



Technical: Principles and Details

Open edX is...

Python/Django REST services,
React micro-frontends,
and legacy server-rendered templates
consisting of millions of lines of code
across ~200 repositories
written by hundreds of people
over the past ten years

Standard Stack: Backend

Application Code

- Python 3.8
- Django 3.2 (LTS)
- Django REST Framework
- Celery
- pytest

Supporting Servers

- MySQL 5.7 (soon to be 8.0)
- Memcached (caching)
- Redis (async jobs queue)
- Elasticsearch (search, may be replaced by OpenSearch)
- *Planned: Kafka (inter-service async communication)*

Stuff we're trying to get rid of:

- MongoDB
- Ruby / Sinatra (forums uses this)

Standard Stack: Frontend

Application Code

- Domain-specific micro-frontend applications
- JavaScript / React / Redux
- Paragon Design System based on Bootstrap
- Shared i18n/analytics/logging/auth ([frontend-platform](#))
- Shared build and deployment process ([frontend-build](#))

Stuff we're trying to get rid of:

- Legacy server-rendered templates
- Previous iterations of UI frameworks and pattern libraries

Open edX UIs and Services

User-facing MFEs

Account
Admin Portal
Ecommerce
Enterprise Learner Portal
Gradebook
Learning
Payment
Profile
Programs Learner Portal
Program Manager
Publisher

User-facing Web Sites

Credentials
Ecommerce
Insights
LMS
Studio (CMS)

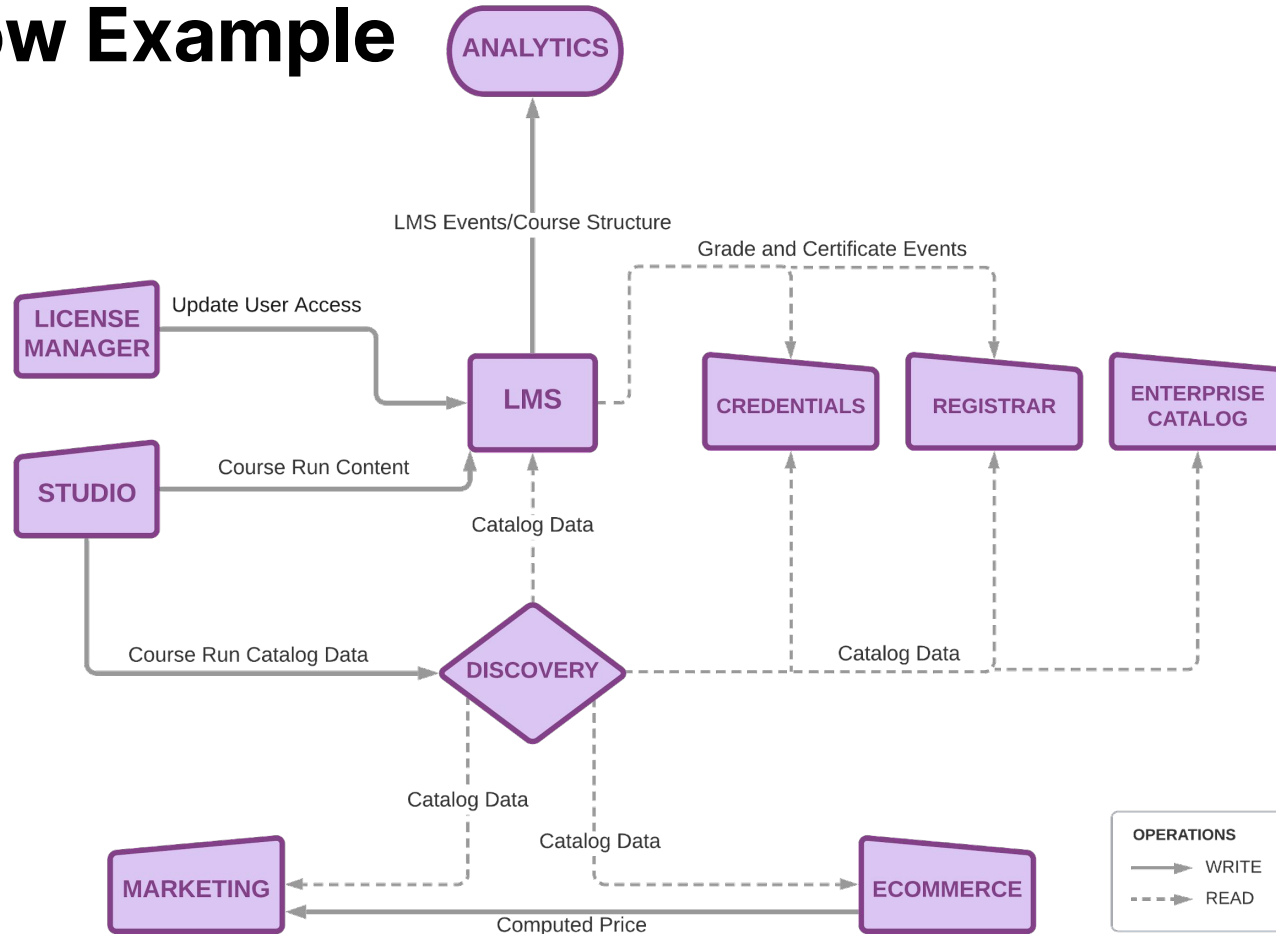
Native Apps

Android App
iOS App

API Backend Services

Analytics API
Blockstore
Course Discovery
Enterprise Catalog
Forums
License Manager
Notes
Registrar
XQueue + Watcher

Data Flow Example



Development Environments

Devstack

- Docker-based development environment
- Ansible + Docker = Pain
- Runs the full range of Open edX services

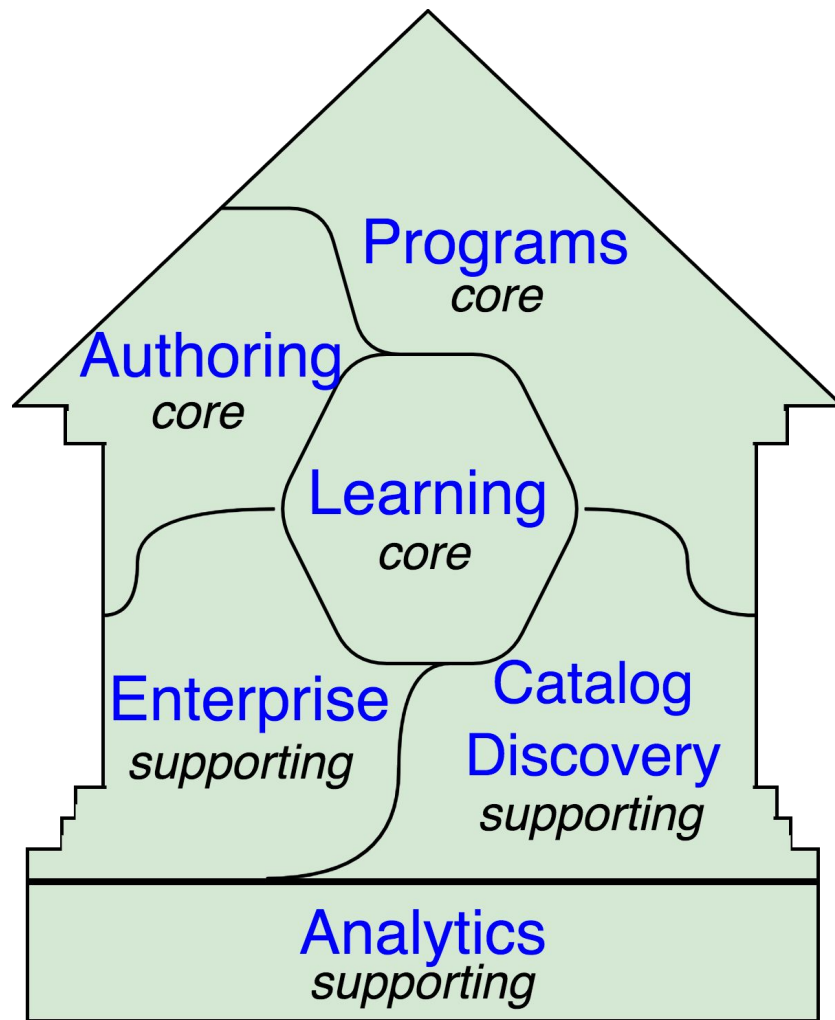
Tutor

- Official Docker-based Open edX distribution
- Developed by Régis Behmo
- Faster, more lightweight, easier to set up
- Plugin system makes extension more straightforward
- Incomplete support for services, MFE development
- [Tutor Adoption Initiative](#) in progress

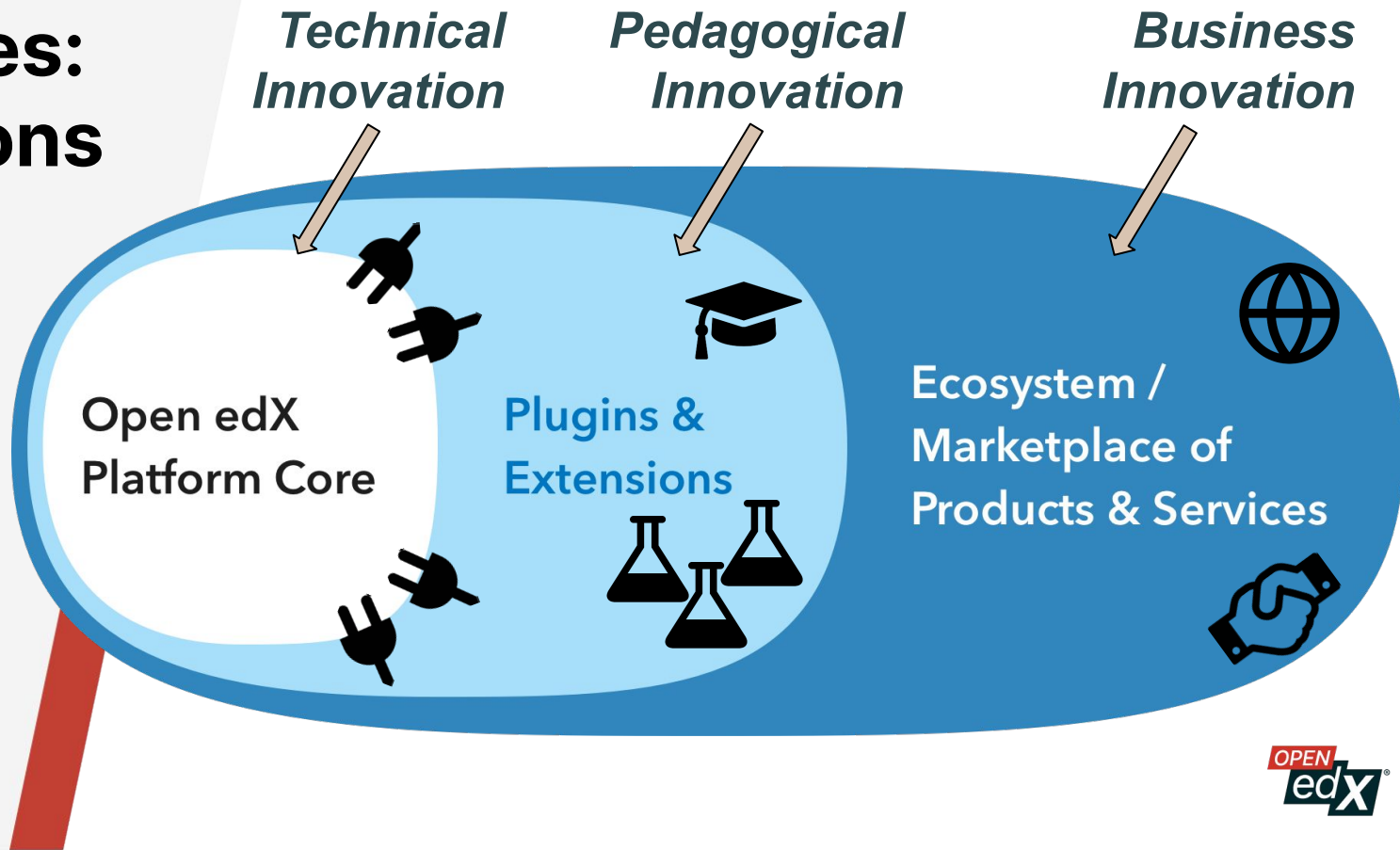
Architectural Principles: BEES



Architectural Principles: Boundaries (Domain Concepts)



Architectural Principles: Extensions



CHEMICAL EQUATION PROBLEM (1 point possible)

Some problems may ask for a particular chemical equation. Practice by writing out the following reaction in the box below.

$$\text{H}_2\text{SO}_4 \longrightarrow \text{H}^+ + \text{HSO}_4^-$$

$$\text{H}_2\text{SO}_4 \rightarrow \text{H}^+ + \text{HSO}_4^-$$

Some tips:

- Use real element symbols.
- Create subscripts by using plain text.

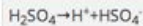
Lab 2

(1 point possible)

A circuit that combines two or more signals combines the signals generated by two voltages.

- V1 is a 1 kHz square wave that varies between 0V and 5V.
- V2 is a 5 kHz sine wave that varies between 0V and 5V.

Please design a circuit that mixes V1 and V2.

$$\text{H}_2\text{SO}_4 \rightarrow \text{H}^+ + \text{HSO}_4^-$$


Some tips:

- Use real element symbols.
- Create subscripts by using plain text.

Lab 2

(1 point possible)

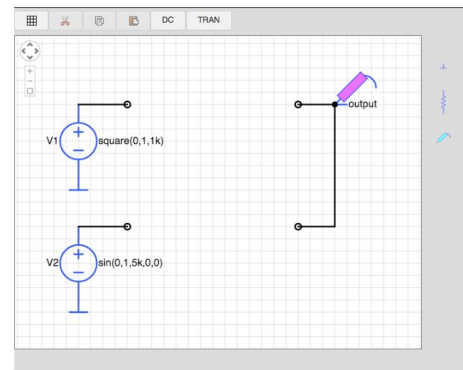
A circuit that combines two or more signals is called a *mixer*. In this lab, your goal is to build a mixer that combines the signals generated by two voltage sources, V1 and V2, where:

- V1 is a 1 kHz square wave that varies between 0V and +1V, and
- V2 is a 5 kHz sine wave that varies between -1V and +1V.

Please design a circuit that mixes V1 and V2 to produce Vout such that

$$V_{\text{out}} \approx \frac{1}{2}V_1 + \frac{1}{6}V_2.$$

Enter your circuit below, using the appropriate configuration of resistors. Please do not modify the wiring or parameters of the voltage sources -- your goal is to take the signals they generate and combine them, not to change what is generated. Run a 5ms transient analysis to verify the correct operation of your circuit. We will be checking for the transient waveform at the "output" node.



Describe your favorite meal, and write a persuasive argument. Support your position with facts and examples from your research.

Your response

Mmmm sushi

2 Learn to Assess Responses

✓ COMPLETE

3 Assess Peers

✓ 3 COMPLETED

4 Assess Your Responses

✓ COMPLETE



Architectural Principles: Events

- Events help keep systems decoupled
- Example: Many systems update when course content changes
- Django Signals are an in-process method we use extensively
- [openedx-events](#) is an evolution of this
 - makes writing and maintaining plugins easier
 - proposed and created by eduNEXT
 - accelerated with edX funding
- openedx-events will eventually work across services via Kafka
 - this can replace some of our ad hoc data replication

Architectural Principles: Standards

- Leverage ed tech industry standards where possible
- LTI 1.3 Advantage certified
- xAPI, Caliper ([OEP-26](#), [event-routing-backends](#))
- Limited Common Cartridge / QTI conversion support
- Proctoring Services?
- Comprehensive Learner Records?
- OneRoster?