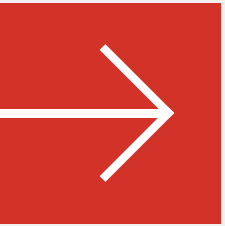


Technical: Principles and Details

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)

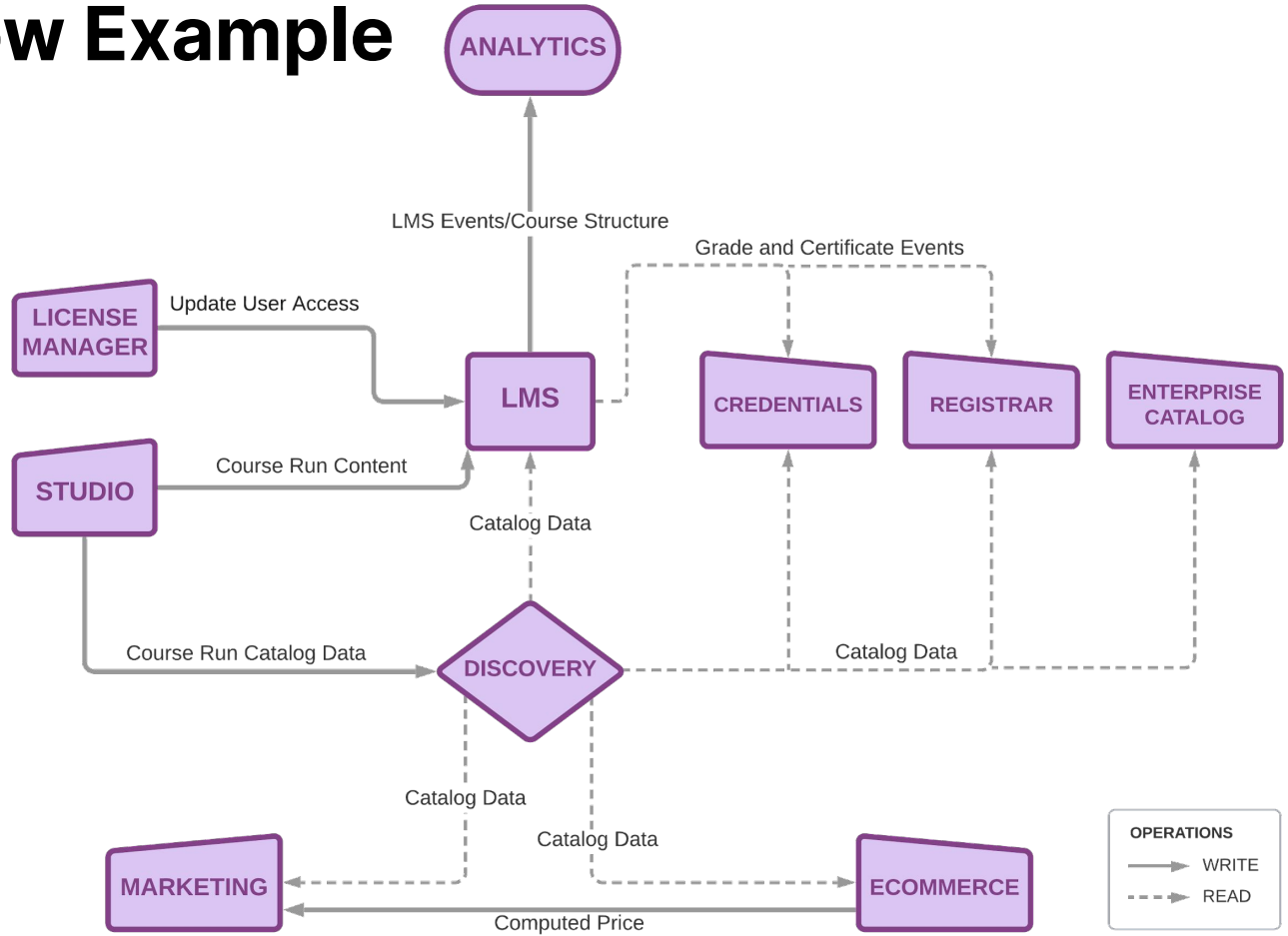
Native Apps

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

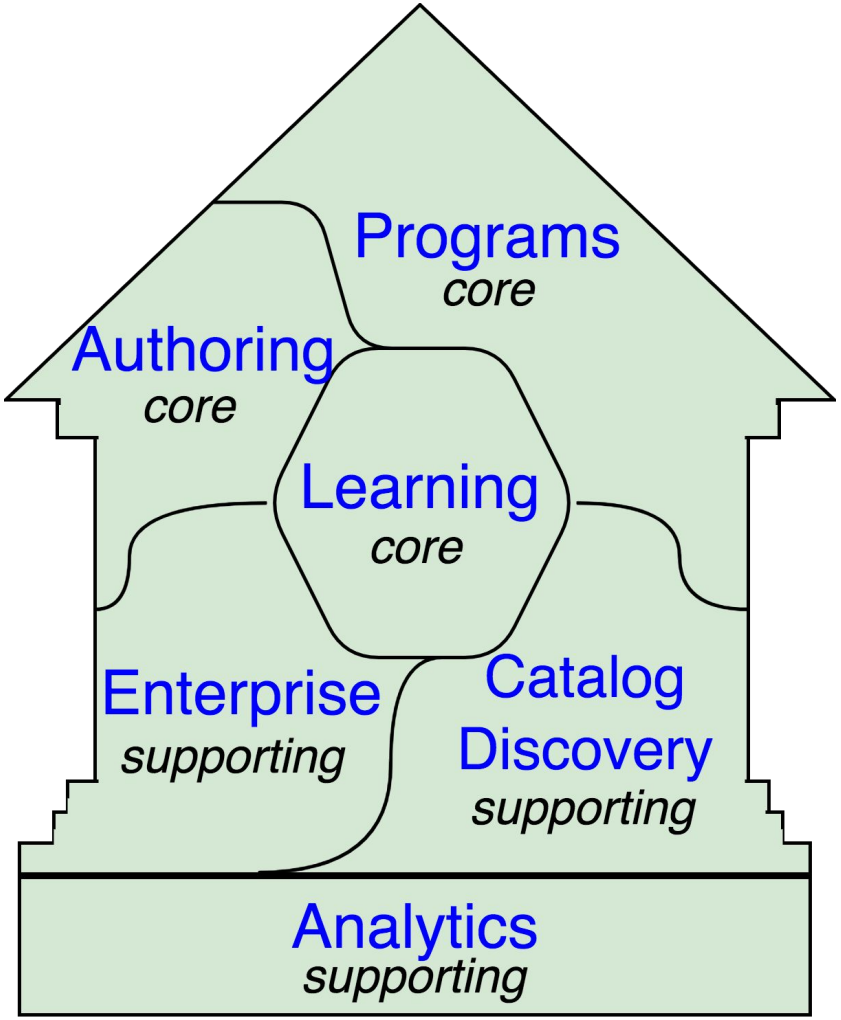
- Docker-based development environment
- Ansible + Docker = Pain
- Runs the full range of Open edX services

Tutor

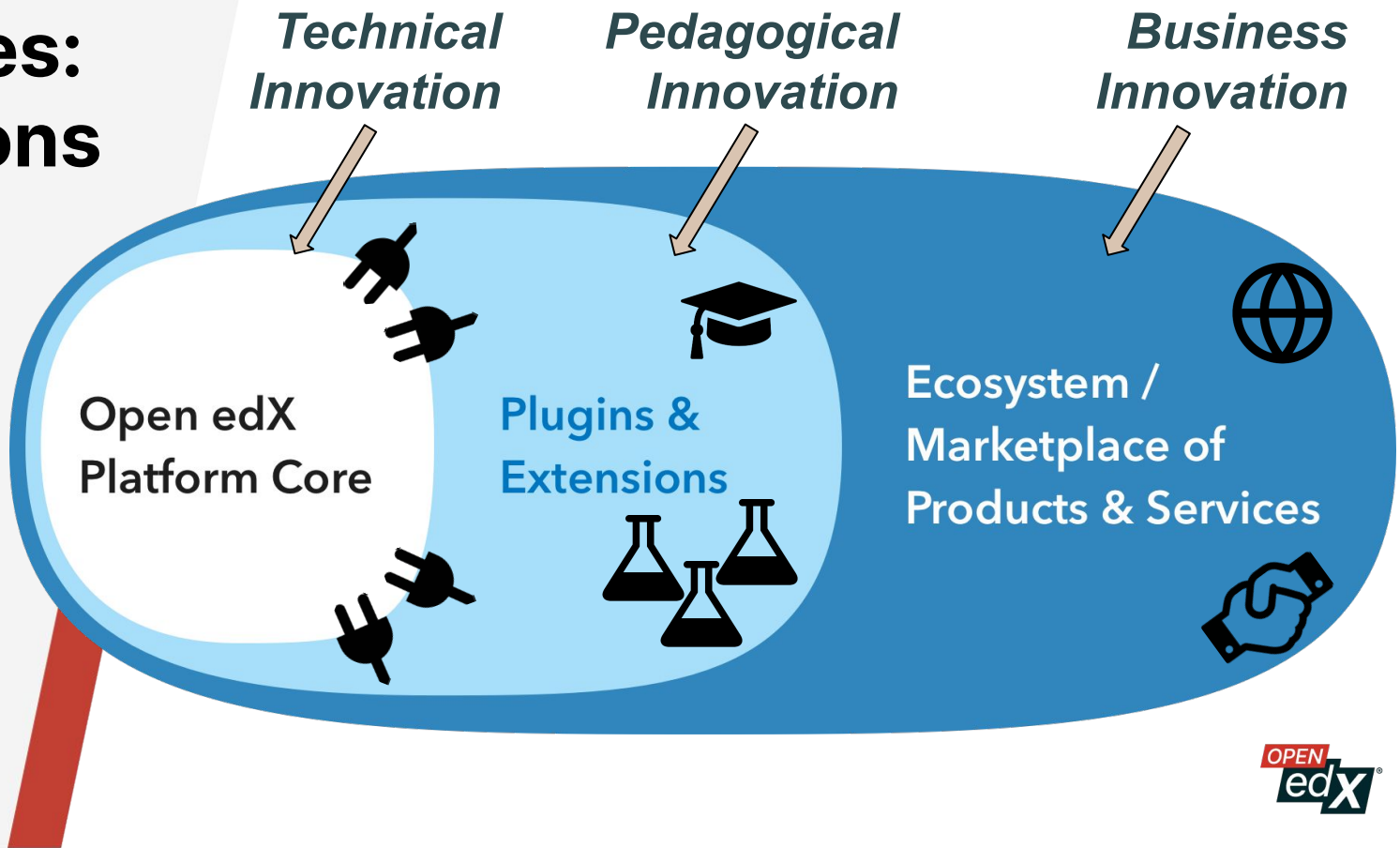
- Official Docker-based Open edX distribution
- Developed by Régis Behmo
- Faster, more lightweight, easier to set up
- Plugin system makes extension more straightforward
- Incomplete support for services, MFE development
- Tutor Adoption Initiative in progress

Architectural Principles: BEES

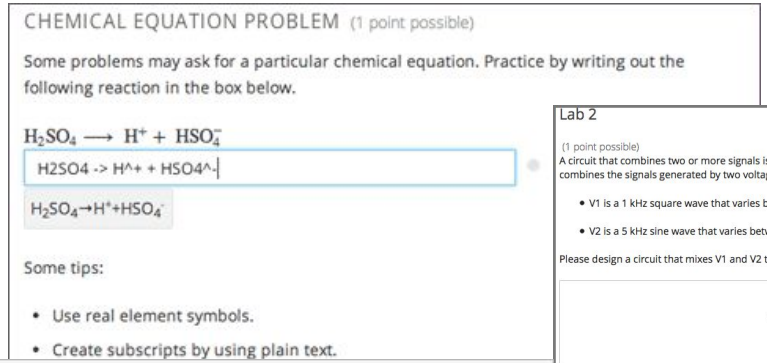
Architectural Principles: Boundaries (Domain Concepts)

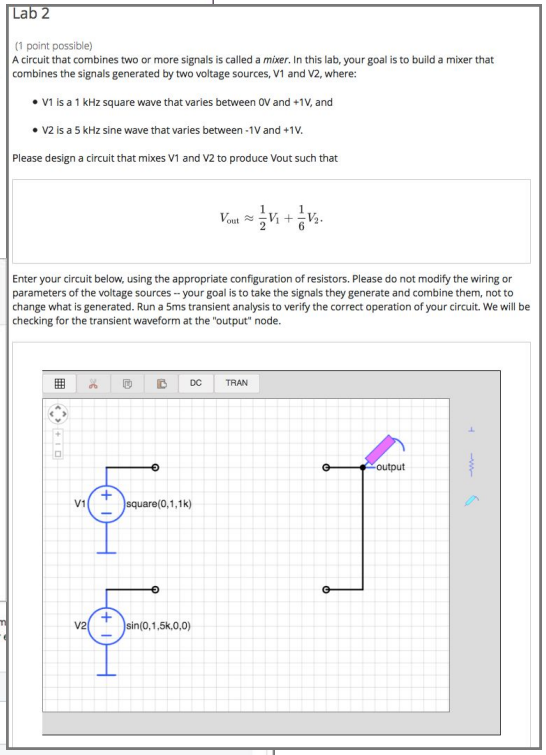


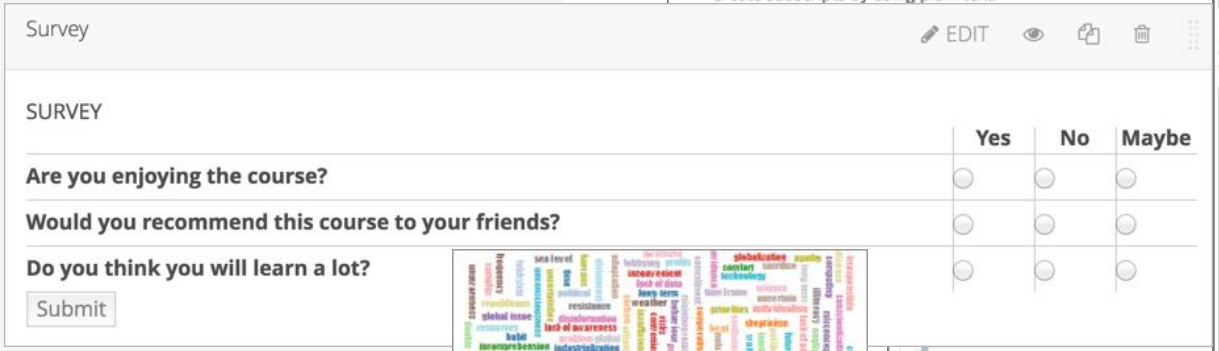
Architectural Principles: Extensions

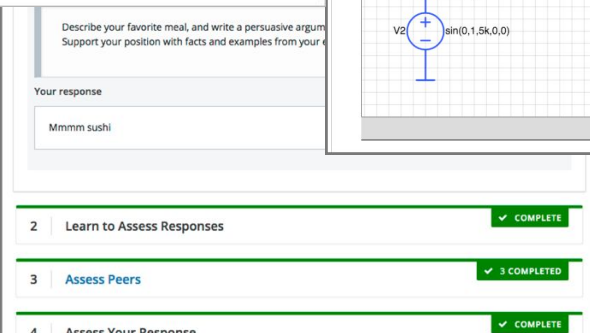


Extensions Example: XBlock









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 a 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?