Real-Time Targeted Advertising on Meetup

Final Project

BIGDATA 220: Building the Data Pipeline

Pete Champlin

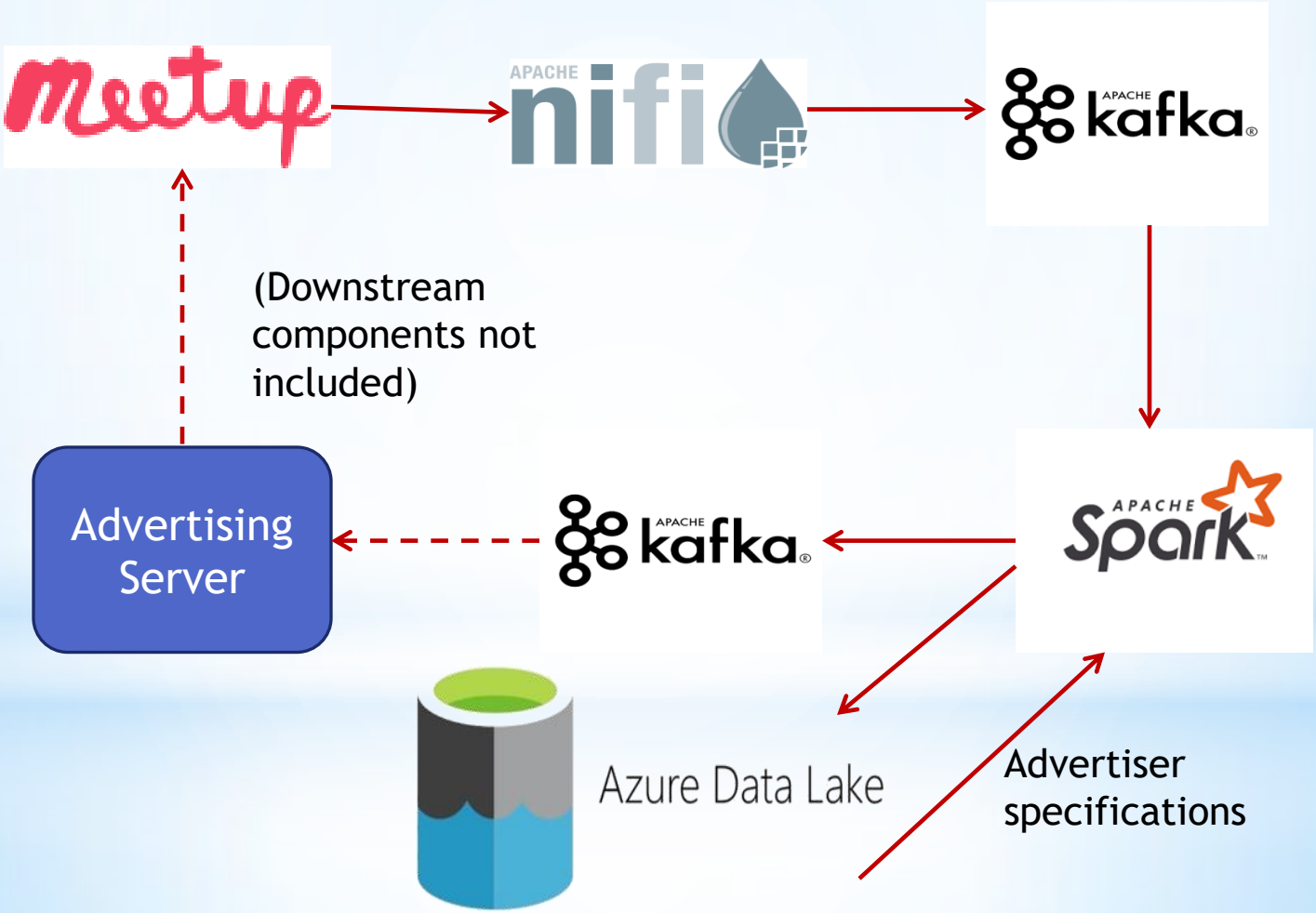
March, 2020

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Data Pipeline Overview

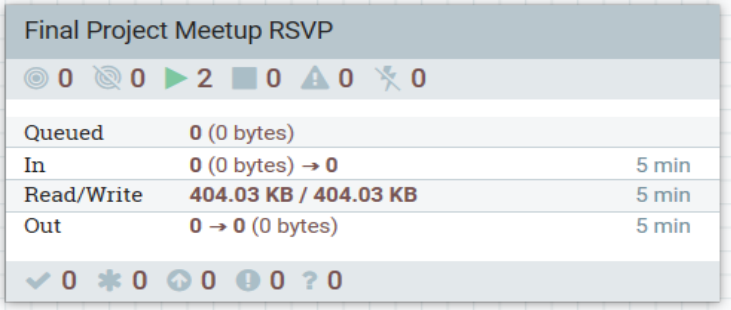
1. Meetup will now allow advertising on their website and app, based on events members plan on attending.
2. As soon as a positive Meetup RSVP is received, the dataflow will determine whether the event meets criteria for advertising.
3. Both the Meetup group's location and keywords will be evaluated against advertiser specifications.
4. Data needed for targeted advertising will then be forwarded to an advertising dataflow.

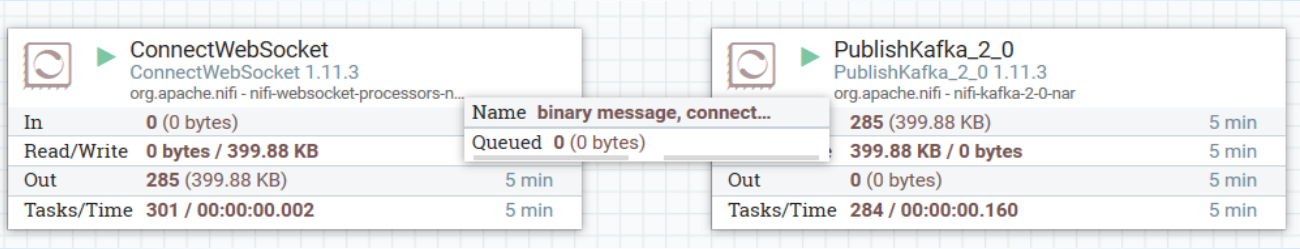
Data Pipeline Flow



NiFi

- Consume Meetup RSVP data stream
- Deliver to Kafka topic without transformation
- Writing own Kafka Producer was an option





Azure Databricks

Apache Spark

- Consume Meetup RSVP stream from Kafka
- Structure JSON into DataFrame columns, using nested schema and explode() for array of RSVP topics

```
jsonSchemaRSVP = (
  StructType()
  .add("venue", StructType()
       .add("lon", DoubleType())
       .add("lat", DoubleType()))
  .add("response", StringType())
  .add("member", StructType()
       .add("member_id", LongType()))
  .add("rsvp_id", LongType())
  .add("mtime", LongType())
  .add("event", StructType()
       .add("event_name", StringType()))
  .add("group", StructType()
       .add("group_topics", ArrayType(StructType()
            .add("topic_name", StringType()))))
)
```

```
meetupDF = (
  meetupRawDF
     .select(from_json("value", jsonSchemaRSVP)
     .alias("meetup"))
      #select individual fields instead of "*" to flatten nested JSON
     .select("meetup.rsvp_id", "meetup.member.member_id", "meetup.mtime", "meetup.response",
             "meetup.event.event_name", "meetup.venue.lon", "meetup.venue.lat",
             "meetup.group.group_topics.topic_name")
).select("rsvp_id", "member_id", "mtime", "response", "event_name", "lon", "lat",
         explode("topic_name").alias("keyword")
  ).filter("response = 'yes'")
```

Apache Spark

- Import static advertiser specification data
- Join with Meetup RSVP data and evaluate for a matching location and keyword

```
[{
  "advertiser": "Databricks",
  "city": "San Francisco",
  "maxlat": 38.522940,
  "minlat": 37.178392,
  "maxlon": -121.763077,
  "minlon": -122.938614,
  "keywords": ["spark","hadoop","big data","machine learning","ai","data pipeline"]
},
{
  "advertiser": "REI",
  "city": "Seattle",
  "maxlat": 47.751501,
  "minlat": 47.471908,
  "maxlon": -122.130405,
  "minlon": -122.428409,
  "keywords": ["hiking","camping","outdoor","adventure","climbing"]
}]
```

```
joinPredicate = "mu.lon between ad.minlon and ad.maxlon AND mu.lat between ad.minlat and ad.maxlat"

meetupAdDF = (
  meetupDF.alias("mu").join(
    advertiserDF.alias("ad"),
    expr(joinPredicate)
  )
)
```

```
meetupAd2DF = (
  meetupAdDF
    .withColumn("adKeywords", explode("keywords"))
    .filter(lower(col("keyword")) == col("adKeywords"))
    .select("rsvp_id", "member_id", "mtime", "response", "event_name", "lon", "lat",
            "advertiser", "city", "keywords", "minlon", "maxlon", "minlat", "maxlat")
    .distinct()
)
```

Apache Spark

Joined Meetup and Advertiser data

event_name	lon	lat	advertiser	city	keywords	minlon	maxlon	minlat	maxlat
KubeFlow +Keras/TensorFlow 2.0 +TF Extended (TFX) +Kubernetes +Airflow +PyTorch	-122.399445	37.788803	Databricks	San Francisco	["spark","hadoop","big data","machine learning","ai","data pipeline"]	-122.938614	-121.763077	37.178392	38.5229

Matching Meetup RSVPs (with Databricks San Francisco)

rsvp_id	member_id	event_name	advertiser	city
1833241628	212641076	KubeFlow +Keras/TensorFlow 2.0 +TF Extended (TFX) +Kubernetes +Airflow +PyTorch	Databricks	San Francisco
1833242400	187579401	KubeFlow +Keras/TensorFlow 2.0 +TF Extended (TFX) +Kubernetes +Airflow +PyTorch	Databricks	San Francisco
1833243534	193660532	Deep Learning (Tensor Flow, DJL and DL4J) for Java Developers	Databricks	San Francisco
1833245145	184104333	Deep Learning (Tensor Flow, DJL and DL4J) for Java Developers	Databricks	San Francisco

Matching data pushed to Kafka

```
bash-4.4# bin/kafka-console-consumer.sh --bootstrap-server ubuntuserver010.westus2.cloudapp.azure.com:9092 --topic meetupad
{"member_id":212641076,"event_name":"KubeFlow +Keras/TensorFlow 2.0 +TF Extended (TFX) +Kubernetes +Airflow +PyTorch","advertiser":"Databrick
s","city":"San Francisco"}
{"member_id":187579401,"event_name":"KubeFlow +Keras/TensorFlow 2.0 +TF Extended (TFX) +Kubernetes +Airflow +PyTorch","advertiser":"Databrick
s","city":"San Francisco"}
{"member_id":193660532,"event_name":"Deep Learning (Tensor Flow, DJL and DL4J ) for Java Developers","advertiser":"Databricks","city":"San Fr
ancisco"}
{"member_id":184104333,"event_name":"Deep Learning (Tensor Flow, DJL and DL4J ) for Java Developers","advertiser":"Databricks","city":"San Fr
ancisco"}
```

Key Conclusions

- NiFi simple to setup, but would have preferred to write my own Kafka producer.
- Spark Structured Streaming as “unbounded table” enables development against static data, which then “magically” works the same against streaming data. Very efficient.
- Interested in developing a time-windowed data use case, but not sufficient time.
- NiFi, Kafka, and Spark allows one to focus on the data manipulations and not what’s happening “under the covers.” There may be good and bad points to this approach, given higher data volume and velocity.