# Title: Deals forecasting-pipeline

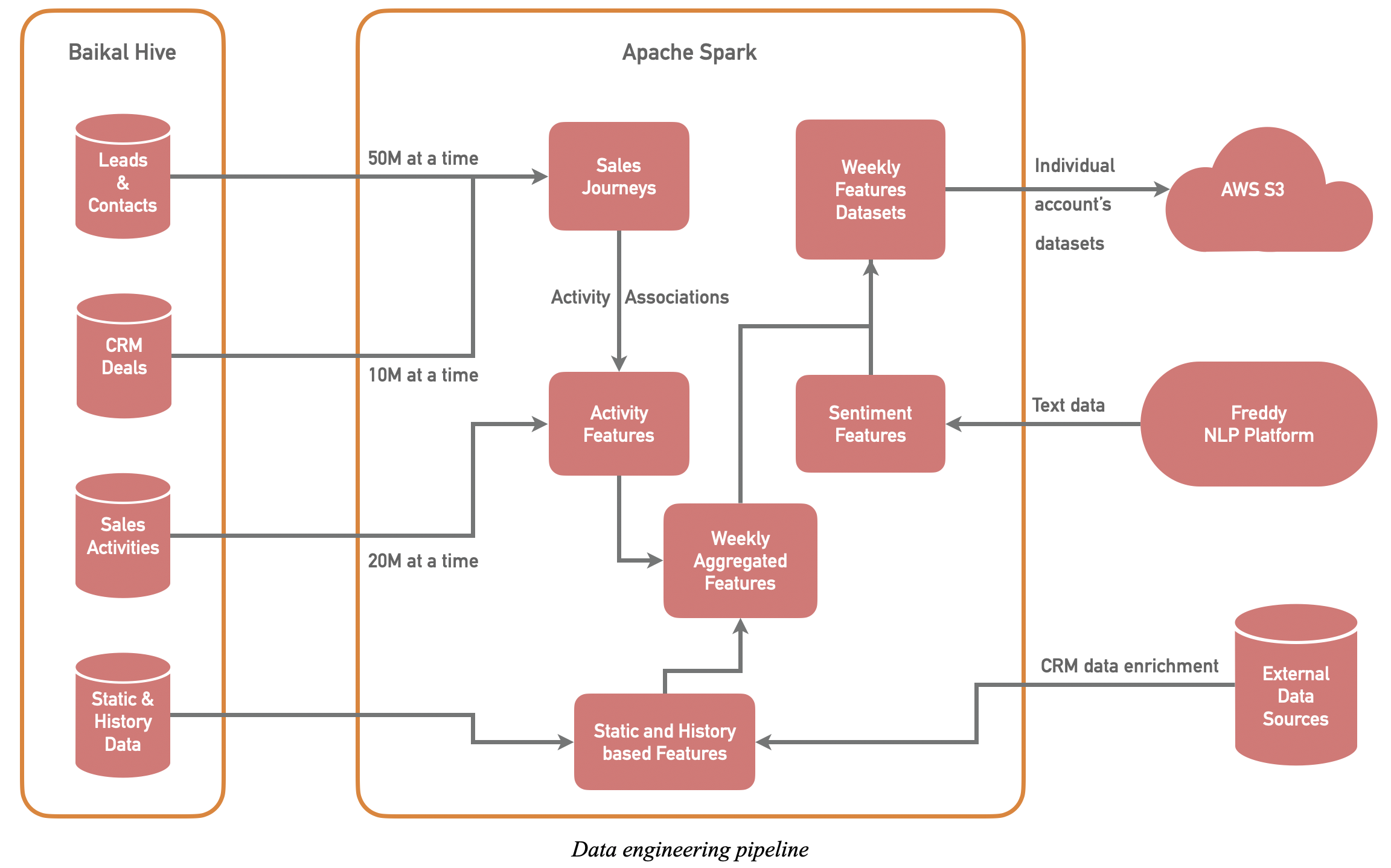
Thousands of businesses are using Freshworks CRM, to manage their end-to-end sales processes, including identifying new customers, pitching their products and services, negotiating a sale, and closing deals. Given that sales operations directly impact inventory management, manufacturing, and several other aspects of a business, companies need to plan upcoming sales activities well ahead of time. Sales, anticipated well in advance, can help a business optimize its GTM operations and build an efficient sales pipeline.

Sales forecasting is a crucial problem in CRM and we tried to address this problem by predicting the closure of sales deals. By analyzing sales deals closed in a specific duration and the value of each sales deal, we can further determine the overall revenue forecast for CRM users. This document depicts the overall architecture and setup instructions of deals forecasting feature in Freshworks CRM.

## Architecture

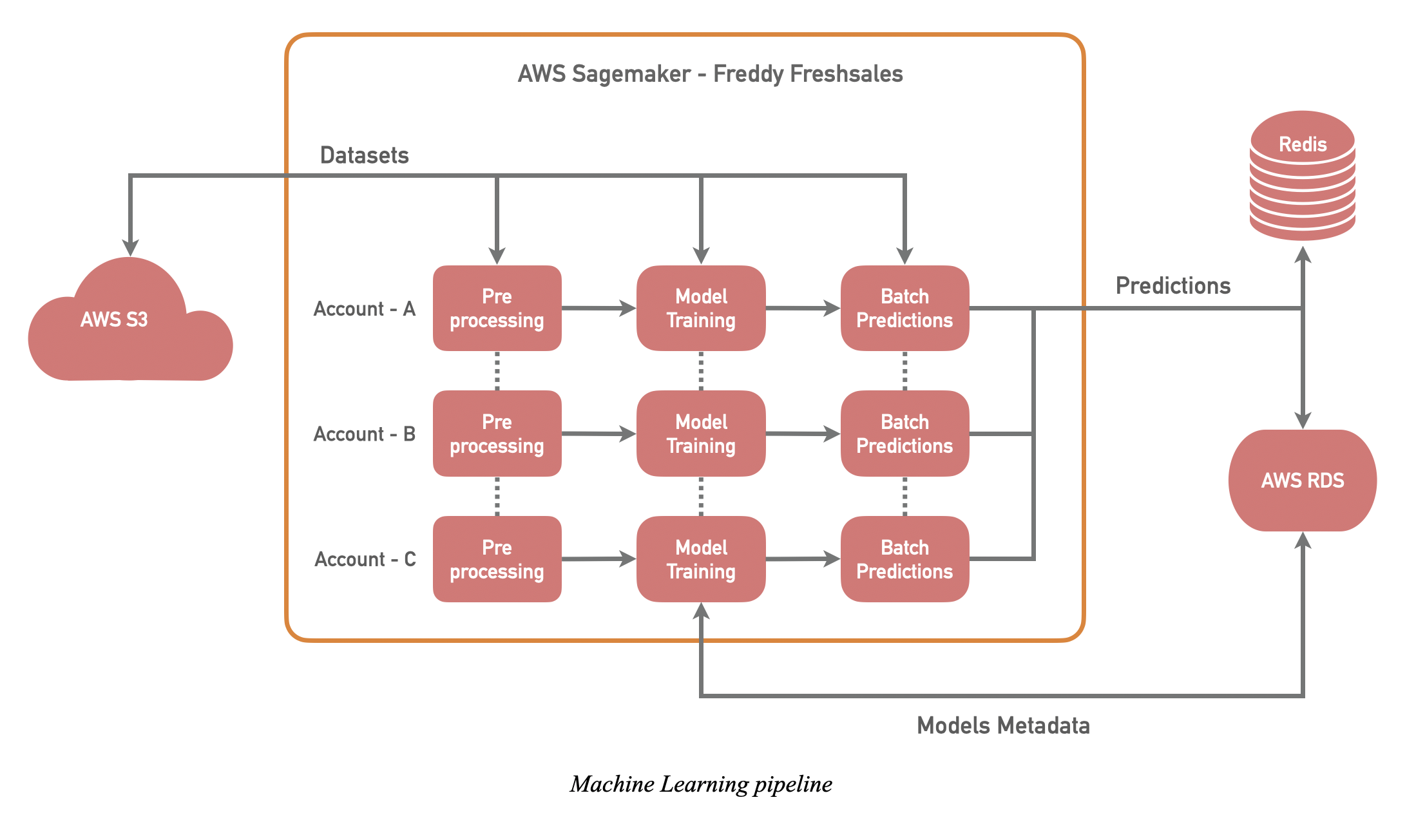
#### Data engineering in Baikal:

* Freshsales\_pvt database in Baikal Hive is used as the primary source of data for deals forecasting. This DB contains information about sales accounts, leads, contacts, deals, sales activities, emails metadata etc.
* At an initial stage, Spark retrieves leads and contacts associated with deals and their active time periods. These entities are combined together to form sales journeys.
* Activities in sales journeys are fetched from DB, refined and filtered in spark pipeline, and then aggregated over every week. Deal progression features like acceleration and age features are derived from these.
* Activity counts are combined with static features of deals and sales accounts. Email-based features are also combined with this to form final weekly features.
* Deals, sales journeys, sales activities, and final sales features are written to the S3 bucket under the 'datasets' directory. This directory is created every week, so we have data for every week in S3 at all times.



#### Machine Learning using AWS Sagemaker:

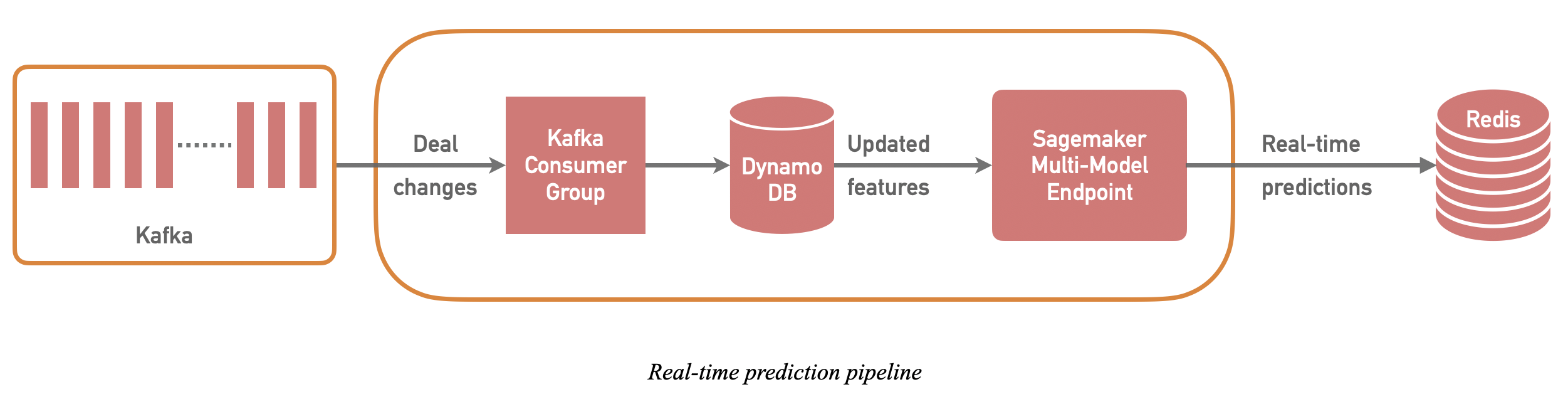
* Spark pipeline writes data to S3 every week and AWS lambda triggers ML pipeline every time this dataset is created.
* The pipeline reads all the accounts in datasets and fetches account-level settings required for ML preprocessing, training and prediction.
* Every account's data undergoes preprocessing then training and then prediction job in AWS Sagemaker. All accounts' overall processes are run concurrently to speed up the whole ML pipeline duration.
* Once batch prediction jobs are completed, these results are dumped to RDS and valid predictions are sent to Redis (handled by the FS engineering team).



#### Real-time prediction pipeline:

As soon as sales agents make progress in a deal, Freddy's forecasting reflects an immediate change so that our users can make quick decisions to achieve their sales targets. The following steps outline the overall process of real-time predictions.

* Deals’ weekly features are copied from S3 to DynamoDB to handle high scalability and low-latency retrieval of data for real-time processing.
* Activities performed by sales agents and corresponding changes in deals are captured in central Kafka. These payloads in Kafka topics are consumed and deal features in the DynamoDB are updated accordingly.
* All account’s ML-models are deployed in AWS Sagemaker as a multi-model endpoint. As soon as a feature is updated, this endpoint sends a new prediction to the CRM.



## DB Schema

**Jobs:**

| Column | Data type | Description |
| --- | --- | --- |
| id | int | Unique id for every call to ML process |
| run\_id | int | Id for every ML pipeline job |
| deploy\_mode | bool | 1 if the output is sent to Redis, otherwise 0 |
| start\_time | timestamp | The time when the job is started |
| end\_time | timestamp | The time when the job is completed. In case of errors: Null. |

**Models:**

| Column | Data type | Description |
| --- | --- | --- |
| id | int | Unique id for every ML model |
| account\_id | int | Unique id for every CRM account |
| model\_name | string | Name of the model in sagemaker |
| modeling\_date | date | Modeling date (Monday) as per data used for ML job |
| run\_id | int | Id for every ML pipeline job |
| threshold | float | Threshold for prediction |
| create\_time | timestamp | The time when the model is created |

**Predictions:**

| Column | Data type | Description |
| --- | --- | --- |
| id | int | Unique id for every experiment |
| model\_id | int | Model corresponding to the prediction |
| account\_id | int | Account corresponding to prediction and model |
| deal\_id | int | Deal corresponding to prediction |
| output | float | Prediction output created using sagemaker |
| threshold | float | Threshold for prediction |
| run\_id | int | Id for every ML pipeline job |
| create\_time | timestamp | The time when the prediction is created |

**closed\_deals:**

| Column | Data type | Description |
| --- | --- | --- |
| account\_id | int | Account corresponding to deal |
| deal\_id | int | Unique id for a sales deal |
| journey\_end\_date | date | Closure date for the deal |
| forecast\_type | string | Open or Closed Won or Closed Lost |
| create\_time | timestamp | The time when deal is added to DB table |
| update\_time | timestamp | The time when deal's data is updated in the DB table |

## How to deploy deals forecasting in Baikal?

Requirements: Git, DB, S3, cluster accesses in Baikal gateway instances.

1. Login to freddyfs\_team user and clone GIT repo in $HOME.
2. Copy ~/resources/cron/deal\_forecasting\* and ~/resources/fs\_forecasting/\* from ffs02 and modify files under ~/resources/fs\_forecasting/ as per the region.
3. Download, install anaconda. Refer to setup/pyspark\_env.sh for these further steps.
4. Create conda env for fs\_forecasting, install python dependencies and create archive and symlinks for spark virtual env.
5. Create keytab file, verify if it's working.
6. Create crontab as per ffs02.

## How to deploy deals forecasting in AWS?

### Weekly run

Requirements: Stack, layer, instance, the app should be deployed before these steps. Access to DB, Redis, Kafka from the instance.

1. Refer to setup/lambda\_setup and create two lambda functions as mentioned. Create S3 triggers via console and not using cmd.
2. Login to the instance and setup aws configure with the appropriate region.
3. Create a database in MySQL and create tables using commons/schema.sql.
4. Verify Redis access via setup/integration.py.

### Realtime run

TODO

## How to execute weekly spark job:

cd ~/fs\_forecasting  
nohup bash run\_spark.sh mode=train output=s3 user=freddyfs\_team keytab=freddyfs\_kt.txt > spark.log 2>&1 &

This starts a spark job as follows:

spark2-submit  
       --principal freddyfs\_team  
       --keytab freddyfs\_kt.txt  
       --master yarn  
       --deploy-mode cluster  
       --num-executors 30  
       --executor-cores 4  
       --executor-memory 12G  
       --driver-memory 8G  
       --conf spark.dynamicAllocation.enabled=False  
       --conf spark.yarn.appMasterEnv.PYSPARK\_PYTHON=forecasting\_env/forecasting/bin/python  
       --archives forecasting.zip#forecasting\_env  
       --py-files commons.zip,feature\_engg.zip,config/features.txt  
       feature\_engg/app.py  
       --mode train  
       --database freshsales\_pvt  
       --bucket deals-forecasting

## How to execute weekly sagemaker pipeline:

cd ~/fs\_forecasting  
nohup bash run\_spark.sh mode=15 > sagemaker.log 2>&1 &

This starts sagemaker pipeline python job as follows:

python modeling/pipeline.py --mode 15

'mode' argument in the above command determined steps that sagemaker pipeline would take. This argument is a 4-bit integer where each bit is for Redis, prediction, training, preprocessing from right to left.

So for only training job, mode value should be 2. For only preprocessing and prediction job, the mode should be 5.

Besides, mode = 0 means it's tuning mode and mode = 16 means it's regression testing mode. So 0 <= mode <= 16.