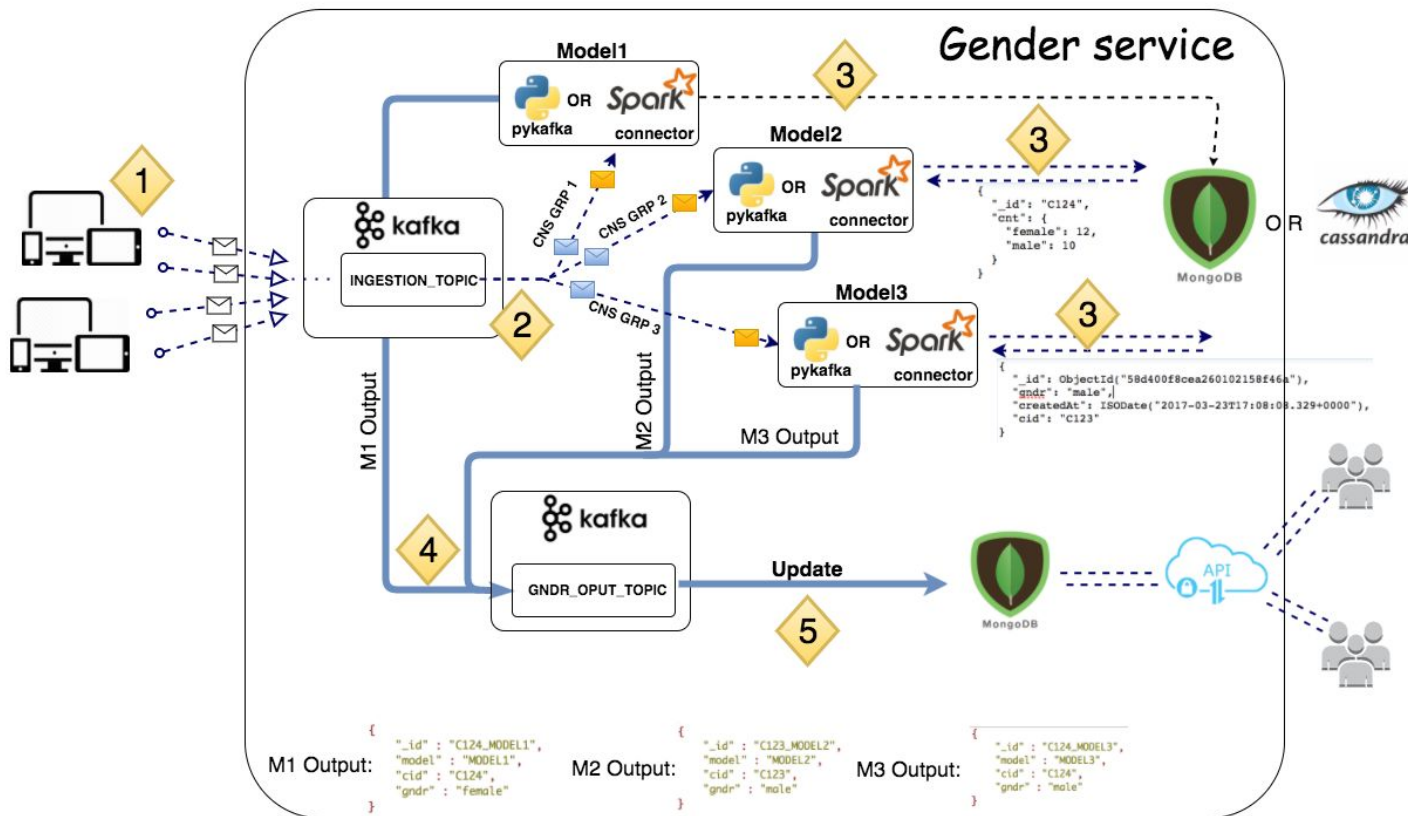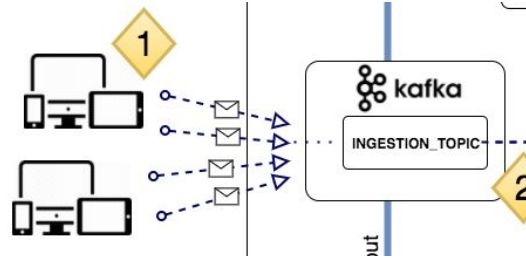# Gender Service Data Pipeline
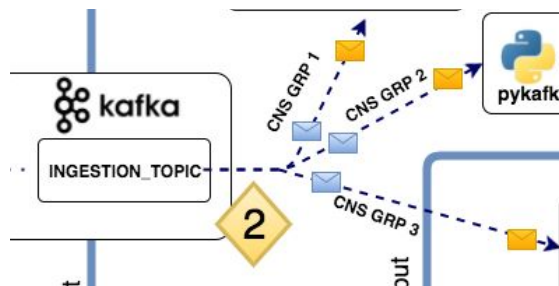
Hamed Saljooghinejad

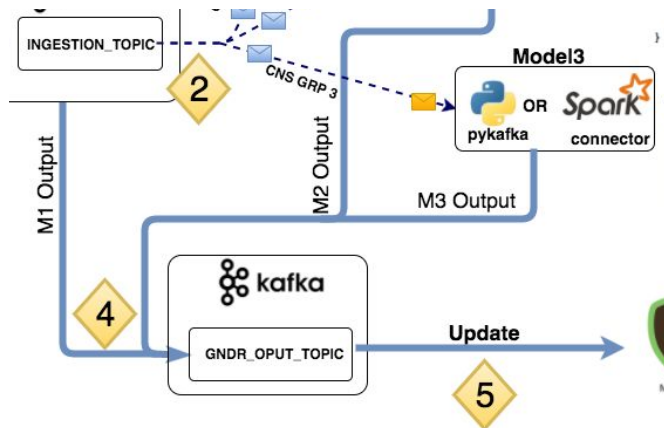# Architecture Overview

Ingestion:

- Ingest from Farfetch app or Farfetch internal web services
- High Scalability for ingestion
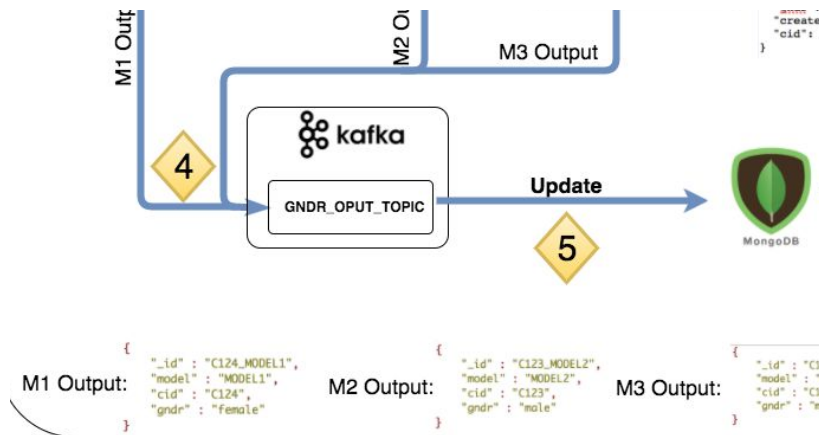- No bind on specific language for ingestion

Consuming message:

- Each model uses it 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
- To scale multiple consumers can consume from different kafka partition
- Output returned from function pushed to kafka output topic(GNDR_OPUT_TOPIC)

Output:

M1 Output:
```
{
    "_id" : "C124_MODEL1",
    "model" : "MODEL1",
    "cid" : "C124",
    "gndr" : "female"
}
```

M2 Output:
```
{
    "_id" : "C123_MODEL2",
    "model" : "MODEL2",
    "cid" : "C123",
    "gndr" : "male"
}
```

M3 Output:
```
{
    "_id" : "C1
    "model" : "
    "cid" : "C1
    "gndr" : "m
```



```
{
    "_id" : "C124_MODEL3",
    "model" : "MODEL3",
    "cid" : "C124",
    "gndr" : "male"
}
```

- Each model asynchronously can push to kafka output topic(GNDR_OPUT_TOPIC)
- Consume output topic
- Update cid_model in database
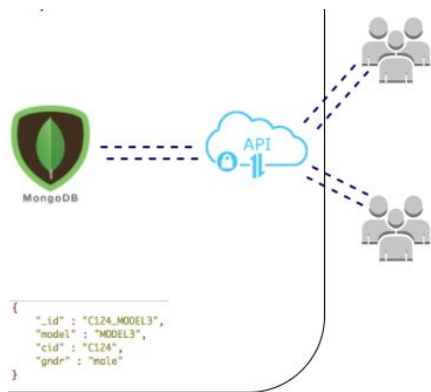
- API call to request the gender for cid/model
  - **curl 127.0.0.1:5000/api/v1/getGender/C124\?model=MODEL2**
- Query to db(mongo) using _id(cid_model)
- Respond to user

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 condition
  - 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)

- Consume json input:

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

- Insert entry with CreatedAt field
  - Older Records deleted after TTL time expired

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

- Aggregate the genders for Cid
- Pick gender with higher value

- 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).

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

    rslt_entry = self.identify_gender(gndr_entry)

    self.prod.produce(rslt_entry)
```

→ Consume kafka

→ Call the solution function

→ 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 relying entirely on AWS infrastructure to build this service as well.

Gender service
(Alternative Solution)
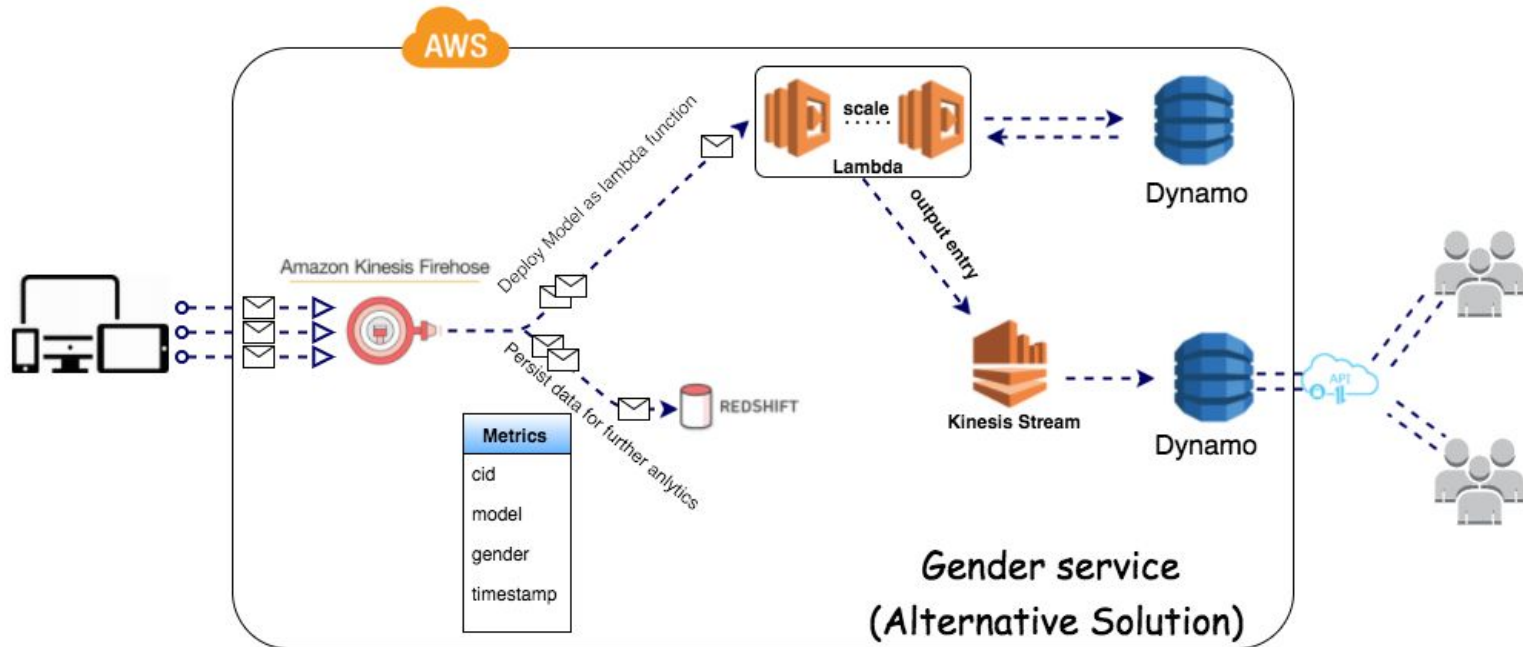
Advantages:

- 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)