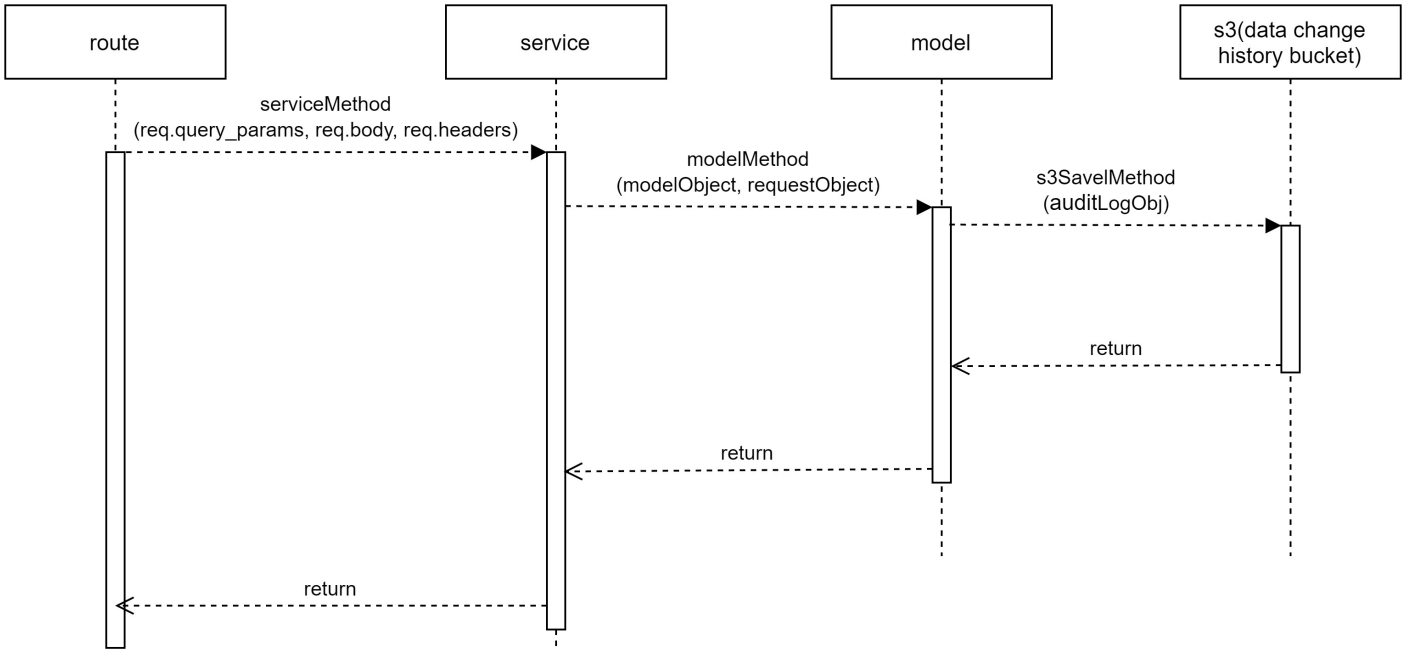
Intro

- Change data capture is a concept where we track data changes occurs at each transaction in row/document for each table/collection.
- Always answer **what**, **when**, **who**. Like a newspaper article, not a book with a lot of discourse but the right amount of information to have context.
    - **WHAT:** The action that happened, a user insert or updated row or documents. We need to what columns in the row or attributes in the document are modified.
    - **WHEN:** Crucial aspect of the audit/history table, a time stamp (date and time) of when the event happened. It should be utc timestamp or it can be "**ud**" in row/document
    - **WHO:** Custom application level or domain entity id who acts as the performer of the action. Might be the system, a user or another third party application. In our case it will be "user_id" which will be available in jwt token in header of each request.
    - **CORRELATION ID;** Without this, it's not an audit. Everything you audit needs to have a correlation id. A correlation id is a way to identify how does an event relates to a request/action/requirement. In this case it is **"request-id**" in header which will come from upstream services or if request in originating from service it will "**UUID**"

Collecting the **aduitLogObject:**

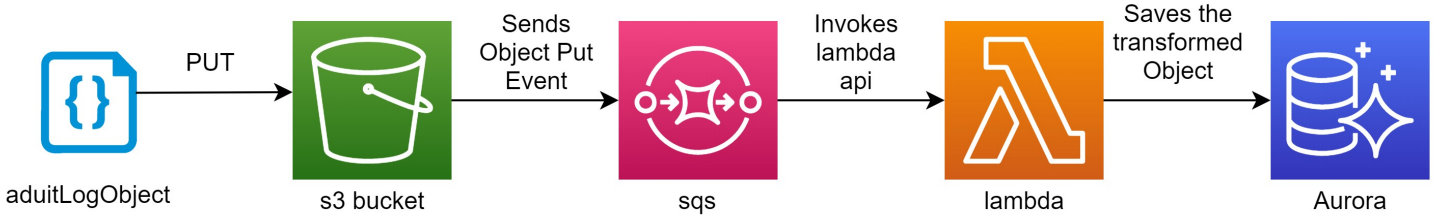


- These are the sequence of steps for any create/update/delete operation apis.
    - Route: Incomming request will be handled by route module then it passess "req.query_params", "req.body", "req.header" to service.
    - Service: Service will impliment the application logic or bussiness logic and then calls modelMethod to create/update/delete the entity
    - Model: Model will intract with database server and performs the create/update/delete and after successfull operations then it will get will current data and previous whole data along with requestObj this will be passed to s3 save
    - s3_save(async): This will put the auditLogObject into s3.

- aduitLogObject example object


```
{
  "table_id": "products_master",
  "entity_id": "1",
  "micro_service_name":"product_mangement",
  "action":"update",
  "old_data": {
    "_id": 1,
    "title": "Redmi note 9 max | 6gb Ram | 128gb Storage",
    "sku_id": "sku_1",
    "inventory": 1
  },
  "new_data": {
    "_id": 1,
    "title": "Redmi note 9 max | 4gb Ram | 128gb Storage",
    "sku_id": "sku_1",
    "inventory": 10
  },
  "request_info":{
    "time_stamp":"2020-07-29T19:45:00.047Z",
    "ip_address":"203.109.108.86",
    "request_id":"a20e389c-134b-40e7-9827-1c94eb895412",
    "user": {
      "account_id":"eb257dff-dddb-482d-afb0-5934ace42c13",
      "user_id":"206483f8-f11e-4ee8-87df-962b6f92d610",
      "email": "pavan@eunimart.com",
      "user_name":"Pavan Kumar"
    },
    "miscellaneous": {
      "user_agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/84.0.4147.89 Safari/537.36"
    }
  }
}
```


Processing the **aduitLogObject:**



- In our system we will follow these sequence of steps to impliment this concept:
  - **aduitLogObject**: An "aduitLogObject" is json file which contains old_data, new_data and user_info will be put in to s3 bucket.

- **s3 Bucket**: In s3 bucket we will configure to emit a event when there is a put operation then we willl configure s3 to send this event as a new message to sqs.
- **SQS:** Once sqs receives a message it will invoke configured lambda function and if fails to deliver the message then it will retry to deliver for max number of retires.
- **Lamda**: Lambda function will be calculating diff of old_data and new_data and creating a new payload. This new payload is then stored to Aurora(MySQL).
    - NOTE: For every attribute change a new entity will be created after transformation.


```
[
    {
        "date":"2020-07-29",
        "time":"19:45:00",
        "table_id":"products_master",
        "entity_id":"1",
        "microservice_name":"product_mangement",
        "action":"update",
        "request_id":"a20e389c-134b-40e7-9827-1c94eb895412",
        "ip_address":"203.109.108.86",
        "account_id":"eb257dff-dddb-482d-afb0-5934ace42c13",
        "user_id":"206483f8-f11e-4ee8-87df-962b6f92d610",
        "user_email":"pavan@eunimart.com",
        "user_name":"Pavan Kumar",
        "attribute":"title",
        "previous_value":"Redmi note 9 max | 6gb Ram | 128gb Storage",
        "current_value":"Redmi note 9 max | 4gb Ram | 128gb Storage",
        "browser_name":"chrome",
        "device_type":"web",
        "os":"windows 10"

    },
    {
        "date":"2020-07-29",
        "time":"19:45:00",
        "table_id":"products_master",
        "entity_id":"1",
        "microservice_name":"product_mangement",
        "action":"update",
        "request_id":"a20e389c-134b-40e7-9827-1c94eb895412",
        "ip_address":"203.109.108.86",
        "account_id":"eb257dff-dddb-482d-afb0-5934ace42c13",
        "user_id":"206483f8-f11e-4ee8-87df-962b6f92d610",
        "user_email":"pavan@eunimart.com",
        "user_name":"Pavan Kumar",
        "attribute":"inventory",
        "previous_value":"1",
        "current_value":"10",
        "browser_name":"chrome",
        "device_type":"web",
        "os":"windows 10"
    }
]
```

- **Aurora**: Aurora is severless database and we will using MySQL engine to store our database.

Querying the change history:

- For querying the change history we will impliment another lambda function and expose it through API Gateway