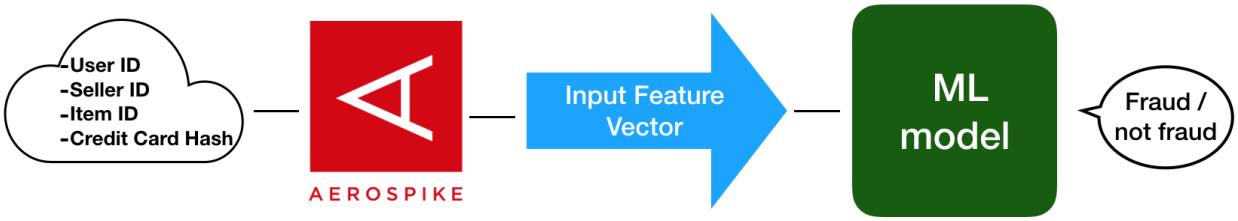
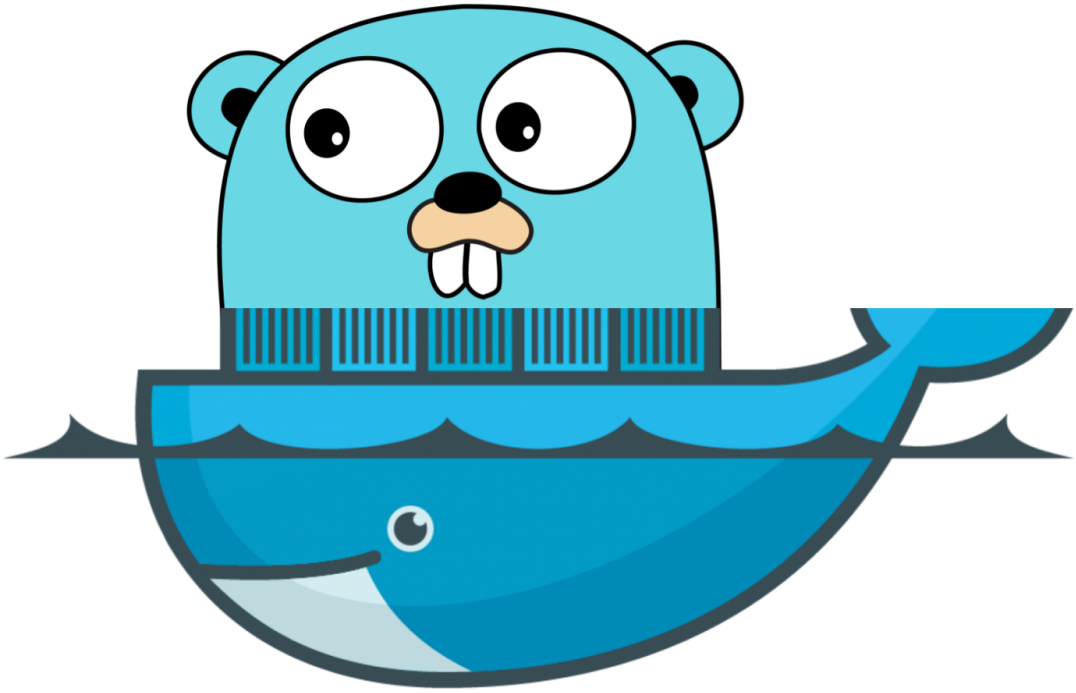
Tools Preparation Before the Workshop

- Watch this talk. This is the opening talk I give to put this workshop into context.
  It's 40 min long, so you might not be able to fit it in your schedule - it's ok, it's not a prerequisite.
- Install and set up GitHub set up on your machine
  - Clone the scaffold repo
    `git clone git@github.com:aerospike-examples/fraud-ml-workshop.git`
- Install Go 1.16
  - Complete the Tour of Go to get familiarized with the language basics
- Install latest Docker
- Pull latest Docker images:
  - `docker pull aerospike/aerospike-server`
  - `docker pull aerospike/aerospike-tools`
  - `docker pull tensorflow/serving`
- Download this dataset
- Format it using the script.
  Here are detailed instructions for formatting it:
  - From the downloaded dataset you need the CSV format which is a file named *creditcard.csv*
  - Go to the folder into which you cloned the repo
  - Copy the file *creditcard.csv* into the folder *hints*
  - Make sure the file *creditcard.csv* is on the same level (same folder) as the file named *script* and from that folder, using a terminal, run the command `bash script`
  - You will get an output named *creditcard.aerospike.csv*
    This is what we'll need in the workshop


-User ID
-Seller ID
-Item ID
-Credit Card Hash
AEROSPIKE
Input Feature Vector
ML model
Fraud / not fraud



Setup the Tools

Docker

Official images repo https://hub.docker.com/r/aerospike/ but we'll use the private one.

Start aerospike-server

```
$ docker run --rm -tid --name aerospike -p 3000:3000 -p 3001:3001 -p 3002:3002
-p 3003:3003 aerospike/aerospike-server
```

Check out to see the images you have:

```
$ docker image ls
aerospike/aerospike-server   4.9.0.3             6722c76dcc71        7 days ago
136MB
```

Start aerospike-tools

In the tools package lives the AQL and asloader tools which we use to communicate with the server

```
$ docker run -it --name aerospike-tools aerospike/aerospike-tools aql -h
<MY IP>
(-it interactive terminal)
```

or

```
$ docker run -it  --name aerospike-tools aerospike/aerospike-tools aql -h
$(docker inspect -f '{{.NetworkSettings.IPAddress }}' aerospike)
```

```
$ docker ps
```

Let’s Get to Work

1. Fill the database with data

Get familiar with AQL

Now we can start using AQL!
A tool for examination of the DB and lightweight maintenance tasks

Let’s see what do we have:

```
aql> show namespaces
```

You can think of sets as tables. Let’s see what’s in the default:

```
aql> show sets
```

Adding an entry as a primary key to set - creates it:

```
aql> insert into test.<setname> (PK, <key-name>) values ('<key>', '<val>')
```

E.g.

```
aql> insert into test.seller (PK, seller) values ('1', ff33838a')
OK, 1 record affected.
```

Let’s see what’s there now:

```
aql> show sets
```

Let’s add another entry in another set:

```
aql> insert into test.txn (PK, txnid) values ('12', '8808ee7e)
```

```
aql> show sets
```

Let’s see what’s in our set boo:

```
aql> select * from test.boo
```

What will we store in the database?

- Numerical input variables which are the result of a PCA transformation.

We will load (import) the data

Let’s get to the downloaded dataset from Kaggle.

```
"Time","V1","V2","V3","V4","V5","V6","V7","V8","V9","V10","V11","V12"
,"V13","V14","V15","V16","V17","V18","V19","V20","V21","V22","V23","V
24","V25","V26","V27","V28","Amount","Class"

0,-1.3598071336738,-0.0727811733098497,2.53634673796914,1.37815522427
443,-0.338320769942518,0.462387777762292,0.239598554061257,0.09869790
12610507,0.363786969611213,0.090794171978931 6,-0.551599533260813,-0.6
17800855762348,-0.991389847235408,-0.311169353699879,1.46817697209427
,-0.470400525259478,0.207971241929242,0.0257905801985591,0.4039929602
55733,0.251412098239705,-0.018306777944153,0.277837575558899,-0.11047
3910188767,0.0669280749146731,0.128539358273528,-0.189114843888824,0.
133558376740387,-0.0210530534538215,149.62,"0"
0,1.19185711131486,0.26615071205963,0.16648011335321,0.44815407846091
1,0.0600176492822243,-0.0823608088155687,-0.0788029833323113,0.085101
6549148104,-0.255425128109186,-0.166974414004614,1.61272666105479,1.0
6523531137287,0.48909501589608,-0.143772296441519,0.635558093258208,0
.463917041022171,-0.114804663102346
```

Task:
Load it using the asloader tool in the Aerospike-tools Docker image:

https://aerospike.com/docs/tools/asloader/index.html
https://aerospike.com/docs/tools/asloader/options.html

Steps:

1. Check out the config file and see what's missing:

- creditcard.csv file needs two extra columns:
    - ID column identifies row uniquely
    - set_name column identifies set (table name)
- config json defines the column used for identifier
- config json defines the column used for set name
- config json describes every column and its type

2. Inspect the csv
3. Run bash script to adjust the csv
4. Check out the asloader command
5. Mount the files
6. Run the command

Guidance:

```
$ docker run -it -v path to files:/data  aerospike/aerospike-tools asloader -h localhost -p 3000 -c config csv
```

Solution:

```
$ docker run -it -v $(PWD)/:/data/ aerospike/aerospike-tools asloader  -h $(docker inspect -f '{{.NetworkSettings.IPAddress }}' aerospike) -p 3000 -c /data/config.json /data/creditcard.aerospike.csv
```

2. Write Go Code: data

Infrastructure considerations

- Data dependency

// gomod
go mod init github.com/my/repo

// follow scaffold
Using https://github.com/aerospike/aerospike-client-go

3. Serve the Model

Infrastructure considerations

- Model complexity
- Anti patterns: pipeline jungle, zombie patches, blur architecture, design smells

General knowledge:
There is a TF Go API. To understand a pre-trained model we can use the inspect model CLI.

Back to the workshop:

The model was trained based on Kaggle creditcard data and extracted from this notebook .

(In the context of SKO: here's again the model training playground)

To serve the model:

```
$ docker run -d --rm --name tensorflow serving -p 8501:8501 -v
"$(PWD)/models:/models" -e MODEL NAME=fraud tensorflow/serving
```

Construct the payload

- The Tensorflow server URL is http://localhost:8501/v1/models/fraud:predict
- Input:
    - Use the PCA values (columns V1-V28) and Amount

    - Use the log of amount number to compress the range
      (for example in Go: math.Log(1234.567))
    - example based on first row in creditcard.csv:
      amount is 149.62, log(149,62) is 5.0090987
    - construct JSON payload, eg:

      `{ "inputs": [[v1,v2,v3,...,logAmount]]}` and send to tensforflow server

- Output:
    - response will be `{ "outputs": [[float]]}`
    - Where : 1 - fraud ; 0 - not fraud; The threshold is dynamic

Example based on first line in creditcard.csv

```
curl -d
'{"inputs":[[-1.3598071336738,-0.0727811733098497,2.53634673796914,1.3781552242
7443,-0.338320769942518,0.462387777762292,0.239598554061257,0.0986979012610507,
0.363786969611213,0.0907941719789316,-0.551599533260813,-0.617800855762348,-0.9
91389847235408,-0.311169353699879,1.46817697209427,-0.470400525259478,0.2079712
41929242,0.0257905801985591,0.403992960255733,0.251412098239705,-0.018306777944
153,0.277837575558899,-0.110473910188767,0.0669280749146731,0.128539358273528,-
0.189114843888824,0.133558376740387,-0.0210530534538215,5.00909874644
]]}' -X POST http://localhost:8501/v1/models/fraud:predict

{
  "outputs": [
    [
      0.261857092
    ]
  ]
}
```

For further reading:
[hub/reusable_saved_models.md at master · tensorflow/hub · GitHub]

[TensorFlow Serving with Docker | TFX]

4. Go Code: model serving

Infrastructure considerations

- Config debt
- Model versioning
- Feedback loop

- Glue code

// follow scaffold

5. Go Code: prediction verification

Infrastructure considerations

- Data bias

// follow scaffold

Build a binary:

```
$ go build
```

Run the binary:

```
$ ./fraud-ml-workshop
```

Run a transaction through the system:

```
$ curl -POST "127.0.0.1:8090" -d '{"UserID":"<ID from CSV>"}'
```

Congratulations!

Upload your work to GitHub to show off.
If you want to share this on Twitter - tag Natalie and Aerospike, we will retweet.