HDF 3.1 Streaming Webinar

George Vetticaden

VP of Emerging Products

Reference Architecture: Real-time Streaming Analytics

HORTONWORKS

Trucking company w/ large fleet of international trucks

A truck generates millions of events for a given route; an event could be:

- 'Normal' events: starting / stopping of the vehicle
- 'Violation' events: speeding, excessive acceleration and breaking, unsafe tail distance
- 'Speed' Events: The speed of a driver that comes in every minute.

Company uses an application that monitors truck locations and violations from the truck/driver in real-time

Route?

Truck?

Driver?

Analysts query a broad history to understand if today's violations are part of a larger problem with specific routes, trucks, or drivers

HORTONWORKS

Real-time Analytics Architecture with HDF

Centralized Schema Repository

SCHEMA REGISTRY

Sensor Sources

Truck Sensors

Truck Sensors

Truck Sensors

minifi

Truck Sensors

minifi

Cloud Instance in Different Geo Locations

China Cloud Instance (IBM)

Germany Cloud Instance (Azure)

US Cloud Instance (Amazon)

Flow Management Clusters

Ingress Gateway

Nifi Site to Site Protocol

kafka

nifi

Egress Gateway

kafka

Stream Analytics Clusters

Event Broker Cluster

kafka

Ingest Streams

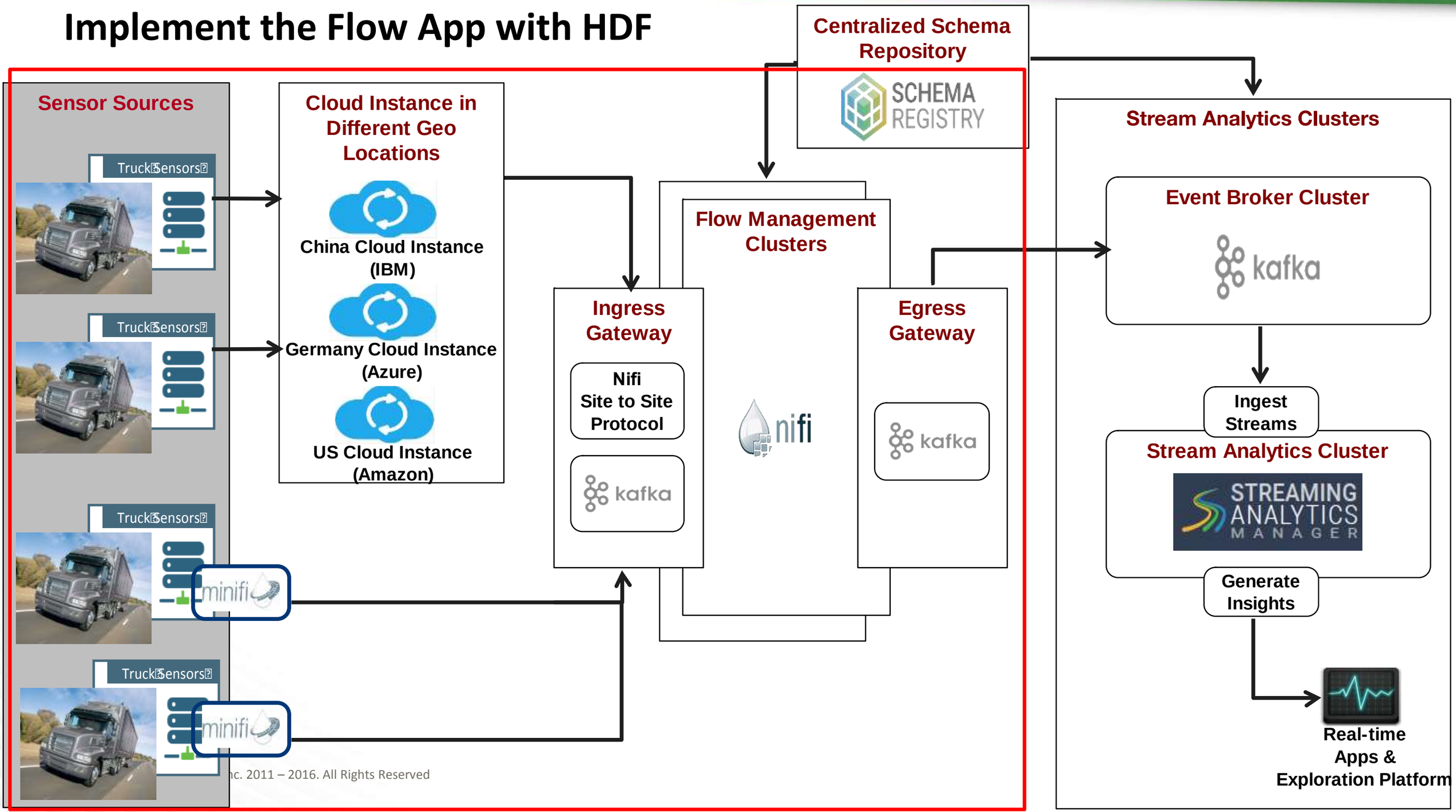
Stream Analytics Cluster

STREAMING ANALYTICS MANAGER

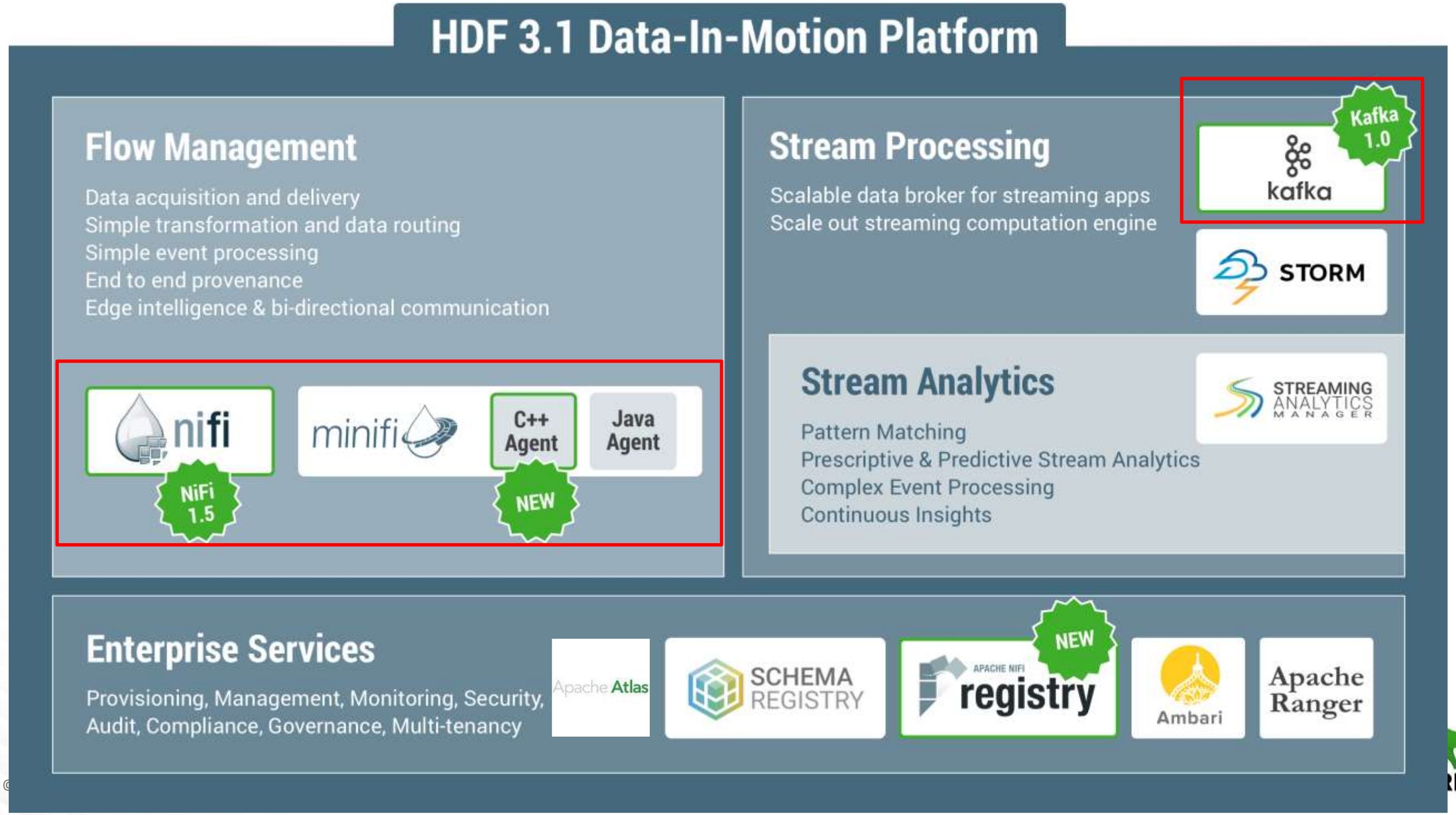
Generate Insights

Real-time Apps & Exploration Platform

Implement the Flow App with HDF



Implementing the Flow Requirements with Apache NiFi and Kafka



Apache Kafka 1.0

Key Highlights

- Kafka 1.0 was the **most asked for feature**/component in HDF Kafka 1.0
- Key Features introduced in Kaka 0.11/1.0 included
  - Kafka **Message Header support**
  - **Transactional Support**
  - **Performance** improvements
- These features are critical for customers who are building streaming apps
- Customer didn't' just want support for Kafka 1.0 but **want full HDF integration with Kafka 1.0** including Nifi, Ambari, Ranger, and Atlas Integration

Kafka 1.0 - NiFi & SAM

- **New Nifi Kafka 1.0 Processors**
  - Kafka header & transaction support
- **SAM Source/Sink for Kafka 1.0**

Kafka 1.0 - Ambari

- **Ambari support for Kafka 1.0**
  - Install, configure, manage & monitor Kafka 1.0 multi-node secure clusters.

Kafka 1.0 - Ranger

- **Ranger ACL support for Kafka 1.0**
  - Support for resource and tag based access control for Kafka 1.0

Kafka 1.0 - Atlas

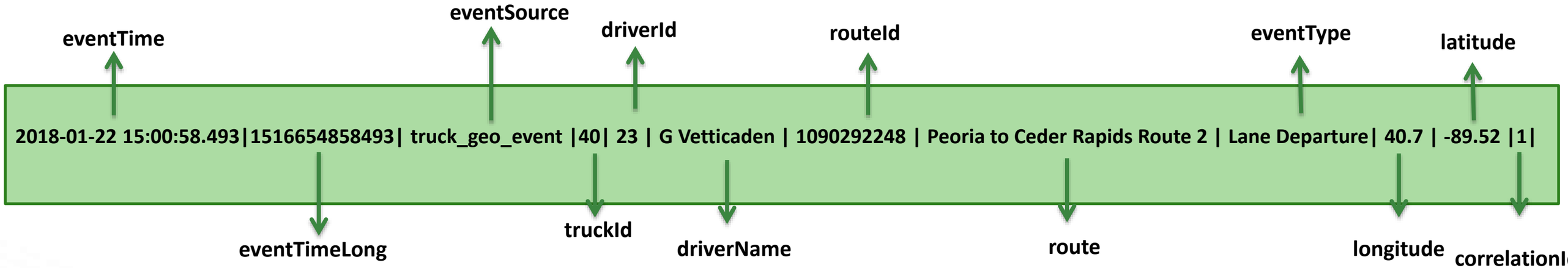
- **Lineage of Kafka Topics**
  - Who are all the consumers and producers

HORTONWORKS

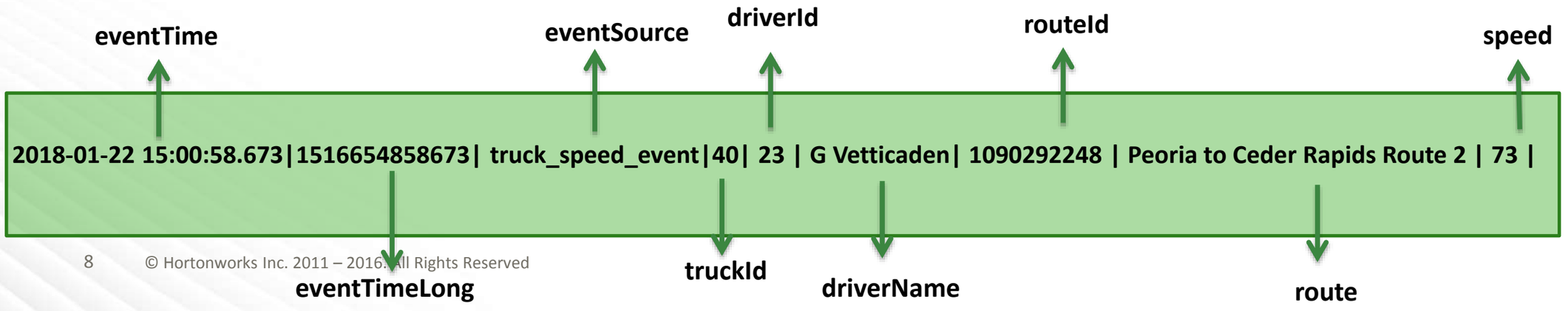
HDF Ref App Data Sources: TruckGeoEvent and TruckSpeedEvent Streams

- Each Truck emits different event stream
  - Truck Geo Event
  - Truck Speed Event

Truck Geo Event:



Truck Speed Event:



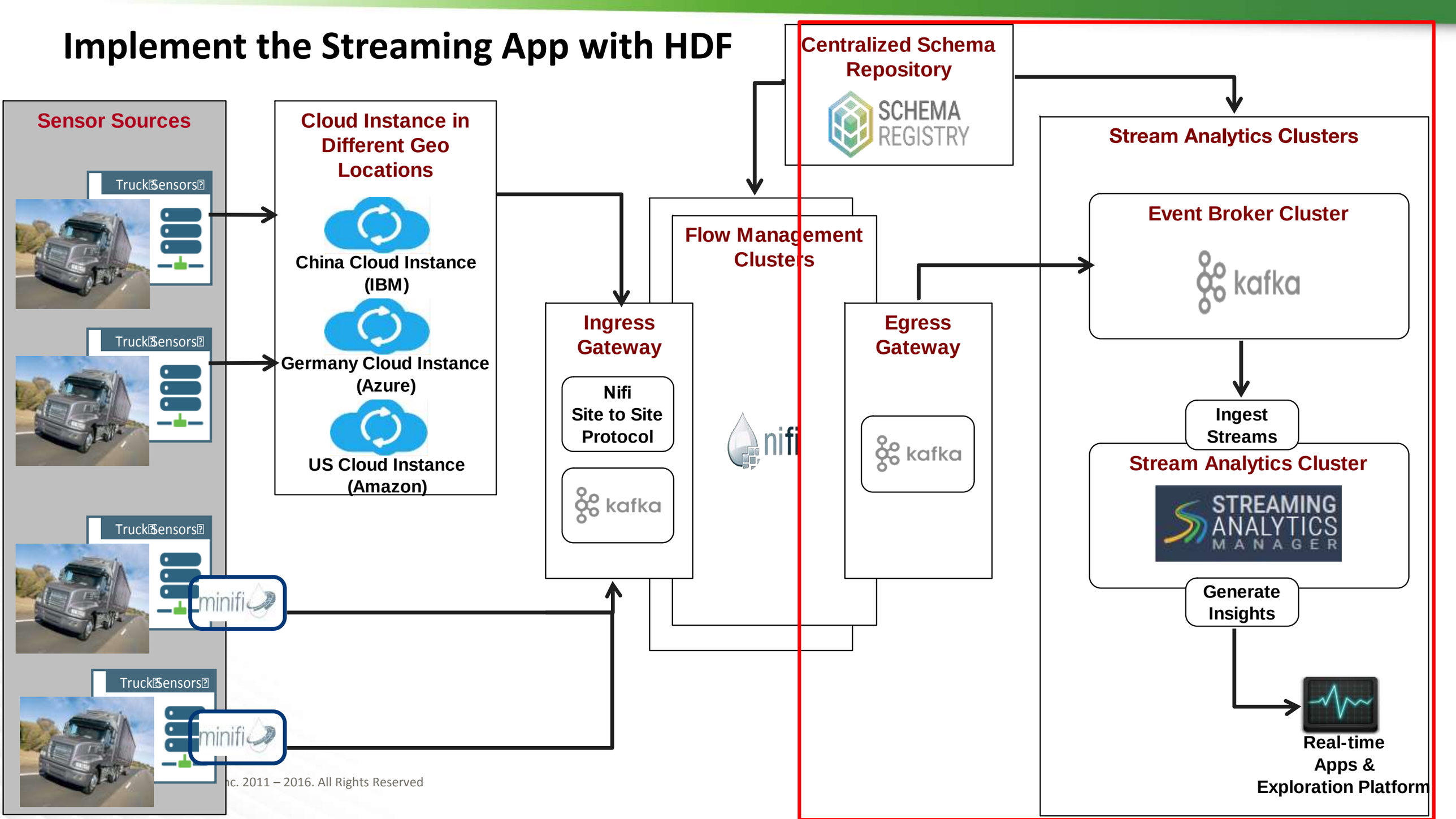
Common Flow Management Requirements

Flow Requirement #	Requirement Description
Req. #1	**Edge deployed data collection service** needs to capture data from the two sensors and stream to an **IOT gateway powered by Apache Kafka.**
Req. #2	The ingestion service will **deliver events in CSV format from each sensor to a Kafka topic** (call it *raw-all_truck_events_csv*) in a **secure cluster**
Req. #3	**Metadata headers** need to be sent with each event like the **schema key that identifies the schema** for the event in a **centralized schema registry store**.
Req. #4	**Consumers** of this raw sensor data need to **inspect the meta headers to lookup the schema** and do **routing, filtering, and enrichment**.
Req. #5	**Producers** need to publish the **enriched streams to their own respective Kafka topics** for consumption for downstream analytics (let's call the Kafka topics for the two streams: *truck_events_avro*, *truck_speed_events_avro*)
Req. #6	**Agility is key**. Developer should be able to create consumers and producers quickly in a code-less approach, preferably UI driven.

Demo Flow Management App

HORTONWORKS

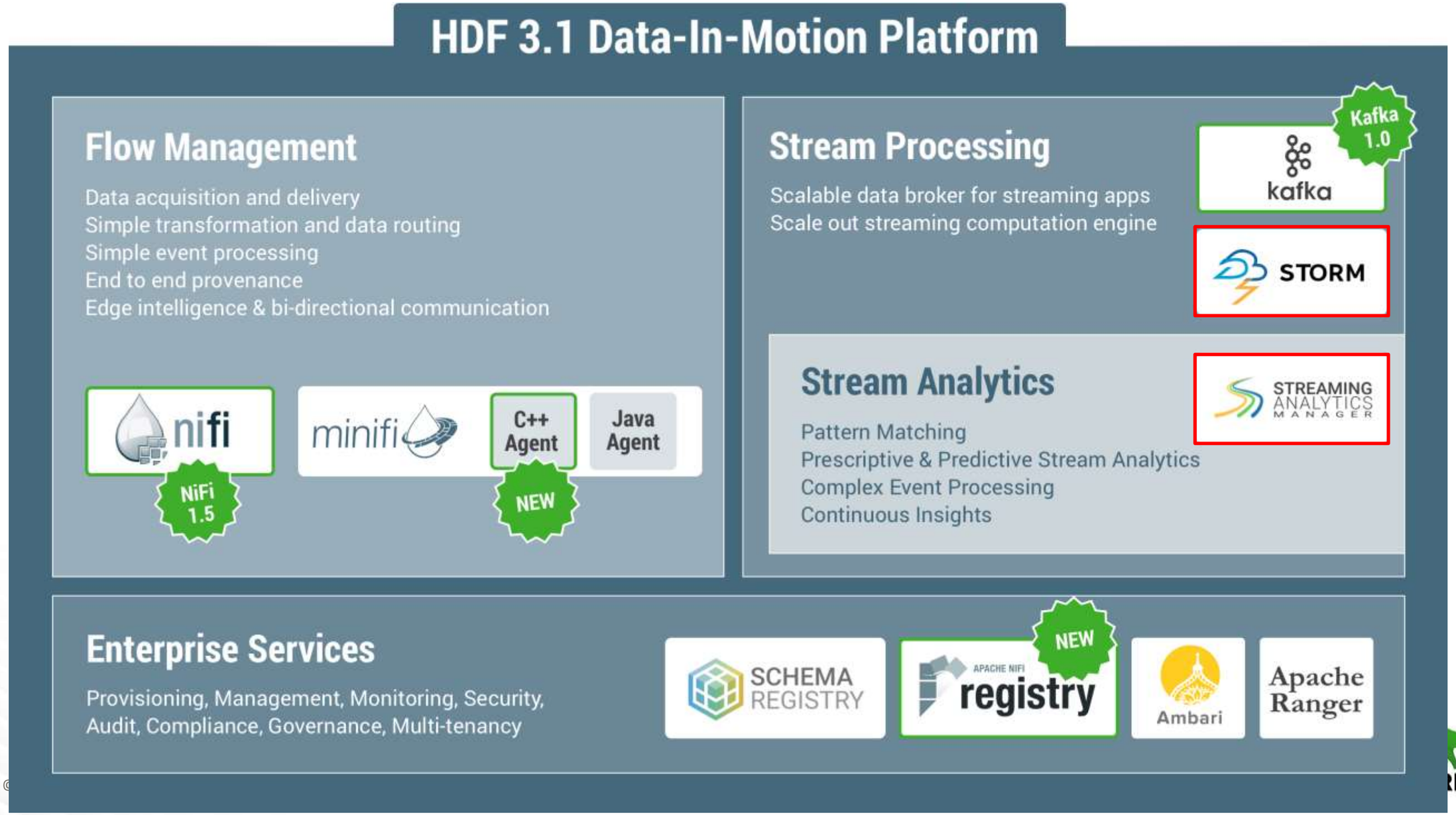
Implement the Streaming App with HDF



Common Streaming Analytics Requirements

Streaming Analytics Requirement #	Requirement Description
Req. #1	**Create streams consuming from the two Kafka topics** that NiFi delivered the enriched geo and speed streams to.
Req. #2	**Join the streams** of the Geo and Speed sensors over a **time based aggregation window**.
Req. #3	**Apply rules on the stream** to filter on events of interest.
Req. #4	**Enrich the stream** with features required for a machine learning (ML) model. The enrichment entails **performing lookups** for driver HR info, hours/miles driven in the past week, weather info.
Req. #5	**Normalize the events** in the stream to feed into the PMML model.
Req. #6	**Execute a predictive logistical regression model on the stream** built with Spark ML to predict if a driver is in danger going to commit a violation.
Req. #7	**Alert and feed into real-time dashboard** if model predicted a violation.

Implementing the Streaming Analytics Requirements with SAM



SAM's Value Proposition

- **Build and deploy complex stream analytics applications without writing any code**
- **Only open source tool in the market** with graphical programming paradigm
- Speed time-to-market for complex **stream apps**
- Build stream analytics apps **without specialized skillsets**.
- Decouple data schema from the streaming application itself
- Support multiple underlining streaming engine

HORTONWORKS

Who Uses SAM?

OPERATIONS

Stream Ops Module

Tooling to manage service pools, environments, deploy and monitor stream apps.

BUSINESS ANALYST

Stream Insight Module

Tooling to generate time-series and real-time analytics dashboards, charts and graphs, and create rich customizable visualization of data from ad/hoc dynamic queries.

APP DEVELOPER

Stream Builder Module

A tooling to build stream analytics app easier with capabilities such as creating input streams, applying aggregate functions over windows, transforms, splitting/ joining streams and persisting streams to downstream system.

Common API that abstracts out the underlying Streaming Engine

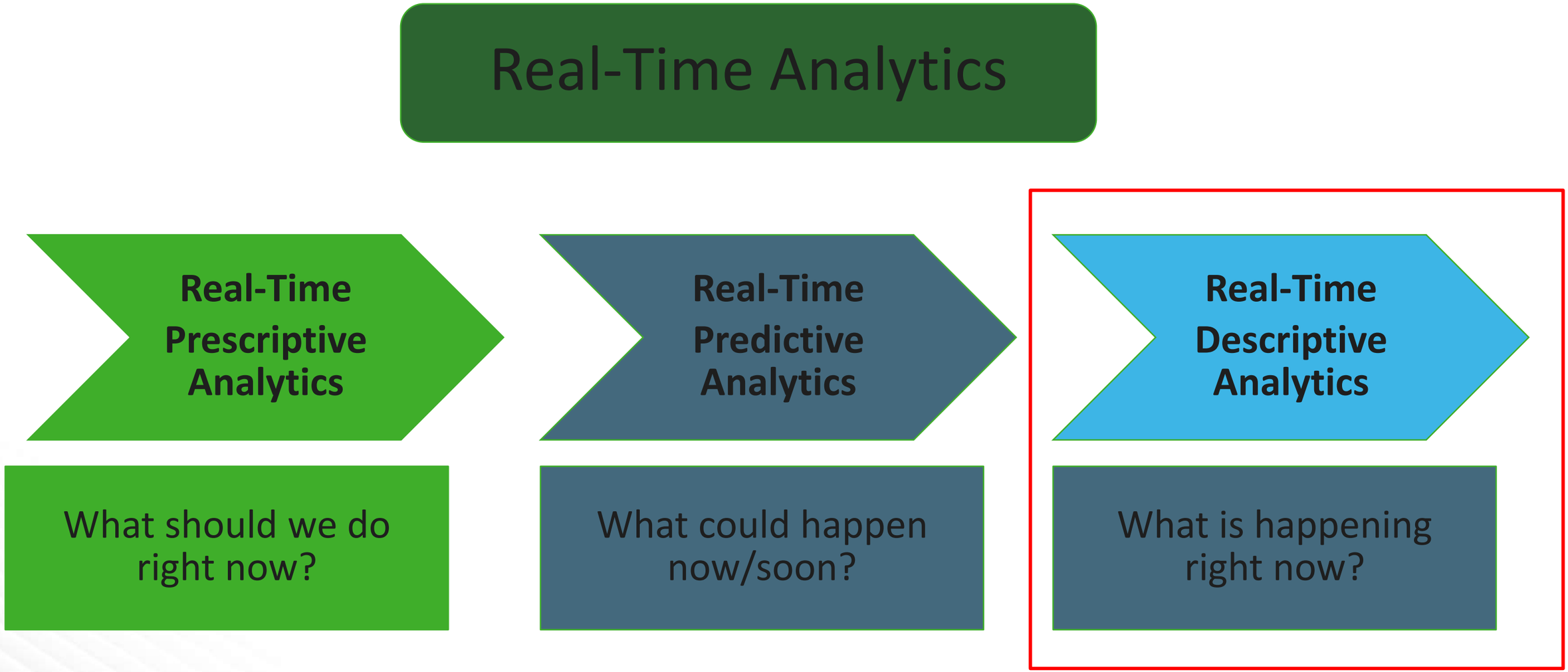
UNIFIED STREAMING API

DISTRIBUTED STREAMING COMPUTATION ENGINE

Different Streaming Engines that powers higher level services to build stream application.

HORTONWORKS

SAM is All about Doing Real-Time Analytics on the Stream

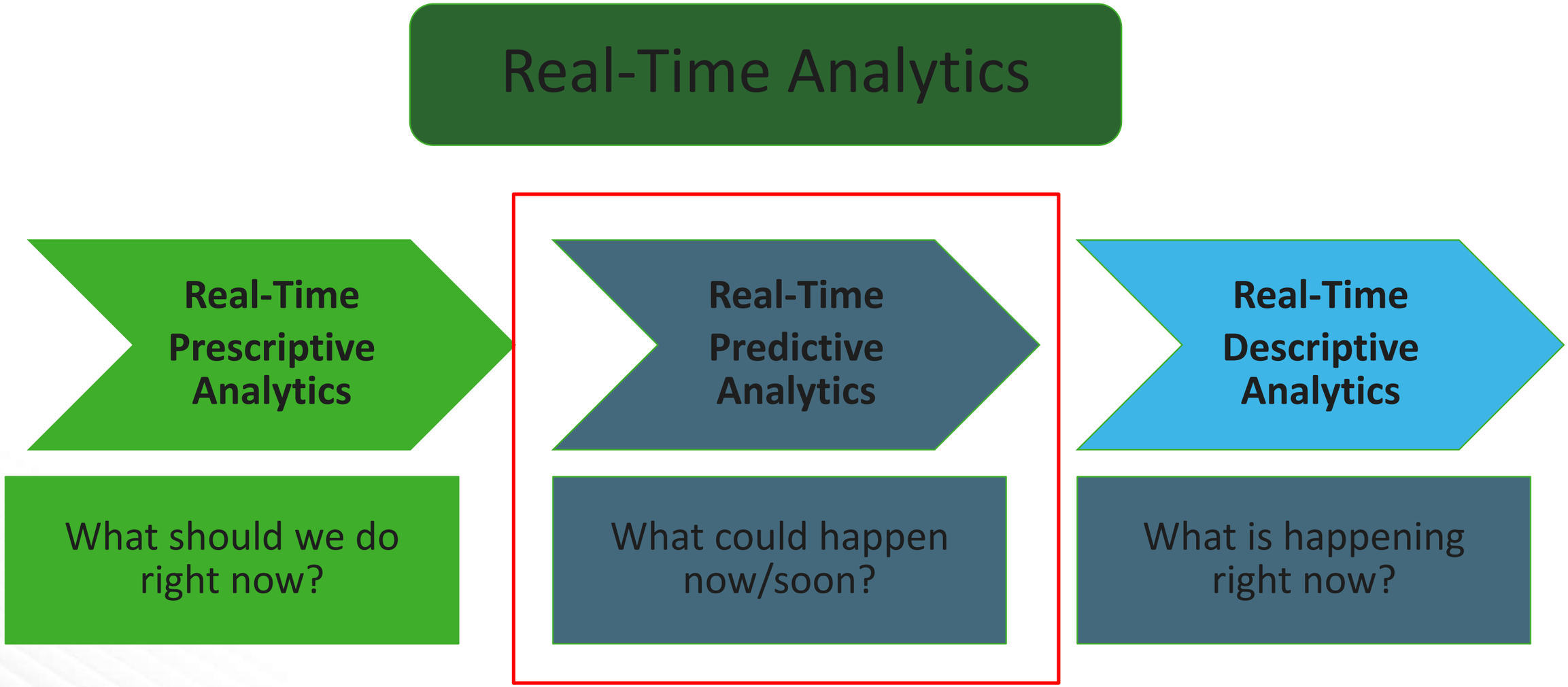


HORTONWORKS

Demo Real-Time Descriptive Analytics

HORTONWORKS

Predictive Analytics with SAM



HORTONWORKS

Predictive Analytics

HORTONWORKS

Real-Time Predictive Analytics

- Question: No violation events but what might happen that I need to be worried about?

- My data science team has a model that can predict that based on
  - Weather
  - Roads
  - Driver HR info like driver certification status, wagePlan
  - Driver timesheet info like hours, and miles logged over the last week

HORTONWORKS

Building the Predictive Model on HDP

1

Explore small subset of events to identify predictive features and make a hypothesis. E.g. hypothesis: *"foggy weather causes driver violations"*

2

Identify suitable ML algorithms to train a model – we will use classification algorithms as we have labeled events data

3

Transform enriched events data to a format that is friendly to Spark MLlib – many ML libs expect training data in a certain format

4

Train a logistic classification Spark model on YARN, with above events as training input, and iterate to fine tune generated model

HORTONWORKS

Logistical Regression Model

Logistical Regression Model to Predict if Driver Will Commit a Violation

```
<PMML xmlns="http://www.dmg.org/PMML-4_1" version="4.1">
  <Header copyright="DMG.org"/>
  <DataDictionary numberOfFields="8">
    <DataField name="Model_Feature_Certification" optype="continuous" dataType="integer"/>
    <DataField name="Model_Feature_WagePlan" optype="continuous" dataType="integer"/>
    <DataField name="Model_Feature_FatigueByHours" optype="continuous" dataType="double"/>
    <DataField name="Model_Feature_FatigueByMiles" optype="continuous" dataType="double"/>
    <DataField name="Model_Feature_FoggyWeather" optype="continuous" dataType="double"/>
    <DataField name="Model_Feature_RainyWeather" optype="continuous" dataType="double"/>
    <DataField name="Model_Feature_WindyWeather" optype="continuous" dataType="double"/>
    <DataField name="ViolationPredicted" optype="categorical" dataType="string">
        <Value value="yes"/>
        <Value value="no"/>
    </DataField>
  </DataDictionary>
  <RegressionModel modelName="Binary Classification for Truck Demo" functionName="classification"
                   algorithmName="logisticRegression" normalizationMethod="softmax"
                   targetFieldName="ViolationPredicted">
    <MiningSchema>
      <MiningField name="Model_Feature_Certification"/>
      <MiningField name="Model_Feature_WagePlan"/>
      <MiningField name="Model_Feature_FatigueByHours"/>
      <MiningField name="Model_Feature_FatigueByMiles"/>
      <MiningField name="Model_Feature_FoggyWeather"/>
      <MiningField name="Model_Feature_RainyWeather"/>
      <MiningField name="Model_Feature_WindyWeather"/>
      <MiningField name="ViolationPredicted" usageType="predicted"/>
    </MiningSchema>
    <RegressionTable targetCategory="yes" intercept="0">
        <NumericPredictor name="Model_Feature_Certification" coefficient="-0.5484931520986547"/>
        <NumericPredictor name="Model_Feature_WagePlan" coefficient="0.32167608426097444"/>
        <NumericPredictor name="Model_Feature_FatigueByHours" coefficient="-0.11878325692728164"/>
        <NumericPredictor name="Model_Feature_FatigueByMiles" coefficient="-0.05352068317534395"/>
        <NumericPredictor name="Model_Feature_FoggyWeather" coefficient="0.7557630499793003"/>
        <NumericPredictor name="Model_Feature_RainyWeather" coefficient="0.5753110023672502"/>
        <NumericPredictor name="Model_Feature_WindyWeather" coefficient="6.491968184728098E-4"/>
    </RegressionTable>
    <RegressionTable targetCategory="no" intercept="0"/>
  </RegressionModel>
</PMML>
```

Input Features to the Model

Details of the Algorithm being used

Output of the model:
yes = Violation Predicted
no = No Violation predicted

Coefficients of the Model

HORTONWORKS

Scoring the Predictive Model on HDF

5  **Model Registry**

Export the Spark Mllib model and import into the HDF's Model Registry

6 Enrich with Features 

Use SAM's enrich/custom processors to enrich the event with the features required for the model

7 Transform/Normalize 

Use SAM's projection/custom processors to transform/normalize the streaming event and the features required for the model

8 Score Model 

Use SAM's PMML processor to score the model for each stream event with its required features

9 Alert / Notify / Action 

Use SAM's rule and notification processors to alert, notify and take action using the results of the model

HORTONWORKS

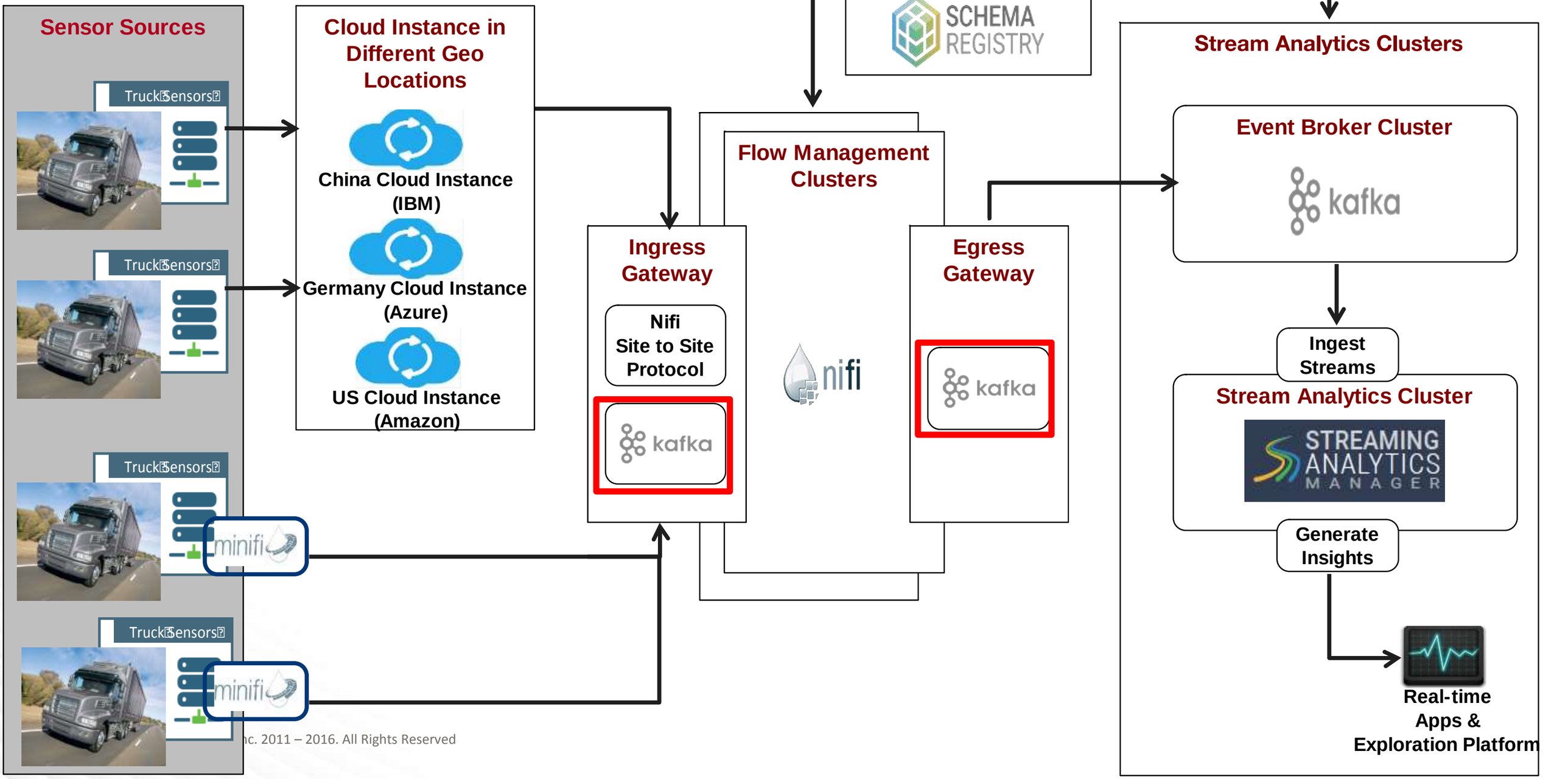
Demo Predictive Analytics with SAM

HORTONWORKS

Atlas Integration

HORTONWORKS

Where did the data come from? Where did it go? Who Produced/Consumed?



Apache Atlas Integration for Flow and Streaming Components

Problem Statement:

- Common ref architectures use all the HDF components including Nifi for Flow and Storm/SAM for streaming analytics with Schema registry as the glue.
- Hence, as data flows through these different components in HDF, **governance requirements such as lineage/provenance, chain of custody, security, audit are key requirements** for every large enterprise.

Solution:

- With HDF 3.1, **Flow and Streaming components will be integrated with Atlas:**
  - **Nifi is integrated with Atlas** so that Atlas contains meta information of Nifi Data Flows including source data and target systems of the flow
  - **SAM/Storm is integrated with Atlas** that contains meta information about SAM App topologies including source data and target systems

Why Should You Care?

- Atlas Integration with HDF components **allows enterprise to meet governance requirement**s allowing users to track data as it travels across the data-in-motion platform (HDF) and into the data-at-rest platform (HDP).

HORTONWORKS

Demo Atlas

HORTONWORKS