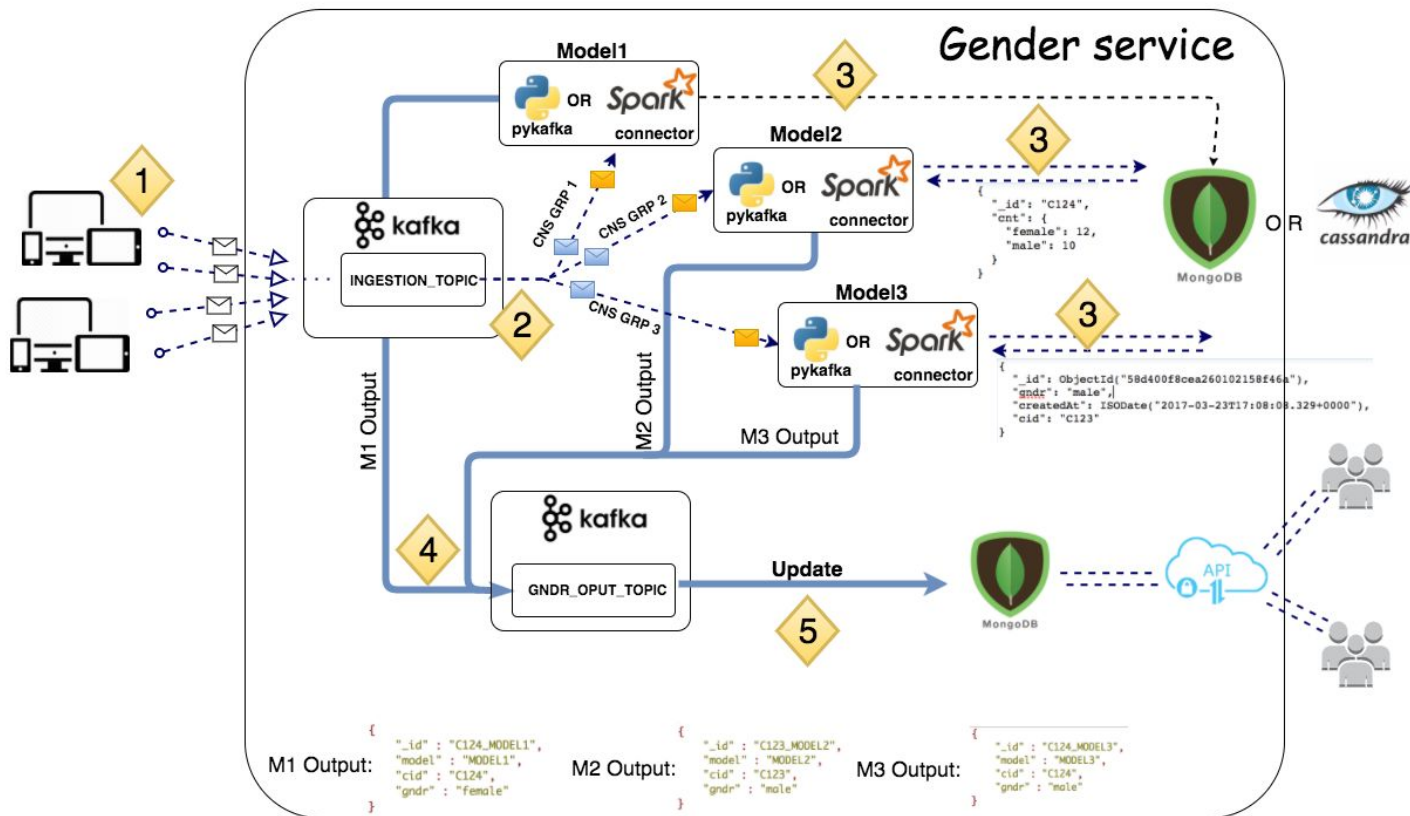


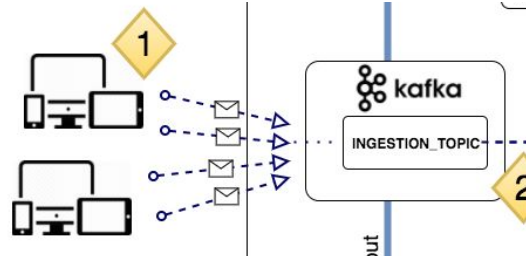
Gender Service Data Pipeline

Hamed Saljooghinejad

Architecture Overview

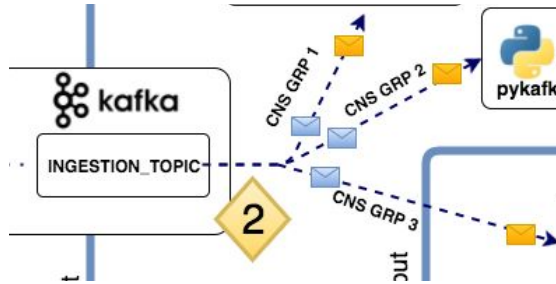


Ingestion:

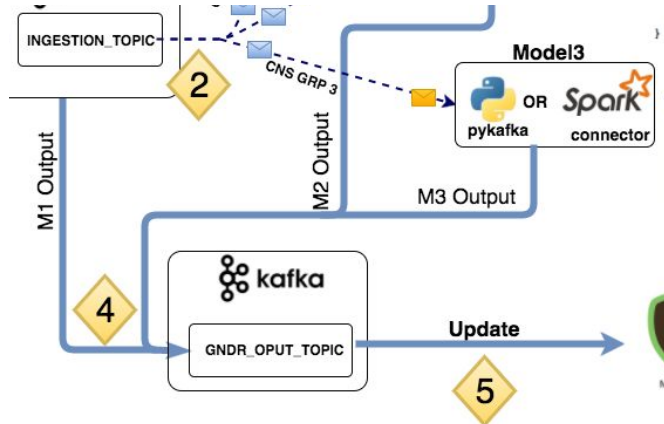


- Ingest from Farfetch app or Farfetch internal web services
- Kafka is massively scalable. There is no limitation on the volume of input data.
- Metrics can be pushed from anywhere/any device Regardless of which programming language is used.
- Preserve the order of input ingested

Consuming message:

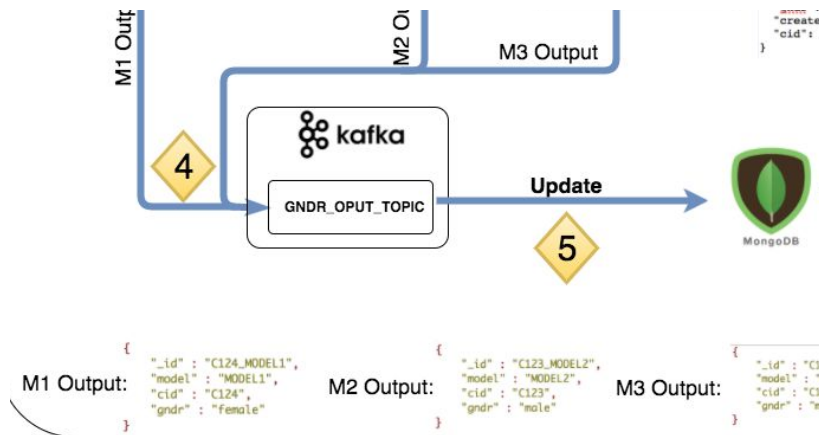


- Each model uses its own consumer group
- Each consumer group gets a copy of ingested input (cid, gender_page_visit)

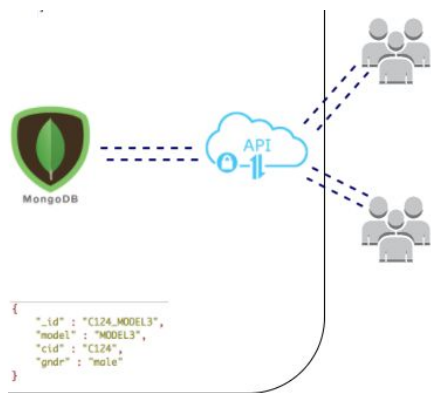


- Each model consumes and calls its own solution function
- By defining multiple partitions for kafka topics each model can run simultaneously as many times as number of partitions to increase parallelism.
- Output returned from each model pushed to kafka output topic(GNDR_OPUT_TOPIC)

Output:



- Each model can asynchronously push into kafka output topic(GNDR_OPUT_TOPIC)
- Update module consumes from output topic and update the target cid_model in database



- API call to request the gender for a cid/model
 - `curl 127.0.0.1:5000/api/v1/getGender/C124?model=MODEL2`
- Query to db(mongo) using _id(cid_model)
- Respond the gender for cid to the user

Model 1 (Last Gender Visit)

- Consume json input:
- Enrich json by adding Model name
- Push to kafka output topics

```
{  
  "cid": "C124",  
  "gndr": "male"  
}
```



```
{  
  "_id" : "C124_MODEL1",  
  "model" : "MODEL1",  
  "cid" : "C124",  
  "gndr" : "female"  
}
```

Model 2(Top Gender Visit)

- Consume json input:
- Send a call to database to increment gender
 - Atomic update to avoid race conditioning
 - Return the output count for each gender
- Enrich json by adding Model name
- Push to kafka output topics

```
{  
  "cid": "C124",  
  "gndr": "male"  
}
```



```
{  
  "_id": "C124",  
  "cnt": {  
    "female": 12,  
    "male": 10  
  }  
}
```



```
{  
  "_id" : "C124_MODEL2",  
  "model" : "MODEL2",  
  "cid" : "C124",  
  "gndr" : "female"  
}
```

Model 3(Top Gender Visit - Last x Days)

- Consume json input:
- Insert entry with CreatedAt field
 - Older Records deleted after TTL time expires
- Aggregate the genders for Cid
- Pick gender with higher value

```
{  
  "cid": "C123",  
  "gndr": "male"  
}
```



```
{  
  "_id": ObjectId("58d400f8cea260102158f46a"),  
  "gndr": "male",  
  "createdAt": ISODate("2017-03-23T17:08:08.329+0000"),  
  "cid": "C123"  
}
```



- Enrich json by adding Model name
- Push to kafka output topics

```
{  
  "_id" : "C123_MODEL3",  
  "model" : "MODEL3",  
  "cid" : "C123",  
  "gndr" : "male"  
}
```


Base Class - Model

- Previous models inherited from this base model
- Has a function called Dispatch that uses template design pattern to define common steps for each subclass(Previous slides).

```
for gndr_entry in self.consume():
```

→ Consume kafka

```
    rslt_entry = self.identify_gender(gndr_entry)
```

→ Call the solution function

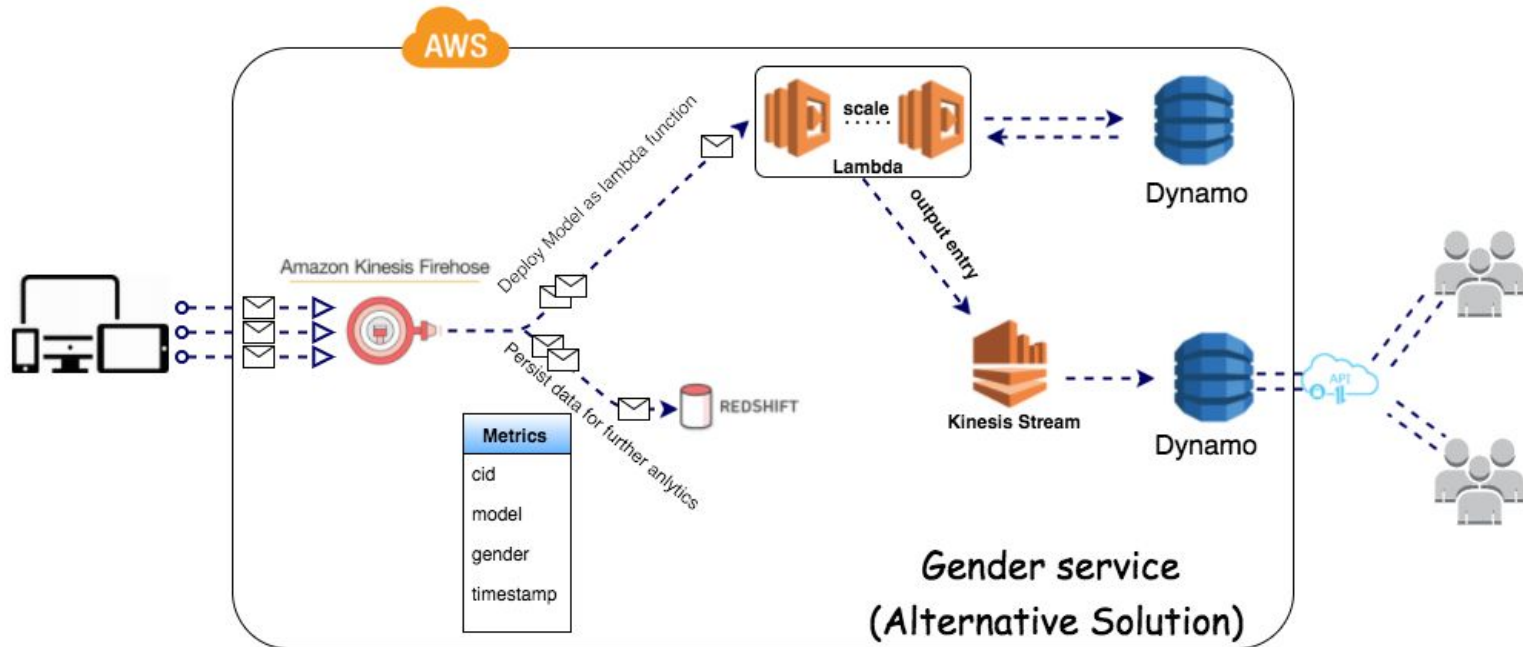
```
    self.prod.produce(rslt_entry)
```

→ Push output to kafka output topic

- Abstract function called `identify_gender` to be implemented in subclass
 - Easy to test any new solution (Data scientist need to just implement `identify_gender` function)
 - Easy to test new model as it reads from new consumer group/push the result with it own model name

Alternative Architecture:

- Based on my past experience, we can rely entirely on AWS infrastructure to build this service as well.



Advantages:

- AWS will manage the infrastructure. Less administration hassle for different applications like kafka, spark and MongoDB/Cassandra
- Lambda function is elastically scalable. The higher the ingested input metrics the more function calls by lambda service.
- Easier deployment: Once the function uploaded in lambda and firehose start listening. Ready to go.
- Lambda Function gets triggered automatically as soon as a new metric Ingested into kinesis firehose.

Notes:

- Code is also available here:
 - https://github.com/hamedhsn/gender_pipeline
- Code is fully functional and I have tested it (please look at the docs/doc.pdf - section 'how to run it')
- For more details please look at the document file. It is a 5 page document explaining about the project. It is available in repository under docs/doc.pdf)