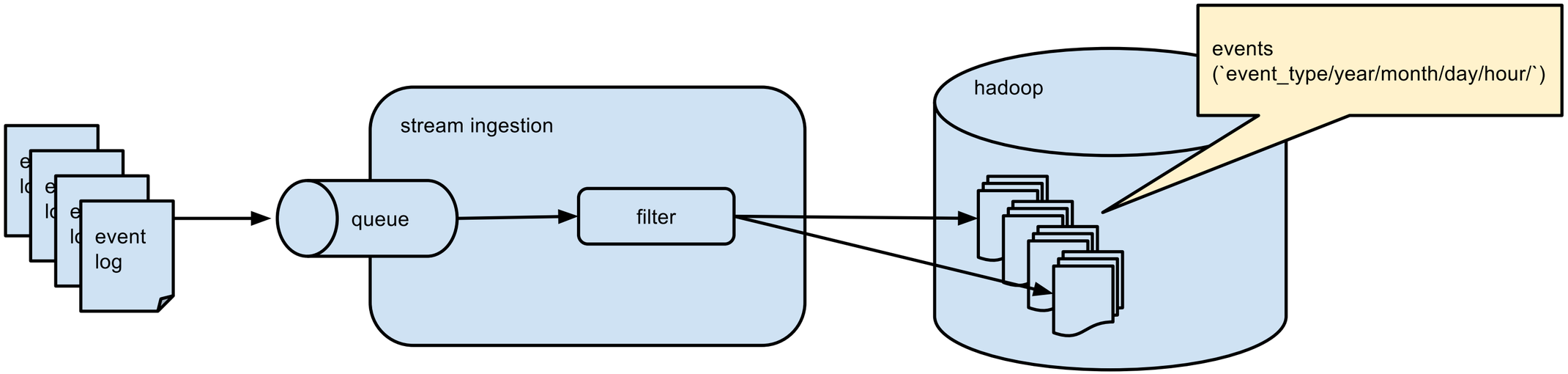
Fundamentals of Data Engineering

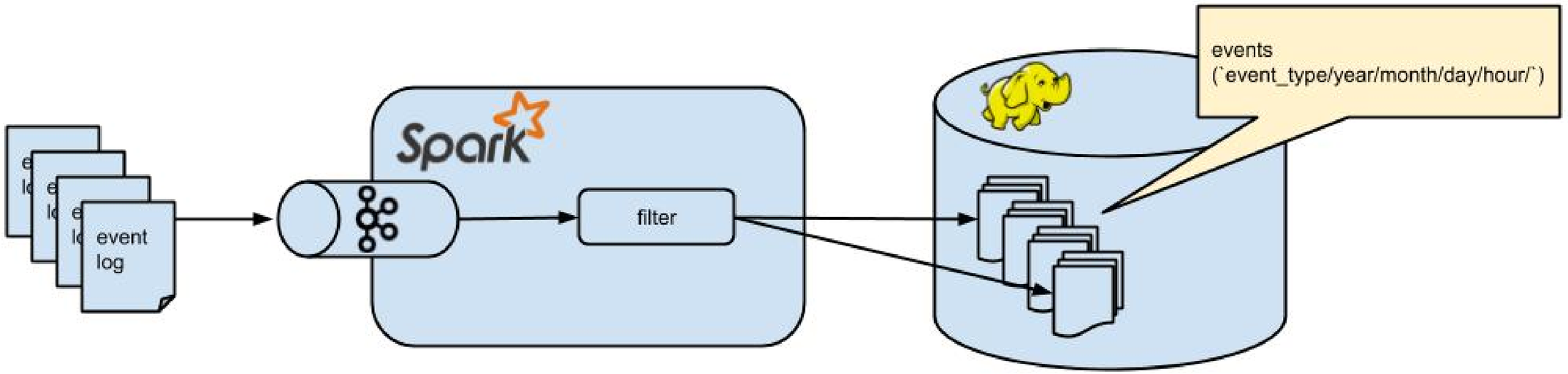
Week 08 - sync session

datascience@berkeley

Project Review

- Review your Project 2
- Get ready to share


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


event
log
Spark
filter
events
(`event_type/year/month/day/hour/`)

Spark Stack with Kafka and HDFS

Setup

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

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

```
cp ~/w205/course-content//08-Querying-Data/docker-compose.yml .
```

Download the dataset for World Cup players

- In `~/w205/`

```
curl -L -o players.json https://goo.gl/vsuCpZ
```

Spin up the cluster

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

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

Example: World Cup Players

Check out Hadoop

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

Should see something like:

```
funwithflags:~/w205/spark-with-kafka-and-hdfs $ docker-compose exec c
Found 2 items
drwxrwxrwt   - mapred mapred              0 2018-02-06 18:27 /tmp/hac
drwx-wx-wx   - root   supergroup          0 2018-02-20 22:31 /tmp/hiv
```

Create a topic `players`

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

Should show

```
Created topic "players".
```

Use kafkacat to produce test messages to the `players` topic

```
docker-compose exec mids \
  bash -c "cat /w205/<your_workspace>/players.json \
    | jq '.[]' -c \
    | kafkacat -P -b kafka:29092 -t players"
```

Spin up a pyspark process using the `spark` container

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

At the pyspark prompt, read from kafka

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

Cache this to cut back on warnings later

```
raw_players.cache()
```

See what we got

```
raw_players.printSchema()
```

Cast it as strings (you can totally use `INT`s if you'd like)

```
players = raw_players.select(raw_players.value.cast('string'))
```

or

```
players = raw_players.selectExpr("CAST(value AS STRING)")
```

Write this to hdfs

```
players.write.parquet("/tmp/players")
```

Check out results (from another terminal window)

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

and

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

However (back in spark terminal window)

- What did we actually write?

```
players.show()
```

Extract Data

Deal with unicode

```
import sys
sys.stdout = open(sys.stdout.fileno(), mode='w', encoding='utf8', buf
```

What do we have?

- Take a look at

```
import json
players.rdd.map(lambda x: json.loads(x.value)).toDF().show()
```

```
extracted_players = players.rdd.map(lambda x: json.loads(x.value)).to
```

```
from pyspark.sql import Row
extracted_players = players.rdd.map(lambda x: Row(**json.loads(x.valu
```

```
extracted_players.show()
```

Save that

```
extracted_players.write.parquet("/tmp/extracted_players")
```

Do

```
players.show()
```

```
extracted_players.show()
```

Example: GitHub Commits

check out hadoop

Let’s check out hdfs before we write anything to it

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

Create a topic

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

Download the dataset for github commits

```
curl -L -o github-example-large.json https://goo.gl/Y4MD58
```

Publish some stuff to kafka

```
docker-compose exec mids \
  bash -c "cat /w205/<your_workspace>/github-example-large.json \
    | jq '.[]' -c \
    | kafkacat -P -b kafka:29092 -t commits"
```

Spin up a pyspark process using the `spark` container

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

Read stuff from kafka

- At the pyspark prompt, read from kafka

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

Cache this to cut back on warnings

```
raw_commits.cache()
```

See what we got

```
raw_commits.printSchema()
```

Take the `value`s as strings

```
commits = raw_commits.select(raw_commits.value.cast('string'))
```

Of course, we *could* just write this to hdfs

```
commits.write.parquet("/tmp/commits")
```

- but let's extract the data a bit first...

Extract more fields

- Let's extract our json fields again

```
extracted_commits = commits.rdd.map(lambda x: json.loads(x.value)).to
```

and see

```
extracted_commits.show()
```

hmmm... did all of our stuff get extracted?

```
extracted_commits.printSchema()
```

- Problem: more nested json than before

Use SparkSQL

- First, create a Spark “TempTable” (aka “View”)

```
extracted_commits.registerTempTable('commits')
```

Then we can create DataFrames from queries

```
spark.sql("select commit.committer.name from commits limit 10").show
```

```
spark.sql("select commit.committer.name, commit.committer.date, sha f
```

Grab what we want

```
some_commit_info = spark.sql("select commit.committer.name, commit.co
```

Write to hdfs

- We can write that out

```
some_commit_info.write.parquet("/tmp/some_commit_info")
```

Check out results

-You can see results in hadoop

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

and

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

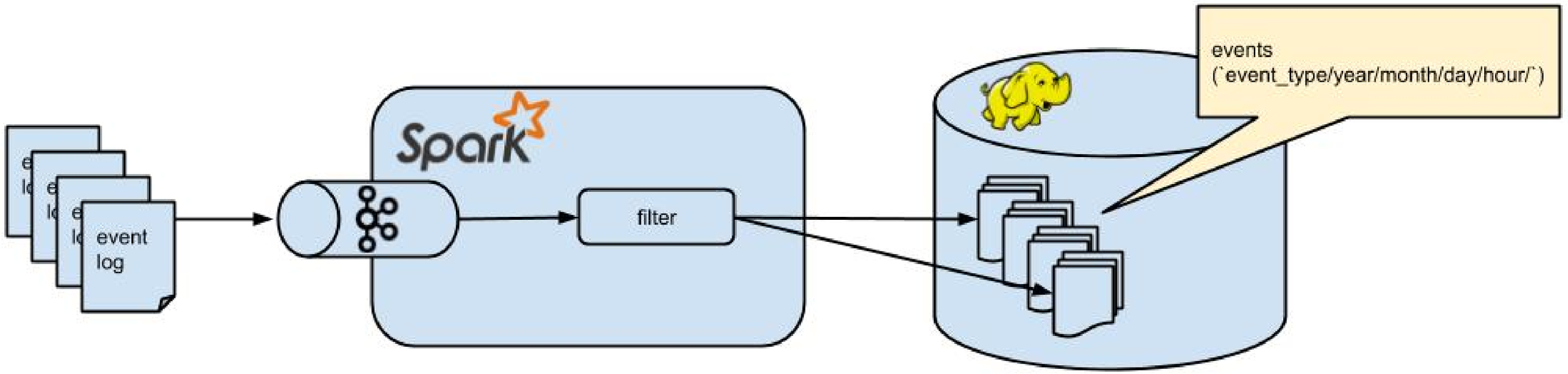
Exit

- Remember, you can exit pyspark using either `ctrl-d` or `exit()`.

Down

```
docker-compose down
```

Summary


event
log
Spark
filter
events
(`event_type/year/month/day/hour/`)

Week 8 Videos

- Context for similar pipeline in reality(aka, at scale, in production)
- Presto query walkthrough
- Partitioning
- How query jobs are scheduled & executed with distributed resources

Berkeley SCHOOL OF INFORMATION