Fundamentals of Data Engineering

Week 05 - sync session

datascience@berkeley

While we're getting started

- Review your Assignment 04
- Get ready to share

Due Friday (PR)

Where are we?

Today

- Assignment 04
- Run standalone kafka cluster
- NoSQL stores with docker compose

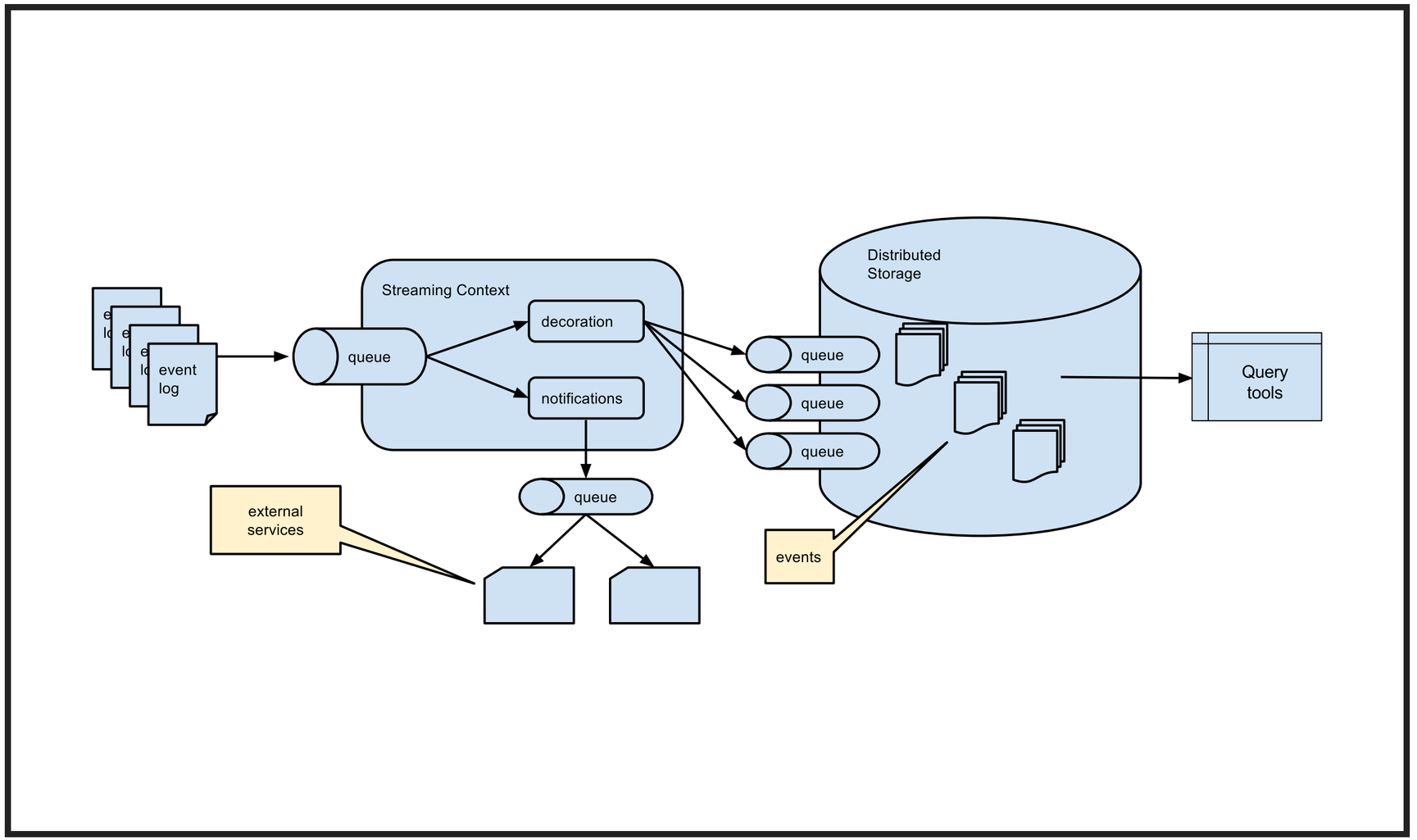
Between Class 5 & Class 6

- async material in Week 5 syllabus (Virtualization, Hadoop)
- Readings in Week 5 syllabus
- Assignment 05 (notebook to present)
- Final Assignment 04 due on Friday

Class 6

- Final presentations of Query Project Notebooks/Recommendations
- Tracking User Activity Project (spans Assignments 6-8)

Where are we in the pipeline?


Distributed
Storage
Streaming Context
decoration
queue
notifications
event
log
queue
queue
queue
queue
Query
tools
external
services
events

Docker-compose with Redis

Setup

Create a workspace for this example

```
mkdir ~/w205/redis
cd ~/w205/redis
```

Save the following to `docker-compose.yml` in that directory

```
---
version: '2'
services:
  redis:
    image: redis:latest
    expose:
      - "6379"
    extra_hosts:
      - "moby:127.0.0.1"

  mids:
    image: midsw205/base:latest
    stdin_open: true
    tty: true
    extra_hosts:
      - "moby:127.0.0.1"
```

Spinup

Start up the cluster

```
docker-compose up -d
```

Check stuff

```
docker-compose ps
```

Should see

```
          Name                        Command                State     |
-------------------------------------------------------------------------
redisexample_midsbase_1   /bin/bash                           Up       8
redisexample_redis_1      docker-entrypoint.sh redis ...      Up       6
```

Peek at the logs

```
docker-compose logs redis
```

Should see log output ending in

```
Ready to accept connections
```

Run stuff

Connect to the mids container

```
docker-compose exec mids bash
```

At the prompt, run

```
ipython
```

Try out redis

```
import redis
r = redis.Redis(host='redis', port='6379')
r.keys()
exit
```

Exit that container

```
exit
```

Tear down your stack

```
docker-compose down
```

Verify

```
docker-compose ps
```

Jupyter Notebooks

Change the `docker-compose.yml` file

```
---
version: '2'
services:
  redis:
    image: redis:latest
    expose:
      - "6379"
    extra_hosts:
      - "moby:127.0.0.1"

  mids:
    image: midsw205/base:latest
    stdin_open: true
    tty: true
    expose:
      - "8888"
```

Save that and bring it up

```
docker-compose up -d
```

Start up a notebook

```
docker-compose exec mids jupyter notebook --no-browser --port 8888 -
```

Copy token… should look something like

```
open http://0.0.0.0:8888/?token=<your token>
```

Open a browser

```
http://0.0.0.0:8888
```

Paste token

Drop the cluster when you're done

```
docker-compose down
```

Automate notebook startup

Just for fun,

```
---
version: '2'
services:
  redis:
    image: redis:latest
    expose:
      - "6379"
    extra_hosts:
      - "moby:127.0.0.1"

  mids:
    image: midsw205/base:latest
    stdin_open: true
    tty: true
    expose:
      - "8888"
```

Test it out

```
docker-compose up -d
```

Run to get the token

```
docker-compose logs mids
```

Open a browser

```
open http://0.0.0.0:8888/?token=<your token>
```

Open New Python3 Notebook

Try redis

```
import redis
r = redis.Redis(host='redis', port='6379')
r.keys()
```

Add some values

```
r.set('foo', 'bar')
value = r.get('foo')
print(value)
```

Drop cluster

```
docker-compose down
```

Redis to track state

```
~/w205/redis

curl -L -o trips.csv https://goo.gl/MVNVhW
```

Spin up cluster

```
docker-compose up -d
```

Run to get the token

```
docker-compose logs mids
```

Open a browser

```
open http://0.0.0.0:8888/?token=<your token>
```

Open New Python3 Notebook

```
import redis
import pandas as pd
```

```
trips=pd.read_csv('trips.csv')

date_sorted_trips = trips.sort_values(by='end_date')

date_sorted_trips.head()
```

```
for trip in date_sorted_trips.itertuples():
  print(trip.end_date, '', trip.bike_number, '', trip.end_station_na
```

```
current_bike_locations = redis.Redis(host='redis', port='6379')
current_bike_locations.keys()
```

Add values

```
for trip in date_sorted_trips.itertuples():
  current_bike_locations.set(trip.bike_number, trip.end_station_name
```

```
current_bike_locations.keys()
```

Where is bike 92?

```
current_bike_locations.get('92')
```

Drop cluster

```
docker-compose down
```

Summary

Extras

Athena & AWS cli tool (aws)

Berkeley SCHOOL OF INFORMATION