Ralph

The ultimate toolbox for your learning analytics.

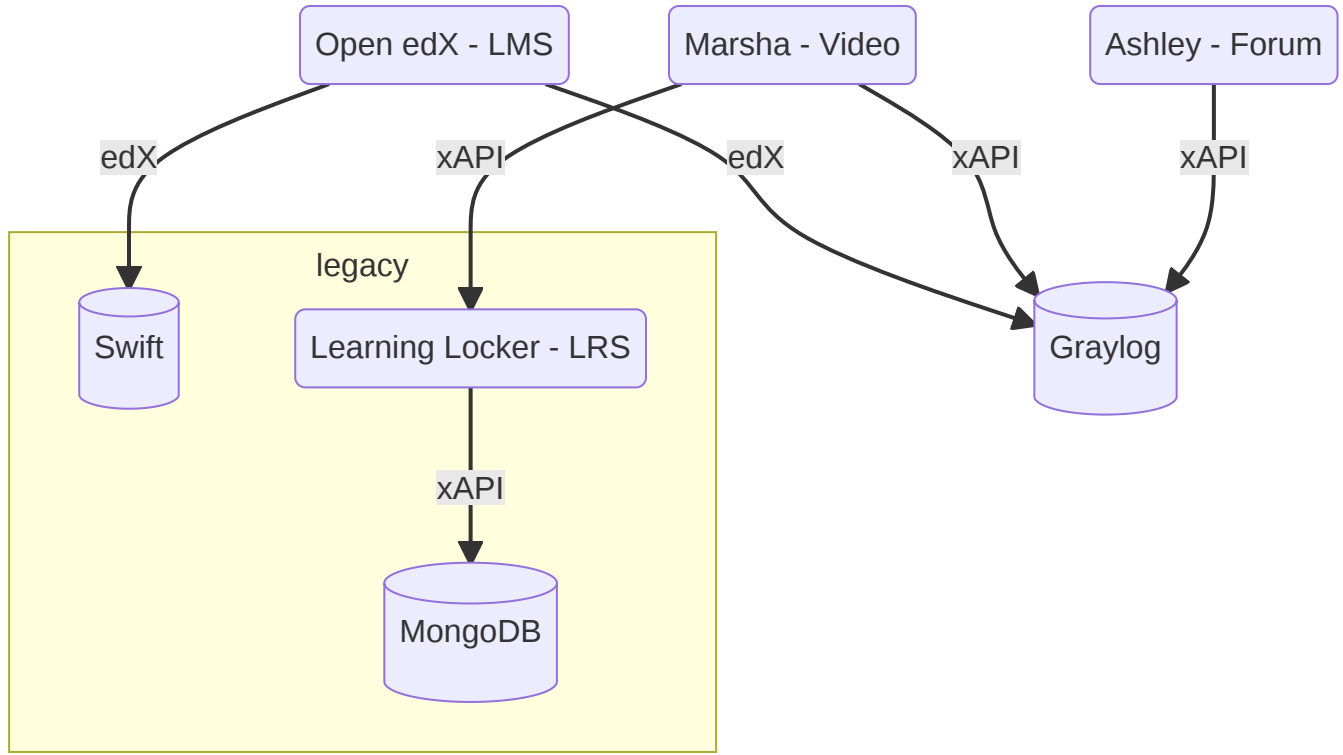
// @julienmaupetit // @open_fun //

Outline

1. Context
2. Ralph is a CLI
3. Ralph is an ETL
4. Ralph is a LRS
5. Ralph is a library
6. What's next?

Context

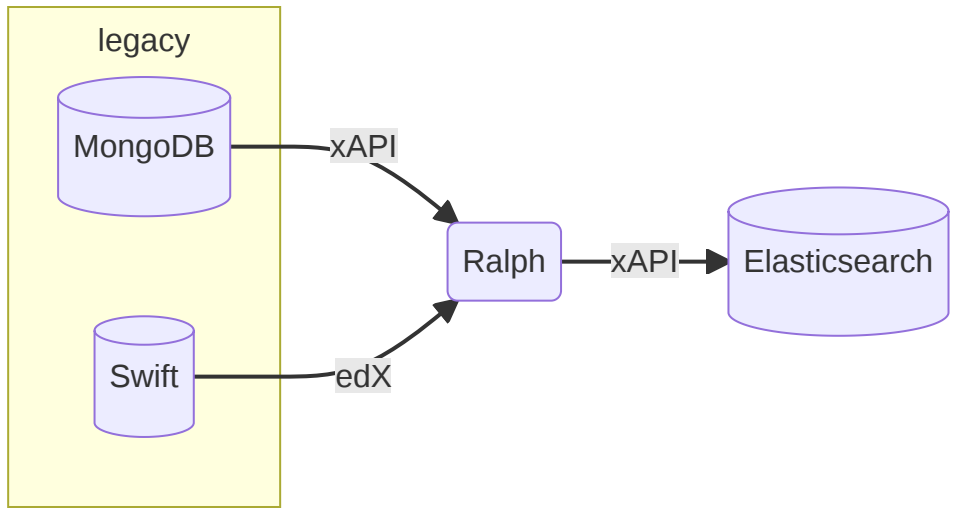
How it started



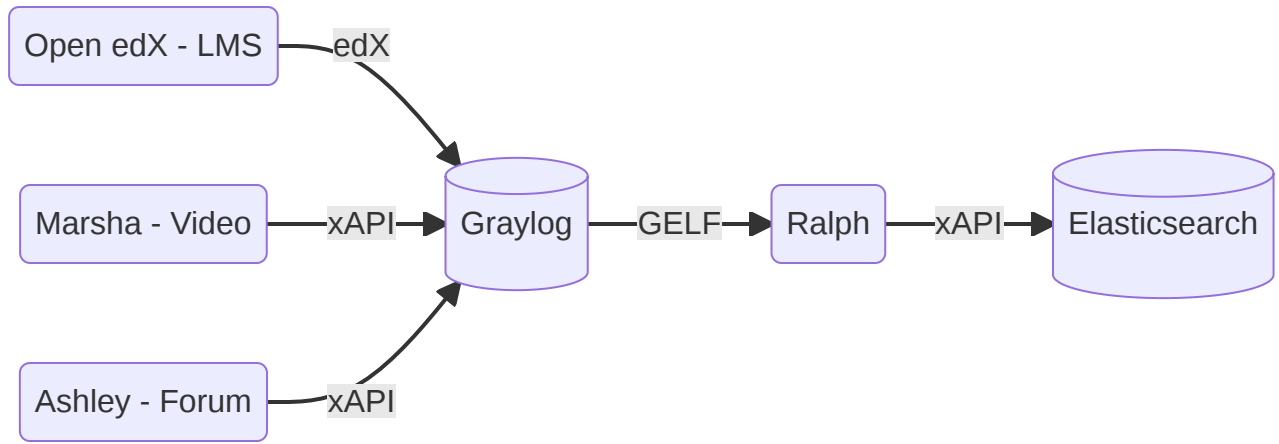
The plan

- Setup an Elasticsearch-based **data lake** dedicated to learning analytics
- Develop a Python tool able to **fetch** & **transform** data from various backends

How it’s going (1)



How it’s going (2)



Ralph is a CLI

Installation

Ralph is distributed as a python package:

```
# Create a new virtualenv (optional)
$ python -m venv venv
$ source venv/bin/activate

# Install the package (in a virtualenv)
(venv) $ pip install ralph-malph
```

... and a Docker image:

```
$ docker run --rm -i fundocker/ralph:latest \
    ralph --help
```

Usage

As `git` or `docker`, `ralph` implements (sub)commands:

```
Usage: ralph [OPTIONS] COMMAND [ARGS]...

  Ralph is a stream-based tool to play with your logs.

Options:
  -v, --verbosity LVL  Either CRITICAL, ERROR, WARNING, INFO (default) or
                       DEBUG
  --help               Show this message and exit.

Commands:
  convert    Converts input events to a given format.
  extract    Extracts input events from a container format using a...
  fetch      Fetch an archive or records from a configured backend.
  list       List available archives from a configured storage backend.
  push       Push an archive to a configured backend.
  runserver  Runs the API server for the development environment.
  validate   Validates input events of given format.
```

Example workflows

Push a local archive to MongoDB:

```
zcat 20220923-lms-statements.jsonl.gz | \
  ralph push --backend mongo
```

Import new data from a LDP[1] stream:

```
ralph list --backend ldp --new |
  xargs -I {} -n 1 bash -c "
    ralph fetch --backend ldp {} |
    gunzip |
    ralph extract --parser gelf |
    ralph push --backend es"
```

1. Log Data Platform (OVH's Graylog) ↩

Quickly explore your data

Get the top-100 accounts that have generated the most events:

```
ralph fetch --backend swift 20220923.xapi.gz | \
    jq .actor.account.name | \
    sort | \
    uniq -c | \
    sort -rn | \
    head -n 100
```

Supported backends

- Database
  - Elasticsearch
  - MongoDB
- Storage
  - Swift
  - LDP (Log Data Platform)
  - AWS S3[1]
- Stream
  - WebSocket

1. Pull request in review ↩

Ralph is an ETL

The `convert` (sub)command

```
Usage: ralph convert [OPTIONS]

  Converts input events to a given format.

Options:
  From edX to xAPI converter options:
    -u, --uuid-namespace TEXT     The UUID namespace to use for the `ID` field
                                  generation
    -p, --platform-url TEXT       The `actor.account.homePage` to use in the
                                  xAPI statements  [required]
  -f, --from [edx]                Input events format to convert  [required]
  -t, --to [xapi]                 Output events format  [required]
  -I, --ignore-errors             Continue writing regardless of raised errors
  -F, --fail-on-unknown           Stop converting at first unknown event
  --help                          Show this message and exit.
```

ETL: Extract Transform Load

Fetch Open edX events wrapped in a zipped GELF archive, convert it to xAPI and feed the elasticsearch data lake with it.

```
ralph fetch --backend swift | \
    gunzip | \
    ralph extract --parser gelf | \
    ralph convert \
      --from edx \
      --to xapi \
      --platform-url "https://www.fun-mooc.fr" | \
    ralph push --backend es
```

Pydantic models

```python
# ralph/models/edx/navigational/statements.py
class UIPageClose(BaseBrowserModel):
    """Represents edx `page_close` browser statement."""
    __selector__ = selector(
        event_source="browser",
        event_type="page_close"
    )

    event: Literal["{}"]
    event_type: Literal["page_close"]
    name: Literal["page_close"]

# ralph/models/xapi/navigation/statements.py
class PageTerminated(BaseXapiModel):
    """Represents a page terminated xAPI statement."""
    __selector__ = selector(
        object__definition__type="http://activitystrea.ms/schem
        verb__id="http://adlnet.gov/expapi/verbs/terminated",
    )

    object: PageObjectField
    verb: TerminatedVerbField = TerminatedVerbField()
```

Converter

```python
# ralph/models/edx/converters/xapi/navigational.py
class UIPageCloseToPageTerminated(BaseXapiConverter):
    """Converts a common edX `page_close` event to xAPI.

    Example Statement:

    John terminated https://www.fun-mooc.fr/ page.
    """

    __src__ = UIPageClose
    __dest__ = PageTerminated

    def _get_conversion_items(self):
        """Returns a set of ConversionItems used for
        conversion.
        """

        conversion_items = super()._get_conversion_items()
        return conversion_items.union(
            {
                ConversionItem("object__id", "page")
            }
        )
```

Ralph is a LRS

Learning Record Store (LRS)

> The LRS, as defined by the xAPI specification, is “a server (i.e. system capable of receiving and processing web requests) that is responsible for receiving, storing, and providing access to Learning Records.”

🎉 Ralph bundles a FastAPI-based lightweight LRS that handles the `/xAPI/statements` endpoint with `POST` and `GET` requests to receive and return xAPI statements.

🎉 Elasticsearch and MongoDB backends are supported.

🎉 Implements statements forwarding for complex workflows.

Let’s play!

Run the development server for testing:

```
ralph runserver --backend es
```

The LRS server should be up and running on the 8100 port, ready to receive xAPI statements:

```
$ zcat 20220923.xapi.gz | \
    jq -s . | \
    http \
      --json \
      --auth julien:password \
      POST \
      http://localhost:8100/xAPI/statements/

HTTP/1.1 200 OK
content-length: 1951
content-type: application/json
date: Sun, 25 Sep 2022 20:43:56 GMT
server: uvicorn
```

Ralph is a library

A CLI is cool, but write Python code is even cooler!

- Ease developers onboarding
- Maintainability
- Jupyter notebooks

Use xAPI models library

```
// Front-end event payload
{
  "actor": {
    "objectType": "Agent",
    "account": {
      "name": "John Doe",
      "homePage": "http://fun-mooc.fr"
    }
  },
  "verb": {
    "id": "https://w3id.org/xapi/video/verbs/played"
  },
  "object": {
    "definition": {
      "type": "https://w3id.org/xapi/video/activity-type/video'
    },
    "id": "uuid://4484f345-6711-5aea-a0d6-42a5004c879f",
    "objectType": "Activity"
  }
  /* [...] */
}
```

```
# Backend
import uuid
import requests

from ralph.models.xapi.video.statements import VideoPlayed
from . import app

@app.post("/events/", status_code=201)
def post(event: str) -> uuid.UUID:
    """Validate front-end event and add identifier field
    if missing.
    """

    # Parse and validate event JSON string
    statement = VideoPlayed.parse_raw(event)
    statement.id = uuid.uuid4() if statement.id is None

    # Send event to an LRS
    response = requests.post(
        "https://lrs.org/xAPI/statements/",
        data=statement.dict()
    )

    return response.json()
```

Use builtin converters

```
from ralph.models.converter import ConversionItem
from ralph.models.edx.server import Server
from ralph.models.xapi.navigation.statements import PageViewed

from .base import BaseXapiConverter

class ServerEventToPageViewed(BaseXapiConverter):
    """Converts a common edX server event to xAPI.
    Example Statement: John viewed https://www.fun-mooc.fr/ page.
    """
    __src__ = Server
    __dest__ = PageViewed

    def _get_conversion_items(self):
        """Returns a set of ConversionItems used for conversion."""
        conversion_items = super()._get_conversion_items()
        return conversion_items.union(
            {
                ConversionItem(
                    "object__id",
                    "event_type",
                    lambda event_type: self.platform_url + event_type,
                ),
            }
        )
```

Generate test fixtures

Use Hypothesis with Pydantic models to generate fixtures for your tests:

```
from ralph.models.xapi.navigation.statements import PageTerminated
from tests.fixtures.hypothesis_strategies import custom_given


@custom_given(PageTerminated)
def test_models_xapi_page_terminated_statement(statement):
    """Tests that a page_terminated statement has the expected verb.id and
    object.definition.
    """

    assert statement.verb.id == "http://adlnet.gov/expapi/verbs/terminated"
    assert statement.object.definition.type == "http://activitystrea.ms/schema/1.0/page"
```

Multiple backends, same API.

```python
# ralph.backends.database.base module
from abc import ABC, abstractmethod

class BaseDatabase(ABC):
    """Base database backend interface."""
    name = "base"
    query_model = BaseQuery

    @abstractmethod
    @enforce_query_checks
    def get(
        self,
        query: BaseQuery = None,
        chunk_size: int = 10
    ):
        pass

    @abstractmethod
    def put(
        self,
        stream: Union[BinaryIO, TextIO],
        chunk_size: int = 10,
        ignore_errors: bool = False,
    ) -> int:
        pass
```

An ongoing effort. 😎

```python
    @abstractmethod
    def query_statements(
        self,
        params: StatementParameters
    ) -> StatementQueryResult:
        """Returns the statements query payload using xAPI
        parameters.
        """

    @abstractmethod
    def query_statements_by_ids(
        self,
        ids: list[str]
    ) -> list:
        """Returns the list of matching statement IDs from
        the database.
        """
```

What’s next?

Ralph v3.0.0

- Production-ready minimalist LRS server
- MongoDB backend

Coming in 2023.1

- LRS server generic backend support
- Improved documentation (tutorial)
- Improve library support (dependencies, APIs)
- Helm chart

Check this out!

https://github.com/openfun/ralph


Sergio Simonian
😻 Data team 😻
Quitterie Lucas