# Fundamentals of Data Engineering

Week 07 - sync session

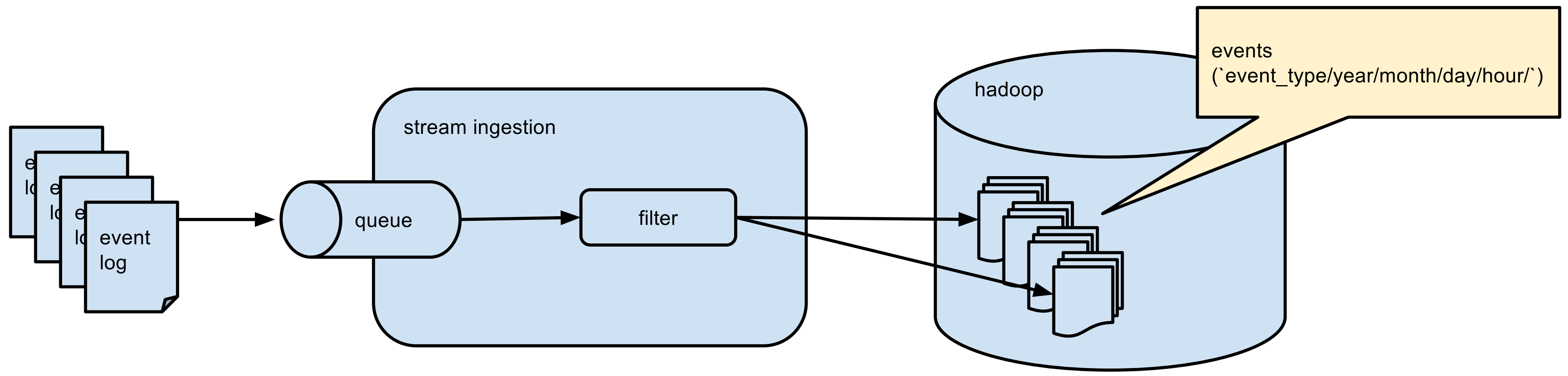
datascience@berkeley

## While we're getting started

- Review your Assignment 06
- Get ready to share

::: notes Breakout at about 5 after the hour: - Check in with each group - have students share screen :::

Due Friday (PR)

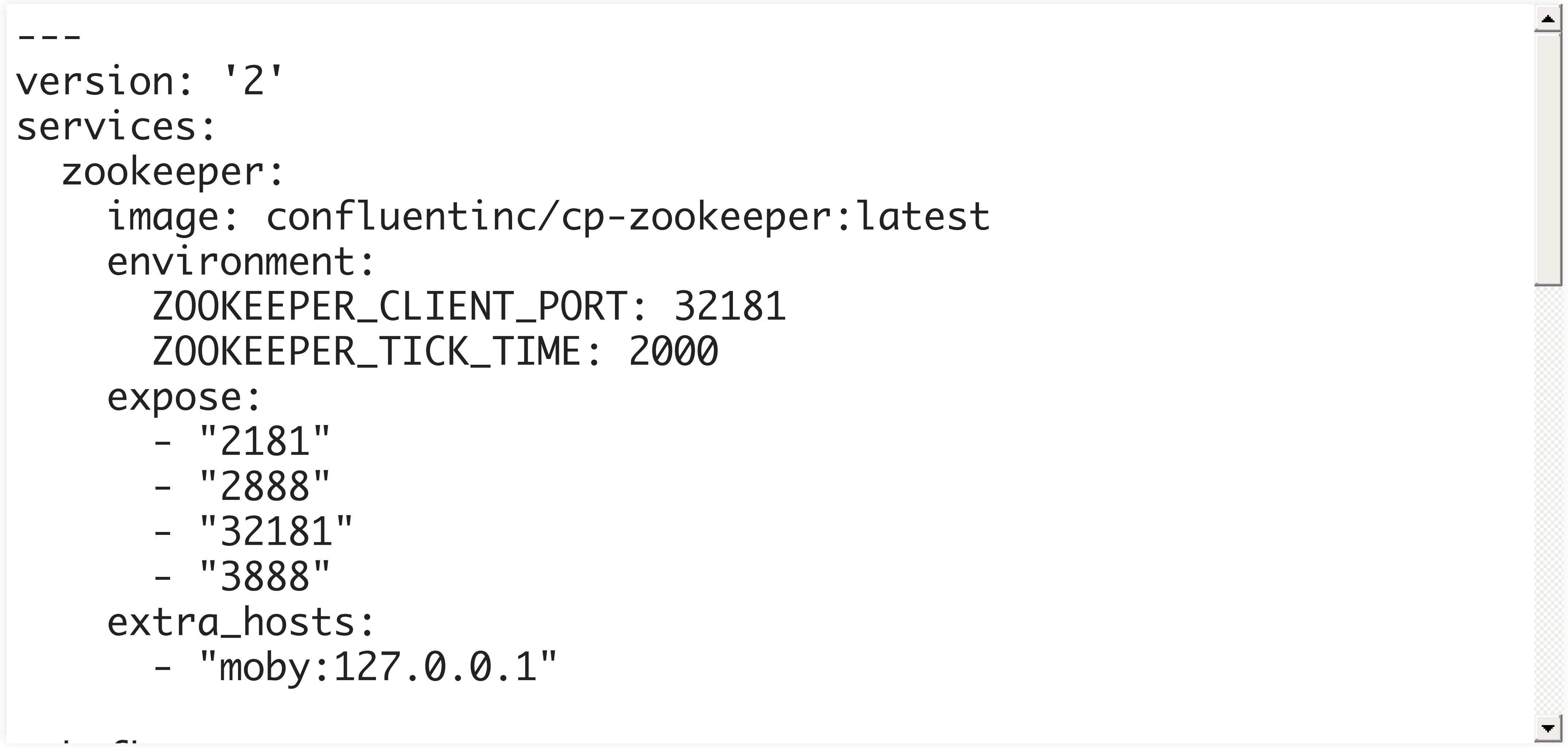


::: notes - Last week we published and consumed messages with kafka - Now we'll consume messages

## Spark Stack with Kafka

## Setup

# docker-compose.yml



```
---
version: '2'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    environment:
      ZOOKEEPER_CLIENT_PORT: 32181
      ZOOKEEPER_TICK_TIME: 2000
    expose:
      - "2181"
      - "2888"
      - "32181"
      - "3888"
    extra_hosts:
      - "moby:127.0.0.1"
```

::: notes and save a data file.

:::

## up

```
docker-compose up -d

docker-compose logs -f kafka
```

::: notes Now spin up the cluster

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

and watch it come up

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

when this looks like it's done, you can safely 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
```

::: notes First, 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 show

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

## Check the topic

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

## Should show

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

## Publish some stuff to kafka

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

::: notes Use the kafka console producer to publish some test messages to that topic

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

:::

## Should show

```
Produced 42 messages.
```

## Run spark using the `myspark` container

```
docker-compose exec myspark pyspark
```

::: notes Spin up a pyspark process using the `myspark` container

docker-compose exec myspark pyspark

We have to add some kafka library dependencies on the cli for now. :::

# read stuff from kafka

At the pyspark prompt,

```
numbers = spark \
  .read \
  .format("kafka") \
  .option("kafka.bootstrap.servers", "kafka:29092") \
  .option("subscribe","foo") \
  .option("startingOffsets", "earliest") \
  .option("endingOffsets", "latest") \
  .load()
```

::: notes At the pyspark prompt,

read from kafka

numbers = spark
.read
.format("kafka")
.option("kafka.bootstrap.servers", "kafka:29092")

## See the schema

```
numbers.printSchema()
```

## Cast it as strings

```
numbers_as_strings=numbers.selectExpr("CAST(key AS STRING)", "CAST(v
```

::: notes cast it as strings (you can totally use INTs if you'd like)

numbers_as_strings=numbers.selectExpr("CAST(key AS STRING)", "CAST(value AS STRING)") :::

## Take a look

```
numbers_as_strings.show()
```

```
numbers_as_strings.printSchema()
```

```
numbers_as_strings.count()
```

::: notes then you can exit pyspark using either `ctrl-d` or `exit()`.

`numbers_as_strings.show()`

numbers_as_strings.printSchema()

:::

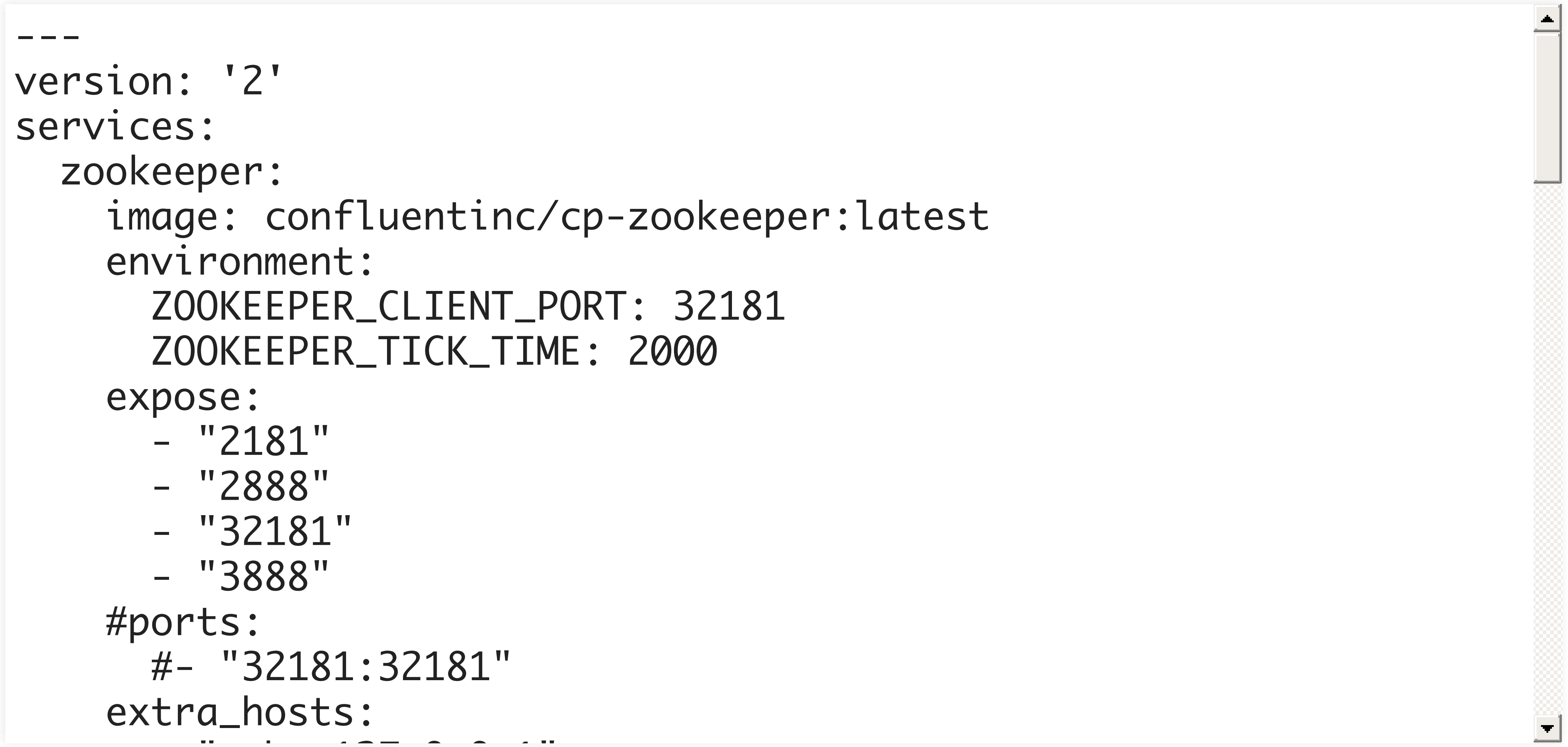
## down

```
docker-compose down
```

# Spark stack with Kafka with "real" messages

::: notes :::

# 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"
    #ports:
      #- "32181:32181"
    extra_hosts:
```

::: notes - See ~/w205/spark-with-kafka-json folder's docker-compose.yml - may use this cluster, may have it working w/o myhdfs container :::

## Pull data

```
curl -L -o github-example-large.json https://goo.gl/Hr6erG
```

## Spin up the cluster

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

## Watch it come up

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

- Detach with `Ctrl-C`

::: notes when this looks like it's done, detach :::

use it

datascience@berkeley

## create a topic

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

::: notes 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
```

::: notes 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/github-example-large.jso
docker-compose exec mids bash -c "cat /w205/github-example-large.jso
```

::: notes ugly 1st pretty print :::

## Individual messages

```
docker-compose exec mids bash -c "cat /w205/github-example-large.jso
```

::: notes Go over | jq stuff :::

## Publish some test messages to that topic with the kafka console producer

```
docker-compose exec mids \
  bash -c "cat /w205/github-example-large.json \
    | jq '.[]' -c \
    | kafkacat -P -b kafka:29092 -t foo && echo 'Produced 100 messag
```

::: notes docker-compose exec mids bash -c "cat /w205/github-example-large.json | jq '.' -c | kafkacat -P -b kafka:29092 -t foo && echo 'Produced 100 messages.'" :::

## Should see something like

```
Produced 100 messages.
```

## Run spark using the `myspark` container

```
docker-compose exec myspark pyspark
```

::: notes Spin up a pyspark process using the `myspark` container

docker-compose exec myspark pyspark

We have to add some kafka library dependencies on the cli for now. :::

# read stuff from kafka

At the pyspark prompt,

```
messages = spark \
  .read \
  .format("kafka") \
  .option("kafka.bootstrap.servers", "kafka:29092") \
  .option("subscribe","foo") \
  .option("startingOffsets", "earliest") \
  .option("endingOffsets", "latest") \
  .load()
```

::: notes At the pyspark prompt,

read from kafka

messages = spark

.read

.format("kafka")

.option("kafka.bootstrap.servers", "kafka:29092")

## See the schema

```
messages.printSchema()
```

## Cast as strings

```
messages_as_strings=messages.selectExpr("CAST(key AS STRING)", "CAST
```

::: notes cast it as strings (you can totally use INTs if you'd like)

messages_as_strings=messages.selectExpr("CAST(key AS STRING)", "CAST(value AS STRING)") :::

datascience@berkeley

## Take a look

```
messages_as_strings.show()
```

```
messages_as_strings.printSchema()
```

```
messages_as_strings.count()
```

::: notes then you can exit pyspark using either `ctrl-d` or `exit()`.

`messages_as_strings.show()`

messages_as_strings.printSchema()

messages_as_strings.count() :::

# Unrolling json

```
messages_as_strings.select('value').take(1)
```

```
messages_as_strings.select('value').take(1)[0].value
```

```
import json
```

```
first_message=json.loads(messages_as_strings.select('value').take(1)
```

```
first_message
```

```
print(first_message['commit']['committer']['name'])
```

::: notes messages_as_strings.select('value').take(1)

messages_as_strings.select('value').take(1)[0].value >>> import json >>> first_message=json.loads(messages_as_strings.select('value').take(1 [0].value) >>> first_message >>> print(first_message['commit']['committer']['name']) Nice

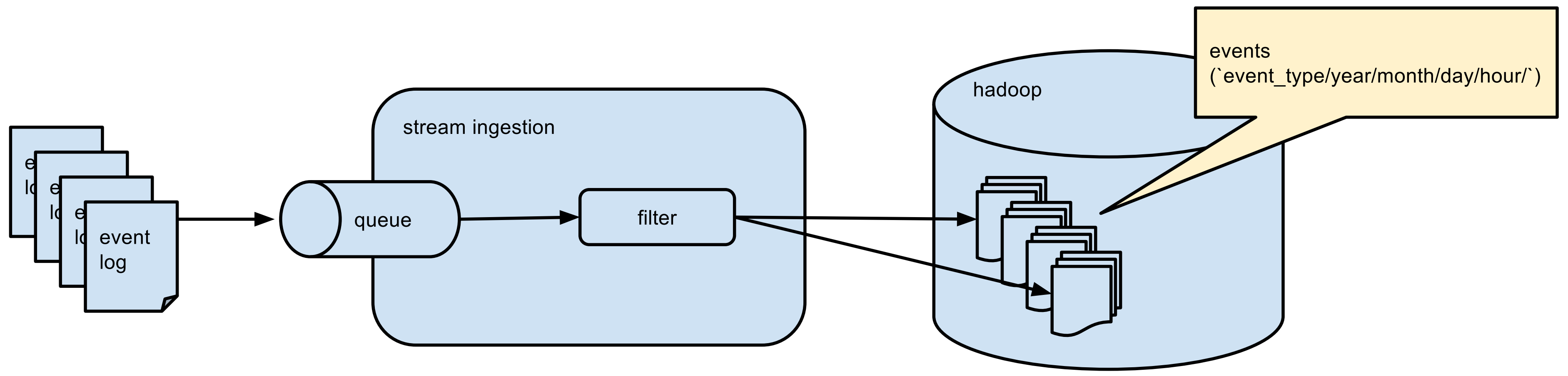
## Down

```
docker-compose down
```

## Assignment 07

- Step through this process using the Project 2 data
- What you turn in:
- In your `/assignment-07-<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`)

# Summary



::: notes - Review what we just did :::