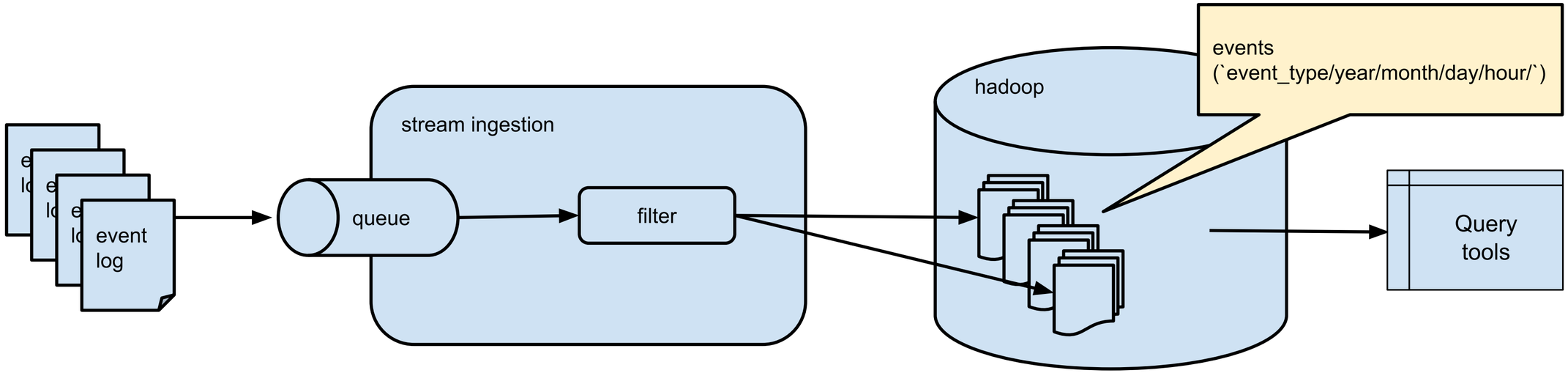
Fundamentals of Data Engineering

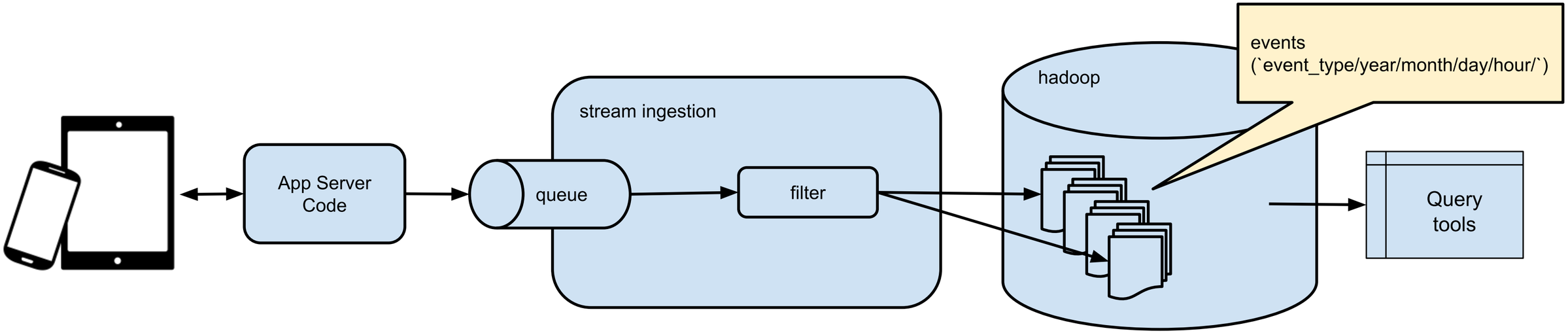
Week 10 - sync session

datascience@berkeley

Assignment Review

- Review your Assignment 09
- Get ready to share
- `docker pull midsw205/base:latest`
- `git pull` in `~/w205/course-content`


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


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

Project 3 options

- All: Essential game shopping cart data for homework
- Advanced option 1: Generate and filter more types of items
- Advanced option 2: Enhance the API to accept parameters for purchases (sword/item type)
- Advanced option 3: Shopping cart data & track state (e.g., user's inventory)

Flask with Kafka and Spark

Set up directory, get docker-compose

```
mkdir ~/w205/flask-with-kafka-and-spark/
```

```
cd ~/w205/flask-with-kafka-and-spark/
```

```
cp ~/w205/course-content/10-Transforming-Streaming-Data/docker-compos
```

The `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"

  kafka:
```

Spin up the cluster

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

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".
```

Web-app

```
#!/usr/bin/env python
from kafka import KafkaProducer
from flask import Flask
app = Flask(__name__)
producer = KafkaProducer(bootstrap_servers='kafka:29092')
topic = 'events'

@app.route("/")
def default_response():
    producer.send(topic, 'default'.encode())
    return "This is the default response!\n"

@app.route("/purchase_a_sword")
def purchase_a_sword():
    producer.send(topic, 'purchased_sword'.encode())
    return "Sword Purchased!\n"
```

More informative events

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

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


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


@app.route("/")
def default_response():
    default_event = {'event_type': 'default'}
```

Run it

```
docker-compose exec mids \
  env FLASK_APP=/w205/flask-with-kafka-and-spark/game_api_with_json_e
  flask run --host 0.0.0.0
```

Test it by generating events

```
docker-compose exec mids curl http://localhost:5000/
```

```
docker-compose exec mids curl http://localhost:5000/purchase_a_sword
```

Read from kafka

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

Should show

```
{"event_type": "default"}
{"event_type": "default"}
{"event_type": "default"}
{"event_type": "purchase_sword"}
{"event_type": "purchase_sword"}
{"event_type": "purchase_sword"}
{"event_type": "purchase_sword"}
...
```

Add more events

- Let’s add more stuff to the events we’re sending.
- Will do this over 2 breakouts.

Breakout 1 Discussion

- Discuss business requirements for the project
- What are the kinds of events that I need to track as data scientist at the game company?
  - List events
  - Give business reasons for tracking them
  - Groups report out
    - e.g., one event is “join a guild”, what sort of business reason would you be tracking “join a guild”?
    - What other events other than the ones we’re tracking now would you want to track?

Even *more* informative events

```
#!/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/flask-with-kafka-and-spark/game_api_with_extend
  flask run --host 0.0.0.0
```

Test it - generate events

```
docker-compose exec mids curl http://localhost:5000/
```

```
docker-compose exec mids curl http://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":
...
```

Breakout 2 Discussion

- What info is available with these events?
- How should the events be structured?
- i.e.,
  - what flask request objects?
  - what info do they have?
  - what is the schema?

Spark it up

Run a spark shell

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

Read from kafka

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

Explore our events

```
events = raw_events.select(raw_events.value.cast('string'))
```

```
extracted_events = events.rdd.map(lambda x: json.loads(x.value)).toDF
```

```
extracted_events.show()
```

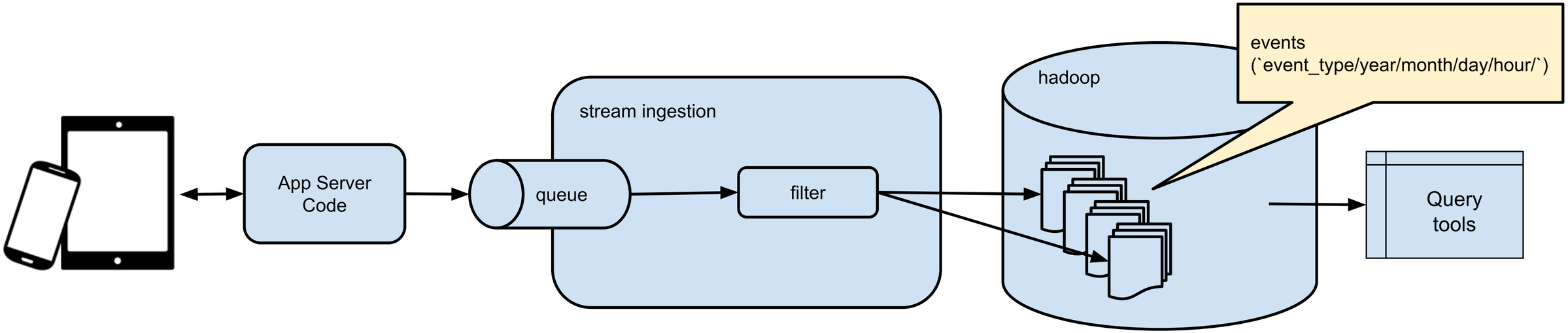
down

```
docker-compose down
```

Week 10 Videos

- Context for how what we're doing in Project 3 works in reality.
- Streaming, batch, and microbatch
- Microbatch window size
- Performing computations- in memory vs.cached
- How to set up pipelines to take action(recommendations, promotions etc) when streaming events come in

Summary


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

Berkeley SCHOOL OF INFORMATION