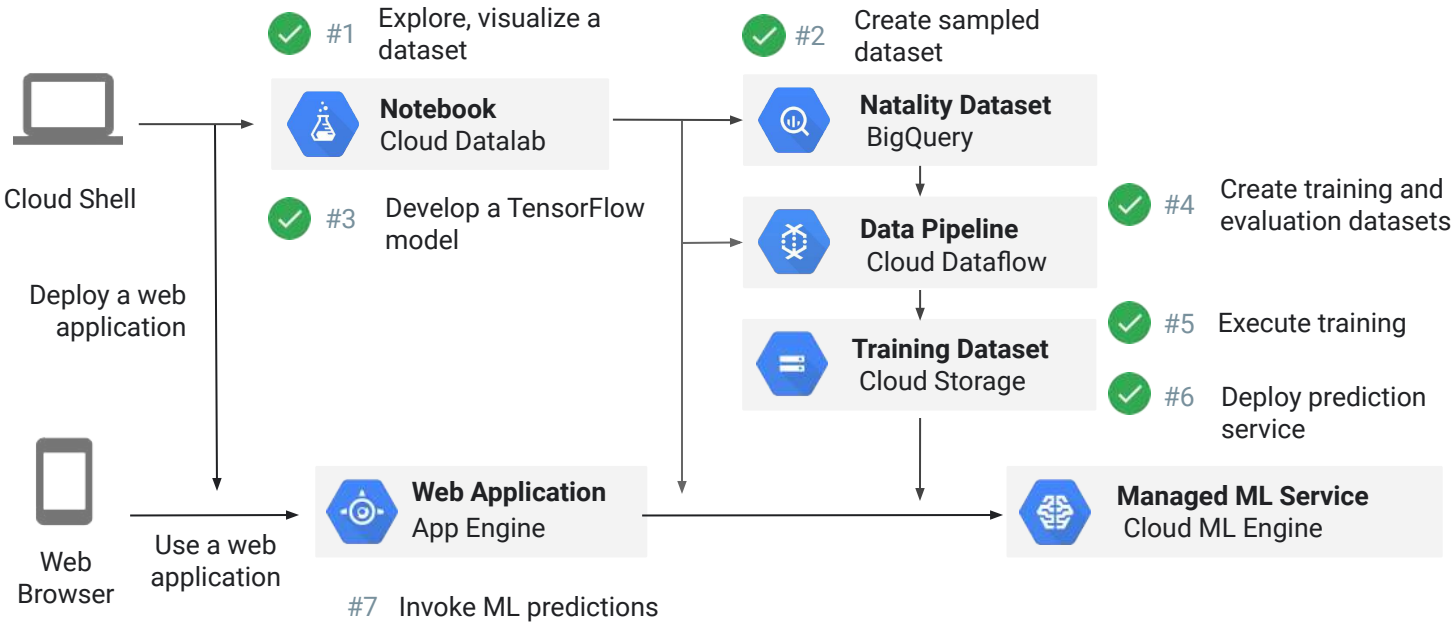
Google Cloud

Summary

Summary: An end-to-end process to operationalize ML models



Summary of learning

1. **Explore, visualize a dataset**
2. Create sampled dataset
3. Develop a TensorFlow model
4. Create training and evaluation datasets
5. Execute training
6. Deploy prediction service
7. Invoke ML predictions

Visualise the dataset in Cloud Datalab

In [1]:
```
# Create SQL query using natality data after the year 2000
query = """
SELECT
  weight_pounds,
  is_male,
  mother_age,
  plurality,
  gestation_weeks,
  ABS(FARM_FINGERPRINT(CONCAT(CAST(YEAR AS STRING), CAST(month AS STRING))))
FROM
  publicdata.samples.natality
WHERE year > 2000
"""
```

In [2]:
```
# Call BigQuery and examine in dataframe
import google.datalab.bigquery as bq
df = bq.Query(query + " LIMIT 100").execute().result().to_dataframe()
df.head()
```

Out[2]:

	weight_pounds	is_male	mother_age	plurality	gestation_weeks	hashmonth
0	3.562670	True	25	1	30	1403073183891835564
1	3.999185	False	30	1	32	7146494315947640619
2	7.438397	True	13	1	34	7146494315947640619
3	4.806077	True	19	1	34	8904940584331855459
4	4.812691	True	22	3	34	2126480030009879160

Summary of learning

1. Explore, visualize a dataset
2. **Create sampled dataset**
3. Develop a TensorFlow model
4. Create training and evaluation datasets
5. Execute training
6. Deploy prediction service
7. Invoke ML predictions

Create a sampled dataset

```
traindf.head()# Let's see a small sample
traindf = preprocess(traindf)
evaldf = preprocess(evaldf)
traindf.head()
```

	weight_pounds	is_male	mother_age	plurality
0	5.436599	True	12	Single(1)
1	6.499227	True	13	Single(1)
2	6.686620	False	13	Single(1)
3	7.577288	True	13	Single(1)

Summary of learning

1. Explore, visualize a dataset
2. Create sampled dataset
3. **Develop a TensorFlow model**
4. Create training and evaluation datasets
5. Execute training
6. Deploy prediction service
7. Invoke ML predictions

Develop a model utilizing TensorFlow model techniques

```
# Create estimator to train and evaluate
def train_and_evaluate(output_dir):
  EVAL_INTERVAL = 300
  run_config = tf.estimator.RunConfig(save_checkpoint
                                      keep_checkpoint
  estimator = tf.estimator.DNNRegressor(
                       model_dir = output_dir,
                       feature_columns = get_cols(),
                       hidden_units = [64, 32],
                       config = run_config)
```

Summary of learning

1. Explore, visualize a dataset
2. Create sampled dataset
3. Develop a TensorFlow model
4. **Create training and evaluation datasets**
5. Execute training
6. Deploy prediction service
7. Invoke ML predictions

Peprocess and create .csv files in Cloud Dataflow to create training and evaluation datasets

```
def preprocess(in_test_mode):
  import shutil, os, subprocess
  job_name = 'preprocess-babyweight-features'
M%S')
```

Summary of learning

1. Explore, visualize a dataset
2. Create sampled dataset
3. Develop a TensorFlow model
4. Create training and evaluation datasets
5. **Execute training**
6. Deploy prediction service
7. Invoke ML predictions

Execute training in the Cloud

```
# Run the model
shutil.rmtree('babyweight_trained', ignore_errors = True)
train_and_evaluate('babyweight_trained')
```

When I ran it, the final lines of the output (above) were:

```
INFO:tensorflow:Saving dict for global step 1000: avera
00, loss = 635.9226
INFO:tensorflow:Restoring parameters from babyweight_tr
```

Summary of learning

1. Explore, visualize a dataset
2. Create sampled dataset
3. Develop a TensorFlow model
4. Create training and evaluation datasets
5. Execute training
6. **Deploy prediction service**
7. Invoke ML predictions

Deploy the model

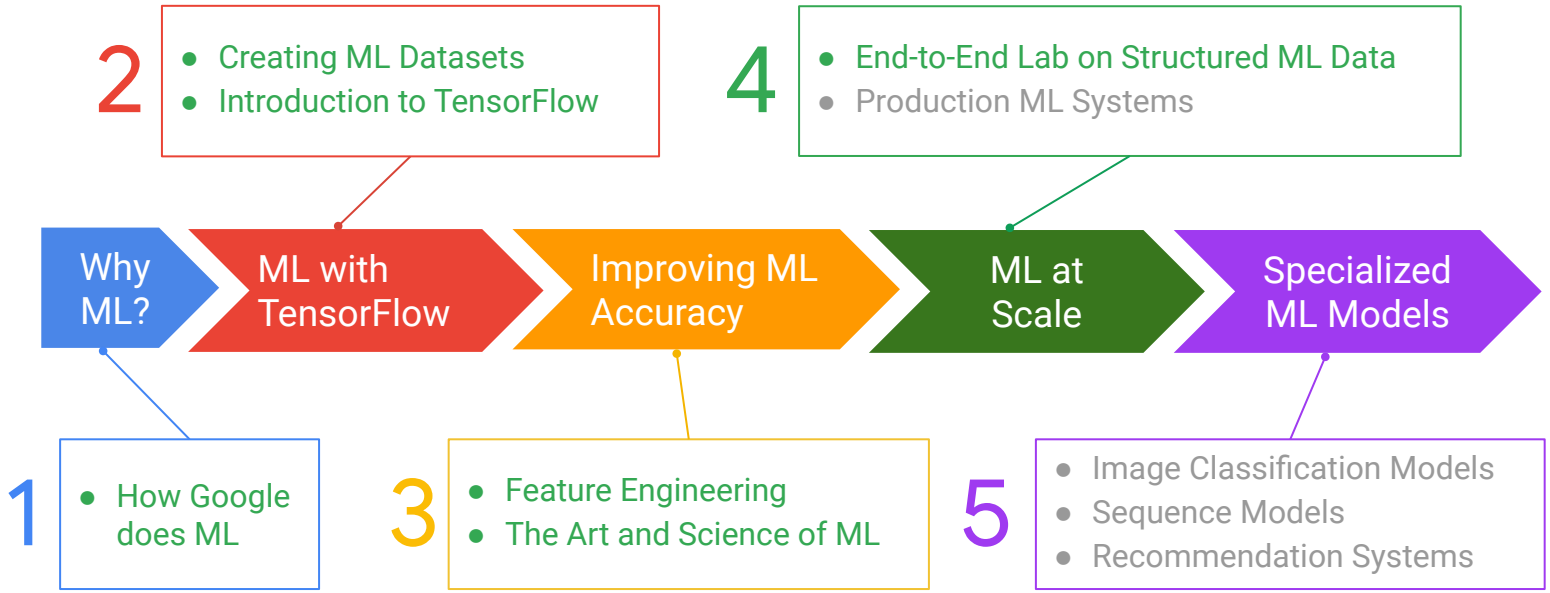
```
%bash
MODEL_NAME="babyweight"
MODEL_VERSION="ml_on_gcp"
MODEL_LOCATION=$(gsutil ls gs://${BUCKET}
echo "Deleting and deploying $MODEL_NAME $
```

Summary of learning

1. Explore, visualize a dataset
2. Create sampled dataset
3. Develop a TensorFlow model
4. Create training and evaluation datasets
5. Execute training
6. Deploy prediction service
7. **Invoke ML predictions**

Deploy a Flask application using Python and App Engine

Machine learning on Google Cloud Platform



cloud.google.com