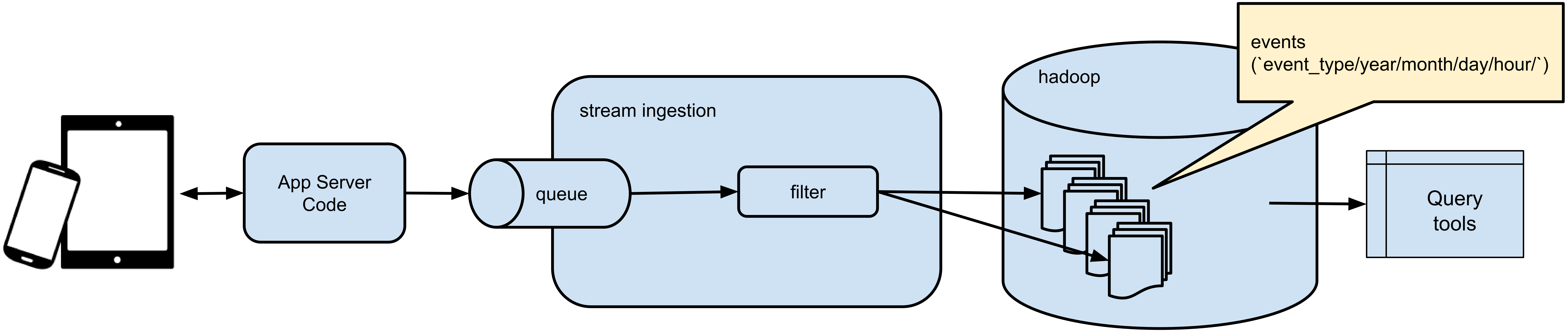
Fundamentals of Data Engineering

Week 11 - sync session

datascience@berkeley

Assignment Review

- Review your Assignment 10
- Get ready to share
- `docker pull midsw205/base:latest`
- `git pull` in `~/w205/course-content`


App Server Code
queue
stream ingestion
filter
hadoop
events (`event_type/year/month/day/hour/`)
Query tools

Project 3 Group Breakout

- Plan for project
- Which events will you include?
- Which parameters will you include?
- What will you track the state of?
- How will you need to change:
  - flask app code?
  - pyspark code?
  - code to implement tracking state?

Running Spark Jobs

Setup

Set up directory, get docker-compose

```
mkdir ~/w205/spark-from-files/
cd ~/w205/spark-from-files
cp ~/w205/course-content/11-Storing-Data-III/docker-compose.yml .
cp ~/w205/course-content/11-Storing-Data-III/*.py .
```

The docker-compose.yml

Create a docker-compose.yml with the following

```yaml
---
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"
```

Spin up the cluster

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

Wait for things to come up

```
docker-compose logs -f cloudera
```

Check out hadoop

```
docker-compose exec cloudera hadoop fs -ls /tmp/
```

Create a topic

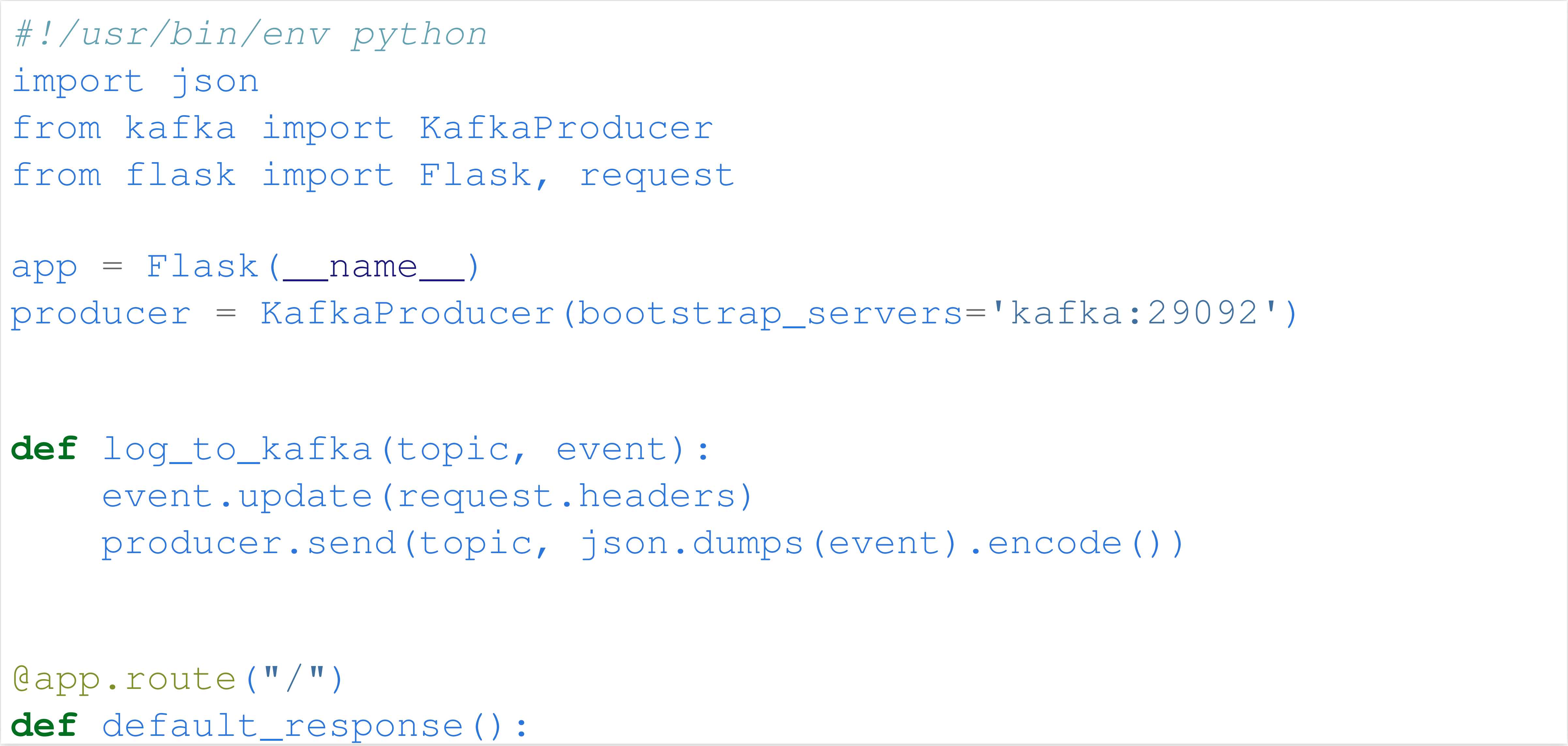
```
docker-compose exec kafka \
  kafka-topics \
    --create \
    --topic events \
    --partitions 1 \
    --replication-factor 1 \
    --if-not-exists --zookeeper zookeeper:32181
```

Should show

```
Created topic "events".
```

Flask

Take our flask app - with request.headers



```
#!/usr/bin/env python
import json
from kafka import KafkaProducer
from flask import Flask, request

app = Flask(__name__)
producer = KafkaProducer(bootstrap_servers='kafka:29092')


def log_to_kafka(topic, event):
    event.update(request.headers)
    producer.send(topic, json.dumps(event).encode())


@app.route("/")
def default_response():
```

Run it

```
docker-compose exec mids \
  env FLASK_APP=/w205/spark-from-files/game_api.py \
  flask run --host 0.0.0.0
```

Generate events from browser

- localhost:5000/
- localhost:5000/purchase_a_sword

Read from kafka

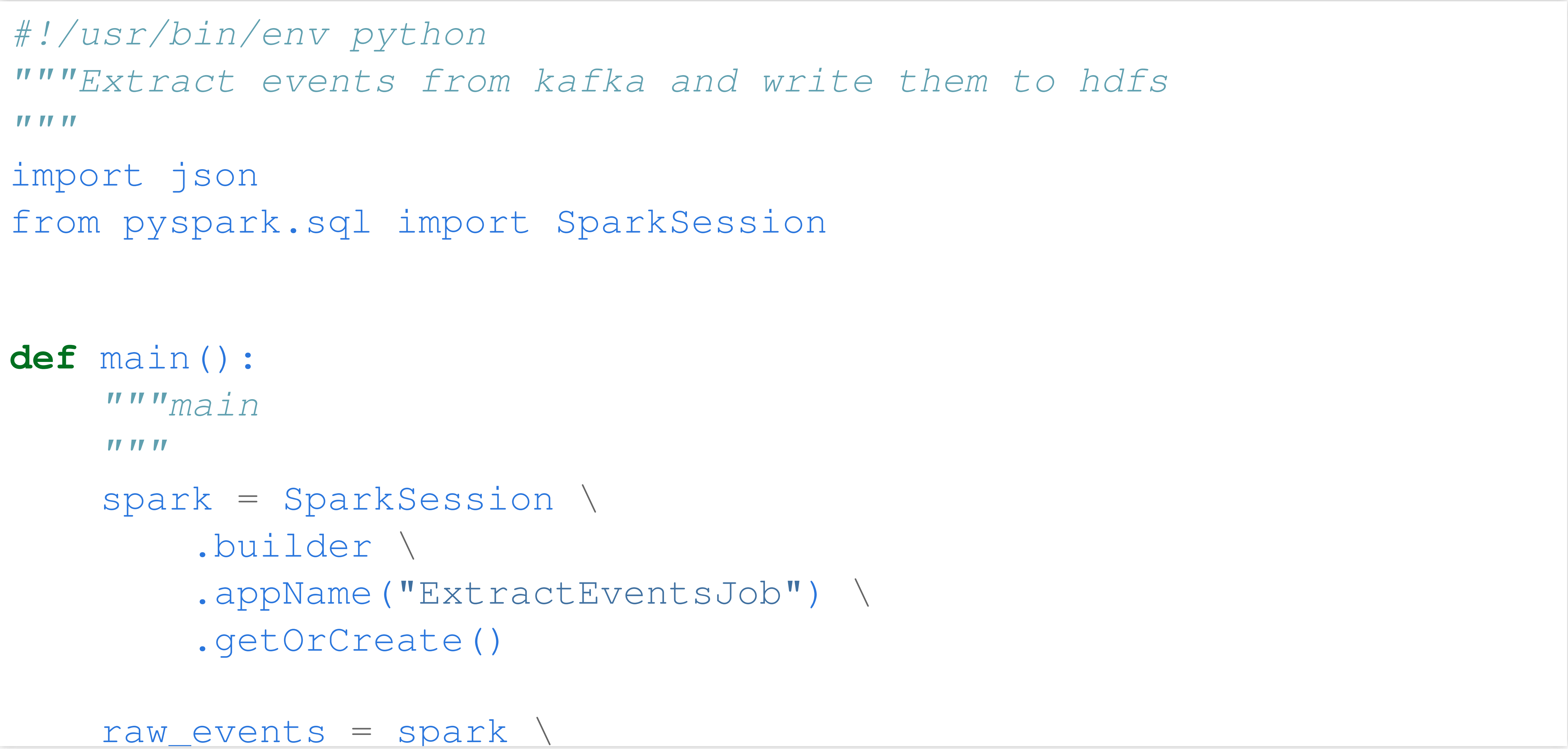
```
docker-compose exec mids \
  kafkacat -C -b kafka:29092 -t events -o beginning -e
```

Should see

```
{"Host": "localhost:5000", "event_type": "default", "Accept": "*/*",
{"Host": "localhost:5000", "event_type": "default", "Accept": "*/*",
{"Host": "localhost:5000", "event_type": "default", "Accept": "*/*",
{"Host": "localhost:5000", "event_type": "purchase_sword", "Accept":
{"Host": "localhost:5000", "event_type": "purchase_sword", "Accept":
{"Host": "localhost:5000", "event_type": "purchase_sword", "Accept":
{"Host": "localhost:5000", "event_type": "purchase_sword", "Accept":
...
```

Spark

Capture our pyspark code in a file this time



```
#!/usr/bin/env python
"""Extract events from kafka and write them to hdfs
"""
import json
from pyspark.sql import SparkSession


def main():
    """main
    """
    spark = SparkSession \
        .builder \
        .appName("ExtractEventsJob") \
        .getOrCreate()

    raw_events = spark \
```

run it

```
docker-compose exec spark \
  spark-submit \
    /w205/spark-from-files/extract_events.py
```

if you didn’t generate any events

```
Traceback (most recent call last):
  File "/w205/spark-from-files/extract_events.py", line 35, in <modul
    main()
  File "/w205/spark-from-files/extract_events.py", line 27, in main
    extracted_events = events.rdd.map(lambda x: json.loads(x.value)).
  File "/spark-2.2.0-bin-hadoop2.6/python/lib/pyspark.zip/pyspark/sql
  File "/spark-2.2.0-bin-hadoop2.6/python/lib/pyspark.zip/pyspark/sql
  File "/spark-2.2.0-bin-hadoop2.6/python/lib/pyspark.zip/pyspark/sql
  File "/spark-2.2.0-bin-hadoop2.6/python/lib/pyspark.zip/pyspark/sql
  File "/spark-2.2.0-bin-hadoop2.6/python/lib/pyspark.zip/pyspark/rdd
ValueError: RDD is empty
```

check out results in hadoop

```
docker-compose exec cloudera hadoop fs -ls /tmp/
```

and

```
    docker-compose exec cloudera hadoop fs -ls /tmp/extracted_events/
```

Deploying a Spark job to a cluster

```
docker-compose exec spark spark-submit filename.py
```

is really just

```
docker-compose exec spark \
  spark-submit \
    --master 'local[*]' \
    filename.py
```

standalone

```
docker-compose exec spark \
  spark-submit \
    --master spark://23.195.26.187:7077 \
    filename.py
```

(this won’t work here)

yarn

```
docker-compose exec spark \
  spark-submit \
    --master yarn \
    --deploy-mode cluster \
    filename.py
```

(this won’t work here)

mesos

```
docker-compose exec spark \
  spark-submit \
    --master mesos://mesos-master:7077 \
    --deploy-mode cluster \
    filename.py
```

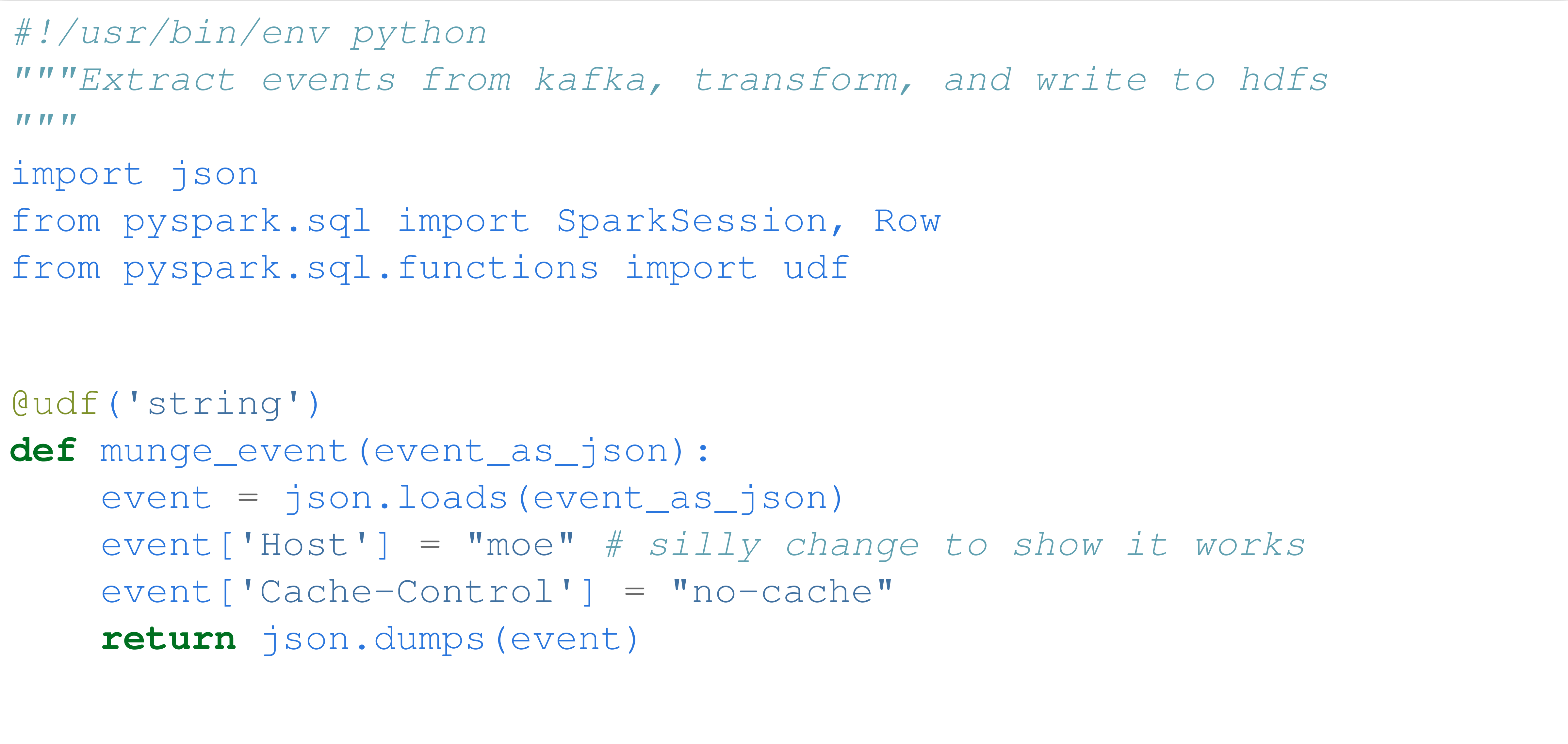
(this won’t work here)

kubernetes

```
docker-compose exec spark \
  spark-submit \
    --master k8s://kubernetes-master:443 \
    --deploy-mode cluster \
    filename.py
```

(this won’t work here)

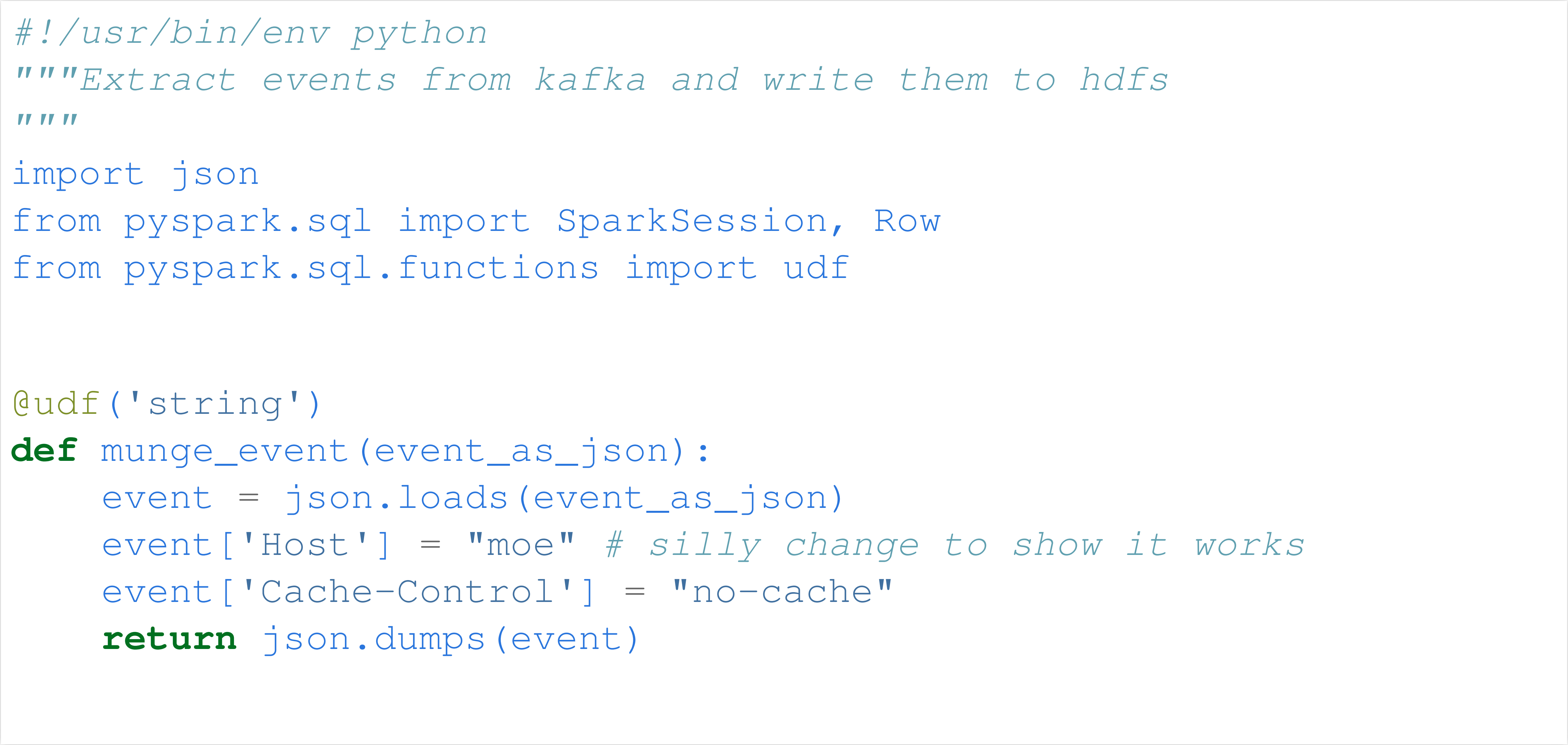
More Spark!



```python
#!/usr/bin/env python
"""Extract events from kafka, transform, and write to hdfs
"""
import json
from pyspark.sql import SparkSession, Row
from pyspark.sql.functions import udf


@udf('string')
def munge_event(event_as_json):
    event = json.loads(event_as_json)
    event['Host'] = "moe" # silly change to show it works
    event['Cache-Control'] = "no-cache"
    return json.dumps(event)
```

Let’s look at separating events



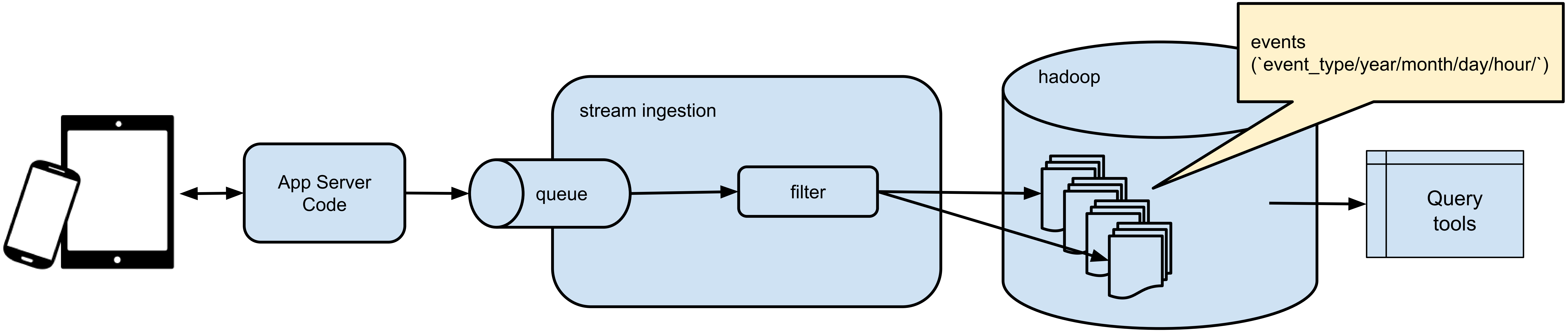
```python
#!/usr/bin/env python
"""Extract events from kafka and write them to hdfs
"""
import json
from pyspark.sql import SparkSession, Row
from pyspark.sql.functions import udf


@udf('string')
def munge_event(event_as_json):
    event = json.loads(event_as_json)
    event['Host'] = "moe" # silly change to show it works
    event['Cache-Control'] = "no-cache"
    return json.dumps(event)
```

Remember to tear down your cluster

```
docker-compose down
```

Summary


App Server Code
stream ingestion
queue
filter
hadoop
events (`event_type/year/month/day/hour/`)
Query tools

Berkeley SCHOOL OF INFORMATION