Available online www.jsaer.com

Journal of Scientific and Engineering Research, 2021, 8(3):240-248

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

The Role of Kubernetes in Automating Data Pipeline Operations: From Development to Monitoring

Chandrakanth Lekkala

Email id: Chan.lekkala@gmail.com

Abstract In the time of big data and cloud computing, organizations now depend on data processing pipelines, having several things to process, turn, and analyze much data. Such pipelines require management from their development and deployment until monitoring stages, which is often challenging. Kubernetes, the open-source container orchestration platform, the most loved technology for 2019, has gained popularity in automating the data pipelines' operations. This paper analyzes what Kubernetes provides when data distribution, development and testing are involved. Starting from deployment through scaling and processing, we handle those Kubernetes components. It includes Kubernetes specifics of core concepts, architectural patterns, integration with data processing frameworks, and the integrated toolset for logging, monitoring, and troubleshooting. This paper also digs deeper into hands-on use cases and top-notch practices that you can utilize while building your Kubernetes-driven data pipeline for ultimate scalability, reliability, and observability. Lastly, it concludes research frontiers and challenges that this is the fastest-growing technology area.

Keywords: Kubernetes, data pipelines, automation, container orders, microservices, DevOps, monitoring & logging, and distributed systems.

Introduction

The data pipelines have been cited as the critical part of any system that relies on the data. Pipelines of these pipelines are the channels through which the data travels in steps of differing processing until the insights and value are delivered. However, building and operating data pipelines at scale presents several challenges: However, building and operating data pipelines at scale presents several challenges:

A. Complexity

Sometimes, the data pipelines contain multiple interconnected services, and each service has its own dependency, configurations, and resource requirement. Dealing with such complexity manually will be difficult cause of occurring errors and time consumed [1].

B. Scalability

More data travels faster, and then, of course, pipelines should grow bigger to handle alternating workloads. Stress on infrastructure must be considered for peak loads, which may result in over provisioning and, thus, higher costs.

C. Reliability:

The data pipeline's resilience against failure, which leads to good data integrity, is significantly important. To improve and sustain pipeline oversight, the team must tackle failures and remain consistent throughout a project's life.

D. Observability

Despite the power of distributed data pipelines, identifying and solving emerging issues can become problematic without explicitly actionable logging, monitoring, and tracing facilities [2].

To overcome these obstacles, many services, such as Kubernetes, are utilizing these technologies of containerization and container orchestration. Kubernetes offers a configuration and mode of operation that gradually scales and manages containerized applications within a cluster pool [6]. What is impressive is that

Kubernetes Data Pipeline because of the number of tasks that can be done automatically while developers concentrate on generating business.

This work highlights how Kubernetes automates data pipeline operations across different phases. Section II covers the Kubernetes principles and key construction elements. Part III shows the pros and ways of designing Kubernetes for data pipelines. Part IV examines the Kubernetes ecosystem of logs, monitoring tools, and tumbling. Section V comprehensively discusses the technique's application and the lessons learned from it in the real world. The last section identifies the implications and prospects for further study (VI). The closing culminates this path (VII).

Overview of Kubernetes

Kubernetes, an open-source platform, orchestrates the automation cycle of containerized applications' deployment, scaling, and management [3]. Google co-created it initially, and the Cloud Native Computing Foundation (CNCF) now maintains it.

Critical Concepts in Kubernetes Include

A. Pods: The fundamental working unit for deploying in Kubernetes. Therefore, the pod, which is just as well marked with one or more containers, exists in between the closest regions where the containers share common resources like storage and networking.

B. Services: A toggle layer that is set remains the same network identity for a group of processes. Services make microservices independent, thus enhancing decoupling. If a component changes, the whole service does not break [4].

C. Deployments are a defined form of performing the task of the desired declared condition of the set of pods. Operations are responsible for rollout support, rollbacks, and scale docs if needed.

D. Stateful Sets: A custom-made controller for admin management of stateful applications like databases that have stable identities and require persistent storage, equivalent to regular driver software [5].

E. Config Maps and Secrets: Objects for spearheading separation of configurations from application codes and anspr binding sensitive data [6].

F. Namespaces: A method that can slice a cluster into individual sections for multiple clients and resource isolation [7].

Kubernetes Architecture

As Figure 1 shows, Kubernetes follows a master-worker (a.k.a. cluster) architecture. The Head Node coordinates the cluster's state and allocates tasks to suitable workers. It provides APIs for interaction with the cluster (e.g., to access the storage) to its users and clients. Worker nodes, as the minions are called, are where pods with the actual workloads run, and pods are nothing but containers that enable a faster startup and more accessible management in a container orchestration environment. They use proxies called kubelets and a messaging system [3].

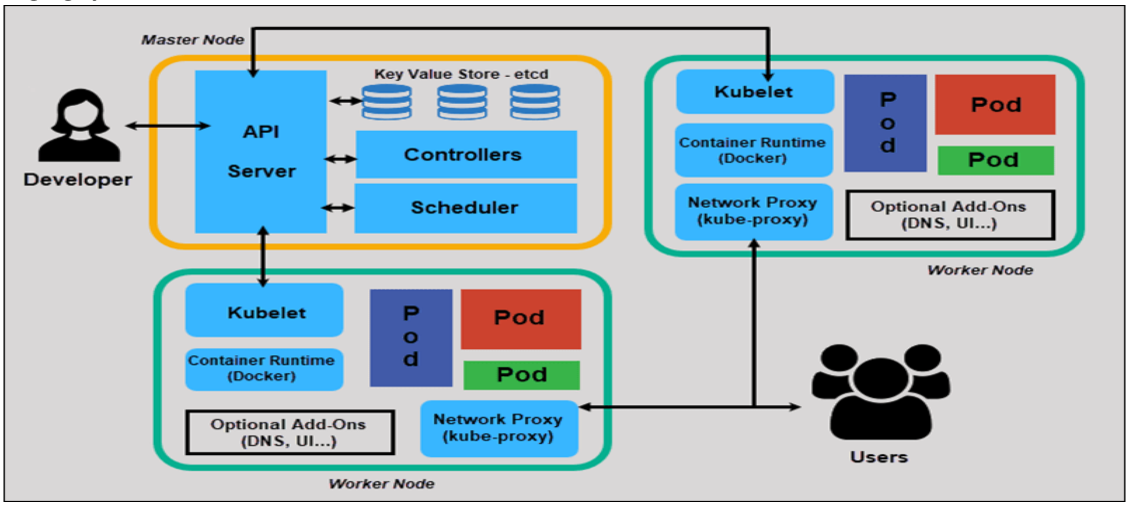
Master Node
Key Value Store - etcd
API Server
Controllers
Scheduler
Developer
Kubelet
Container Runtime (Docker)
Network Proxy (kube-proxy)
Pod
Pod
Pod
Optional Add-Ons (DNS, UI...)
Worker Node
Kubelet
Container Runtime (Docker)
Pod
Pod
Pod
Optional Add-Ons (DNS, UI...)
Network Proxy (kube-proxy)
Worker Node
Users

Figure 1: Kubernetes Architecture [9]

It also comes with add-ons and extenders to address networking, identification, storage, and more. Popular add-ons include:

A. Kubernetes, specifically the Pod-to-Pod Network Interface (CNI) Plugins.
B. Container Storage Interface (CSI) plugins are also configured to make the storage of the containers persistent.
C. Kubernetes can realize this idea by providing a reliable and scalable ingress controller for managing external traffic routing. Prometheus and Grafana are two of the most essential parts of the whole system for monitoring and visualization.
D. Flub and Snpm living logs were analyzed.

This is horizontal scaling, resulting in load balancing on many servers so that they carry the equivalent amount of work.

Kubernetes for Data Pipelines

Modernism's advantages and patterns also apply to other areas of life. Deploying and managing data pipelines on Kubernetes offers several benefits

A. Declarative Infrastructure: Kubernetes manifests yet another great benefit; they can be used for pipeline instances coming in – they aid in the configuring pipeline instances with a desired state; they allow for version control and reproducibility because of gitOps workflows [8].
B. Scalability and Elasticity: One of the scaling features of Kubernetes, such as Horizontal Pod Autoscaler, is the auto-adjustment of the resources carving out the pipelines based on the workloads optimization, utilization, and cost [9].
C. Fault Tolerance: Kubernetes does this fail-over mechanism by restarting failed pods and parallel workloads to healthy nodes, which bodes well for pipeline reliability and reduces unplanned downtime.
D. Isolation and Resource Sharing: Pipelines with co-running multiple pipelines and namespace and resource quotas translate into isolation and fair resource allocation on a shared cluster [10].
E. Portability: Pipelines in the form of containers can run equally everywhere (on premise, cloud/ edge), and vendor lock-in can be avoided [11].

Typical Patterns for Deploying Data Pipelines on Kubernetes Include

A. Microservices Architecture: Loosely coupling pipelines into granular components [data ingestion, transformation, serving], which are built as separate containers/deployments [12].
B. Batch Processing: Kubernetes Jobs and CronJobs are used for bounded and sequential processing tasks that require repetition (for example, ETL and model training) [12].
C. Stream Processing: Getting stateful, real-time processing engines like Spark Streaming, Flink, or Kafka Streams with Stateful Sets for their stateful feature [13].
D. Workflow Orchestration: An example intelligent pipeline tool is a workflow engine such as Argo or Airflow, which can be designed as Kubernetes Custom Resources (CRDs) to define and orchestrate complex, multi-stage pipelines [14].
E. Serverless Functions: Install and deploy pipelines as server monitors by using Knative or Kubeless frameworks for event-driven sending, for example [15].

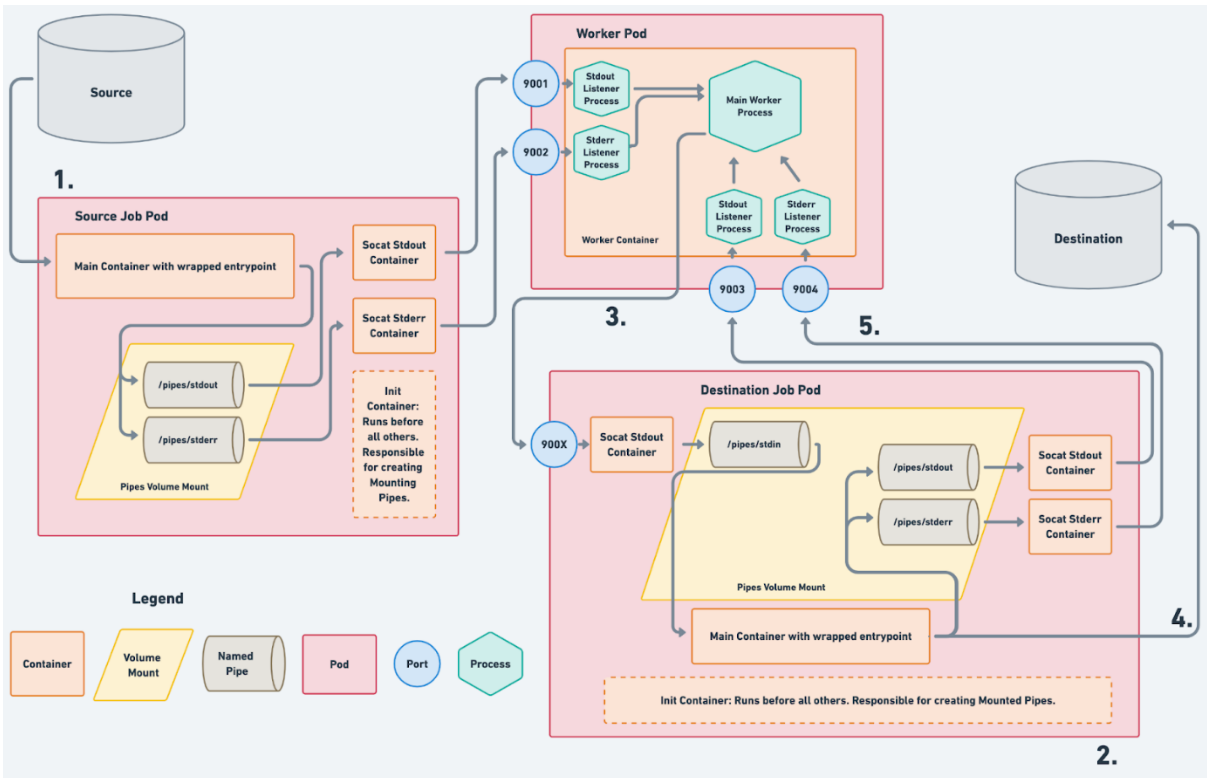

Figure 2: Common data pipeline patterns on Kubernetes [16]

In addition to these patterns, Kubernetes also provides a rich ecosystem of tools and frameworks for building data pipelines, such as: In addition to these patterns, Kubernetes also provides a rich ecosystem of tools and frameworks for building data pipelines, such as:

[1]. Apache Spark: Single checkpoints for the algorithm of large-scale data processing. Kubernetes Spark allows running Spark jobs orchestrators on Kubernetes clusters [17].

[2]. Apache Airflow: The infrastructure layers feature a program for automation workflow authoring, programming, scheduling, and monitoring. The official Helm chart is airflow executive [18]

[3]. Kubeflow: The machine learning tool kit for Kubernetes, along with notebooks, pipelines, and hyper parameter tuning, is there [19]

Pachyderm is a data versioning and pipeline platform built on Kubernetes and optimized for data science workloads and ML, enabling smooth workflow coordination [20].

Kubernetes Ecosystem for Observability

While observability might imply many things, it is vital for getting insight into and debugging data flow operations in a real-time environment. Kubernetes is a pluggable architecture that, on the one hand, unites login, monitoring, and tracing with open-source tools, on the other hand.

Logging

Kubernetes uses different logging schemes like node-level logging, sidecar containers and centralized logging, among others [21]. The most popular option is to log containers at a node level using a Fluent or File beat agent and forward logs into the central repository, e.g., Elastic search, as illustrated in Figure 3.

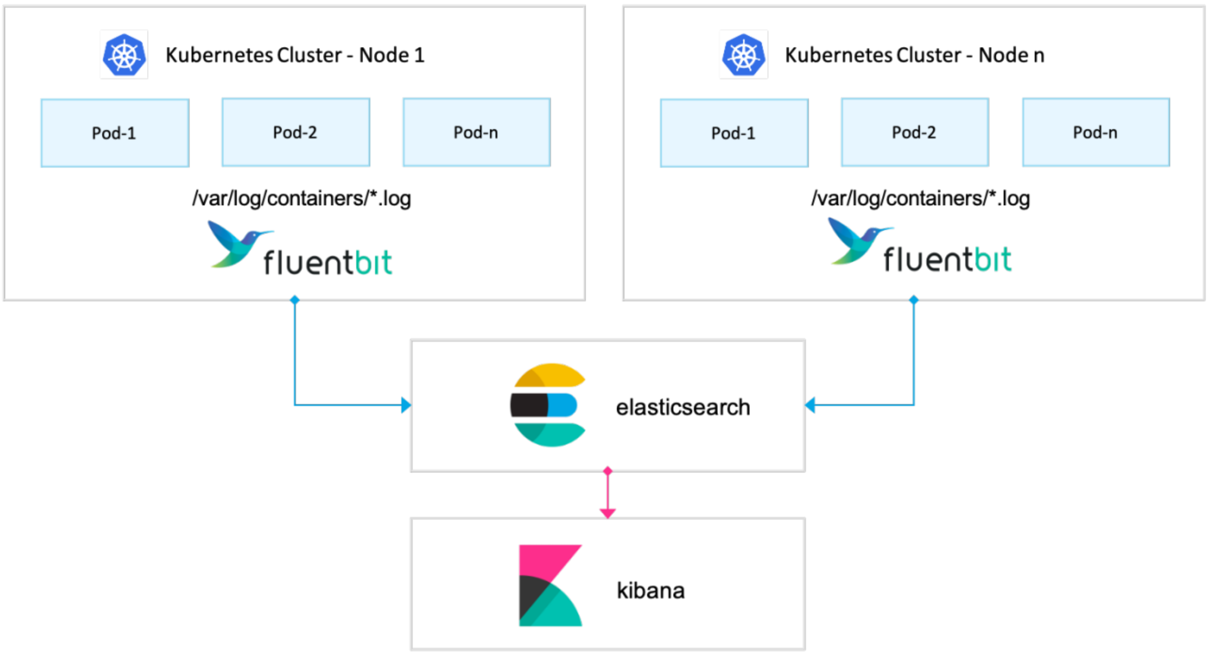
Kubernetes Cluster - Node 1
Pod-1
Pod-2
Pod-n
/var/log/containers/*.log
fluentbit
Kubernetes Cluster - Node n
Pod-1
Pod-2
Pod-n
/var/log/containers/*.log
fluentbit
elasticsearch
kibana

Figure 3: Kubernetes logging architecture with Fluentd and Elasticsearch [22]

Key features of a kubernetes logging setup include

A. Logs collection automatically by all minions and container kernels
B. Queries log processing: filter, parsing and enrichment (e.g. context meta-information addition).
C. Scalable log storage and efficient operation are necessary for massive volumes of data.
D. Searching and visualization tools like (Kibana and Grafana) are an effective data visualization method.
E. Policies on log retention or archiving.

Some best practices for logging in kubernetes include:

A. Use structured logging formats such as JSON, which significantly simplify the log parsing and analysis.
B. Use one naming convention consistently in the constructed log entries and labels.
C. Properly select verbosity levels and hide the logs from sensitive data.
D. Define the tool's settings and the retention policies to avoid storing unnecessary information and control the logging costs.

Monitoring

Monitoring Kubernetes-based data pipelines involves collecting, aggregating, and visualizing metrics from various sources such as:

[1]. Measurement of Infrastructure (e.g., CPU, memory and disk usage).
[2]. Kubernetes controls the server and the control plane metrics through its API server.
[3]. Metrics designed for using and monitoring pipelines specific to the application (e.g., throughput, ease level, and errors).

Along with Prometheus, the Kubernetes community widely utilizes the project that graduated from CNCF. Metrics will be derived from the different targets using a pull demand model, which will involve scraping at regular intervals [23].

Key components of a prometheus setup include:

[1]. Prometheus is a platform to scrape and store metrics.

[2]. The requirements are written to export the metrics from various places (for example, node exporter, kube-state-metrics).

[3]. A dashboard with an alert manager for handling alerts and notifications.

[4]. Using Grafana to design dashboards is thus very important for data visualization.

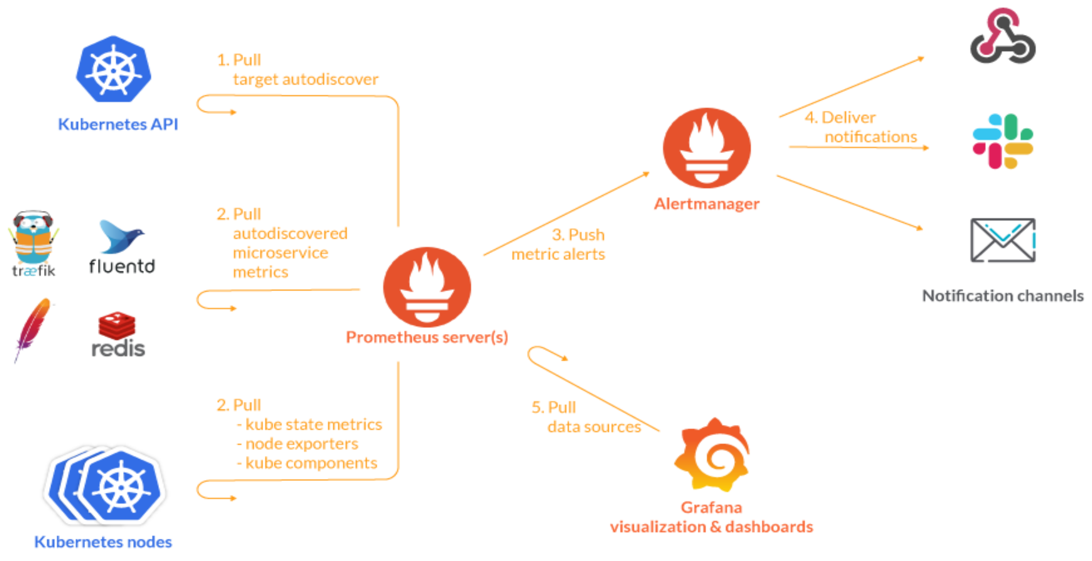
Kubernetes API
1. Pull target autodiscover
træfik
fluentd
redis
2. Pull autodiscovered microservice metrics
Prometheus server(s)
3. Push metric alerts
Alertmanager
4. Deliver notifications
Notification channels
2. Pull - kube state metrics - node exporters - kube components
Kubernetes nodes
5. Pull data sources
Grafana visualization & dashboards

Figure 4: Kubernetes monitoring architecture with Prometheus and Grafana [24]

Some best practices for monitoring data pipelines on kubernetes include.

[1]. List out the parameters (SLIs) and lenses (SLOs) for the pipeline analysis to ensure continuity and performance on that line.

[2]. Label and annotate Kubernetes resources using their metadata to enrich data with metadata.

[3]. Set up alerts of failures and degradations of critical pipeline performances as a priority.

[4]. Develop dashboards for diverse audiences (e.g. programmers, deployers and business decision-makers)

[5]. It is essential to keep looking at and changing the designs of the configurations based on both the pattern of usage and costs.

Tracing

Distributed tracing gives visibility to the requested journey across a data pipeline implemented using a micro services approach and helps locate bottlenecks and errors. Kubernetes does not provide built-in tracing capabilities but integrates well with open-source tracing solutions like Kubernetes does not provide built-in tracing capabilities, but integrates well with open-source tracing solutions like:

[1]. Jaeger: A rolled CNCF project that outlines the end-to-end distributed tracing. Jaeger utilizes the OpenTracing API, which is used together with different backends to store traces.

[2]. OpenTelemetry: "CorteX" is an incubating project from CNCF that provides unified services for distributed tracing, metrics collection, and managing different APIs and SDKs [25].

To allow for tracing in the data streams of Kubernetes-based applications, developers have to code-instrument their application code using standard tracing libraries established through Open Tracing or Open Telemetry. Code with instruments producing a trace span intercepted by agents and forwarded to a tracing backend for storage and analysis.

Some best practices for tracing data pipelines on kubernetes include.

A. Adopt a consistent scheme of span naming and adhere to the tagging conventions across services to benefit the users.
B. Spread the server interface across the breadth of the protocol stack and microservices by promulgating it.
C. Set sampling rates as low as clemency allows, keeping in mind the overhead and costs of tracing.
D. Combine with queries to diagnostics and watchdogs for cross checking for correlation and context.
E. Analyze trace data frequently to identify problems and determine opportunities for optimization.

Real-life applications and studies

Many organizations issuing their data pipelines can work just fine on the clusters of Kubernetes or even launch even more successfully when the cluster is operational and running Kubernetes. Some notable use cases include:

A. Spotify: Actually, Spotify uses Kubernetes to operate its data infrastructure, which comprises Hadoop, Spark, and Cassandra. They have released Snakebite - a self-serve platform where engineers can quickly deploy and run data pipelines on Kubernetes [26].
B. Airbnb: Airbnb uses Airflow on Kubernetes to schedule basic and more advanced tasks in a complex environment. It has provided Airflow with its Kubernetes resource allocation tool, enabling scalability and resource utilization [27].
C. Lyft: Lyft eventuates by having Kubernetes create its ML system, which covers dataset preparedness, features engineering, and model training and delivery. They have set up a pipeline framework named Flyte that represents a tool for formulating and launching ML functions on Kubernetes [28].
D. Gusto: Gusto effortlessly leverages PySpark code on Kubernetes clusters to carry out its scalable and secure billing data pipelines. They do OS namespaces and network policies to provide an isolated and safe environment [29].

Some common lessons learned from these and other kubernetes adoption stories include.

[1]. Getting started with a basic version that can be progressively enhanced based on the needs is recommended. Do not try to dip your toes in the sea from the very first morning.
[2]. Dedicate funds to constructing reusable modules, libraries, and tools to enhance automation and improve pipeline design and operations.
[3]. Establish a non-ambiguous proprietorship and Smart Leveling Agreements (SLAs) for shared Kubernetes resources and common infrastructure.
[4]. Monitoring is necessary to continue and enhance resource utilization and decrease costs, especially for the workloads that auto-scale.
[5]. Using proper security controls, data access, network isolation policies, and compliance factors are all important.
[6]. Build team synergy and share data knowledge among the data engineers, DevOps, and other stakeholders.

Future directions and challenges

As Kubernetes continues to evolve and mature, there are several future directions and challenges for automating data pipeline operations: As Kubernetes continues to evolve and mature, there are several future directions and challenges for automating data pipeline operations:

A. Serverless Data Processing: Serverless computing and Fun-as-a-Service (FaaS) involving offerings such as AWS Lambda and Google Cloud Functions have brought growing attention to running data pipelines as server less workloads. The full potential of serverless frameworks such as Knative and OpenFaaS is yet to be fully tapped, probably to build enterprise pipelines that are event-driven and will have low operational costs [30].
B. Machine Learning Operations (MLOps): Integrating machine learning into numerous pipelines belonging to a significant number of companies brings out unique issues related to ML lifecycle

management. Using Kubeflow, MLflow, and Sagemaker as examples, MLOps frameworks automating ML pipelines on Kubernetes exist. However, they must still be completed for other segments, such as model versioning, monitoring, and regulation [31].

C. Multi-Cloud and Hybrid Deployments: As multi-cloud and hybrid architecture become more adopted, a common problem is managing data pipelines over different, heterogeneous Kubernetes environments. Through efforts like Cross plane (cross-cluster deployment) and KubeEdge (edge deployments), the complexity of such operations may be reduced; however, we must strive for more compatibility, portability, and observability [32].

D. Data Security and Compliance: Among the many challenges that Kubernetes users face, data privacy and security standards like GDPR and CCPA have become increasingly strict. Thus, complying with the regulations for Kubernetes-based data pipelines is particularly critical. Approaches such, as encryption, disguising, and access restrictions should be implemented regularly throughout the entire pipeline production cycle.

E. Operator Burden and Complexity: Kubernetes, among many infrastructure details, makes it easier to abstract, but implementing production-grade clusters still requires expertise and effort. Although using managed Kubernetes services might lessen the load a bit, building the Kubernetes skills and processes in-house is something the organizations still find challenging and resource draining endeavor.

Conclusion

In conclusion, Kubernetes has indeed turned out to be a potent tool for automating processes throughout the build, run, and monitoring stages of data pipelines. With its declarative APIs, stability, and resilience to failures, Kubernetes offers organizations an opportunity to build more reliable, adaptable, and transparent pipelines that deal with ever-changing business trends. Regrettably, this is not enough to fully unfold the abilities of Kubernetes in this area. This calls for a cultural transition to DevOps and Site Reliability Engineering (SRE) principles and building the right skill sets, tools, and processes.

As data, volumes and velocity go higher along with such new technologies as server less computing and machine learning, we will witness Kubernetes's even more significant role in data pipeline development. Throughout the process of putting their finger on these latest developments and incorporating up-to-date best practices, alongside flexible teamwork within the Kubernetes community, data engineers and platform teams get the possibility to optimize their data streaming workflows significantly.

Reference

[1]. Cingolani, P., Sladek, R. and Blanchette, M., 2015. BigDataScript: a scripting language for data pipelines. Bioinformatics, 31(1), pp.10-16.

[2]. O'Donovan, P., Leahy, K., Bruton, K. and O'Sullivan, D.T., 2015. An industrial big data pipeline for data-driven analytics maintenance applications in large-scale smart manufacturing facilities. Journal of big data, 2, pp.1-26.

[3]. Luksa, M., 2017. Kubernetes in action. Simon and Schuster.

[4]. Iorio, M., Palesandro, A. and Risso, F., 2020. CrownLabs—a collaborative environment to deliver remote computing laboratories. IEEE Access, 8, pp.126428-126442.

[5]. Sayfan, G., 2017. Mastering kubernetes. Packt Publishing Ltd.

[6]. Baier, J., Sayfan, G. and White, J., 2019. The The Complete Kubernetes Guide: Become an expert in container management with the power of Kubernetes. Packt Publishing Ltd.

[7]. Sayfan, G., 2017. Mastering kubernetes. Packt Publishing Ltd.

[8]. Farcic, V., 2019. The DevOps 2.4 Toolkit: Continuous Deployment to Kubernetes: Continuously Deploying Applications with Jenkins to a Kubernetes Cluster. Packt Publishing Ltd.

[9]. Arundel, J. and Domingus, J., 2019. Cloud Native DevOps with Kubernetes: building, deploying, and scaling modern applications in the Cloud. O'Reilly Media.

[10]. Kubernetes, T., 2019. Kubernetes. Kubernetes. Retrieved May, 24, p.2019.

[11]. Hausenblas, M. and Schimanski, S., 2019. Programming Kubernetes: Developing cloud-native applications. O'Reilly Media.

[12]. Foy, G.M., 2018. Run the Storm: A Savage Hurricane, a Brave Crew, and the Wreck of the SS El Faro. Simon and Schuster.

[13]. Rossi, F., Cardellini, V., Presti, F.L. and Nardelli, M., 2020. Geo-distributed efficient deployment of containers with Kubernetes. Computer Communications, 159, pp.161-174.

[14]. Brown, D.M., Soto-Corominas, A., Surez, J.L. and de la Rosa, J., 2017. Overview—The social media data processing pipeline. The SAGE handbook of social media research methods, pp.125-145.

[15]. McKendrick, R., 2018. Kubernetes for Serverless Applications: Implement FaaS by effectively deploying, managing, monitoring, and orchestrating serverless applications using Kubernetes. Packt Publishing Ltd.

[16]. Reiter, E., 2007, June. An architecture for data-to-text systems. In proceedings of the eleventh European workshop on natural language generation (ENLG 07) (pp. 97-104).

[17]. Bureva, V., 2019. Index matrices as a tool for data lakehouse modelling. Annual of "Informatics" Section Union of Scientists in Bulgaria, 10, pp.81-105.

[18]. Barakhnin, V.B., Kozhemyakina, O.Y., Mukhamediev, R.I., Borzilova, Y.S. and Yakunin, K.O., 2019. The design of the structure of the software system for processing text document corpus. Бизнес-информатика, 13(4 (eng)), pp.60-72.

[19]. Hapke, H. and Nelson, C., 2020. Building machine learning pipelines. O'Reilly Media.

[20]. Gannon, D., Barga, R. and Sundaresan, N., 2017. Cloud-native applications. IEEE Cloud Computing, 4(5), pp.16-21.

[21]. Rozenberg, B., Shmelkin, R., Bozdemir, B., Ermis, O., Önen, M., Azraoui, M., Canard, S., ORA, B.V., Perez, A.P., KAU, T.P. and ORA, S.C., 2019. D4. 1–FUNCTIONAL DESIGN AND PLATFORM ARCHITECTURE.

[22]. [22] Gormley, C. and Tong, Z., 2015. Elasticsearch: the definitive guide: a distributed real-time search and analytics engine. " O'Reilly Media, Inc.".

[23]. Bastos, J. and Araújo, P., 2019. Hands-On Infrastructure Monitoring with Prometheus: Implement and scale queries, dashboards, and alerting across machines and containers. Packt Publishing Ltd.

[24]. Turnbull, J., 2018. Monitoring with Prometheus. Turnbull Press.

[25]. Mengistu, D.M., 2020. Distributed Microservice Tracing Systems: Open-source tracing implementation for distributed Microservices build in Spring framework.

[26]. Mineiro, L., 2019. Are We All on the Same Page? Let's Fix That.

[27]. Apperloo, E., 2020. Tracking provenance of change in data science pipelines (Doctoral dissertation).

[28]. Mitchell, R., Pottier, L., Jacobs, S., da Silva, R.F., Rynge, M., Vahi, K. and Deelman, E., 2019, December. Exploration of workflow management systems emerging features from users perspectives. In 2019 IEEE International Conference on Big Data (Big Data) (pp. 4537-4544). IEEE.

[29]. Ericsson, S., 2020. Cloud cost optimization: Finding unused cloud resources using machine learning and heuristics.

[30]. Eismann, S., Scheuner, J., Van Eyk, E., Schwinger, M., Grohmann, J., Herbst, N., Abad, C.L. and Iosup, A., 2020. Serverless applications: Why, when, and how?. IEEE Software, 38(1), pp.32-39.

[31]. Alla, S. and Adari, S.K., 2021. Beginning MLOps with MLFlow. Apress: New York, NY, USA.

[32]. Larsson, L., Gustafsson, H., Klein, C. and Elmroth, E., 2020, December. Decentralized kubernetes federation control plane. In 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC) (pp. 354-359). IEEE.