Fast-Track Machine Learning Operationalization Through Streaming Integration

May 2018
Speakers and Agenda

- Changsha Ma
- Codin Pora
- Steve Wilkes

- Background
- Introduction To Striim
- Lab: Network Intrusion Detection System
- Q&A
The value of ML is based on the real-time handling of high data volume, velocity, and variety at scale.

Intensive real-time pre-processing and feature extraction is required before feeding raw data into models.

Static models cannot fit dynamic data in operational systems even they are fine-tuned offline.

Operational systems demand continuous insights from model serving and minimal human intervention.
Filter, enrich and otherwise prepare streaming data

Land data continuously, in an appropriate format for training a machine learning model

Handle model lifecycles, enabling retraining if the model no longer fits the data

Integrate a trained model into the real-time data stream to make continuous predictions

Visualize the real-time data and associated predictions, and alert on issues
Data Arrives in Streams - Not in Batches

- Humans
- Machines
- Devices
- Database
- Logs
- Events

... Streaming Integration has emerged as a major infrastructure requirement
Striim Is Built For Streaming Integration

Handling extreme volumes of data at scale with high throughput

Processing, analyzing, and correlating data in flight

Making data valuable, verifiable, and visible in real time

It’s all about continuously moving any enterprise data, while:
## Why Are Companies Using Streaming Integration?

| Data Distribution and Consistency | Kafka Integration and Processing  
| Cloud Adoption | Zero Downtime Migration to Cloud  
| Integration for Analytics | Hadoop Integration  
| Real-Time Analytics | Next-Generation Analytics  
| Internet of Things | IoT Edge Processing and Analytics  

© 2018 Striim, Inc. All rights reserved.
Customers Are Modernizing

Moving to a “streaming first” data architecture, supporting cloud, streaming analytics, and IoT.

Bridge the old and new worlds of data.

Streaming Integration is the foundation for data modernization initiatives.
Striim Platform for Streaming Integration with Intelligence

**ENTERPRISE GRADE PLATFORM**

- Continuous Data Collection
- Real-Time Continuous Data Movement
- Integration & Extensibility

**SCALABLE RELIABLE SECURE**

- Real-Time Dashboards & Alerts
- Correlation & Anomaly Detection
- SQL In-Memory Stream Processing

**CONTINUOUS DELIVERY**

- Databases / DWs
- Kafka / Messaging
- Cloud
- Big Data / NoSQL
- Log Files
- Email / SMS

**DATA IS BORN**

REAL-TIME, CONTINUOUS, HETEROGENEOUS INTEGRATION

© 2018 Striim, Inc. All rights reserved.
Streaming Integration with Intelligence

CONTINUOUS DATA INGESTION

DBs
- Oracle CDC
- MS SQL CDC
- MySQL CDC
- HPE NSK CDC
- Maria DB CDC
- JDBC/SQL

Cloud
- Amazon S3
- AWS RDS
- Salesforce

Files
- Log Files
- System Files
- Batch Files

Network
- TCP
- UDP
- HTTP
- MQTT
- Netflow
- PCAP
- OPC-UA

Messaging
- Kafka
- Flume
- JMS
- AMQP

Big Data
- HDFS
- HBase
- Hive
- Hazelcast

Cloud
- Azure Blob
- Azure SQL DB
- Azure HDInsight
- Amazon S3
- Amazon Redshift
- Amazon Kinesis
- Google Big Query

Alerting
- Email
- SMS

Formats
- Delimited
- JSON
- XML
- AVRO
- Template

CONTINUOUS DATA DELIVERY

DBs/DWs
- Oracle
- MS SQL
- MySQL
- Teradata
- PostgreSQL
- MemSQL
- JDBC/SQL

Files
- MQTT
- OPC-UA

Network
- Kafka
- JMS
- AMQP

Big Data
- HDFS
- HBase
- Hive
- Hazelcast

Cloud
- Azure Blob
- Azure SQL DB
- Azure HDInsight
- Amazon S3
- Amazon Redshift
- Amazon Kinesis
- Google Big Query

Alerting
- Email
- SMS

Formats
- Delimited
- JSON
- XML
- AVRO
- Template

Parsers
- Delimited
- JSON
- XML
- Free Text
- Binary
- Name/Value
- Zipped
- AVRO
- GoldenGate
- Apache Log
- Sys Log
- MS Event Log
- Mail Log
- SNMP
- CollectD
- CEF
- DHCP Log
- WCF
- +Others

STREAM PROCESSING WITH INTELLIGENCE

FILTER

TRANSFORM

AGGREGATE

ENRICH

CORRELATE

DRAG & DROP UI

REAL-TIME VISUALIZATION

CONTINUOUS DATA DELIVERY

Alerting
- Email
- SMS

Formats
- Delimited
- JSON
- XML
- AVRO
- Template

Parsers
- Delimited
- JSON
- XML
- Free Text
- Binary
- Name/Value
- Zipped
- AVRO
- GoldenGate
- Apache Log
- Sys Log
- MS Event Log
- Mail Log
- SNMP
- CollectD
- CEF
- DHCP Log
- WCF
- +Others

CONTINUOUS DATA INGESTION

Alerting
- Email
- SMS

Formats
- Delimited
- JSON
- XML
- AVRO
- Template

Parsers
- Delimited
- JSON
- XML
- Free Text
- Binary
- Name/Value
- Zipped
- AVRO
- GoldenGate
- Apache Log
- Sys Log
- MS Event Log
- Mail Log
- SNMP
- CollectD
- CEF
- DHCP Log
- WCF
- +Others
Visualization Through Streaming Dashboards
Striim – Integration Overview

- TQL API (REST / Web Sockets / ODBC & JDBC)
- External Views Into KPIs

Enterprise Grade
Clumped, Distributed, Scalable, Reliable and Secure

- Custom Functions
- Built-In Functions
- External libraries
- ML framework

Publish API

- Transform, Enrich & Aggregate
- Perform Advanced Analytics
CREATE APPLICATION MultiLogApp;
CREATE FLOW MonitorLogs;
CREATE SOURCE AccessLogSource USING...
CREATE TYPE AccessLogEntry ...;
CREATE STREAM AccessStream OF...
CREATE CQ ParseAccessLog ...
## Core Striim Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source</strong></td>
<td>Source accesses external data and provides real-time continuous events into streams</td>
</tr>
<tr>
<td><strong>Stream</strong></td>
<td>Stream carries data between components and nodes</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Type is a named set of fields, each has a name and a data type, such as <em>Integer</em> or <em>String</em></td>
</tr>
<tr>
<td><strong>Window</strong></td>
<td>Window provides moving snapshot/collection of events for aggregates and models</td>
</tr>
<tr>
<td><strong>Cache</strong></td>
<td>Cache is external contextual data made available using distributed in-memory grid</td>
</tr>
<tr>
<td><strong>CQ</strong></td>
<td>A Continuous Query emits big data records after processing real-time streaming events (can process data from streams, windows, caches, event tables, and stores)</td>
</tr>
<tr>
<td><strong>Target</strong></td>
<td>Target outputs real-time big data records to external systems</td>
</tr>
<tr>
<td><strong>Results</strong></td>
<td>A Results Store contains results of processing within the Striim server, defines the data allowed in the store and how it is persisted to a backend system (default ES)</td>
</tr>
</tbody>
</table>
Lab: Network Intrusion Detection System (NIDS)

Start Striim server in virtual machine striimAIDC

Navigate to Striim Web UI in your host/VM browser

Note: if the URL is not accessible from you host browser, go to Applications -> Sundry -> Firewall -> Options -> Change default zone to trusted.
Network Intrusion Detection System (NIDS)

• Data:
  – Network flows with robust features from tcpdump analyzer

• Tasks:
  – Detect abnormal flows with low false positive rate
  – Automatically adapt model serving to data evolution
  – Continuously monitor system and alert on issues in real time

• Algorithms
  – One-Class SVM (Weka LibSVM)
  – Time series spike detection
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>srcip</td>
<td>nominal</td>
<td>Source IP address</td>
</tr>
<tr>
<td>sport</td>
<td>integer</td>
<td>Source port number</td>
</tr>
<tr>
<td>dstip</td>
<td>nominal</td>
<td>Destination IP address</td>
</tr>
<tr>
<td>dsport</td>
<td>integer</td>
<td>Destination port number</td>
</tr>
<tr>
<td>proto</td>
<td>nominal</td>
<td>Transaction protocol</td>
</tr>
<tr>
<td>state</td>
<td>nominal</td>
<td>Indicates to the state and its dependent protocol</td>
</tr>
<tr>
<td>dur</td>
<td>Float</td>
<td>Record total duration</td>
</tr>
<tr>
<td>sbytes</td>
<td>Integer</td>
<td>Source to destination transaction bytes</td>
</tr>
<tr>
<td>dbytes</td>
<td>Integer</td>
<td>Destination to source transaction bytes</td>
</tr>
<tr>
<td>sttl</td>
<td>Integer</td>
<td>Source to destination time to live value</td>
</tr>
<tr>
<td>dttl</td>
<td>Integer</td>
<td>Destination to source time to live value</td>
</tr>
<tr>
<td>sloss</td>
<td>Integer</td>
<td>Source packets retransmitted or dropped</td>
</tr>
<tr>
<td>dloss</td>
<td>Integer</td>
<td>Destination packets retransmitted or dropped</td>
</tr>
<tr>
<td>service</td>
<td>nominal</td>
<td>http, ftp, smtp, ssh, dns, ftp-data, irc</td>
</tr>
<tr>
<td>Sload</td>
<td>Float</td>
<td>Source bits per second</td>
</tr>
<tr>
<td>Dload</td>
<td>Float</td>
<td>Destination bits per second</td>
</tr>
<tr>
<td>Spkts</td>
<td>integer</td>
<td>Source to destination packet count</td>
</tr>
<tr>
<td>Dpkts</td>
<td>integer</td>
<td>Destination to source packet count</td>
</tr>
<tr>
<td>swin</td>
<td>integer</td>
<td>Source TCP window advertisement value</td>
</tr>
<tr>
<td>dwin</td>
<td>integer</td>
<td>Destination TCP window advertisement value</td>
</tr>
<tr>
<td>stcpb</td>
<td>integer</td>
<td>Source TCP base sequence number</td>
</tr>
<tr>
<td>dtcpb</td>
<td>integer</td>
<td>Destination TCP base sequence number</td>
</tr>
<tr>
<td>smeansz</td>
<td>integer</td>
<td>Mean of the packet size transmitted by the src</td>
</tr>
<tr>
<td>dmeansz</td>
<td>integer</td>
<td>Mean of the packet size transmitted by the dst</td>
</tr>
<tr>
<td>trans_depth</td>
<td>integer</td>
<td>Represents the pipelined depth into the connection</td>
</tr>
<tr>
<td>Sjit</td>
<td>Float</td>
<td>Source jitter (mSec)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Djit</td>
<td>Float</td>
<td>Destination jitter (mSec)</td>
</tr>
<tr>
<td>res_bdy_len</td>
<td>integer</td>
<td>Actual uncompressed content size of the data transferred from the server’s http service.</td>
</tr>
<tr>
<td>Sintpkt</td>
<td>Float</td>
<td>Source interpacket arrival time (mSec)</td>
</tr>
<tr>
<td>Dintpkt</td>
<td>Float</td>
<td>Destination interpacket arrival time (mSec)</td>
</tr>
<tr>
<td>tcprtt</td>
<td>Float</td>
<td>TCP connection setup round-trip time, the sum of ‘synack’ and ‘ackdat’.</td>
</tr>
<tr>
<td>synack</td>
<td>Float</td>
<td>TCP connection setup time, the time between SYN and SYN_ACK.</td>
</tr>
<tr>
<td>ackdat</td>
<td>Float</td>
<td>TCP connection setup time, the time between SYN_ACK and ACK.</td>
</tr>
<tr>
<td>is_sm_ips_ports</td>
<td>Binary</td>
<td>If source and destination IP addresses equal and port numbers equal then, this variable takes value 1 else 0</td>
</tr>
<tr>
<td>ct_state_ttl</td>
<td>Integer</td>
<td>No. for each state according to specific range of values for source/destination time to live.</td>
</tr>
<tr>
<td>ct_flow_http_mthd</td>
<td>Integer</td>
<td>No. of flows that has methods such as Get and Post in http service.</td>
</tr>
<tr>
<td>is_ftp_login</td>
<td>Binary</td>
<td>If the ftp session is accessed by user and password then 1 else 0.</td>
</tr>
<tr>
<td>ct_fsp_cmd</td>
<td>Integer</td>
<td>No of flows that has a command in ftp session.</td>
</tr>
<tr>
<td>ct_srv_src</td>
<td>Integer</td>
<td>No. of connections that contain the same service and source address in 100 connections according to the last time.</td>
</tr>
<tr>
<td>ct_srv_dst</td>
<td>Integer</td>
<td>No. of connections that contain the same service and destination address in 100 connections according to the last time.</td>
</tr>
<tr>
<td>ct_dst_ltm</td>
<td>Integer</td>
<td>No. of connections of the same destination address in 100 connections.</td>
</tr>
<tr>
<td>ct_src_ltm</td>
<td>Integer</td>
<td>No. of connections of the same source address in 100 connections.</td>
</tr>
<tr>
<td>ct_src_dport_ltm</td>
<td>Integer</td>
<td>No. of connections of the same source address and destination port in 100 connections.</td>
</tr>
<tr>
<td>ct_dst_sport_ltm</td>
<td>Integer</td>
<td>No. of connections of the same destination address and the source port in 100 connections.</td>
</tr>
<tr>
<td>ct_dst_src_ltm</td>
<td>Integer</td>
<td>No. of connections of the same source and destination address in 100 connections.</td>
</tr>
</tbody>
</table>
Deploy anomalyDetection application, and view Flow

Anomalies
NIDS Task 1: Anomaly Detection on Network Flows

- Ingest data
- Filter data fields
- Preprocess raw data
- Aggregate events
- Extract features
- Detect anomalies
- Persist results
NIDS Task 1: Anomaly Detection on Network Flows

- Ingest data
NIDS Task 1: Anomaly Detection on Network Flows

**Filter data fields**

```
SELECT "NIDS", TO_DATE(TO_LONG(data[0])*1000),
       TO_STRING(data[1]), TO_STRING(data[2]),
       TO_Double(data[3]), TO_STRING(data[4]), TO_STRING(data[5]),
       TO_STRING(data[6]), TO_Double(data[7]), TO_Double(data[8]),
       TO_Double(data[9]), TO_Double(data[10]), TO_Double(data[11]),
       TO_Double(data[12]), TO_Double(data[13]), TO_Double(data[14]),
       TO_Double(data[15]), TO_Double(data[16]), TO_Double(data[17]),
       TO_Double(data[18]), TO_Double(data[19]), TO_Double(data[20]),
       TO_Double(data[21]), TO_Double(data[22]), TO_Double(data[23]),
       TO_Double(data[24]), TO_Double(data[25]), TO_Double(data[26]),
       TO_Double(data[27]), TO_Double(data[28]), TO_Double(data[29]),
       TO_Double(data[30]), TO_Double(data[31]), TO_Double(data[32]),
       TO_Double(data[33]), TO_Double(data[34]), TO_Double(data[35]),
       TO_Double(data[36]), TO_Double(data[37]), TO_Double(data[38]),
       TO_Double(data[39]), TO_Double(data[40]), TO_Double(data[41]),
       TO_Double(data[42]), TO_Double(data[43]), TO_Double(data[44]),
FROM dataStream c WHERE PAUSE(15000L, c)
```
NIDS Task 1: Anomaly Detection on Network Flows

- Preprocess raw data
  - Enrich stream with standardization parameters from cache
NIDS Task 1: Anomaly Detection on Network Flows

- Preprocess raw data
NIDS Task 1: Anomaly Detection on Network Flows

- Aggregate events
  - 10 seconds as observation interval
  - Two adjacent points in time series have 10-second time difference
NIDS Task 1: Anomaly Detection on Network Flows

• Extract features

select last(datetime) as datetime, last(name) as name, count(datetime) as total, list(src) as src, list(dst) as dst, list(dur) as f1, list(proto) as f2, list(service) as f3, list(state) as f4, list(spkts) as f5, list(dpkts) as f6, list(sbytes) as f7, list(dbytes) as f8, list(rate) as f9, list(sttl) as f10, list(dttl) as f11, list(sload) as f12, list(dload) as f13, list(sloss) as f14, list(dloss) as f15, list(sinpkt) as f16, list(dinpkt) as f17, list(sjit) as f18, list(djit) as f19, list(swin) as f20, list(sicpnb) as f21, list(dtcpb) as f22, list(dwin) as f23, list(tcprrt) as f24, list(synack) as f25, list(ackdat) as f26, list(smean) as f27, list(dmean) as f28, list(trans_depth) as f29, list(response_body_len) as f30, list(ct_srv_src) as f31, list(ct_state_ttl) as f32, list(ct_dst_ttl) as f33, list(ct_src_sport) as f34, list(ct_dst_sport) as f35, list(ct_dst_src_ttl) as f36, list(is_fip_login) as f37, list(ct_fip_cmd) as f38, list(ct_flw_http_mthd) as f39, list(ct_src_ttl) as f40, list(ct_srv_dst) as f41, list(is_sm_ips_ports) as f42 from tenSecBatch
NIDS Task 1: Anomaly Detection on Network Flows

• Detect anomalies
  – One-class SVM

select datetime, name, total, AIDCSVMscore(f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16, f17, f18, f19, f20, f21, f22, f23, f24, f25, f26, f27, f28, f29, f30, f31, f32, f33, f34, f35, f36, f37, f38, f39, f40, f41, f42, '/home/admin/Striim/IntelAICon/appData/arffHeader.txt', '/home/admin/Striim/IntelAICon/appData/testData.arff', '/home/admin/Striim/IntelAICon/idsOneClass.model') as anomalySum, anomalySum.size() as anomalyNum, anomalyExtract(anomalySum, src) as anomalySrc, anomalyExtract(anomalySum, dst) as anomalyDst from NIDSfeatureStream;
NIDS Task 1: Anomaly Detection on Network Flows

• Persist results
NIDS Task 1: Anomaly Detection on Network Flows

1. Run application to see streaming processing results at each step (Click one stream and then click Preview on Run).

2. Go to /home/admin/Striim to see the persisted results in alertSummary.csv

3. Stop and undeploy the application
NIDS Task 2 (1): Detect Data Pattern Change

Deploy *anomalyDetection* and then deploy *retrain*

View Flow of *retrain*

No. of anomalies increases due to data evolution
NIDS Task 2 (1): Detect Data Pattern Change

- Difference time series
- Aggregate recent time events and prepare algorithm input
- Detect time series peaks
NIDS Task 2 (1): Detect Data Pattern Change

- **Difference time series**

  - **twoAlertEvent**
    - Mode: Sliding

  - **alertDifference**
    - QUERY
      ```
      SELECT last(datetime) as datetime, name, to_double(last(total)-first(total)) as totalDiff, to_double(last(anomalyNum)-first(anomalyNum)) as anomalyDiff FROM twoAlertEvent having count(*) = 2
      ```

  - **alertDifference1**
    - QUERY
      ```
      SELECT last(datetime) as datetime, name, 0.0 as totalDiff, 0.0 as anomalyDiff FROM twoAlertEvent having count(*) < 2
      ```
NIDS Task 2 (1): Detect Data Pattern Change

- Aggregate recent time events
  - Constrain peak detection on

```sql
-- ThirtyAlertDiffEvent
SELECT last(datetime) as datetime, name, list(anomalyDiff) as anomalyRun
FROM thirtyAlertDiffEvent
HAVING count(*) = 31
```

```sql
-- PeakAlertDetectPrepare
SELECT last(datetime) as datetime, name, list(0.0) as anomalyRun
FROM thirtyAlertDiffEvent
HAVING count(*) < 31
```
NIDS Task 2 (1): Detect Data Pattern Change

- Detect time series peaks
  - Calculate z-score of the 31th event in the sliding window

```
QUERY

SELECT datetime as datetime, name as name, peakDetect(anomalyNum, 5) as anomalyPeak
FROM peakAlertDetectStream
```
NIDS Task 2 (2): Retrain Model

- Aggregate new training data
- Aggregate peak signals
- Join two parallel flows
- Cancel/trigger retraining
NIDS Task 2 (2): Retrain Model

- Aggregate new training data

**INPUT FROM**

- Input Stream: tcpdumpProcessStream
- Type: tcpdumpType

**SIZE OF WINDOW**

- Time: 5 MINUTE

NIDS Task 2 (2): Retrain Model

• Aggregate peak signals
NIDS Task 2 (2): Retrain Model

• Join two parallel flows
  – Features from *NIDStrainPrepare*
  – Data change signal from *NIDStrainPrepare1*
  – Insert into the same stream
NIDS Task 2 (2): Retrain Model

- Cancel/trigger retraining
  
  ```
  CREATE JUMPING WINDOW matchWin OVER NIDStrainDataStream KEEP 2 rows WITHIN 30 second:
  ```

  ![Diagram showing the retraining process]
NIDS Task 2 (2): Retrain Model

• Cancel/trigger retraining
  – Select fields from upstreams

SELECT first(name) as name, first(datetime) as ts1, last(datetime) as ts2, first(total) as total1, last(total) as total2, first(triggerFlag) as flag1, last(triggerFlag) as flag2, first(f1) as f1, first(f2) as f2, first(f3) as f3, first(f4) as f4, first(f5) as f5, first(f6) as f6, first(f7) as f7, first(f8) as f8, first(f9) as f9, first(f10) as f10, first(f11) as f11, first(f12) as f12, first(f13) as f13, first(f14) as f14, first(f15) as f15, first(f16) as f16, first(f17) as f17, first(f18) as f18, first(f19) as f19, first(f20) as f20, first(f21) as f21, first(f22) as f22, first(f23) as f23, first(f24) as f24, first(f25) as f25, first(f26) as f26, first(f27) as f27, first(f28) as f28, first(f29) as f29, first(f30) as f30, first(f31) as f31, first(f32) as f32, first(f33) as f33, first(f34) as f34, first(f35) as f35, first(f36) as f36, first(f37) as f37, first(f38) as f38, first(f39) as f39, first(f40) as f40, first(f41) as f41, first(f42) as f42 from matchWin having count(*) = 2
NIDS Task 2 (2): Retrain Model

- **Cancel/trigger retraining**
  - If there is data change signal, trigger retraining, otherwise cancel it.

```
SELECT ts1, ts2, flag2 AS signal,
CASE WHEN signal > 0 THEN
  AIDCVMtrain(f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16, f17, f18, f19, f20, f21, f22, f23, f24, f25, f26, f27, f28, f29, f30, f31, f32, f33, f34, f35, f36, f37, f38, f39, f40, f41, f42,
  '/home/admin/Striim/IntelAICon/appData/arffHeader.txt',
  '/home/admin/Striim/IntelAICon/appData/trainData.arff', '-S 2 -K 2 -D 3 -G 0.1 -R 0.0 -N 0.1 -M 40.0 -C 1.0 -E 0.001 -P 0.1',
  '/home/admin/Striim/IntelAICon/idsOneClass.model')
ELSE 1
END as status,
CASE WHEN signal > 0 THEN
  'retrain'
ELSE 'model serving'
END as retrainMsg
FROM triggerStream;
```
NIDS Task 3 (1): Create WactionStores and Alerts

Deploy \textit{monitor} application (\textit{anomalyDetection} and \textit{retrain} have been deployed in Task 2)

View Flow of \textit{monitor}
NIDS Task 3 (1): Create WactionStores and Alerts

- Monitor traffic (normal and abnormal) sources and destinations
- Monitor No. of network flows and anomalies
- Alert on data pattern change
- Alert on retraining
NIDS Task 3 (1): Create WactionStores and Alerts

- Monitor traffic sources and destinations

```sql
select datetime, 'srcIP', 'all', first(src), count(src) from tenSecBatch group by src, datetime having count(*) > 0
```

```sql
select datetime, 'dstIP', 'all', first(dst), count(dst) from tenSecBatch group by dst, datetime having count(*) > 0
```
NIDS Task 3 (1): Create WactionStores and Alerts

- Monitor abnormal traffic sources and destinations

```sql
select datetime, 'srcIP' as name, 'anomaly' as flowType, to_string(lst) as IP from alertStream s, iterator(s.anomalySrc) lst;
```

```sql
select datetime, 'dstIP' as name, 'anomaly' as flowType, to_string(lst) as IP from alertStream s, iterator(s.anomalyDst) lst;
```
NIDS Task 3 (1): Create WactionStores and Alerts

• Monitor abnormal traffic sources and destinations
NIDS Task 3 (1): Create WactionStores and Alerts

• Monitor abnormal traffic sources and destinations

select datetime, name, flowType, first(IP), count(IP) from tenSecBatch2 group by IP, datetime, name having count(*) > 0;
NIDS Task 3 (1): Create WactionStores and Alerts

- Feed monitor stream into WactionStore

```
FeedIPsum
```

```
QUERY

SELECT * FROM IPSumStream
```
NIDS Task 3 (1): Create WactionStores and Alerts

- Monitor No. of network flows and anomalies
- Feed stream into WactionStore

```sql
SELECT datetime, total, anomalyNum FROM alertStream
```
NIDS Task 3 (1): Create WactionStores and Alerts

- Alert on data pattern change

SQL Query:

```sql
SELECT 'Data Drift', s.name,
CASE
  WHEN s.anomalyPeak < 1 THEN 'info'
  ELSE 'warning' END,
CASE
  WHEN s.anomalyPeak < 1 THEN 'cancel'
  ELSE 'raise' END,
CASE
  WHEN s.anomalyPeak > 0 THEN 'Anomaly Peak Detected! Prepare to retrain the model'
  ELSE ''
END
FROM peakAlertStream s
```
NIDS Task 3 (1): Create WactionStores and Alerts

- Alert on retraining

```
SELECT 'Retrain', s.name,
CASE
  WHEN s.flag1 + s.flag2 < 1 THEN 'info'
  ELSE 'warning'
END,
CASE
  WHEN s.flag1 + s.flag2 < 1 THEN 'cancel'
  ELSE 'raise'
END,
CASE
  WHEN s.flag1 + s.flag2 > 0 THEN 'Model retraining!''
  ELSE ''
END
FROM triggerStream s
```
NIDS Task 3 (2): Build Dashboard

1. Start applications *monitor*, then *retrain*, then *anomalyDetection*

2. Navigate to Dashboard and view NIDSDash to see real-time visualization

3. Click “Network Traffic Monitor” on the main page to drill down and to see anomaly details
Learn More About Striim Product

Click *Edit* on the top right corner of dashboard to see how to configure a visualization chart.

Open a new tab

1. Go to Source Preview -> Browse -> choose tcpdumpData.csv to preview
2. Set column delimiter as ",", configure column name and type, save as source
3. Set application name as “firstStriimApp”, and source name as “firstStriimSource”
4. Edit a continuous query with SQL-like language, deploy your application and run to see output.
**Key Takeaways**

- Streaming integration paves the way to ML operationalization
- Striim provides reliable and efficient streaming integration solution
- Support fast-track ML operationalization
- Striim filters, enriches and prepares streaming data
- Striim lands data continuously for model training
- Striim supports continuous model serving on data streams
- Striim handles model lifecycles with high automation
- Striim visualizes the real-time data and predictions, and alerts on issues