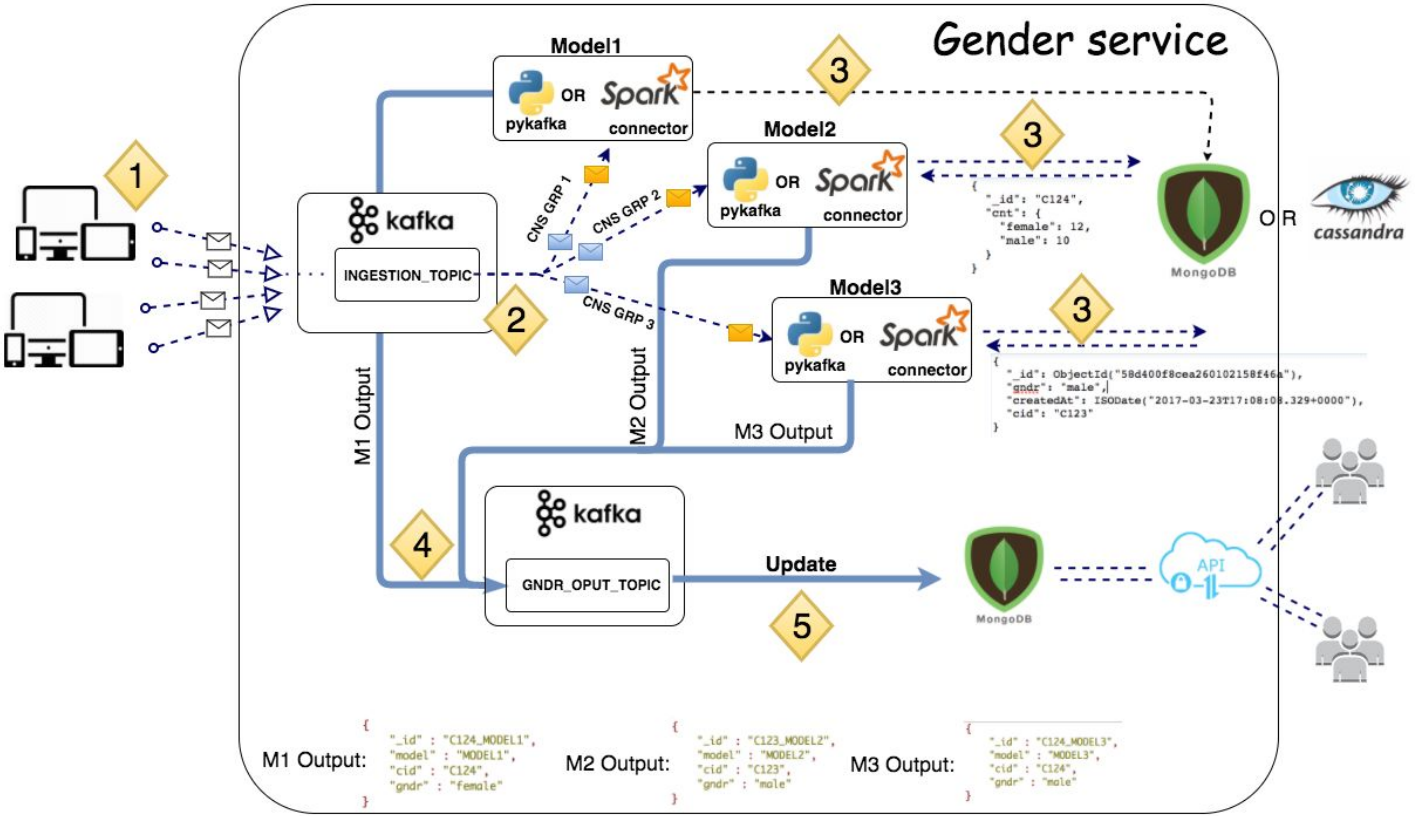
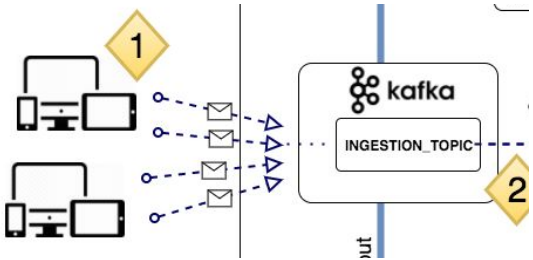
# Gender Service Data Pipeline

Hamed Saljooghinejad

# Architecture Overview

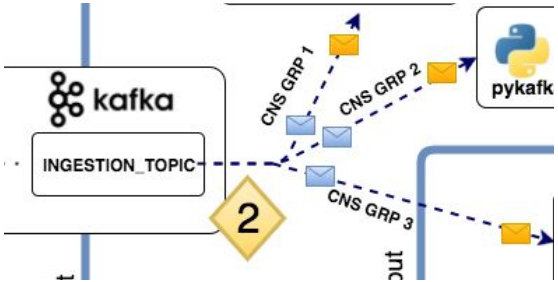


# Ingestion:



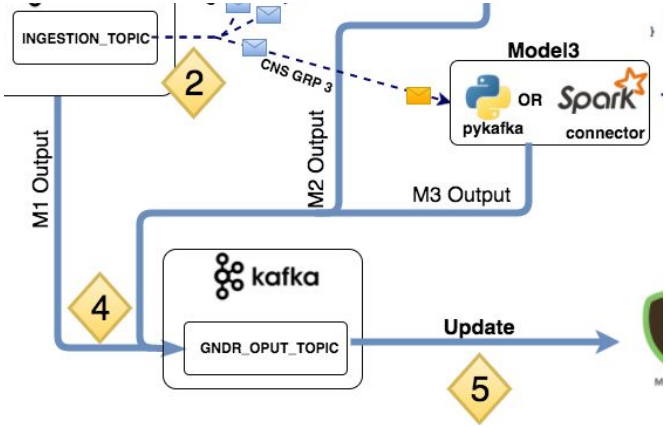

- Ingest from any app or internal web services
- Kafka is massively scalable. There is no limitation on the volume of input data.
- Metrics  can be pushed from anywhere/any device Regardless of which programing language is used.
- Preserve the order of input ingested

# Consuming message:



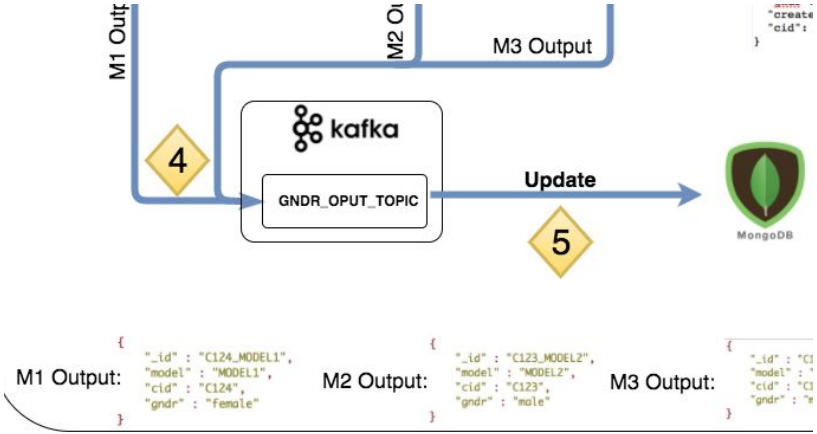

- Each model uses it own consumer group
- Each consumer group gets a copy of ingested input (cid, gender_page_visit)

---

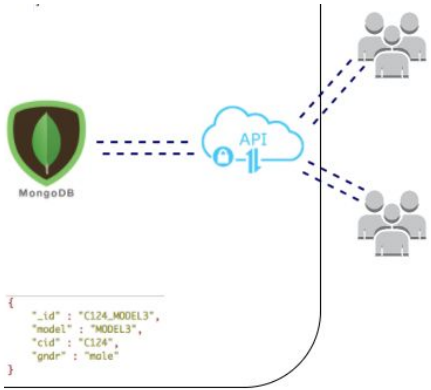



- Each model consumes and calls its own solution function
- By defining multiple partitions for kaka topics each model can run simultaneously as many times as number of partitions to increase parallelism.
- Output returned from each model pushed to kafka output topic(GNDR_OPUT_TOPIC)

# 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



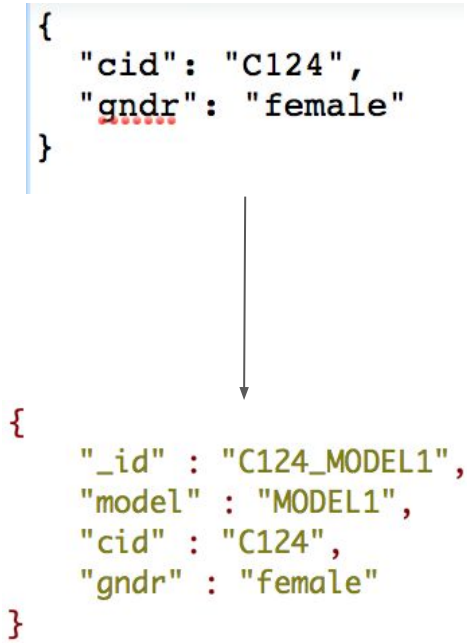

- API call to request the gender for a cid/model
  - **curl 127.0.0.1:5000/api/v1/getGender/C124\?model=MODEL2**
- Query to db(mongo) using _id(cid_model)
- Respond the gender for cid to the user

# Model 1(Last Gender Visit)

- Consume json input:

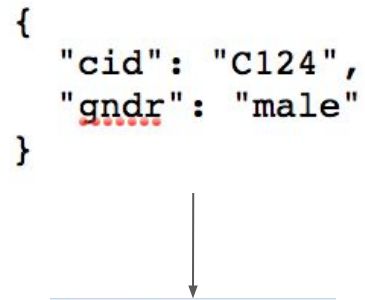
- Enrich json by adding Model name

- Push to kafka output topics

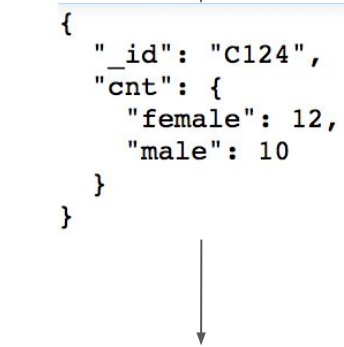


# Model 2(Top Gender Visit)

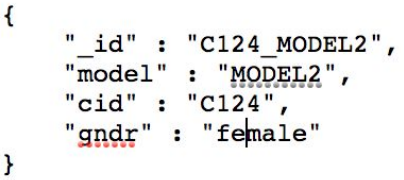
- Consume json input:




- Send a call to database to increment gender
  - Atomic update to avoid race conditioning
  - Return the output count for each gender
  - Pick the max gender

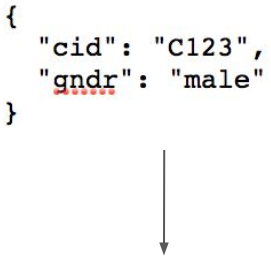



- Enrich json by adding Model name
- Push to kafka output topics

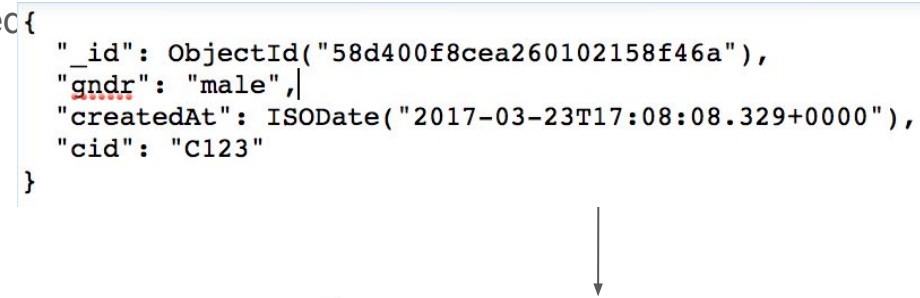


# Model 3(Top Gender Visit - Last x Days)

- Consume json input:




- Insert entry with CreatedAt field
  - Older Records deleted after TTL time expired
- Aggregate the genders for Cid
- Pick gender with higher value




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

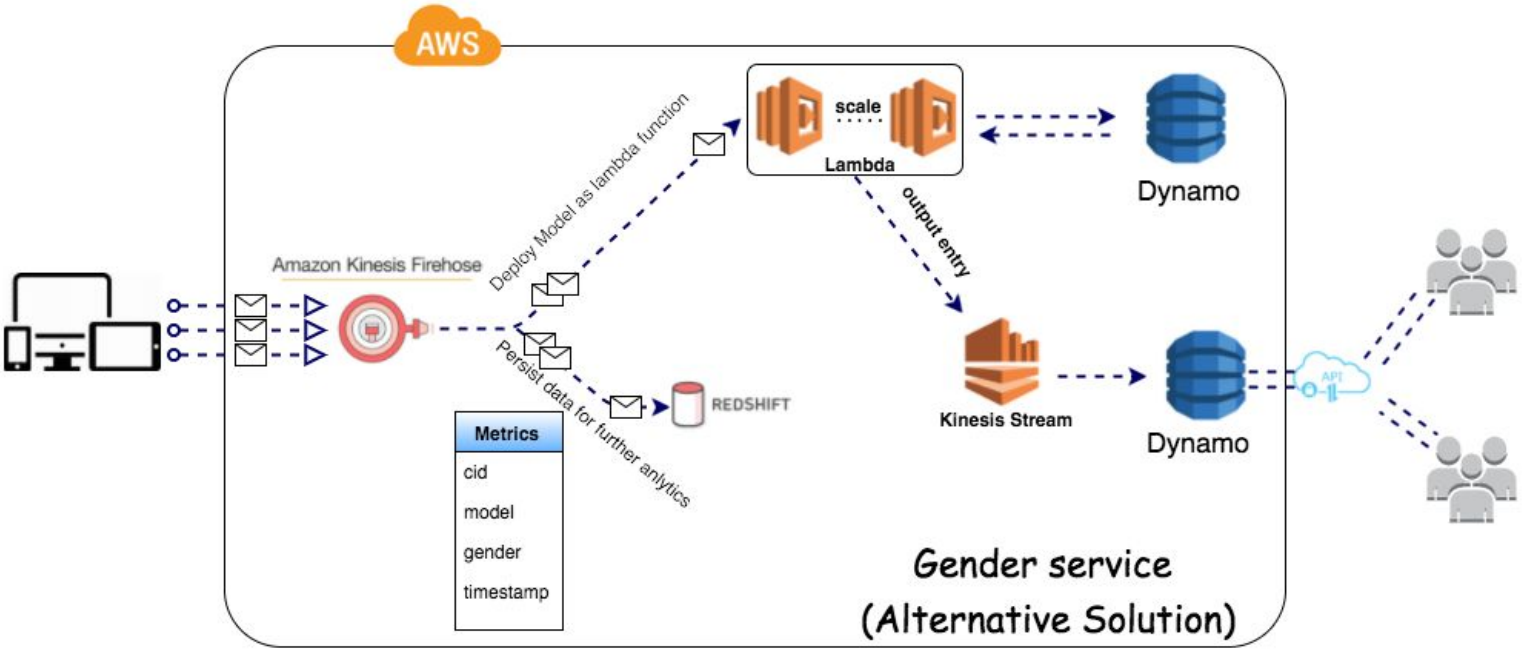
# Base Class - Model

- 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).


- Apply pre-enrichment of consumed msg inside self.consume using generator
- 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.



# 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)