Machine Learning Infrastructure with Amazon SageMaker and Terraform —Basic Fraud Detection Example

A. Introduction

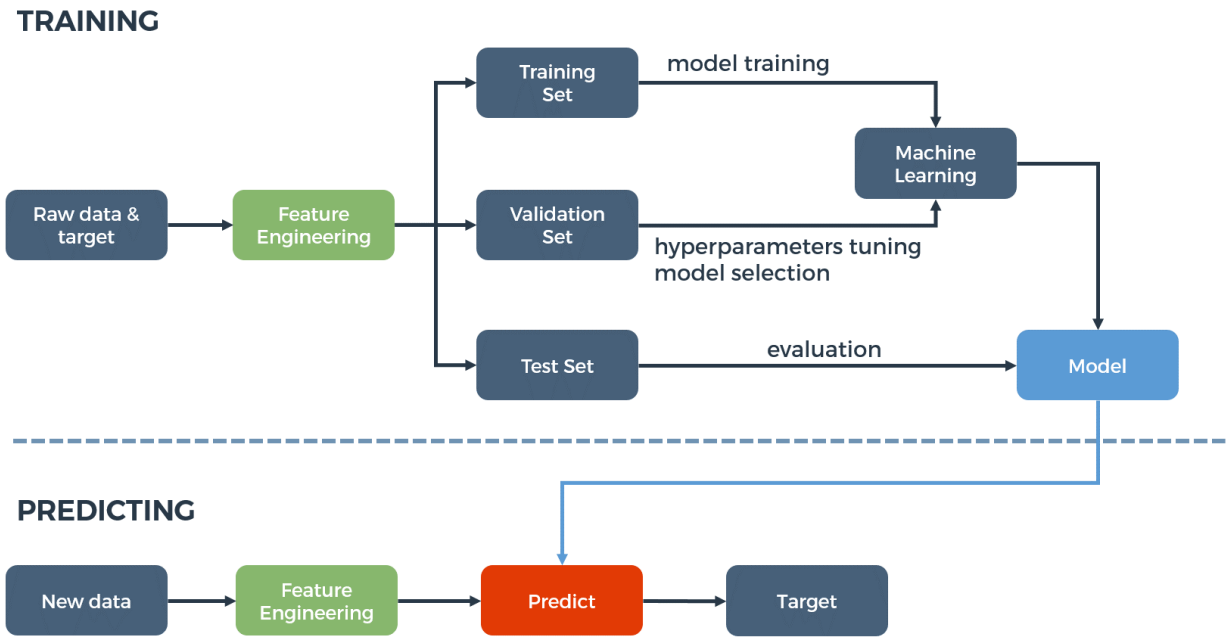
1. The goal of this solution is to predict whether a given transaction is fraudulent or not
2. This is a binary (yes/no) classification solution
3. It typically uses a logistic regression algorithm
4. In Amazon SageMaker, this is accomplished using the built-in linear learner algorithm with binary classifier predictor type.

B. Solution overview

There are 3 main parts to this solution:

1. Training a model based on a sample transaction dataset
2. Storing the model in an accessible interface
3. Using the model to predict whether a generated transaction is fraudulent

The typical machine learning process looks like this:



Source: https://hackernoon.com/a-brief-overview-of-automatic-machine-learning-solutions-automl-2826c7807a2a

---

C. Implementation walkthrough

Walking through the implementation based on the above 3 solution steps.

1. Training a model based on a sample transaction dataset.

To train a model, the following components are needed:

i. Transaction **dataset** where fraudulent transactions have already been identified (a.k.a. labeled)

ii. A machine learning **algorithm** stored as a Docker image

iii. A **server** with pre-loaded machine learning libraries for data scientists to prepare, visualize and enhance the training data, and then set the algorithm's parameters (in layman's term; hyperparameters in machine learning speak) before starting the training process.

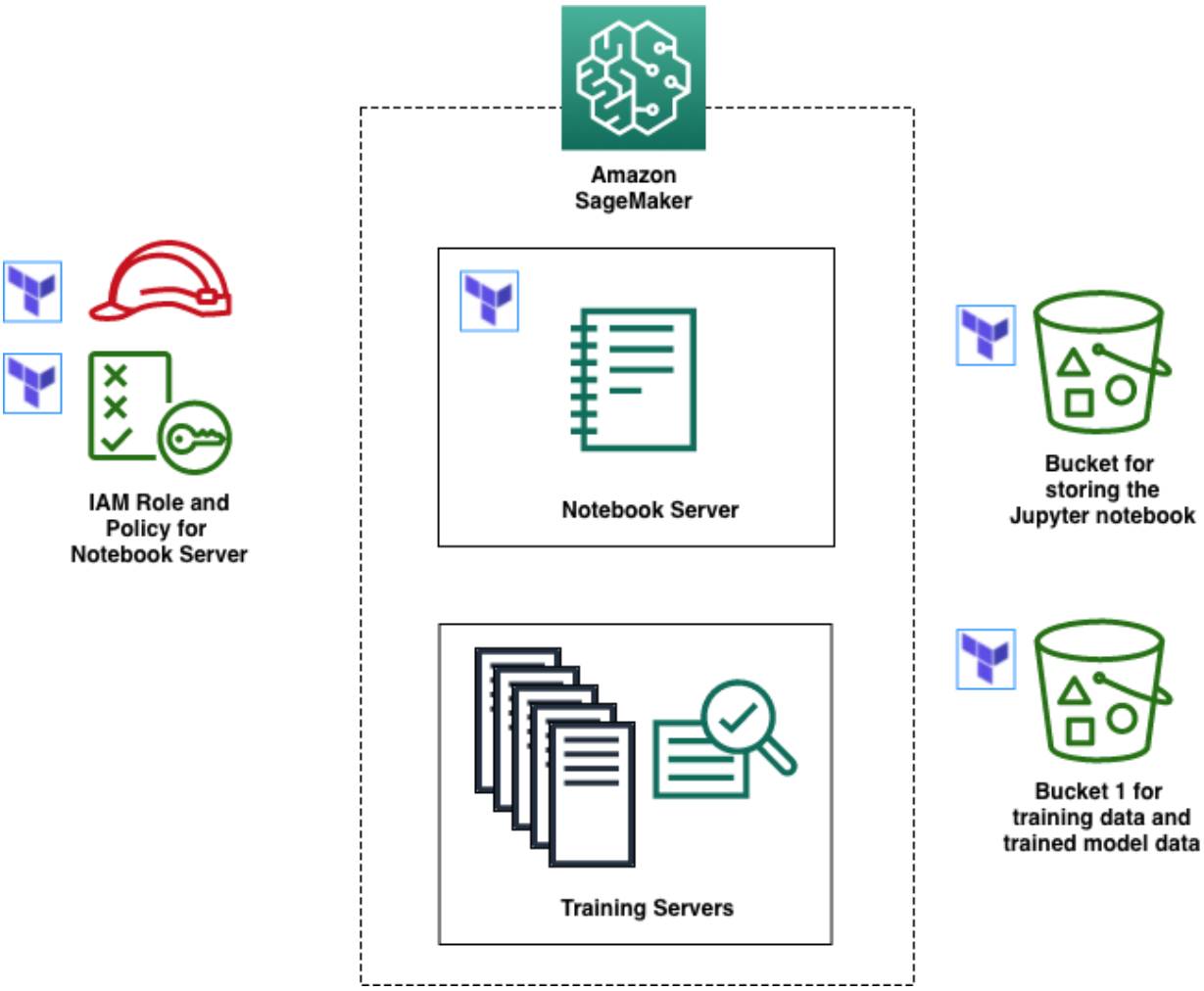
iv. **Server(s)** to train the model using a particular algorithm, this is essentially the "**learning**" part of machine learning, which also is the one that requires the most computational power.

Amazon SageMaker provides **ii**, **iii** and **iv** as managed services, while **i** is taken from a public data source.

1a. Infrastructure

In terms of infrastructure for this part of the solution, we only need to set up **iii** (the notebook instance), which contains all necessary libraries and one **Jupyter notebook**, basically, the data scientists' tooling code (cum document). The **Jupyter notebook,** located at `source/notebooks/sagemaker_fraud_detection.ipynb` in the source code repository, includes code that retrieves **ii** (algorithm) and delivers **iv** (training instances) using Amazon SageMaker's library.

Within AWS world, an instance usually refers to a server, which can be a virtual machine or a container.



As seen above, the notebook instance requires a few supporting resources including:

- IAM Role and Policy for the notebook instance:

```
# iam_sagemaker.tf
resource "aws_iam_role" "sm_notebook_instance_role" {
  name = "sm-notebook-instance-role"
  ...
}
resource "aws_iam_policy" "sm_notebook_instance_policy" {
  name        = "sm-notebook-instance-policy"
  description = "Policy for the Notebook Instance to manage training jobs, models and endpoints"
  ...
}
resource "aws_iam_role_policy_attachment" "sm_notebook_instance" {
  role       = "${aws_iam_role.sm_notebook_instance_role.name}"
  policy_arn = "${aws_iam_policy.sm_notebook_instance_policy.arn}"
}
```

Code: https://github.com/keshava/tf-fraud-detection-using-machine-learning/blob/master/terraform/iam_sagemaker.tf

- S3 bucket for storing the Jupyter notebook:

```
# s3_lambda.tf
resource "aws_s3_bucket" "fraud_detection_function_bucket" {
  bucket = "${var.function_bucket_name}-${var.aws_region}"
  acl    = "private"
  ...
}
```

Code: https://github.com/keshava/tf-fraud-detection-using-machine-learning/blob/master/terraform/s3_lambda.tf

- S3 bucket for storing training data as well as generated model data:

```
# s3_sagemaker.tf
resource "aws_s3_bucket" "s3_bucket_1" {
  bucket = "${var.s3_bucket_name_1}-${var.aws_region}"
  acl    = "private"
}
resource "aws_s3_bucket_object" "s3_fraud_detection_notebook" {
  bucket = "${aws_s3_bucket.fraud_detection_function_bucket.id}"
  key    = "fraud-detection-using-machine-learning/${var.function_version}/notebooks/sagemaker_fraud_detection.ipynb"
  source = "${path.module}/../source/notebooks/sagemaker_fraud_detection.ipynb"
}
```

Code: https://github.com/keshava/tf-fraud-detection-using-machine-learning/blob/master/terraform/s3_sagemaker.tf

- SageMaker notebook instance:

```
# sagemaker.tf
resource "aws_sagemaker_notebook_instance" "basic" {
  name          = "FraudDetectionNotebookInstance"
  role_arn      = "${aws_iam_role.sm_notebook_instance_role.arn}"
  instance_type = "ml.t2.medium"
  ...
}
```

Code: https://github.com/keshava/tf-fraud-detection-using-machine-learning/blob/master/terraform/sagemaker.tf

---

1b. Notebook instance init script

In order for the notebook instance to obtain the Jupyter notebook located in our source code, we have to add an init script that downloads the notebook from the **aws_s3_bucket.fraud_detection_function_bucket** above (we uploaded using the **aws_s3_bucket_object**). This is done using the notebook instance's lifecycle configuration:

```
resource "aws_sagemaker_notebook_instance" "basic" {
  name          = "FraudDetectionNotebookInstance"
  role_arn      = "${aws_iam_role.sm_notebook_instance_role.arn}"
  instance_type = "ml.t2.medium"
  lifecycle_config_name = "${aws_sagemaker_notebook_instance_lifecycle_configuration.basic_lifecycle.name}"
}
resource "aws_sagemaker_notebook_instance_lifecycle_configuration" "basic_lifecycle" {
  name     = "BasicNotebookInstanceLifecycleConfig"
  on_start = "${base64encode(data.template_file.instance_init.rendered)}"
}
data "template_file" "instance_init" {
  template = "${file("${path.module}/template/sagemaker_instance_init.sh")}"
  ...
}
```

Code: https://github.com/keshava/tf-fraud-detection-using-machine-learning/blob/master/terraform/sagemaker.tf

```
cd /home/ec2-user/SageMaker
aws s3 cp s3://${function_bucket_name}-${aws_region}/fraud-detection-using-machine-
learning/${function_version}/notebooks/sagemaker_fraud_detection.ipynb .
sed -i 's/fraud-detection-end-to-end-demo/${s3_bucket_name_1}/g' sagemaker_fraud_detection.ipynb
```

sagemaker_instance_init.sh

1c. Jupyter Notebook Logic for training

These are the main steps in the Jupyter notebook:

i. Download sample data and extract features and label (fraud/nonfraud).

ii. Convert the n-dimensional arrays into RecordIO format (a highly efficient data format).

```
import sagemaker.amazon.common as smac
buf = io.BytesIO()
smac.write_numpy_to_dense_tensor(buf, features, labels)
```

iii. Store the RecordIO data into S3 bucket.

```
bucket = "fraud-detection-end-to-end-demo"
prefix = 'linear-learner'
key = 'recordio-pb-
data'boto3.resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train',
key)).upload_fileobj(buf)
```

iv. Retrieve the Docker image for the linear learner algorithm.

```
container = get_image_uri(boto3.Session().region_name, 'linear-learner')
```

v. Create a training job with the desired instance type and instance count, change the (hyper)parameters of the algorithm and start training using the training data uploaded to S3 earlier. You can see how simple it is to set up a cluster of servers to train a model and only pay for the time that it takes to train, a major cost saver.

```
import sagemaker
s3_train_data = 's3://{}/{}/train/{}'.format(bucket, prefix, key)
output_location = 's3://{}/{}/output'.format(bucket, prefix)linear =
sagemaker.estimator.Estimator(container,
                                get_execution_role(),
                                train_instance_count=1,
                                train_instance_type='ml.c4.xlarge',
                                output_path=output_location,

sagemaker_session=session)linear.set_hyperparameters(feature_dim=features.sha
pe[1],
                           predictor_type='binary_classifier',
                           mini_batch_size=200)linear.fit({'train':
s3_train_data})
```

Code: https://github.com/keshava/tf-fraud-detection-using-machine-learning/blob/master/source/notebooks/sagemaker_fraud_detection.ipynb

---

2. Storing the model in an accessible interface.

To store a model and present it to the outside world for prediction later on, these components are required:

i. Model data

ii. Other model metadata (container, training job, …)

iii. An endpoint (interface) configuration

iv. Actual endpoint (a set of servers to run prediction on-demand)

Amazon Sagemaker hosts **ii**, **iii** and **iv** while **i** (the model data) is stored in S3 as we saw in the previous code (`output_location`). Both **i** and **ii** (the model specification) are created at the end of the call `linear.fit(...)` .

The endpoint configuration (**iii**) and the actual endpoint (**iv**) are created by this part of the Jupyter notebook:

```
linear_predictor = linear.deploy(initial_instance_count=1,
                                 endpoint_name="fraud-detection-endpoint",
                                 instance_type='ml.m4.xlarge')
```

Code: https://github.com/keshava/tf-fraud-detection-using-machine-learning/blob/master/source/notebooks/sagemaker_fraud_detection.ipynb

---

3. Using the model to predict whether a generated transaction is fraudulent.

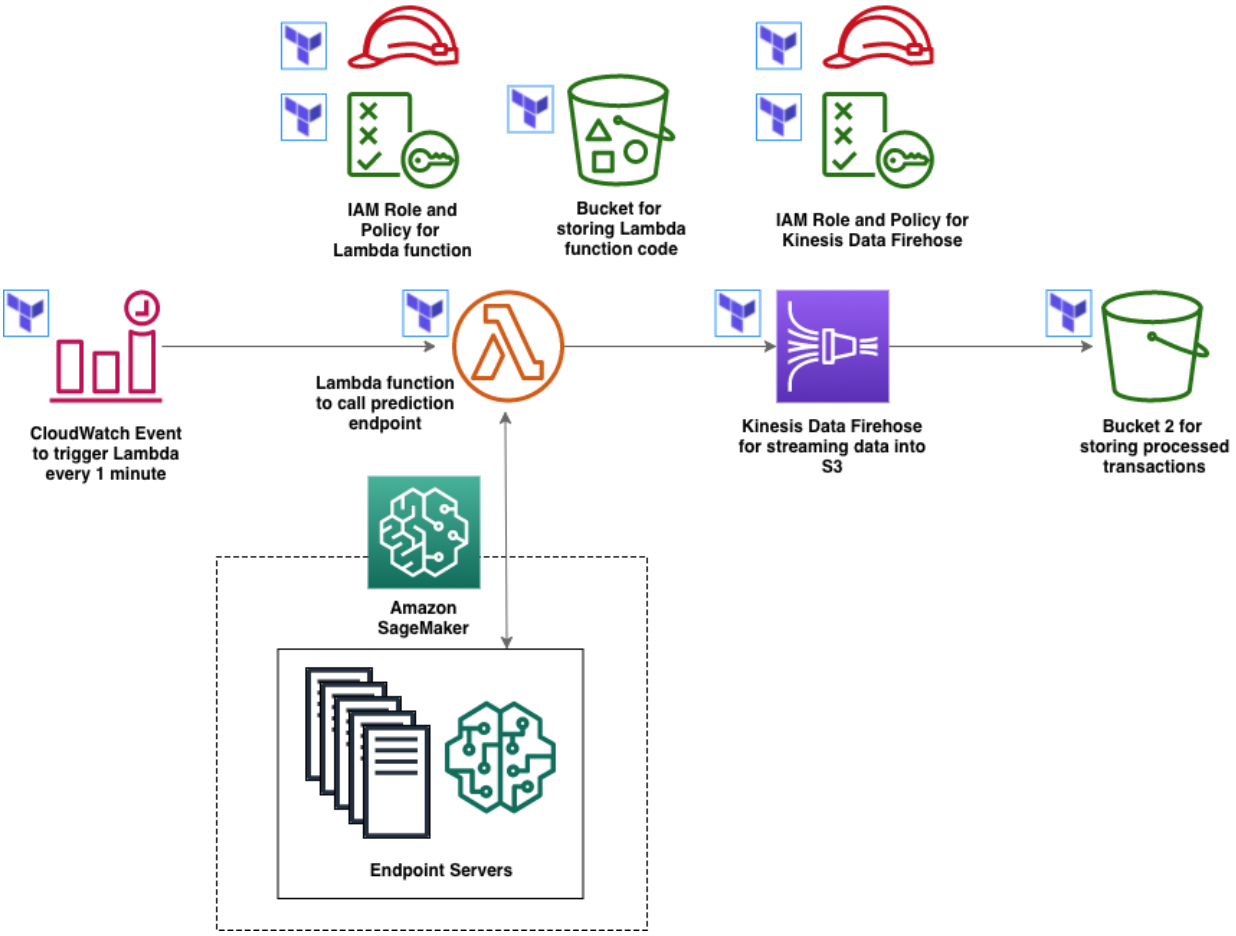
With the created endpoint, we can now use it on any generated transaction.

To simulate transactions coming in for prediction, we need:

- a CloudWatch Event is created to trigger Lambda function every 1 minute.
- an AWS Lambda function is created using code located at `source/fraud_detection/index.py` in the code repository. It follows these steps: randomly selects a predefined fraud or nonfraud transaction, sends the transaction to

the endpoint to obtain a fraud prediction, sends the result to a Kinesis Data Firehose after some minor processing.

- a Kinesis Data Firehose that stores streamed results into S3.
- and finally, the S3 bucket.



As seen above, the resources with the blue Terraform logo are required:

- CloudWatch event:

```
# cloudwatch_event.tf
resource "aws_cloudwatch_event_rule" "fraud_detection_scheduled_rule" {
  name                = "fraud-detection-scheduled-rule"
  schedule_expression = "rate(1 minute)"
  ...
}
```

Code: https://github.com/keshava/tf-fraud-detection-using-machine-learning/blob/master/terraform/cloudwatch_event.tf

- IAM Role and Policy for Lambda prediction function:

```
# iam_lambda.tf
resource "aws_iam_role" "fraud_detection_lambda_role" {
  name = "fraud-detection-lambda-role"
  ...
}
resource "aws_iam_role_policy_attachment" "fraud_detection_lambda" {
  role       = "${aws_iam_role.fraud_detection_lambda_role.name}"
  policy_arn = "${aws_iam_policy.fraud_detection_lambda_policy.arn}"
}
resource "aws_iam_policy" "fraud_detection_lambda_policy" {
  name        = "sm-notebook-instance-policy"
  description = "Policy for the Notebook Instance to manage training jobs, models and endpoints"
  ...
}
```

Code: https://github.com/keshava/tf-fraud-detection-using-machine-learning/blob/master/terraform/iam_lambda.tf

- S3 bucket for storing the function code, taken from `source/fraud_detection/index.py` in our code repository

```
# s3_lambda.tf
resource "aws_s3_bucket" "fraud_detection_function_bucket" {
  bucket = "${var.function_bucket_name}-${var.aws_region}"
  acl    = "private"
  ...
}
data "archive_file" "fraud_detection_archive" {
  type        = "zip"
  source_file = "${path.module}/../source/fraud_detection/index.py"
  output_path = "${path.module}/../dist/fraud_detection.zip"
}
resource "aws_s3_bucket_object" "s3_fraud_detection_archive" {
  bucket = "${aws_s3_bucket.fraud_detection_function_bucket.id}"
  key    = "fraud-detection-using-machine-learning/${var.function_version}/fraud_detection.zip"
  source = "${data.archive_file.fraud_detection_archive.output_path}"
}
```

Code: https://github.com/keshava/tf-fraud-detection-using-machine-learning/blob/master/terraform/s3_lambda.tf

- The Lambda function which processes new transactions:

```
# lambda.tf
resource "aws_lambda_function" "fraud_detection_event_processor" {
  handler       = "index.lambda_handler"
  function_name = "fraud-detection-event-processor"
  role          = "${aws_iam_role.fraud_detection_lambda_role.arn}"
  s3_bucket     = "${aws_s3_bucket.fraud_detection_function_bucket.id}"
  s3_key        = "fraud-detection-using-machine-learning/${var.function_version}/fraud_detection.zip"
  ...
}
```

Complete code: https://github.com/keshava/tf-fraud-detection-using-machine-learning/blob/master/terraform/lambda.tf

- IAM Role and Policy for the Kinesis Data Firehose:

```
# iam_kinesis.tf
resource "aws_iam_role" "fraud_detection_firehose_role" {
  name = "fraud-detection-firehose-role"
  ...
}
resource "aws_iam_role_policy_attachment" "fraud_detection_firehose" {
  role       = "${aws_iam_role.fraud_detection_firehose_role.name}"
  policy_arn = "${aws_iam_policy.fraud_detection_firehose_policy.arn}"
}
resource "aws_iam_policy" "fraud_detection_firehose_policy" {
  name        = "fraud-detection-firehose-policy"
  description = "Policy for the Amazon Kinesis Data Firehose to save data to S3 bucket"
  ...
}
```

Complete code: https://github.com/keshava/tf-fraud-detection-using-machine-learning/blob/master/terraform/iam_kinesis.tf

- S3 bucket to store processed transactions from the Kinesis Data Firehose:

```
# s3_kinesis.tf
resource "aws_s3_bucket" "s3_bucket_2" {
  bucket        = "${var.s3_bucket_name_2}-${var.aws_region}"
  acl           = "private"
  ...
}
```

Code: https://github.com/keshava/tf-fraud-detection-using-machine-learning/blob/master/terraform/s3_kinesis.tf

- And finally, the Kinesis Data Firehose:

```
# kinesis.tf
resource "aws_kinesis_firehose_delivery_stream" "fraud_detection_firehose_stream" {
  name        = "fraud-detection-firehose-stream"
  destination = "s3"

  s3_configuration {
    bucket_arn         = "${aws_s3_bucket.s3_bucket_2.arn}"
    ...
  }
}
```

Code: https://github.com/keshava/tf-fraud-detection-using-machine-learning/blob/master/terraform/kinesis.tf

---

D. Deployment

Terraform code is located in the folder `terraform` (original CloudFormation can be found in `cloudformation`).

Before running `terraform apply`, there are a few set-ups to perform:

i. Copy `terraform_backend.tf.template` to `terraform_backend.tf` and modify values accordingly. You need to manually create an S3 bucket or use an existing one to store the Terraform state file.

```
terraform {
  backend "s3" {
    bucket = "<bucket-name>"
    key    = "fraud-detection/terraform.tfstate"
    region = "<region>"
  }
}
```

terraform_backend.tf.template

Using remote backend for Terraform state is a good practice. It should be practiced even for a simple scenario like this.

ii. Copy `terraform.tfvars.template` to `terraform.tfvars` and modify values accordingly. You don't need to create any buckets specified in here; they're to be created by `terraform apply`.

```
aws_region="<region>"
aws_profile="<profile>"
function_bucket_name="<bucket-for-lambda-function>"
function_version="<version>"
s3_bucket_name_1="<bucket-1>"
s3_bucket_name_2="<bucket-2>"
```

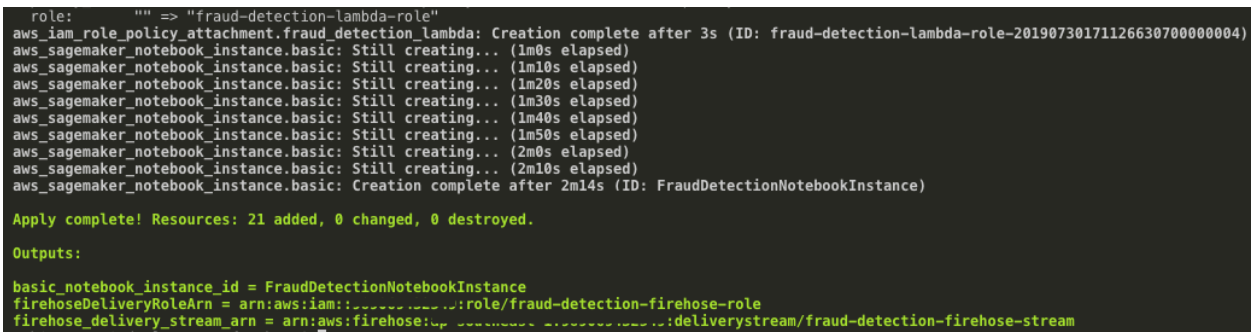
`terraform.tfvars.template`

iii. Once the above files are created, simply run through the following terraform commands. Remember to ensure all commands return ok and to **review the terraform plan before applying**.

```
# Set default AWS profile,
# use 'set' instead of 'export' for Windows.
export AWS_PROFILE=<your desired profile>terraform init
terraform validate
terraform plan -out=tfplan
terraform apply --auto-approve tfplan
```

The Terraform plan output should look like this:

And the result of `terraform apply`:



Final manual steps

Once all Terraform resources are set up, you need to follow these manual steps as documented by the AWS site:

Run the Notebook

1. Navigate to the Amazon SageMaker console.
2. In the navigation pane, select **Notebook instances**.
3. Select **FraudDetectionNotebookInstance**.
4. The notebook instance should already be running.
5. Select **Open Jupyter**.
6. In the Jupyter notebook interface, open the `sagemaker_fraud_detection.ipynb` file.
7. In the **Cell** dropdown menu, select **Run All** to run the file.

Enable the CloudWatch Events Rule

1. Navigate to the AWS Lambda console.
2. In the navigation pane, select **Functions**.
3. Select the `fraud_detection_event_processor` Lambda function.
4. In the diagram in the **Designer** tab, select **CloudWatch Events**.
5. In the **CloudWatch Events** tab, select `<stackname>-ScheduleRule-<id>`.
6. Select **Actions** > **Enable**.
7. Select **Enable**.

Verify the Lambda Function Is Processing Transactions

1. Navigate to the AWS Lambda console.
2. In the navigation pane, select **Functions**.
3. Select the `fraud_detection_event_processor` Lambda function.
4. Select **Monitoring** and verify that the **Invocations** graph shows activity.
5. After a few minutes, check the results Amazon S3 bucket for processed transactions.

E. Cleanup

Once you are done with the experiment, simply perform the followings to delete all resources and save costs. Again, remember to **review the terraform plan before applying**.

```
terraform plan -destroy -out=tfplan
terraform apply tfplan
```

You should be able to see output like this:

```
aws_s3_bucket.s3_bucket_1: Destruction complete after 0s
aws_s3_bucket.fraud_detection_function_bucket: Destruction complete after 0s
aws_iam_role.sm_notebook_instance_role: Destruction complete after 3s

Apply complete! Resources: 0 added, 0 changed, 21 destroyed.
```

F. Summary

Amazon SageMaker is a powerful platform. AWS Solution Builders Team have many examples of how Amazon SageMaker can be utilized. Terraform simplifies the infrastructure setup for an enterprise setup where DevOps and Continuous Delivery are critical.

In my next example I will look into a geoscience Deep Learning example and will also explore how Terraform infrastructure can be designed to fit into a CI/CD pipeline within a machine learning setup.