

Ready Solutions for AI & Data Analytics

Edge Analytics for Industry 4.0 with Confluent Platform

1.0.0.0

Abstract

This reference architecture provides overview, architecture, and design information for Confluent Platform 5.5 software for deployment on Dell EMC PowerEdge servers and Dell EMC PowerSwitch networking.

Data-Centric Workloads & Solutions

Notes, cautions, and warnings

 **NOTE:** A NOTE indicates important information that helps you make better use of your product.

 **CAUTION:** A CAUTION indicates either potential damage to hardware or loss of data and tells you how to avoid the problem.

 **WARNING:** A WARNING indicates a potential for property damage, personal injury, or death.

Chapter 1: Executive summary	5
Introduction.....	5
IIoT and the edge.....	5
Industry 4.0 and Industrial IoT.....	6
The rise of software in Industry 3.0 and 4.0.....	7
Event streaming.....	7
About Apache Kafka.....	8
About Confluent Platform.....	9
Intended audience.....	9
We value your feedback.....	9
Chapter 2: Predictive maintenance use case	10
Overview.....	10
A use case story.....	10
Starting point.....	11
Proposed solution.....	11
Use case workflows.....	12
Chapter 3: Production infrastructure	13
Traditional approaches.....	13
Alternative approaches.....	14
Integrating existing infrastructure.....	14
Production Kafka	15
MLOps and DevOps.....	17
Chapter 4: Deployment scenario	19
Architecture overview.....	19
Edge environment.....	20
Edge infrastructure.....	20
Data collection.....	21
Anomaly detection and model execution.....	21
Alert generation.....	22
Data flow.....	22
Core environment.....	22
Core infrastructure.....	22
Data flow.....	23
Data lake.....	23
Data science environment.....	23
Production management.....	24
Chapter 5: Infrastructure summary	25
Software.....	25
Hardware infrastructure.....	25
Edge configurations.....	26

Core configurations.....	27
Network configurations.....	29
Chapter 6: Conclusions.....	32
Document summary.....	32
Chapter 7: References.....	33
Dell EMC documentation.....	33
Confluent documentation.....	33
Dell EMC Customer Solution Centers.....	33
Dell Technologies InfoHub.....	34
More information.....	34

Executive summary

There is one undeniable fact regarding Industry 4.0 - data is cheap and plentiful. For over 15 years, all quality pieces of industrial equipment have come out of the crate with onboard telemetry, and one or more network connections. The cost of data availability for that inventory has sunk. For older equipment and commodity peripherals, it has either already been retrofitted with data collection sensors or the cost to have them added is trivial.

Topics:

- [Introduction](#)
- [IIoT and the edge](#)
- [Industry 4.0 and Industrial IoT](#)
- [Event streaming](#)
- [About Apache Kafka](#)
- [About Confluent Platform](#)
- [Intended audience](#)
- [We value your feedback](#)

Introduction

The ingesting, moving, processing, and storing of data incurs incremental costs that are sizeable for any project with the potential for significant return on investment. If data was streamed for all sources from every piece of equipment and sensor in a modern industrial facility, disk storage appliances could fill up faster than they could be ordered and installed. But to what end? It is unlikely that creating a fully populated data lake in the context of Industry 4.0 would produce a positive return on investment (ROI).

This understanding leads to being able to define perhaps the biggest challenge of Industry 4.0 - data curation. How can an organization filter the sea of data already available in order to identify the streams of information that will help produce a better ROI? The areas of possibility include reducing cost, reducing safety risk, or improving productivity to name a few. However, most organizations have not yet identified an efficient method to get data from the vast population of industrial Internet "things" to a destination where the right people and tools can be applied to extract useful insights.

Then, after data curation is solved, there are two additional steps in the value chain that must be put in place. What tools and systems are required to support data scientists and software application developers so they can create intelligent cyber-physical systems that are based on the curated data? And, lastly, how can the developers get their software and models deployed near the sources of the data required to feed inference and decision support. This reference architecture starts with the assumption that data is cheap and plentiful, and from there sheds insight on these significant questions.

There are no silver bullets in an environment as complex as Industry 4.0. Therefore, no single technology or solution exists that solves the challenges of building and deploying cyber-physical systems for every class of problem. The best solutions are built with technology that is enough to be used for multiple applications and use cases. This reference architecture focuses on the use of Kafka and the Confluent Platform.

Kafka is a proven technology platform that has found success in many large-scale enterprise and Internet scale applications. Although the early Kafka successes were mainly in the realm of enterprise IT and application integration, the use of Kafka in both Industry 4.0 and consumer IoT environments has been getting more attention in the last few years. Dell EMC believes that the Confluent Platform can have a significant role for building a complete end-to-end data analysis value chain for Industry 4.0 applications.

IIoT and the edge

A concrete definition of "the edge" given the complex landscape of IoT environments is elusive. Rigid definitions of IoT that imply a universal architecture that is built from technologies like digital sensors, networks, and AI is unhelpful as it hinders the analysis of alternative system architectures that are needed for unique use cases like Industrial IoT (IIoT).¹ The only thing that can universally be agreed on is that when the "things" are stationary they define the outer boundary of the edge.

¹ Boyes, H., Hallaq, B., Cunningham, J., & Watson, T. (2018). The industrial internet of things (IIoT): An analysis framework. Computers in Industry, 101, 1-12.

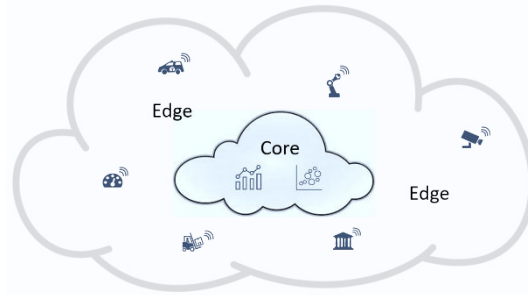


Figure 1. IIoT edge and core

For IIoT, the edge was initially defined as a band technology that includes sensors and controllers along with some gateways, switches, and even routers for data collection and communication. Recently edge devices are also starting to be associated with some types of data processing technology used for analytics. How broad this new band is, and what is included, is up to each group of architects designing systems.

For most use cases, it is advantageous to explicitly define the boundaries of the edge. That definition will help make clear the connections necessary to connect to other layers such as a core or cloud data center. The IIoT use case in this document, using the Confluent Platform, uses two zones:

- The edge
- A core data center

This document shows how data that is generated in the edge flow to the core, and how data models developed in the core flow back to the edge. Anyone designing cyber-physical systems for IIoT applications should be wary of jumping on the "all analytics are moving to the edge" bandwagon too quickly. There are many good reasons to design for the smallest footprint edge possible, especially for organizations that have many geographically dispersed manufacturing facilities. The use case that is described shows how applying sophisticated data filtering and analytics close to the data sources can have significant operational benefits.

Dell EMC also designed this solution to take advantage of centralized investment in corporate IT data centers and data science personnel. This design follows a pattern of keeping the edge "thin" by concentrating big data analytics technology and skilled staff in a central location in a core data center.²

This use case demonstrates hosting an anomaly detection model inferencing against near real-time data at the edge. Data for model training was moved from the edge to the core for use by a data science team for model development. New versions of the fully trained model are then moved back to the edge as needed. This procedure involved:

- Selecting and routing data from the edge to the core for training.
- Designing a method for moving the trained model artifact back to the edge.

The next two chapters have more to say about how this procedure was accomplished.

Industry 4.0 and Industrial IoT

This topic traces the progress from Industry 1.0 to Industry 4.0, and describes the promises and challenges of Industry 4.0.

The progression of industrial advancement can be divided into four broadly themed periods:

- Industry 1.0** The Industrial Revolution was the transition in Europe and the United States from largely agrarian economies to new manufacturing-based economies. This revolution, starting around 1760, was largely driven by the invention of the steam engine.
- Industry 2.0** The Second Industrial Revolution, starting around 1840, was characterized by the move from steam power to electrification for assembly line methods, and a focus on productivity.
- Industry 3.0** The Third Industrial Revolution, or Digital Revolution, was characterized by the introduction of computers and electronics for automation, starting in the late 1950s.
- Industry 4.0** Today, the world is in the Fourth Industrial Revolution, which is characterized by the rise of digital artificial intelligence (AI) and data-driven processes.

The big promise of Industry 4.0 is that the rapidly expanding list of "smart" initiatives includes not only smart factories, but also to name a few:

² Sahal, R., Breslin, J. G., & Ali, M. I. (2020). Big data and stream processing platforms for Industry 4.0 requirements mapping for a predictive maintenance use case. *Journal of Manufacturing Systems*, 54, 138-151. <https://doi.org/10.1016/J.JMSY.2019.11.004>

- Smart cars
- Smart homes
- Smart buildings
- Smart cities

The Fourth Industrial Revolution creates a vision for the future that is dependent upon solving a bold set of technology development and integration challenges. Realization of the Industry 4.0 vision requires at a minimum:

1. Integrating new smart equipment with onboard digital processing capability, native telemetry collection, and easily integrated into the fabric of existing networking and older technology.
2. Developing and deploying sensors and network connectivity that operational technology (OT) staff can easily deploy and manage, for traditional physical equipment lacking "smart" technology.
3. Improving the existing architectures for connecting people and assets in the data center (IT) with people, technology, and data at the edge (OT).
4. Creating software applications using embedded data models that are applied to the Industry 4.0 problem domain, by interdisciplinary teams of industry professionals and data scientists.
5. Establishing ROI metrics for when and how to invest in new Industry 4.0 capabilities that included uncertainty and risk.

The progress in both initiative 1 and 2 above has been impressive. The number of deployed smart machines, additional sensors for data collection, and automated processes has improved productivity and safety in almost every industrial setting. This trend continues and expands as ROI benefits are documented, and older equipment and facilities are replaced.

The focus of this document is primarily initiatives 3 and 4 above. Continued progress on new software applications that use data driven models requires a cost-effective way to get data off the edge and into a data center. The data center is where data is accessible to specialized personnel and tools. Most industrial facilities can deliver a virtual tsunami of data. However, only a fraction of the available streams of telemetry is used regularly. There must be investment that is focused on improving data management between the edge and core or cloud before initiative 4 above can be realized.

The commercial software industry and the open-source software communities have made tremendous advancements in big data storage solutions and data science tools for model development. The movement of data from source to "processing plant", and the movement of applications that are developed in a central plant back to the edge, are complex problems for most organizations developing Industry 4.0 capabilities. The tools and techniques used by the Internet scale technology companies like Uber, Facebook, Google, and Microsoft are now accessible to the pioneers of Industry 4.0. The biggest remaining challenge is building better bridges between the data producers at the edge and the big data technology in the data center.

The rise of software in Industry 3.0 and 4.0

Over 250 years elapsed between the inventions of the steam engine and the first modern digital computer - the defining technologies for Industry 1.0 and Industry 3.0. Advances in the design of the physical stock that is used in industry drove many of the productivity gains throughout those early periods.

With the advent of Industry 3.0, software became a new source of automation control and productivity gains. The importance of the programmable logic controller, which was introduced with the digital era, is still preeminent today. Hundreds of new models are available, with dozens of programming languages, and readily available books and training.

Industry 4.0 is even more software-driven, but now the focus is on the interface of cyber systems to physical equipment. All of the technology that is required to transform factories into smart factories is available now.

The exact cost and ROI of transforming industrial facilities is uncertain. So, the cost of large-scale replacement of traditional software systems with Industry 4.0 systems would certainly be too great and disruptive. The transition to Industry 4.0 will be incremental. Any new systems must interface with many of the software systems providing functionality in response to requirements defined some 20 to 40 years ago.

Industrial plants of every age that are trying to maintain high uptime availability and low operational overhead must determine if the cost is worth the ROI.

Event streaming

As noted in [Introduction](#) on page 5, a modern industrial facility can generate many TBs to PBs of data per day. The largest contributors to this massive data generation potential come from machine-generated log files and numeric telemetry. Log files typically parse nicely along "event" boundaries that include a timestamp, source, and message along with other metadata. These events may be forwarded to an upstream logging function one at a time or be written to a local file and processed in batches. For systems that only sporadically generate events, the logging system design must be reliable but can use a relatively simple scale up approach. When the devices produce prolific amounts of log message and there are many devices, a more complex scale-out computing and storage design is required. These systems are commonly called event streaming platforms.

Event streaming platforms can also be used for continuous time series of numeric telemetry. This scenario typically involves some preprocessing before sending the data upstream. Most telemetry data are produced as byte streams without any clear event boundaries. Every stream will typically have a frequency of data recording. Data streams where the interval between data points is one second or more is often called "1 sec", "5 sec", "30 sec", and so on. Many IIoT devices produced data recordings with thousands of points per second. For this class of data, we use frequency designation data points per second in kHz. For example, 1,000 recorded points per second would be called a "1kHz" data stream.

Many types of IIoT require high frequency data for analysis since the significant changes in levels happen so quickly that one second or longer recording intervals would hide the details necessary to assess the health of the equipment or process. When processing high-frequency streams with an event streaming platform, the data must be preprocessed into time "windows" that look like events to the receiving function. Developers must understand the performance and reliability characteristics of the event streaming platform as well as the network link between the data generation process and the destination receiving function in order to make good choices about window size parameters. High-velocity data may be temporarily stored in a local queue and then sent for processing in "mini-batches" to cut down on communication overhead. However, this process would most likely be considered a data streaming analysis so long as the latency between generation and processing is low.

Some event streaming platforms have integrated analytics technology that can reduce or eliminate the need for integration with other technology. Integrating analytics with the event processing platform is one of the biggest changes that has led to the differentiation of streaming platforms from traditional messaging systems. The most flexible event streaming platforms support both in-process data analytics and developer APIs capable of calling out to remote systems for other types of analytics needs.

About Apache Kafka

Kafka development began primarily to address several limitations of "messaging" systems, including queuing and the publish-subscribe patterns that are already well known and widely used by enterprise software architects. It is easy to see the relationship to prior architectures. The unit of data in Kafka is a still called a message, and Kafka messages are organized into topics that contained ordered lists of messages. Data Producers publish their messages to a topic, and Consumers subscribe to a topic. It is not hard to understand why even the newest versions of Apache Kafka are still compared with traditional messaging systems.

The goals for developing Kafka involved more than just creating a "better" messaging system. Even with the similarities noted above, Kafka has several important advancements that have contributed to the creation of a new class of software called streaming platforms. Kafka improved and extended the prior best practices of messaging systems by defining:

1. A modern distributed cluster architecture for enterprise scale and reliability consistent with other platforms in the "big data" open-source ecosystem
2. A scalable and reliable storage with replication and configurable persistence settings
3. Stream processing services that support near real-time analytics and derived streams

The flexibility of a Kafka-based streaming platform supports many streaming architectures. Common architectural patterns include microservice interfaces for existing or legacy systems, including:

- Ingestion of big data into analytics clusters for batch processing (HDFS, MapReduce, Spark, Hive, and so on)
- Streaming extract, transform, load (ETL) for platform integration and change data capture
- Integration of microservice-oriented business applications (for example, credit card processing, and electronic health record updates)
- IIoT sensor data processing, analytics, device management

This reference architecture is focused on applications for IIoT only. It focuses on replication, reliability, and stream processing for analytics at the edge.

The Kafka Streams API is a powerful, lightweight library that enables real-time data processing. Confluent KSQL is an alternative open source, stream processing API with SQL-like programming and storage. It provides an easy-to-use, yet powerful, interactive SQL interface for stream processing on Kafka. KSQL is scalable, elastic, fault-tolerant, and it supports a wide range of streaming operations, including:

- Data filtering
- Transformations
- Aggregations
- Joins
- Windowing
- Sessionization

MirrorMaker is a related open-source project that provides geo-replication support for Kafka clusters. MirrorMaker replicates events across multiple data centers or cloud regions. Common scenarios include:

- Backup and recovery data protection using active/passive copies
- Creating multiple copies of events to support data locality requirements using active/active copies

For this IIoT solution, Dell EMC chooses to perform replication from the edge to the core using Confluent Replicator. Confluent Replicator is a more complete solution that handles topic configuration and data, and integrates with Kafka Connect and Control Center to improve availability, scalability, and ease of use.

About Confluent Platform

Confluent Platform enables data science and IT practitioners to collaborate on building real-time data pipelines and streaming applications, by integrating data from multiple sources and locations into a single, central event streaming platform. Confluent Platform simplifies connecting data sources to Kafka, building applications with Kafka services, and securing, monitoring, and managing Kafka infrastructure.

Confluent, the company that develops and supports Confluent Platform, was founded by the original creators of Apache Kafka. At the core of Confluent Platform is Apache Kafka, widely reported to be the most popular open source distributed streaming platform. The [Powered by Apache Kafka website](#) lets users share an overview of their Kafka implementations with other interested parties.

Confluent Platform integrates three sources of code and tools:

1. Key components of the Kafka open-source project
 - a. Kafka Brokers
 - b. Kafka Java Client APIs
2. Confluent commercial features
 - a. Confluent Control Center
 - b. Confluent Operator
 - c. Confluent Replicator
 - d. Confluent Auto Data Balancer
 - e. Confluent JMS Client
 - f. Confluent MQTT Proxy
 - g. Confluent Security Plugins
3. Kafka community features
 - a. ksqldb
 - b. Confluent Connectors
 - c. Confluent Clients
 - d. Confluent Schema Registry
 - e. Confluent REST Proxy

Many of the important features useful for IoT applications are covered in later sections of this document. Details on all these components can be found at the [Confluent online documentation](#).

Intended audience

This document is intended for data center managers and IT architects who are involved with engineering, operation, or planning for Confluent Platform on Dell EMC infrastructure.

This document assumes some familiarity with Confluent Platform capabilities and functions.

We value your feedback

Dell EMC and the authors of this document welcome your feedback on the solution and the solution documentation. Contact the Dell EMC Solutions team by [email](#) or provide your comments by completing our [documentation survey](#).

Authors: Dell Technologies Data-Centric Workloads Engineering and Technical Marketing teams

 **NOTE:** For links to additional documentation for this solution, see the [Dell EMC Solutions InfoHub for Data Analytics](#).

Predictive maintenance use case

Use case scenarios that are associated with Industry 4.0 are categorized into one of three areas: intelligent products, intelligent processes, and intelligent machines.³

Topics:

- [Overview](#)
- [A use case story](#)
- [Starting point](#)
- [Proposed solution](#)
- [Use case workflows](#)

Overview

Intelligent machine scenarios focus on performance improvement, improving output quality, and arguably the most lucrative area - preventing unplanned outages through data model driven predictive maintenance. Since the inception of Industry 4.0, the goal of estimating remaining-useful-life (RUL) remains elusive.^{4, 5}

Data model driven predictive maintenance requires the integration of Industrial IoT (IoT) data management technologies with data science capabilities that are the vision of Industry 4.0. This use case was constructed to exercise all the important components that are required to:

1. Collect data from devices on the edge.
2. Route that data to the core.
3. Build a predictive model in the core.
4. Deploy a model back to the edge for near real-time evaluation of new data.

A use case story

Many models are used to estimate the remaining useful life for a system or component. These models rely on data collection techniques to detect and record the occurrence of data patterns that are not typically observed during normal operations. These patterns are considered anomalies.

This use case focuses on the operation of an "hypothetical" industrial furnace. It uses synthetic temperature data that is generated using a software program to control the mix of normal data vs. anomalies. That data is then used to ensure that the detection model is working as expected.

Dell EMC created a scenario as a story that is simple to understand, yet its solution requires developing all the important components of a realistic data analytics project. That project spans systems on the edge and in a data center. The story uses the following assumptions:

- The customer has multiple production lines in a single location that all require furnaces to preheat material at several stages of the process. An unplanned outage of one of these furnaces would stop the production line. Hot material in the process would cool during an outage and require expensive remediation to prepare the line for a restart.
- One of the customer's furnace manufacturers has notified them of premature failures being reported in one of the models that are used in this plant. It is thought that failures are preceded by unexpected drops in temperatures during warm-up. Since the furnace is used over a wide range of temperatures for different processes and materials, the manufacturer cannot provide a hard threshold to observe.

³ Sahal, R., Breslin, J. G., & Ali, M. I. (2020). [Big data and stream processing platforms for Industry 4.0 requirements mapping for a predictive maintenance use case](#). *Journal of Manufacturing Systems*, 54, 138-151.

⁴ Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008, October). Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management* (pp. 1-9). IEEE.

⁵ Bengtsson, M., & Lundström, G. (2018). On the Importance of Combining "the New" with "the Old"—One Important Prerequisite for Maintenance in Industry 4.0. *Procedia Manufacturing*, 25, 118-125.

- The plant operators want the data science team to develop a data model to detect unexpected temperature patterns. This model helps determine if the equipment in the plant should be taken offline for repair or replacement. The decision on which approach to take depends on the number and severity of any detected anomalous behaviors.

Starting point

The customer has recently deployed Confluent Platform at the edge and in the data center core with stream replication from the edge to the core. The plan is to collect data from the furnaces that are potentially faulty by:

1. Streaming data into the Kafka brokers at the edge.
2. Allowing replication to manage the transfer to the data center.

Once the data has been reliably captured in the data center, a Kafka to HDFS connector populates a data store for the data analytics team. The final data management step is to delete the furnace data from the data center Kafka topics.

The data science team and OT staff at the plant must agree on how the model will be used to scan new data for anomalies after the training process is complete. Many data science models never get deployed into production when the application developers and the data science team skip this planning step. The data science team and OT staff must agree on the data in the stream, and how to account for missing or zero-value data.

One of the reasons that the customer chose the Confluent Platform data streaming solution is the availability of stream processing with customized functions close to the data source. For scenarios where OT staff must be alerted quickly for any signs of rapidly declining component health status, they prefer to scan and filter the data for warning signs at the edge.

Confluent Platform uses the Kafka KSQL functions to enable simplified stream processing in near real time. KSQL implements a familiar data access language that is familiar to developers with Structured Query Language (SQL) experience. The KSQL execution engine translates the query into a job that matches the streaming platform topology. It then performs the action without requiring the developer to know the platform internals. Confluent Platform also supports the development of user-defined functions (UDF). This use case uses a design pattern example, developed by Confluent, for implementation of a Deep Learning UDF for anomaly detection.^{6 7}

Proposed solution

There are many potential approaches to developing a model-based approach to data filtering for detecting anomalies.

The Confluent-developed example above uses an open-source framework from [H2O.ai](#) to create the models. Dell Technologies recently completed a white paper for the PowerEdge Reference Architecture series titled [Unleash the power of AI using H2O.ai on Dell EMC infrastructure optimized for machine learning](#).

For this use case, Dell EMC chose to use a deep learning model that is developed in Python using Keras with a TensorFlow backend.⁸ The use of deep learning for anomaly detection is an evolving area of research beyond the scope of this document. It is therefore important to highlight the opportunities and challenges in implementing uses cases that support the Industry 4.0 vision - merging data science, IT and OT.

For deployment to production, data scientists can save an H2O model as a Plain Old Java Object (POJO) or Model Object, Optimized (MOJO). This feature integrates well with the programming environment that is supported for KSQL on the Confluent platform. These artifacts are not tied to a specific version of H2O because they are plain Java code. As such, they do not require an H2O cluster to be running for use in inferencing.⁹

For this use case, Dell EMC chose the [MLFlow open source platform](#) to manage the machine learning life cycle, including:

- Experimentation
- Reproducibility
- Deployment
- A central model registry

The primary deep learning models that are used for anomaly detection use an encoder and decoder pattern, which is known as an autoencoder. The prior "state of the art" for automated anomaly detection used supervised machine learning or statistical regression models. The detection tool is trained to distinguish among two pre-labeled classes (healthy and unhealthy).¹⁰

⁶ [How to Build a UDF and/or UDAF in KSQL 5.0](#); March 13, 2020

⁷ [KSQL UDF Java Implementation \(Deep Learning for Anomaly Detection\)](#)

⁸ [TensorFlow Core Documentation for Keras](#)

⁹ <https://docs.h2o.ai/h2o/latest-stable/h2o-docs/save-and-load-model.html#saving-loading-downloading-and-uploading-models>

¹⁰ Borghesi, A., Bartolini, A., Lombardi, M., Milano, M., & Benini, L. (2019, July). Anomaly detection using autoencoders in high performance computing systems. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 9428-9433).

Autoencoders are trained with only normal (healthy) behavior eliminating the requirement to find and label anomalies before the model can be deployed. The trained autoencoder can then be used to identify any abnormal conditions. When anomalies occur in the data stream, the model cannot accurately encode the input data. It then decoded the values back to a close approximation of the original state.

The use of a Keras and TensorFlow model also fits this scenario. Keras supports saving a single HDF5 format file containing the model's architecture, weights values, and compile information. That information can be used with a Java application through KSQL UDFs. HDF5 is a light-weight Keras model output format, and is an alternative to the more recent SavedModel format.¹¹

Use case workflows

In summary, the use case project that Dell EMC runs in the engineering lab must accomplish the following workflows:

1. Configure Kafka brokers in the plant (edge) to receive data from the simulated equipment sensors.
2. Use the Confluent Platform replication tools to move data from the edge to the Kafka cluster residing in the data center.
3. Write selected data from the Kafka cluster topics to an HDFS data store accessible to data scientists.
4. Delete data from the appropriate Kafka core cluster after validating it is persisted in HDFS.
5. Develop and train a deep learning autoencoder using the data that are acquired from one or more edge sensors.
6. Export the trained model in a format usable on the edge Kafka infrastructure to scan new data from the furnaces that may develop problems based on the information that is received from the manufacturer.
7. Raise alerts to OT staff in the plant when anomalies are detected.
8. Continue to stream data to the data center Kafka cluster and on to HDFS after it is scanned at the edge if further modeling or analysis is required.

¹¹ TensorFlow Guide

Production infrastructure

The proliferation of data gathering sensors and programmable logic controllers were the fuel for digital transformation during Industry 3.0. Data management drove automation for industrial facilities that were dominated by Industrial Automation and Control Systems (IACS or ICS). Most of the devices, systems, networks, and controls used to operate or automate industrial processes were highly specialized and local to each plant. These systems are still the automation backbones in many diverse industries including manufacturing, transportation, and utilities. Before Industry 4.0 and the potential to implement more sophisticated cyber-physical systems, these systems only had to meet the needs of the Operational Technology (OT) organization.

Topics:

- [Traditional approaches](#)
- [Alternative approaches](#)
- [Integrating existing infrastructure](#)
- [Production Kafka](#)
- [MLOps and DevOps](#)

Traditional approaches

The manufacturing industry is trying to increase their analytics capability by extending the deployment of familiar technology for application and data integration. These technologies are based on traditional data messaging and queuing technology, primarily MQTT.

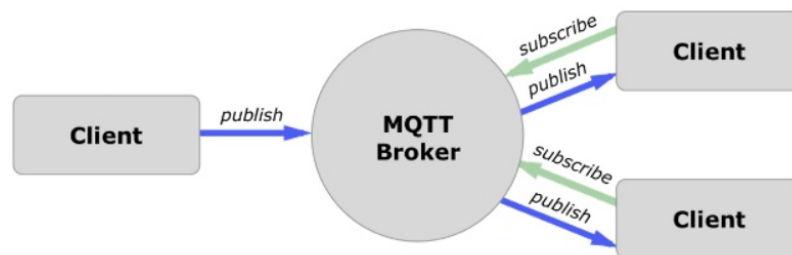


Figure 2. MQTT

The advantages of MQTT include:

- Broad industry experience
- The ability to handle unreliable networks
- The availability of many open-source broker and client packages
- Lightweight footprint for easy deployment on many types of devices

The design of MQTT predates the massive deployment of data generating and consuming devices that are encountered when implementing Industry 4.0. There are some significant limits to the technology. The ability to work asynchronously for long periods of network unavailability limit its usefulness for near-real time applications. Many enterprise applications have limited support for MQTT, hindering its ability to be a common data exchange technology. Many MQTT broker implementations have limited scalability, and therefore do not handle systems with many clients or large bursts of messages.

A second approach for data management and integration is based on a "hub and spoke" approach. The hub of the data storage and access is frequently called a data historian. PR Newswire predicts that the data historian software market will grow to \$1.2 billion by 2023.¹² The data historian software category was also developed long before the advent of Industry 4.0. There is a wide array of vendor offerings in the market with only a small number with a significant market share. These products were typically evaluated, purchased, and operated by the OT organization without significant integration with IT systems. This lack of integration capability is the single largest challenge with data management strategies that are built around data historians. Many of the same challenges that IT has had implementing large-scale data lakes also apply to data historians.

¹² <https://www.prnewswire.com/news-releases/data-historian-market-worth-1-271-1-million-by-2023-845765143.html>

The market and products have evolved over time. Data Historians have expanded into many different industries and applications where time series data is prevalent. They have evolved from single site, on-premises implementations to multisite cloud-based implementations. An ecosystem of System Integrators and software providers has emerged to implement these systems and create value added applications on top of Data Historians.

As with any software application category that is over 35 years old, pundits have predicted the demise of Data Historians since before the advent of Industry 4.0. The consensus that is emerging is that Data Historians will remain important to process engineers and OT for a long time. However, they are unlikely to solve any of the new challenges being surfaced in the move to developing more advanced analytics. Better data integration between OT and IT silos is required. It must be dynamic, and better integrated with technologies from the world of big data and data science.

Alternative approaches

The adoption and deployment of Industry 4.0 concepts such as cyber-physical systems are driving the requirement to move beyond traditional messaging systems. Building on the IACS architecture pattern by adding more point-to-point integrations is increasing complexity and operating cost. A new architectural pattern that is geared toward data curation and routing is an essential next step in getting to Industry 4.0 cyber-physical systems development. Some of the organizational and technology challenges that are outlined above are relevant to the consumer Internet of Things (IoT). However, the industrial sector is unique in many ways and therefore deserves a dedicated branch of design for Industrial IoT (IIoT).¹³

Meeting the wide range of application and integration diversity in IIoT scenarios is a major challenge that developers face today.¹⁴ There are many middleware options available for general-purpose enterprise application integration that are positioned for IoT and IIoT despite not being designed for those purposes. A better approach that is gaining traction is the use of event streaming platforms that were designed for the big data ecosystem. These platforms are built to take advantage of scale-out designs using either physical or virtualized resources.

Integrating existing infrastructure

Most applications that were developed and deployed during the Third Industrial Revolution were focused on improving automation and control. In those cases, where a new automation required integration with multiple systems, the communication was implemented only for those applications. It was rare for OT to invest in standardized enterprise class platforms for integration. The only option for process integration without a true platform option was to keep adding on to an evolving set of point-to-point message-based communications.

Development of each additional integration connection was done by itself as new equipment and applications were added. This approach kept things running with the least short-term disruption but has become overly complex and cannot support the data integration needs of Industry 4.0.

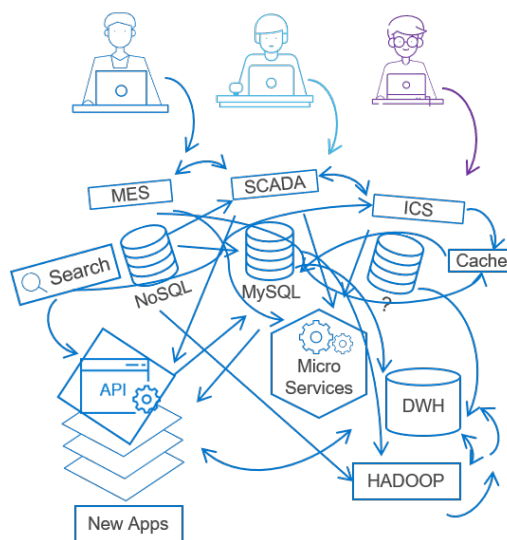


Figure 3. Complex legacy integration

¹³ Leber, J. (2017). General Electric's San Ramon Software Center Takes Shape MIT Technology Review [online],(2012) Available: <https://www.technologyreview.com/2012/11/28/114725/general-electric-pitches-an-industrial-internet/>. Accessed September, 8.

¹⁴ Agarwal, P., & Alam, M. (2020). Investigating IoT Middleware Platforms for Smart Application Development. In Smart Cities—Opportunities and Challenges (pp. 231-244). Springer, Singapore.

The manufacturing sector is not alone in confronting a complex process integration environment that has developed over decades, and that does not readily support cross-functional analytics. Also, the health care industry has historically preferred to adopt many "best-of-breed" software applications. That adoption has resulted in data integration challenges, especially since the advent of IoT for patient care. The "best of breed" solutions for different hospital functions, all linked with multiple point-to-point integration connections to manage operations and control, include:

- Intensive Care Unit monitoring
- Pharmacy
- Surgery
- Imaging
- Physical therapy
- And many more

System downtime in health care is at least as costly as the manufacturing industry. Also, the health care industry must manage distributed application and data processing with several large corporate IT centers. These IT centers are connected to many regional and local hospitals. The health care industry has also realized the potential benefits from new advanced analytics and AI applications. These applications must leverage communication across the individual edge sites and processes within a site with connections to and from central data centers.

Given the many parallels, it is worth studying the data management and advanced analytics approach that many health care companies have adopted. The industry has made significant progress over the last 20 years in reducing the complexity of application integration. Progress has also been made in the incorporation of new sources of data from IoT devices, by implementing message-based middleware.¹⁵

The use of message-based middleware in the industrial sector would have many parallels to health care. The typical industrial plant will have many "production lines" that may share some equipment, raw material inventory, packing and shipping services. Separate control systems, with isolated data management, control all these components.

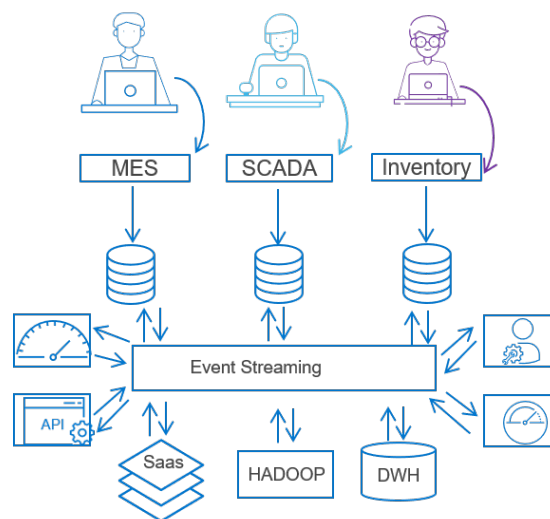


Figure 4. Simple modern integration

As with health care, the switch to a central messaging or event streaming platform is incremental usually starting with two or at most a few systems. The risk of system disruption that would come with converting all integrations to run through a central platform is too great. The flexibility of the Kafka platform in handling both integration using traditional messaging and high velocity streaming has been valuable in both health care and manufacturing applications.

Production Kafka

Kafka was designed for scale-out performance and high availability, and is based on the same principles that were used to develop big data platforms like the Hadoop ecosystem. The most basic production deployment that has support for high availability is a single-site, three-broker design, as shown in the following figure.

¹⁵ Bellagente, P., Depari, A., Ferrari, P., Flammioni, A., Sisinni, E., & Rinaldi, S. (2018, May). M 3 IoT-message-oriented middleware for M-health Internet of Things: Design and validation. In 2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) (pp. 1-6). IEEE.



Figure 5. Three-broker Kafka deployment

This design pattern can be an appropriate middleware design for a single in-plant event streaming platform. The system can be deployed and managed by Operational Technology (OT) staff. This middleware enables the conversion from point-to-point integrations towards a central service capable of handling many systems as both producers and consumers using a common framework.

Organizations that integrate data from computing and storage technology at the edge with core systems in one or more IT-managed core data centers require a more sophisticated design. Multiple-instance, cross data center Kafka implementations are now common. Many distributed computing uses cases require zero data loss and high availability service level agreements.

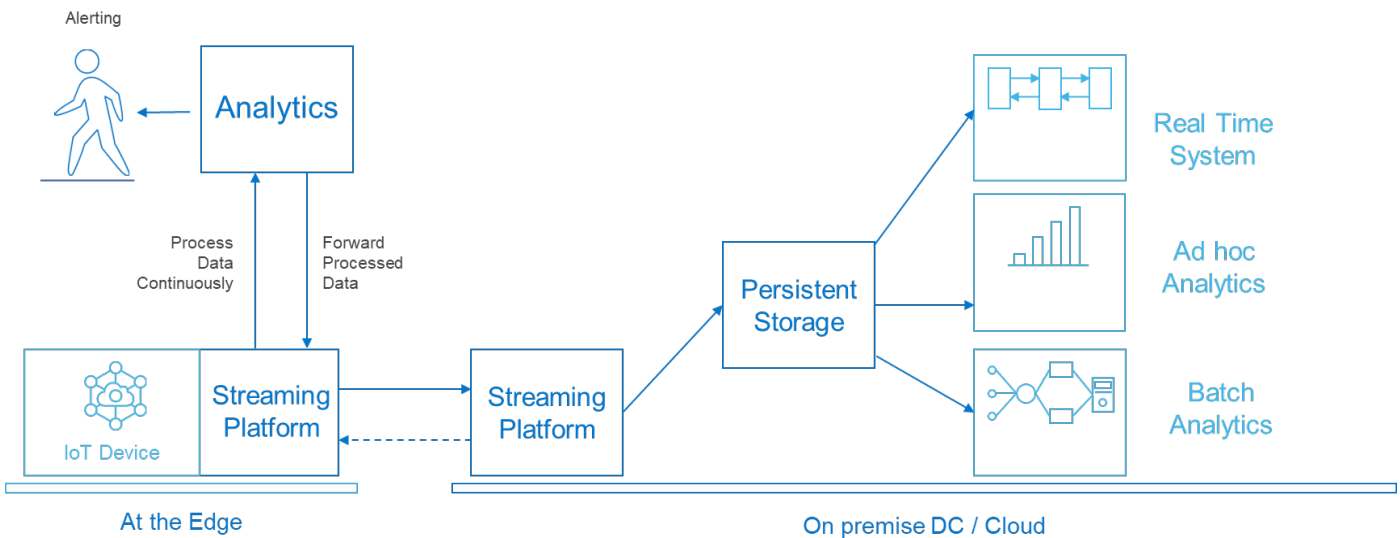


Figure 6. Cross-data center streaming deployment

The design that is shown in the previous figure matches what Dell EMC deployed in its engineering lab to test the anomaly detection use case described in [Predictive maintenance use case](#) on page 10. Details are in [Deployment scenario](#) on page 19. This design can be extended to include more than two sites using common deployment, configuration, replication and high availability features at all sites. See the following figure.

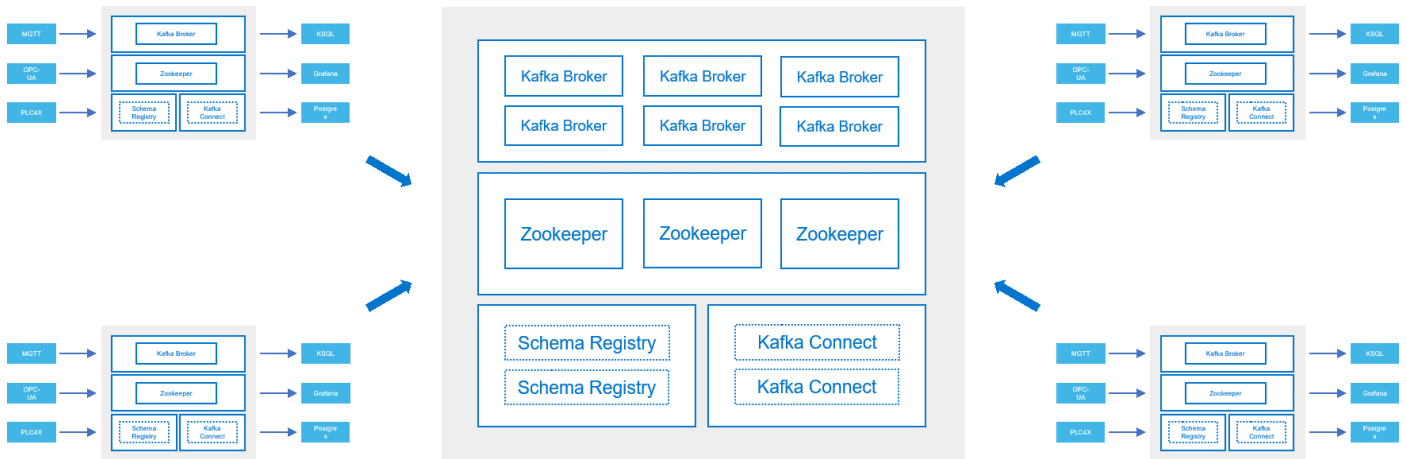


Figure 7. Multiple-site streaming deployment

MLOps and DevOps

The practice of Development Operations (DevOps) emerged from the challenges that arose in deploying new or updated software applications when the software development and IT operations teams functioned almost independently. The situation is often described as a process where software developers would build applications and "throw them over the wall" to the IT staff that are expected to deploy the software and support users. In order to understand how this scenario connects to data science and machine learning, a data model must be defined.

In 2018, the Wall Street Journal published an opinion editorial titled "Models Will Run the World".¹⁶ The author concluded that the impact of software development on business, commonly called the Digital Transformation, was ready to transition to the next phase of maturity. That level is intelligent applications with embedded machine learning technology. Intelligent applications that incorporate the predictive power of these data analytics models is one of the most common forms of applied artificial intelligence (AI). These same concepts are being developed in this transition from the last industrial revolution. This transition is fueled by digital transformation to the current industrial revolution, and depends on developing intelligent cyber-physical systems that are based on AI principles.

Machine Learning Operations (MLOps) is an attempt to build on the success of Dev/Ops while recognizing that managing the deployment of production class machine learning models has key differences. These differences depend on how tightly integrated the model is with the application or service that uses it. The following figure shows a loosely coupled application where the model serving is hosted on a separate platform from the service that uses the model. In this diagram, the service calling into the model is a process running unattended on an event streaming platform.

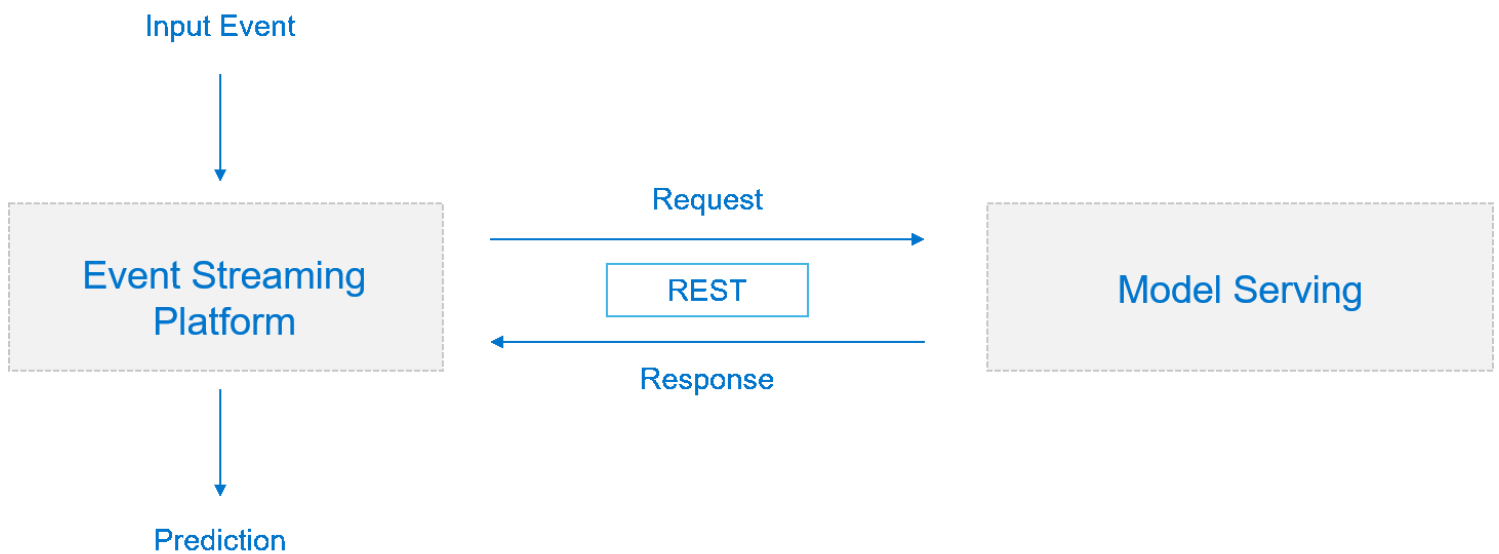


Figure 8. Loosely-coupled application

¹⁶ Cohen, Steven A. and Granade, Matthew W. (April 2018) Models Will Run the World <https://www.wsj.com/articles/models-will-run-the-world-1534716720>

The languages and toolsets that are used for the streaming service and model hosting are decoupled using a lightweight REST interoperability layer. The requirement to deploy and manage two independent platforms adds to the complexity of this model. The complexity increases if this solution is deployed at an edge location where it is desirable to have the smallest maintenance footprint possible.

The deployment of a tightly coupled application and model is at the other end of the spectrum. Java is still the language with the most support for integrating machine learning and application development. There are several modeling frameworks that support exporting trained models into Java classes that can be linked into an application project before compilation. Data scientists will typically use Python, R, or Scala for data preparation, model development, and model training.

Many Python and R machine learning libraries have features for exporting trained model objects that can be imported and used in other server environments. This approach requires carefully managing both the language and model library dependencies on the target platforms. Container virtualization and management tools make this job easier, but there are still other challenges that must be addressed. Applications that are developed in compiled languages must access these model objects that will run in a separate process on the server. This approach adds more dependencies for libraries that support the remote calls and has significant performance overhead. These examples are a few of many issues that lead to unreliable and failed deployment of models into production environments. These challenges are especially complicated when working with distributed application environments with few support resources at the remote sites like IIoT.

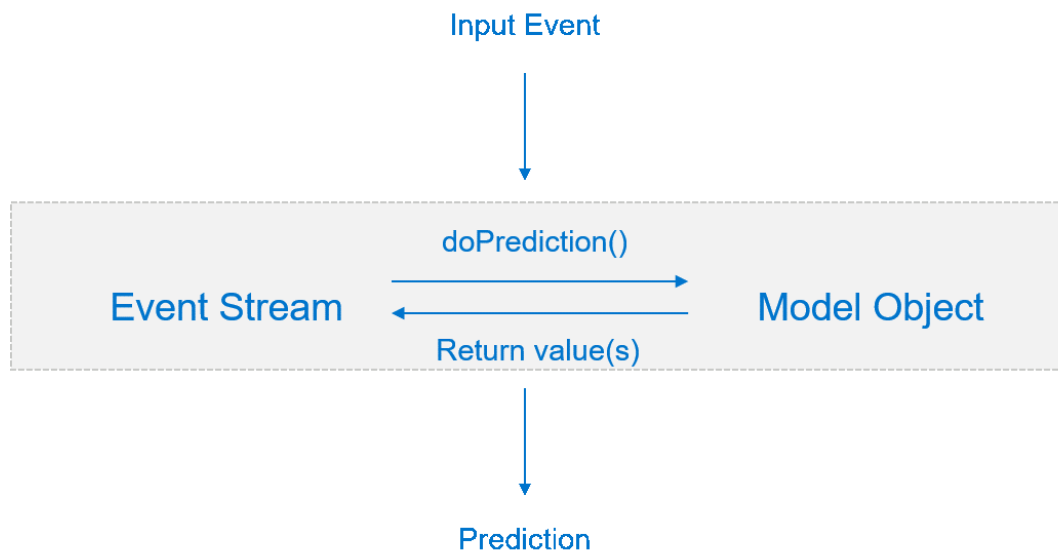


Figure 9. Trained model

After the model is trained with a framework that supports creation of Java objects from trained models, the data scientist "exports" a model object (the structure and parameters of the model) to a text formatted Java class. Before some frameworks added this feature, software developers would "hand code" the model structure with embedded parameters before building the final application.

Deployment scenario

The architecture of Ready Solutions for AI & Data Analytics – Edge Analytics for Industry 4.0 with Confluent Platform is designed to support Industry 4.0 deployment scenarios in industrial facilities.

Topics:

- [Architecture overview](#)
- [Edge environment](#)
- [Core environment](#)

Architecture overview

This architecture uses an edge and core model to address the following requirements

- Support for multiple edge locations
- Support for multiple pieces of equipment of varying types at each edge location
- Anomaly detection at the edge, while disconnected from the core
- Flexible handling of alert notifications
- Support for a data science environment integrated with the data collection system
- Scalable data collection and storage

The following figure shows the overall edge to core architecture. Confluent Platform is used to provide the data streaming capabilities from the edge to the core and the management of that system. Red Hat OpenShift Container Platform is used for the runtime environment at the core, including Confluent Platform. The data science environment uses multiple components running under OpenShift, and a Dell EMC Isilon storage array provides the data lake storage.

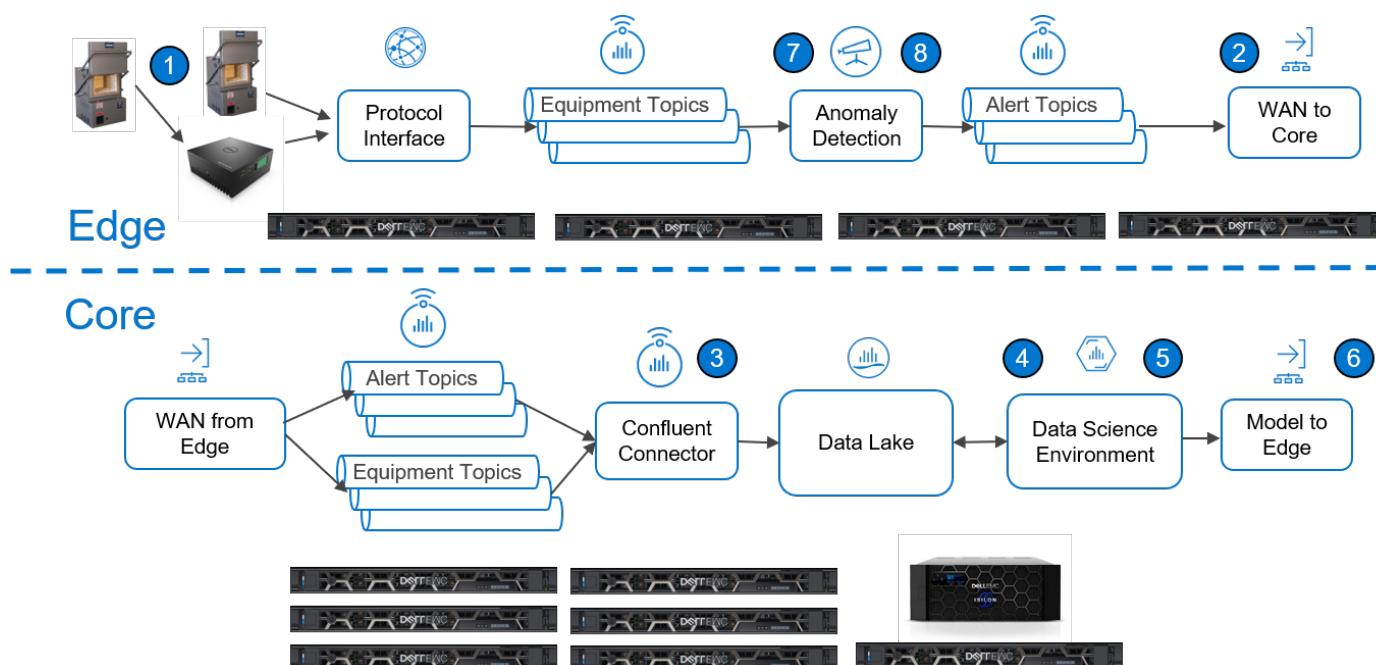


Figure 10. Architecture overview

The following table provides a legend for the figure above.

Table 1. Architecture overview legend

Callout	Description
1	Data collection from the edge locations
2	Data flow to the core
3	Flow to the data lake
4	Data science model development
5	Production model management
6	Model deployment from the core
7	Anomaly detection with machine learning (ML)
8	Alert generation

Edge environment

The figure in [Edge connectivity](#) on page 20 shows the primary functional components at the edge.

Edge infrastructure

The edge environment runs under Red Hat Enterprise Linux Server or CentOS. The recommended infrastructure for the edge environment is covered in [Edge configurations](#) on page 26.

Kafka cluster infrastructure

The edge environment runs the Confluent Platform directly under Linux. Three broker nodes are recommended at the edge to provide high availability for the Kafka topics. A single broker can be used to minimize cost at the expense of high availability. This single broker configuration may be appropriate for environments with many edge installations where overall cost is more important than reliability of a single edge site. A single edge installation can support many pieces of equipment by scaling the number of broker nodes.

The Kafka topic data is stored directly on local storage in each broker node. The amount of storage that is required depends on:

- The number of monitored systems
- The amount of data generated from the system
- How much data should be retained at the edge
- How long the data should be retained

The architecture is designed to forward the data to the core. The retention time can be relatively short, and only must provide enough storage to queue data when the interface to the core is temporarily unavailable.

The correct sizing for a specific environment can be calculated from the [Capacity Planning](#) section of the Confluent Platform.

Edge connectivity

The architecture allows the edge to operate independently of the core. All the necessary functions for anomaly detection and alert generation are hosted at the edge. The edge requires connectivity to the core for replication of edge data, forwarding of alerts, and updates to the machine learning models.

The details of the connection are site and installation specific. The connection should have enough bandwidth to support the volume of data coming from all the systems in the equipment topics, plus any aggregated streams created at the edge. In general, the connection is not latency sensitive since alerts are processed locally. If alerts must be forwarded to the core, then latency becomes a consideration.

All connections between the edge and the core should be firewalled for security.

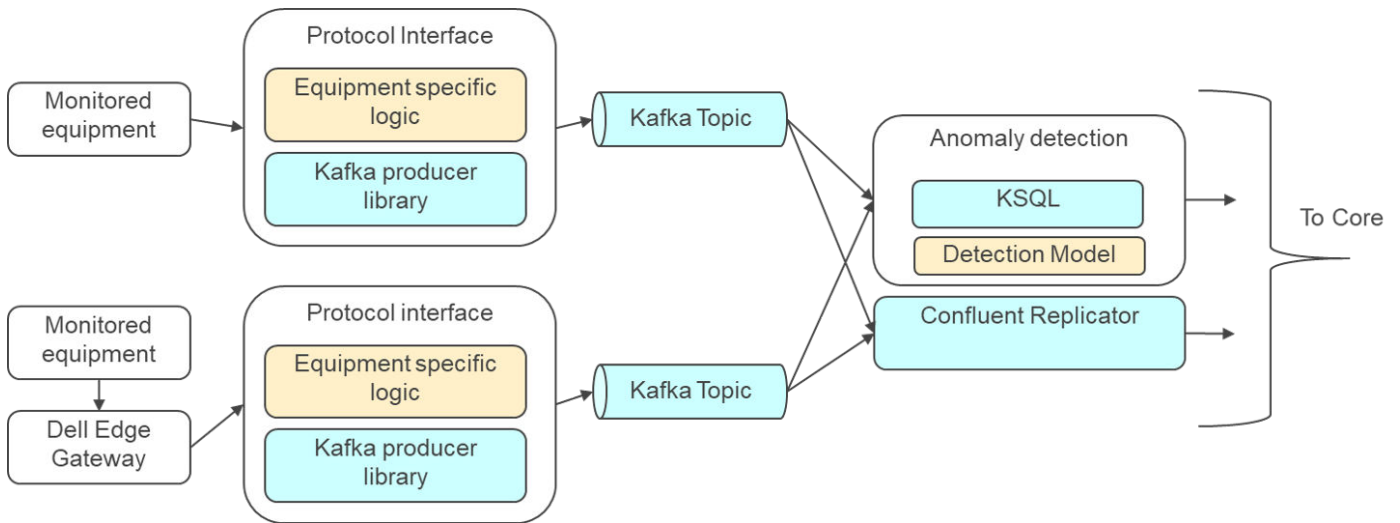


Figure 11. Edge environment

Data collection

Data collection from the monitored equipment is specific to the type of equipment, the interfaces and data formats available, and the parametric data necessary for the detection models. The collection and formatting of the data is performed in the protocol interface blocks. This step will typically involve some equipment-specific software to collect the data and perform any necessary formatting and conversion. This software can use the Kafka producer library to pass the data into a Kafka topic for further processing. Depending on the specific telemetry interfaces supported by the equipment, a Dell Edge Gateway may be necessary to support the physical hardware interfaces. The Dell Edge Gateway 3001 provides multifunction general-purpose I/O (GPIO) hardware support, while the Dell Edge Gateway 3002 provides a CANbus interface for industrial equipment that supports the CANbus standard.

After the data has been collected, formatted, or converted as appropriate, the data is passed to a Kafka topic for each individual machine using the Kafka producer library. The data is also converted to Apache Avro format before it is passed to Kafka. Avro is used to provide a compact data representation with a defined schema. The Confluent Schema Registry is used to ensure data compatibility between the edge data format and the rest of the environment.

The lab development environment used a data generator program to simulate the equipment and its telemetry. The data generator was designed to create data, including anomalies, so the machine learning models could be tested. This design also allowed the simulation of multiple pieces of equipment.

Anomaly detection and model execution

Anomaly detection at the edge is done using a machine learning model that is developed in TensorFlow and Keras. Model development is done at the core, and the model is pushed to the edge for execution. Confluent KSQL and its user-defined function (UDF) capability is used to run the model on each data record and generate events for anomalies.

KSQL user-defined functions allow developers to extend KSQL with Java functions including:

- Metadata annotations that inform KSQL of the function definition
- The arguments names and types
- The return value type

User-defined function provides a clean method of integrating the machine learning model with the data streams and support execution on each data record. This method also eliminates the need for a separate model serving and execution capability at the edge.

The UDF is written in Java. It uses the Java API for TensorFlow to load and run the model. The UDF is added to KSQL by copying it to the KSQL extension directory before starting KSQL.

The data from each individual machine topic in Kafka is fed to Confluent KSQL by creating a KSQL stream from the topic. A second KSQL stream is created from the first stream as a `SELECT` that calls a KSQL user-defined function. The user-defined function then runs the model with the stream data and returns the results as a second topic.

Model deployment and updates are accomplished by deploying a new version of the KSQL user-defined function or the model file.

Alert generation

The handling of alerts typically depends on the implementation requirements and must be flexible. In this architecture, the second topic that is created using KSQL contains the data describing any anomalies (that is, the results of the model execution.) It can also contain equipment identification, time references, or any other information relevant to generating an alert. This data can be used in multiple alert scenarios:

- The alert events can be read directly from the topic or by KSQL, and alerts can be generated locally.
- The alert events can be fed to another topic for additional local processing.
- The alert topics can be fed to a topic in the core for additional remote processing.

Any single approach or a combination of these approaches can be used in a specific deployment and may use any features of the stream processing platform.

Data flow

The architecture is designed to support multiple edge locations. Topics from each edge location are copied to the core using Confluent Replicator. Replicator runs in the core, and pulls data from the edges. This design provides data protection and enables analysis of the raw data in the core.

Core environment

Figure 12. Core environment on page 23 illustrates the primary functional components at the core.

Core infrastructure

The core environment runs under Red Hat OpenShift Container Platform (OCP). The recommended infrastructure for the core environment is covered in [Core configurations](#) on page 27.

OpenShift is used in the core to provide the flexibility and scalability that is required to support multiple edge locations. The Confluent Platform components are run under OpenShift using the Confluent Operator for Kubernetes. With the operator, the streaming platform can be managed as a cloud-native application with resiliency and elastic scaling. Scaling the Kafka brokers to handle the load from multiple edge sites becomes straightforward.

OpenShift also runs the various tools and utilities that are needed for the data science environment. This solution provides a large amount of flexibility since most of the widely used machine learning tools that are run as cloud-native applications and in containers.

OpenShift supports the Kubernetes container storage interface (CSI). This design allows many internal and external storage systems to be used for dynamic and persistent storage. The lab environment used a Dell EMC Isilon H600 system, with dynamic storage provisioned using the Network File System (NFS) protocol. Isilon with HDFS was also used to host the data lake and to store the Kafka topic data replicated from the edge locations.

Core connectivity

The architecture uses connectivity to the core from all edges to consolidate data.

The details of the connection are site and installation-specific. The connection should have enough bandwidth to support the volume of data coming all the edge locations. In general, the connection is not latency sensitive since alerts are processed at the edge. Depending on the data rate, oversubscription of the bandwidth may be appropriate in instances where data from the edges occurs in bursts because of production schedules.

All connections between the core and the edges should be firewalled for security.

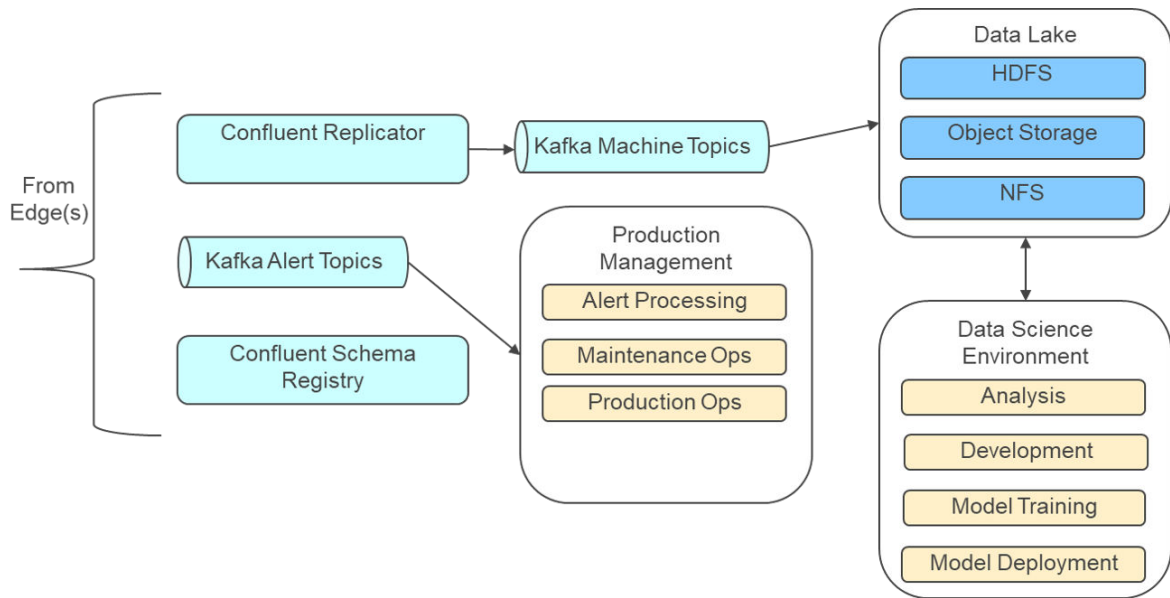


Figure 12. Core environment

Data flow

Data at the edge is stored in Kafka topics. Since Kafka also runs in the core, this design provides a lot of flexibility in how data moves to the core.

In this architecture, Confluent Replicator is used to copy the topic data from the edges to the core. The data is written to an HDFS file system in the data lake, which provides a copy of the edge data for data protection. The data is also available to the data science environment for analysis and model development. In the development lab environment, a custom version of the `cp-server-connect-operator` image from Confluent Operator was needed. This customization added `kafka-connect-replicator`, `kafka-connect-hdfs`, and `kafka-connect-hdfs2-source` to the base image.

The alert topics from each edge location can also be streamed to the core. This information can be used to analyze model behavior and for additional alert processing in the core.

After models have been developed and tested or refined, they can be pushed out to the edges to the KSQL nodes for execution.

Data lake

The architecture supports a data lake to store the data from the edges so it can be analyzed. The data lake is hosted outside the OpenShift environment, and is typically an HDFS compatible file system such as a Hadoop cluster or an Isilon system. The lab testing environment used an Isilon H600 with HDFS. Since HDFS is a network file system protocol, the data lake can be accessed directly from any containers that are hosted in the OpenShift environment without any additional configuration beyond authentication. The HDFS protocol is also supported directly by most data science tools, making it a good choice for the data lake.

Data science environment


The architecture supports a wide variety of machine learning and data science tools. Since the core infrastructure is based on Kubernetes, most of the widely used data science tools are available. The choice of data science tools often depends on developer, data scientist and use case specifics. The architecture deliberately does not select any specific tool chain. Rather, it is designed to allow flexibility and evolution, while allowing the use of modern DevOps and MLOps workflows.

The development lab environment used [MLFlow](#) to manage the machine learning life cycle. It used a PostgreSQL database under OpenShift to host the MLFlow metadata, and HDFS to store MLFlow artifacts. During development, the MinIO S3 compatible storage system was also used for artifacts, with MinIO running under OpenShift.

In the development lab environment, the models were developed using TensorFlow and Keras. Java was used to develop a KSQL user-defined function. The Java API for TensorFlow was used to call the model from the KSQL user-defined function.

Since the Confluent Platform runs in the core, the models and user-defined functions can be tested and validated in the data science environment at the core before they are release to production and pushed to the edge.

Production management

 **NOTE: "Production management" represents the systems used by the operational technology team.**

The architecture supports interfaces to a production management environment for the industrial equipment. Since the production management system is specific to each deployment, the architecture does not specify it. This system can be hosted under OpenShift or can be external to the OpenShift environment.

If the production management system provides an API, alerts and other relevant information can be forwarded to it from the topics using the Kafka Streams API and custom interface programs.

Infrastructure summary

This chapter describes the Dell EMC recommended software and hardware configurations for Confluent Platform.

Topics:

- [Software](#)
- [Hardware infrastructure](#)

Software

Table 2. [Software components](#) on page 25 lists the software components and versions that are supported for Edge Analytics for Industry 4.0 with Confluent Platform.

Table 2. Software components

Category	Component	Version
Operating system - Edge	RHEL	8
Operating system - Container	CoreOS	4.4.9
File system - Edge	XFS	N/A
File system - Container	XFS (/)	N/A
	EXT4 (/boot)	N/A
	VFAT (/boot/efi)	N/A
Confluent	Confluent Platform	5.5
	Confluent Operator	5.5
Container platform	Red Hat OpenShift Container Platform	4.4.9
Java Virtual Machine	Open JDK	1.8.0.252.b09-3.el8_2
Firmware	iDRAC	4.10.10.10
	BIOS	2.5.4
	Network interface card	14.25.10.20
	Dell EMC PERC H330	25.5.6.0009
	Dell EMC PERC H740P	25.5.6.0009
	Dell EMC HBA330	50.5.0-2819

Hardware infrastructure

This section describes the Dell EMC recommended server and network configurations for Confluent Platform. Topics include:

- [Edge configurations](#) on page 26
- [Core configurations](#) on page 27
- [Network configurations](#) on page 29

The Dell EMC PowerEdge XE2420 Edge Server can be used as an alternative to the Dell EMC PowerEdge R640. The Dell EMC PowerEdge XE2420 provides durability, speed, storage capacity, management capability, and serviceability for industrial environments. Contact your Dell EMC sales representative for more information.

Edge configurations

Edge configurations are run on bare metal in each edge or factory location. The configuration consists of a minimum of four machines, and supports an entire edge facility. The cluster design can be scaled up for extra capacity.

Control Center Node

The Control Center Node is used to host Confluent Control Center. The Dell EMC recommended hardware configuration is listed in the following table.

Table 3. Control Center Node configuration

Machine function	Component
Platform	Dell EMC PowerEdge R640
Chassis	2.5" chassis with up to eight hard drives and three PCIe slots
Processor	Intel Xeon Gold 5218 2.3 G, 16C/32T, 10.4 GT/s, 22 M Cache, Turbo, HT (125 W) DDR4-2666
RAM	Six 8 GB RDIMM, 2666 MT/s, single rank
Network daughter card	Mellanox ConnectX-4 LX dual port 10/25 GbE SFP28, rNDC
Boot configuration	From RAID controller
Storage controller	Dell EMC PERC H330 RAID controller, mini card
Disk - SSD	Two 480 GB SSD SATA mix use 6 Gbps 512 2.5 in hot-plug AG Drive, 3 DWPD, 2628 TBW
Drive configuration	C7, unconfigured RAID for HDDs or SSDs (Mixed drive types allowed.)

Platform Node

The Platform Node configuration is used for multiple roles, including Kafka Connect, the Confluent Schema Registry, and the Confluent RESTful API Proxy. The Dell EMC recommended configuration is listed in the following table.

Table 4. Platform Node configuration

Machine function	Component
Platform	Dell EMC PowerEdge R640
Chassis	2.5" chassis with up to eight hard drives and three PCIe slots
Processor	Intel Xeon Gold 6226 2.7 G, 12C/24T, 10.4 GT/s, 19.25 M cache, turbo, HT (125 W) DDR4-2933
RAM	Six 8 GB RDIMM, 2666 MT/s, single rank
Network daughter card	Mellanox ConnectX-4 LX dual port 10/25 GbE SFP28, rNDC
Boot configuration	From SSD
Storage controller	Dell EMC PERC H330 RAID controller, mini card
Disk - SSD	Two 240 GB SSD SATA mixed use 6 Gbps 512e 2.5" hot-plug S4610 Drive
Drive configuration	C7, unconfigured RAID for HDDs or SSDs (Mixed drive types allowed.)

KSQL Node

The KSQL Node is used to host Confluent KSQL. The Dell EMC recommended configuration is listed in the following table.

Table 5. KSQL Node configuration

Machine function	Component
Platform	Dell EMC PowerEdge R640
Chassis	2.5" chassis with up to eight hard drives and three PCIe slots
Processor	One Intel Xeon Gold 6226 2.7 G, 12C/24T, 10.4 GT/s, 19.25 M cache, turbo, HT (125 W) DDR4-2933
RAM	Six 8 GB RDIMM, 2666 MT/s, single rank
Network daughter card	Mellanox ConnectX-4 LX dual port 10/25 GbE SFP28, rNDC
Boot configuration	From RAID controller
Storage controller	Dell EMC PERC H330 RAID controller, mini card
Disk - SSD	Two 480 G SSD SATA mix use 6 Gbps 512 2.5" hot-plug AG drive, 3 DWPD, 2628 TBW
Drive configuration	C7, unconfigured RAID for HDDs or SSDs (Mixed drive types allowed.)

Broker Node

The Broker Node is used to host the Kafka Broker, and provides a balance of compute and storage. The Dell EMC recommended configuration is listed in the following table.

Table 6. Broker Node configuration

Machine function	Component
Platform	Dell EMC PowerEdge R640
Chassis	2.5" chassis with up to 10 hard drives, two 2.5" SATA/SAS drives and one PCIe slot, two CPUs only
Processor	Two Intel Xeon Gold 5218 2.3 G, 16C/32T, 10.4 GT/s, 22 M cache, turbo, HT (125 W) DDR4-2666
RAM	12 8 GB RDIMM, 2666 MT/s, single rank
Network daughter card	Mellanox ConnectX-4 LX dual port 10/25 GbE SFP28, rNDC
Boot configuration	From BOSS controller
Storage controller	Dell EMC PERC H740P RAID controller, 8 GB NV cache, mini card
Boot optimized storage cards	BOSS controller card + with two M.2 sticks 240 G (RAID 1), LP

NOTE: The Kafka partitions should use a reliable storage layer - either RAID 5, RAID 6, or RAID 10.

Dell EMC recommends RAID 6 for the best performance and reliability with the Dell EMC PERC H740P storage controller. RAID 5 provides slightly larger storage capacity but lower redundancy. RAID 10 has lower performance than RAID 6 during normal I/O operations, and when running with a degraded RAID array.

Core configurations

The core runs Confluent Platform underneath Red Hat OpenShift Container Platform, and is designed to scale support for multiple edge facilities. These configurations provide a solid base for a modern analytics environment. The OpenShift environment is large enough to support additional machine learning and MLOps capabilities. The OpenShift cluster is based on the [Dell EMC Ready Stack for Red Hat OpenShift Container Platform 4.3](#).

The minimum OpenShift deployment consists of seven nodes:

- One cluster services administration host (CSAH)
- Three Master Nodes

- Three Worker Nodes

Table 7. OpenShift node descriptions on page 28 describes the Worker Node types.

CSAH Node

The CSAH Node is used to manage the operation and installation of the container ecosystem cluster. It runs the infrastructure services required for the OpenShift platform, including:

- HAProxy
- DNS
- DHCP
- TFTP
- Web server
- PXE server

Configuration details are described in Table 8. OpenShift Dell EMC PowerEdge R640 CSAH Node/Master Nodes configuration on page 28.

Master Node

Master Nodes provide an application programming interface (API) for overall resource management. These nodes run `etcd`, the API server, and the Controller Manager Server.

Configuration details are described in Table 8. OpenShift Dell EMC PowerEdge R640 CSAH Node/Master Nodes configuration on page 28.

Worker Node

Worker Nodes are where the application containers are deployed . A minimum of two Worker Nodes must always be operating.

There are two alternative configurations:

- Table 9. OpenShift Dell EMC PowerEdge R640 Worker Nodes configuration on page 29
- Table 10. OpenShift Dell EMC PowerEdge R740xd Worker Nodes configuration on page 29

Table 7. OpenShift node descriptions

Type	Description	Count	Notes
CSAH Node	Dell EMC PowerEdge R640 server	One	Runs the infrastructure services required for the OpenShift platform including HAProxy, DNS, DHCP, TFTP, web server, and PXE server.
Master Nodes	Dell EMC PowerEdge R640 server	Three	Run the OpenShift control plane services.
Worker Nodes	Dell EMC PowerEdge R640 or Dell EMC PowerEdge R740xd server	Three or more	Run the OpenShift container services.

Table 8. OpenShift Dell EMC PowerEdge R640 CSAH Node/Master Nodes configuration

Machine function	Component
Platform	Dell EMC PowerEdge R640
Chassis	2.5" chassis with up to 10 hard drives, eight NVMe drives, and three PCIe slots, two CPUs only
Processor	Two Intel Xeon Gold 6238R 2.2 G, 28C/56T
RAM	192 GB RAM
Network daughter card	Mellanox ConnectX-4 LX dual port 10/25 GbE SFP28, rNDC
Boot configuration	From SSD
Storage controller	Dell EMC HBA330 controller, 12 Gbps mini card

Table 8. OpenShift Dell EMC PowerEdge R640 CSAH Node/Master Nodes configuration (continued)

Machine function	Component
Disk - NVMe	One to eight Dell 1.6 TB, NVMe, mixed use Express Flash, 2.5" SFF drive, U.2, P4610 with carrier
Disk - SSD	Two 960 GB SSD SAS read intensive 12 Gbps
Drive configuration	RAID 1, JBOD

Table 9. OpenShift Dell EMC PowerEdge R640 Worker Nodes configuration

Machine function	Component
Platform	Dell EMC PowerEdge R640
Chassis	2.5" chassis with up to 10 hard drives, eight NVMe drives, and three PCIe slots, two CPUs only
Processor	Two Intel Xeon Gold 6238R 2.2 G, 28C/56T
RAM	192 GB RAM
Network daughter card	Mellanox ConnectX-4 LX dual port 10/25 GbE SFP28, rNDC
Boot configuration	From SSD
Storage controller	Dell EMC HBA330 controller, 12 Gbps mini card
Disk - NVMe	One to eight Dell 1.6 TB, NVMe, mixed use Express Flash, 2.5" SFF drive, U.2, P4610 with carrier
Disk - SSD	Two 960 GB SSD SAS read intensive 12 Gbps
Drive configuration	RAID 1, JBOD

Table 10. OpenShift Dell EMC PowerEdge R740xd Worker Nodes configuration

Machine function	Component
Platform	Dell EMC PowerEdge R740xd
Chassis	Chassis with up to 24 2.5" hard drives including a maximum of 12 NVME drives
Processor	Two Intel Xeon Gold 6252 2.1 G, 24C/48T
RAM	192 GB RAM
Network daughter card	Mellanox ConnectX-4 LX dual port 10/25 GbE SFP28, rNDC
Additional network card	Mellanox ConnectX-4 LX dual port 10/25 GbE SFP28, network adapter
Boot configuration	From SSD
Storage controller	Dell EMC HBA330 controller, 12 Gbps mini card
Disk - NVMe	One to 12 Dell 1.6 TB, 3.2 TB, or 6.4 TB, NVMe, mixed use Express Flash, 2.5" SFF drive, U.2, P4610 with carrier, CK
Disk - SSD	One to 24 800 GB, 1.92 TB, or 3.84 TB SSD SAS mixed use 12 Gbps 512e 2.5" hot-plug AG drive with carrier
Drive configuration	RAID 1, JBOD

Network configurations

Edge Analytics for Industry 4.0 with Confluent Platform uses two networks - one for an edge or factory facility, and one for the core cluster in a data center:

- [Edge Cluster network](#) on page 30
- [Core Cluster network](#) on page 30

Edge Cluster network

Dell EMC recommends the Dell EMC PowerSwitch S5248F-ON as the rack level switch for deployments. It is a high-density 25/10/1 GbE form factor switch that provides 4.0 Tbps (full-duplex) nonblocking, cut-through switching fabric and delivers line-rate performance under full load.

Networking requirements vary based on the deployment scenario. For single-rack cluster deployments, a single switch is adequate and can support 40 nodes, with spare connections for integration. The following figure illustrates a typical single-rack scenario.

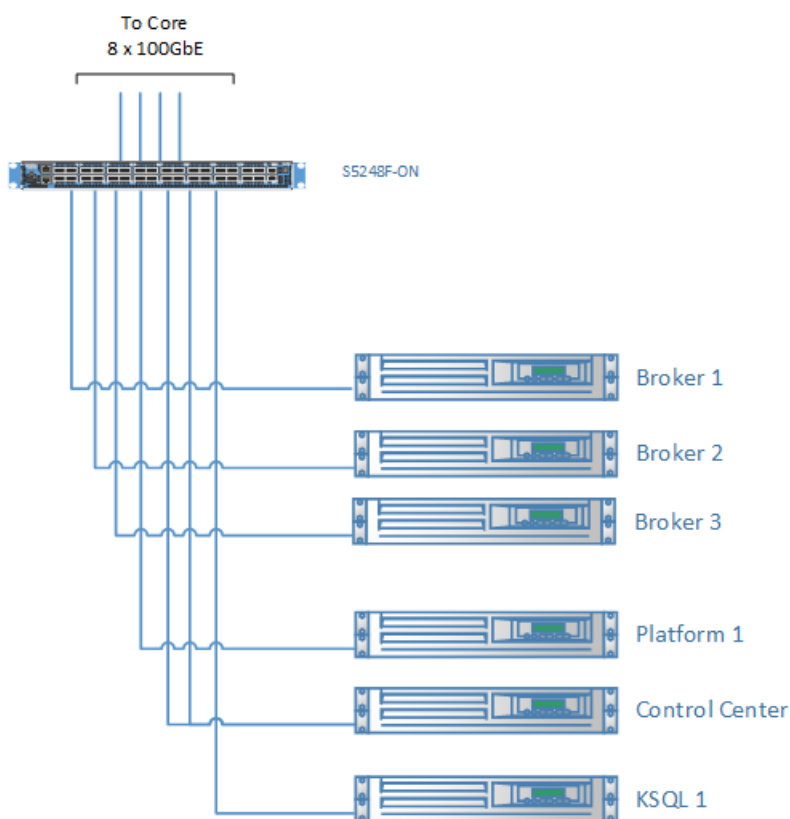


Figure 13. Single-rack Edge cluster network connections

Multi-rack deployments are often integrated into existing network infrastructure. In multi-rack scenarios, nodes from each function should be distributed across racks to increase resiliency and avoid single points of failure. In some application scenarios, using the RESTful API and a load balancer using session affinity ("sticky" sessions) may be required. In other cases, IP rotation or simple load balancing are adequate.

Dell EMC recommends the Dell EMC PowerSwitch S3148-ON for iDRAC or management network interfaces. See the following table.

Table 11. Network configurations

Function	Recommended switch hardware
Top of rack or cluster	Dell EMC PowerSwitch S5248F-ON
Management	Dell EMC PowerSwitch S3148-ON
Load balancing	Hardware load balancer or IP rotation

Core Cluster network

Dell EMC networking products are designed for ease of use and to enable resilient network creation. Red Hat OpenShift Container Platform 4.4.9 introduces various advanced networking features to enable containers for high performance and monitoring. The Dell EMC recommended design applies the following principles:

- Meet the network capacity and segregation requirements of the container pod.

- Configure dual-homing of the OpenShift Platform Node to two Virtual Link Trunked (VLT) switches.
- Create a scalable and resilient network fabric to increase cluster size.
- Provide monitoring and tracing of container communications.

Container network capacity and segregation

Container networking takes advantage of the high speed (25/100 GbE) network interfaces of the Dell Technologies server portfolio. In addition, pods can attach to more networks using available Container Networking Interface (CNI) plug-ins to meet network capacity requirements.

Extra networks are useful when network traffic isolation is required. Networking applications such as Container Network Functions (CNFs) have control traffic and data traffic. These different traffic types have different processing, security, and performance characteristics.

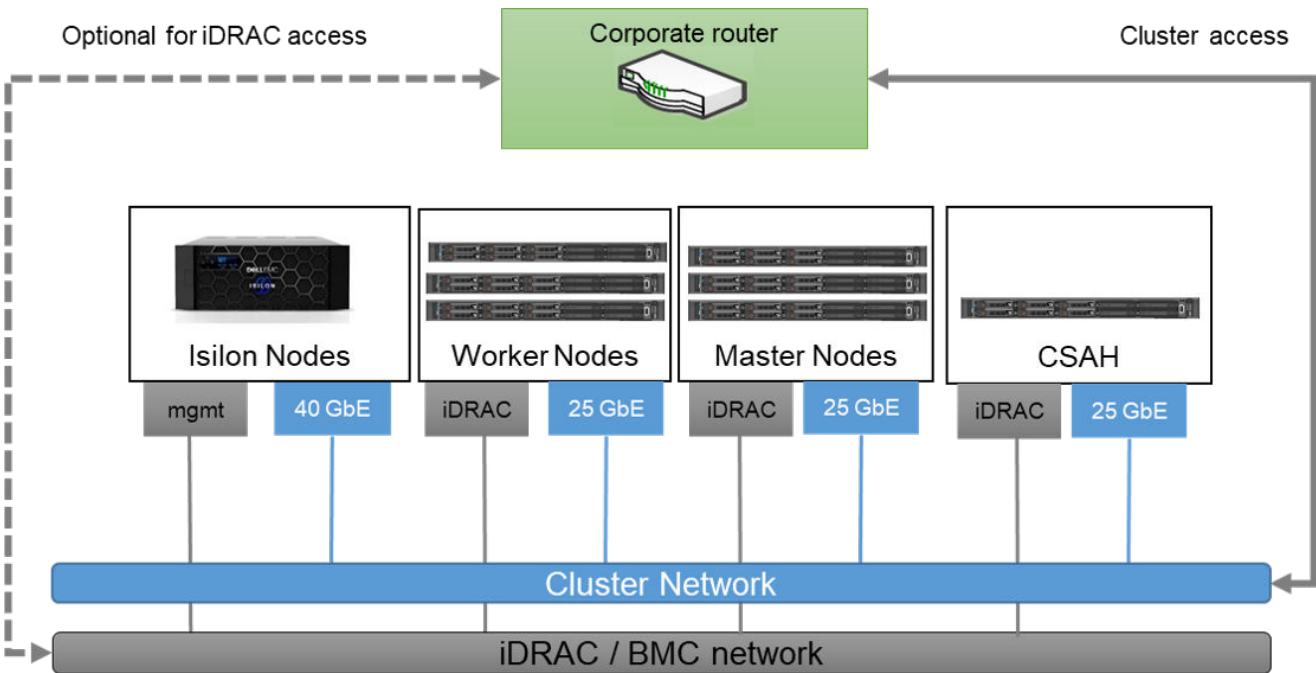


Figure 14. Core Cluster network organization

Dell EMC recommends the switches listed in the following table for the Core Cluster network.

Table 12. Network switches

Switch function	Recommended switch
Leaf (Top of Rack)	Dell EMC PowerSwitch S5248F-ON
Spine	Dell EMC PowerSwitch Z9100-ON
iDRAC network and Out of Band (OOB) management	Dell EMC PowerSwitch S3148-ON

NOTE: A typical installation uses two leaf switches per rack for high availability.

The Core Cluster network organization is shown in [Figure 14. Core Cluster network organization](#) on page 31.

Conclusions

The massive deployment of sensors, controllers, and other Internet of Things (IoT) devices make up the Industrial IoT (IIoT) can produce such massive volumes of data that superlatives like "big data" no longer describe the challenge. There is a vast opportunity for data scientists to discover associations and understanding from the patterns and trends within IIoT data. When the right IIoT data is made available for synthesis and analysis, the revealed associations, patterns, and trends can fuel the development of intelligent software applications and the advanced cyber-physical systems that define Industry 4.0.

Topics:

- [Document summary](#)

Document summary

Successful data analytics strategies for Industry 4.0 require dynamically selecting and routing required application data from a subset of devices. In turn this data must flow through processing at the edge, or upstream at a core or cloud facility. In addition to managing data flows, Industry 4.0 practitioners need the ability to "flow" data model artifacts to infrastructure that hosts useful, problem-solving applications such as:

- Improving equipment uptime
- Managing worker health and safety
- Improving yields
- And more

When selecting the most efficient strategies for IIoT data management, processing, and application deployment architectures, you must consider:

- Limited computational power at the edge
- Networking constraints between the edge and core
- Storage availability at every level in these complex distributed computing environments

This document has shown how the Confluent Platform can be applied to the challenges of IIoT and Industry 4.0. The Confluent event streaming platform is based on Apache Kafka that empowers organizations in the industrial sector to easily access IIoT device data as real-time streams. Event streams can then feed many of the traditional big data analytics (BDA) tools, leveraging staff with existing skills to develop operational and corporate IT business intelligence for IIoT environments. If BDA technologies, algorithms, and techniques can be leveraged in the development of intelligent cyber-physical IIoT systems, the promise of Industry 4.0 will be fulfilled quicker.

Development of the convergence of IIoT "edge analytics" and BDA using the Confluent Platform is already producing measurable results in the automotive industry and others. The extreme scale that is required for autonomous driving applications proves that the Kafka engine can handle the frequency, volume, and variety of IIoT environments. This document did not explore the scale limits of Kafka. Dell EMC developed a data generator with the capability to produce large numbers of high-frequency data streams. This design verified that transforming raw streams into messages for the Confluent event streaming platform was robust.

The Confluent Platform combines open-source developed resources with community developed features and commercial features to produce a complete solution for IIoT, including enterprise support. This document described integrating the event streaming engine, Confluent Replicator, Confluent Connector for HDFS, and KSQL for an end-to-end anomaly detection application use case. Dell EMC used the integration of the Confluent Replicator with the Kafka schema registry to coordinate data schema changes between the edge and the core. This document demonstrated how to combine traditional big data analytics tools including:

- Python
- TensorFlow
- Keras
- Open-source H2O
- MLFlow

These designs close the loop from model development with data that is collected at the edge, to model deployment back to edge, using MLOps best practices.

References

Additional information can be obtained at the [Dell EMC InfoHub for Data Analytics](#). If you need additional services or implementation help, contact your Dell EMC sales representative.

Topics:

- [Dell EMC documentation](#)
- [Confluent documentation](#)
- [Dell EMC Customer Solution Centers](#)
- [Dell Technologies InfoHub](#)
- [More information](#)

Dell EMC documentation

The following Dell EMC documentation provides additional and relevant information. Access to these documents depends on your login credentials. If you do not have access to a document, contact your Dell EMC representative.

- [Dell EMC PowerEdge R640 Spec Sheet](#)
- [Dell EMC PowerEdge R640 Manuals & Documents](#)
- [Dell EMC PowerEdge XE2420 Spec Sheet](#)
- [Dell EMC PowerEdge XE2420 Manuals & Documents](#)
- [Dell EMC PowerEdge R740xd Spec Sheet](#)
- [Dell EMC PowerEdge R740xd Manuals & Documents](#)
- [Dell EMC PowerSwitch S3148-ON Spec Sheet](#)
- [Dell EMC PowerSwitch S3148-ON Manuals & Documents](#)
- [Dell EMC PowerSwitch S5248F-ON Spec Sheet](#)
- [Dell EMC PowerSwitch S5248F-ON Manuals & Documentation](#)
- [Dell EMC PowerSwitch Z9100-ON Spec Sheet](#)
- [Dell EMC PowerSwitch Z9100-ON Manuals & Documents](#)
- [Dell EMC Ready Solutions for Data Analytics Real-Time Data Streaming Architecture Guide](#)
- [Dell Technologies Red Hat OpenShift Container Platform guides](#)

Confluent documentation

The following documentation on the [Confluent website](#) provides additional and relevant information:

- [Confluent Platform documentation](#)
- [Confluent blog](#)
- [Confluent developer resources](#)

Dell EMC Customer Solution Centers

Our global network of dedicated [Dell EMC Customer Solution Centers](#) are trusted environments where world class IT experts collaborate with customers and prospects to share best practices; facilitate in-depth discussions of effective business strategies using briefings, workshops, or Proofs of Concept (PoCs); and help business become more successful and competitive.

Dell EMC Customer Solution Centers reduce the risks that are associated with new technology investments, and can help improve speed of implementation.

All of the services of the Customer Solution Centers are available to all Dell Technologies customers at no charge. Contact your account team today to submit an engagement request.

Dell Technologies InfoHub

The [Dell Technologies InfoHub](#) is your one-stop destination for the latest information about Dell EMC Solutions and Networking products. New material is frequently added, so visit often to keep up to date on our expanding portfolio of cutting-edge products and solutions.

More information

For more information, contact your Dell EMC or authorized partner sales representative.