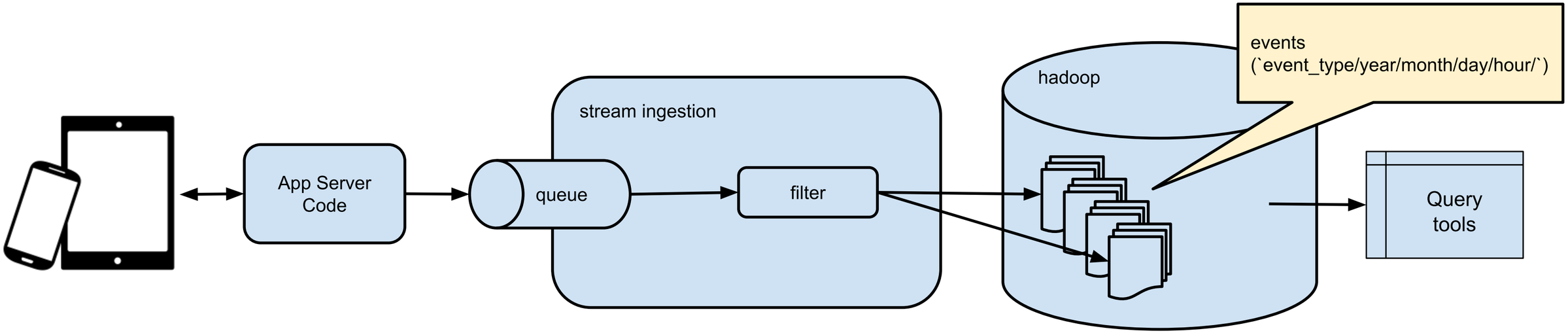
Fundamentals of Data Engineering

Week 13 - sync session

datascience@berkeley

Get Started

```
git pull in ~/w205/course-content
mkdir ~/w205/full-stack2/
cd ~/w205/full-stack2
cp ~/w205/course-content/13-Understanding-Data/docker-compose.yml .
docker-compose pull
cp ~/w205/course-content/13-Understanding-Data/*.py .
```


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

Flask-Kafka-Spark-Hadoop-Presto Part II

Setup

The `docker-compose.yml`

Create a `docker-compose.yml` with the following

```
---
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"

  kafka:
```

Spin up the cluster

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

Web-app

- Take our instrumented web-app from before
  `~/w205/full-stack2/game_api.py`

```
#!/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 flask

```
docker-compose exec mids \
  env FLASK_APP=/w205/full-stack2/game_api.py \
  flask run --host 0.0.0.0
```

Set up to watch kafka

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

Apache Bench to generate data

```
docker-compose exec mids \
  ab \
    -n 10 \
    -H "Host: user1.comcast.com" \
    http://localhost:5000/
```

```
docker-compose exec mids \
  ab \
    -n 10 \
    -H "Host: user1.comcast.com" \
    http://localhost:5000/purchase_a_sword
```

```
docker-compose exec mids \
  ab \
    -n 10 \
    -H "Host: user2.att.com" \
    http://localhost:5000/
```

```
docker-compose exec mids \
  ab \
    -n 10 \
    -H "Host: user2.att.com" \
    http://localhost:5000/purchase_a_sword
```

Some Spark to Write Events

```
#!/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('boolean')
def is_purchase(event_as_json):
    event = json.loads(event_as_json)
    if event['event_type'] == 'purchase_sword':
        return True
    return False
```

Run this

```
docker-compose exec spark spark-submit /w205/full-stack2/filtered_wri
```

See purchases in hdfs

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

Queries from Presto

Hive metastore

- Track schema
- Create a table

Hard Way

```
docker-compose exec cloudera hive
```

```
create external table if not exists default.purchases2 (
    Accept string,
    Host string,
    User_Agent string,
    event_type string,
    timestamp string
  )
  stored as parquet
  location '/tmp/purchases'
  tblproperties ("parquet.compress"="SNAPPY");
```

Or... we can do this an easier way

```
docker-compose exec spark pyspark
```

```
df = spark.read.parquet('/tmp/purchases')
df.registerTempTable('purchases')
query = """
create external table purchase_events
  stored as parquet
  location '/tmp/purchase_events'
  as
  select * from purchases
"""
spark.sql(query)
```

Can just include all that in job

```
#!/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('boolean')
def is_purchase(event_as_json):
    event = json.loads(event_as_json)
    if event['event_type'] == 'purchase_sword':
        return True
    return False
```

Run this

```
docker-compose exec spark spark-submit /w205/full-stack2/write_hive_t
```

See it wrote to hdfs

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

and now ...

- Query this with presto

```
docker-compose exec presto presto --server presto:8080 --catalog hive
```

What tables do we have in Presto?

```
presto:default> show tables;
   Table
-----------
 purchases
(1 row)

Query 20180404_224746_00009_zsma3, FINISHED, 1 node
Splits: 2 total, 1 done (50.00%)
0:00 [1 rows, 34B] [10 rows/s, 342B/s]
```

Describe `purchases` table

```
presto:default> describe purchases;
   Column    |  Type   | Comment
-------------+---------+---------
 accept      | varchar |
 host        | varchar |
 user-agent  | varchar |
 event_type  | varchar |
 timestamp   | varchar |
(5 rows)

Query 20180404_224828_00010_zsma3, FINISHED, 1 node
Splits: 2 total, 1 done (50.00%)
0:00 [5 rows, 344B] [34 rows/s, 2.31KB/s]
```

Query `purchases` table

```
presto:default> select * from purchases;
 accept |       host          |   user-agent     |   event_type    |
--------+---------------------+------------------+-----------------+-----
 */*    | user1.comcast.com | ApacheBench/2.3 | purchase_sword | 2018
 */*    | user1.comcast.com | ApacheBench/2.3 | purchase_sword | 2018
 */*    | user1.comcast.com | ApacheBench/2.3 | purchase_sword | 2018
 */*    | user1.comcast.com | ApacheBench/2.3 | purchase_sword | 2018
 */*    | user1.comcast.com | ApacheBench/2.3 | purchase_sword | 2018
 ...
```

Streaming

Getting our spark ready for streaming

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


def purchase_sword_event_schema():
    """
    root
    |-- Accept: string (nullable = true)
    |-- Host: string (nullable = true)
    |-- User-Agent: string (nullable = true)
    |-- event_type: string (nullable = true)
```

Run

```
docker-compose exec spark spark-submit /w205/full-stack2/filter_sworc
```

Turn that into a stream

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


def purchase_sword_event_schema():
    """
    root
    |-- Accept: string (nullable = true)
    |-- Host: string (nullable = true)
    |-- User-Agent: string (nullable = true)
    |-- event_type: string (nullable = true)
```

Run it

```
docker-compose exec spark spark-submit /w205/full-stack2/filter_sworc
```

Kick some more events

```
docker-compose exec mids \
  ab \
    -n 10 \
    -H "Host: user1.comcast.com" \
    http://localhost:5000/
```

```
docker-compose exec mids \
  ab \
    -n 10 \
    -H "Host: user1.comcast.com" \
    http://localhost:5000/purchase_a_sword
```

```
docker-compose exec mids \
  ab \
    -n 10 \
    -H "Host: user2.att.com" \
    http://localhost:5000/
```

```
docker-compose exec mids \
  ab \
    -n 10 \
    -H "Host: user2.att.com" \
    http://localhost:5000/purchase_a_sword
```

Write from a stream

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


def purchase_sword_event_schema():
    """
    root
    |-- Accept: string (nullable = true)
    |-- Host: string (nullable = true)
    |-- User-Agent: string (nullable = true)
    |-- event_type: string (nullable = true)
```

Run it

```
docker-compose exec spark spark-submit /w205/full-stack2/write_swords
```

Feed it

```
while true; do
  docker-compose exec mids \
    ab -n 10 -H "Host: user1.comcast.com" \
      http://localhost:5000/purchase_a_sword
  sleep 10
done
```

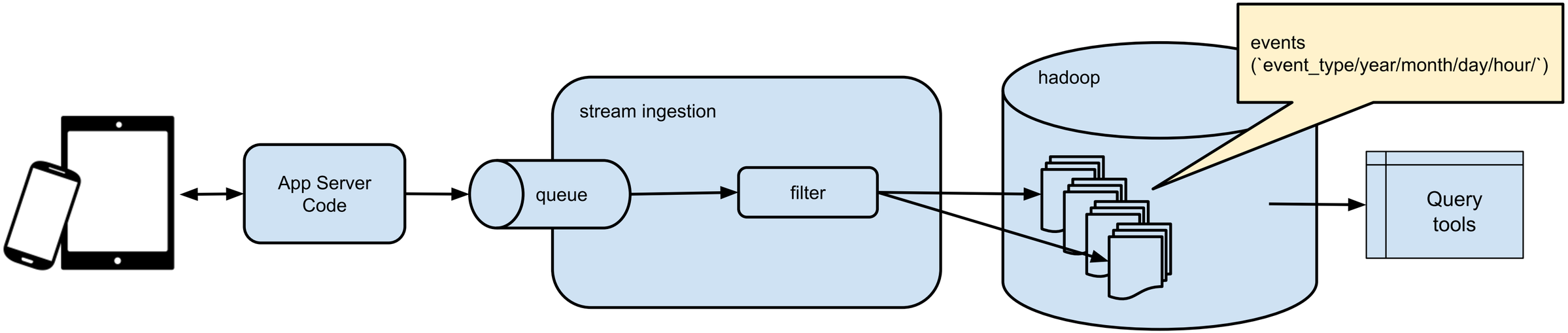
Check what it wrote to Hadoop

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

down

```
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