Redpanda

COMPARISON

# Kafka® vs. Redpanda

## Introduction

Redpanda and Apache Kafka® are two event-streaming platforms for high-performance data pipelines, real-time analytics, and mission-critical workloads. Kafka was initially introduced at LinkedIn as a distributed message queue and was subsequently open-sourced in early 2011. Redpanda is a C++ rewrite of Kafka, backed by Redpanda Data, which aims to make real-time streaming more accessible for the great majority of developers, who are underserved by existing solutions.

There are several commonalities between Redpanda and Kafka. Both drive their vision based on community feedback and contribution through open-source or source-availalbe licensing. Both utilize distributed commit-log or immutable append-only log architectures as the basis for their data storage. Apart from that, both expose compatible APIs to the outside—Redpanda is API-compatible with Kafka.

However, while Redpanda and Kafka are trying to address the same problem in the streaming space, significant differences exist in their approaches. Runtime is one area where they diverge. Redpanda is based on a C++ runtime, optimized to squeeze the maximum resources out of its hardware, whereas Kafka is based on the JVM and depends on hands-on engineering to gain performance. These will be discussed in the performance section of this paper.

Moreover, Redpanda puts a lot of effort into increasing developer productivity. Redpanda, by design, utilizes the Raft protocol for distributed consensus. That eliminates the dependency on ZooKeeper®, simplifying the operation in production. Also, Redpanda packages capabilities that advance the state of data streaming. Onboard data transforms and intelligent tiered storage are examples. All these factors contribute to a greater developer experience in the long term.

The coming sections of this paper compare and contrast both technologies in broader detail. The sections are categorized as follows for more clarity.

1. Development experience
2. Performance and throughput
3. Total cost of ownership
4. Reliability and enterprise readiness
5. Advanced capabilities

## Development and operational experience

Redpanda's single-process deployment combined with developer-friendly experience eliminates complexity routine to Kafka environments.

### Kafka API compatibility

Redpanda is API-compatible with Kafka.

This means Redpanda is a drop-in replacement for existing Kafka-based systems, providing an easy migration path without requiring changes to existing applications. Moreover, Kafka API compatibility enables Redpanda to seamlessly integrate with the Kafka ecosystem and tooling built up over the years. For example, existing Kafka Connectors, CLI tools, and schema registries can be integrated with Redpanda.

Thinking from a developer's standpoint, switching to Redpanda means bringing their existing Kafka skills and toolsets to the trade. No up-skilling is necessary.

### Single-process deployment

Typical Kafka production deployment consists of separate systems such as ZooKeeper, schema registry to manage event schemas, and an HTTP proxy to securely expose Kafka APIs to the outside. These systems come as separate binaries, requiring separate deployment, management, and monitoring efforts.

Redpanda bundles these systems together and packages them into a single binary, making it simpler for operators to deploy and manage. For example, Redpanda has a built-in schema registry, providing tools to describe the stored events. Also, Pandaproxy is an integrated HTTP proxy that exposes Redpanda features as REST APIs.

### No ZooKeeper to manage

A production Kafka cluster relies on ZooKeeper for cluster coordination and metadata management. ZooKeeper runs as a separate cluster that must be managed by an operator. Typically, a ZooKeeper cluster requires three nodes at minimum to obtain a quorum.

ZooKeeper has been seasoned well over the past decade, delivering value as expected. However, it requires a specialized skill to deal with problems as they inevitably occur, leading to operational overhead.

Kafka has recently eliminated the dependency on ZooKeeper with its version 3.0 release. The famous KIP-500 was instrumental in getting there.

Conversely, Redpanda took a different approach. It was a design decision to leverage the open-source Raft consensus algorithm to eliminate the need for a third-party consensus system like ZooKeeper. That thoughtful decision helped Redpanda to reduce the operational complexity.

## JVM-free

Historically, many streaming technologies were centered on Java. Building, troubleshooting, and operating these systems was challenging for a typical developer without sufficient knowledge in distributed systems, Java, and JVM performance engineering.

Redpanda breaks this pattern by moving away from the JVM and opening doors to non-JVM communities to build streaming architectures in a scalable way.

## Runs anywhere

Redpanda runs easily and efficiently on many platforms, including desktops, ARM devices, IoT platforms, etc. Its single binary approach makes it especially convenient for small footprint local environments.

## Built-in observability, rpk, and operator friendliness

Redpanda packs many features to make DevOps and SRE life easier. For example, rebalancing of load, data, and leadership is automatically handled, freeing operators from manual work. Also, Redpanda provides a native K8s operator and built-in Prometheus support to speed up infrastructure provisioning and integrate with existing monitoring capabilities.

Redpanda also provides Redpanda Keeper (rpk), a developer-friendly CLI utility that helps manage Redpanda deployments on your desktop or in production.

## The best UI for managing Kafka environments

Redpanda Console gives application developers and Redpanda operators the easiest, most complete web UI for visibility into data streams and powerful features for debugging and cluster administration. Avoid resorting to a patchwork of command line tools and log files to build your own troubleshooting and administration workflows. Redpanda Console is a comprehensive UI for:

- Time travel debugging through historical messages to identify and debug problems
- Quickly sift through real-time messages to understand the "now"
- Easily view messages encoded in Avro, Protobuf, Binary, JSON, XML, and more
- Leveraging search and programmable filters to monitor millions of messages
- Visually managing Kafka ACLs, topics, schemas, and Kafka Connect
- Monitoring consumer groups, partitions, broker settings, and more

Redpanda Console features many industry-first capabilities, such as the Programmable Push Filters which allows you to quickly narrow down the range of messages you are looking for. Push filters are custom pieces of logic written in JavaScript or TypeScript that allow developers or admins to express the types of messages that are of interest and surface them in the console.

Also unique to the Redpanda Console is a comprehensive capability to manage Kafka ACLs, an ability limited heretofore to configuration files and command lines. Redpanda Console's visual interface enables admins to more easily understand security policies, and helps admins avoid configuration mistakes that could result in security breaches.

## Performance and throughput

Written in Java, Kafka takes advantage of cheap disks to store and cache its data and delivers a good performance by utilizing the file system to its fullest.

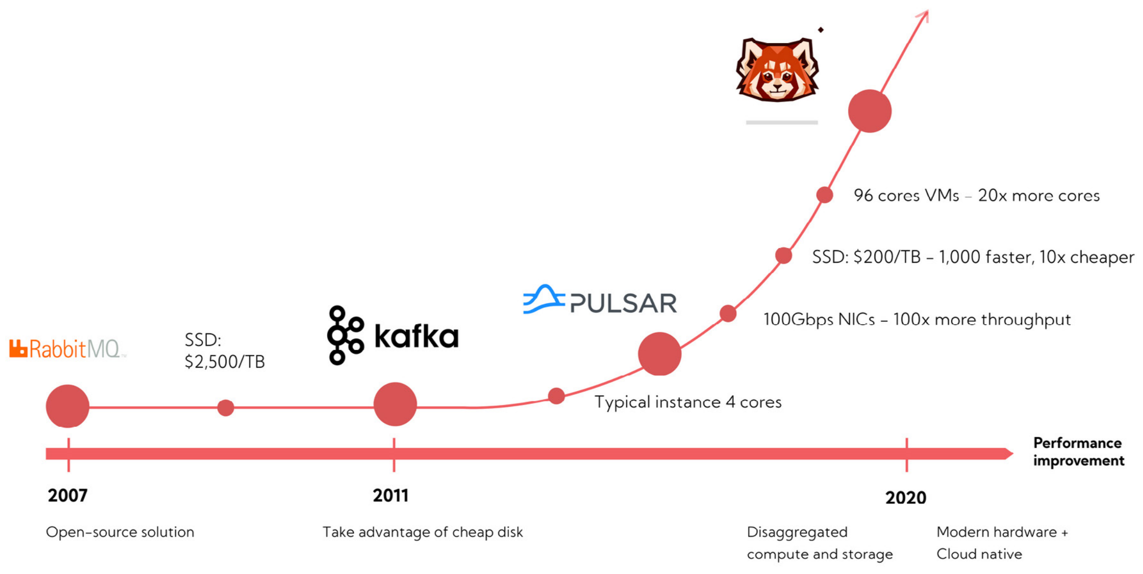
Kafka favors an approach towards the economy of cheap disks scaling horizontally. Under the hood, it leverages sequential IO and Zero Copy principles to deliver a low-latency read and write performance.

### What makes Redpanda faster?

Written in lower-level C++, Redpanda goes beyond the performance boundaries set by Kafka to deliver a stellar, predictable performance. Redpanda leverages the Seastar framework to extract the most out of modern hardware resources with intelligent manipulation of CPU, memory, disk, and network..

### Designed for modern hardware

Hardware is fast, but most do not architect software to take advantage of these hardware advances. Redpanda's thread-per-core architecture is optimized for today's multi-core hardware and squeezes out every last bit of performance, fully exploiting the resources it runs on. Redpanda also comes with intelligent auto-tuning out-of-the-box, which automatically generates optimal settings for a specific hardware/kernel/Redpanda setup.



RabbitMQ
SSD: $2,500/TB
kafka
PULSAR
Typical instance 4 cores
100Gbps NICs – 100x more throughput
SSD: $200/TB – 1,000 faster, 10x cheaper
96 cores VMs – 20x more cores
Performance improvement
2007
Open-source solution
2011
Take advantage of cheap disk
2020
Disaggregated compute and storage
Modern hardware + Cloud native

**Figure 01** - The evolution of modern hardware over time

## 100% of memory allocated upfront

Redpanda has no locks on the hot path. By design, Redpanda allocates almost all the machine memory upfront (minus some small amount for OS) and partitions the allocated memory between all CPU cores. That amortizes the cost of memory management (allocation, deallocation) and provides predictable latency.

## Bypasses Linux page cache

Redpanda uses Direct Memory Access (DMA) for all its disk IO, bypassing the page cache for reads and writes. Custom memory management techniques are used to align memory according to the layout of the filesystem, ensuring Redpanda only flushes as little data as to the disk.

Moreover, Redpanda leverages the XFS partitions to drive NVMe SSDs at their maximum throughput at all times. Redpanda favors XFS due to its sparse file system support to flush concurrent, non-overlapping pages.

## Leveraging cgroups

To run at peak performance for extended periods, Redpanda leverages cgroups to isolate its processes. This shields Redpanda from “noisy neighbors,” processes running alongside Redpanda which demand sharing resources that adversely affect performance.

## The outcome

Combining all these factors, Redpanda delivers at least 10x faster tail latencies than Apache Kafka and uses up to 3x fewer nodes to do so. A recent performance benchmarking study confirms this fact.

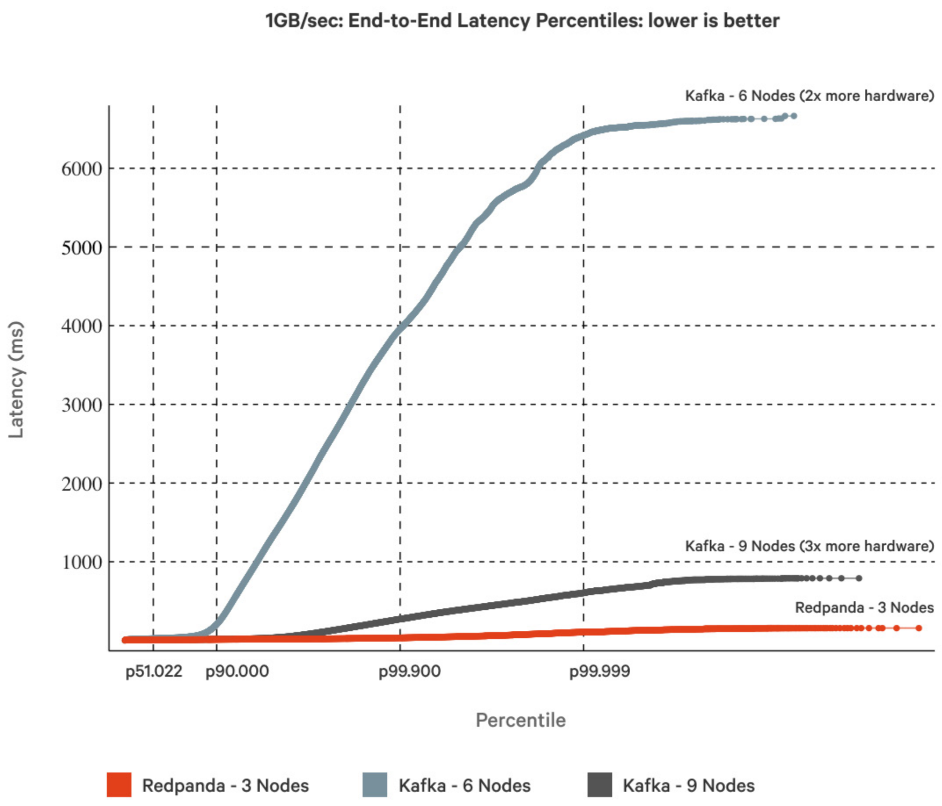


**Figure 02** - Kafka vs. Redpanda latency comparison

## Total cost of ownership (TCO)

Redpanda offers significantly lower TCO over Kafka, le ading up to 3x to 6x more cost savings compared to Kafka, reducing cloud costs, engineering time, and maintenance time.

### What makes Redpanda more cost-effective than Kafka?

Innovative features in Redpanda's architecture contribute to delivering a significantly lower TCO,  in terms of infrastructure and administration.

### Infrastructure (Compute and Storage)

Redpanda is built to fully saturate fast SSD and NVMe devices and to take advantage of multi-core and high-memory machines. Redpanda instances use hardware optimally, allowing for smaller deployments with low end-to-end latency that are consistent even at high throughputs.

Tiered storage is another Redpanda feature contributing to reduced cloud storage costs because S3 storage is significantly cheaper than SSD/NVMe-based instances. That also reduces the operational complexity of running a large Kafka cluster that is sized simply for retention.

### Administration

Redpanda users spend significantly less time monitoring and tuning a Redpanda cluster over a Kafka cluster. That's because Redpanda has a number of features that make Redpanda easier to maintain. These include things like an autotuner, leadership balancing, continuous data balancing, maintenance mode, and rolling updates.

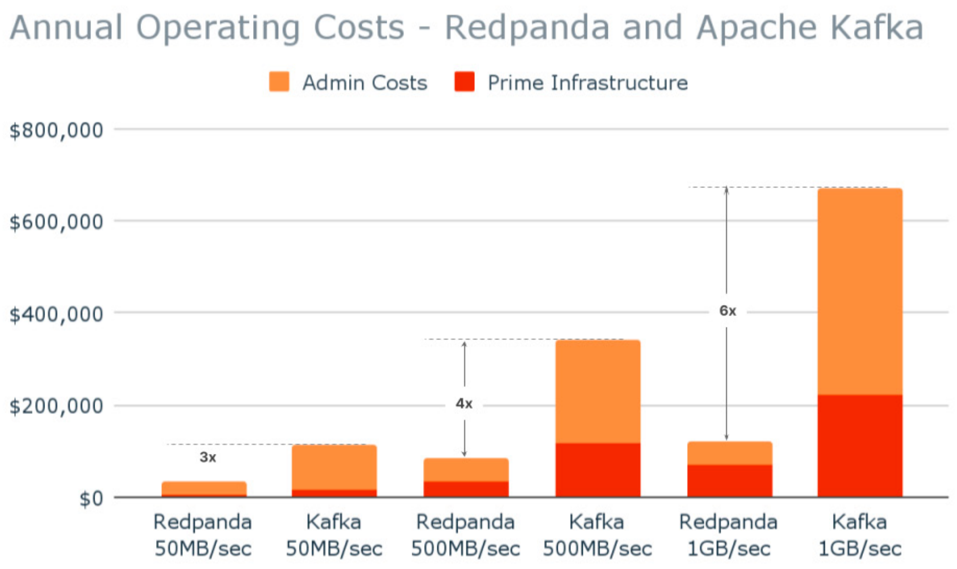


**Figure 03** - Consolidated Total Cost of Ownership comparison of Kafka and Redpanda across all workloads.

A recent TCO comparison revealed that Redpanda is up to 6x more cost-effective than Apache Kafka, and 10x faster.

## Reliability and enterprise readiness

Kafka achieves fault tolerance by replicating its partitions across multiple brokers. When Kafka receives a message, it is written to a partition and subsequently replicated to other brokers in the cluster based on the replication factor. In case of a broker failure, its partitions can be recovered from the remaining brokers.

Kafka's replication is a complicated process that is executed synchronously with the strong involvement of the ZooKeeper. While ZooKeeper handles scenarios like leader election and broker coordination, which is critical for recovering from failures, it often reduces the performance and adds management overhead.

### What makes Redpanda reliable and scalable?

Redpanda promotes a highly scalable architecture with zero data loss and predictable performance under high loads to support mission-critical systems.

### Instant horizontal scalability

Redpanda leverages the Raft protocol to scale to millions of partitions, simply starting from laptops to petabytes of data without impacting code. That enables gaining more capacity by adding more nodes to a Redpanda cluster.

### Ensures zero data loss

Redpanda is safe by default, ensuring zero data loss without compromising performance.

### Support for Kafka transactions

Redpanda's support for Kafka transactions ensures data correctness in highly concurrent scenarios.

### Ensures predictable performance at scale

Redpanda utilizes a thread-per-core model, which leads to no locking, minimal context switching, and thread-local memory access.

## Workload isolation

Redpanda's read replicas are powered by shadow indexing for fast global data and workload isolation, eliminating noisy neighbors.

## Commitment towards open-source

Redpand is source-free, with a large growing community of like-minded developers. That enables businesses to minimize the risk.

## Advanced capabilities

Redpanda is packed with capabilities that fast track the development of streaming applications, and eliminate much of the administrative complexity native to traditional Kafka deployments.

### Intelligent tiered storage

Redpanda's native tiered storage facility enables you to retain large amounts of data with a very small number of nodes. Built on Redpanda's proprietary Shadow Indexing technology, tiered storage intelligently offloads data to cloud object stores, keeping only what your consumers might need local to your cluster. This significantly reduces the cost and complexity of operating large production systems. Not only that, but tiered storage in Redpanda also gives your business services low latency access to large windows of data, unlocking a new generation of real-time analytics use cases.

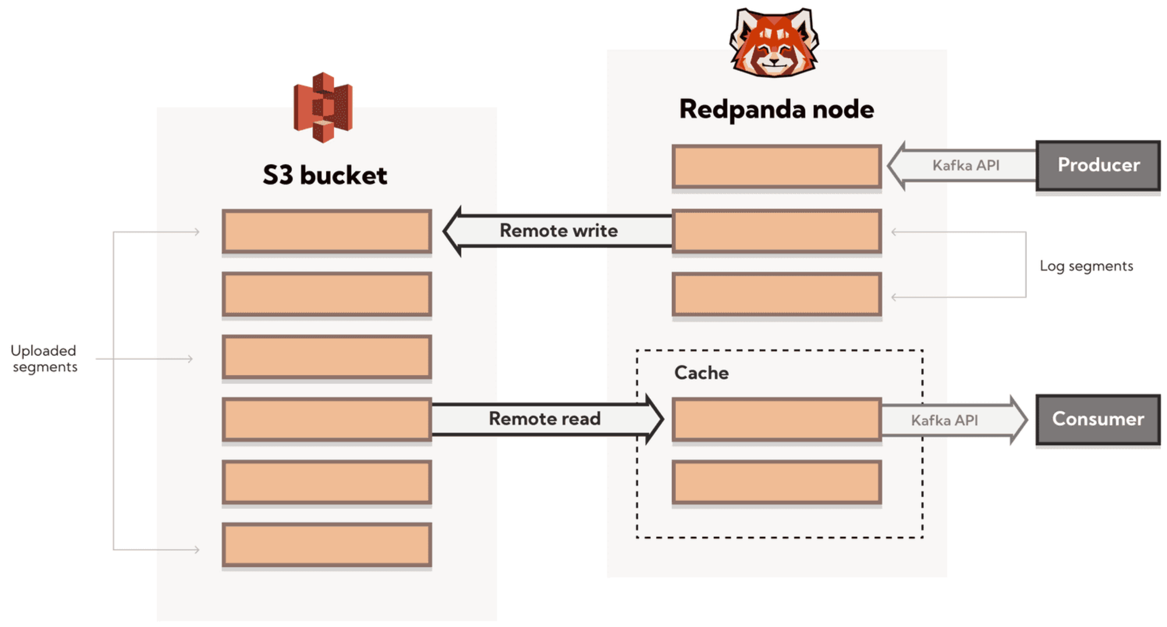


**Figure 03** - Redpanda's Shadow Indexing architecture

## Remote read replicas

Building upon the tiered storage capability, Remote Read Replicas allow users to pick topics from operational clusters and serve them from analytics clusters without duplicating data or deploying any additional software. Read replicas distribute data close to the workloads and free engineers from the pressure of picking the right data architecture from the get go. This feature can propagate new topics on-demand in minutes. Because the data resides in your cloud storage, Remote Read Replicas can be served on ephemeral hardware with minimal local storage, enabling a more cost-effective hardware footprint.

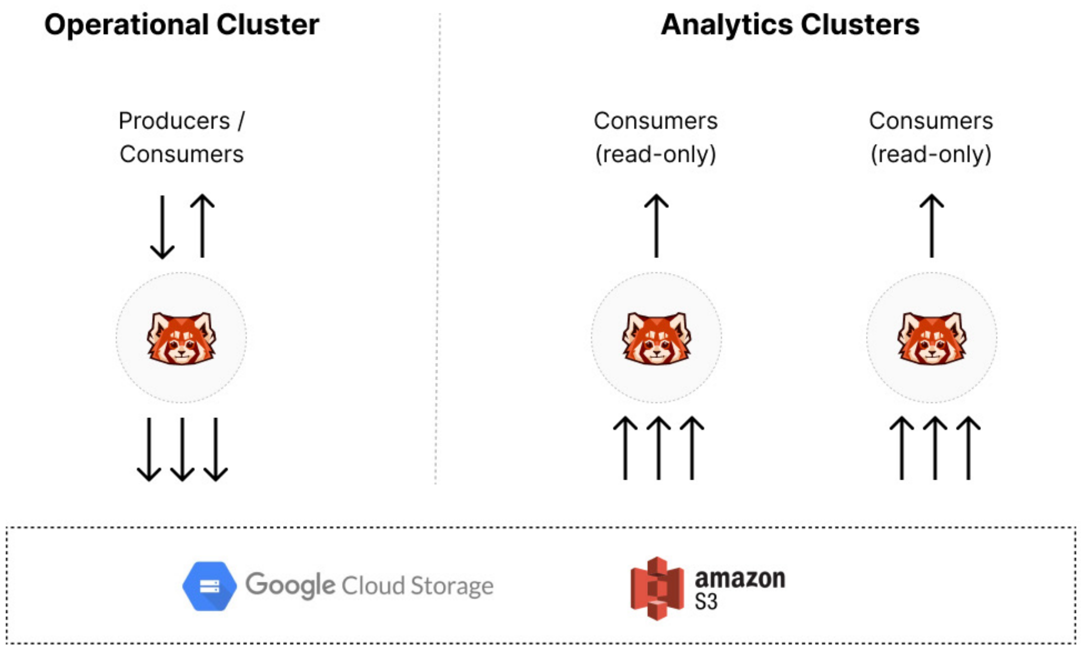


**Figure 04** - Remote Read Replica architecture

## Continuous data balancing

One of the most complex tasks in streaming data management is protecting clusters against data seasonality - where 10% of your nodes handle 90% of the traffic. It is inevitable that one of your customers will generate 80% of your traffic and fill up your disks, dominate the network, and saturate resources, resulting in performance degradation. This is the "noisy" data neighbor we all fear.

The continuous data balancing features in Redpanda actively monitor your cluster and automatically rebalance when adverse events are spotted, keeping data seasonality at bay. These features liberate your SREs and sysadmins from having to manually administer storage in your clusters. Continuous data balancing relies on multiple anti-entropy mechanisms to keep your cluster in its optimal condition. These include:

- Automatic leadership balancing: Works with the Raft quorum to ensure an even distribution of leaders across all partition-replicas.
- Data balancing on node additions: Triggers partition rebalancing when new nodes are added to the system.
- Data balancing on node failures: Moves partitions hosted by a failed node to other nodes in the cluster automatically.
- Data balancing on disk usage: Partitions are redistributed across the cluster when disk usage reaches a specified percentage of disk capacity.

## Onboard data transforms

Redpanda Data Transforms enable developers to easily create modules to perform simple data transformations on topics within Redpanda. This reduces "data ping-pong" by eliminating the need to send data out to a stream processor for common transformation tasks like data scrubbing, cleaning, normalizations, etc. The first iteration of Data Transforms offers support for Javascript-based modules that are executed in an asynchronous sidecar. Future versions will offer support for transforms in other languages as well as transforms in line with the write path.

More on the WASM-based data transformation engine architecture can be found here.

## Summary

Redpanda and Kafka may look similar on the surface, trying to solve the same set of problems in the event streaming space. But each technology has its internal differences, especially in architecture, performance, and overall developer experience. Redpanda advances the state of streaming data with its approach:

- **Simple** - Compatible with the Kafka API, Redpanda takes a big step ahead of Kafka by eliminating the development and operational complexity native to Kafka. Delivered as a single binary, free from ZooKeeper, built-in support for a schema registry, Prometheus, and an HTTP proxy. Redpanda also comes with the Redpanda Console, an easy to use web UI to simplify debugging and management of Kafka environments.
- **Fast and reliable** - Written in C++, Redpanda has been engineered for greater performance by utilizing all the resources available in server hardware. This also contributes to achieving consistent, predictable tail latencies. Besides that, Raft protocol-based replication architecture plays a vital role in achieving zero data loss.
- **Lower TCO** - Capabilities like tiered storage, Remote Read Replicas, and Continuous Data Balancing enable you to store and manage infinite streaming data, and unlock new use cases. Moreover, since Redpanda automates the bulk of these operational procedures and leverages cloud storage transparently, you save on costs for hardware, as well as hours sunk into routine administration.

To learn more about Redpanda, please visit redpanda.com or talk to our product experts at redpanda.com/contact.

## Learn more

To find out more about Redpanda, please contact us or join our community. We're glad to meet with you to show the advantages Redpanda can bring to your company.

| | |
|---|---|
| **Website:** redpanda.com | **Twitter:** @redpandadata |
| **Documentation:** docs.redpanda.com | **Contact us:** hi@redpanda.com |
| **Slack:** https://redpanda.com/slack | **Github:** github.com/redpanda-data/redpanda |