# Ready Solutions for Data Analytics

## Real-Time Data Streaming Architecture Guide

February 2019

H17557.1

DELL EMC

# Contents

# List of figures

# List of tables

# Trademarks

# Notes, cautions, and warnings

**Note:** A **Note** indicates important information that helps you make better use of your system.

**CAUTION:** A **Caution** indicates potential damage to hardware or loss of data if instructions are not followed.

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

This document is for informational purposes only and may contain typographical errors and technical inaccuracies. The content is provided as is, without express or implied warranties of any kind.

# Chapter 1

# Executive summary

**Topics:**



This Ready Architecture Guide describes recommended Dell EMC infrastructure for implementing Lambda and Kappa stream processing architectures using Apache Kafka and Confluent Enterprise.

# Introduction

Companies are modernizing their IT architecture to enable innovation, increase agility, and support the creation of new digital products and services. These trends are also driving a fundamental shift in how data powers a company's core business, as companies transition from batch processing to real-time data and event-driven applications.

The journey to a real-time enterprise starts with a streaming platform for data. A streaming platform combines the data distribution strength of a publish-subscribe model with a storage layer and a processing layer. This makes it much easier to create data pipelines and connect them to all of your systems. Deploying a streaming platform can avoid the latency of batch processing, and make data available in real time to applications that need it.

Apache Kafka is a streaming platform designed to solve these problems in a modern, distributed architecture. Originally envisioned as a fast and scalable distributed messaging queue, it has rapidly expanded into a full-scale streaming platform.

Confluent Platform improves Apache Kafka by expanding its integration capabilities, adding tools to optimize and manage Kafka clusters, and methods to ensure the streams are secure. Confluent Platform makes Kafka easier to build and easier to operate. Confluent Open Source is freely downloadable, while Confluent Enterprise is available through subscription.

In most customer environments, there is a logical progression in the adoption of streaming data analytics, as shown in *Figure 1: Streaming adoption journey* on page 9 [1]. Choosing the correct infrastructure for a streaming platform is critical for success, performance, and scalability. This guide describes Dell EMC's recommended infrastructure for deploying Kafka and Confluent Enterprise.

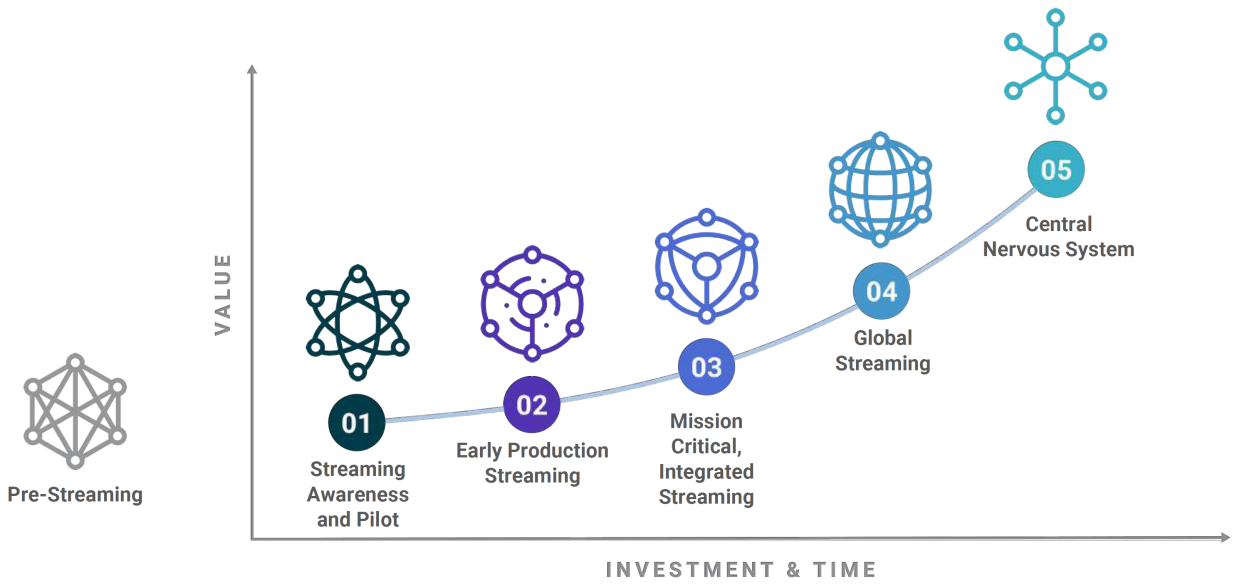


**Figure 1: Streaming adoption journey**

[1] Figure courtesy of Confluent, Inc.

# Chapter 2

# Streaming overview

**Topics:**



This chapter provides an overview of stream processing, key Kafka concepts, and architectures for stream processing in the context of Big Data Analytics.

# Streaming introduction

Apache Kafka is an open source, distributed streaming platform capable of handling trillions of events a day. Confluent Platform improves Kafka with additional open source and commercial features designed to enhance the streaming experience of both operators and developers in production, at massive scale.

This streaming platform provides three core capabilities:

- Publish and subscribe — see *Publish and subscribe* on page 11
- Storage — see *Storage* on page 11
- Processing — see *Processing* on page 11

## Publish and subscribe

Publish and subscribe describes the way data enters and leaves a streaming platform. The unit of data in Kafka is a *message*. Kafka messages are organized into *topics* that contained ordered messages. Data Producers *publish* their messages to a topic, and Consumers *subscribe* to a topic. A Kafka server is called a *Broker*. The Broker receives messages from Producers, and responds to Consumer requests. Multiple Brokers are normally operated as part of a cluster for scalability and redundancy. A topic can be partitioned into subsets across multiple Brokers, and *partitions* can also be replicated between Brokers.

The Confluent Platform provides multiple *APIs* for reading and writing data. The Producer/Consumer API provides a low level interface directly to and from the Brokers. The Kafka Connect API and Connectors support data integration with other systems, while the Kafka Streams API is optimized for stream processing. The *RESTful API Proxy* API can also be used as an *HTTP* interface to the cluster.

Brokers, Kafka Connect, and the RESTful API Proxy can all be scaled independently to meet capacity and redundancy requirements.

## Storage

Storage of messages is provided by the Brokers, and uses local disks. A topic can be partitioned into subsets across multiple Brokers, and partitions can also be replicated between Brokers for scalability and redundancy. Each topic can be configured with a retention policy, to store messages for some period of time or until a topic reaches a certain size in bytes. Once these limits are reached, messages are expired and deleted.

Kafka Connect can be used to transfer topic data to external systems such as:

- Relational and *NoSQL* databases
- Hadoop
- Object Stores
- Dell EMC Isilon

## Processing

The Kafka Streams *API* is a powerful, lightweight library that enables real-time data processing against Apache Kafka. Confluent *KSQL* is an open source, streaming *SQL* engine built upon Kafka Streams. 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

# Streaming architecture

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

- Fanout to multiple downstream systems optimized for specific use cases
- *SIEM* offload, replacement and augmentation
- Transactional and exactly once operations
- Real-time alerts at edge interfaces

In the Big Data Analytics space, two implementation patterns have emerged, commonly known as:

- Lambda architecture — see *Lambda architecture* on page 12
- Kappa architecture — see *Kappa architecture* on page 12

Both of these architectures process data into a serving system, which could be *Hadoop HDFS* files, *HBase*, or another *NoSQL* database. The service system is then interfaced to applications.

## Lambda architecture

The Lambda architecture is a stream processing architecture sometimes used to integrate real-time data into existing batch processing systems. *Figure 2: Lambda architecture* on page 12 illustrates this architecture.

In the Lambda architecture, there are two distinct processing flows, both writing to a service database:

- One for batch data
- One for streaming data

These flows will typically use different processing systems optimized for batch and stream processing. The speed table contains *recent* data, while the batch table contains *older* data. At regular intervals, usually based on the size of the speed table, the recent data is re-processed through the batch system into batch tables, the speed table is emptied, and the cycle is repeated.

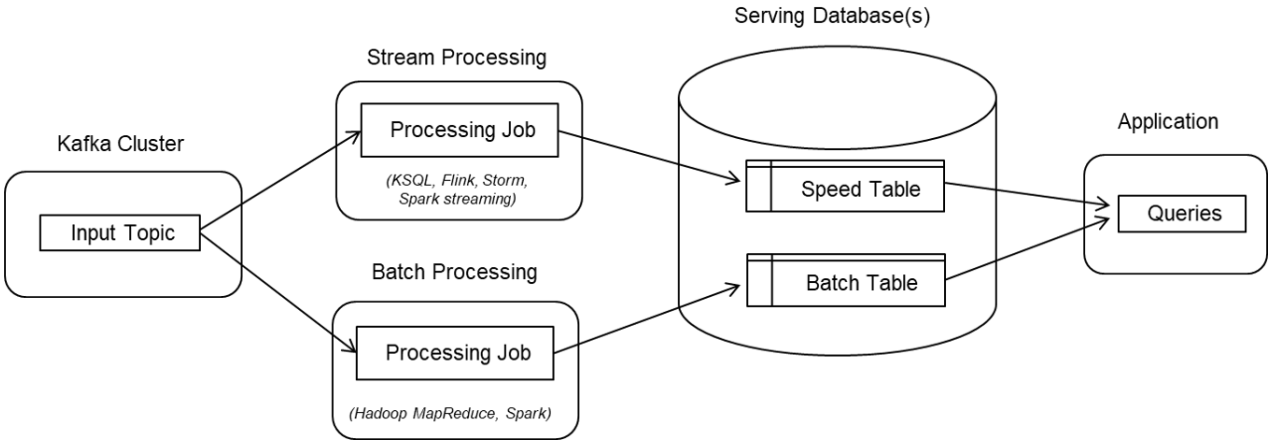


**Figure 2: Lambda architecture**

## Kappa architecture

The Kappa architecture is a stream processing architecture that eliminates all batch processing, and replaces it with streaming. Kappa architecture is sometimes used to replace batch processing; in other instances it is deployed from scratch. *Figure 3: Kappa architecture* on page 13 illustrates this architecture.

In the Kappa architecture, the same processing flow and system is used for all data, and the data is processed directly into the serving database. This architecture requires a high performance stream processing platform capable of keeping up with the flow of real-time data.

The Kappa architecture has some advantages over Lambda for stream processing. The primary advantage is the use of a single processing flow, which avoids re-processing and maintenance of the same processing logic in two different systems. Another advantage is the ability to quickly implement versioning of processing logic. Whenever logic needs to be changed, a new streaming job can be deployed, writing into a new table, in parallel with any existing logic.

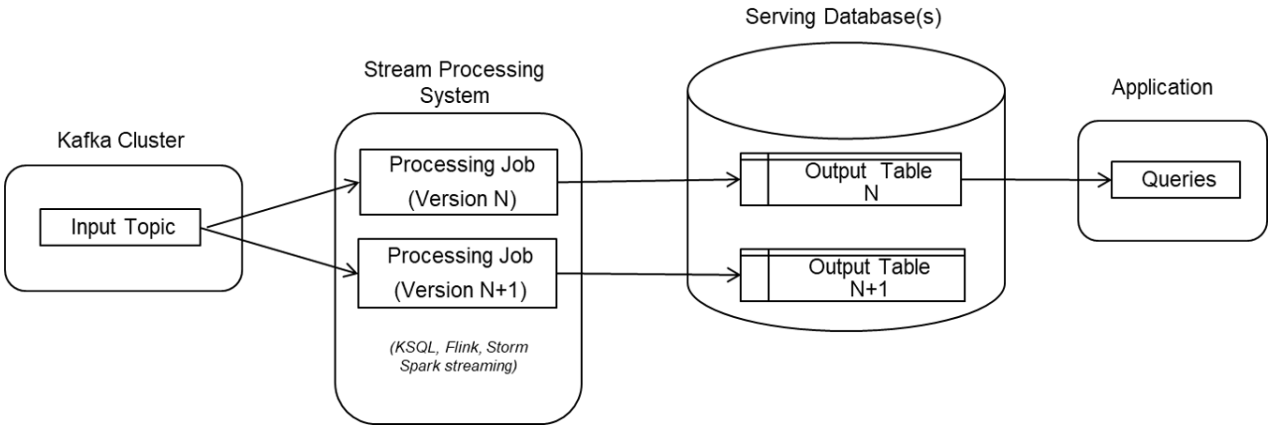


**Figure 3: Kappa architecture**

# Chapter 3

# Cluster architecture

**Topics:**



Kafka is both flexible and scalable, with multiple deployment options that complicate infrastructure selection. This architecture is provided to assist customers in selecting the correct infrastructure for implementing Confluent Enterprise in production environments.

## Architecture overview

The architecture follows the general recommendations in the *Apache Kafka and Confluent Enterprise Reference Architecture* and can be used to implement small or large clusters, with a clear path to scaling from small to large. The architecture also addresses the performance, latency, and reliability requirements of mission-critical production deployments.

The Confluent Enterprise components are grouped into seven main categories:

1. Confluent Control Center
2. Kafka Brokers
3. *ZooKeeper*
4. Kafka Connect and Connectors
5. Confluent *Schema* Registry
6. Confluent *KSQL*
7. Confluent *RESTful API Proxy*

This grouping is chosen to allow each category to be scaled independently, based on capacity and performance requirements. It also simplifies scaling from initial deployments to larger implementations, since some functions can be co-located on physical nodes, and later moved to additional physical nodes.

*Figure 4: Standard cluster architecture* on page 15 illustrates a typical initial deployment architecture, while *Figure 5: Large cluster architecture* on page 16 illustrates a scaled implementation where all components have been scaled out. Actual implementations will typically fall somewhere between these two examples.

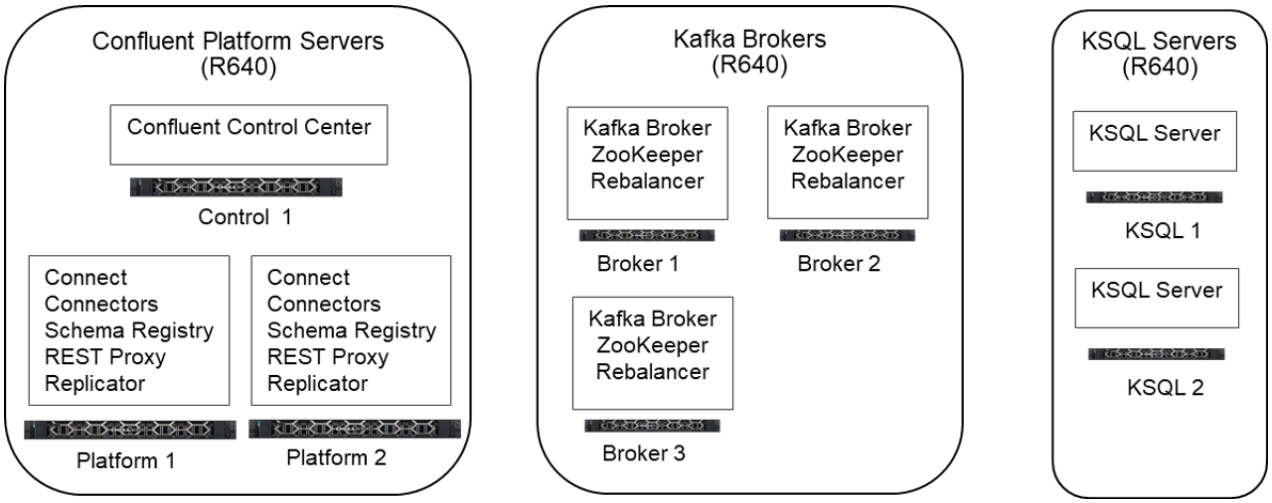


**Figure 4: Standard cluster architecture**

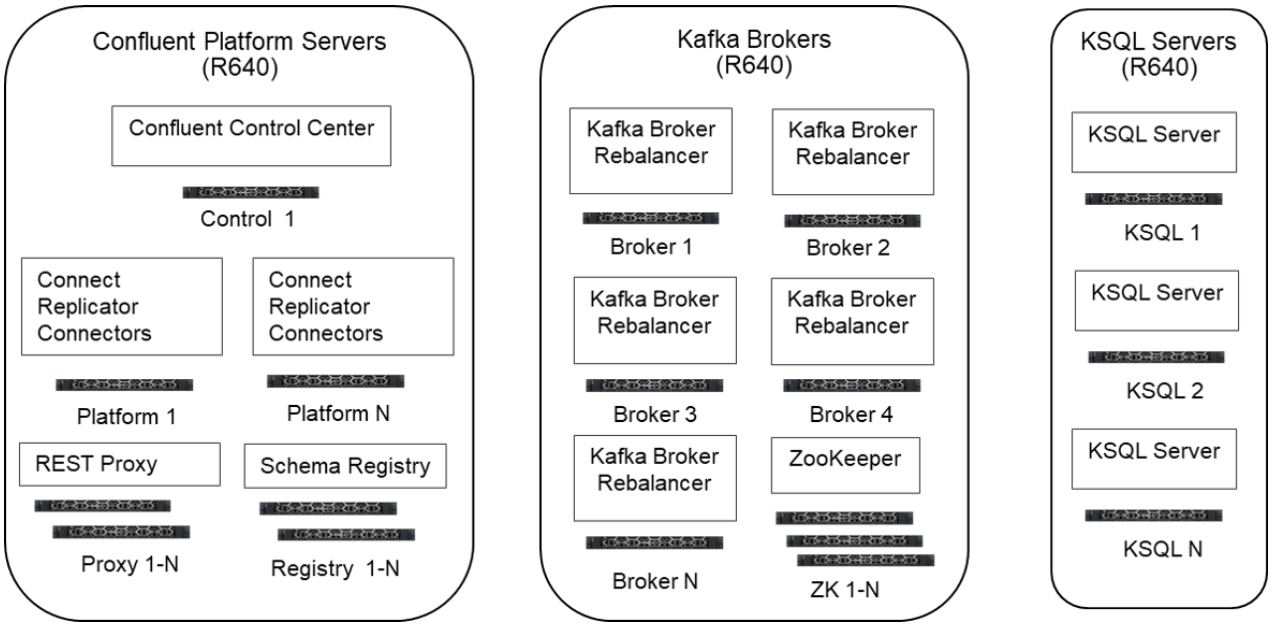


**Figure 5: Large cluster architecture**

To implement the infrastructure, the categories are mapped to functional roles for the nodes in the cluster. The roles define which services are provided by the nodes. The roles are then mapped to a corresponding hardware configuration. *Table 1: Node roles and services* on page 16 shows the node roles for the small and large cluster examples, and the services running on each one. *Table 2: Node roles and hardware configurations* on page 17 shows the mapping between the node roles and Dell EMC's recommended hardware configurations.

**Table 1: Node roles and services**

| Cluster node role | Primary services | |
|---|---|---|
| | **Small cluster** | **Large cluster** |
| Confluent Control Center | Confluent Control Center | Confluent Control Center |
| Kafka Broker | Kafka Broker<br>*Rebalancer*<br>ZooKeeper | Kafka Broker<br>Rebalancer |
| ZooKeeper | Consolidated with Kafka Broker role | ZooKeeper |
| Confluent Platform | Kafka Connect<br>Connectors<br>*Replicator*<br>Schema Registry<br>RESTful API Proxy | |
| Kafka Connect | Consolidated with Confluent Platform role | Kafka Connect<br>Connectors<br>Replicator |

| Cluster node role | Primary services | |
|---|---|---|
| | Small cluster | Large cluster |
| Confluent Schema Registry | Consolidated with Confluent Platform role | Schema Registry |
| Confluent RESTful API Proxy | Consolidated with Confluent Platform role | RESTful API Proxy |
| Confluent KSQL | KSQL Server | KSQL Server |

**Table 2: Node roles and hardware configurations**

| Cluster node role | Hardware configuration |
|---|---|
| Confluent Control Center | *Control Center Node* |
| Kafka Broker | *Broker Node* on page 22<br>or<br>*Broker Node - high performance* on page 24 |
| ZooKeeper | *Platform Node* |
| Kafka Connect | *Platform Node* |
| Confluent Schema Registry | *Platform Node* |
| Confluent RESTful API Proxy | *Platform Node* |
| Confluent KSQL | *KSQL Node* |

## Network architecture

The network is an important part of any Confluent Enterprise implementation. The network organization is shown in *Figure 6: Network layout* on page 18. The network is flat, and accounts for traffic both in and out of the cluster, and traffic within the cluster. This architecture supports scaling of the individual components and of the cluster as a whole.

Dell EMC recommends 25 GbE technology for best price and performance. The switches are high-performance, non-blocking models, with multi-rate capabilities to support external interfaces. In some deployments using the RESTful API proxy, a hardware load balancer may be required and can be easily added to the implementation.

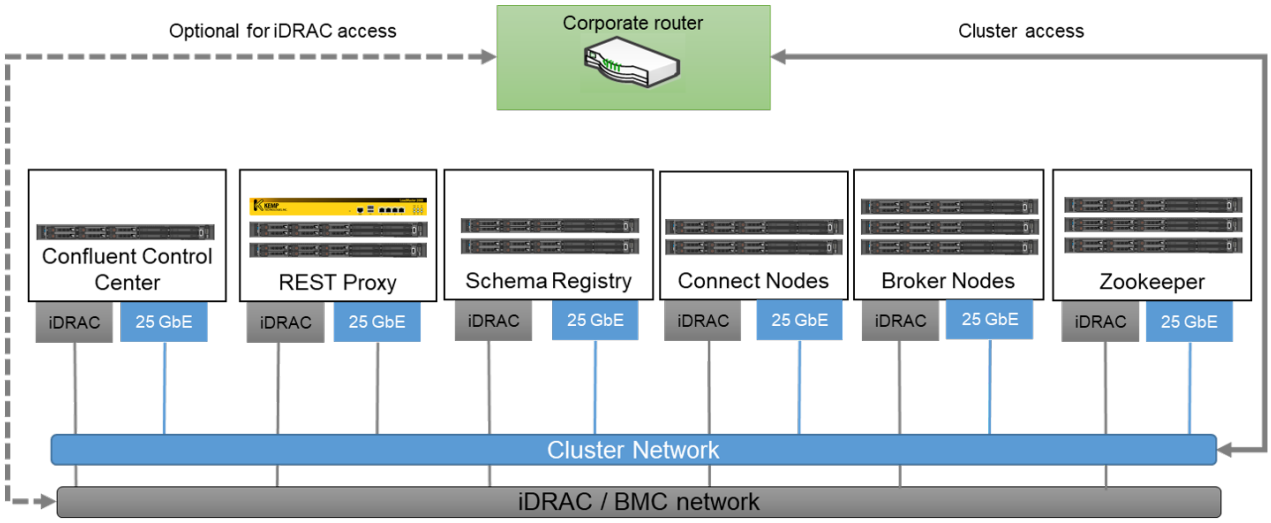


**Figure 6: Network layout**

# Chapter 4

# Hardware configurations

**Topics:**



This chapter describes the Dell EMC recommended server and network configurations for Confluent Enterprise.

## Control Center Node

The Control Center Node is used to host Confluent Control Center. Dell EMC's recommended hardware configuration is listed in *Table 3: Hardware configuration – Control Center Node* on page 20.

**Table 3: Hardware configuration – Control Center Node**

| Machine function | Control Center Node |
|---|---|
| Platform | Dell EMC PowerEdge R640 |
| Chassis | 2.5" Chassis with up to 8 hard drives and 3 *PCIe* slots |
| Processor | Single Intel Xeon Gold 5118 2.3 GHz, 12 core, 16.5 MB cache |
| RAM | 48 GB (6 x 8 GB 2667 MT/s) |
| Network daughter card | Mellanox ConnectX-4 Lx Dual Port 25 GbE *SFP28 rNDC* |
| Storage controller | Dell EMC PERC H330 |
| Storage configuration | (2) 480 GB *SATA SSD* |
| Boot configuration | From *PERC* controller |
| Power supply | Dual, hot-plug, redundant, 750W |

**Table 4: Volumes - Control Center Node**

| Physical disks | Usage | Volume type |
|---|---|---|
| 0-1 | Operating system | *RAID* 1 |

**Table 5: Partitions - Control Center Node**

| Virtual disk | Partition | Mount point | Size | Filesystem type | Description |
|---|---|---|---|---|---|
| Virtual 0 | Primary | /boot | 1074 MB | xfs | Contains *BIOS* boot files that must be within first 2 GB of disk |
| Virtual 0 | LVM | / | 100 GB | xfs | Root filesystem |
| Virtual 0 | LVM | swap | 4 GB | swap | Operating system swap space partition |
| Virtual 0 | LVM | /home | 1 GB | xfs | User home directories |
| Virtual 0 | LVM | /var | 350.6 GB | xfs | Contains variable data, like system logging files, spool directories, and *RocksDB* storage |

## Platform Node

The Platform Node configuration is used for multiple roles, including Kafka Connect, the Confluent *Schema* Registry, and Confluent *RESTful API Proxy*. Dell EMC's recommended configuration is listed in *Table 6: Hardware configurations – Platform Node* on page 21.

**Table 6: Hardware configurations – Platform Node**

| Machine function | Platform Node |
|---|---|
| Platform | Dell EMC PowerEdge R640 |
| Chassis | 2.5" Chassis with up to 8 hard drives and 3 *PCIe* slots |
| Processor | Single Intel Xeon Gold 6126 2.6 GHz, 12 core, 19.25 MB cache |
| RAM | 48 GB - 6 x 8 GB *DIMM* |
| Network daughter card | Mellanox ConnectX-4 Lx Dual Port 25 GbE *SFP28 rNDC* |
| Storage controller | Dell EMC PERC H330 |
| Storage configuration | (2) 240 GB *SATA SSD* |
| Boot configuration | From *PERC* controller |
| Power supply | Dual, hot-plug, redundant, 750W |

**Table 7: Volumes - Platform Node**

| Physical disks | Usage | Volume type |
|---|---|---|
| 0-1 | Operating system | *RAID* 1 |

**Table 8: Partitions - Platform Node**

| Virtual disk | Partition | Mount point | Size | Filesystem type | Description |
|---|---|---|---|---|---|
| Virtual 0 | Primary | /boot | 1074 MB | xfs | Contains *BIOS* boot files that must be within first 2 GB of disk |
| Virtual 0 | LVM | / | 50 GB | xfs | Root filesystem |
| Virtual 3 | LVM | swap | 4 GB | swap | Operating system swap space partition |
| Virtual 4 | LVM | /home | 1 GB | xfs | User home directories |
| Virtual 5 | LVM | /var | 167.5 GB | xfs | Contains variable data, like system logging files, spool directories, and *RocksDB* storage |

## KSQL Node

The KSQL Node is used to host Confluent *KSQL*. Dell EMC's recommended configuration is listed in *Table 9: Hardware configurations – KSQL Node* on page 21.

**Table 9: Hardware configurations – KSQL Node**

| Machine function | KSQL Node |
|---|---|
| Platform | Dell EMC PowerEdge R640 |
| Chassis | 2.5" Chassis with up to 8 hard drives and 3 *PCIe* slots |
| Processor | Single Intel Xeon Gold 6126 2.6 GHz, 12 core,19.25 MB cache |

| Machine function | KSQL Node |
|---|---|
| RAM | 48 GB - 6 x 8 GB *DIMM* |
| Network daughter card | Mellanox ConnectX-4 Lx Dual Port 25 GbE *SFP28 rNDC* |
| Boot configuration | From *PERC* controller |
| Storage controller | Dell EMC PERC H330 |
| Storage configuration | (2) 480 GB *SAS SSD* |
| Boot configuration | From PERC controller |
| Power supply | Dual, hot-plug, redundant, 750W |

**Table 10: Volumes - KSQL Node**

| Physical disks | Usage | Volume type |
|---|---|---|
| 0-1 | Operating system | *RAID* 1 |

**Table 11: Partitions - KSQL Node**

| Virtual disk | Partition | Mount point | Size | Filesystem type | Description |
|---|---|---|---|---|---|
| Virtual 1 | Primary | /boot | 1074 MB | xfs | Contains *BIOS* boot files that must be within first 2 GB of disk |
| Virtual 2 | LVM | / | 50 GB | xfs | Root filesystem |
| Virtual 3 | LVM | swap | 4 GB | swap | Operating system swap space partition |
| Virtual 4 | LVM | /home | 1 GB | xfs | User home directories |
| Virtual 5 | LVM | /var | 390.6 GB | xfs | Contains variable data, like system logging files, spool directories, and *RocksDB* storage |

## Broker Node

The Broker Node is used to host the Kafka Broker, and provides a balance of compute and storage. Dell EMC's recommended configuration is listed in *Table 12: Hardware configurations – Broker Node* on page 22

**Table 12: Hardware configurations – Broker Node**

| Machine function | Broker Node |
|---|---|
| Platform | Dell EMC PowerEdge R640 |
| Chassis | 2.5" Chassis with up to 10 hard drives, 2 x 2.5" *SATA*/*SAS* drives and 1 *PCIe* slot |
| Processor | Dual Intel  Xeon Gold 5118 2.3 GHz, 12 core 16.5 MB cache |
| RAM | 96 GB - 12 x 8 GB |
| Network daughter card | Mellanox ConnectX-4 Lx Dual Port 25 GbE *SFP28 rNDC* |

| Machine function | Broker Node |
|---|---|
| Storage controller | Dell EMC PERC H740P |
| Storage configuration | *BOSS* card + with two 240 GB M.2<br>(10) 2TB 7.2K *NL-SAS*<br>(1) 480 GB *SSD* (rear bay) |
| Boot configuration | From BOSS controller |
| Power supply | Dual, hot-plug, redundant, 750W |

**Table 13: Volumes - Broker Node**

| Physical disks | Usage | Volume type |
|---|---|---|
| 0-1 | Operating System | *RAID* 1 |
| 2-9 | Kafka partitions | *RAID* 6 (*Refer to note.*) |
| 10-11 | *ZooKeeper* data | RAID 1 |
| 12 | ZooKeeper log | RAID 1 |

**Note:** The Kafka partitions should use a reliable storage layer, either RAID 5, RAID 6, or RAID 10. We recommend 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.

**Table 14: Partitions - Broker Node**

| Virtual disk | Partition | Mount point | Size | Filesystem type | Description |
|---|---|---|---|---|---|
| DellBOSS 1 | Primary | /boot | 1074 MB | xfs | Contains *BIOS* boot files that must be within first 2 GB of disk |
| DellBOSS 2 | LVM | / | 50 GB | xfs | Root filesystem |
| DellBOSS 3 | LVM | swap | 4 GB | swap | Operating system swap space partition |
| DellBOSS 4 | LVM | /home | 1 GB | xfs | User home directories |
| DellBOSS 5 | LVM | /var | 167.5 GB | xfs | Contains variable data, like system logging files, databases, mail and printer spool directories, transient and temporary files |
| Virtual 0 | LVM | /kafka | 12 TB | xfs | Kafka Broker data |
| Virtual 1 | LVM | /var/zk | 2 TB | xfs | ZooKeeper data |
| Virtual 2 | LVM | /var/log/ zookeeper | 480 GB | xfs | ZooKeeper log data |

## Broker Node - high performance

The High performance Broker Node is used to host Kafka Brokers. This configuration uses *NVMe* drives for high performance storage. Dell EMC's recommended configuration is listed in *Table 15: Hardware configurations – Broker Node high performance* on page 24.

**Table 15: Hardware configurations – Broker Node high performance**

| Machine function | Broker Node |
|---|---|
| Platform | Dell EMC PowerEdge R640 |
| Chassis | 2.5" Chassis with up to 10 hard drives, 8 NVMe drives and 3 *PCIe* slots |
| Processor | Dual Intel Xeon Gold 6126 2.6 GHz, 12 core, 19.25 MB cache |
| RAM | 96 GB - 12 x 8 GB |
| Network daughter card | Mellanox ConnectX-4 Lx Dual Port 25 GbE *SFP28 rNDC* |
| Storage controller | Dell EMC PERC H330 |
| Storage configuration | (2) 240GB *SATA SSD*<br>(6) 1.6 TB U.2 NVMe flash |
| Boot configuration | From *PERC* controller |
| Power supply | Dual, hot-plug, redundant, 750W |

**Table 16: Volumes - Broker Node high performance**

| Physical disks | Usage | Volume type |
|---|---|---|
| 0-1 | Operating system | *RAID* 1 |
| 2-7 | Kafka partitions | No RAID |

**Table 17: Partitions - Broker Node - high performance**

| Virtual disk | Partition | Mount point | Size | Filesystem type | Description |
|---|---|---|---|---|---|
| Virtual 0 | Primary | /boot | 1074 MB | xfs | Contains *BIOS* boot files that must be within first 2 GB of disk |
| Virtual 0 | LVM | / | 50 GB | xfs | Root filesystem |
| Virtual 0 | LVM | swap | 4 GB | swap | Operating system swap space partition |
| Virtual 0 | LVM | /home | 1 GB | xfs | User home directories |
| Virtual 0 | LVM | /var | 167.5 GB | xfs | Contains variable data, like system logging files, databases, mail and printer spool directories, transient and temporary files |

| Virtual disk | Partition | Mount point | Size | Filesystem type | Description |
|---|---|---|---|---|---|
| nvme0-nvme5 | nvme0p1-nvme5p1 | /kafka0-/kafka5 | 1.6 TB | xfs | Kafka Broker data |

**Note:** For high performance Broker Nodes, no provision is made for *ZooKeeper* data. These nodes are intended to be used with dedicated ZooKeeper nodes.

**Note:** High performance Broker Nodes must be used in *JBOD* mode since NVMe configurations do not support RAID.

## Network configurations

Dell EMC recommends the Dell EMC Networking S5148-ON as the rack level switch for deployments. It is a high-density 25/10/1 GbE 1 *RU* form factor switch with up to 48 ports of native 25 GbE (*SFP28*) ports, supporting 25 GbE without breakout cables. It also provides six multi-rate 100 GbE ports, supporting 10/25/40/50 GbE to simplify integration to existing network infrastructures. This switch uses a high-performance 3.6 Tbps (full-duplex) non-blocking, cut-through switching fabric and delivers line-rate performance under full load.

Networking requirements will vary somewhat based on the actual deployment scenario. For single rack cluster deployments, a single switch is adequate and will comfortably support 40 nodes, with spare connections for integration. *Figure 7: Single rack network connections* on page 26 illustrates a typical single rack scenario.

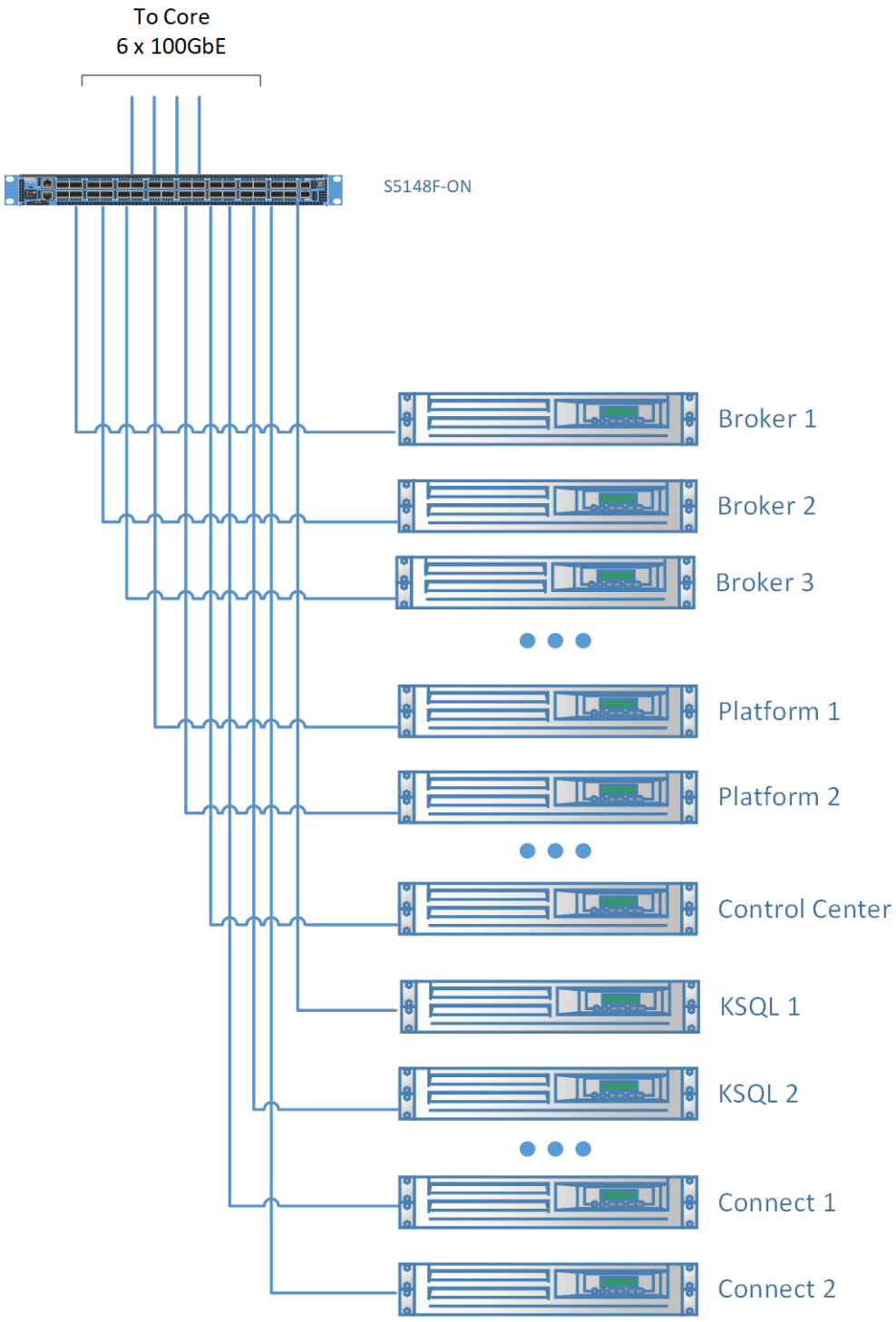


**Figure 7: Single rack network connections**

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*, RESTful API *Proxy*, and a load balancer using *session affinity* ("sticky" sessions) may be required. In other cases, IP rotation or simple load balancing will be adequate.

Dell EMC recommends the Dell EMC Networking S3048-ON for *iDRAC* or management network interfaces.

**Table 18: Network configurations**

| Function | Switch model |
|---|---|
| Top of rack or cluster | Dell EMC Networking S5148-ON |
| Management | Dell EMC Networking S3048-ON |
| Load balancing | Hardware load balancer, or IP rotation. |

# Chapter 5

# Integration, sizing, and scaling

**Topics:**



This chapter describes the aspects of integrating Confluent Enterprise into existing environments.

# Integration

*Figure 8: Client data interfaces* on page 29 illustrates the Confluent Enterprise *APIs* that can be used for external interfaces with the cluster. The architecture supports all of them simultaneously. However, the actual deployment scenario will define which interfaces are used.

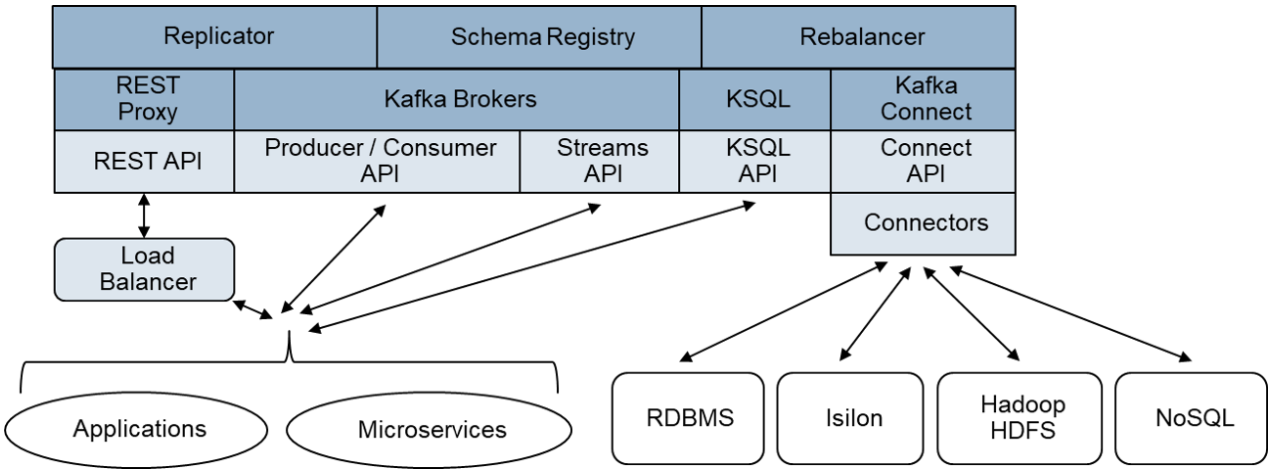


**Figure 8: Client data interfaces**

## Integration with external data sources

As a data streaming platform, a Confluent Enterprise deployment requires network connectivity to multiple data sources and targets. If Kafka Connect is being used, the external message traffic will be via the nodes running Kafka Connect, while traffic using either the Producer/Consumer API or the streams API will be via theBroker Nodes.

All the cluster nodes are accessible via the Cluster Data network. The Dell EMC Networking S5148-ON has 25 GbE and 100 GbE ports available for low latency connectivity to the external systems. One or more ports can be used depending on the bandwidth requirements.

*Figure 9: Hadoop integration example* on page 30 illustrates one possible scenario where streaming data is being consumed by Spark running under *YARN*, with the Spark streaming integration using the Consumer API, while stream data is also being written to *HDFS* using the Kafka Connect HDFS Connector.

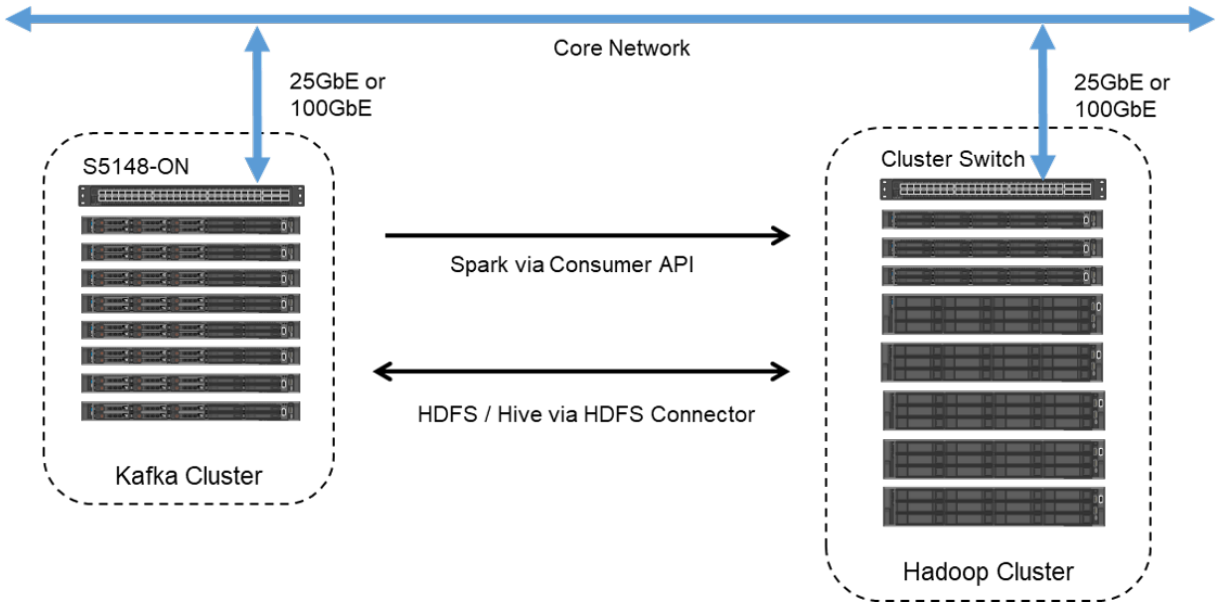


**Figure 9: Hadoop integration example**

## Integration with Isilon

Dell EMC Isilon provides a large, high performance, and reliable storage pool, and can be accessed from Kafka using HDFS or NFS protocols. In some implementations, a large amount of stream data needs to be stored for extended periods. For these cases, Isilon can be used as an archive tier, to offload data from Brokers and free up storage. Dell EMC recommends the HDFS interface for this scenario, since it supports direct access to the archived data from many analytics tools, while still providing easy access through Kafka.

*Figure 10: Isilon integration example* on page 30 illustrates this scenario.

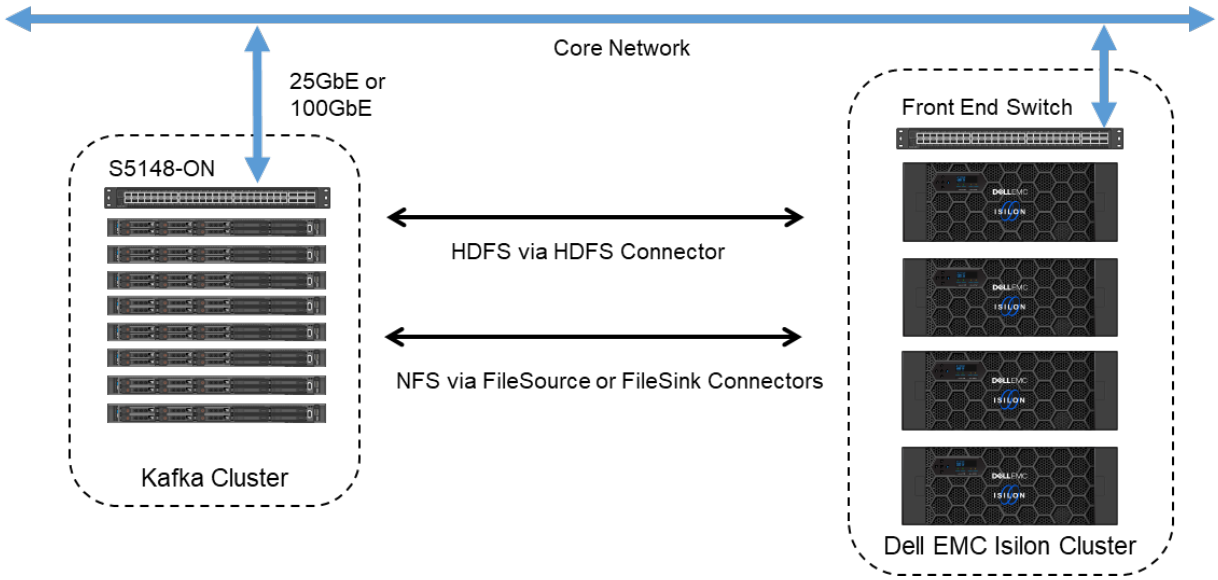


**Figure 10: Isilon integration example**

## Physical rack integration

*Figure 11: Single rack deployment* on page 31 shows the physical rack layout of nodes for a standard installation. The node service assignments are described in *Table 1: Node roles and services* on page 16. This installation can be scaled within the single rack to add capacity. Additional Broker Nodes, KSQL Nodes, or Platform Nodes can be independently added as needed. The primary limitation of a single rack configuration is high availability, since the entire rack is a single fault zone if power or a network switch fails. Redundant power rails and redundant network switches can be used to mitigate this.

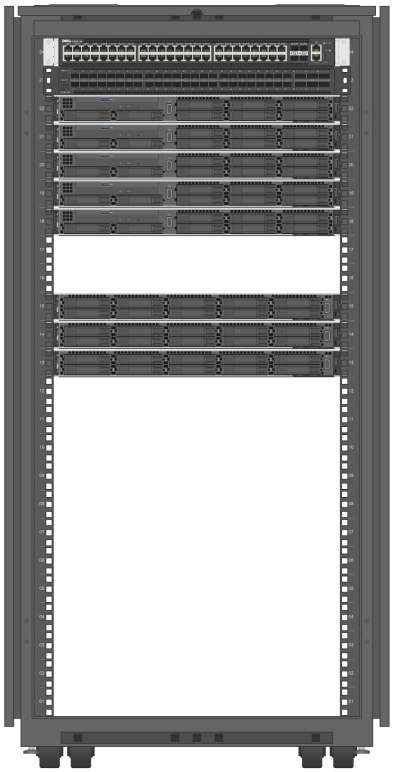


**Figure 11: Single rack deployment**

Confluent Enterprise is rarely deployed as a standalone system; it is typically integrated with existing systems such as Hadoop. In these deployments, the nodes are usually distributed across existing racks. *Figure 12: Multiple rack deployment* on page 32 shows an example of a deployment where the nodes are distributed across the racks of a Hadoop installation to provide additional high-availability. The node service assignments are described in *Table 1: Node roles and services* on page 16. In many instances, there is enough network, power and cooling capacity to add the new nodes without any infrastructure changes.

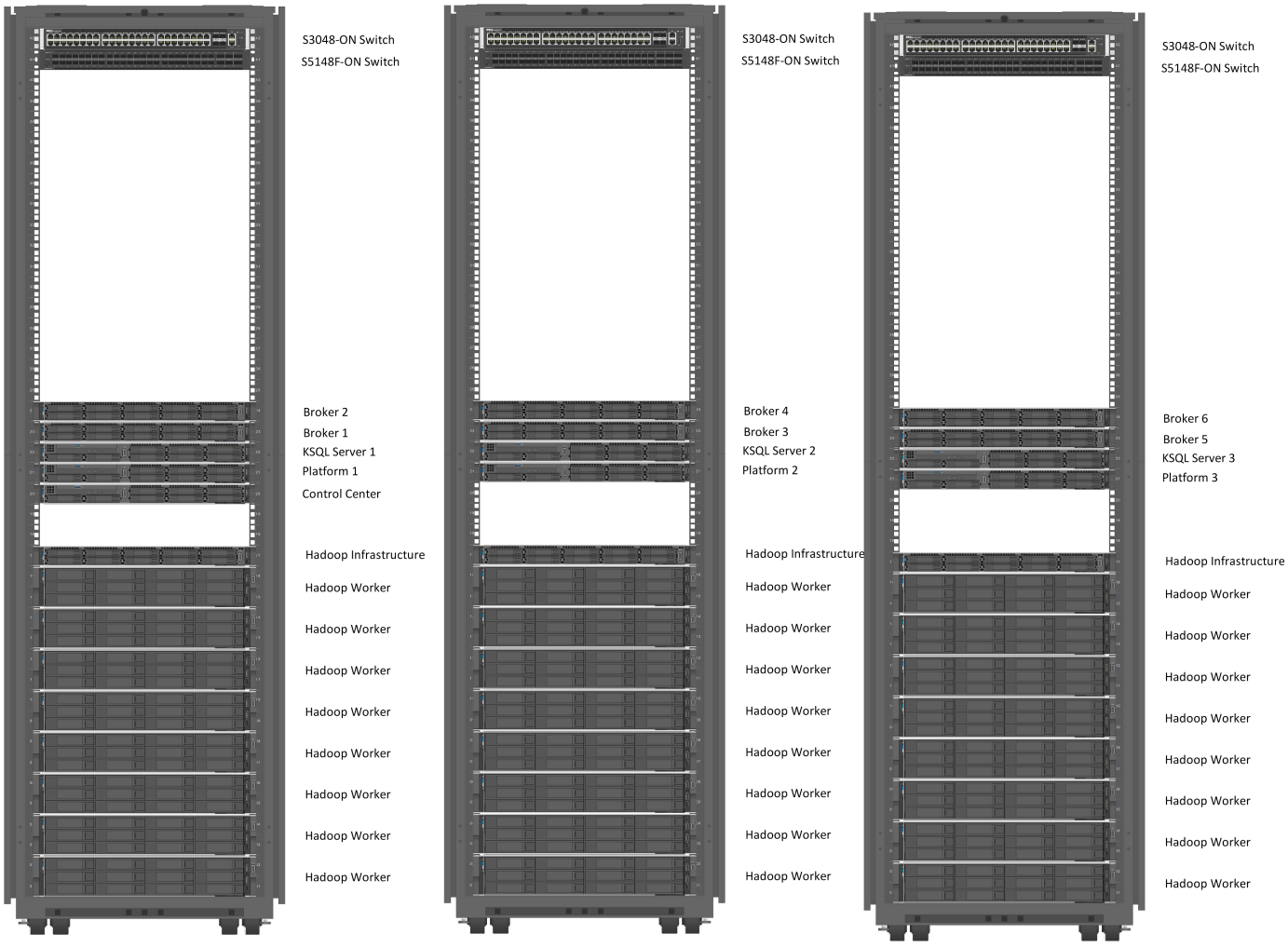


**Figure 12: Multiple rack deployment**

# Sizing and scaling

As described in *Architecture overview* on page 15, a Confluent Enterprise cluster can be scaled to meet performance and capacity requirements of the workloads.

## Broker and ZooKeeper scaling

Dell EMC recommends a minimum of three Broker Nodes. Each Broker provides approximately 8 TB of storage for stream data using a RAID 10 configuration, 14 TB for RAID 5 and 12 TB for RAID 6. Kafka topics and partitions can be distributed across these nodes. If additional storage is required for more topics or longer retention, Brokers can be added to the cluster as needed. Brokers can also be added to increase performance for higher data rates on the Producer or Consumer side. Each additional Broker adds bandwidth and processing capacity to the cluster in addition to storage. Broker Nodes should be distributed across racks for High Availability. Very high performance requirements can be met by using the High Performance Broker Node configuration.

Dell EMC's recommended configuration for Brokers includes processing capacity and storage space for *ZooKeeper*, and assumes the ZooKeeper ensemble is running on the Broker Nodes. The minimum number of ZooKeeper servers is three. For larger Kafka clusters, or multiple clusters, ZooKeeper can be hosted on a dedicated set of ZooKeeper nodes using the Platform Node configuration. This approach frees up capacity on the Broker Nodes, and allows ZooKeeper to be scaled independently of the Brokers. The

ZooKeeper ensemble should be scaled in odd increments to maintain a quorum in the ensemble. When using the High Performance Broker Node ZooKeeper must be hosted on a separate set of machines.

## KSQL and Kafka Connect scaling

KSQL can be scaled by adding additional KSQL nodes, and can scale independently of ZooKeeper and Broker Nodes.

Kafka Connect can also be scaled independently to meet capacity requirements. Dell EMC's recommended architecture co-locates the *RESTful API Proxy* and Schema Registry functions on the Kafka Connect nodes. Either or both of these functions can be split out to independent nodes, and they can also be scaled to meet capacity requirements.

# Chapter 6

# Testing and performance results

**Topics:**



This chapter presents results based on running Kafka Real-Time Data Streaming workloads on a cluster of Brokers based upon the Real-Time Data Streaming Ready Architecture, as described in *Hardware configurations* on page 19.

## Overview

The main objective of performance characterizations was to use Kafka distributed data streams to stress the resources of the underlying infrastructure, in order to:

- Understand the behavior of the Kafka cluster under load
- Identify bottlenecks

**Note:** These were baseline tests. Apart from optimizing Kafka properties for functionality, no extra tuning or tweaking was applied.

## System under test

The Broker Node System Under Test (SUT) consisted of Dell EMC PowerEdge R640 servers based upon this architecture, as described in *Table 15: Hardware configurations – Broker Node high performance* on page 24. The Control Center Node, Platform Node, and KSQL Node also used Dell EMC's recommended configurations as described in *Hardware configurations* on page 19.

The software components used are as described in *Tested component versions* on page 42. *Intel Platform Analysis Technology (PAT)* was used for performance monitoring and analysis.

## Performance tests

The performance tests that were executed on the SUT are listed in *Results* on page 35.

Before the tests, ZooKeeper and Kafka Server properties were appropriately configured for functionality. The Kafka cluster used in these tests consists of Producers that send messages (records) to the cluster of Brokers for storage. The cluster makes these records available to Consumers. The records sent by the Producers are appended to write-ahead logs known as Topics. Consumers subscribe to specified topics and any changes therein. Essentially, these tests consist of creating partitioned (distributed) topics, with each partition being serviced by a lead Broker.

For redundancy, other Brokers (followers) are assigned to the partition so that it can be replicated. A replication factor of 3 was used. Synchronous replication (`acks=-1` or `all`) was used where the leader of a partition tracks the progress of the follower replicas, and ensures that Kafka does not notify Consumers or Producers about specific records until they are fully acknowledged by replicas. Synchronous replication ensures that messages will not be lost as long as one in-sync replica remains.

Kafka performance classes are used to generate data streams of 50 million, 100-byte records which are written to the logs by Brokers, and then subsequently ingested by Consumers. The rate at which these records are written and/or consumed was a primary metric in these tests.

The goal of the first test was to find the number of partitions (or range) that results in optimal publication/consumption throughput for the 3-Broker configuration. Subsequent tests involved varying the number of Producers or Consumers or a combination of the two, and obtaining the resultant Throughput, Latency and/or Fetch Times. The Intel PAT tool was used to monitor the impact of these data stream variations on the CPU utilization and I/O (disk/network) traffic of the underlying infrastructure. Additional tests were undertaken to measure throughput gain resulting from Broker scalability.

## Results

This section describes the details and results of the following performance tests:

- *Producer and consumer variations* on page 38
- *Broker scalability* on page 40
- *Message size variations* on page 41


## Partitioning

As a baseline, the number of Producers on the standard 3-Broker configuration, for one topic, was raised from 1 to 4. At each Producer (1 through 4) iteration test, the number of partitions was raised from 1 to (up to) 900. For every test, the throughput and message latency were noted.

The best throughput was 591.38 MB/s with 11 partitions from the 4-Producer test. The four tests indicated that the best throughput was attained between 9 to 20 partitions. For subsequent tests, 15 partitions were used.

## Producer variations

Using 15 partitions, the number of Producer instances was raised from 1 to 25. The test was conducted with the parameters shown in *Table 19: Producer variations test parameters* on page 36.

**Table 19: Producer variations test parameters**

| Parameter | Value |
|---|---|
| Number of partitions | 15 |
| Replication factor | 3 |
| Minimum in sync replicas | 2 |
| Producer ACK | All (durable) |
| Message size | 100 Bytes |
| Number of records | 50 million per producer |
| Compression | LZ4 |
| Bufer size | 64 MB |
| Batch | 16 KB |
| Partitioner | Default - round robin (no message key) |

*Figure 13: Throughput versus number of producers* on page 37 graphs the throughput performance variations per the number of Producers.

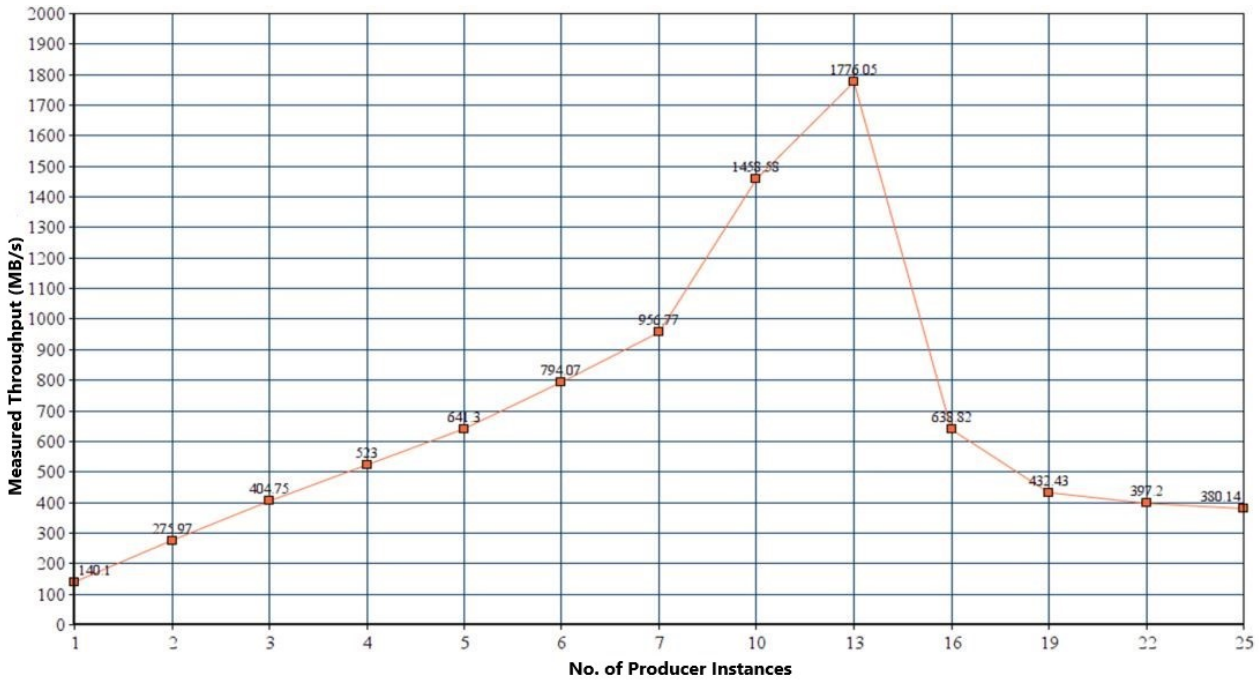


**Figure 13: Throughput versus number of producers**

The best throughput, almost 1.8 GB/s (about 18 million records per second), was observed with 13 Producers. Further adding Producer instances resulted in a sharp rise in message latency, which lead to an almost exponential drop in throughput.

As the number of Producer instances increases beyond 13, fewer records per second are processed. For instance with 25 Producers, 1.2 billion records are sent but they are processed at a rate of about 4 million per second.

## Consumer variations

Similar to the Producer variations tests, the number of Consumer instances was raised from 1 to 30. The test was conducted with the parameters shown in *Table 20: Consumer variations test parameters* on page 37.

**Table 20: Consumer variations test parameters**

| Parameter | Value |
|---|---|
| Number of partitions | 15 |
| Replication factor | 3 |
| Minumum in-sync replicas | 2 |
| Message size | 100 bytes |
| Consumer offset details | Name: __consumer_offsets<br>Partitions: 50<br>Replication factor: 3 |

*Figure 14: Throughput versus number of consumers* on page 38 graphs the throughput performance variations per the number of Consumers.

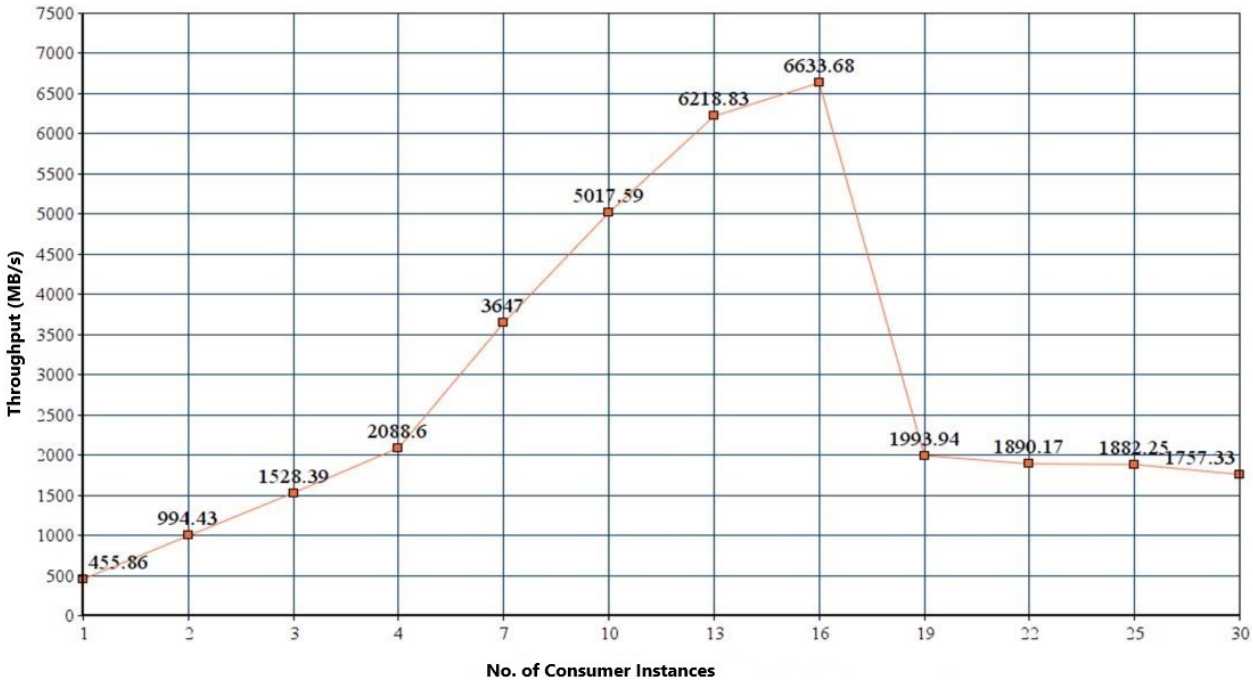


**Figure 14: Throughput versus number of consumers**

The best throughput, almost 6.7 GB/s (67 million+ records per second) was attained with 16 Consumer instances with 800 million messages consumed (50 million message per consumer group). With 30 Consumer instances, 1.5 billion messages are read (consumed) but they are processed at a rate of about 18 million per second, down from the peak 67 million per second.

## Producer and consumer variations

For these tests, Producer and Consumer instances were deployed simultaneously on the same Broker instances. The test was conducted with the parameters shown in *Table 21: Producer and consumer variations test parameters* on page 38.

**Table 21: Producer and consumer variations test parameters**

| Parameter | Value |
|---|---|
| Number of partitions | 15 |
| Replication factor | 3 |
| Minumum in-sync replicas | 2 |
| Producer ACK | All (durable) |
| Message size | 100 bytes |
| Number of records | 50 million per producer |
| Compression | LZ4 |
| Buffer size | 64 MB |
| Batch | 16 KB |
| Partitioner | Default - round robin (no message keys) |

| Parameter | Value |
|---|---|
| Consumer offsets topic details | Name: __consumer_offsets<br>Partitions: 50<br>Replication factor: 3 |

*Figure 15: Throughput versus number of producers and consumers* on page 39 graphs the throughput performance variations per the number of Producers and Consumers.

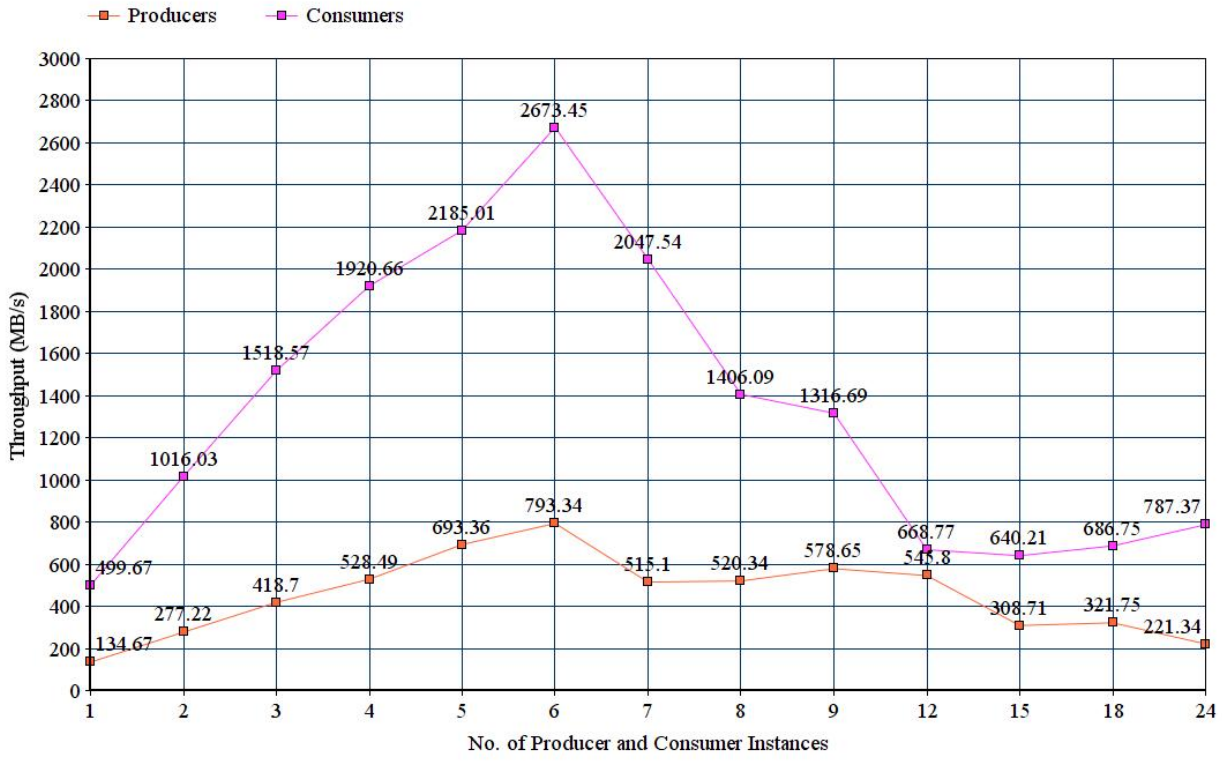


**Figure 15: Throughput versus number of producers and consumers**

The best throughput was attained when 6 Producers and 6 Consumers (300 million message each), were run simultaneously, as shown in *Table 22: Results with 6 producers and 6 consumers* on page 39.

**Table 22: Results with 6 producers and 6 consumers**

| Metric | Producers | Consumers |
|---|---|---|
| Maximum throughput (GB per second) | 0.793 | 2.673 |
| Maximum # records (millions per second) | 7.933 | 26.735 |

As the number of Producer/Consumer instances were increased, CPU utilization increased. At 24 Producers and Consumers, maximum CPU utilization was attained.

## Broker scalability

The Producer and Consumer variability tests were repeated using 6 partitions, but initially serviced by a single Broker. Additional Brokers were then added up to 3.

*Figure 16: Broker scalability: maximum throughput* on page 40 and *Figure 17: Broker scalability: maximum records per second* on page 40 show that Broker Consumer scalability (throughput and number of records per second) is almost perfectly linear for the first 3 Brokers. Broker scalability for Producers is linear but less steep.

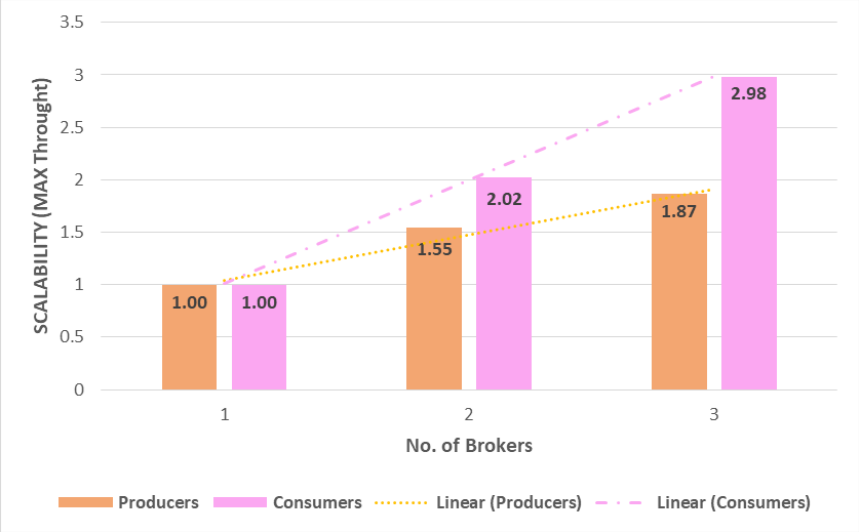


**Figure 16: Broker scalability: maximum throughput**

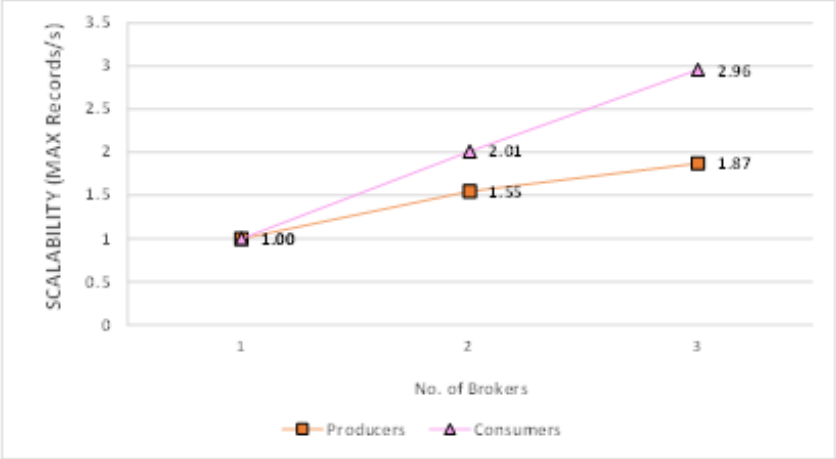


**Figure 17: Broker scalability: maximum records per second**

### Bottlenecks and throughput limitations

Throughput limitations were noticed in all tests. Beyond a particular point (number of Producers, Consumers and Partitions) performance drops below the maximum value. By analyzing resource utilization data gathered by PAT, these limitations were not caused by hardware bottlenecks. As the number of Producers and Consumers are raised, CPU Utilization increases. With the Producer/Consumer test, maximum CPU utilization was noticed with 24 Producers/Consumers.

## Message size variations

The record size (message) was varied from 100 B to 250 KB. Throughput increases as the record size increases up to 5 KB, and then starts to drop beyond that. *Figure 18: Message size versus throughput* on page 41 shows that the number of records that Kafka can send per second decreases as the records get larger in size.

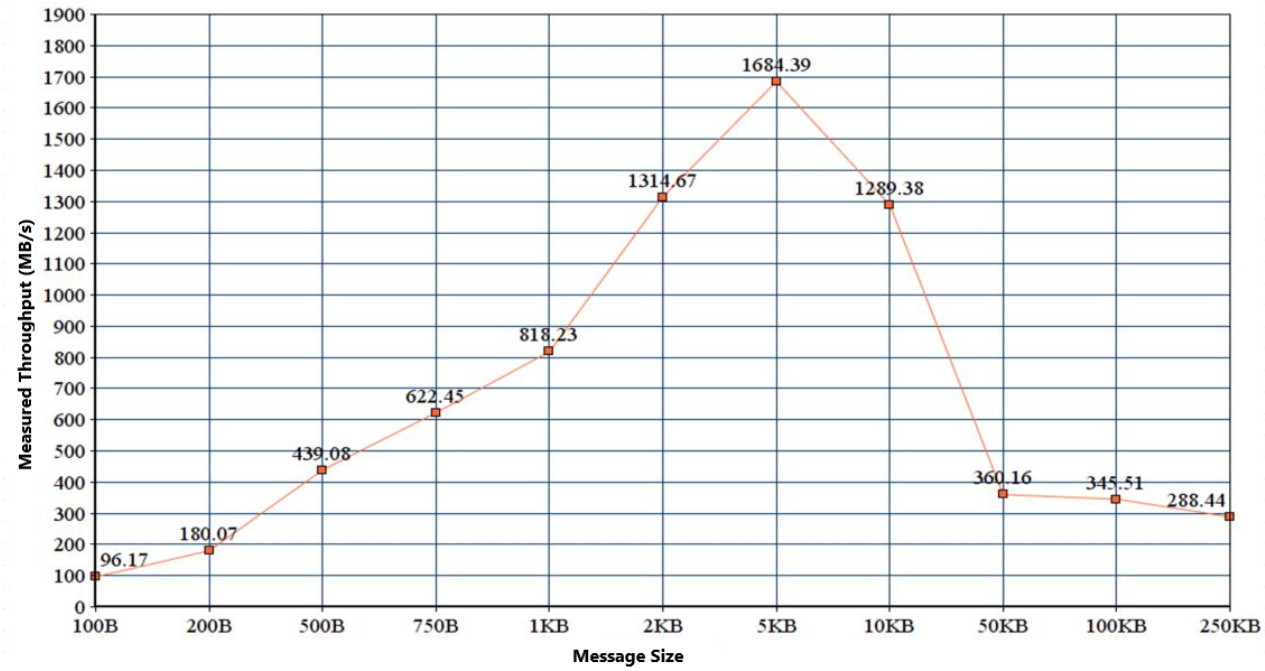


**Figure 18: Message size versus throughput**

# Conclusion

A number of tests were performed to characterize the performance of Kafka Real Time Data streaming on a standard Real-Time Data Streaming architecture based on a Dell EMC PowerEdge R640 hardware platform. The results shown in this chapter are specific to the SUT used in these tests. Performance results may vary for larger-sized clusters and different hardware architectures. These results can be used as guidelines, but Dell EMC recommends that throughput limitation points are obtained for particular configurations in order to determine what works best.

Please contact the *Dell EMC Customer Solution Centers* for specific assistance.

# Appendix A

## Tested component versions

**Topics:**



This appendix describes the versions of software and firmware used during validation of this architecture.

## Software versions

*Table 23: Tested software versions* on page 43 shows the software versions that were tested for this architecture.

**Table 23: Tested software versions**

| Component | Version |
|---|---|
| Operating system | RHEL 7.5 |
| Confluent Enterprise | 5.0 |

## Network switch firmware versions

*Table 24: Tested switch firmware versions* on page 43 shows the network switch firmware versions that were tested for this architecture.

**Table 24: Tested switch firmware versions**

| Component | Version |
|---|---|
| Dell EMC S5148-ON | 9.12(1.0) |
| Dell EMC S3048-ON | 9.11(2.4) |

## Dell EMC PowerEdge R640 firmware versions

*Table 25: Dell EMC PowerEdge R640 tested firmware versions* on page 43 shows the Dell EMC PowerEdge R640 firmware versions that were tested for this architecture.

**Table 25: Dell EMC PowerEdge R640 tested firmware versions**

| Component | Version |
|---|---|
| *BIOS* | 1.6.12 |
| *iDRAC* with *LC* | 3.21.26.22 |
| Mellanox ConnectX-4 LX 25 GbE SFP *rNDC* | 14.21.30.12 |
| Driver for *OS* deployment | 18.10.17 |
| Dell EMC 12 Gb expander firmware for 14G servers | 2.25 |
| Dell EMC PERC H740P | 50.5.0.1750 |
| *CPLD* | 1.0.2 |

# Appendix B

## Related documentation

**Topics:**



This chapter provides references to partner documentation related to this architecture.

In addition to this architecture guide the partner products documentation provided here will further your understanding of Real-Time Data Streaming.

## Confluent documentation

Related Confluent documentation includes:

- *Confluent Enterprise Reference Architecture*
- *Confluent Platform Documentation*

## Dell EMC documentation

Related Dell EMC documentation includes:

- *Running Kafka with Dell EMC Isilon White Paper*
- *Dell EMC Configuration Guide for the S5148-ON System 9.14.0.0*

## Operating systems documentation

Related operating systems documentation includes:

- *CentOS Documentation*
- *Red Hat Enterprise Linux Server Documentation*

## Apache Software Foundation documentation

Related Apache Software Foundation documentation includes:

- *Apache Hadoop Documentation*
- *Apache Kafka Documentation*

# Appendix C

## References

**Topics:**



Additional information can be obtained at *Big Data Analytics Info Hub for Ready Solutions*.

If you need additional services or implementation help, please contact your Dell EMC sales representative.

## About Confluent

Founded by the team that built Apache Kafka, Confluent, Inc. builds a streaming platform that enables companies to easily access data as real-time streams. Confluent Enterprise improves Apache Kafka by expanding its integration capabilities, adding tools to optimize and manage Kafka clusters, and methods to ensure the streams are secure. Confluent Enterprise makes Kafka easier to build and easier to operate.


## About 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 businesses become more successful and competitive.

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


## To learn more

For more information on this architecture, visit *Big Data and Big Data Analytics Solutions*.

# Glossary

**API**
Application Programming Interface

**BDaaS**
Big Data as a Service

**BIOS**
Basic Input-Output System

**BMC**
Baseboard Management Controller

**BMP**
Bare Metal Provisioning

**BOSS**
The Boot Optimized Storage Solution enables customers to segregate operating system and data on server-internal storage. This is helpful in the Hyper-Converged Infrastructure and Software Defined Storage arenas, to separate operating system drives from data drives, and implement hardware *RAID* mirroring (RAID1) for *OS* drives.

**Clos**
A multiple-stage, non-blocking network switch architecture. It reduces the number of required ports within a network switch fabric.

**CPLD**
A Complex Programmable Logic Device combines programmable AND/OR arrays, and macrocells.

**CPU**
Central Processing Unit

**DBMS**
Database Management System

**DIMM**
Dual In-line Memory Module

**ECMP**
Equal Cost Multi-Path

**EDW**
Enterprise Data Warehouse

**EIPT**
The Dell EMC Enterprise Infrastructure Planning Tool is a model-driven tool supporting a large number of products and configurations for infrastructure sizing purposes.

**EoR**
End-of-Row Switch/Router

**ETL**

Extract, Transform, Load is a process for extracting data from various data sources; transforming the data into proper structure for storage; and then loading the data into a data store.

**FQDD**

A Fully Qualified Device Descriptor is a method used to describe a particular component within a system or subsystem, and is used for system management and other purposes.

**FQDN**

A Fully Qualified Domain Name is the portion of an Internet Uniform Resource Locator (URL) that fully identifies the server to which an Internet request is addressed. The FQDN includes the second-level domain name, such as "dell.com", and any other levels as required.

**HA**

High Availability

**Hadoop**

Hadoop is an open source, distributed framework from the Apache Software Foundation that manages data processing and storage for clustered Big Data applications.

**HBA**

Host Bus Adapter

**HBase**

Apache HBase is *Hadoop*'s distributed, scalable database.

**HDD**

Hard Disk Drive

**HDFS**

*Hadoop* Distributed File System

**HTTP**

Hypertext Transfer Protocol is the set of rules for transferring files on the World Wide Web.

**HVE**

*Hadoop* Virtualization Extensions

**iDRAC**

Integrated Dell Remote Access with Lifecycle Controller

**IPMI**

Intelligent Platform Management Interface

**JBOD**

Just a Bunch of Disks

**KPI**

Key Performance Indicators are critical progress markers that provide focus on what matters most to an organization, in terms of strategic and operational improvement, and informed decision making.

**KSQL**

Confluent KSQL is an open source *SQL* engine that enables processing of real-time streaming data against Apache Kafka.

**LACP**
Link Aggregation Control Protocol

**LAG**
Link Aggregation Group

**LC**
Lifecycle Controller

**LFF**
Large Form Factor 3.5" industry-standard disk drives

**LOM**
Lights-out management enables a system administrator to monitor and manage servers remotely.

**Microservices**
The microservices architecture style structures applications as collections of related services. Microservices enable continuous delivery and deployment of applications.

**MTU**
A maximum transmission unit is the largest size packet or frame, in octets, that can be sent over a packet/frame-based computer network.

**NIC**
Network Interface Card

**NL-SAS**
Near Line Serial Attached *SCSI*

**NoSQL**
Not Only *SQL* is an alternative to relational databases. Data is put into tables, and the data schema is modeled before the database is built.

**NTP**
Network Time Protocol

**NVMe**
Non-Volatile Memory Express is a host controller interface and storage protocol that enables high-speed data transfers between systems and solid-state drives (*SSD*), over a computer's Peripheral Component Interconnect Express (*PCIe*) bus.

**OS**
Operating System

**OS-HCTK**
A configuration utility with sample scripts and configuration files used to automate the setup and configuration of *BIOS* and *RAID* settings for Dell EMC servers in OpenStack and *Hadoop* open source software solutions.

**PAM**
Pluggable Authentication Modules, a centralized authentication method for Linux systems.

**PCIe**
Peripheral Component Interconnect Express is a standard for high-bandwidth connections between computers and peripherals.

**PERC**

Dell EMC PowerEdge RAID Controller is Dell EMC's family of enterprise-class controllers designed for enhanced performance, increased reliability, fault tolerance, and simplified management — providing a powerful, easy-to-manage way to create a robust infrastructure and help maximize server uptime.

**Proxy**

A service that acts as an intermediary between computer networks, often used to provide secure Internet access to devices that do not have a routed connection.

**RAID**

Redundant Array of Inexpensive Disks

**RDBMS**

Relational Database Management System is a database management system (*DBMS*) that forms the basis for *SQL*, and all modern database systems.

**RDIMM**

Registered Dual In-line Memory Module

**Rebalancer**

A Rebalancer redistributes cluster data and indexes among available nodes when one or more nodes have been added to, or removed from, a cluster.

**Redfish**

Redfish is a standard *API*, designed by the Distributed Management Task Force to provide management for converged, hybrid IT as well as Software Defined Data Centers.

**Replicator**

The Confluent Replicator enables an operator to reliably replicate topics from one Kafka cluster to another.

**REST**

REpresentational State Transfer is an architectural style for developing web services, using standard *HTTP* status codes.

**RESTful API**

A RESTful *API* uses *HTTP* requests to GET, PUT, POST, and DELETE data. It is based upon *REST* technology.

**rNDC**

Rack Select Network Daughter Card

**RocksDB**

RocksDB is an embedded, persistent, high-performance database for key-value data.

**RPM**

Red Hat Package Manager

**RSTP**

Rapid Spanning Tree Protocol

**RTO**

Recovery Time Objectives

**RU**

A Rack Unit measures 1.75 inches, or 44.45 mm, in a 19-inch or 23-inch electronic equipment rack frame.

**SAS**

Serial-attached *SCSI* is a method for accessing computer peripherals, using a serial (one bit at a time) means of digital data transfer over thin cables.

**SATA**

Serial Advanced Technology Attachment is a standard for connecting and transferring data to and from hard disk drives (*HDD*s) to computers.

**Schema**

A schema defines a database's record structure.

**SCSI**

Small Computing System Interface

**SFF**

Small Form Factor 2.5" industry-standard disk drives

**SFP28**

Small Form-Factor Pluggable 28 is the third generation of the SFP interconnect systems designed for 25G performance, as per the IEEE 802.3by specification (25GBASE-CR).

**SIEM**

Security Information and Event Management

**SLA**

Service Level Agreement

**SP**

Scalable Processor

**SQL**

Structured Query Language is a standard programming language used to manage relational databases, and perform operations on their data.

**SSD**

Solid-state Drive (or Solid-state Disk)

**TCP**

Transmission Control Protocol is a standard that defines how to both establish and maintain network conversations.

**TCP Offload Engine**

A technology used to move the *TCP* stack processing from the *CPU* to a dedicated processor.

**THP**

Transparent Huge Pages

**ToR**

Top-of-Rack Switch/Router

**UID**

A code identifying each user on a Unix and/or Unix-like computer system

**VLT**

Virtual Link Trunking

**VRRP**
Virtual Router Redundancy Protocol

**YARN**
Yet Another Resource Negotiator

**ZooKeeper**
Apache ZooKeeper is a high-performance coordination service for distributed applications.

# Index

## L



## M



## N



## O



## P



## R



## S



## T

## V



## W



## Z