

Contents

HowestX documentation	2
Preface	2
Technical documentation	2
Pre-requisite knowledge	2
edX architecture	4
edX license	9
Development and deployment recipes	9
AWS deployment	15
Usability	24
Comparison of different MOOC platforms	24
Personas	27
Evaluation and Conclusion	27
Is the platform user friendly?	27
Can the platform be put in production easily?	27
Is the platform still being developed?	28
Is the future of the platform certain?	28
Are others also using the platform?	28
Is it easy to adapt the platform?	28
Is is easy to set up a development environment?	28
Is the platform maintainable?	28
Is there internationalisation within the platform?	28
Can we monetise the platform?	29
Can we brand the platform?	29
Is there paid support?	29
What are the possibiltie for the future?	29

HowestX documentation

Preface

This is the documentation for HowestX. HowestX is a pilot project for Howest (Howest University College West Flanders). The goal of this project was to evaluate the feasibility of adapting a MOOC platform. Three groups each evaluated a MOOC platform of their choice, HowestX evaluated the edX Open Platform.

Technical documentation

Pre-requisite knowledge

This section lists pre-requisite knowledge, tools and technologies that the implementors should be familiar with before reading on. This section will briefly introduce these technologies in an edX context, and include links to online documentation to read up on the subjects presented.

Python

Python is a dynamic, high-level programming language with a focus on readability. Most of the edX platform is written in Python.

Virtual environments

Python uses packages to manage dependencies. This can introduce problems when you run multiple Python applications on a single machine: when upgrading a packages for one application, the second application may break if depends on a different version than the new dependencies.

For this, the Python community has introduced *virtual environments*. These provide isolate packages in separate environments. Usually, there will be one environment per application.

Virtual environments (or *venvs* for short) need to be *activated* before you can do work in them. There will be a `bin/` folder in a virtual environment, containing multiple scripts. You can *source* the `activate` script to enable the virtual environment. This is done like this:

```
1 $ . ./venv/bin/activate
```

Notice the dot. This means the argument will be a script to source.

After sourcing, you will notice that your shell `$PATH` has changed. The `python`, `pip` and other executables will be those from inside the virtual environment.

When you install a package with `pip`, it will be installed in the virtual environment.

pip

pip is the most popular package manager for Python. Packages can be installed using `pip install packagename`.

Django

Django is the most popular Python web framework.

Less and Sass

Less and Sass are CSS pre-processors. They extend CSS with variables and functions, leading to a more maintainable frontend codebase. Less and Sass files are compiled down to CSS files so a browser can read them.

Less and Sass serve the same purpose, and have similar features, but are not compatible. Sass is mostly used in the Ruby world, Less is mostly used in the node.js world.

SSH

Secure Shell, or SSH, is a protocol for connecting to remote computers. It's also commonly used for connecting to remote git repositories. SSH is an encrypting protocol that can make use of both cryptographic keys and passwords.

Vagrant

The goal of Vagrant in the edX project is to give all developers the same development environment. It does this through the use of virtual machines: Vagrant is a wrapper around a VM provider backend. For the edX platform, the VirtualBox backend is used.

In edX, Vagrant is used to run the development stack. This makes sure all developers are working with the same software, and the same versions. This makes it easier to reproduce problems and bugs.

Vagrant machines are specified using a **Vagrantfile**, a file that specifies the virtual machine's properties: RAM, vCPUs, what ports that should be forwarded from the host and, most important of all, the base image.

Base images, also known as base boxes, are virtual machines on which projects can base themselves. For example there is an Ubuntu 12.04 base, a FreeBSD base, ...

The first time a box is started, it will be *provisioned*. The base image will be started, and on it, the provision scripts will run. This only has to happen once, but you can force reprovisioning if something is wrong.

Ansible

Ansible is a way to automate provisioning of (virtual) machines. It's used extensively in the edX project due to the complexity of edX deployments. With Ansible, tasks are neatly split up (there are roles to set up repositories, roles to set up Elasticsearch, roles to install Python packages, ...).

LDAP

LDAP (*Lightweight Directory Access Protocol*) is a protocol that allows clients to search, modify and connect to internet directories. We will use it here in an authentication context: we will make clients authenticate to an LDAP server.

LDAP uses port 389 (not to be confused with RDP, which uses port 3389). LDAPS, LDAP over SSL, uses port 636. LDAP uses both TCP and UDP.

Central Authentication Service (CAS)

A CAS server provides Single Sign-On for an organisation. This means that users sign in once, and are then signed in automatically across other applications. This is not the case with LDAP: using LDAP, users have to sign in into every application and authentication happens on the LDAP.

A CAS server provides multiple pluggable backends. You can plug in a SQL database, an LDAP server...

Using a CAS server is comparable to OAuth. When users click *Log in*, they are taken to the CAS server's login page. After authenticating with the CAS server, they are taken back to the application.

edX architecture

The architecture of edX is very complex. To give you a rough idea, in the codebase you can find Python, Ruby, node.js and Java sources. Here, we will take a quick look at the edX Platform architecture, and take a quick look at how edX is expected to be used.

Differences between edX distributions

edX has different distributions (often called *stacks*), each with a different purpose..

Devstack

The *devstack* is intended for local development. Its lifecycle is managed through Vagrant. To create a devstack virtual machine, you download a **Vagrantfile** from edX, and run **vagrant up**. This will download a VirtualBox base image and provision the machine.

Development on the devstack is easy, because Vagrant exposes a few folders (**edx-platform**, **themes ora** and **cs_comments_service**) as NFS mounts, allowing you to edit file on the host computer. This means that you don't have to edit files on the guest.

Fullstack

The *fullstack* is a distribution that runs edX in production mode, with all services on one machine. This is not recommended for production deployments. This distribution is ideal for staging and evaluating.

Production stack

The *production stack* is what is recommended for production. The production stack is hard to deploy, since it's very complex, but an Amazon Web Services template is provided. It contains multiple machines, databases and load balancers.

Persistence

These are the main database systems in use for edX:

- MySQL, a popular relational database
- MongoDB, a popular JSON-based NoSQL database

Some online resources may say that Sqlite is used in the devstack, but this is no longer the case. Devstack provisions a MySQL database.

Usage of MySQL

MySQL is used to keep user data (login data, user progress, ...).

Usage of MongoDB

MongoDB is used to store courses (the courses you design in edX Studio). Historically, this information was stored in XML files. MongoDB was a better fit than a relational database to store this data in.

Authentication

Authentication is handled through standard Django. The Django framework provides a flexible way to handle authentication. It's possible to define multiple *authentication backends*, the framework will try each of them in succession until either one successfully authenticates the user, or none are left. User details, such as login name, email address, and for the standard backend, a password, are saved in a relational database.

CAS

LDAP isn't a great match for edX. It requires the installation of extra packages, requires modifications to the edX platform, and feels clunky.

edX actually has a built-in solution for central authentication: it provides support for a Central Authentication Service server. Enabling a CAS server is a matter of enabling a few simple configuration entries.

However, this is not ideal if you want to allow users to register themselves. You can't easily use the existing capabilities of edX/Django, you would have to write a registration handler and authentication backend for CAS, separate from your edX codebase.

Thus, CAS is not a good match if you want hybrid authentication (authentication through LDAP, but also allowing registering and logging in directly through edX).

Default users

There are a number of default user accounts available:

- `staff@example.com`, password: `edx`, staff account that can create courses
- `verified@example.com`, password: `edx`, student account for testing verified courses
- `audit@example.com`, password: `edx`, student account for testing course auditing (meaning: following the course without paying for it)
- `honor@example.com`, student account for testing honor code certificates

LMS and CMS

The edX platform is split in two main front-facing sites: the LMS (Learning Management System) and the CMS (Content Management System).

The Learning Management System

The *Learning Management System*, or LMS for short, is where students spend their time. This is the main platform, which lists courses, allows students to enroll into courses and allows students to partake in courses.

The Content Management System

The *Content Management System*, often called edX Studio, Studio or CMS, allows teachers to create, edit, preview and publish courses.

Themes

edX now has a flexible theme structure. Aside from configuring to use the new theme, the platform isn't modified in any way when creating or updating a theme.

Themes go into a `themes/` folder. The theme to use can then be specified in the configuration.

Existing themes

There are only a few existing themes on which we can base a custom theme.

- The built-in theme: this is the theme that is inside the edX platform core, and the default that gets loaded. This means that you have to edit the platform to update the theme, which is not something you want to do, especially since edX supports external themes.
- The Stanford theme: this is the theme that Stanford University uses. This theme is not responsive, which was a dealbreaker for us.
- IONISx theme: the theme developed by the French IONIS Education Group. This theme is based on Bootstrap, and responsive by default.

Choice of theme

We decided to adapt the IONISx theme to our needs. Being responsive from the start is a huge plus, and the external theming support helps provides a clean split between frontend and backend code.

Less in the IONISx theme

The IONISx theme used Less (not Sass, like the edX platform). The edX platform doesn't understand Less. Installing a Less pre-processor on the platform would require a whole slew of packages to be brought in and an extra build pipeline. That's not ideal.

Our solution is to check in the compiled files into source control. That's not very pretty, but it works. A `grunt` watcher watches for changes, and executes the build.

This makes it easy to release the theme: the production edX platform can just pull in the repository.

Certificates

Some people just want to participate in a course for personal education, but others want to participate in order to obtain proof they are qualified in certain subjects. These last people require certification in order to proof they have the necessary skills to get a certain job or be accepted in certain educational institutions.

Certification in edx

Edx has the capability to provide a couple of forms of certification, here's a list of the types of certificates you can obtain in edx.

Honor code certificates

These are basic certificates that say you passed a certain course, there is however no verification at all about your identity. This makes this an easy to obtain certificate, but with little real value.

Verified certificates

These certificates say you passed a course and verifies your identity. This happens by using pictures from your webcam and your ID. To obtain one of these certificates, a user must specify he is interested in one when he participates in a course. The user will also have to pay a fee for this certificate that can vary on a course-to-course basis.

XSeries certificates

XSeries certificates can be obtained when you pass for a series of courses that all fit within a certain subject. These might be useful in implementing the Howest tracks. There could be an XSeries certificate for a web developer, for example. XSeries certificates also require a fee that can vary.

What about archived courses?

There are no certificates available for archived courses. These courses can only be audited and some features, such as the discussion forums, may not be available.

Are refunds possible?

Yes, there is a period of 2 weeks from the start date of a course that someone can change their mind. He can still continue his course, but without the intention of obtaining a certificate. Of course, whether you offer refunds and for how long is a decision every organisation must make for itself.

What happens when someone obtains a certificate?

The certificate will be generated and sent to the user via email. In that email will be a link to a PDF-file that contains the certificate.

Configuration

Configuration in edX is very complex. Here are some hints on the possible files:

- `lms.env.json` and `cms.env.json` are generated by provisioning script. You can edit them to test.
- `common.py` sets all the default variables, which can be overwritten by Ansible configuration
- `server-vars.yml` (full location: `/edx/app/edx_ansible/server-vars.yml`) sets configuration that will be used to provision the machine. This will be used to populate `*.env.json`. A list of all possible option can be found [here](#).

Be wary that there are `.json` and `.yaml` files. JSON and YAML have different syntaxes.

edX license

For a detailed description, see the [Open edX Licensing page](#) on the Open edX website.

The gist is that `edx-platform`, `edx-configuration` and `edx-ora2` are released under the AGPL. This means that derived code must also be released under the AGPL, that you may modify the code, and that you must disclose the source code. Other source code is Apache licenced, meaning that you do not need to distribute sources.

Development and deployment recipes

Vagrant recipes

Installing Vagrant on a Mac OS machine

Mac OS users can download Vagrant as a `.dmg` from [its website](#). Installation works as usual, but you will have to install VirtualBox seperately from [here](#).

Installing Vagrant on a Linux machine

We suggest that Linux users **do not** install Vagrant using their package manager. The Vagrant in the Debian and Ubuntu repositories is often out of date, and this may cause strange errors when developing on the edX platform. Instead, we encourage Linux users to install Vagrant from [its website](#).

VirtualBox may be installed through the package manager, but Debian users will have to [enable non-free packages](#).

Setting up a Vagrant devstack machine

You can set up a Vagrant virtual machine for local development and testing. These instructions are for Mac and Linux systems, Windows users can use [Cygwin](#), but, as mentioned previously, we do not recommend this.

```
1 $ mkdir howestx
2 $ cd howestx
3 $ curl -L https://raw.githubusercontent.com/edx/configuration/
  master/
4   vagrant/release/devstack/Vagrantfile > Vagrantfile
5 $ vagrant plugin install vagrant-vbguest
6 $ vagrant up
```

This is what happens:

- Inside a new `howestx` directory, we download a `Vagrantfile`.
- We then install the `vagrant-vbguest` plugin. This only needs to happen once.
- Then we start up the virtual machine using `vagrant up`:
- If this is the first time you set up an edX Vagrant development box, it will have to download a VirtualBox base image. This image will be roughly 3GB.
- If this is the first time this specific virtual machine starts up, the machine be provisioned. This happens automatically. The virtual machine will download and run ansible scripts. This may take a while (15 minutes). This only happens once, unless you destroy or reprovision the virtual machine.
- If the virtual machine already exists, it will be booted. This only takes a few minutes.

Accessing a running Vagrant devstack machine

You can access a running Vagrant devstack machine through SSH:

```
1 $ vagrant ssh
```

Shutting down a running Vagrant machine

You may shut down a running Vagrant machine using `halt`:

```
1 $ vagrant halt
```

It is very important to **halt all Vagrant virtual machines before shutting down the host machine**. This does not happen automatically. If you shut down the host with running Vagrant guests, corruption of the virtual machines may occur.

Dealing with corrupted Vagrant virtual machines

If your Vagrant guest has gotten corrupted or otherwise unworkable, you may do one of two things:

- Reprovision the virtual machine: reprovisioning the virtual machine will re-run the Ansible scripts on the existing virtual machine.
- Destroy the virtual machine: destroying the Vagrant virtual machine will remove the NFS mounts and all VirtualBox files.

Usually, destroying the virtual machine makes more sense, since you will then start from a clean slate and creating a new Vagrant virtual machine won't take much longer than reprovisioning (provided that the VirtualBox image is still cached).

You may reprovision a Vagrant virtual machine like this:

```
1 $ vagrant provision
```

You may destroy a Vagrant virtual machine like this:

```
1 $ vagrant destroy
```

Note that destroying a Vagrant virtual machine will not remove the **Vagrantfile**. You can immediately **vagrant up** again. Also, if you have ever brought this box up, the VirtualBox base image will be cached, and Vagrant will not have to download the base image.

Advanced Vagrant trouble

Occasionally, Vagrant will still have trouble (happens rarely, e.g. when the host crashes, or if you manually edit Vagrant's private files). You can then try to follow the following steps:

- Purge Vagrant from your system
- Delete `~/.vagrant.d`, if it still exists
- Delete all Vagrant VMs using the VirtualBox GUI
- Remove all Vagrant NFS exports from `/etc/exports`
- Reinstall Vagrant

Devstack recipes

This section assumes you have a running devstack virtual machine with Vagrant.

Access to the Vagrant guest

SSH access

File access

Fixing MongoDB

Occasionally, MongoDB will have problems. This will happen when it quit unexpectedly. The `mongod` service will then refuse traffic and throw errors. You can try this to fix it:

```
1 $ su vagrant
2 $ sudo rm /edx/var/mongo/mongodb/mongod.lock
3 $ sudo service mongod restart
```

Accessing the MySQL server

You may access the MySQL server by using this command:

```
1 $ mysql -u root
```

This will drop you in a SQL shell. The primary database edX uses is `edxapp`.

It's also possible to access the database from a GUI (for example, MySQL Workbench). Turn on the *Tunnel through SSH* option on with the IP address of the Vagrant guest, use `vagrant` as the username and `vagrant` as the password, and you're good to go.

Theming recipes

Bootstrapping a custom theme based on the IONISx theme

You can bootstrap a custom theme on devstack by following these steps:

- Install global dependencies: install [node.js](#) from its website, then install `grunt-cli` and `bower` globally (you only need to do this once on every development machine):

```
$ npm install -g grunt-cli bower
```
- Clone the IONISx theme to a repository of your own. This will allow you to push back commits you make to this repository.
- Go into the `themes/` folder of edX on your local machine. It's mounted into the Vagrant virtual machine, so you can access it from your own computer.
- Clone the repository here. You may rename the folder. The name of the folder is your theme name, and this is very important. This is what you will have to provide to the configuration files.

- Rename the `_ionisx.scss` file located in the `static/sass/` folder to match your theme name (name of the folder). Your theme will not function if you don't do this.
- Now install the dependencies of the theme:
`$ npm install $ bower install`
- Run `grunt`. The default mode will watch until you press `^C`. As soon as you save, Grunt will pick it up and recompile all the assets. The edX platform can't process the Less files, so it's important to compile these down to CSS.

Using a custom theme in devstack

To use a custom theme in the devstack, you can update `lms.env.json`. This file is located in the home directory of the `edxapp` user.

```
1 $ vim ~/lms.env.json
```

Two modifications need to be made:

1. In the `FEATURES` list, set `USE_CUSTOM_THEME` to `true`
2. Set the `THEME_NAME` entry to the name of your theme (for example, `howestx-theme`)

It should look like this:

```
1 {
2   ...
3   "FEATURES": {
4     ...,
5     "USE_CUSTOM_THEME": true
6   },
7   ...
8   "THEME_NAME": "howestx-theme",
9   ...
10 }
```

You need to run (or restart if it's already running) `paver` to see the effect in your browser:

```
1 edxapp@~/edx-platform$ paver devstack lms
```

Editing a custom theme

Using a custom theme in fullstack

Edit the `/edx/app/edx_ansible/server-vars.yml` file on the fullstack server and add the following variables:

```
1 edxapp_use_custom_theme: true
2 edxapp_theme_name: 'howestx-theme'
3 edxapp_theme_source_repo: 'git://github.com/howestx/howestx-theme
  .git'
4 edxapp_theme_version: 'HEAD'
```

Then run the update script:

```
1 $ sudo /edx/bin/update edx-platform master
```

You may specify another branch of the `edx-platform` by changing `master` in the update command.

Deployment recipes

Fullstack deployment

On an Ubuntu 12.04 machine

You can install a fullstack deployment on an Ubuntu 12.04. It's important that you do this on a "fresh" server (nothing installed yet) using the following:

```
1 $ sudo apt-get update -y
2 $ sudo apt-get upgrade -y
3 $ sudo reboot
```

After rebooting, you can install using the one-step install script:

```
1 $ wget https://raw.githubusercontent.com/edx/configuration/master
  /util/install/sandbox.sh -O - | bash
```

You can also install using named release:

```
1 $ export OPENEDX_RELEASE=named-release/birch
2 $ wget https://raw.githubusercontent.com/edx/configuration/
  $OPENEDX_RELEASE/util/install/vagrant.sh -O - | bash
```

Be aware that this does **not** work with Ubuntu 12.04.5 (the standard on Azure and DigitalOcean)! You should install the latest release using the first script.

Elasticsearch SSL certificate trouble

In the latest release, you may see an error with Elasticsearch's certificate:

```

1 TASK: [elasticsearch | download elasticsearch]
    *****
2
3 failed: [localhost] => {"failed": true, "item": ""}
4
5 msg: Failed to validate the SSL certificate for download.
    elasticsearch.org:443. Use validate_certs=no or make sure
    your managed systems have a valid CA certificate installed.
    Paths checked for this platform: /etc/ssl/certs, /etc/pki/ca-
    trust/extracted/pem, /etc/pki/tls/certs, /usr/share/ca-
    certificates/cacert.org, /etc/ansible

```

A work-around is to edit the Ansible configuration file:

```

1 $ vim /var/tmp/configuration/playbooks/roles/elasticsearch/
    defaults/main.yml

```

Change the variable `elasticsearch_url` from `https` to `http`. **Only use this as a temporary work-around for testing!** The certificate will probably be fixed in a coming release. You now need to reprovision the server:

```

1 $ cd /var/tmp/configuration/playbooks && sudo ansible-playbook -c
    local ./edx_sandbox.yml -i "localhost,"

```

Using a edX platform fork

You use a custom `edx-platform` for the fullstack too. Add the following to `/edx/app/edx_ansible/server-vars.yml`:

```

1 edx_platform_repo: "https://github.com/HowestX/edx-platform.git"

```

Obviously using your own repository. Then you need to remove the existing platform:

```

1 $ sudo rm -rf /edx/app/edxapp/edx-platform

```

Then reprovision the server:

```

1 $ sudo /edx/bin/update edx-platform release

```

`release` is a git branch. You can always specify a custom branch.

Production deployment

AWS deployment

There are public AMI's available for AWS, for europe that's `ami-aa76d0dd`. It's recommended to deploy this on an `t2.medium` instance.

Start the server and connect to it via `ssh` (the user is 'ubuntu')

```
1 chmod 400 {path-to-keypair}
2 ssh -i {path-to-keypair} ubuntu@{public-ip}
```

Then update the codebase

```
1 sudo /edx/bin/update configuration release
2 sudo /edx/bin/update edx-platform release
```

In case you get an ‘Unable to resolve host’ error, ass the following to `/etc/hosts`

```
1 127.0.1.1 {whatever ip}
```

If there’s an error in the ‘edx-platform-release’ execute the following

```
1 cd /edx/app/edxapp/edx-platform
2 sudo -u edxapp git remote prune origin
```

You can now connect with the LMS on port 80 and the CMS on port 18010. If you get a 502 error, restart the mongo ansible role

```
1 cd /edx/app/edx_ansible/edx_ansible/playbooks && sudo /edx/app/
  edx_ansible/venvs/edx_ansible/bin/ansible-playbook -i
  localhost, -c local run_role.yml -e 'role=mongo' -e '
  mongo_create_users=True'
```

The default authentication for the site is

```
1 username: edx
2 password: edx
```

The dafeult logincredentials are

```
1 user: staff@example.com
2 password: edx
```

If you run into a 500 error, do the following

```
1 cd /edx/app/edxapp/edx-platform && sudo -u www-data /edx/bin/
  python.edxapp manage.py lms syncdb --migrate --settings aws
2 cd /edx/app/edxapp/edx-platform && sudo -u www-data /edx/bin/
  python.edxapp manage.py cms syncdb --migrate --settings aws
```

More on managing a fullstack: <https://github.com/edx/configuration/wiki/edX-Managing-the-Full-Stack>

Fullstack recipes

Here are some things you might want to do on a fullstack server.

Enabling course previewing

Edit `server-vars.yml`, add the variables

```
1 EDXAPP_LMS_BASE: ""
2 EDXAPP_PREVIEW_LMS_BASE: ""
3 EDXAPP_CMS_BASE: ""
```

Give these variable the appropriate content, an example could be

```
1 EDXAPP_LMS_BASE: "howestx.be"
2 EDXAPP_PREVIEW_LMS_BASE: "howestx.be"
3 EDXAPP_CMS_BASE: "studio.howestx.be"
```

Reprovision the server

If, for any reason, you want to reprovision the fullstack server, use the following command:

```
1 $ sudo /edx/bin/update edx-platform release
```

You can use any branch instead of `release`.

Change the name of the platform

Edit `/edx/app/edx_ansible/server-vars.yml`, and add the following:

```
1 EDXAPP_PLATFORM_NAME: 'howestX'
```

Where `howestX` is the name of your platform.

Enable search

By default, there is no search possibility in fullstack. Edit `/edx/app/edx_ansible/server-vars.yml`, and add the following:

```
1 FEATURES:
2   - ENABLE_DASHBOARD_SEARCH: true
3   - ENABLE_COURSEWARE_SEARCH: true
4   - ENABLE_COURSE_DISCOVERY: true
5   - ENABLE_COURSEWARE_INDEX: true
6   - ENABLE_LIBRARY_INDEX: true
7   - SEARCH_ENGINE: "search.tests.mock_search_engine.
      MockSearchEngine"
```

Now reprovision the server:

```
1 $ sudo /edx/bin/update edx-platform release
```

Deleting a course

If you have created a few test courses, you will have noticed there is no `delete` functionality in edX Studio. You have to delete courses manually from MongoDB.

To do this, log in to your fullstack server and start a `mongo` shell:

```
1 root@howestx-staging:~# mongo
2 MongoDB shell version: 2.6.10
3 connecting to: test
4 >
```

We first need to select the correct database, just like in SQL shells:

```
1 > use edxapp
2 switched to db edxapp
```

Now we can check if we are using the correct database. The name of the document where courses are stored is `modulestore`. An empty `find()` will select all records, and `pretty()` will make it print out pretty JSON:

```
1 > db.modulestore.find().pretty()
2 {
3   "_id" : {
4     "tag" : "i4x",
5     "org" : "howestX",
6     "course" : "CAT1_W110",
7     "category" : "about",
8     "name" : "entrance_exam_id",
9     "revision" : null
10  },
11  "definition" : {
12  ...
```

This is the correct database. Now we can remove the record we want:

```
1 > db.modulestore.remove({ "_id.course": "CAT1_W110"})
```

LDAP recipes

Testing LDAP on *NIX systems

You can use the `ldapsearch` program (packaged in a package called `ldap-utils` by most distributions) to query LDAP servers and test them.

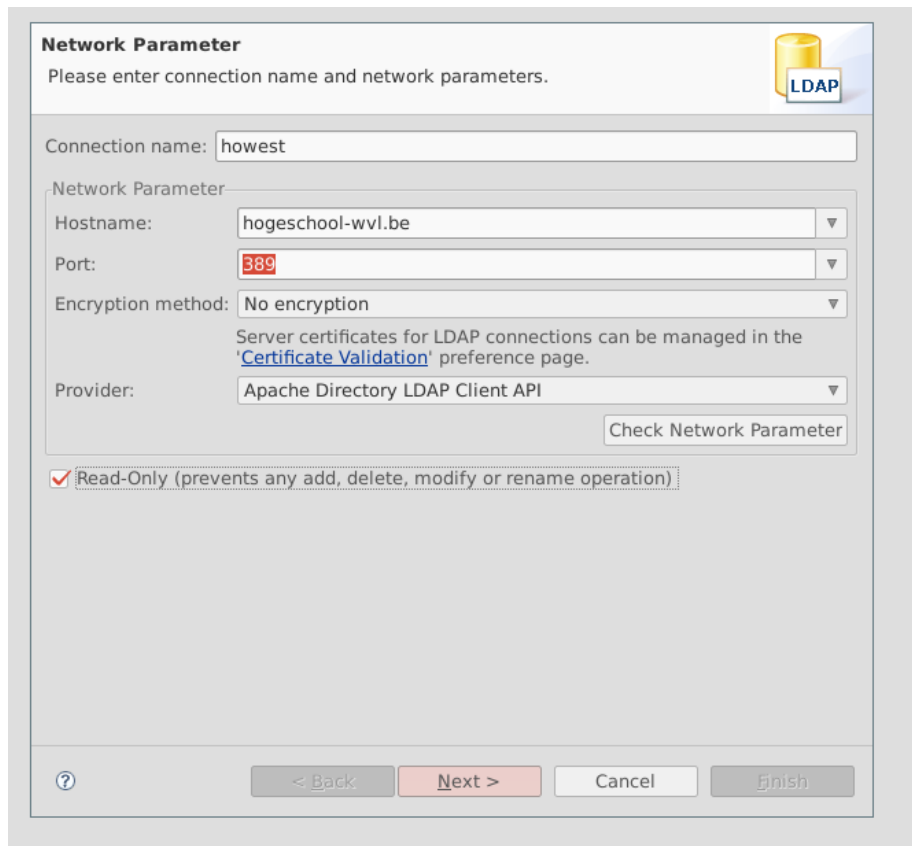
This example command can be used to *bind* to an LDAP server:

```
1 $ ldapsearch -D "glenn@howestedx.local" -W -H ldap://howest-test-
   ad.cloudapp.net -b "dc=howestedx,dc=local"
```

Using a GUI for LDAP testing

You can use [Apache Directory Studio](#) (not to be confused with Apache Directory), a tool based on Eclipse, to experiment with LDAP.

After downloading, run the Apache Directory Studio and create a new connection:



The screenshot shows the 'Network Parameter' dialog box in Apache Directory Studio. The title bar is 'Network Parameter' with a yellow cylinder icon and the text 'LDAP'. Below the title bar, it says 'Please enter connection name and network parameters.' The dialog has several input fields: 'Connection name:' with the value 'howest', 'Hostname:' with 'hogeschool-wvl.be', 'Port:' with '389', 'Encryption method:' with 'No encryption', and 'Provider:' with 'Apache Directory LDAP Client API'. There is a 'Check Network Parameter' button. Below these fields, there is a checkbox labeled 'Read-Only (prevents any add, delete, modify or rename operation)' which is checked. At the bottom, there are four buttons: a help button '?', '< Back', 'Next >', 'Cancel', and 'Finish'.

Network Parameter
Please enter connection name and network parameters.

Connection name: howest

Network Parameter

Hostname: hogeschool-wvl.be

Port: 389

Encryption method: No encryption

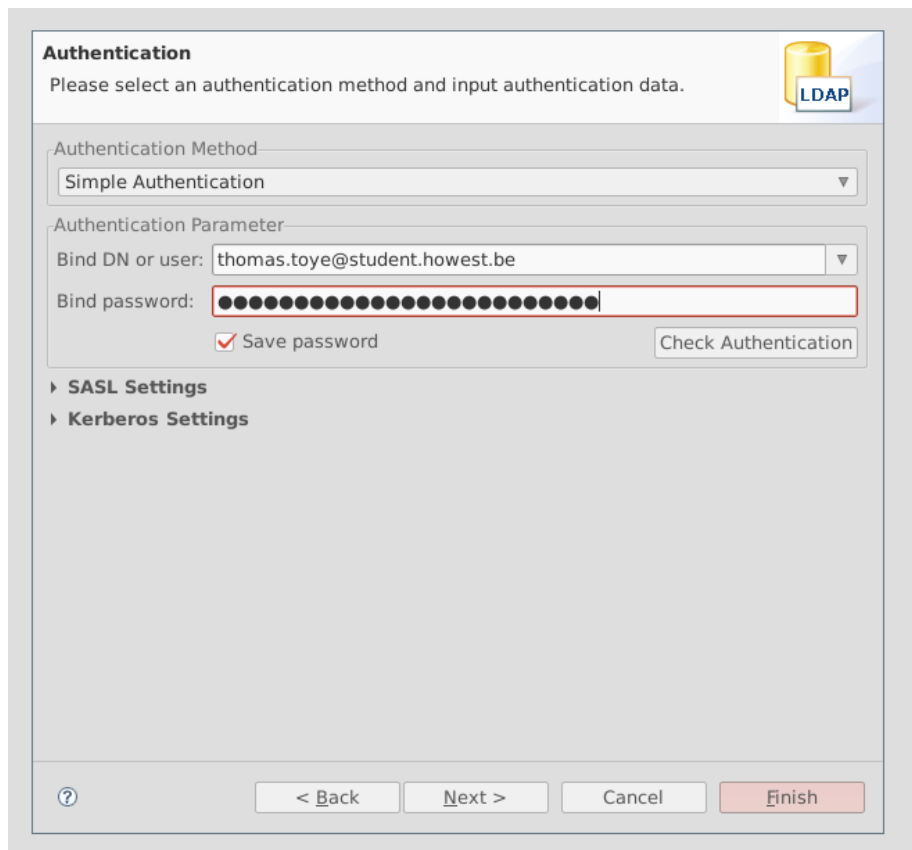
Server certificates for LDAP connections can be managed in the '[Certificate Validation](#)' preference page.

Provider: Apache Directory LDAP Client API

Check Network Parameter

☒ Read-Only (prevents any add, delete, modify or rename operation)

? < Back Next > Cancel Finish



Authentication
Please select an authentication method and input authentication data.

Authentication Method
Simple Authentication

Authentication Parameter
Bind DN or user: thomas.toye@student.howest.be
Bind password: [redacted]
☒ Save password

▶ SASL Settings
▶ Kerberos Settings

? < Back Next > Cancel Finish

Now we can browse the LDAP server:

Using LDAP with Python

Note: some networks (the one at Howest GKG in particular) block LDAP traffic to outside networks. You may circumvent this by using a VPN, or by using an LDAP server inside that network.

You can try this out in a Vagrant devstack box. We recommend working in a Python virtual environment. First of all, install the `python-ldap` package and its dependencies:

```
1 $ sudo apt-get install libldap2-dev libsasl2-dev
2 $ pip install python-ldap
```

Now you can start the Python interpreter and import `python-ldap`:

```
1 $ python
2 Python 2.7.10 (default, Jun 1 2015, 16:21:46)
3 [GCC 4.9.2] on linux2
```

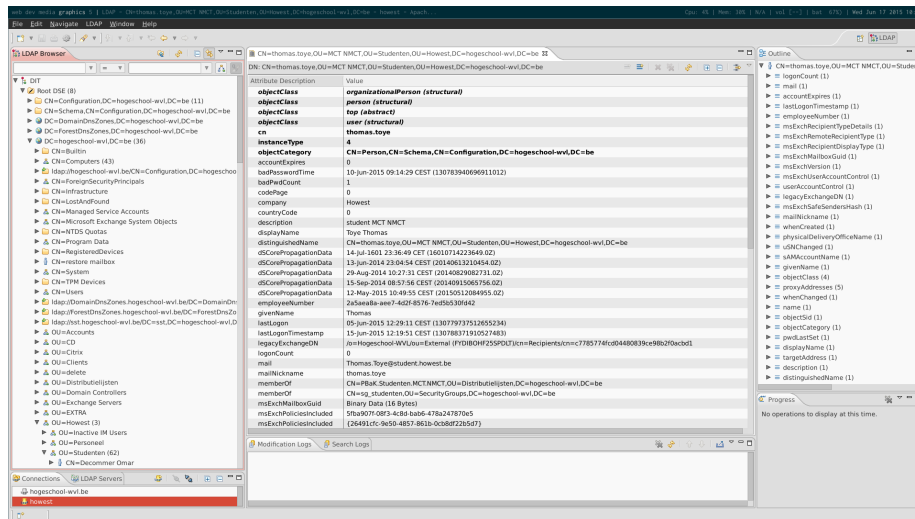


Figure 1: An overview of user information retrieved over LDAP

```

4 Type "help", "copyright", "credits" or "license" for more
  information.
5 >>> import ldap
6 >>>

```

You can now create a connection to the LDAP server. This will only create a connection object, it won't connect yet.

```

1 >>> conn = ldap.initialize('ldap://howest-test-ad.cloudapp.net')

```

We need to set up a few parameters on this object before we can start querying:

```

1 >>> conn.protocol_version = 3
2 >>> conn.set_option(ldap.OPT_REFERRALS, 0)

```

Now we can try to authenticate (*bind*) to the LDAP server:

```

1 >>> conn.simple_bind_s('glenn@howestedx.local', '123')

```

If you don't get any exceptions, you have connected successfully.

Using LDAP with edX

Certificates recipes

Turning on certificates in fullstack

To turn on certificates in fullstack, first edit the `lms/envs/common.py` file:

```
1 $ vim /edx/app/edxapp/edx-platform/lms/envs/common.py
```

Search for an option called `CERTIFICATES_HTML_VIEW`, and set it to true:

```
1 ...
2 # Certificates Web/HTML Views
3 'CERTIFICATES_HTML_VIEW': True,
4 ...
```

Now sync and migrate the databases:

```
1 $ . . . /venvs/edxapp/bin/activate
2 $ ./manage.py cms syncdb --settings=aws
3 $ ./manage.py lms syncdb --settings=aws
4 $ ./manage.py cms migrate --settings=aws --delete-ghost-
   migrations
5 $ ./manage.py lms migrate --settings=aws --delete-ghost-
   migrations
```

Importing and exporting courses

EdX already provides the functionality to import and export courses. This provides an easy way to for example take a course from your platform and put it on edX's own platform.

To Export or import a course, open the course in the studio. Then click on **settings**. In the sub-menu that opens you have the options to import and export. Courses are saved in a `.tar.gz` format.

Internationalisation recipes

When offering an online service it might be useful to provide that service in multiple languages to expand your possible userbase. But properly translating an entire service takes a lot of effort and time, luckily edX also provides a full translation of it's contents in a lot of languages. Setting it up is a breeze.

Offering multiple languages for users to choose

First of all you will need a `.transifexrc` file. This file contains the information that edX will use to login to transifex, the service that provides the translations.

Make a `.transifexrc` file on the following location

```
1 $ nano ~/.transifexrc
```

Give it the following content

```
1 [https://www.transifex.com]
2 hostname = https://www.transifex.com
3 username = user
4 password = pass
5 token =
```

Change 'user' and 'pass' to your own credentials. Token is to remain empty. Then run the following commands

```
1 $ . /edx/app/edxapp/edxapp_env
2 $ cd /edx/app/edxapp/edx-platform
```

Now we must make sure that the languages we wish to support are marked as active in `conf/locale/config.yaml`. Open that file and uncomment any language you wish to support.

```
1 $ nano conf/locale/config.yaml
```

Any new languages have to be pulled in using the following command

```
1 $ tx pull -l <lang_code>
```

Run the following command to make sure the changes take effect

```
1 $ paver i18n_fastgenerate
```

Restart the lms and cms

```
1 $ sudo /edx/bin/supervisorctl restart edxapp:
```

Now go to the language settings on your Django admin panel, you can find these at `<your_website>/admin/dark_lang`. For example: `http://www.howestx.be/admin/dark_lang`.

There you must add a configuration, this configuration contains what languages users can select. Note that everything must be typed in lowercase, that ' ' becomes '-', and everything should be comma separated. For example:

```
1 en,nl-nl,fr,ar,es-419
```

Save this configuration. Now edX will be displayed in the user's preferred language. Users can select this from their user account setting.

Setting the default language for the platform

If you would want to change the default language of the entire platform, you must edit the `EDXAPP_LANGUAGE_CODE` in `/edx/app/edx_angular/server-vars.yml`.

Usability

Comparison of different MOOC platforms

If a modern college is interested in offering the capability of massive online learning, a MOOC platform is the way to go. However, developing one is an extremely complex and time consuming process. Luckily there are already a lot of free open source MOOC platforms that could, in theory, be easily adapted to fit a college's specific needs. Here, we are going to briefly compare a couple of MOOC platforms.

We will compare the following platforms: * edX * Peer 2 Peer University * openMooc

We are interested in the following criteria: * Usability (is the platform easy to use as an end user?) * Activity (is the platform still actively developed?) * Complexity and adaptability (is it easy to adapt to our needs?)

edX

EdX is a large open source MOOC platform that was developed as a joint venture between the Massachusetts Institute of Technology and the Harvard university. UC Berkeley has also joined.

The platform already has a lot of success, boasting more than 3 million users (as of October 2014), and being used by institutions such as Stanford.

Usability

EdX as a platform is very usable to an end user, it offers an intuitive user interface which is also rather attractive to the eye. This does not translate over to mobile, as the default theme is not mobile ready. It also seems to have a lot of functionality already built-in.

Activity

We were happy to see edX being a very active platform. Not only does it have a remarkable amount of users, it also has a lot of developers behind it. When looking up the github page of the platform, we saw commits as recent as 2 hours ago.

It's also a rather 'hot topic' online, showing a surge in activity.

Complexity

EdX so far appears to be the perfect choice for any school interested in setting up a MOOC platform of their own. However, its sheer complexity might cause some issues.

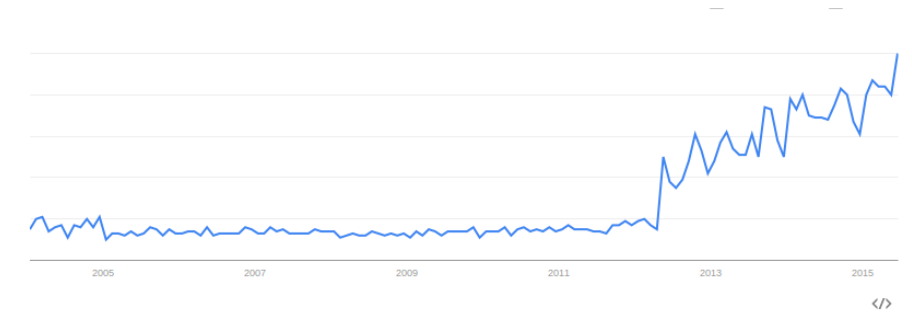


Figure 2: Google trends graphic on edX

EdX is an absolutely huge platform, that is also highly fragmented. Simply finding the source code of a specific part of it is a huge task on its own.

However, there is also an immense amount of documentation and an active community ready to help. edX claims to offer support to an institution attempting to adapt edX to its own needs.

Peer 2 Peer University

P2PU was founded with funding from the Hewlett foundation, the Shuttleworth foundation and the university of California Irvine. It was founded by a group of people that felt the existing ways of online learning were inadequate, especially the social aspect.

Usability

P2PU offers an attractive and intuitive user interface. It also appears to be mobile ready, which is a big plus.

Activity

P2PU is definitely still used by a large user base, but it shows signs of decline. Its GitHub is not that active and contains open issues as old as 3 years (the oldest bug dates from 25 May 2012). Google trends also reveals a steady decline.

Complexity

P2PU is what could be called a medium sized platform. It appears to be a nicely organised platform, that should be easy to adapt. It also offers quite a bit of documentation. The big issue for us is the fact that there are so many old open issues. Is the development grinding to a halt?

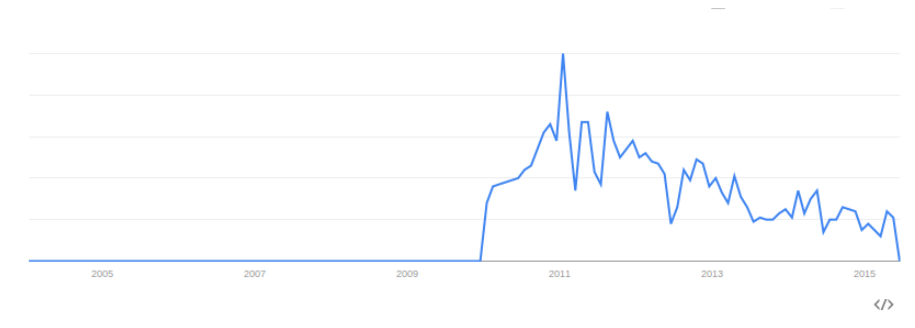


Figure 3: Google trends graphic on P2PU

OpenMooc

OpenMooc is a MOOC platform that hails from Spain, it was made to promote virtual education by using IT in higher education. Since it was made in Spain, multilanguage support has been built into its core.

Usability

OpenMooc's default user interface is atrocious. It's also not mobile ready. A lot of work would have to be put in to make this a usable platform. A couple of things did draw our attention however, like the platform's ability to follow up on your own as well as other people's questions and it's native support for LDAP.

Activity

As far as we can tell, OpenMooc isn't a widely used platform. A couple of smaller institutions allegedly use it but there is no proof of any major organisation showing interest in this platform. It's GitHub is also very inactive, most commits are already a year old. Google trends also confirms this.



Figure 4: Google trends graphic on OpenMooc

Complexity

This is a rather small MOOC platform. It seems rather well organised, but very badly documented. Also setting up this platform seems to not be a trivial matter.

Conclusion and choice of platform

There are a lot of platforms out there, the platforms we compared are the most prominent. They all offer the ability to host a decent MOOC platform, yet edX and P2PU are by far the largest and offer the most usability out of the box. Deciding between these two is not easy, edX offers more functionality whereas P2PU offers a seemingly easier to adapt codebase. However, edX is seeing way more use and development. P2PU as a platform seems to be heavily on the decline, so trying to build a lasting MOOC solution based on that platform appears unwise.

In the end, edX seems the best choice. It may be a complex system that will require a thorough analysis, but its active development, good support and documentation should easily make up for that.

Personas

See the attachments for this documentation.

Evaluation and Conclusion

We will now evaluate the edX platform according to criteria specified by Howest

Is the platform user friendly?

EdX as a platform is very user friendly. For students it has a very nice user interface that allows them to quickly find what they need. Teachers can easily create, manage and share courses.

Can the platform be put in production easily?

This is where the most issues will be found. EdX is an immensely complex system that is difficult to set up. Furthermore when we tried to setup a named release, it kept failing to deploy. EdX was also developed with AWS in mind, not really offering a lot of support for Azure.

Is the platform still being developed?

There is still a very healthy and active development going on. When checking the github page we generally see commits that are 2-3 hours old. More and more people are also starting to use edX and there are even consultants who are specializing in this platform.

Is the future of the platform certain?

EdX has an immense userbase and a lot of big organisations like MIT and Harvard backing it up. Google trends is also showing a positive trend.

Are others also using the platform?

Yes, it's a very widely used platform. A lot of major institutions have opted for edX as their MOOC platform of choice. It also boasts more than 3 million users (as of 2014).

Is it easy to adapt the platform?

It's a very big and complex platform, adapting it is not an easy matter. Sometimes simply finding the correct files to work in can be a task on its own. However, there is plenty of very detailed documentation and a very active and helpful community that should alleviate most issues.

Is it easy to set up a development environment?

In order to set up a development environment, vagrant is preferable. On Linux and Mac this is an easy thing to use, but on Windows it has proven to be very difficult. It's also very fragile and can break easily, requiring you to reset the development environment.

Is the platform maintainable?

Once the development environments are set up, maintaining is easy. The back-end and front-end are fully separated and fully pluggable, leading to easy development.

Is there internationalisation within the platform?

EdX provides internationalisation that is a breeze to set up. You can make use of a simple web interface. It makes use of transifex to obtain its translations. Sadly, dutch is not yet translated.

Can we monetise the platform?

Yes, it's possible to ask money for verified certificates. This model has worked for organisations such as edx.org, MongoDB university,...

Can we brand the platform?

It's possible to brand the entire platform, there is no dependence on edX for this matter. It's also very easy to do this.

Is there paid support?

No paid support is offered, however edX provides some support to any organisation looking to set up their own fork. The community is also a source of help just like the excellent documentation. If necessary there are even paid consultants specialised in edX.

What are the possibilities for the future?

This platform can certainly be used as a base to develop upon. There is a lot that can be done with interactivity, Azure Active Directory can be integrated and courses could be published on edx.org. The next named release is to be called **Cypress** and will be released between July 15th and 20th with a first release candidate planned for June 26th. It will allegedly contain changes in social profile, 3rd party auth and single sign on, among other features. The 3rd party auth and single sign on are looking very interesting, as they would fit Howest's needs perfectly.