Fundamentals of Data Engineering

Week 06 - sync session

datascience@berkeley

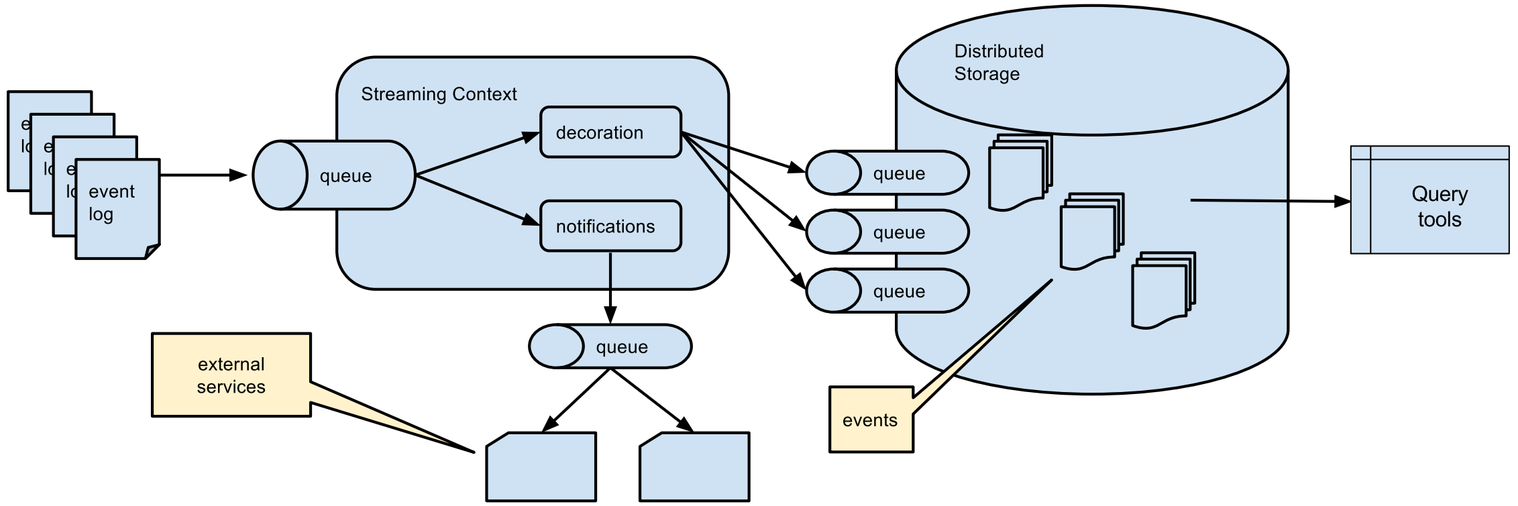
While we're getting started

- Review your Project 1
- Get ready to share

Pipes

```
cat junk.csv | sort | uniq | wc -l
```

Where are we in the pipeline

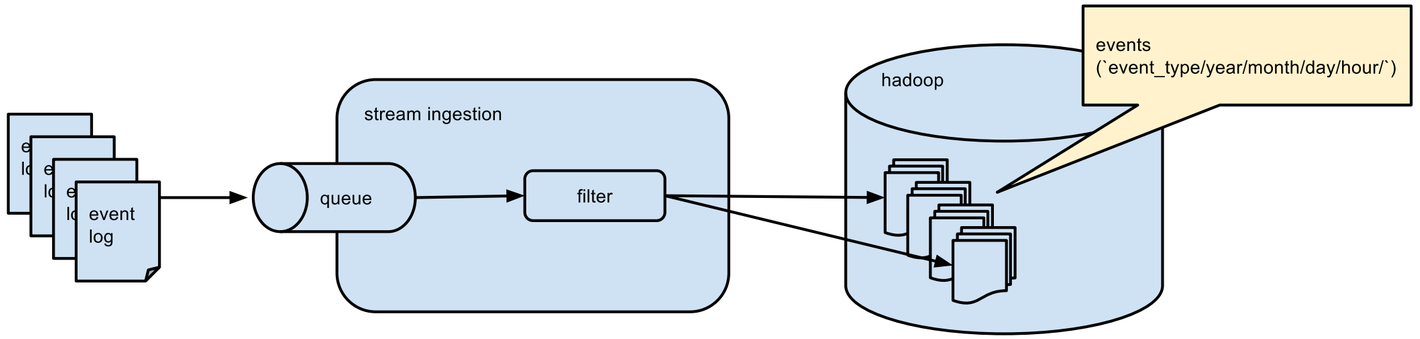

event
log
queue
Streaming Context
decoration
notifications
queue
external
services
queue
queue
queue
Distributed
Storage
events
Query
tools

Starting into Project 2: Tracking User Activity

- In this project, you work at an ed tech firm.
- You've created a service that delivers assessments.
- Now lots of different customers (e.g., Pearson) want to publish their assessments on it.
- You need to get ready for data scientists who work for these customers to run queries on the data.

Project 2 actitivities

- Through 3 different activites, you will spin up existing containers and prepare the infrastructure to land the data in the form and structure it needs to be to be queried.
  1. Publish and consume messages with kafka.
  2. Use spark to transform the messages.
  3. Use spark to transform the messages so that you can land them in hdfs.


event
log
queue
stream ingestion
filter
hadoop
events
(`event_type/year/month/day/hour/`)

Kafka

Update your course content repo in w205

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

Docker compose .yml file

- `cd w205`
- `mkdir kafka`
- save `docker-compose.yml` from recently pulled `~/w205/course-content` to recently created `~/w205/kafka` directory

```
---
version: '2'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    environment:
      ZOOKEEPER_CLIENT_PORT: 32181
      ZOOKEEPER_TICK_TIME: 2000
    expose:
      - "2181"
      - "2888"
      - "32181"
      - "3888"

  kafka:
    image: confluentinc/cp-kafka:latest
    depends_on:
```

Docker compose spin things up

- `cd ~/w205/kafka`
- `docker-compose up -d`
- `docker-compose ps`

Can check with

- `docker-compose ps`

Should see something like

which should show something like

```
Name                              Command               State    Ports
-------------------------------------------------------------------------
kafkasinglenode_kafka_1         /etc/confluent/docker/run   Up
kafkasinglenode_zookeeper_1     /etc/confluent/docker/run   Up
```

Check zookeeper

```
docker-compose logs zookeeper | grep -i binding
```

Should see something like

```
zookeeper_1   | [2016-07-25 03:26:04,018] INFO binding to port 0.0.0.0
```

Check the kafka broker

```
docker-compose logs kafka | grep -i started
```

Should see something like

```
kafka_1      | [2017-08-31 00:31:40,244] INFO [Socket Server on Broke
kafka_1      | [2017-08-31 00:31:40,426] INFO [Replica state machine
kafka_1      | [2017-08-31 00:31:40,436] INFO [Partition state machir
kafka_1      | [2017-08-31 00:31:40,540] INFO [Kafka Server 1], start
```

Create a Topic `foo`

```
docker-compose exec kafka \
  kafka-topics \
    --create \
    --topic foo \
    --partitions 1 \
    --replication-factor 1 \
    --if-not-exists \
    --zookeeper zookeeper:32181
```

Should see something like

```
Created topic "foo".
```

Check the topic

```
docker-compose exec kafka \
  kafka-topics \
    --describe \
    --topic foo \
    --zookeeper zookeeper:32181
```

Should see something like

```
Topic:foo   PartitionCount:1    ReplicationFactor:1 Configs:
Topic: foo  Partition: 0    Leader: 1    Replicas: 1  Isr: 1
```

Publish Messages

```
docker-compose exec kafka \
  bash -c "seq 42 | kafka-console-producer \
    --request-required-acks 1 \
    --broker-list localhost:29092 \
    --topic foo && echo 'Produced 42 messages.'"
```

Should see something like

```
Produced 42 messages.
```

Consume Messages

```
docker-compose exec kafka \
  kafka-console-consumer \
    --bootstrap-server localhost:29092 \
    --topic foo \
    --from-beginning \
    --max-messages 42
```

Should see something like

```
1
....
42
Processed a total of 42 messages
```

Tearing things down

```
docker-compose down
```

Kafka with “real” messages

- We’ll deal with json for the project

Kafka with json example

- To address json, we'll need kafkacat

kafkacat

docker-compose.yml file

```
---
version: '2'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    environment:
      ZOOKEEPER_CLIENT_PORT: 32181
      ZOOKEEPER_TICK_TIME: 2000
    expose:
      - "2181"
      - "2888"
      - "32181"
      - "3888"

  kafka:
    image: confluentinc/cp-kafka:latest
    depends_on:
```

Pull data

```
cd ~/w205/kafka/
curl -L -o github-example-large.json https://goo.gl/Y4MD58
```

Spin up the cluster

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

Watch it come up

```
docker-compose logs -f kafka
```

- Detach with `Ctrl-C`

use it

create a topic

```
docker-compose exec kafka \
kafka-topics \
  --create \
  --topic foo \
  --partitions 1 \
  --replication-factor 1 \
  --if-not-exists \
  --zookeeper zookeeper:32181
```

Should see something like

```
Created topic "foo".
```

Check the topic

```
docker-compose exec kafka \
  kafka-topics \
    --describe \
    --topic foo \
    --zookeeper zookeeper:32181
```

Should see something like

```
Topic:foo   PartitionCount:1    ReplicationFactor:1 Configs:
Topic: foo  Partition: 0    Leader: 1    Replicas: 1  Isr: 1
```

Publish some stuff to kafka

Check out our messages

```
docker-compose exec mids bash -c "cat /w205/kafka/github-example-larg
docker-compose exec mids bash -c "cat /w205/kafka/github-example-larg
docker-compose exec mids bash -c "cat /w205/kafka/github-example-larg
```

Publish some test messages

```
docker-compose exec mids bash -c "cat /w205/kafka/github-example-larg
```

Should see something like

```
Produced 100 messages.
```

Consume the messages

We can either do what we did before

```
docker-compose exec kafka \
  kafka-console-consumer \
    --bootstrap-server kafka:29092 \
    --topic foo \
    --from-beginning \
    --max-messages 42
```

or

```
docker-compose exec mids bash -c "kafkacat -C -b kafka:29092 -t foo -
```

and maybe

```
docker-compose exec mids bash -c "kafkacat -C -b kafka:29092 -t foo -
```

Down

```
docker-compose down
```

Project 2

- Step through this process using the Project 2 data
- What you turn in:
- In your `/project-2-<user-name>` repo:
  - your `docker-compose.yml`
  - once you've run the example on your terminal
    - Run `history > <user-name>-history.txt`
    - Save the relevant portion of your history as `<user-name>-annotations.md`
    - Annotate the file with explanations of what you were doing at each point (See `htmartin-annotations.md`)

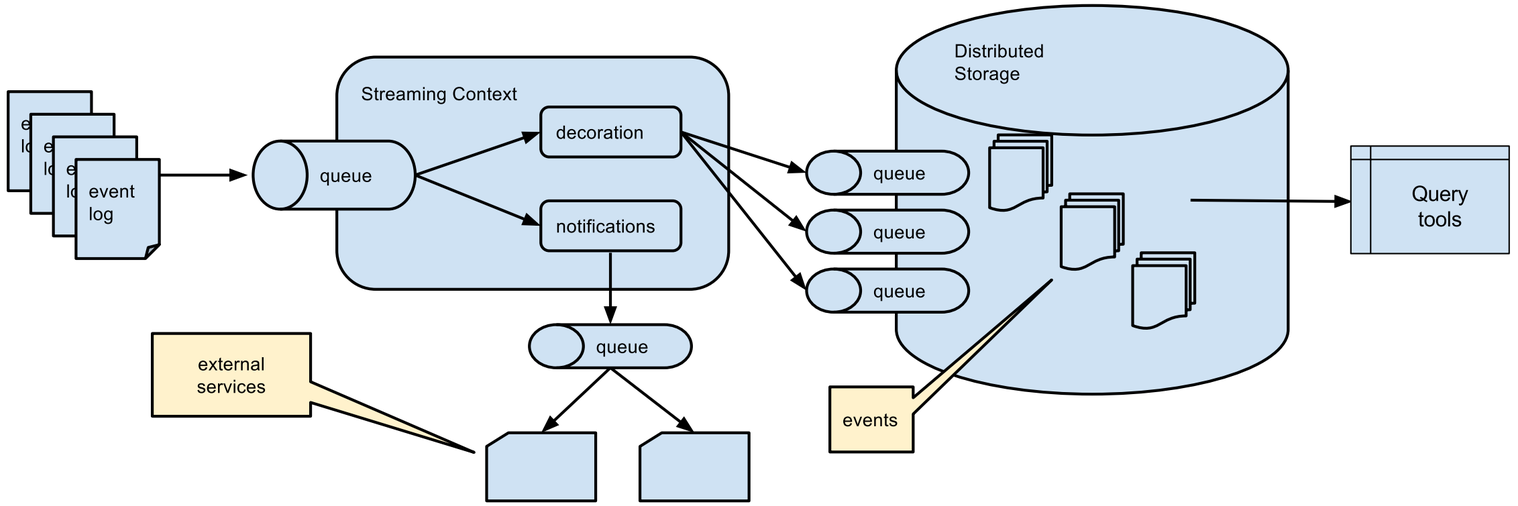
Week 6 Videos

- Discussion of traditional ETL (get it out of the way).
- contrast with Hadoop ETL.
- How Spark handles compute.
- Distribution of jobs across clusters.
- Schedulers
- Give you a context for how what we're doing in Project 2 would work in reality.

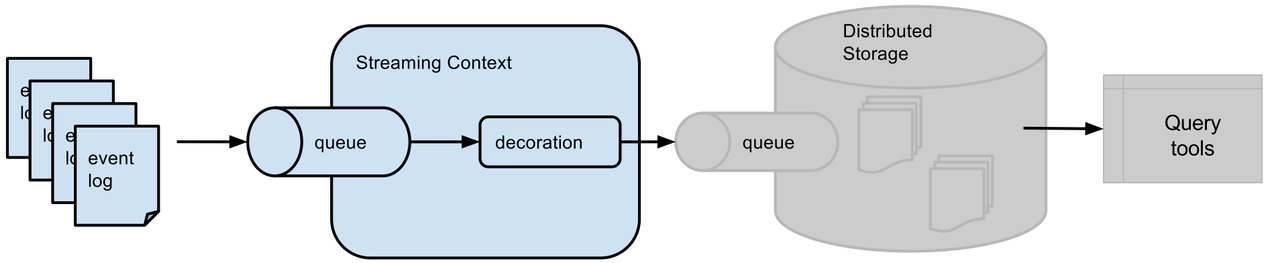
Summary

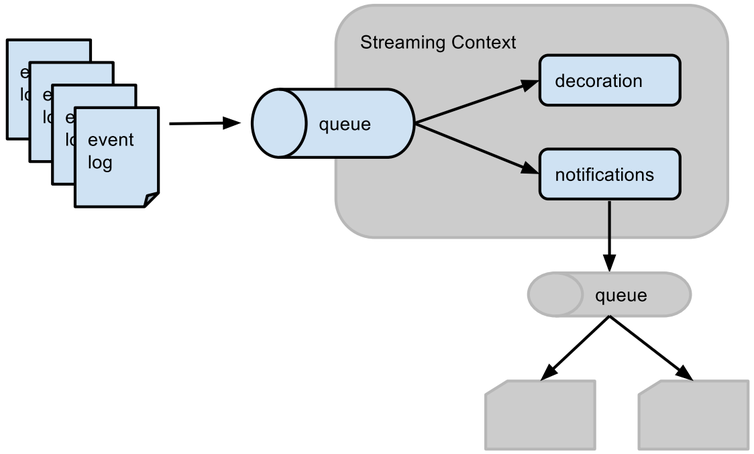
- Test that we can spin up containers, publish & consume messages with simple numbers messages.
- Work through some actual data from github
- Prep for Project 2

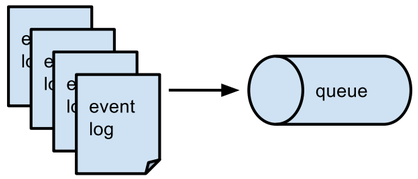
Berkeley SCHOOL OF INFORMATION

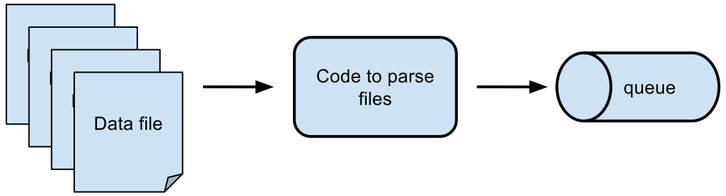

event
log
queue
Streaming Context
decoration
notifications
queue
external
services
queue
queue
queue
Distributed
Storage
events
Query
tools

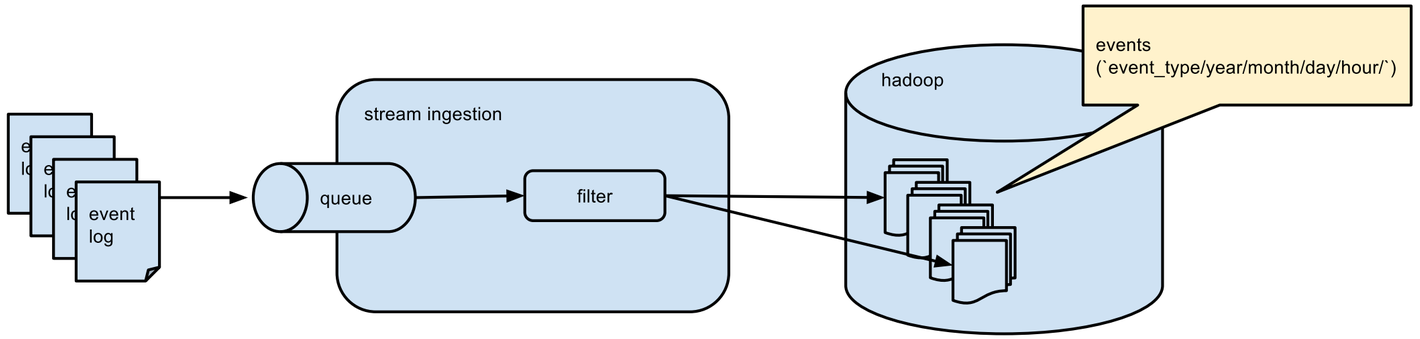
cool pics


event
log
Streaming Context
queue
decoration
queue
Distributed
Storage
Query
tools


Streaming Context
event log
queue
decoration
notifications
queue


event
log
queue


Data file
Code to parse files
queue


event
log
queue
stream ingestion
filter
hadoop
events
(`event_type/year/month/day/hour/`)