Architecture Hierarchy

- To implement ML based system on top of micro service architecture, Docker tech. were used in this project.
- We have three different services, such as analytic layer, presentation layer, caching layer.
- Each layer is using different required libraries.

README

- You can use docker-compose command to launch those three micro services
- After you create three different container, you can use simple shell scripts (predict.sh, predict_random.sh)
  - predict.sh: It is sending features to our service and receiving response.
  - predict_random.sh: It is picking a row over validation set, randomly and then testing it our ML model under resources directory
- Under resources directory, you can find our final pickle files, and dataset files.
- To take a look at data processing stage, you can go to directly notebook/code/model.ipynb file, or
- You can open it on your browser using the ip address (http://localhost:8888/?token=<token>) of notebook.
- Please, note that you need to replace <Token> field with your specific token you have in the address.

Data Analytic Layer

- In this layer, we are applying any kind of approaching via Jupyter Notebook.
- Its required libs:
  - Scipy, numpy, pandas, scikit-learn, statsmodels, spacy, ipykernel, jupyter, matplotlib, seaborn, gensim, tensorflow, keras, redis

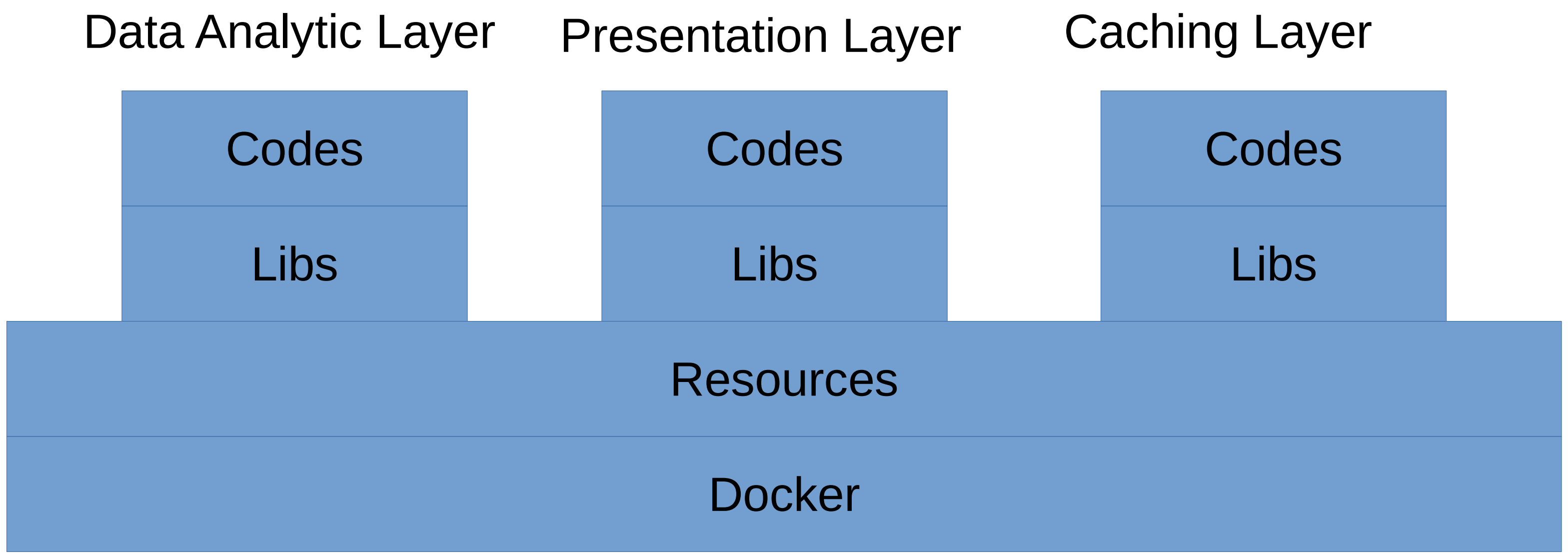
Presentation Layer

- In this layer, we implemented REST API to use model we built in data analytic layer.
- Its required libs:
  - scipy
  - numpy
  - pandas
  - scikit-learn
  - flask
  - redis

Caching Layer

- We can receive same queries through REST API. To improve performance, we can keep the result we have already queried before in the caching.
- In this layer we are using Redis. We can use another NoSQL framework.
- Its required libs:
  - image: "redis:alpine"

Overview of Architecture


Data Analytic Layer
Presentation Layer
Caching Layer
Codes
Libs
Codes
Libs
Codes
Libs
Resources
Docker