HPE Reference Architecture for data pipeline with Apache Kafka 2.1 and Apache Spark 3.0 on HPE ProLiant DL Servers

Using Red Hat OpenShift Container Platform 4.10 and HPE Ezmeral Data Fabric 6.2

CONTENTS

EXECUTIVE SUMMARY

As enterprises get larger over time, the amount of data they tend to accumulate undoubtedly increases. Enterprises are continuously acquiring data - from their customers, suppliers, partners, and competitors in their market. In addition, a growing number of real-time data sources such as smart devices, log data, sensors, social media, and clickstream data are generating a massive volume of data. As data gets generated in near real-time for consumption, the data must be processed before it can be consumed for insights. The real-time analytics are key business drivers across multiple industry verticals, including Financial Services, Healthcare, Retail, and Telecom. The exploding data growth results in complexity for enterprises responsible for ingesting, processing and storing datasets and deriving key business insights.

This Reference Architecture is based on a modular end-to-end data pipeline framework running on HPE Ezmeral Runtime Enterprise and HPE Ezmeral Data Fabric, and addresses some key challenges:

- Building a Decoupled Microservice-based data pipeline framework with the right technologies and tools.
- To accommodate the requirements of heterogeneous workloads such as batch, near real-time, and interactive analytics on the single framework.
- To glue functional modules in the data pipeline and to handle heterogeneous data sources with the rapidly growing data volumes to get insights.
- To build a robust, full-stack pipeline with flexibility and modularity for each tier such as data ingestion, data processing, data storage, etc.

The modular based end-to-end data pipeline framework is an ideal solution for real-time analytics workloads, as data makes a journey from Edge-to-Core and has distinct phases. The initial phases of the pipeline, such as data ingestion and data processing, are significant phases before aggregating the data for dashboards or using processed data for predictive analytics with machine learning models. For enterprises, the right scale out solution helps organizations turn data insights into business actions. This Reference Architecture highlights the performance characteristics for data ingestion using Apache Kafka and data processing using Apache Spark. The solution leverages HPE Ezmeral Runtime Enterprise 5.3 to run Apache Kafka and Apache Spark as containers. The processed data is persisted on HPE Ezmeral Data Fabric 6.2.

HPE Ezmeral Runtime Enterprise is a secure, enterprise-grade platform to build and deploy cloud-native and non-cloud-native (i.e., legacy) applications at scale across data centers, multiple clouds, and at the edge for a wide range of use cases. It provides all the tools enterprise customers need to build, modernize, deploy, monitor, and manage a wide range of AI and analytics workloads to unleash their data's full potential and accelerate their data-driven digital transformation.

HPE Ezmeral Data Fabric is a platform for data-driven analytics, Machine Learning (ML), and AI workloads. The platform serves as a secure data store and provides file storage, NoSQL databases, object storage, and event streams. The patented file system architecture was designed and built for performance, reliability, and scalability.

Target audience: This document is intended for subject matter experts, domain experts, data engineers, IT operation managers, pre-sales engineers, services consultants, partner engineers, and customers who are interested in implementing big data analytics workloads in their existing environment or create a new deployment.

Document purpose: This Reference Architecture highlights performance characteristics using Apache Kafka and Apache Spark running on the HPE Ezmeral Container platform and HPE Ezmeral Data Fabric 6.2. Organizations can readily apply the blueprint tested in this document to implement a microservice based data ingestion and data processing environment. This document also provides details about data ingestion and data processing of different sized near real-time data and gives key technology personas to decide on when building production ready environments with guidelines for optimizing configurations.

This Reference Architecture describes solution testing performed in October 2022.

SOLUTION OVERVIEW

The Microservice based end-to-end data pipeline architecture consists of several layers including data ingestion, data processing, datastore, data lake, and data visualization along with AI/ML. Organizations find it complex and error-prone when implementing a data pipeline, especially difficult to select technologies and tools that have proven to work along with the right set of hardware configurations.

In this Reference Architecture, we provide guidance on tools selection and integration with infrastructure configurations. Our framework is fully automated with Helm charts and designed with flexibility and agility in mind. Each functional layer is a module that can be used individually and can be scaled out or down independently, however, each layer can communicate and collaborate with other layers to perform integrated functions seamlessly.
Figure 1 solution architecture for Microservice based end-to-end data pipeline:

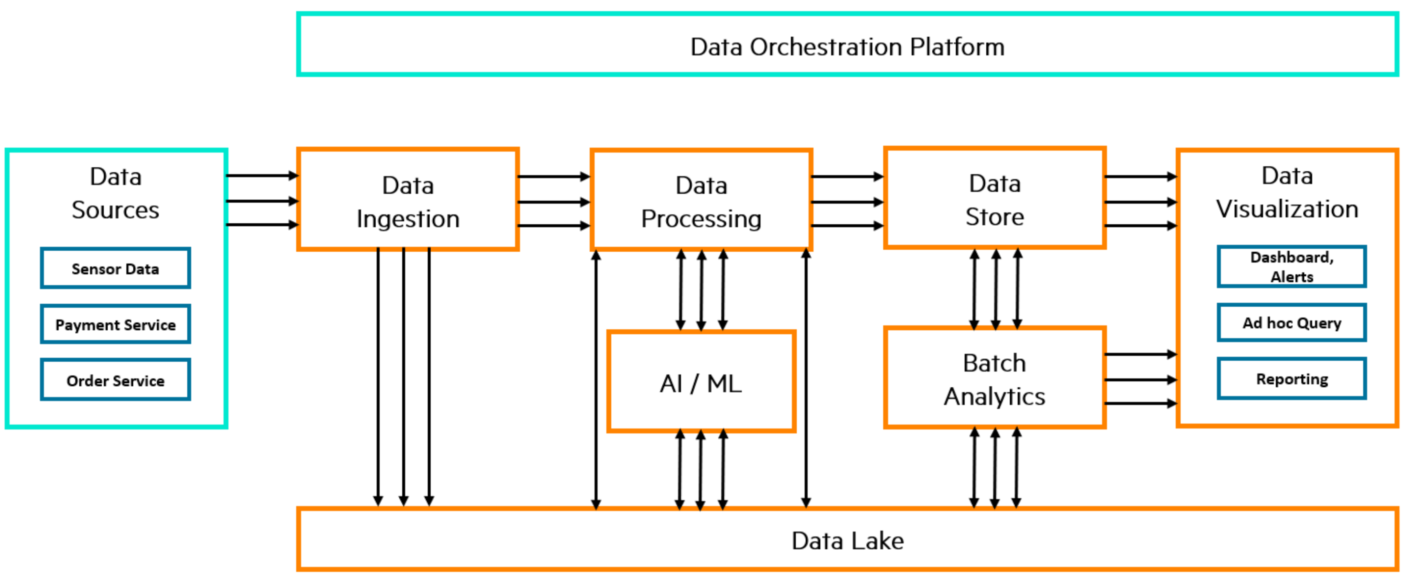

FIGURE 1. Solution Architecture for Microservices based end-to-end data pipeline

In this solution, the end-to-end data pipeline is deployed on Kubernetes cluster orchestrated by Red Hat OpenShift Container Platform and HPE Ezmeral Data Fabric is used for persisting both raw and processed data. As this Reference Architecture focuses on performance characterization of data ingestion and data processing layer and functionality of other layers are not discussed in this solution.

Data Ingestion:
In this phase, data is moved from a source to a destination, where the destination can be a data lake to run batch analytics or a data processing layer to run analytics to predict the event at a near-real time. The data ingestion is the backbone of any analytics framework as downstream systems highly rely on consistent and accessible data. In this test, we use Kafka as data ingestion tool to transport the data to the processing layer and data lake.

Data Processing:
The data collected from the previous layer must be processed in this layer and it is the first layer for analytics to happen. It serves as the decision layer for both batch-processing for offline analytics and near-real time processing for streaming analytics. In this test, we use Spark as a processing tool to support both batch and near-real time analytics.

Data Lake:
It is a centralized storage repository that holds a vast amount of raw data in its native format to serve all other layers of the data pipeline. In this test, we use HPE Ezmeral Data Fabric for data persistence which supports both CSI plugins and NFS mounts.

For detailed information on HPE Ezmeral Data Fabric, refer https://www.hpe.com/in/en/software/data-fabric.html.

For detailed information on Red Hat OpenShift Container Platform, refer https://www.redhat.com/en/technologies/cloud-computing/openshift/container-platform.

The key features of the solution are:

- The right choice of Microservice based simplified infrastructure is one that is scalable easy to manage and yields optimal performance.
- Seamless integration to build and deploy the model into production for real-time prediction.
- There is no vendor lock-in at any layer in the data pipeline framework.
- The pipeline can be deployed as a whole or can be plugged into the existing layer through integration.
- It supports heterogenous workloads along with multi-tenancy support for the users.

SOLUTION COMPONENTS

In this Reference Architecture, there are distinct building blocks that can be used individually or can be stitched together to form a data solution that fits into the customer's environment or addresses the customer's unique challenges. This section explains details about Red Hat OpenShift Container Platform (RHOCP) control, compute and HPE Ezmeral Data Fabric (HPE EDF) configuring on the HPE ProLiant DL325 Gen10 Plus v2 servers and HPE ProLiant DL385 Gen10 Plus v2 servers to implement the data pipeline. The blocks illustrated in this section can be modified (for example, processor model, memory, etc.) to meet the customer's specific requirements.

Our solution provides maximum elasticity allowing organizations to build and scale their infrastructure in alignment with different phases of analytics implementations. Specific workload requirements will drive hardware selection and configuration. Below are the infrastructure building blocks required to implement the data pipeline.

Table 1 shows the solution components.

TABLE 1. Solution components

Blocks	Model
RHOCP Control Node	3x HPE DL325 Gen10 Plus v2 servers
RHOCP Compute Node	3x HPE DL385 Gen10 Plus v2 servers
Persistent Storage Block	5x HPE DL385 Gen10 Plus v2 servers
Network Block	2x Aruba 8325-48Y8C 48p 25G SFP/+/28 8p 100G 1x Aruba 6300M 48-port 1GbE
Rack Block	1200mm or 1075mm

Single rack configuration:

The following rack design demonstrates a single rack configuration of the data pipeline on HPE ProLiant servers. As shown in Figure 2 the solution is designed with modularity, flexibility, and performance in mind.

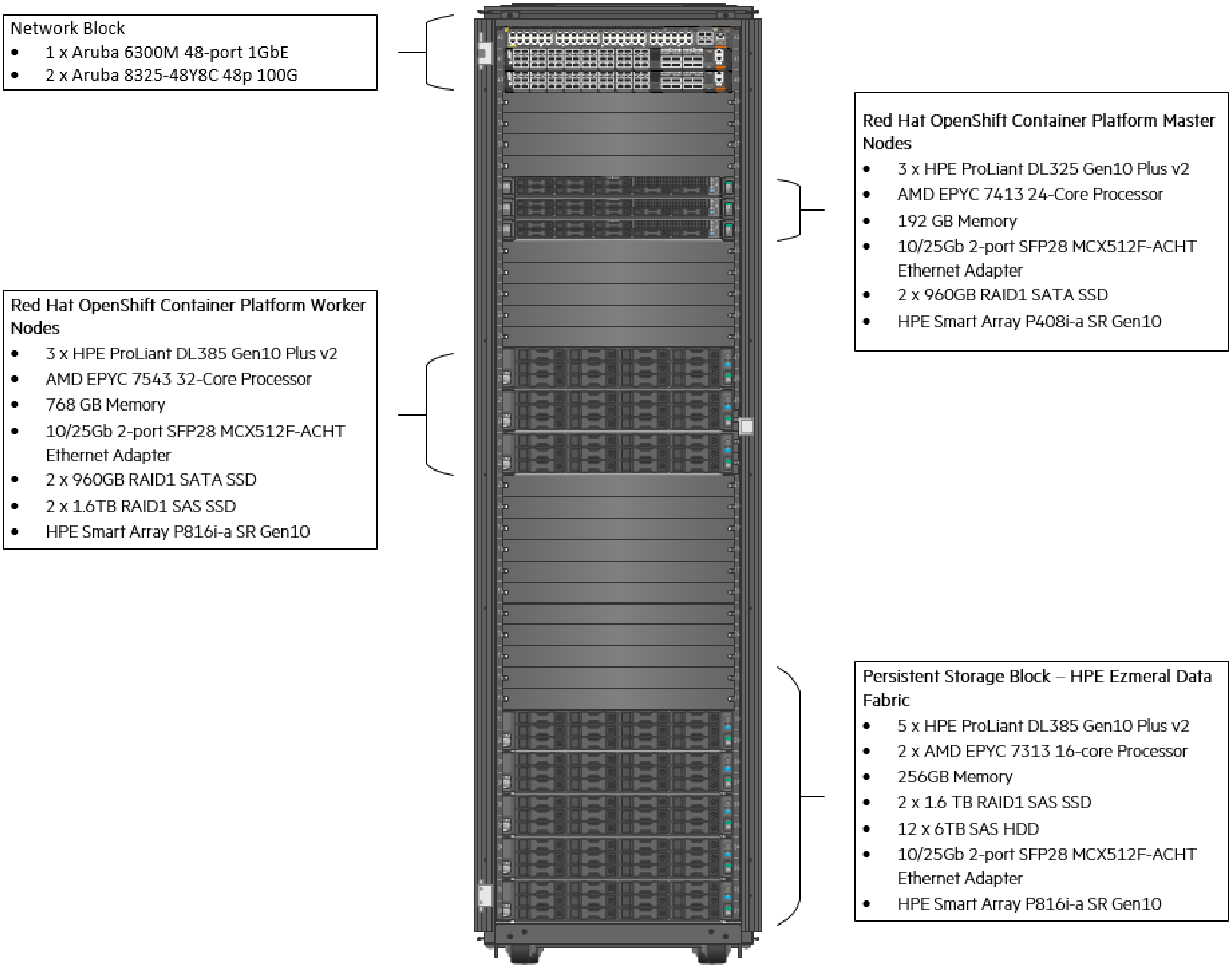

FIGURE 2. Basic conceptual diagram of single rack for micro-service based end-to-end data pipeline

Hardware

- **HPE ProLiant DL325 Gen10 Plus v2 servers (RHOCP Control Plane Nodes):** The HPE ProLiant DL325 Gen10 Plus v2 servers is a secure and versatile single socket server for virtualization, I/O intensive and storage intensive workloads. HPE ProLiant DL325 Gen10 Plus servers offers the 2nd generation AMD® EPYC™ 7003 Series processor in a densely packed 1U,1P form factor. These nodes serve as RHOCP Control Plane nodes for data pipeline.
- **HPE ProLiant DL385 Gen10 Plus v2 servers (RHOCP Compute Nodes/HPE EDF Data Nodes):** The HPE ProLiant DL385 Gen10 Plus v2 Servers is a purpose-built platform for virtualization, High Performance Compute and memory centric workloads. The HPE ProLiant DL385 Gen10 Plus v2 Servers offers the 2nd generation AMD® EPYC™ 7003 Series processor in a densely packed 2U,2P form factor. The HPE ProLiant DL385 Plus v2 servers performs as elastic storage nodes and allows organizations to build multi-temperature data lake environments. The HPE ProLiant DL385 Gen10 Plus v2 servers can be leveraged as a storage block for low latency workloads, and for environments where failure domain considerations prevent higher density storage nodes.

For more detailed information, refer to:

HPE ProLiant DL325 Gen10 Plus v2 servers, https:/www.hpe.com/psnow/doc/a50002564enw.

HPE ProLiant DL385 Gen10 Plus v2 servers, https://www.hpe.com/psnow/doc/a50002560enw.

- **Aruba 8325 switch:** The Aruba 8325 switch is an enterprise-class, game-changing solution, offering a flexible approach to deal with the new application, security, and scalability demands of the mobile, cloud, and IoT (Internet of Things) era. It offers the following benefits:
  - Simplify your IT operations with AOS-CX
  - Accelerating IT Provisioning
  - Unparalleled visibility and analytics
  - No downtime, even during upgrades

Software

The following software's were installed in our configuration. Red Hat OpenShift Container Platform was used for building the data pipeline. HPE Ezmeral Data Fabric was used as a Data Lake.

Red Hat OpenShift Container Platform

Red Hat OpenShift Container Platform unites developers and IT operations on a single platform to build, deploy, and manage applications consistently across hybrid cloud and multi-cloud infrastructures. Red Hat OpenShift helps businesses achieve greater value by delivering modern and traditional applications with shorter development cycles and lower operating costs. Red Hat OpenShift is built on open-source innovation and industry standards, including Kubernetes and Red Hat Enterprise Linux.

Red Hat Enterprise Linux CoreOS

Red Hat OpenShift Container Platform uses Red Hat Enterprise Linux CoreOS (RHCOS), a new container-oriented operating system that combines some of the best features and functions of the CoreOS and Red Hat Atomic Host operating systems. RHCOS is specifically designed for running containerized applications from RHOCP and works with new tools to provide fast installation, operator-based management, and simplified upgrades. For RHOCP 4.10 deployment on bare metal infrastructure, you must use RHCOS for all control plane machines, Bootstrap node, and worker nodes.

HPE Ezmeral Data Fabric

HPE Ezmeral Data Fabric is a platform for data-driven analytics, ML, and AI workloads. The platform serves as a secure data store and provides file storage, NoSQL databases, object storage, and event streams. The patented filesystem architecture was designed and built for performance, reliability, and scalability. HPE Ezmeral Data Fabric was used as a Data Lake – for storing large volume of data in a distributed manner along with fault tolerance and Container Persistent Storage – provides persistent storage for RHOCP compute layer components with the help CSI plugin interface.

TABLE 2. IDP (Intelligent Data Pipeline) software components

Software	Description
Red Hat OpenShift Container Platform 4.10	Container Platform
HPE Ezmeral Data Fabric 6.2	Recommended Data Store
Red Hat Enterprise Linux CoreOS 4.10	Recommended OS for RHOCP Control Nodes
Red Hat Enterprise Linux 8.5	Recommended OS for RHOCP Compute Nodes
Kafka	Distributed event streaming platform
Zookeeper	Centralized service for cluster co-ordination
Spark streaming	Processing Engine for Real Time Streaming Data
Spark SQL	Processing Engine for structured data Processing
Redis	In-Memory Key-Value data structure store
Cassandra	NoSQL distributed database

Software	Description
Grafana	Analytics and interactive visualization web application

Building the solution environment

In this solution, Red Hat OpenShift Container Platform 4.10 was deployed with three HPE ProLiant DL325 Gen10 Plus v2 servers as master nodes and three HPE ProLiant DL385 Gen10 Plus v2 servers as worker nodes. For more details about deploying Red Hat OpenShift Container Platform are available in reference link, https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html/installing/index.

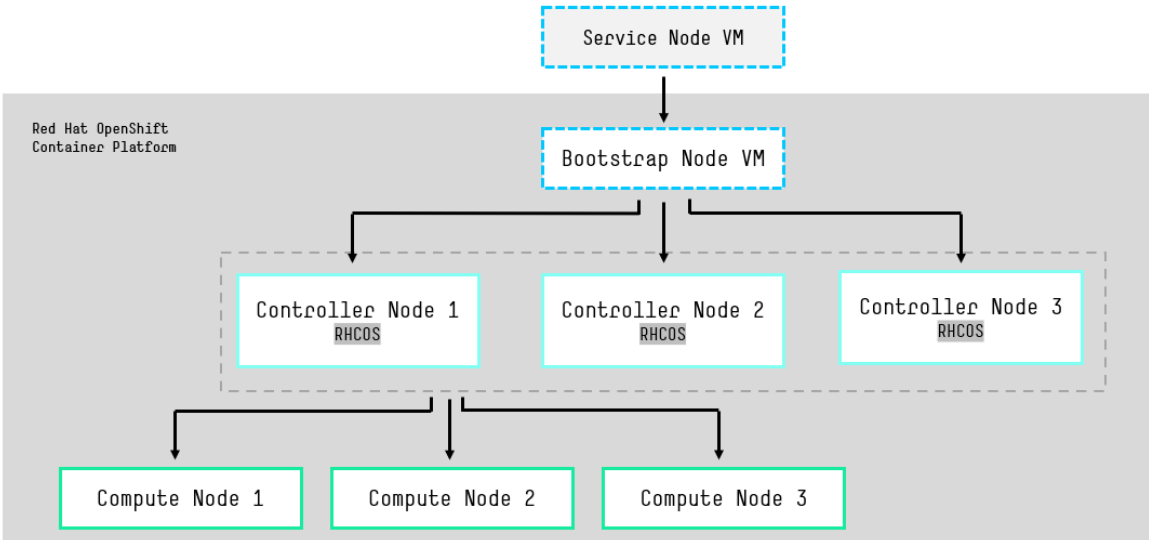

FIGURE 3. Physical architecture of Red Hat OpenShift Container Platform cluster

To support data persistence for the end-to-end data pipeline HPE Ezmeral Data Fabric was deployed on five HPE ProLiant DL385 Gen10 Plus v2 servers.

For this solution, five instances of Apache Kafka and Apache Spark were deployed, the Yaml scripts mentioned in Appendix B are used in Helm charts for the deployment. Similarly, a storage class and CSI driver were created to access HPE Ezmeral Data Fabric cluster, and the scripts mentioned in Appendix B.

BEST PRACTICES AND CONFIGURATION GUIDANCE FOR THE SOLUTION

Below are some of the best practices followed during the deployment of data pipeline solution.

Data Pipeline Components Security best practices

For smooth and secure integration of all the end-to-end data pipeline components, the following security best practices are recommended for respective components in the pipeline.

TABLE 3. Description of files for Kafka Consumer Application Deployment

Parameter	Value
RunAsUser:	1000
ReadOnlyRootFilesystem:	True
AllowPrivilegeEscalation:	False
SeLinuxOptions: level:	"s0:c123,c456"

Kubernetes best practices

The following best practices are recommended to use Kubernetes in an optimized manner.

TABLE 4. Best Practices while using Kubernetes

Parameter	Description
Namespaces	Segregate Workloads and AI/ML into different Kubernetes namespaces
Resource requests and limits	Set resource requests and limits to avoid a single component from draining the cluster resources
Node affinity	Use node affinity to place certain pods at specific nodes for better resource utilization and ease of management
Readiness and Liveness Probe	Set Readiness and Liveness Probe to proactively avoid pod failures
RBAC	Use Role-Based Access Control (RBAC) to control the access of the Kubernetes cluster among users
Helm	Parameterize the workload deployment using helm

Server best practices

HPE ProLiant DL3x5 Gen10 Plus Servers BIOS Configuration

Hewlett Packard Enterprise recommends changing the default BIOS settings to the following using Workload Profile High-Performance Compute (HPC) on all servers hosting the end-to-end data pipeline services to ensure the highest performance.

TABLE 5. HPE ProLiant DL3x0 Gen10 BIOS settings

Parameter	Description	Settings
Boot_Mode	Recommended Operating System	UEFI_Mode
UEFI_Optimized_Boot	Java Development Kit	Enable
Workload_Profile	Sets power and performance settings for application workloads	High_Performance_Compute
Processor_x2APIC_Support	Enables or disables x2APIC support	Disabled
Intel_VT	Controls whether a Virtual Machine Manager (VMM) supporting Virtualization Technology can use hardware capabilities provided by UEFI Intel processors	Disabled
Intel_VT-d	Enables or disables Intel Virtualization Technology for Directed I/O (VT-d) on a Virtual Machine Manager (VMM)	Disabled
SR-IOV	Enables or disables the BIOS to allocate more PCI resources to PCIe devices	Disabled
Power_Regulator	Sets the power regulator mode	Static_High_Performance_Mode
Minimum_Processor_Idle_Power_Core_C-State	Sets the lowest processor idle power state (C-State)	No_C-States
Minimum_Processor_Idle_Power_Package_C-State	Sets the lowest processor idle power state (C-State)	No_Package_State
Energy/Performance_Bias	To optimize the processor's performance and power usage	Maximum_Performance
Collaborative_Power_Control	Enables or disables collaborative power control for operating systems that support the Processor Clocking Control (PCC) interface	Disabled
Intel_Hyper-Threading	Enables or disables the logical processor cores on processors supporting Intel Hyperthreading technology	Enabled
Intel_Turbo_Boost_Technology	Enables or disables Intel Turbo Boost Technology to control whether the processor transitions to a higher frequency than the processor's rated speed if the processor has available power and is within the temperature	Enabled
Energy_Efficient_Turbo	Controls whether the processor uses an energy efficient based policy	Disabled
Maximum_Memory_Bus_Frequency	Configures the system to run memory at a lower maximum speed than that supported by the installed processor and DIMM configuration	Auto
Channel_Interleaving	Enables or disables a higher level of memory interleaving.	Enabled
Intel_UPI_Link_Power_Management	To place the Ultra Path Interconnect (UPI) links into a low power state when the links are not being used	Disabled
Intel_TXT_support	Enables or disables Intel TXT (Trusted Execution Technology) support for servers with Intel processors	Disabled

Parameter	Description	Settings
Embedded_SATA_Configuration	Sets the mode for the embedded SATA controller	Enable_AHCI_Support
NUMA_Group_Size_Optimization	The number of logical processors in a NUMA (Non-Uniform Memory Access) node.	Clustered
Uncore_Frequency_Scaling	Controls the frequency scaling of the processor's internal buses (the uncore).	Maximum

Workload Description

This section of the Reference Architecture demonstrates performance characterization of Apache Kafka and Apache Spark running as containers on Red Hat OpenShift Container Platform. The performance tests were carried out separately. The performance metrics of CPU, network and memory were captured using Grafana dashboard.

Apache Kafka Testing

For testing Apache Kafka, a native performance tools “kafka_producer_perf_test” and “kafka_consumer_perf_test” were used. These native tools were deployed as a client running as pods and created a job to run for varied sizes of the messages. The size of messages varied from 50 million to 2 billion messages for the performance test of Apache Kafka accordingly. From Kafka client pod “kafka_producer_perf_test” and “kafka_producer_perf_test” tests were executed interactively, and metrics were captured using Grafana dashboard with JMX enabled. The Apache Kafka performance was evaluated using latency to process the messages and throughput observed during the test. Figure 4 shows the performance of Apache Kafka, the results show that the latency and the throughput were consistent for different messages size during the test. This demonstrates Apache Kafka performance testing with containers running on HPE ProLiant DL325 Gen10 Plus v2 servers (Control nodes) and HPE ProLiant DL385 Gen10 Plus v2 servers (Worker nodes), consistently performed well in maintaining the latency though the number of messages were increased. We noticed the solution can further scale linearly and provide high performance when network infrastructure supports required bandwidth to process the messages.

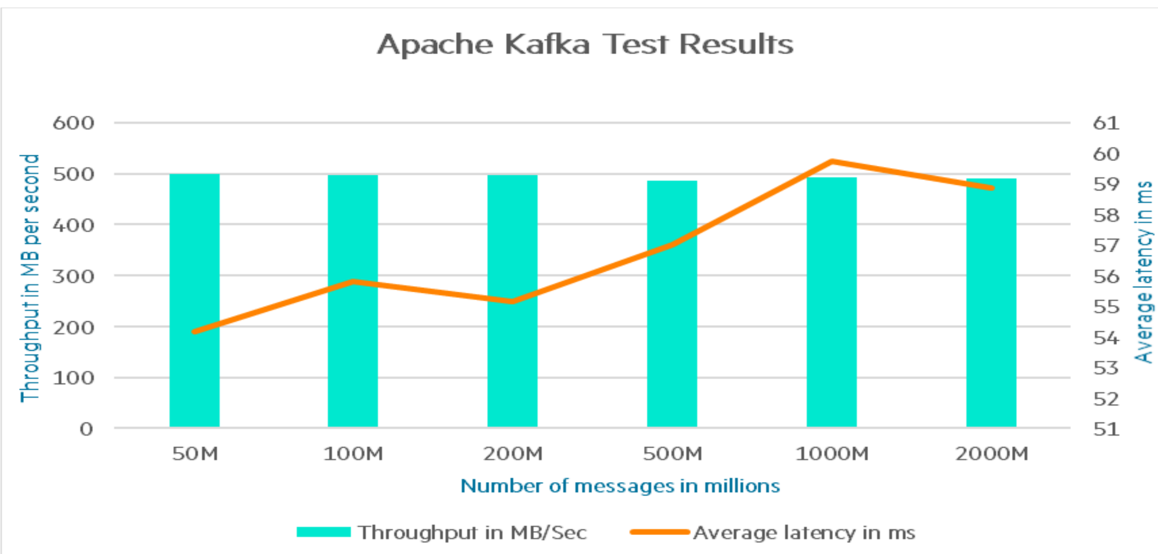

FIGURE 4. The graph shows Apache Kafka test results

Apache Spark Testing

For Apache Spark testing, the TPC-DS tool was used. The Apache Spark was tested with different data sets ranging from 100 GB to 1TB. The data generation and query execution were done using the TPC-DS tool to evaluate the performance. This procedure was done subsequently for different data size. For this testing five instances of Apache Spark were deployed as containers. Apache Spark's driver and executor memory was provisioned with 8GB memory. As the data size was varied for testing, memory for executor was changed accordingly to handle the workload capacity. During the test all 99 queries were executed, and the output was measured about the memory usage by the Apache Spark containers, duration of queries execution against the data size which was put under test. These metrics are captured through Grafana for analysis. Figure 5 shows the Apache Spark TPC-DS test performance results. The result shows the average time to execute queries scaled linearly as data size increased. This demonstrates that the solution exhibits optimal Apache Spark performance for linearly scaling data. When there is requirement to scale the solution as per data additional HPE ProLiant DL385 Gen10 Plus v2 servers (worker node) can be added to the cluster to gain more performance for Apache Spark workload.

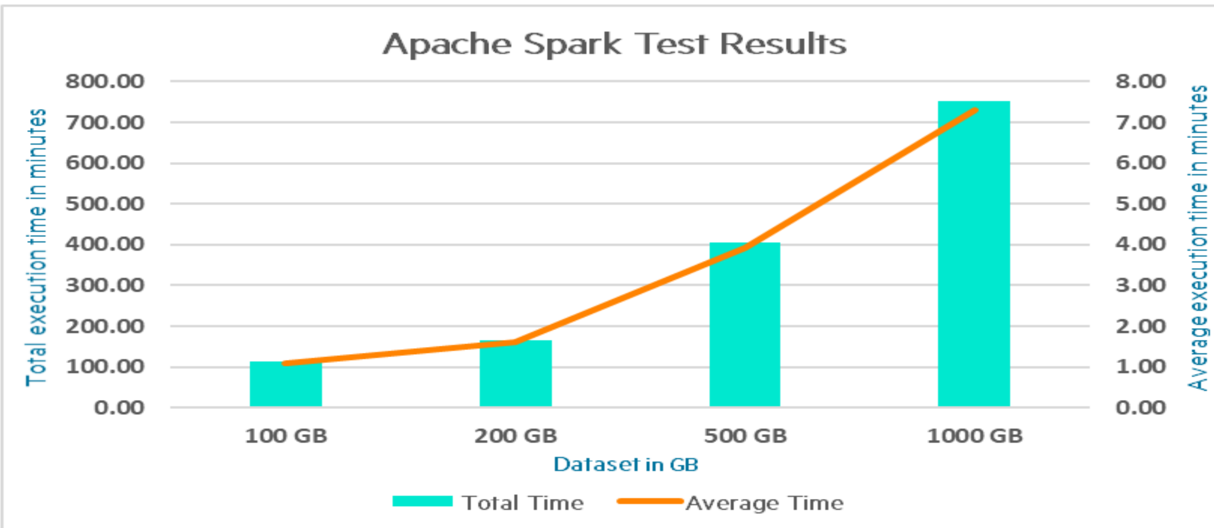

FIGURE 5. The graph shows Apache Spark test results.

SUMMARY

Hewlett Packard Enterprise allows the organization to derive new business insights from next-gen end-to-end data pipeline applications like fraud analytics, predictive analytics, streaming analytics, interactive analytics, etc. by providing a platform to store, manage, and process data in real-time or near-real-time. However, it is important to understand the requirements of critical stages such as data ingestion and data processing to design and build the solution to seamlessly scale and provide optimal performance. This Reference Architecture provides a validated end-to-end data pipeline to run Apache Kafka for data ingestion and Apache Spark for data processing on Kubernetes cluster which are hosted on HPE ProLiant DL325 Gen10 Plus v2 servers and HPE ProLiant DL385 Gen10 Plus v2 servers. The blueprint leverages HPE servers, storage, and networking, along with an integrated software stack and HPE advisory services to offer a scalable, flexible, and highly performant platform. It aids organizations in the rapid design and deployment of the data pipeline and gives customers the options to pick and choose the modules to be deployed based on their environment, data strategy, and business needs.

IMPLEMENTING A PROOF OF CONCEPT

As a matter of best practice for all deployments, Hewlett Packard Enterprise recommends implementing a proof-of-concept using a test environment that matches as closely as possible the planned production environment. In this way, proper performance and scalability characterizations can be obtained. For help with proof-of-concept, contact a Hewlett Packard Enterprise Services representative (https://www.hpe.com/us/en/services/consulting.html) or your Hewlett Packard Enterprise partner.

APPENDIX A: BILL OF MATERIALS

The Bill of Material (BOM) outlined in this section are based on a tested configuration for a single-rack Reference Architecture with three (3) Red Hat OpenShift Container Platform Master nodes (HPE ProLiant DL325 Gen10 Plus v2 servers), and three (3) Worker nodes (HPE ProLiant DL385 Gen10 Plus v2 servers). Five (5) storage nodes (HPE ProLiant DL385 Gen10 Plus v2 servers) were used to deploy HPE Ezmeral Data Fabric. Two (2) ToR (Top of Rack) switches were used for connectivity. The following tables depict the bill of materials.

Red Hat OpenShift Container Platform Master nodes BOM for HPE ProLiant DL325 Gen10 Plus v2 servers

Table A1 shows the bill of materials for the HPE ProLiant DL325 Gen10 Plus v2 servers.

TABLE A1. Bill of materials for HPE ProLiant DL325 Gen10 Plus v2 servers

Part number	Quantity	Description
P38471-B21	3	HPE ProLiant DL325 Gen10 Plus v2 8SFF Configure-to-order Server
P38675-B21	3	AMD EPYC 7413 2.65GHz 24-core 180W Processor for HPE
P07642-B21	36	HPE 16GB (1x16GB) Dual Rank x8 DDR4-3200 CAS-22-22-22 Registered Smart Memory Kit
P40503-B21	6	HPE 960GB SATA 6G Mixed Use SFF BC Multi-Vendor SSD
P40561-B21	6	HPE 1.6TB SAS 12G Mixed Use SFF BC PM1645a SSD
869081-B21	3	HPE Smart Array P408i-a SR Gen10 (8 Internal Lanes/2GB Cache) 12G SAS Modular LH Controller
P08449-B21	3	Intel I350-T4 Ethernet 1Gb 4-port BASE-T OCP3 Adapter for HPE
P13188-B21	3	Mellanox MCX512F-ACHT Ethernet 10/25Gb 2-port SFP28 Adapter for HPE
P41197-B21	3	HPE ProLiant DL325 Gen10 Plus v2 High-Performance Heat Sink Kit
P38997-B21	6	HPE 1600W Flex Slot Platinum Hot Plug Low Halogen Power Supply Kit

Red Hat OpenShift Container Platform Worker node BOM for HPE ProLiant DL385 Gen10 Plus v2 servers

Table A2 shows the bill of materials for the HPE ProLiant DL385 Gen10 Plus v2 servers.

TABLE A2. Bill of materials for the HPE ProLiant DL385 Gen10 Plus v2 servers

Part number	Quantity	Description
P38411-B21	3	HPE ProLiant DL385 Gen10 Plus v2 servers 8SFF Configure-to-order Server
P38687-B21	6	AMD EPYC 7543 2.8GHz 32-core 225W Processor for HPE
P07646-B21	72	HPE 32GB (1x32GB) Dual Rank x4 DDR4-3200 CAS-22-22-22 Registered Smart Memory Kit
P40561-B21	6	HPE 1.6TB SAS 12G Mixed Use SFF BC PM1645a SSD
P40503-B21	6	HPE 960GB SATA 6G Mixed Use SFF BC Multi-Vendor SSD
P13188-B21	3	Mellanox MCX512F-ACHT Ethernet 10/25Gb 2-port SFP28 Adapter for HPE
P01366-B21	3	HPE 96W Smart Storage Lithium-ion Battery with 145mm Cable Kit
804338-B21	3	HPE Smart Array P816i-a SR Gen10 (16 Internal Lanes/4GB Cache/SmartCache) 12G SAS Modular Controller
P08449-B21	3	Intel I350-T4 Ethernet 1Gb 4-port BASE-T OCP3 Adapter for HPE
P38997-B21	1	HPE 1600W Flex Slot Platinum Hot Plug Low Halogen Power Supply Kit

HPE Ezmeral Data Fabric storage node BOM for HPE ProLiant DL385 Gen10 Plus v2 servers

Table A3 shows the bill of materials for the HPE ProLiant DL385 Gen10 Plus v2 servers.

TABLE A3. Bill of materials for HPE ProLiant DL385 Gen10 Plus v2 servers

Part number	Quantity	Description
P38410-B21	5	HPE ProLiant DL385 Gen10 Plus v2 servers 12LFF Configure-to-order Server
P38669-B21	10	AMD EPYC 7313 3.0GHz 16-core 155W Processor for HPE
P07642-B21	80	HPE 16GB (1x16GB) Dual Rank x8 DDR4-3200 CAS-22-22-22 Registered Smart Memory Kit
P26922-B21	5	HPE ProLiant DL38X Gen10 Plus 2SFF x4 Tri-Mode 24G U.3 BC Front/Tertiary Drive Cage Kit
P40503-B21	10	HPE 960GB SATA 6G Mixed Use SFF BC Multi-Vendor SSD
P35418-B21	5	HPE ProLiant DL385 Gen10 Plus v2 2SFF 4x U.3 Premium Secondary Riser Kit
P40561-B21	10	HPE 1.6TB SAS 12G Mixed Use SFF BC PM1645a SSD
861746-B21	60	HPE 6TB SAS 12G Midline 7.2K LFF (3.5in) LP 1yr Wty 512e HDD
P13188-B21	5	Mellanox MCX512F-ACHT Ethernet 10/25Gb 2-port SFP28 Adapter for HPE
804338-B21	5	HPE Smart Array P816i-a SR Gen10 (16 Internal Lanes/4GB Cache/SmartCache) 12G SAS Modular Controller
P38997-B21	10	HPE 1600W Flex Slot Platinum Hot Plug Low Halogen Power Supply Kit
P13771-B21	5	HPE Trusted Platform Module 2.0 Gen10 Plus Black Rivets Kit
P14610-B21	5	HPE DL38X Gen10 Plus High-Performance Heat Sink Kit

BOM for network components of the solution

Table A4 shows the bill of materials for the Network components of the solution.

TABLE A4. Bill of materials for network components of the solution

Part number	Quantity	Description
R9F67A	2	Aruba 8325-32C Power to Port Airflow 6 Fans 2 Power Supply Units Bundle for HPE
R9F59A	2	Aruba 4-post Rack Kit for HPE
R9F63A	1	Aruba 6300M 48G Power to Port Airflow 2 Fans 1 Power Supply Unit Bundle for HPE
R9F61A	1	Aruba 6300M 12VDC 250W 100-240VAC Power to Port Airflow Power Supply Unit for HPE
R9F57A	1	Aruba 1U Universal 4-post Rack Mount Kit for HPE
845416-B21	6	HPE 100Gb QSFP28 to 4x25Gb SFP28 3m Direct Attach Copper Cable
C7535A	11	HPE RJ45 to RJ45 Cat5e Black M/M 7.6ft 1-pack Data Cable
R9F77A	1	Aruba 100G QSFP28 to QSFP28 1m Direct Attach Copper Cable for HPE

APPENDIX B: SCRIPTS

Yaml Script Apache Kafka

```
apiVersion: v1
kind: Service
metadata:
  name: kafka-svc
  labels:
    app: kafka
spec:
  ports:
  - port: 9093
    name: server
  clusterIP: None
  selector:
    app: kafka
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: kafka
spec:
  serviceName: kafka-svc
  replicas: {{ .Values.kafkaReplicas }}
  selector:
    matchLabels:
      app: kafka
  template:
    metadata:
      labels:
        app: kafka
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: nodetype
                operator: In
                values:
{{ toYaml .Values.nodes | indent 16 }}
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:
                  - key: "app"
                    operator: In
                    values:
                    - kafka
              topologyKey: "kubernetes.io/hostname"
      terminationGracePeriodSeconds: 300
      containers:
      - name: k8skafka
        imagePullPolicy: IfNotPresent
        image: {{ .Values.kafkaimage }}
        securityContext:
          allowPrivilegeEscalation: false
          seLinuxOptions:
            level: "s0:c123,c456"
```

```
        resources:
          requests:
            memory: "{{ .Values.resources.kafkamemory }}"
            cpu: {{ .Values.resources.kafkacpu }}
        ports:
        - containerPort: 9093
          name: server
        command:
        - sh
        - -c
        - "exec kafka-server-start.sh /opt/kafka/config/server.properties --override broker.id=${HOSTNAME##*-} \
          --override listeners=PLAINTEXT://:9093 \
          --override zookeeper.connect=zk-0.zk-svc.data-pipeline.svc.cluster.local:2181,zk-1.zk-svc.data-
pipeline.svc.cluster.local:2181,zk-2.zk-svc.data-pipeline.svc.cluster.local:2181 \
          --override log.dir=/var/lib/kafka \
          --override auto.create.topics.enable=true \
          --override auto.leader.rebalance.enable=true \
          --override background.threads=10 \
          --override compression.type=producer \
          --override delete.topic.enable=true \
          --override leader.imbalance.check.interval.seconds=300 \
          --override leader.imbalance.per.broker.percentage=10 \
          --override log.flush.interval.messages=9223372036854775807 \
          --override log.flush.offset.checkpoint.interval.ms=60000 \
          --override log.flush.scheduler.interval.ms=9223372036854775807 \
          --override log.retention.bytes=-1 \
          --override log.retention.hours=168 \
          --override log.roll.hours=168 \
          --override log.roll.jitter.hours=0 \
          --override log.segment.bytes=1073741824 \
          --override log.segment.delete.delay.ms=60000 \
          --override message.max.bytes=1000012 \
          --override min.insync.replicas=1 \
          --override num.io.threads=8 \
          --override num.network.threads=3 \
          --override num.recovery.threads.per.data.dir=1 \
          --override num.replica.fetchers=1 \
          --override offset.metadata.max.bytes=4096 \
          --override offsets.commit.required.acks=-1 \
          --override offsets.commit.timeout.ms=5000 \
          --override offsets.load.buffer.size=5242880 \
          --override offsets.retention.check.interval.ms=600000 \
          --override offsets.retention.minutes=1440 \
          --override offsets.topic.compression.codec=0 \
          --override offsets.topic.num.partitions=50 \
          --override offsets.topic.replication.factor=3 \
          --override offsets.topic.segment.bytes=104857600 \
          --override queued.max.requests=500 \
          --override quota.consumer.default=9223372036854775807 \
          --override quota.producer.default=9223372036854775807 \
          --override replica.fetch.min.bytes=1 \
          --override replica.fetch.wait.max.ms=500 \
          --override replica.high.watermark.checkpoint.interval.ms=5000 \
          --override replica.lag.time.max.ms=10000 \
          --override replica.socket.receive.buffer.bytes=65536 \
          --override replica.socket.timeout.ms=30000 \
          --override request.timeout.ms=30000 \
          --override socket.receive.buffer.bytes=102400 \
          --override socket.request.max.bytes=104857600 \
          --override socket.send.buffer.bytes=102400 \
```

```
          --override unclean.leader.election.enable=true \
          --override zookeeper.session.timeout.ms=6000 \
          --override zookeeper.set.acl=false \
          --override broker.id.generation.enable=true \
          --override connections.max.idle.ms=600000 \
          --override controlled.shutdown.enable=true \
          --override controlled.shutdown.max.retries=3 \
          --override controlled.shutdown.retry.backoff.ms=5000 \
          --override controller.socket.timeout.ms=30000 \
          --override default.replication.factor=1 \
          --override fetch.purgatory.purge.interval.requests=1000 \
          --override group.max.session.timeout.ms=300000 \
          --override group.min.session.timeout.ms=6000 \
          --override inter.broker.protocol.version=0.10.2-IV0 \
          --override log.cleaner.backoff.ms=15000 \
          --override log.cleaner.dedupe.buffer.size=134217728 \
          --override log.cleaner.delete.retention.ms=86400000 \
          --override log.cleaner.enable=true \
          --override log.cleaner.io.buffer.load.factor=0.9 \
          --override log.cleaner.io.buffer.size=524288 \
          --override log.cleaner.io.max.bytes.per.second=1.7976931348623157E308 \
          --override log.cleaner.min.cleanable.ratio=0.5 \
          --override log.cleaner.min.compaction.lag.ms=0 \
          --override log.cleaner.threads=1 \
          --override log.cleanup.policy=delete \
          --override log.index.interval.bytes=4096 \
          --override log.index.size.max.bytes=10485760 \
          --override log.message.timestamp.difference.max.ms=9223372036854775807 \
          --override log.message.timestamp.type=CreateTime \
          --override log.preallocate=false \
          --override log.retention.check.interval.ms=300000 \
          --override max.connections.per.ip=2147483647 \
          --override num.partitions=1 \
          --override producer.purgatory.purge.interval.requests=1000 \
          --override replica.fetch.backoff.ms=1000 \
          --override replica.fetch.max.bytes=1048576 \
          --override replica.fetch.response.max.bytes=10485760 \
          --override reserved.broker.max.id=1000 "
        env:
        - name: KAFKA_HEAP_OPTS
          value : "-Xmx512M -Xms512M"
        - name: KAFKA_OPTS
          value: "-Dlogging.level=INFO"
        volumeMounts:
        - name: datadir
          mountPath: /var/lib/kafka
        readinessProbe:
          initialDelaySeconds: 15
          periodSeconds: 5
          timeoutSeconds: 2
          failureThreshold: 5
          exec:
           command:
            - sh
            - -c
            - "/opt/kafka/bin/kafka-broker-api-versions.sh --bootstrap-server=localhost:9093"
      securityContext:
        runAsUser: 1000
        fsGroup: 1000
  volumeClaimTemplates:
```

```
  - metadata:
      name: datadir
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: kafkasc
      resources:
        requests:
          storage: {{ .Values.resources.kafkastorage }}
```

Yaml Script Apache Spark:

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: tpcds-benchmark-sql-0100g
  namespace: default
spec:
  type: Scala
  mode: cluster
  image: seedjeffwan/spark:v2.4.5-examples
  sparkVersion: 2.4.5
  mainClass: com.amazonaws.eks.tpcds.BenchmarkSQL
  mainApplicationFile: local:///opt/spark/examples/jars/eks-spark-benchmark-assembly-1.0.jar
  arguments:
  # TPC-DS data localtion
    - "s3a://sparktest/TPCDS-TEST-3G"
  # results location
    - "s3a://sparktest/TPCDS-TEST-3G-RESULT"
  # Path to kit in the docker image
    - "/opt/tpcds-kit/tools"
  # Data Format
    - "parquet"
  # Scale factor (in GB)
    - "100"
  # Number of iterations
    - "1"
  # Optimize queries
    - "false"
  # Filter queries, will run all if empty - "q70-v2.4,q82-v2.4,q64-v2.4"
    - ""
  # Logging set to WARN
    - "true"
  deps:
    jars:
      -
"https://oss.sonatype.org/service/local/repositories/snapshots/content/co/datamechanics/delight_2.12/latest-
SNAPSHOT/delight_2.12-latest-SNAPSHOT.jar"
  sparkConf:
    "spark.speculation": "false"
    "spark.network.timeout": "2400"
    # TPCDs Specific
    "spark.sql.broadcastTimeout": "7200"
    "spark.sql.crossJoin.enabled": "true"
    "spark.sql.parquet.mergeSchema": "false"
    "spark.sql.parquet.filterPushdown": "true"
    # S3 credential
#    "spark.hadoop.fs.s3a.aws.credentials.provider": "com.amazonaws.auth.InstanceProfileCredentialsProvider"
    # S3 Specific config
    "spark.hadoop.fs.s3a.endpoint": "http://172.30.203.31:9003"
```

```
    "spark.hadoop.fs.s3a.access.key": "AKIAIOSFODNN7EXAMPLE"
    "spark.hadoop.fs.s3a.secret.key": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
    "spark.hadoop.fs.s3a.connection.timeout": "1200000"
    "spark.hadoop.fs.s3a.path.style.access": "true"
    "spark.hadoop.fs.s3a.connection.maximum": "200"
    "spark.hadoop.fs.s3a.fast.upload": "true"
    "spark.hadoop.fs.s3a.readahead.range": "256K"
    "spark.hadoop.fs.s3a.input.fadvise": "random"
    # S3A Committer
    "spark.hadoop.mapreduce.outputcommitter.factory.scheme.s3a":
"org.apache.hadoop.fs.s3a.commit.S3ACommitterFactory"
    # "spark.sql.sources.commitProtocolClass": "org.apache.spark.internal.io.cloud.PathOutputCommitProtocol"
    # "spark.sql.parquet.output.committer.class":
"org.apache.hadoop.mapreduce.lib.output.BindingPathOutputCommitter"
    "spark.hadoop.fs.s3a.committer.name": "directory"
    "spark.hadoop.fs.s3a.committer.staging.conflict-mode": "append"
    "spark.delight.accessToken.secret": "1dbb7fe91cfd2d912359d359cd4a7d89f9f937fe767f02d5aa6b8579c0af88f3"
    "spark.extraListeners": "co.datamechanics.delight.DelightListener"
  driver:
    cores: 2
    coreLimit: "2048m"
    memory: "8000m"
    serviceAccount: default
  executor:
    instances: 10
    cores: 1
    memory: "8000m"
    memoryOverhead: "2g"
  restartPolicy:
    type: Never
```

Storage Class Script:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: test-sc
  namespace: default
provisioner: com.mapr.csi-kdf
allowVolumeExpansion: true
reclaimPolicy: Delete
parameters:
    csiProvisionerSecretName: "mapr-provisioner-secrets"
    csiProvisionerSecretNamespace: "default"
    csiNodePublishSecretName: "mapr-ticket-secret"
    csiNodePublishSecretNamespace: "default"
    restServers: "172.30.203.1:8443"
    cldbHosts: "172.30.203.1:7222,172.30.203.2:7222,172.30.203.3:7222"
    cluster: "datapipeline-ecp.cluster.com"
    securityType: "secure"
    namePrefix: "mapr-csi-pv"
    mountPrefix: "/ecp-test"
    advisoryquota: "100M"
    trackMemory: "false"
    logLevel: "error"
    retainLogs: "false"
    replication: "1"
```

RESOURCES AND ADDITIONAL LINKS

HPE Reference Architectures, hpe.com/info/ra

HPE Servers, hpe.com/servers

HPE Storage, hpe.com/storage

HPE Networking, hpe.com/networking

HPE GreenLake Advisory and Professional Services, https://www.hpe.com/us/en/services/consulting.html

HPE ProLiant 325 Gen10 Plus v2 server, https://www.hpe.com/psnow/doc/a50002564enw

HPE ProLiant 385 Gen10 Plus v2 server, https://www.hpe.com/psnow/doc/a50002560enw

HPE GreenLake Advisory and Professional Services, https://www.hpe.com/us/en/services/consulting.html

HPE Ezmeral Data Fabric, https://www.hpe.com/in/en/software/data-fabric.html

HPE Ezmeral Runtime Enterprise, https://www.hpe.com/us/en/software/ezmeral-runtime.html

To help us improve our documents, please provide feedback at hpe.com/contact/feedback.

a50007282enw, January 2025

Hewlett Packard Enterprise