

Fundamentals of Data Engineering

Week 05 - sync session

datascience@berkeley

While we're getting started

- Review your Assignment 04
- Get ready to share

Where are we?

Today

- Assignment 04 wraps up the Query Project
- Creating basic `docker-compose` clusters
- NoSQL stores with `docker-compose`

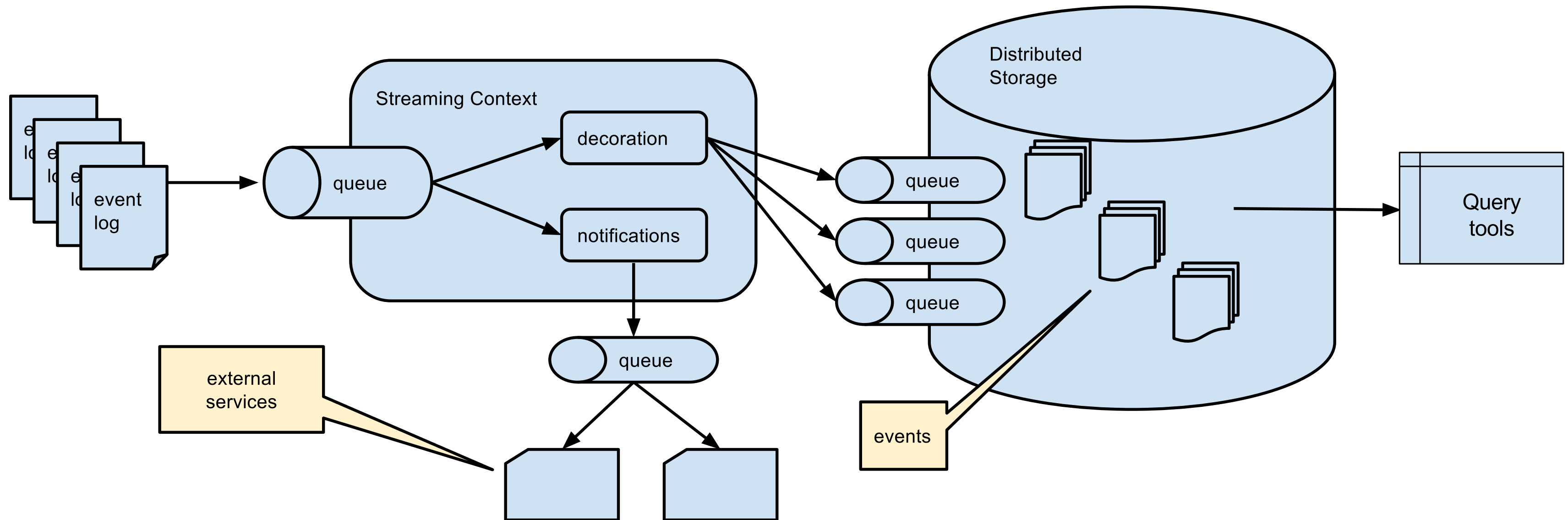
Between Class 5 & Class 6

- async material in Week 5 syllabus (Virtualization, Hadoop)
- Readings in Week 5 syllabus
- Assignment 05

Class 6

- Start “Tracking User Activity” Project (spans Assignments 6-8)

Where are we in the pipeline?



Docker

Docker

Let's play with Redis

spin it up

```
docker run redis
```

```
docker run -d redis
```

```
docker run -d --name redis redis
```

```
docker run -d --name redis -p 6379:6379 redis
```

Docker compose

- helps manage options easily
- manage multiple containers at once

Update your course content repo in w205

```
cd ~/w205/course-content  
git pull --all
```

Docker compose .yaml file

```
mkdir ~/w205/redis-standalone  
cd ~/w205/redis-standalone  
cp ../course-content/05-Storing-Data-II/example-0-docker-compose.yaml
```

i.e.,

```
---  
version: '2'  
services:  
  redis:  
    image: redis  
    expose:  
      - "6379"  
    ports:  
      - "6379:6379"
```

Spinup

Start up the cluster

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

Check stuff

```
docker-compose ps
```

Peek at the logs

```
docker-compose logs redis
```

Should see log output ending in

```
Ready to accept connections
```

Run stuff

```
ipython
```


Try out redis

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

Tear down your stack

```
docker-compose down
```

Verify

```
docker-compose ps
```

More docker-compose with Redis

Setup

Create a workspace for this other example

```
mkdir ~/w205/redis-cluster  
cd ~/w205/redis-cluster  
cp ../course-content/05-Storing-Data-II/example-1-docker-compose.yml
```

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

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

  midsw205:
    image: midsw205/base:latest
    stdin_open: true
    tty: true
```

Spinup

Start up the cluster

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


Check stuff

```
docker-compose ps
```

Should see

| Name | Command | State | IP |
|-------------------------|--------------------------------|-------|----|
| redisexample_midsbase_1 | /bin/bash | Up | 88 |
| redisexample_redis_1 | docker-entrypoint.sh redis ... | Up | 63 |

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"

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

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"

  mids:
    image: midsw205/base:latest
    stdin_open: true
    tty: true
    expose:
      - "8888"
    ports:
      - "8888:8888"
    command: jupyter notebook --no-browser --port 8888 --ip 0.0.0.0 -
```

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
```

Extras

Redis to track state

Setup

Download data:

```
cd ~/w205/  
curl -L -o trips.csv https://goo.gl/QvHLKe
```

Setup

Add volumes to your `docker-compose.yml`:

```
volumes:  
  - ~/w205:/w205
```

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

  midsw205:
    image: midsw205/base:latest
    stdin_open: true
    tty: true
    volumes:
      - ~/w205:/w205
    expose:
      - "8888"
    ports:
```


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_name)
```

```
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()
```

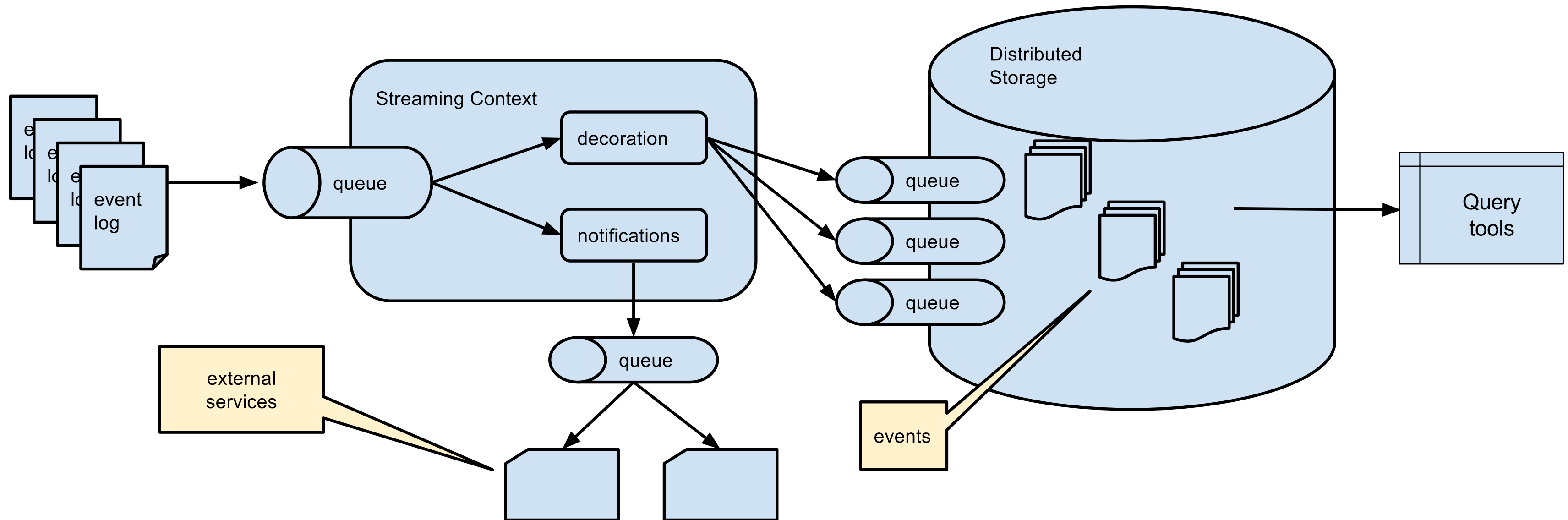
Where is bike 92?

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

Drop cluster

```
docker-compose down
```

Summary



Berkeley

SCHOOL OF
INFORMATION