appears above the course outline. Links to various **Course Tools** including **Bookmarks**, **Reviews** and **Updates** appear at the side of this page. This page is a combination of the former **Home** and **Courseware** pages.

course run

A version of the course that runs at a particular time. Information about a course run includes start and end dates, as well as staff and the languages the course is available in. You can create a course run when you create a course.

course track

See *enrollment track*.

courseware

In OLX (open learning XML) and in data packages, “courseware” refers to the main content of your course, consisting mainly of lessons and assessments. Courseware is organized into sections, subsections, units, and components. Courseware does not include handouts, the syllabus, or other course materials.

Note that the **Course** page was formerly called the **Courseware** page.

course-wide discussion topic

Optional discussion categories that you create to guide how learners find and share information in the course discussion. Course-wide discussion topics are accessed from the **Discussion** page in your course. Examples of course-wide discussion topics include Announcements and Frequently Asked Questions. Learners can contribute to these topics as soon as your course starts. For more information, see [Creating Course Discussions](#) and [Create Course-Wide Discussion Topics](#).

If you use cohorts in your course, you can divide course-wide discussion topics by cohort, so that although all learners see the same topics, they only see and respond to posts and responses by other members of the cohort that they are in. For information about configuring discussion topics in courses that use cohorts, see [Setting up Discussions in Courses with Cohorts](#).

custom response problem

A custom response problem evaluates text responses from learners using an embedded Python script. These problems are also called “write-your-own-grader” problems. For more information, see [Write-Your-Own-Grader Problem](#).

9.3 D

data czar

A data czar is the single representative at a partner institution who is responsible for receiving course data from edX, and transferring it securely to researchers and other interested parties after it is received.

For more information, see the [Using the edX Data Package](#).

discussion

The set of topics defined to promote course-wide or unit-specific dialog. Learners use the discussion topics to communicate with each other and the course team in threaded exchanges. For more information, see [Creating Course Discussions](#).

discussion component

Discussion topics that course teams add directly to units. For example, a video component can be followed by a discussion component so that learners can discuss the video content without having to leave the page.

When you add a discussion component to a unit, you create a content-specific discussion topic. See also [Content Specific Discussion Topic](#).

For more information, see [Working with Discussion Components](#).

discussion thread list

The navigation frame that appears at one side of the **Discussion** page in the LMS. The discussion thread list shows the discussion categories and subcategories in the course. When you select a category, the list shows all of the posts in that category. When you select a subcategory, the list shows all of the posts in that subcategory. Select a post to read it and its responses and comments, if any.

dropdown problem

A problem that asks learners to choose from a collection of answer options, presented as a drop-down list. For more information, see [Dropdown Problem](#).

9.4 E

edX101

An online course about how to create online courses. The intended audience for **edX101** is faculty and university administrators.

edX Edge

edX Edge is a less restricted site than **edX.org**. While only **edX** employees and consortium members can create and post content on **edX.org**, any users with course creator permissions for **Edge** can create courses with **Studio** on **studio.edge.edx.org**, then view the courses on the learning management system at **edge.edx.org**.

edX Studio

The **edX** tool that you use to build your courses. For more information, see [Getting Started with Studio](#).

embargo

An embargo is an official ban on trade or commercial activity with a particular country. For example, due to U.S. federal regulations, **edX** cannot offer certain courses (for example, particular advanced STEM courses) on the **edx.org** website to learners in embargoed countries. Learners cannot access restricted courses from an embargoed country. In some cases, depending on the terms of the embargo, learners cannot access any **edX** courses at all.

enrollment mode

See [enrollment track](#).

enrollment track

Also called **certificate type**, **course mode**, **course seat**, **course track**, **course type**, **enrollment mode**, or **seat type**.

The enrollment track specifies the following items about a course.

- The type of certificate, if any, that learners receive if they pass the course.
- Whether learners must verify their identity to earn a certificate, using a webcam and a photo ID.
- Whether the course requires a fee.
- **audit**: This is the default enrollment track when learners enroll in a course. This track does not offer certificates, does not require identity verification, and does not require a course fee.

- **professional:** This enrollment track is only used for specific professional education courses. The professional enrollment track offers certificates, requires identity verification, and requires a fee. Fees for the professional enrollment track are generally higher than fees for the verified enrollment track. Courses that offer the professional track do not offer a free enrollment track.

Note: If your course is part of a MicroMasters or professional certificate program, your course uses the verified track. These courses do not use the professional enrollment track.

- **verified:** This enrollment track offers verified certificates to learners who pass the course, verify their identities, and pay a required course fee. A course that offers the verified enrollment track also automatically offers a free non-certificate enrollment track.
- **honor:** This enrollment track was offered in the past and offered an honor code certificate to learners who pass the course. This track does not require identity verification and does not require a fee. Note, however, that as of December 2015, edx.org no longer offers honor code certificates. For more information, see [News About edX Certificates](#).

exercises

Practice or practical problems that are interspersed in edX course content to keep learners engaged. Exercises are also an important measure of teaching effectiveness and learner comprehension. For more information, see [Adding Exercises and Tools](#).

export

A tool in edX Studio that you use to export your course or library for backup purposes, or so that you can edit the course or library directly in OLX format. See also [Import](#).

For more information, see [Export a Course](#) or [Export a Library](#).

9.5 F

forum

See [Discussion](#).

fps

Frames per second. In video, the number of consecutive images that appear every second.

9.6 G

grade range

Thresholds that specify how numerical scores are associated with grades, and the score that learners must obtain to pass a course.

For more information, see [Set the Grade Range](#).

grading rubric

See [Rubric](#).

9.7 H

H.264

A standard for high definition digital video.

Home page

See [Course Page](#).

HTML component

A type of component that you can use to add and format text for your course. An HTML component can contain text, lists, links, and images. For more information, see [Working with HTML Components](#).

9.8 I

Image mapped input problem

A problem that presents an image and accepts clicks on the image as an answer. For more information, see [Image Mapped Input Problem](#).

Import

A tool in Studio that you use to load a course or library in OLX format into your existing course or library. When you use the Import tool, Studio replaces all of your existing course or library content with the content from the imported course or library. See also [Export](#).

For more information, see [Import a Course](#) or [Import a Library](#).

instructor dashboard

A user who has the Admin or Staff role for a course can access the instructor dashboard in the LMS by selecting **Instructor**. Course team members use the tools, reports, and other features that are available on the pages of the instructor dashboard to manage a running course.

For more information, see [Managing a Running Course](#).

9.9 K

keyword

A variable in a bulk email message. When you send the message, a value that is specific to the each recipient is substituted for the keyword.

9.10 L

label

See [Accessible Label](#).

LaTeX

A document markup language and document preparation system for the TeX typesetting program. In edX Studio, you can [Import LaTeX Code into an HTML Component](#).

learning management system (LMS)

The platform that learners use to view courses, and that course team members use to manage learner enrollment, assign team member privileges, moderate discussions, and access data while the course is running.

learning sequence

See *Unit Navigation Bar*.

left pane

See *Course Navigation Pane*.

library

A pool of components for use in randomized assignments that can be shared across multiple courses from your organization. Course teams configure randomized content blocks in course outlines to reference a specific library of components, and randomly provide a specified number of problems from that content library to each learner.

For more information, see [Working with Content Libraries and Randomized Content Blocks](#).

live mode

A view that allows the course team to review all published units as learners see them, regardless of the release dates of the section and subsection that contain the units. For more information, see [Viewing Published and Released Content](#).

LON-CAPA

The Learning Online Network with Computer-Assisted Personalized Approach e-learning platform. The structure of CAPA problem types in the edX platform is based on the [LON-CAPA](#) assessment system, although they are not compatible.

See also *CAPA Problems*.

9.11 M

math expression input problem

A problem that requires learners to enter a mathematical expression as text, such as $e=m*c^2$.

For more information, see [Completing Mathematical and Scientific Assignments](#) in the *EdX Learner's Guide*.

MathJax

A LaTeX-like language that you use to write equations. Studio uses MathJax to render text input such as x^2 and $\sqrt{x^2-4}$ as “beautiful math.”

For more information, see [Using MathJax for Mathematics](#).

module

An item of course content, created in an XBlock, that appears on the **Course** page in the edX learning management system. Examples of modules include videos, HTML-formatted text, and problems.

Module is also used to refer to the structural components that organize course content. Sections, subsections, and units are modules; in fact, the course itself is a top-level module that contains all of the other course content as children.

multiple choice problem

A problem that asks learners to select one answer from a list of options. For more information, see [Multiple Choice Problem](#).

9.12 N

NTSC

National Television System Committee. The NTSC standard is a color encoding system for analog videos that is used mostly in North America.

numerical input problem

A problem that asks learners to enter numbers or specific and relatively simple mathematical expressions. For more information, see [Numerical Input Problem](#).

9.13 O

OLX

OLX (open learning XML) is the XML-based markup language that is used to build courses on the Open edX platform.

For more information, see [What is Open Learning XML?](#).

open response assessment

A type of assignment that allows learners to answer with text, such as a short essay and, optionally, an image or other file. Learners then evaluate each others' work by comparing each response to a *rubric* created by the course team.

These assignments can also include a self assessment, in which learners compare their own responses to the rubric, or a staff assessment, in which members of course staff evaluate learner responses using the same rubric.

For more information, see [Introduction to Open Response Assessments](#).

9.14 P

pages

Pages organize course materials into categories that learners select in the learning management system. Pages provide access to the course content and to tools and uploaded files that supplement the course. Links to each page appear in the course material navigation bar.

For more information, see [Managing the Pages in Your Course](#).

PAL

Phrase alternating line. The PAL standard is a color encoding system for analog videos. It is used in locations such as Brazil, Australia, south Asia, most of Africa, and western Europe.

partner manager

Each EdX partner institution has an edX partner manager. The partner manager is the primary contact for the institution's course teams.

pre-roll video

A short video file that plays before the video component selected by the learner. Pre-roll videos play automatically, on an infrequent schedule.

For more information, see [Adding a Pre-Roll Video to Your edX Course](#).

preview mode

A view that allows you to see all the units of your course as learners see them, regardless of the unit status and regardless of whether the release dates have passed.

For more information, see [Previewing Draft Content](#).

problem component

A component that allows you to add interactive, automatically graded exercises to your course content. You can create many different types of problems.

For more information, see [Working with Problem Components and Adding Exercises and Tools](#).

proctored exam

At edX, proctored exams are timed, impartially and electronically monitored exams designed to ensure the identity of the test taker and determine the security and integrity of the test taking environment. Proctored exams are often required in courses that offer verified certificates or academic credit. For more information, see [Managing Proctored Exams](#).

program

A program is a collection of related courses. Learners enroll in a program by enrolling in any course that is part of a program, and earn a program certificate by passing each of the courses in the program with a grade that qualifies them for a verified certificate.

Several types of program are available on edx.org, including MicroMasters, Professional Certificate, and XSeries programs.

program offer

A program offer is a discount offered for a specific program. The discount can be either a percentage amount or an absolute (dollar) amount.

Progress page

The page in the learning management system that shows learners their scores on graded assignments in the course. For more information, see [Checking Your Progress in a Course](#) in the *EdX Learner's Guide*.

9.15 Q

question

A question is a type of post that you or a learner can add to a course discussion topic to bring attention to an issue that the discussion moderation team or learners can resolve.

For more information, see [Creating Course Discussions](#).

9.16 R

Research Data Exchange (RDX)

An edX program that allows participating partner institutions to request data for completed edx.org courses to further approved educational research projects. Only partner institutions that choose to participate in RDX contribute data to the program, and only researchers at those institutions can request data from the program.

For more information, see [Using the Research Data Exchange Data Package](#).

rubric

A list of the items that a learner's response should cover in an open response assessment. For more information, see the [Rubric](#) topic in [Introduction to Open Response Assessments](#).

See also [Open Response Assessment](#).

9.17 S

seat type

See [enrollment track](#).

section

The topmost category in your course outline. A section can represent a time period or another organizing principle for course content. A section contains one or more subsections.

For more information, see [Developing Course Sections](#).

sequential

See [Subsection](#).

short description

The description of your course that appears on the edX [Course List](#) page.

For more information, see [Course Short Description Guidelines](#).

simple editor

The graphical user interface in a problem component that contains a toolbar for adding Markdown formatting to the text you supply. The simple editor is available for some problem types. For more information, see [Editing a Problem in Studio](#).

single sign-on (SSO)

SSO is an authentication service that allows a user to access multiple related applications, such as Studio and the LMS, with the same username and password. The term SSO is sometimes used to refer to third party authentication, which is a different type of authentication system. For information about third party authentication, see [Third Party Authentication](#).

special exam

A general term that applies to proctored and timed exams in edX courses. See [Timed Exam](#) and [Proctored Exam](#).

split test

See [Content Experiment](#).

subsection

A division in the course outline that represents a topic in your course, such as a lesson or another organizing principle. Subsections are defined inside sections and contain units.

For more information, see [Developing Course Subsections](#).

9.18 T

text input problem

A problem that asks learners to enter a line of text, which is then checked against a specified expected answer.

For more information, see [Text Input Problem](#).

timed exam

Timed exams are sets of problems that a learner must complete in the amount of time you specify. When a learner begins a timed exam, a countdown timer displays, showing the amount of time allowed to complete the exam. If needed, you can grant learners additional time to complete the exam. For more information, see [Offering Timed Exams](#).

third party authentication

A system-wide configuration option that allows users who have a username and password for one system, such as a campus or institutional system, to log in to that system and automatically be given access to the LMS. These users do not enter their system credentials in the LMS.

For more information about how system administrators can integrate an instance of Open edX with a campus or institutional authentication system, see [Enabling Third Party Authentication](#).

transcript

A text version of the content of a video. You can make video transcripts available to learners.

For more information, see [Obtain a Video Transcript](#).

9.19 U

unit

A unit is a division in the course outline that represents a lesson. Learners view all of the content in a unit on a single page.

For more information, see [Developing Course Units](#).

unit navigation bar

The horizontal control that appears at the top of the **Course** page in the LMS. The unit navigation bar contains an icon for each unit in the selected subsection. When you move your pointer over one of these icons, the name of the unit appears. If you have bookmarked a unit, the unit navigation bar includes an identifying flag above that unit's icon.

See also *Course Navigation Pane*.

9.20 V

VBR

Variable bit rate. The bit rate is the number of bits per second that are processed or transferred. A variable bit rate allows the bit rate to change according to the complexity of the media segment.

vertical

See *Unit*.

video component

A component that you can use to add recorded videos to your course.

For more information, see [Working with Video Components](#).

9.21 W

whitelist

In edX courses, a whitelist is a list of learners who are being provided with a particular privilege. For example, whitelisted learners can be specified as being eligible to receive a certificate in a course, regardless of whether they would otherwise have qualified based on their grade.

In the grade report for a course, whitelisted learners have a value of “Yes” in the **Certificate Eligible** column, regardless of the grades they attained. For information about the grade report, see [Interpreting the Grade Report](#).

wiki

The page in each edX course that allows both learners and members of the course team to add, modify, or delete content. Learners can use the wiki to share links, notes, and other helpful information with each other. For more information, see [Using the Course Wiki](#).

9.22 XYZ

XBlock

EdX’s component architecture for writing course components: XBlocks are the components that deliver course content to learners.

Third parties can create components as web applications that can run within the edX learning management system. For more information, see [Open edX XBlock Tutorial](#).

XSeries

A set of related courses in a specific subject. Learners qualify for an XSeries certificate when they pass all of the courses in the XSeries. For more information, see [XSeries Programs](#).