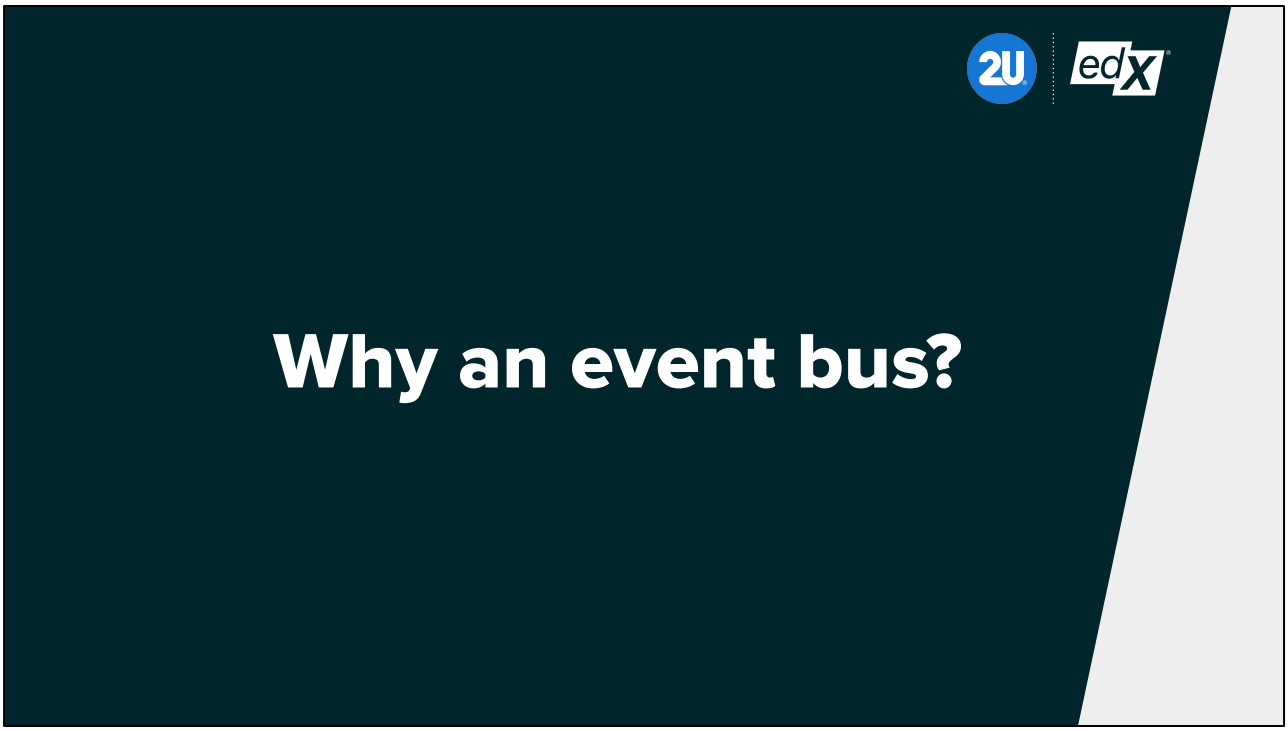
- Acknowledge team

Agenda

- Why an event bus?
- Current stop
- Boarding the bus
- Riding safely
- Next stops
- More questions?



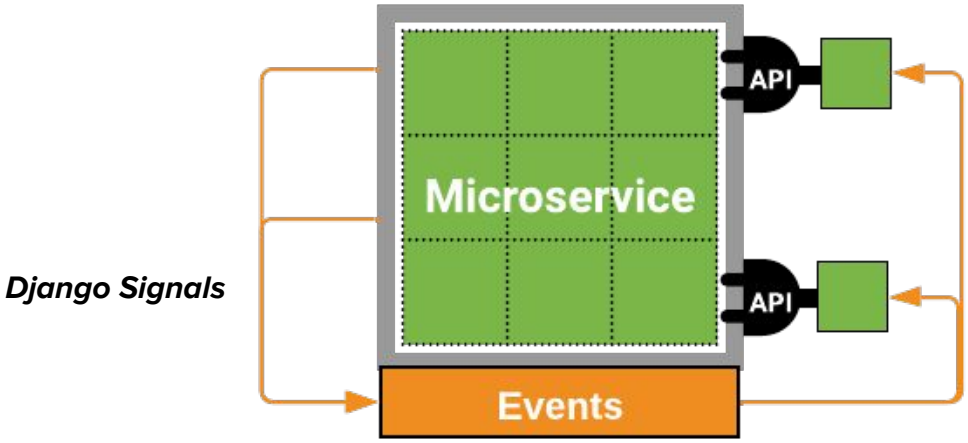
- I’m really starting with “What”, but only to get to the “Why”. It’s not just because it’s shiny.

Why an event bus?

If you are **communicating** across **Open edX services**,
the **event bus** is your new friend.

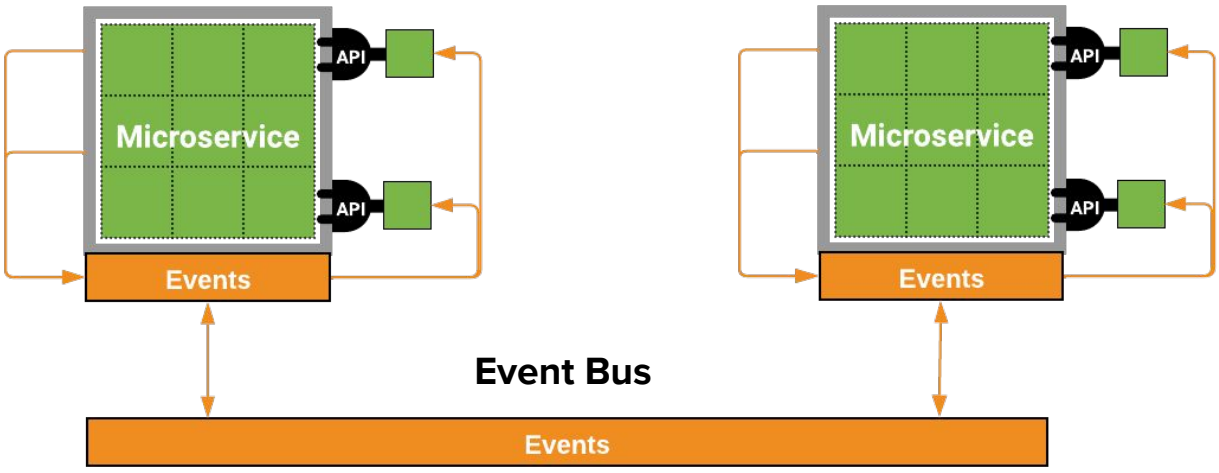
- New paths of communication, or maintenance of old paths.
- Could potentially be used with other services communicating with the Open edX platform
- It's been on our architectural wish list for many years, and is finally here.

Why an event bus?



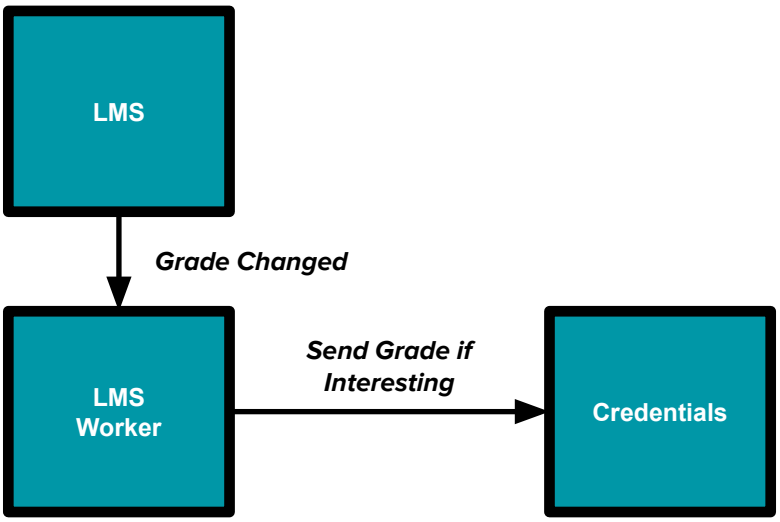
- Hooks framework
- Extensibility
- Plugins
- Hooks: Events
- Django Signals

Why an event bus?



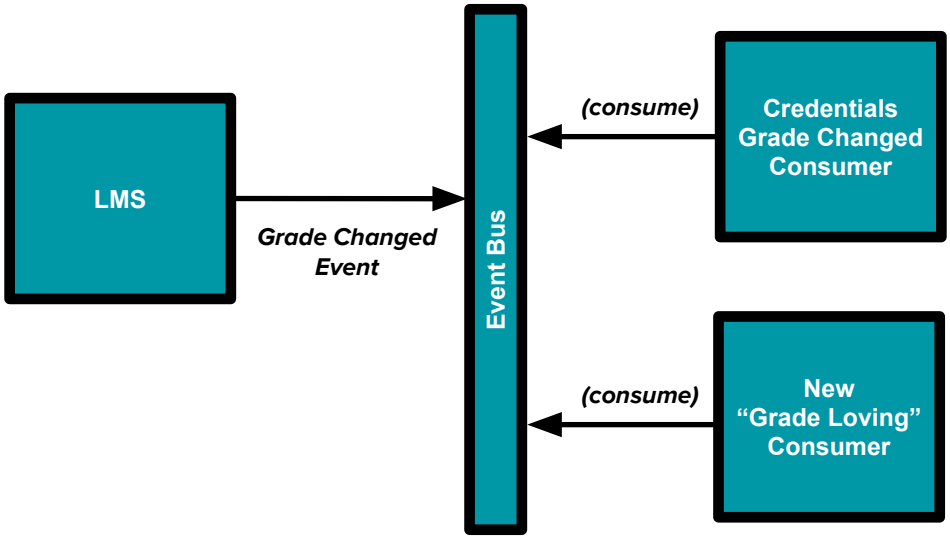
- Extending the same concept, but pushing it across services.

Why an event bus?



- Breaks boundaries
- Tightly coupled
- Need to limits server-to-server calls
- Reliability issues

Why an event bus?

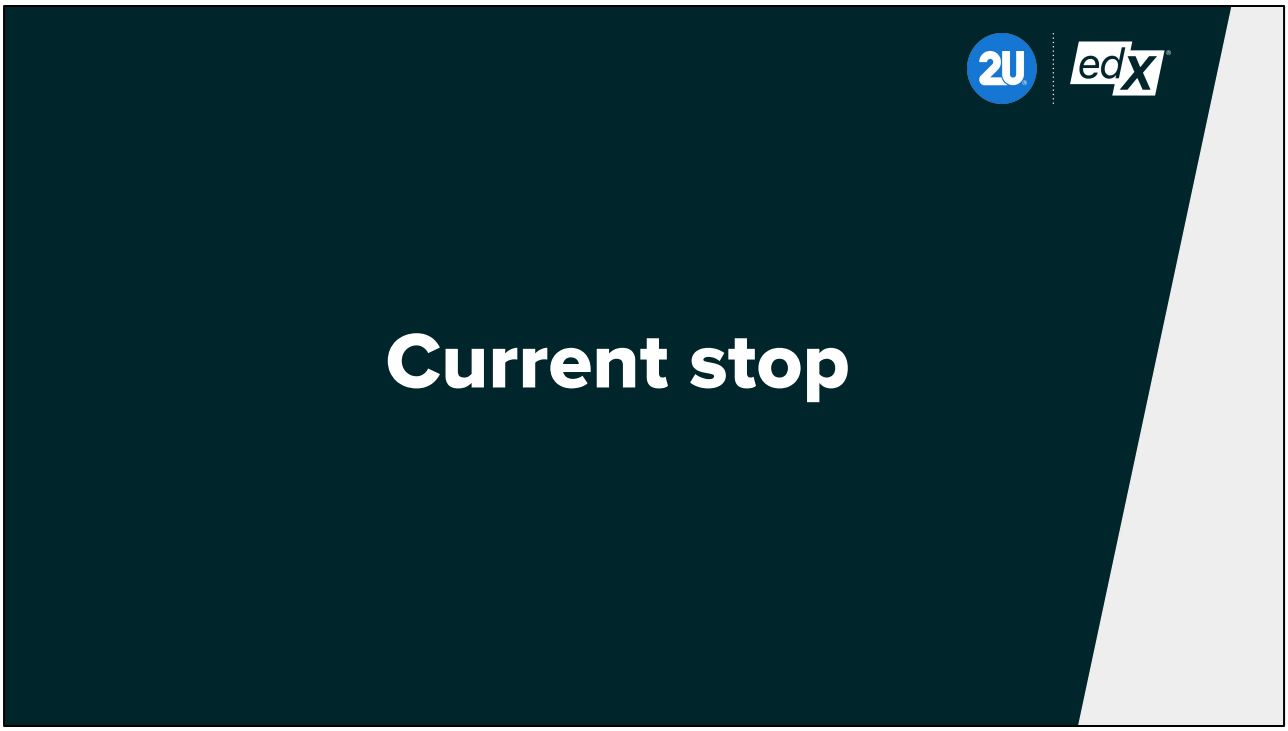


- Consumer can decide what is "interesting"
- Loose coupling
- More resilient
- Extensibility via new consumers that have their own fondness for grades.
- Consumers may or may not need data redundancy, but that is as simple as writing the data.

Why an event bus?

- More features (quantity)
  - Loose coupling
  - Clear bounded contexts
- More types of features (quality)
  - Resilient communication
  - Data redundancy
  - Extensibility
- Eliminate expensive, delayed, batch synchronization

-

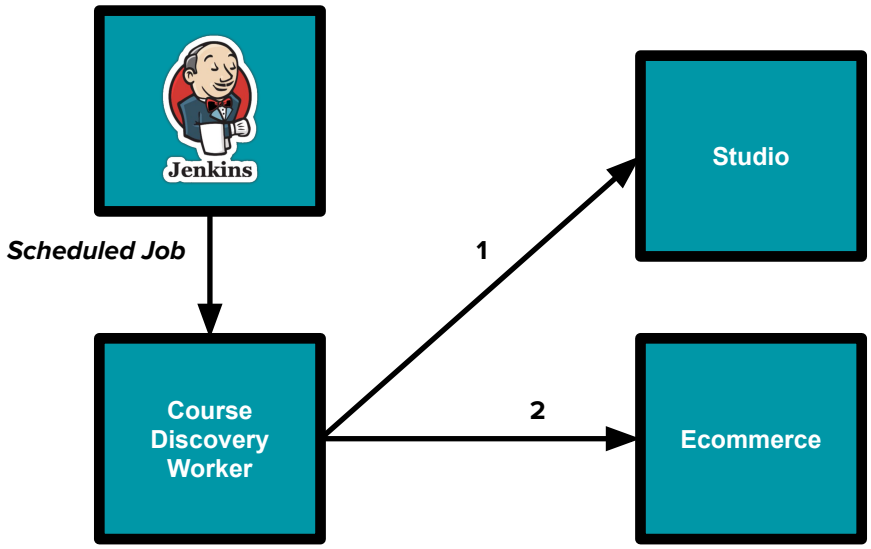


- Or, where are we at now.

Current stop (History)

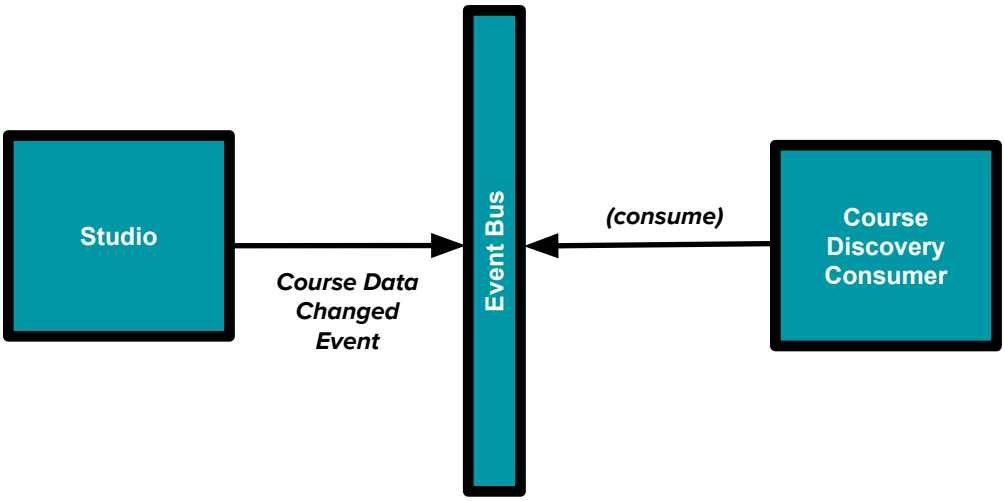
- Happy path with an Enterprise use case
  - Confluent Kafka
  - Avro serialization
  - Consumer container
- Abstraction layer foundations
- OEP-52: Event Bus Architecture (Draft)

Current stop (History)



- Extremely oversimplified picture of sending data from our course authoring service to our catalog service.
- The job is big, slow, flaky, [and tightly coupled]. It also loaded all data for all courses.

Current stop



- We're using Kafka. Events can be replayed.
- Event data could be used to inform other services. Ecommerce as an example.
- Extensibility, Loose Coupling, and other Architectural Jargon.
- Update only the things that are getting changed. Less work.
    - Sometimes things aren't really changing, and that's ok too.

Current stop

Performance

Scheduled Job (Before) Median Latency	Event Bus (After) Median Latency
4-12 hours? (when successful)	< 0.2s

How it makes people feel

“I could cry this is so incredible”

- Why wasn’t this possible before? The event bus opened up new possibilities.

Current stop

- OEP-52: Event Bus Architecture => Provisional
- Fully-functional Kafka implementation
- CloudEvent headers
  - OEP-41: Async Server Event Message Format
- Error and audit logging and monitoring
- Replay capabilities
- `COURSE_CATALOG_INFO_CHANGED` in production for edX.org
- Abstraction layer progress
- Onboarding documentation

- OEP-41 compliance. Header examples: id, time, source
- Audit and error logging. Monitoring in New Relic. (I don't know what the rest of the community uses.)

- Or, where are we at now.

Producing an event

- Discuss with owner
- One-time: Configure
- Create topic in Kafka
- Send event as Django signal
- Write signal receiver
  - Use rollout toggle
  - Send over bus

- Potential for more of this to move to configuration in the future
- One-time per service

Producing an event

```
def make_my_event_happen():
    ...
    MY_EVENT_HAPPENED.send_event('my_top_level_key': {event_data})     # 1 - initial send


@receiver(MY_EVENT_HAPPENED)                                           # 2 - receiver
def listen_for_my_event(sender, signal, **kwargs):
  if (MY_EVENT_ENABLED.is_enabled()):                                  # 3 - rollout toggle
    openedx_events.get_producer().send(
      signal=MY_EVENT_HAPPENED,                                        # 4 - signal
      topic='my_topic',                                                # 5 - topic
      event_key_field='my_primary_key_field',                          # 6 - key
      event_data={'my_top_level_key': kwargs['my_top_level_key']}      # 7 - event data
      event_metadata=kwargs['metadata']                                # 8 - metadata
)
```

- Lifted from how-to linked at end of presentation

Consuming an event

- One-time: Configure
- One-time: Import management command
- Run management command for topic
- Write signal receiver
  - Work your magic

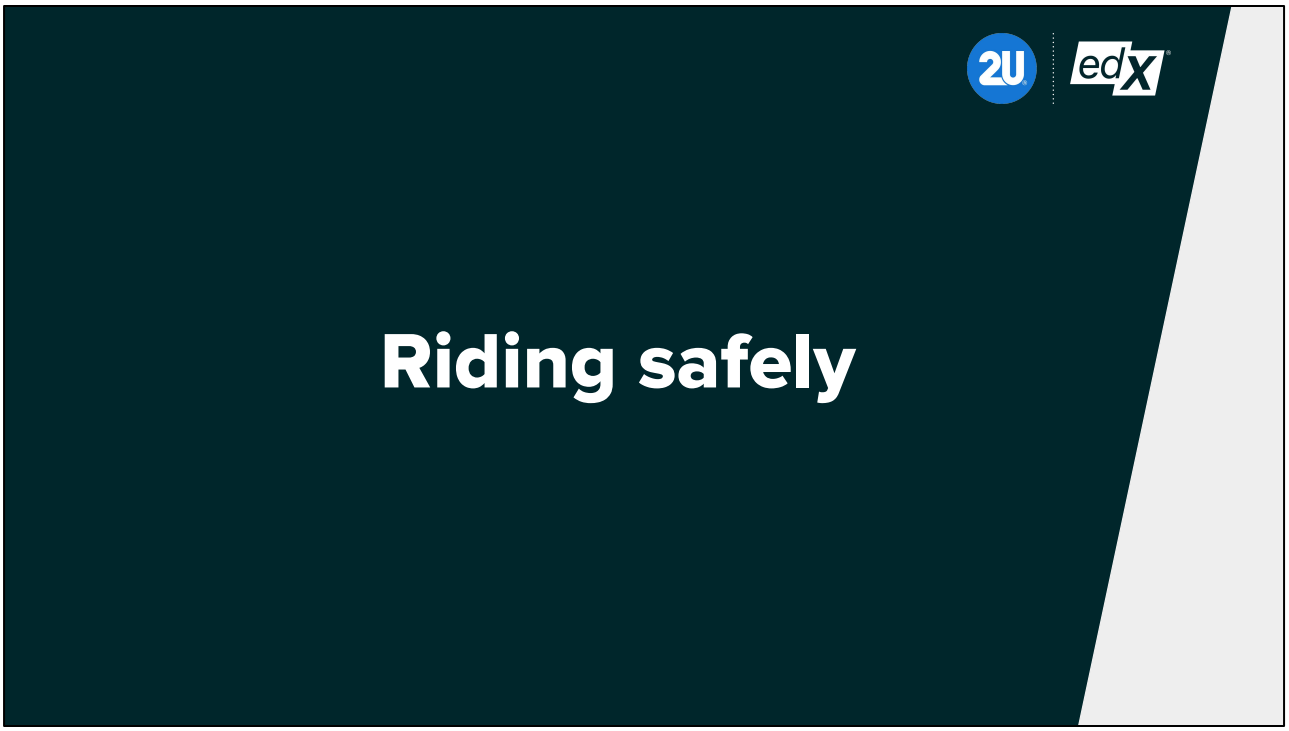
- Infinite loop

Consuming an event

```
./manage.py consume_events -t my-event-happened -g my_group -s my.event.type
```

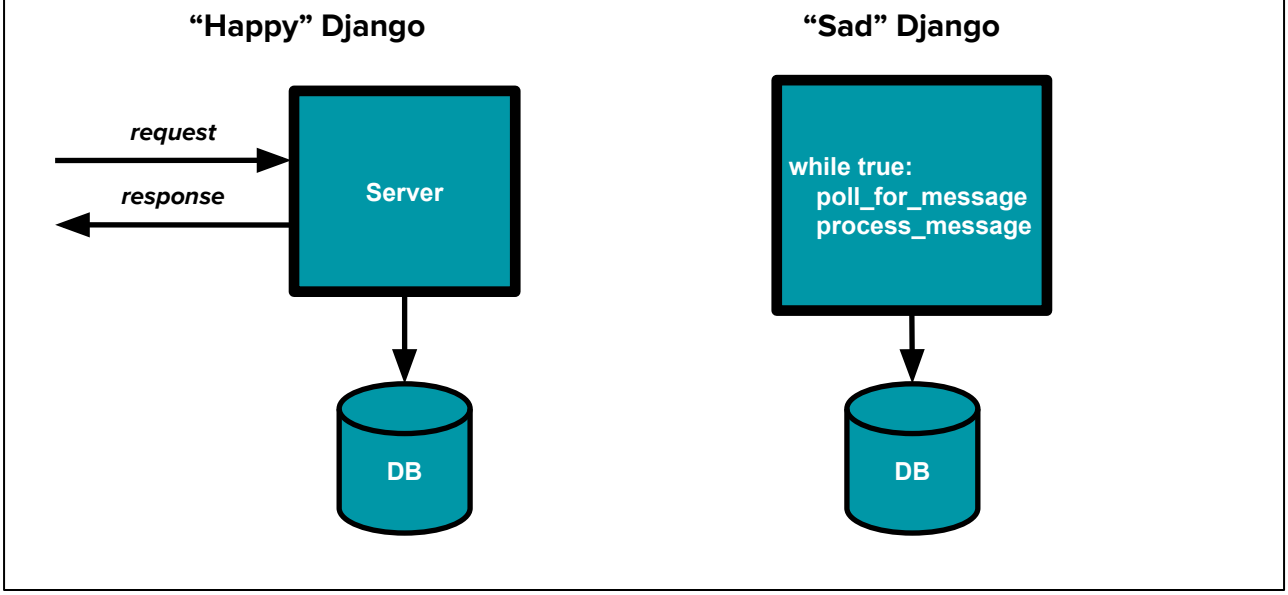
```
@receiver(MY_EVENT_HAPPENED)
def listen_for_my_signal_and_do_things(sender, **kwargs):
  ... do things with the data in kwargs
```

-



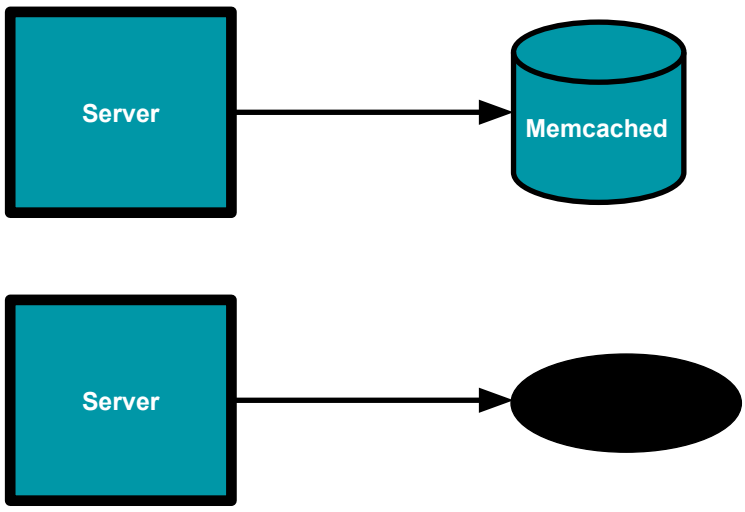
- Or, where are we at now.

Riding safely



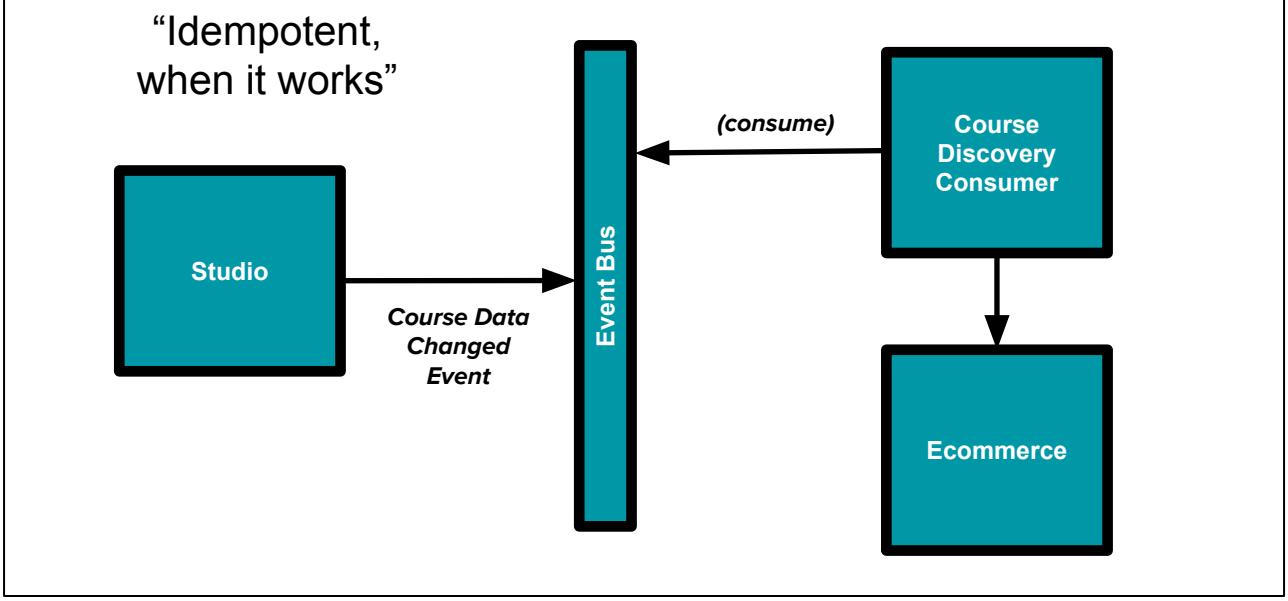
- Django isn’t actually sad, it just isn’t a pattern that Django was designed around
- Infinite loop
- DB fixes
    - Consecutive error fix (generic fix)
    - Connection reset fix
- Be careful: e.g. Atomic Requests

Riding safely



-

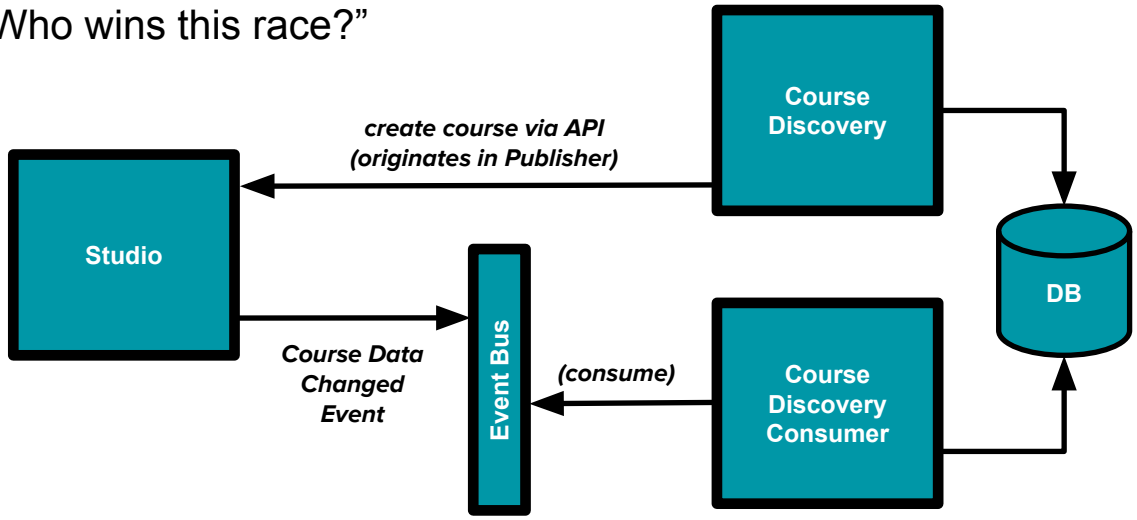
Riding safely



-

Riding safely

"Who wins this race?"



- Bi-directional data flows
- Race conditions
- Fixes
  - Hacked delay
  - Uni-directional
    - Don't allow editing from Publisher, or
    - Have publisher direct create in Studio and pick up changes.

Riding safely

- Infinite loop vs requests
- Memcached connections
- Idempotence
- Race conditions and bi-directional data flow
- More to come…
    - Schema evolution
    - Lifecycle events

- Race conditions and bi-directional data flows
- Some of these safety nets are being built in to the system. Some need to be kept in mind.

Next stops

Next stops

- Redis implementation
- OEP-52: Event Bus Architecture => Accepted
- More events
  - `XBLOCK_DELETED`, `XBLOCK_DUPLICATED`, `XBLOCK_PUBLISHED`
  - Credential awarded
  - Grades (e.g. `PERSISTENT_GRADE_SUMMARY_CHANGED`)
  - Enrollments (e.g. `COURSE_ENROLLMENT_CREATED`, `COURSE_ENROLLMENT_CHANGED`, `COURSE_UNENROLLMENT_COMPLETED`)
  - <insert your event here>

-

Next stops

- *So much more to learn*
  - See Platform Event Bus roadmap issue
  - *#event-bus* in Open edX Slack

-

Recap

- Why an event bus?
- Current stop
- Boarding the bus
- Riding safely
- Next stops
- More questions?

More resources

- How to start using the Event Bus
- Platform Event Bus roadmap
- OEP-52: Event Bus Architecture
- OEP-41: Asynchronous Server Event Message Format
- OEP-50: Hooks extension framework
- Architecture Manifesto (WIP)
- https://github.com/openedx/event-bus-kafka/
- https://github.com/openedx/openedx-events

-

More Questions