White Paper

Reliable and Scalable Apache Kafka® Integrations With Apache Kafka® Connect

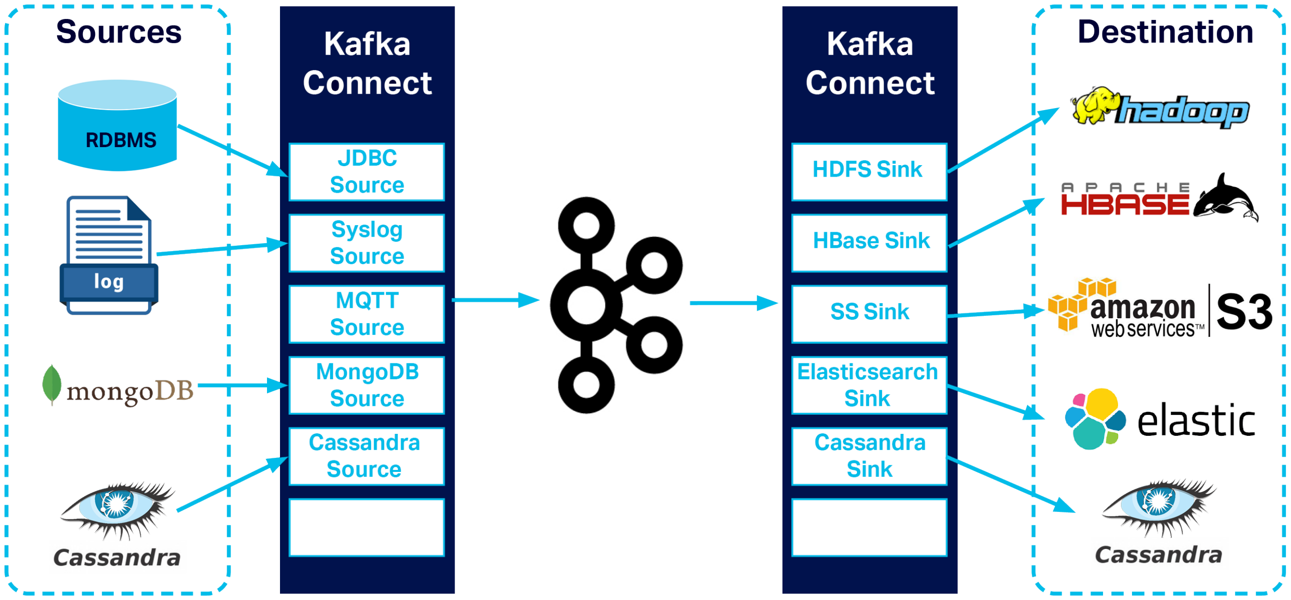
Overview

Apache Kafka® Connect provides a robust enterprise grade integration platform that enables teams to build data pipelines around Kafka by connecting other data systems with it.

Kafka Connect is built with similar design principles as that of Kafka and is inherently scalable and reliable.

instaclustr
Now part of **Spot by NetApp**

A Kafka Connector is a compiled Java program (jar file) to read and write data from a data system to and from a Kafka topic. For instance, a Kafka Connector can be implemented to read data from a table in Apache Cassandra® and compose a Kafka message for each row of table data, and write them to a pre-configured Kafka topic. This has many advantages over manually writing the code to process data from both systems. A developer simply configures a Connector to read and write data from a Kafka topic to another data system. Not only does Kafka Connect lower the barrier to performing ETL type operations, it also allows you to robustly deploy data pipelines integrating Kafka with previously built and well-maintained vendor provided Connectors.



Currently, there are dozens of open source and proprietary Connectors available that allow you to rapidly deploy low-code, config-based Kafka integrations with configurable data transformations from Kafka to and from other systems.

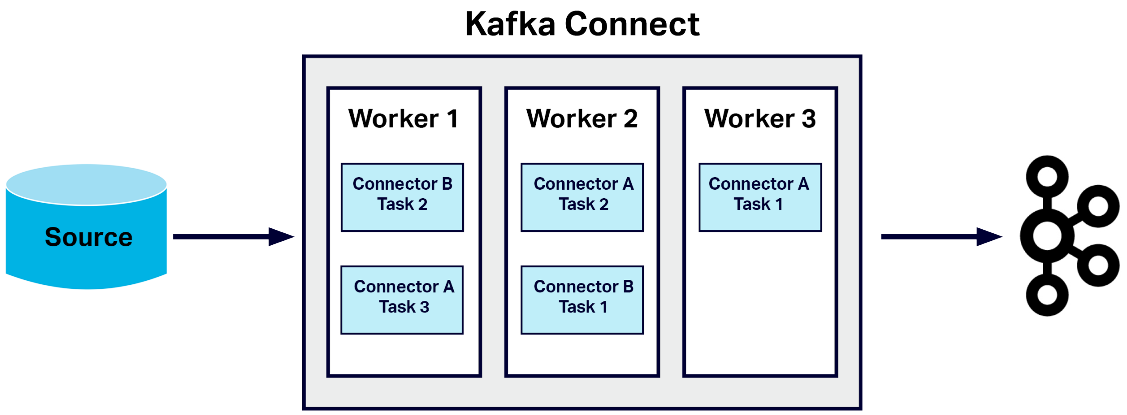
Key Kafka Connect Concepts

Here are a few core concepts to understand when working with Kafka Connect:

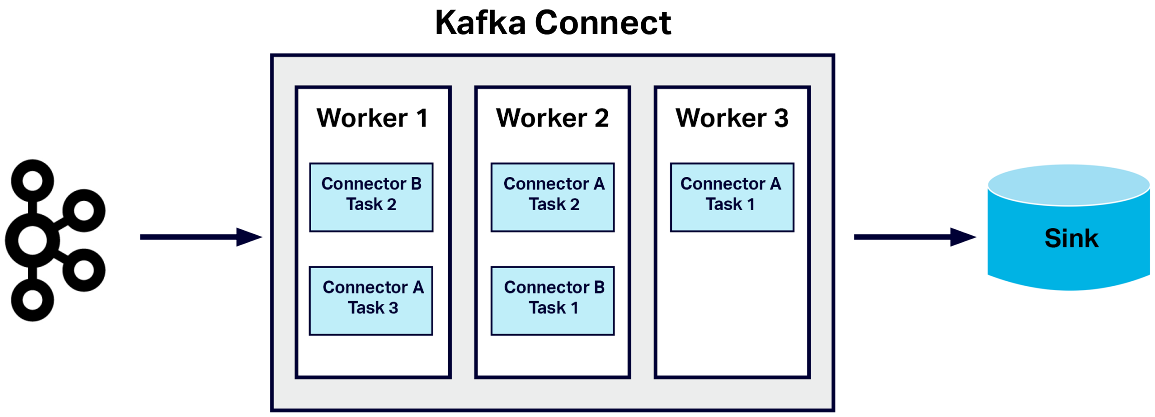
Connector

An object that defines the data import/export tasks needed to integrate a data source or sink with your Kafka cluster.

- **Source Connector:** Creates tasks to read from a data source and write to Kafka.



- **Sink Connector:** Creates tasks to read from Kafka and write to a data source.

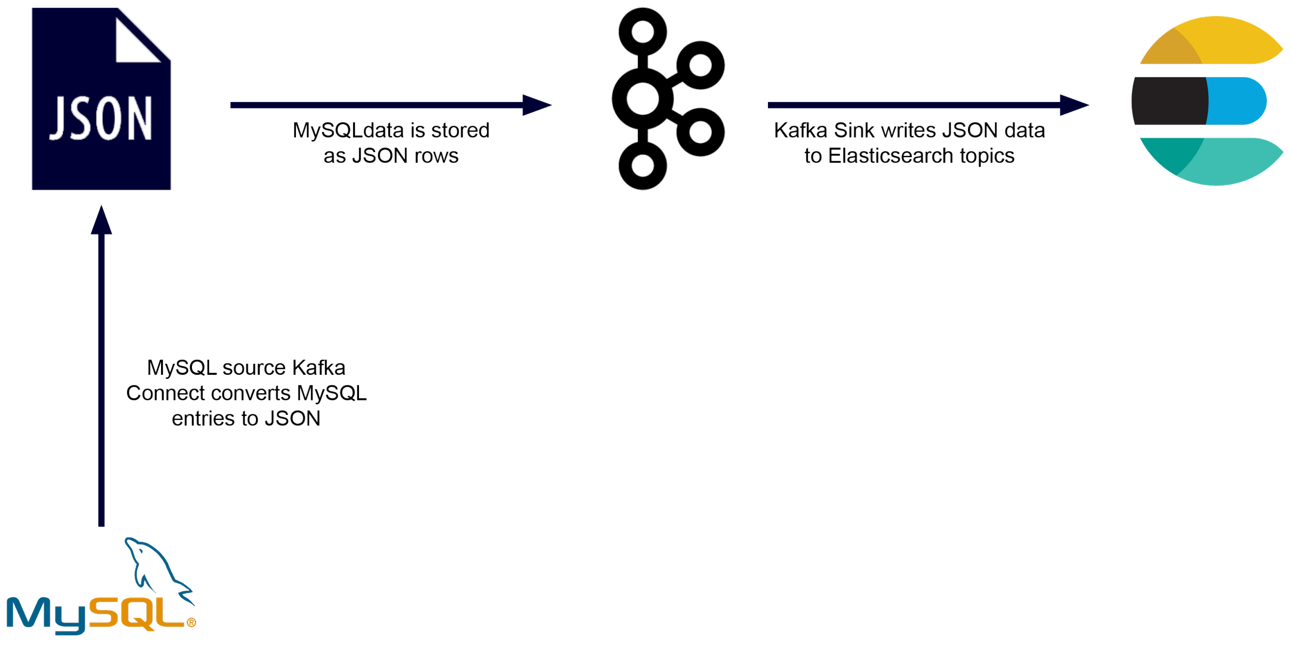


Kafka Connect Worker

A Java-based OS process that executes tasks as directed by Connectors.

Tasks

The “work” of moving data between Kafka and external data sources and sinks.


JSON
MySQLdata is stored as JSON rows
Kafka Sink writes JSON data to Elasticsearch topics
MySQL source Kafka Connect converts MySQL entries to JSON
MySQL

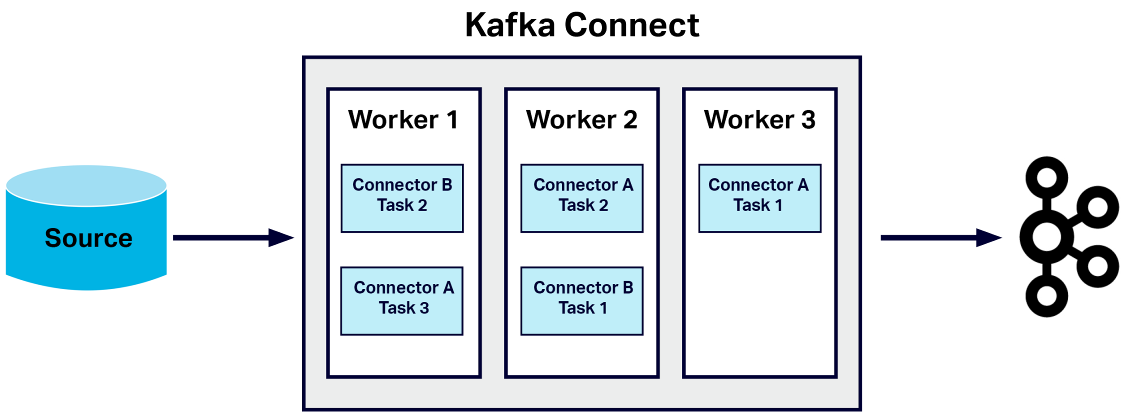
A Kafka Connect workflow using MySQL as a source and Elasticsearch as a sink with a JSON conversion in between.

Converters

Implements logic that converts data format between Kafka Connect and the connecting data transfer system. See example conversion diagram above. Conversion activities are configured for each Connector depending on how the developer wrote the Connectors. Many ETL tasks are supported from field selection to data type conversions.

Key Kafka Connect Features

Kafka Connect brings standardization and simplicity to Kafka integrations with other data systems. Kafka Connectors works in a similar way to consumers in a consumer group. Depending on the number of partitions available in a given topic, a Kafka Connector can perform ETL type operations in parallel when running in distributed mode. Kafka Connect can be thought of as an enterprise grade producer and consumer application for ETL type jobs. These ETL type operations are performed by configuring a Connector using JSON and a powerful administrative REST API.



The Kafka Connect administrative REST API can be used to deploy Connectors to a Kafka Connect cluster, and for ongoing orchestration. Developers also benefit from Kafka Connect's automated offset management, which handles the offset commit process with minimal input from Connectors, simplifying and expediting development.

Kafka Connect leverages Kafka's group management protocol to provide a distributed and highly scalable architecture whereby developers can simply add workers to expand capacity whenever needed. It also features a standalone mode appropriate for testing and down scale deployments.

Sticky Partitions

Kafka Connect workers will generally be assigned the same partitions, which prevents a lot of issues developers face during a consumer group rebalance event.

Automated Recovery Following a Failure

Source Connectors add source location data to the records they send to Kafka Connect. If a failure occurs, Kafka Connect automatically sends that data back to the Connector, enabling it to continue where it left off automatically. This automatic recovery is usually even simpler for Connectors on the sink end of Kafka Connect.

Automated Failover

In the same way a consumer group operates, if a task fails or exits, the workload will be distributed to the other workers in the Connector task group. You have the option to explicitly assign partitions to prevent this behavior if required.

Simple Parallelism

Connectors can spin up task workers equal to the number of partitions in a topic to allow parallel processing. ***Tasks*** are essentially consumers in a consumer group but also perform some producer operations.

Kafka Connect Use Cases

Kafka Connect's features make it a useful tool to create data pipelines between Kafka and another data system, optionally performing ETL operations while moving data. Once your external data is written to a Kafka topic via a Kafka Source Connector, ideally you would perform your application logic in the Kafka ecosystem. Once your application logic decides the data is ready to be moved out of Kafka, you simply write that data to another topic which is then migrated automatically to another 3rd party system via a Kafka Sink Connector.

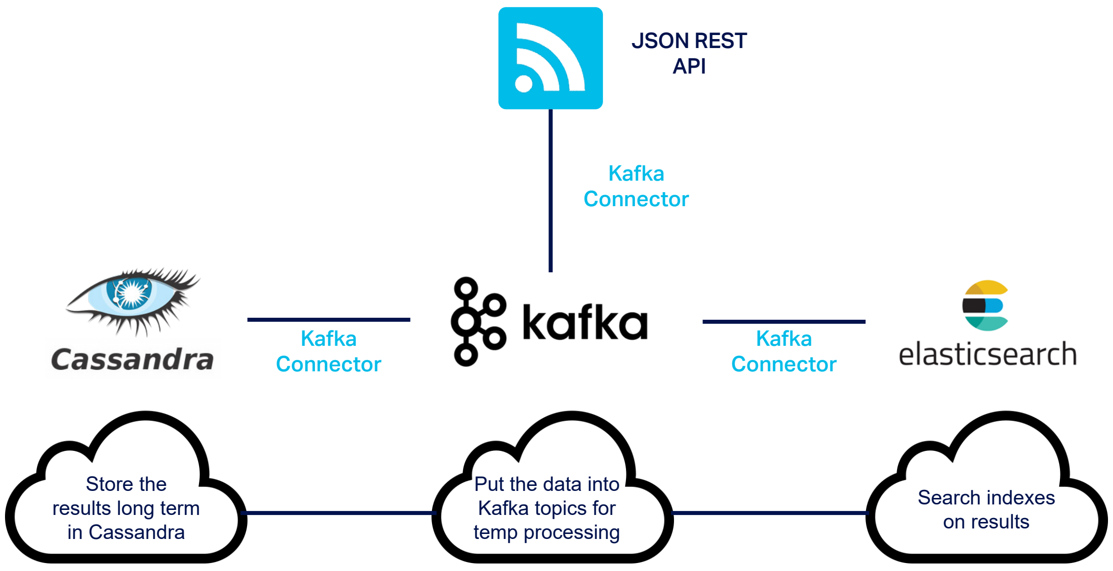
When working with bespoke or high customized applications where additional data transformation is required, there are some additional scenarios writing your own producers or consumers might be the right approach. These include:

- conditional logic;
- domain or business specific logic; or
- event driven logic.

Data Pipeline Example: Streaming

Kafka Connect can be used to disseminate data from a streaming application to a datastore, for example, Cassandra, and to a search technology like Elasticsearch after the streaming data has been processed by Kafka.

In this open source Kafka Connect demo we created, sample data is ingested by Kafka which is then written to Cassandra using a Cassandra Sink Connector and written to Elasticsearch™ using Elasticsearch Sink Connector. The logic to map each Kafka message to the format into which the data has to be written to Cassandra and Elasticsearch is implemented in the respective Connectors. The user of these connectors can simply set up some configurations and deploy the connectors to gain the value of the integration and a data pipeline.


JSON REST API
Kafka Connector
Cassandra
Kafka Connector
kafka
Kafka Connector
elasticsearch
Store the results long term in Cassandra
Put the data into Kafka topics for temp processing
Search indexes on results

While it might be reasonably simple to implement these integrations as your own producers and consumers, a large enterprise might be streaming data from several applications which has to go into 10s and 100s of data systems used by various departments, each with their own data governance. Using Kafka Connect, complex and large data pipelines can be built and deployed rapidly, minimizing time to go-live and saving initial and on-going maintenance costs and headaches.

Data Pipeline Example: Microservices

Application stacks and infrastructures built on the principles of microservices architecture can be made of 100s and 1000s of disparate and loosely-coupled services that implement the business logic for a specific responsibility. One of the prime use cases of Kafka is to act as a bridge between these services to create a streamlined data pipeline that connects them to different data systems like Apache Cassandra, Elasticsearch, AWS S3 store, Salesforce, etc. While some of these services communicate with Kafka via custom producers and consumers, almost all of the data systems typically communicate with Kafka using well known, vendor-offered connectors. There are several benefits that Kafka Connect brings to the table—reduced time to integrate, faster technology adoption, simpler and extensible solution design, technical and solution support for vendor-offered Connectors, low-code config-based integrations with low ongoing investments in maintaining it.

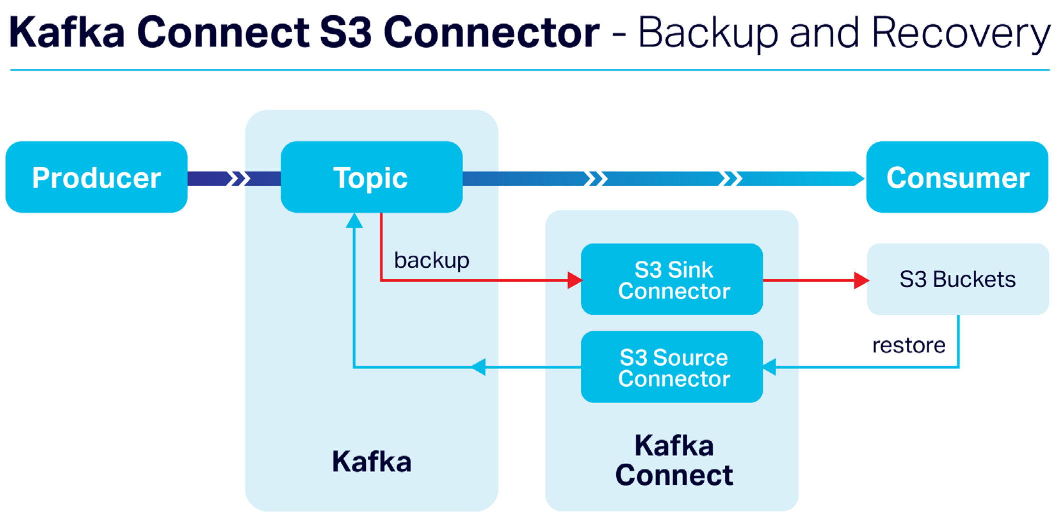
Mirroring Use Cases

Kafka MirrorMaker 2.0 is an in-built Mirroring solution packaged as part of Kafka 2.4 and is designed based on the Kafka Connect framework. The solution is implemented as a Kafka Connector which copies data from one Kafka cluster to its mirrored cluster.

Brooklin MirrorMaker by LinkedIn, Mirus by Salesforce are other examples of Kafka Mirroring solutions that are based on the Kafka Connect framework. Some of the use cases of MirrorMaker 2.0 are:

- **Geo Proximity**: Geographically distributed applications interacting with Kafka with a low-latency requirement may need data to be located close to each region where the application resides. This will require multi-region active-active mirroring which can be achieved using MirrorMaker 2.0.
- **Disaster Recovery:** Business continuity is a critical part of any enterprise solution design. Its objective is to ensure the recovery of business applications and data when there is a region-wide outage, including, potentially losing all the data residing in that region. If the Kafka cluster is mirrored to another region, the application including all producers and consumers can simply failover to the replica cluster in the other region to recover business operations.
- **Migration:** When you want to migrate an existing Kafka cluster to a new environment (a new cloud, or region, or hybrid cloud etc.), you can first deploy MirrorMaker 2.0 on the existing cluster, configure it to mirror the Kafka data to a new cluster in the desired environment, wait for the replica cluster to sync successfully, and then divert all applications to use the replica cluster.

Backup and Restore



You can backup Kafka data to AWS S3 datastore using the open source S3 connector we developed on a Kafka Connect cluster. If there is an unrecoverable issue in the Kafka cluster leading to data loss or data integrity issues, you can use the same S3 Connector, now as a source, to recover the cluster to a stable state using the S3 backup.

MirrorMaker 2.0 can also be configured to replicate unidirectionally to a replica cluster, whereby the replica serves purely as a backup of the Kafka data.

How to Launch a Kafka Connect Worker

A Kafka Connect Worker instance is a Java process that can be launched simply with a shell script. The Worker instance loads (from its Classpath) any custom Connectors named in the Connector configuration. Configuration is read from a Kafka topic in distributed mode or entered via command line in standalone mode. For users working with containerized environments, a standard Docker container image is available for launching workers. All launched instances of the image configured with the same Kafka message broker cluster and group-id will work together in a distributed manner.

Dependencies

In both standalone and distributed modes, Kafka Connect nodes must be Connected to a Kafka message broker cluster. Distributed mode has no other dependencies. Kafka Connect nodes are fully stateless as all required configurations and offset information are stored in a Kafka topic. In standalone mode, local disk storage is necessary to store Connector configurations and offset information to continue reading or writing data.

REST API

Every Kafka Connect worker instance functions as an embedded web server that presents a set of administrative REST APIs for configuration and status queries. Workers in distributed mode have their configurations uploaded via the REST API, and then stored within internal Kafka message broker topics. The REST API configurations are not needed for workers in standalone mode.

REST APIs are utilized to deploy and manage Connectors, enable tasks such as pausing and resuming Connectors, perform status checks, and more.

Kafka Connect Configuration

Kafka Connect workers can be configured through a worker configuration properties file. In distributed mode, all workers configured with the same group.id and share the same Connector configurations, offset data, and status updates (defined by config.storage.topic, offset.storage.topic and status.storage.topic, respectively) will discover one another and form a Kafka Connect cluster automatically. To create another cluster, workers must have a new set of values for each of these properties but most importantly a different group.id.

Workers in distributed mode retrieve their Connector or task configuration from a Kafka topic that is named in the worker config file. Workers in standalone mode can have their config file (pointing to the Connector to use) specified from the command line.

Common worker configurations include key.converter and value.converter properties, specifying the use of converters such the AvroConverter, JsonConverter, ByteArrayConverter, or StringConverter.

Kafka Connect Worker Connections with Kafka Message Brokers

In distributed mode, all workers make a connection with the Kafka message broker cluster. The settings involved with these connections are top-level settings in the worker configuration file, many of which are inherited from Kafka settings. Each Connector also has its own connection to the Kafka message broker cluster. To control the Kafka client settings for these connections and set them differently from the existing Kafka settings, add the prefix "producer" in the configuration for sources, or "consumer" for sinks.

Security

It's critical to ensure that Kafka Connect is configured for the same security measures that are enabled within your Kafka cluster, such as SSL or Kerberos encryption or authentication. To secure the REST API you will have to follow the well documented procedures to creating a secure Kafka Cluster, please refer to the official Apache Kafka documentation on setting up the Kafka Connect Rest API here.

Some of the Popular Open Source Connectors

Here at Instaclustr we value and support true open source solutions. The stream-reactor library has a list of popular Apache 2 licensed Connectors that you may find useful for your Kafka and Kafka connect use cases. We've built an open source S3 Connector and you can get in touch for advice on building your own Connectors.

Summary

Kafka Connect offers powerful options to combine the advantages of Kafka with other data systems, and to build data pipelines to serve your business and operational needs.

About Instaclustr

Instaclustr helps organizations deliver applications at scale through its managed platform for open source technologies such as **Apache Cassandra®**, **Apache Kafka®**, **Apache Spark™**, **Redis™**, **OpenSearch®**, **PostgreSQL®**, and **Cadence®**.

Instaclustr combines a complete data infrastructure environment with hands-on technology expertise to ensure ongoing performance and optimization. By removing the infrastructure complexity, we enable companies to focus internal development and operational resources on building cutting edge customer-facing applications at lower cost. Instaclustr customers include some of the largest and most innovative Fortune 500 companies.