Fundamentals of Data Engineering

Week 14 - sync session

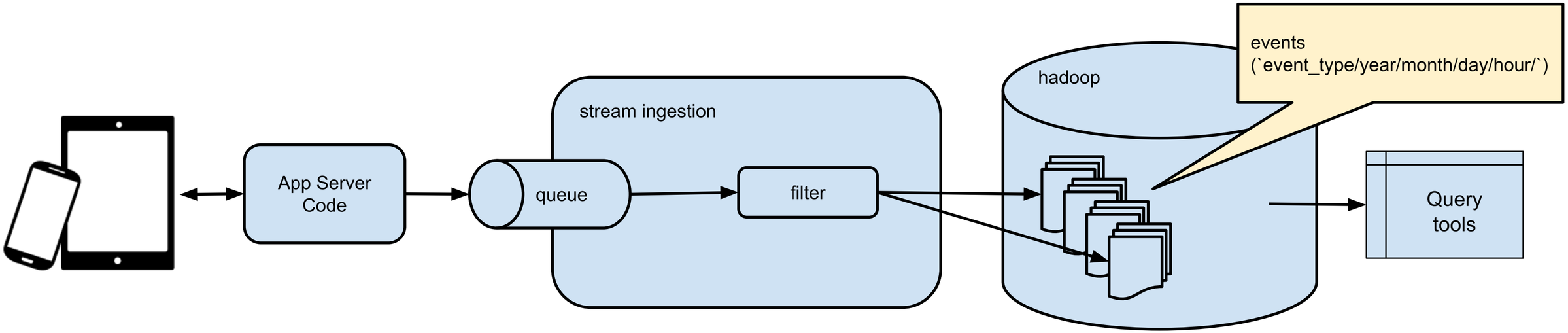
datascience@berkeley

Get Started

```
git pull in ~/w205/course-content
mkdir ~/w205/full-streaming-stack/
cd ~/w205/full-streaming-stack
cp ~/w205/course-content/14-Patterns-for-Data-Pipelines/docker-compos
docker-compose pull
cp ~/w205/course-content/14-Patterns-for-Data-Pipelines/*.py .
```

Announcements

- repos - fork or clone your repos by 27Apr2018
- future - `course-content` tagged by semester


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

Full-Stack Streaming

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"
```

Spin up the cluster

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

Web-app

- Take our instrumented web-app from before
  `~/w205/full-streaming-stack/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-streaming-stack/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
```

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 it

```
docker-compose exec spark spark-submit /w205/full-streaming-stack/wri
```

Check what it wrote to Hadoop

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

Set up Presto

Hive metastore

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

```
create external table if not exists default.sword_purchases (
    raw_event string,
    timestamp string,
    Accept string,
    Host string,
    User_Agent string,
    event_type string

  )
  stored as parquet
  location '/tmp/sword_purchases'
  tblproperties ("parquet.compress"="SNAPPY");
```

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;
```

Describe `sword_purchases` table

```
presto:default> describe sword_purchases;
```

Query `purchases` table

```
presto:default> select * from sword_purchases;
```

Add some data

Seed a little data into the stream

```
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
```

Query `purchases` table

```
presto:default> select * from sword_purchases;
```

More data

Feed the stream more data

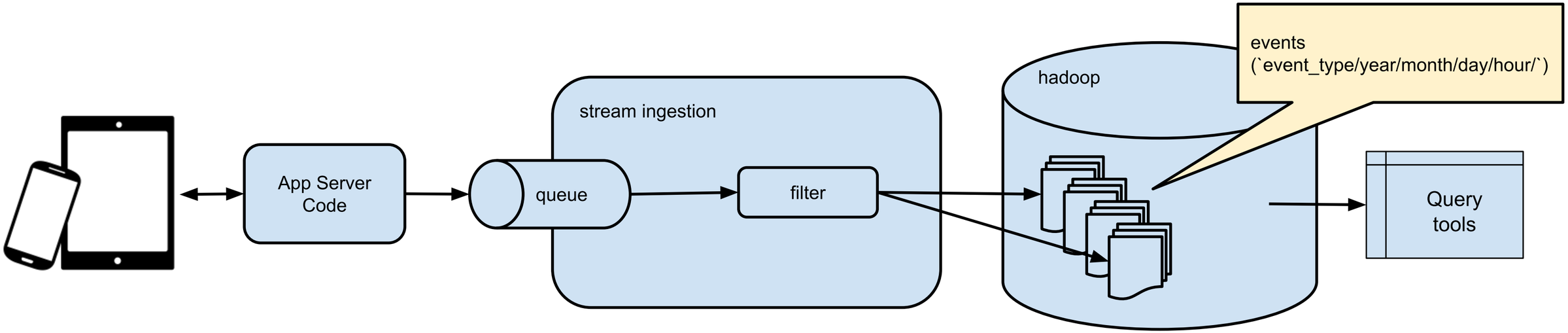
```
while true; do
  docker-compose exec mids \
    ab -n 10 -H "Host: user1.comcast.com" \
      http://localhost:5000/purchase_a_sword
  sleep 10
done
```

Watch presto grow

```
presto:default> select count(*) from sword_purchases;
```

down

```
docker-compose down
```


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

Building Docker Images

Setup

```
mkdir -p ~/w205/docker/mytools
cd ~/w205/docker/mytools
```

The `Dockerfile`

Save this as `Dockerfile` in

`~/w205/docker/mytools/`

```
FROM ubuntu:xenial
MAINTAINER Mark Mims <mark@digitalocean.com>

RUN apt-get -qq update \
  && apt-get -qq install -y jq apache2-utils
```

Build

```
docker build -t <tag> <path>
```

so, from a folder containing a `Dockerfile`,

```
docker build -t mytools .
```

check build ids and tags

```
docker images | grep mytools
```

test a build

```
docker run -it --rm mytools bash
```

then at the prompt

```
which jq
```

What did we do?

```
docker run -it --rm ubuntu:xenial which jq
docker run -it --rm mytools which jq
```

Iterate

You can do more in a `Dockerfile`

```
FROM ubuntu:16.04
MAINTAINER Mark Mims <mark@digitalocean.com>

ENV SPARK_VERSION         2.2.0
ENV SPARK_HADOOP_VERSION 2.6

ENV SPARK_HOME /spark-$SPARK_VERSION-bin-hadoop$SPARK_HADOOP_VERSION
ENV JAVA_HOME  /usr/lib/jvm/java-8-oracle

ENV SPARK_TEMPLATE_PATH $SPARK_HOME/templates
ENV SPARK_CONF_PATH $SPARK_HOME/conf

ENV PATH $SPARK_HOME/bin:$PATH

RUN echo oracle-java8-installer shared/accepted-oracle-license-v1-1 s
  && apt-get update \
```

Examples of different `Dockerfile`s

- nginx
- mongo
- mysql
- python
- etc...

Berkeley SCHOOL OF INFORMATION