IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

VALTRUS INNOVATIONS LTD. and KEY PATENT INNOVATIONS LTD.,	§	Case No. 2:25-cv-
	§	**JURY TRIAL DEMANDED**
Plaintiffs,	§	
v.	§	
FEDEX CORPORATION and FEDEX OFFICE AND PRINT SERVICES, INC.,	§	
Defendants.	§	

COMPLAINT FOR PATENT INFRINGEMENT

Plaintiffs Valtrus Innovations Ltd. ("Valtrus") and Key Patent Innovations Ltd. ("KPI") (collectively, "Plaintiffs") for their Complaint against Defendants FedEx Corporation and FedEx Office and Print Services, Inc. (collectively, "FedEx" or "Defendants") for patent infringement allege as follows:

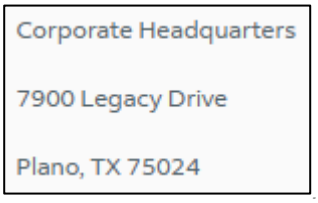
THE PARTIES

1. Valtrus is the successor-in-interest to a substantial patent portfolio created by Hewlett Packard Enterprise and its predecessor, subsidiary, and affiliate companies, including Hewlett-Packard Development Company, L.P. (collectively, "HPE"). Valtrus is an Irish entity duly organized and existing under the laws of Ireland. The address of the registered office of Valtrus is: The Glasshouses GH2, 92 Georges Street Lower, Dun Laoghaire, Dublin A96 VR66, Ireland. HPE's worldwide corporate headquarters is located in Houston, Texas. One of HPE's primary U.S. facilities is located in Plano, Texas.

2. KPI is the beneficiary of a trust pursuant to which Valtrus owns, holds, and asserts the Patents-in-Suit as defined in Paragraph 20. KPI is an Irish entity duly organized and existing under the laws of Ireland. The address of the registered office of KPI is: The Glasshouses GH2, 92 Georges Street Lower, Dun Laoghaire, Dublin A96 VR66, Ireland.

3. On information and belief, Defendant FedEx Corporation is a company organized under the laws of the State of Delaware, with a place of business at 942 South Shady Grove Road, Memphis, TN 38120. On information and belief, Defendant FedEx Corporation may be served through its registered agent for service, CT Corporation System located at 1999 Bryan Street, Suite 900, Dallas, TX 75201-3136. On information and belief, Defendant FedEx Corporation does business in Texas, directly or through intermediaries, and offers its products and/or services, including those accused herein of infringement, to customers and potential customers located in Texas, including in the Judicial District of the Eastern District of Texas.

4. On information and belief, Defendant FedEx Office and Print Services, Inc. is a company organized under the laws of the State of Delaware, with its corporate headquarters located at 7900 Legacy Drive, Plano, TX 75024.



[1]

On information and belief, Defendant FedEx Office and Print Services, Inc. may be served through its registered agent for service, CT Corporation System located at 1999 Bryan Street, Suite 900, Dallas, TX 75201-3136. On information and belief, Defendant FedEx Office and Print Services,

[1] https://www.office.fedex.com/?srsltid=AfmBOoohPlnZSz4frxMjdUUOj-iFtoomxvrQ1mtnCPeHWv5m_p3mHR0t

Inc. is a wholly-owned subsidiary of Defendant FedEx Corporation and carries out the business of its parent throughout the United States and Texas. On information and belief, Defendant FedEx Office and Print Services, Inc. does business in Texas, directly or through intermediaries, and offers its products and/or services, including those accused herein of infringement, to customers and potential customers located in Texas, including in the Judicial District of the Eastern District of Texas.

5. FedEx controls, participates in the commission of, and has a direct financial interest in the infringing acts set forth herein.

JURISDICTION AND VENUE

6. This is an action for patent infringement arising under the patent laws of the United States, 35 U.S.C. §§ 1, *et seq*. This Court has jurisdiction over this action pursuant to 28 U.S.C. §§ 1331 and 1338(a).

7. This Court has personal jurisdiction over Defendants. Defendants regularly conduct business and have committed acts of patent infringement and/or have induced acts of patent infringement by others in this Judicial District and/or have contributed to patent infringement by others in this Judicial District, the State of Texas, and elsewhere in the United States. Defendants, directly and/or through subsidiaries or intermediaries, have committed and/or continue to commit acts of infringement in this District by, among other things, making, using, importing, offering to sell, and/or selling products that infringe the Patents-in-Suit as defined in Paragraph 17.

8. This Court has general jurisdiction over Defendants due to their continuous and systematic contacts with the State of Texas, including through their operation of physical offices in the State of Texas, either directly or through authorized third parties, including this District. These locations include drop-off locations, locations with shipping assistance, locations with

IPR2026-00060 EX2010 003

package holding and/or shipping, or a combination thereof. These physical locations include at least the following: 1300 E Whaley Steet, Longview, TX 75601; 916 W Loop 281, Longview, TX 75604; 3320 Robertson Road, Tyler, TX 75701; 4522 S Broadway Avenue, Tyler, TX 75703; 1200 Capital Avenue, Plano, TX 75074; 901 N. Central Expy, Suite 200, Plano, TX 75075; 1415 E Renner Road, Suite 230, Richardson, TX 75082; and 1565 N Central Expy, Suite 100, Richardson, TX 75080. Additionally, this Court has general jurisdiction over Defendants due to their targeting of Texas residents with the https://www.fedex.com/en-us/home.html and https://www.office.fedex.com/ websites, their ownership and/or lease of land in the State of Texas, and other business activities throughout the State of Texas.

9. Defendants are subject to the specific personal jurisdiction of this Court because Plaintiffs' patent infringement claims against Defendants specifically arise from Defendants' acts of infringement in the State of Texas. These acts of infringement include directing and controlling the operation of the https://www.fedex.com/en-us/home.html and https://www.office.fedex.com/ websites, using the patented inventions, and specifically targeting residents of Texas with these and other websites to further the sale of products and services to those customers online in this District. On information and belief, Defendants operate and/or contract with third parties to operate various locations through the FedEx Authorized ShipCentre program[2], including in the State of Texas.

10. Venue is proper in this District under 28 U.S.C. §§ 1391(b)-(d) and 1400(b). Defendants are registered to do business in the State of Texas, have offices in the State of Texas, have transacted and continue to transact business in the Eastern District of Texas, and have

[2]https://www.fedex.com/en-ca/retail-locations/pickup-dropoff/joinshipcentre.html#:~:text=FASC%20Application,GoFedEx

committed and continue to commit acts of direct and indirect infringement in the Eastern District of Texas.

11. Defendants are subject to this Court's jurisdiction pursuant to due process and/or the Texas Long Arm Statute due at least to their substantial business in this State and Judicial District, including (a) at least part of their past infringing activities, (b) regularly doing or soliciting business in Texas, and/or (c) engaging in persistent conduct and/or deriving substantial revenue from goods and services provided to customers in Texas.

PATENTS-IN-SUIT

12. On February 5, 2013, the United States Patent and Trademark Office duly and legally issued U.S. Patent No. 8,370,416 (the "'416 Patent") entitled "Compatibility Enforcement in Clustered Computing Systems." A true and correct copy of the '416 Patent is attached hereto as Exhibit A.

13. On March 8, 2011, the United States Patent and Trademark Office duly and legally issued U.S. Patent No. 7,904,686 (the "'686 Patent") entitled "Data Security for Use with a File System." A true and correct copy of the '686 Patent is attached hereto as Exhibit B.

14. On December 29, 2009, the United States Patent and Trademark Office duly and legally issued U.S. Patent No. 7,640,332 (the "'332 Patent") entitled "System and Method for Hot Deployment/Redeployment in Grid Computing Environment." A true and correct copy of the '332 Patent is attached hereto as Exhibit C.

15. On October 10, 2006, the United States Patent and Trademark Office duly and legally issued U.S. Patent No. 7,120,832 (the "'832 Patent") entitled "Storage Device Performance Monitor." A true and correct copy of the '832 Patent is attached hereto as Exhibit D.

16. Plaintiffs are the sole and exclusive owner of all legal right, title, and interest in the

IPR2026-00060 EX2010 005

’416 Patent, the ’686 Patent, the ’332 Patent, and the ’832 Patent (collectively, the “Patents-in-Suit” or “Asserted Patents”), and hold the exclusive right to take all actions necessary to enforce their rights to the Patents-in-Suit, including the filing of this patent infringement lawsuit. Plaintiffs also have the right to recover all damages for past, present, and future infringement of the Patents-in-Suit and to seek injunctive relief as appropriate under the law.

FACTUAL ALLEGATIONS

17. The ’416 Patent generally relates to a system and method for a network comprising electronic devices associated with user profiles to allow migration of the user profiles from one electronic device to another. The technology described in the ’416 Patent was developed by David J. Hoover, Bernard A. Reed, III, Marjorie Krueger, and Douglas L. Voigt at Hewlett-Packard Development Company, L.P. For example, the ’416 Patent is infringed through FedEx’s use of Kubernetes for cluster creation.

18. The ’686 Patent generally relates to a system and method for providing data security for use with a file system. The technology described in the ’686 Patent was developed by Ranganath G. Iyengar at Hewlett-Packard Development Company, L.P. For example, the ’686 Patent is infringed by FedEx’s Apache Software Foundation’s Hadoop Distributed File System (“HDFS”).

19. The ’332 Patent generally relates to technology for hot deployment and/or redeployment in a grid computing environment, where the grid computing environment adds a new version of an application from a repository server, determining which grid node is running the application affected by the new version of the software, notifying the client application manager associated with one or more of the affected grid nodes, and hot deploying/redeploying the new version of the application to the affected grid nodes. The technology described in the ’332

IPR2026-00060 EX2010 006

Patent was developed by Sanjay Dahiya at Hewlett-Packard Development Company, L.P. For example, the ’332 Patent is infringed by FedEx’s use of Kubernetes for rolling updates.

20. The ’832 Patent generally relates to technology for monitoring performance of a storage device by intercepting communications between a computer system and said storage device, analyzing the intercepted communications, and reallocating at least some of said data on said storage device to enhance the performance of the storage device. The technology described in the ’832 Patent was developed by Kevin Collins and Michael P. Fleischmann at Hewlett-Packard Development Company, L.P. For example, the ’823 Patent was infringed by FedEx’s use of Apache Kafka Streams using RocksDB Universal Compaction, RocksDB Universal Compaction, Apache Cassandra, and Apache HBase.

21. FedEx has infringed the ’832 Patent by making, using, selling, offering to sell, and/or importing, and by actively inducing others to make, use, sell, offer to sell, and/or import the infringing products identified above.

22. FedEx has infringed and is continuing to infringe the ’416 Patent, the ’686 Patent, and the ’332 Patent by making, using, selling, offering to sell, and/or importing, by actively inducing others to make, use, sell, offer to sell, and/or import, and/or contributing to other’s making, using, selling, offering to sell, and/or importing of the infringing products identified above.

COUNT I
(Infringement of the ’416 Patent)

23. Paragraphs 1 through 22 are incorporated by reference as if fully set forth herein.

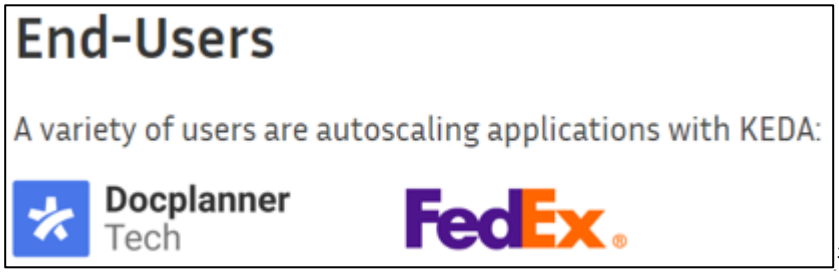
24. Plaintiffs have not licensed or otherwise authorized Defendants to make, use, offer for sale, sell, or import any products that embody the inventions of the ’416 Patent.

IPR2026-00060 EX2010 007

25. Defendants have directly infringed and continue to directly infringe the '416 Patent, either literally or under the doctrine of equivalents, without authority and in violation of 35 U.S.C. § 271, by making, using, offering to sell, selling, and/or importing into the United States products that satisfy each and every limitation of one or more claims of the '416 Patent. Such products include, but are not limited to, Kubernetes.

26. For example, Defendants have directly infringed and continue to directly infringe at least claim 1 of the '416 Patent by making, using, offering to sell, selling, and/or importing into the United States products that include Kubernetes.

27. On information and belief, FedEx has used Kubernetes in the U.S. since at least 2019.[3] On information and belief, FedEx had developers build or refactor FedEx applications on a PaaS (Platform as a Service) system. The PaaS system enables engineers to shuttle applications between Google Cloud Platform and internal datacenters by using container software to make the applications portable and the Kubernetes technology to orchestrate the movement of containers between clouds.[4]



[5]

[3] https://www.cio.com/article/217625/fedex-cloud-dojo-accelerates-digital-transformation.html
[4] https://www.cio.com/article/217625/fedex-cloud-dojo-accelerates-digital-transformation.html. See also, e.g., https://clavent.com/top-devops-trends-that-will-matter-the-most-in-2020/ (discussing FedEx's Cloud Dojo). See also, e.g., https://www.linkedin.com/in/monikamevenkamp/; (showing use of Kubernetes); https://www.linkedin.com/in/asaf-peleg-007/ (showing use of both Kubernetes and Kafka)
[5] https://keda.sh/community/#end-users

Kubernetes Workloads Autoscaling with KEDA

KEDA (Kubernetes Event Driven Autoscaling) is an Open Source tool with the backing of large companies (e.g. Microsoft, RedHat, IBM, etc.), and it is also a CNCF (Cloud Native Computing Foundation) graduate project. It is used by a large variety of companies, big and small: Fedex, Reddit, Xbox, Cisco, etc.

***KEDA** is a Kubernetes-based Event Driven Autoscaler. With KEDA, you can drive the scaling of any container in Kubernetes based on the number of events needing to be processed.*

***KEDA** is a single-purpose and lightweight component that can be added into any Kubernetes cluster. KEDA works alongside standard Kubernetes components like the Horizontal Pod Autoscaler and can extend functionality without overwriting or duplication. With KEDA you can explicitly map the apps you want to use event-driven scale, with other apps continuing to function. This makes KEDA a flexible and safe option to run alongside any number of other Kubernetes applications or frameworks.* [6]

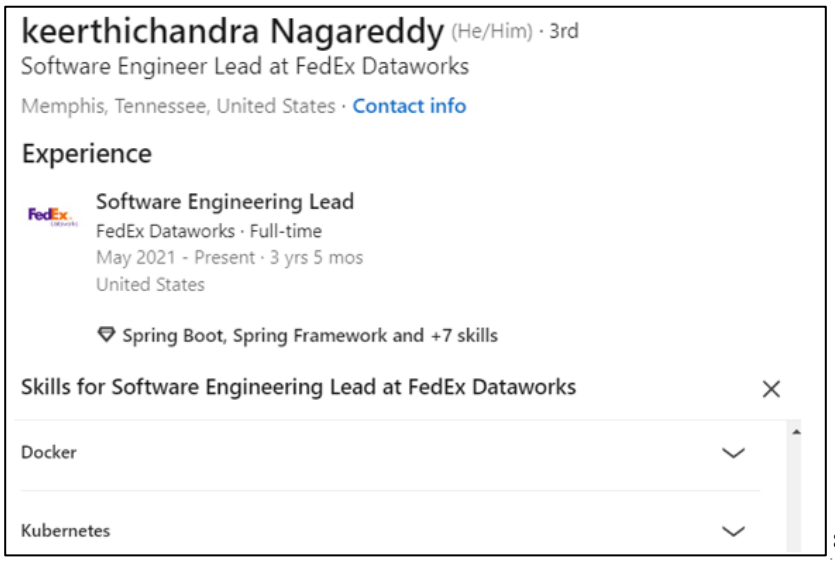
Delivering at the Nexus of the Digital & Physical Worlds: VMware & FedEx

"**At FedEx, we stand at the nexus of the digital and physical worlds,**" said FedEx CIO Rob Carter as he shared FedEx's success story of using VMware services to quickly and efficiently scale to meet the intense demands of delivering the massive rollout of the life-saving Coronavirus vaccines as well as the surge in e-commerce deliveries due to stay-at-home requirements.

[6] https://igboie.medium.com/kubernetes-workloads-autoscaling-with-keda-ccba62b64552

FedEx Senior Enterprise Architect, Matt Lum, called VMware Tanzu "critical" to the company's "day-to-day", especially with FedEx's demanding testing practices. VMware Tanzu, a portfolio of products and services that helps enterprises build, run and manage applications on Kubernetes ("an open-source container orchestration platform") is a service FedEx lauds as helping the company not only in its day-to-day operations, but also in giving it a greater ability to be agile, innovative, and efficient in meeting new demands with new applications.

Lum, credited VMware Tanzu with helping FedEx scale up to meet the demands of delivering the Covid vaccine roll-out as FedEx introduced new applications to more closely track deliveries by providing a trusted infrastructure.[7]

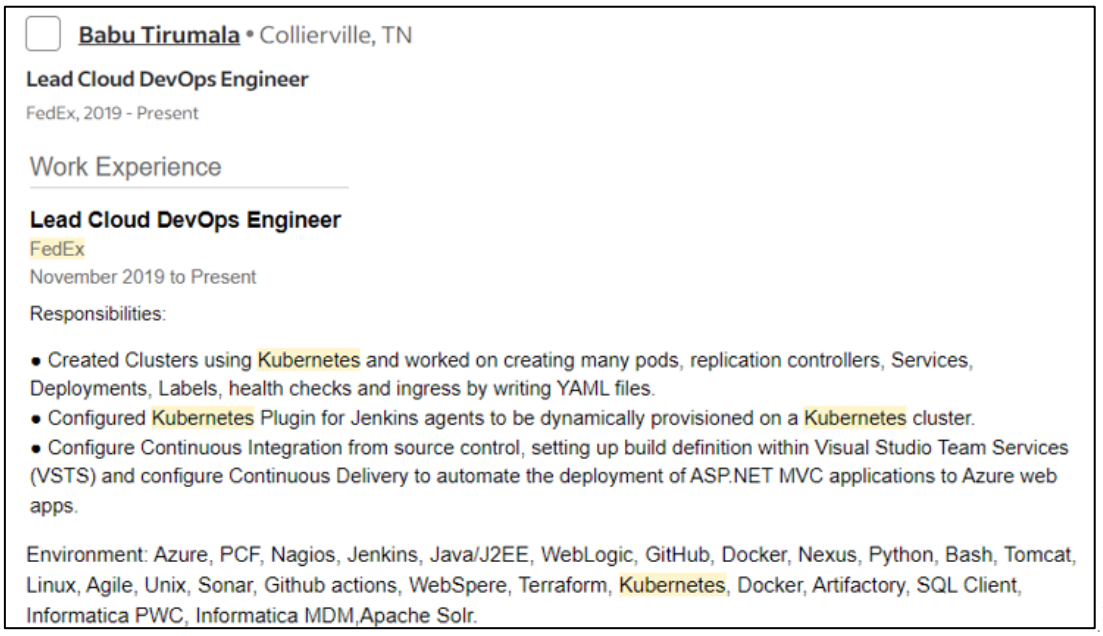


[8]

[7] https://www.linkedin.com/pulse/delivering-nexus-digital-physical-worlds-vmware-fedex-glen-gilmore/
[8] https://www.linkedin.com/in/kcnagareddy/

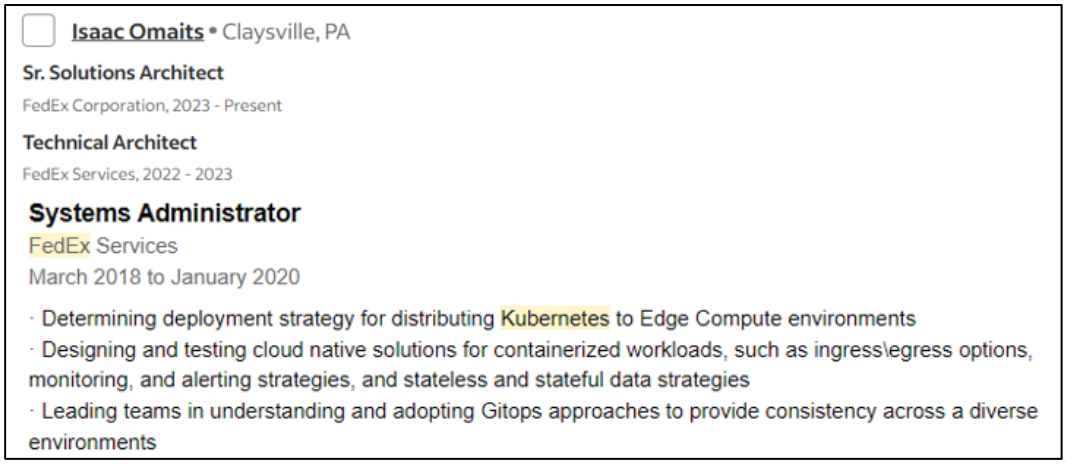


[9]



[10]

[9] https://www.linkedin.com/in/joshua-herndon-60890337/
[10] resumes.indeed.com – Smart Search (Login Required).



Isaac Omaits • Claysville, PA

Sr. Solutions Architect

FedEx Corporation, 2023 - Present

Technical Architect

FedEx Services, 2022 - 2023

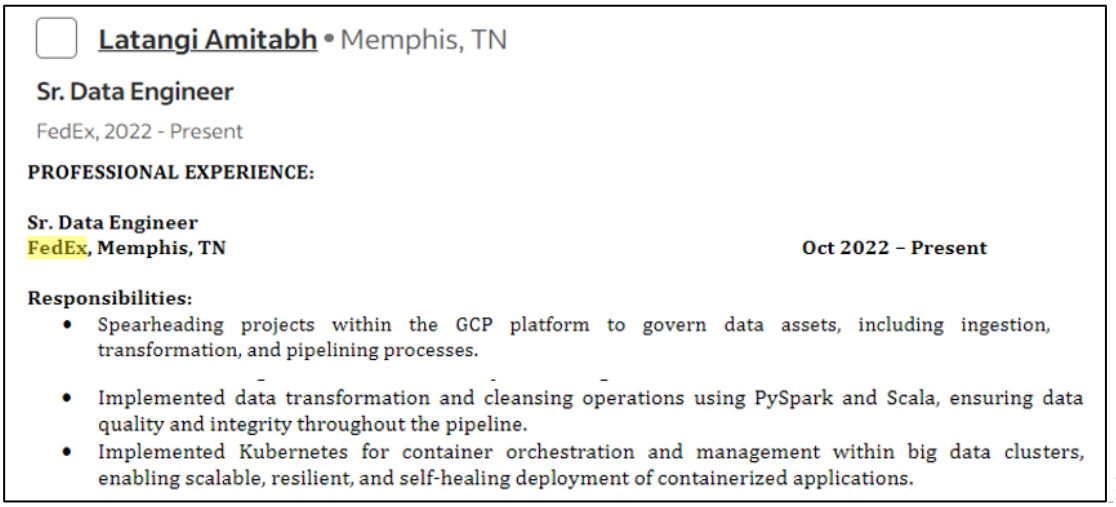
Systems Administrator

FedEx Services

March 2018 to January 2020

· Determining deployment strategy for distributing Kubernetes to Edge Compute environments
· Designing and testing cloud native solutions for containerized workloads, such as ingress\egress options, monitoring, and alerting strategies, and stateless and stateful data strategies
· Leading teams in understanding and adopting Gitops approaches to provide consistency across a diverse environments

[11]



Latangi Amitabh • Memphis, TN

Sr. Data Engineer

FedEx, 2022 - Present

PROFESSIONAL EXPERIENCE:

Sr. Data Engineer
FedEx, Memphis, TN **Oct 2022 – Present**

Responsibilities:

- Spearheading projects within the GCP platform to govern data assets, including ingestion, transformation, and pipelining processes.
- Implemented data transformation and cleansing operations using PySpark and Scala, ensuring data quality and integrity throughout the pipeline.
- Implemented Kubernetes for container orchestration and management within big data clusters, enabling scalable, resilient, and self-healing deployment of containerized applications.

[12]

[11] resumes.indeed.com – Smart Search (Login Required).
[12] resumes.indeed.com – Smart Search (Login Required).



[13]

[13] https://newsroom.fedex.com/newsroom/global/fedex-completes-acquisition-of-shoprunner-expanding-its-e-commerce-capabilities

ShopRunner
3 yrs 11 mos

Principal Engineer
Mar 2021 - Oct 2021 · 8 mos
Conshohocken, Pennsylvania, United States

Designed, managed and programmed the companies internal customer support app that included a React UI, Kotlin Backend and integration with Zendesk

Contributed to the rebuilding of the order management system moving it to a distributed system with a Kafka backbone.

Lead Engineer In Payments
Aug 2019 - Mar 2021 · 1 yr 8 mos

Grew and mentored team of engineers
Developed React Native digital wallet e-commerce application
Led legacy application migration from Python to Kotlin

Lead Engineer In CloudOps
Mar 2019 - Aug 2019 · 6 mos

Migrated container infrastructure to Kubernetes
Created templates for k8s deployments used across the organization
Advocated & built self service model to deploy infrastructure using Terraform [14]

Senior Software Engineer
ShopRunner · Full-time
Nov 2020 - Jul 2022 · 1 yr 9 mos
Conshohocken, Pennsylvania, United States

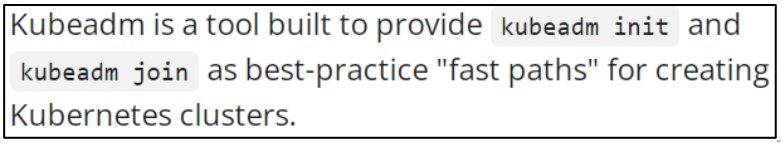
Worked on Kotlin, Python based backend applications that interacted with Mysql/PostgreSQL databases, storing Payment and Order/Shipment data. Made heavy use of ShopRunner's DEVOPS infrastructure including Kubernetes, Jenkins, Spinnaker, DataDog, SumoLogic, and other systems. Mentored more junior team members and gave company wide presentation on my work. [15]

Expert developers, known as "senseis," build or refactor FedEx applications on a PaaS (platform-as-a-service) system. Using container software to make them portable and Kubernetes technology to orchestrate the movement of containers between clouds, the PaaS platform enables engineers to shuttle apps between Microsoft Azure, Google Cloud Platform, Amazon Web Services and internal datacenters, based on business need. [16]

28. Kubernetes performs a method of creating a clustered computing system. For example, in Kubernetes, the Kubeadm tool is used to create clusters.

Kubeadm

[14] https://www.linkedin.com/in/asaf-peleg-007/details/experience/
[15] https://www.linkedin.com/in/monikamevenkamp/
[16] https://www.cio.com/article/217625/fedex-cloud-dojo-accelerates-digital-transformation.html



[17]





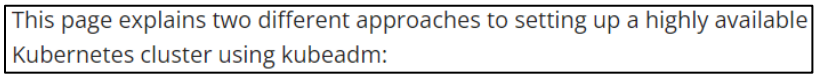
[18]

29. Kubernetes performs the step of storing license information for a computing cluster in a memory module associated with a computing node wherein the licensing information includes a bundle-type parameter that identifies a size of the cluster and at least one additional bundle-type characteristic of the cluster. For example, Kubeadm stores information including the number of worker nodes and number of control plane, and the characteristics of nodes.





[19]



[17] https://kubernetes.io/docs/reference/setup-tools/kubeadm/
[18] https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/
[19] https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/

- Three or more machines that meet kubeadm's minimum requirements for the control-plane nodes. Having an odd number of control plane nodes can help with leader selection in the case of machine or zone failure.
  - including a container runtime, already set up and working
- Three or more machines that meet kubeadm's minimum requirements for the workers
  - including a container runtime, already set up and working [20]

kubeadm's minimum requirements

- Unique hostname, MAC address, and product_uuid for every node. See here for more details. [21]

for handling container operations, such as containerd or CRI-O. A Kubernetes cluster that handles production traffic should have a minimum of three nodes because if one node goes down, both an etcd member and a control plane instance are lost, and redundancy is compromised. You can mitigate this risk by adding more control plane nodes. [22]

1. (Recommended) If you have plans to upgrade this single control-plane `kubeadm` cluster to high availability you should specify the `--control-plane-endpoint` to set the shared endpoint for all control-plane nodes. Such an endpoint can be either a DNS name or an IP address of a load-balancer. [23]

30. Kubernetes performs the steps of initializing the computing cluster in a first computing node and adding one or more available computing nodes to the computing cluster. For example, the "Kubeadm init" command initiates the cluster on a control plane node. Other nodes join this initialized cluster, in turn.

Initializing your control-plane node

[20] https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/
[21] https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/
[22] https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-intro/
[23] https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/

To initialize the control-plane node run:

```
kubeadm init <args>
```

`kubeadm init` first runs a series of prechecks to ensure that the machine is ready to run Kubernetes. These prechecks expose warnings and exit on errors. `kubeadm init` then downloads and installs the cluster control plane components. This may take several minutes. After it finishes you should see:

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a Pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  /docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node
as root:
```

```
kubeadm join <control-plane-host>:<control-plane-port> --token <token>
```

```
--discovery-token-ca-cert-hash sha256:<hash>
```

The nodes are where your workloads (containers and Pods, etc) run. To add new nodes to your cluster do the following for each machine:

- Run the command that was output by `kubeadm init`. For example:

[24]

31. Kubernetes performs the step of activating the computing cluster when the computing cluster includes a number of nodes that comply with the bundle-type parameter. For

[24] https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/

example, when all specified worker nodes are ready, and the control-plane joined and configured, the formed cluster is activated.

- Three or more machines that meet kubeadm's minimum requirements for the control-plane nodes. Having an odd number of control plane nodes can help with leader selection in the case of machine or zone failure.
  - including a container runtime, already set up and working
- Three or more machines that meet kubeadm's minimum requirements for the workers
  - including a container runtime, already set up and working

[25]

Node conformance test is a containerized test framework that provides a system verification and functionality test for a node. The test validates whether the node meets the minimum requirements for Kubernetes; a node that passes the test is qualified to join a Kubernetes cluster. [26]

A few seconds later, you should notice this node in the output from `kubectl get nodes` when run on the control-plane node.

```
[preflight] Running pre-flight checks

... (log output of join workflow) ...

Node join complete:
* Certificate signing request sent to control-plane and response
  received.
* Kubelet informed of new secure connection details.

Run 'kubectl get nodes' on control-plane to see this machine join.
```

1. List the nodes in your cluster, along with their labels:

```
kubectl get nodes --show-labels
```

```
NAME      STATUS    ROLES    AGE     VERSION        LABELS
worker0   Ready     <none>   1d      v1.13.0        ...,kubernetes.io/hostname=worker0
worker1   Ready     <none>   1d      v1.13.0        ...,kubernetes.io/hostname=worker1
worker2   Ready     <none>   1d      v1.13.0        ...,kubernetes.io/hostname=worker2
```

[27]

32. Defendants directly infringe the ’416 Patent through their implementation and use of Kubernetes for cluster creation, performing at least the steps of claim 1.

[25] https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/
[26] https://kubernetes.io/docs/setup/best-practices/node-conformance/
[27] https://kubernetes.io/docs/tasks/configure-pod-container/assign-pods-nodes/

33. Defendants have indirectly infringed and continue to indirectly infringe one or more claims of the ’416 Patent by knowingly and intentionally inducing others, including FedEx customers and end-users, to directly infringe, either literally or under the doctrine of equivalents, by making, using, offering to sell, selling, and/or importing into the United States products that include the infringing technology.

34. Defendants, with knowledge that these products, or the use thereof, infringe the ’416 Patent at least as of October 16, 2024,[28] knowingly and intentionally induced, and continue to knowingly and intentionally induce, direct infringement of the ’416 Patent by providing these products to end-users for use in an infringing manner. Additionally, on information and belief, Defendants have adopted a policy of not reviewing the patents of others, including specifically those related to Defendants’ specific industry, thereby remaining willfully blind to the ’416 Patent at least as early as the issuance of the Patents-in-Suit.

35. Defendants have induced infringement by others, including end-users, with the intent to cause infringing acts by others or, in the alternative, with the belief that there was a high probability that others, including end-users, infringe the ’416 Patent, but while remaining willfully blind to the infringement. Defendants have induced, and continue to induce, infringement by their customers and end-users by supplying them with instructions on how to operate the infringing technology in an infringing manner, while also making publicly available information on the

[28] Plaintiffs sent Defendant FedEx Corporation correspondence on October 16, 2024, providing notice of infringement of the ’416 Patent. Therefore, Defendants had knowledge of the ’416 Patent as of at least October 2024

IPR2026-00060 EX2010 019

infringing technology via Defendants' website, product literature and packaging, and other publications.[29]

36. Plaintiffs have suffered damages as a result of Defendants' direct and indirect infringement of the '416 Patent in an amount to be proven at trial.

37. Plaintiffs have suffered, and will continue to suffer, irreparable harm as a result of Defendants' infringement of the '416 Patent, for which there is no adequate remedy at law, unless Defendants' infringement is enjoined by this Court.

38. Defendants have committed and continue to commit acts of infringement that Defendants actually knew or should have known constituted an unjustifiably high risk of infringement of at least one valid and enforceable claim of the '416 Patent. Defendants' direct and indirect infringement of the '416 Patent has been and continue to be willful, intentional, deliberate, and/or in conscious disregard of rights under the patent. Plaintiffs are entitled to an award of treble damages, reasonable attorney fees, and costs in bringing this action.

COUNT II
(Infringement of the '686 Patent)

39. Paragraphs 1 through 22 are incorporated by reference as if fully set forth herein.

40. Plaintiffs have not licensed or otherwise authorized Defendants to make, use, offer for sale, sell, or import any products that embody the inventions of the '686 Patent.

41. Defendants have infringed, and continue to directly infringe, the '686 Patent, either literally or under the doctrine of equivalents, without authority and in violation of 35 U.S.C. § 271, by making, using, offering to sell, selling, and/or importing into the United States products that

[29] *See, e.g.,* https://www.fedex.com/en-us/developer/get-support.html#:~:text=1.800.GoFedEx%201.800.463.3339%20(,say%20%E2%80%9CInternational%20Services.%E2%80%9D); https://www.fedex.com/en-us/customer-support.html

IPR2026-00060 EX2010 020

satisfy each and every limitation of one or more claims of the ’686 Patent. Such products include HDFS.

42. On information and belief, FedEx has used Apache Hadoop in the U.S. since at least 2019.

FedEx

Data Scientist - All Levels

Company: FedEx Dataworks

Category: Professional

Employment Type: Full Time

Worker Sub-Type: Regular Remote Worker

Remote: Yes

Location:

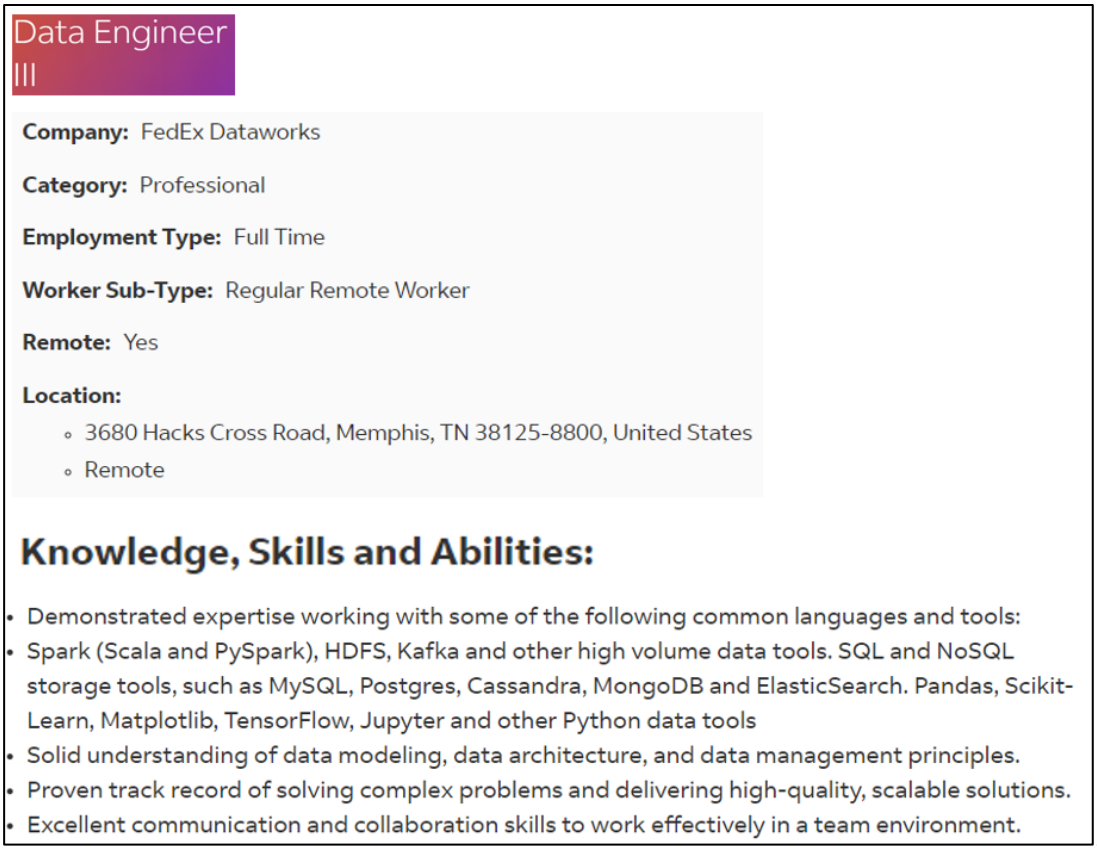
- 3680 Hacks Cross Road, Memphis, TN 38125-8800, United States
- Remote

Skills/Abilities:

- Technical background in computer science, data science, machine learning, artificial intelligence, statistics or other quantitative and computational science
- A compelling track record of Data Science / Engineer expertise, designing and deploying large scale technical solutions, which deliver tangible, ongoing value
- Direct experience having built and deployed robust, complex production systems through Cloud technologies that implement modern, data scientific methods at scale
- SKLearn, XGBoost, Tensorflow, Pytorch, MLlib and other core ML frameworks
- Python, Scala, R, C++, Java and other modern programming languages
- MLFlow, Databricks, Spark, Kafka, Hive, Hadoop and other data tools and frameworks
- CPLEX, Gurobi and other similar optimization modeling packages

[30]

[30] https://careers.fedex.com/data-scientist-all-levels/job/P25-101260-3?utm_source=CareerSiteOrganic



Data Engineer III

Company: FedEx Dataworks

Category: Professional

Employment Type: Full Time

Worker Sub-Type: Regular Remote Worker

Remote: Yes

Location:

- 3680 Hacks Cross Road, Memphis, TN 38125-8800, United States
- Remote

Knowledge, Skills and Abilities:

- Demonstrated expertise working with some of the following common languages and tools:
- Spark (Scala and PySpark), HDFS, Kafka and other high volume data tools. SQL and NoSQL storage tools, such as MySQL, Postgres, Cassandra, MongoDB and ElasticSearch. Pandas, Scikit-Learn, Matplotlib, TensorFlow, Jupyter and other Python data tools
- Solid understanding of data modeling, data architecture, and data management principles.
- Proven track record of solving complex problems and delivering high-quality, scalable solutions.
- Excellent communication and collaboration skills to work effectively in a team environment.

[31]

[31] https://careers.fedex.com/data-engineer-iii/job/P25-150097-1?utm_source=CareerSiteOrganic

Data Engineer II

Company: FedEx Dataworks

Category: Professional

Employment Type: Full Time

Worker Sub-Type: Regular Remote Worker

Remote: Yes

Location:

- 3680 Hacks Cross Road, Memphis, TN 38125-8800, United States
- Remote

Knowledge, Skills and Abilities

- Demonstrated expertise working with some of the following common languages and tools:
- Spark (Scala and PySpark), HDFS, Kafka and other high volume data tools
- SQL and NoSQL storage tools, such as MySQL, Postgres, Cassandra, MongoDB and Elastic Search
- Pandas, Scikit-Learn, Matplotlib, TensorFlow, Jupyter and other Python data tools

[32]

Deepak N. · 3rd

Senior Data Engineer | Databricks Certified Data Engineer Associate | IBM Professional Data Engineer | Microsoft Azure Data Engineer Associate| Expert in ETL, Data Warehousing,andReal-TimeAnalytics | PySpark&Apache spark

United States · Contact info

Experience

FedEx

Sr. Data Engineer

FedEx · Contract

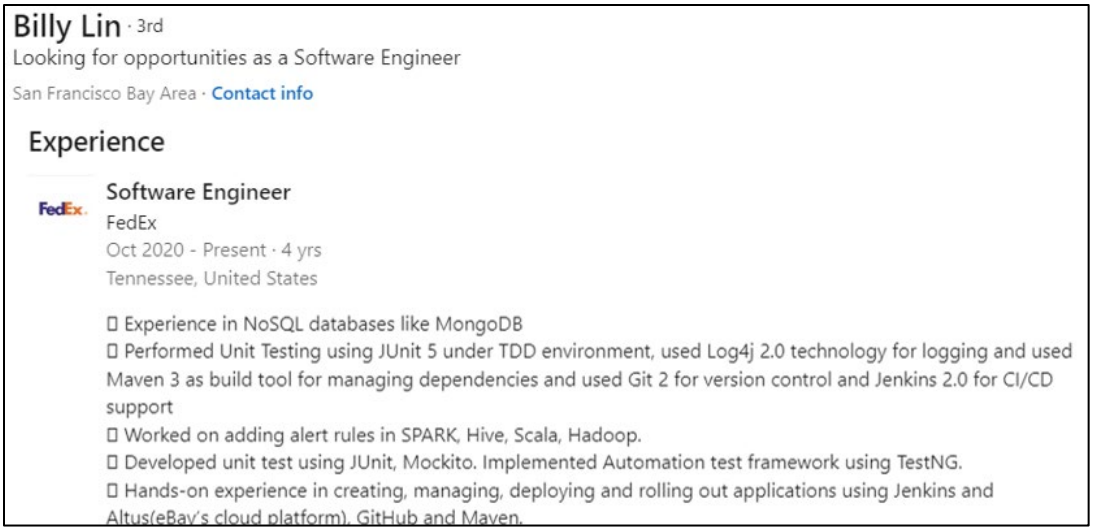
Aug 2022 - Present · 2 yrs 2 mos

Dallas, Texas, United States · Hybrid

- Worked on Spark improving the performance and optimization of the existing algorithms in Hadoop using Spark Context, Spark-SQL, Data Frame, Pair RDD's
- Developed PySpark script to merge static and dynamic files and cleanse the data.
- Created PySpark procedures, functions, packages to load data.
- Solved performance issues in Hive and Pig scripts with understanding of Joins, Group and Aggregation and how does it translate to MapReduce jobs.

[33]

[32] https://careers.fedex.com/data-engineer-ii/job/P25-150088-1?utm_source=CareerSiteOrganic
[33] https://www.linkedin.com/in/deepak-n-7b372915a/

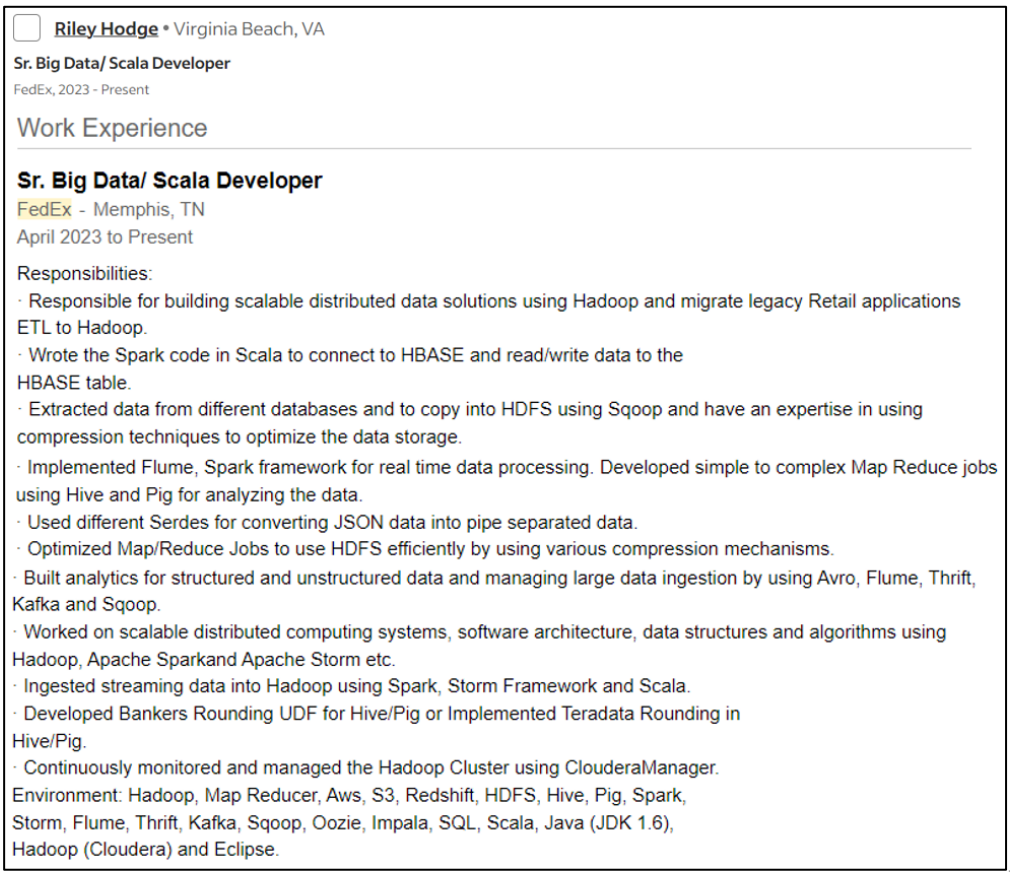


Billy Lin · 3rd
Looking for opportunities as a Software Engineer
San Francisco Bay Area · Contact info

Experience

Software Engineer
FedEx
Oct 2020 - Present · 4 yrs
Tennessee, United States

- Experience in NoSQL databases like MongoDB
- Performed Unit Testing using JUnit 5 under TDD environment, used Log4j 2.0 technology for logging and used Maven 3 as build tool for managing dependencies and used Git 2 for version control and Jenkins 2.0 for CI/CD support
- Worked on adding alert rules in SPARK, Hive, Scala, Hadoop.
- Developed unit test using JUnit, Mockito. Implemented Automation test framework using TestNG.
- Hands-on experience in creating, managing, deploying and rolling out applications using Jenkins and Altus(eBay's cloud platform), GitHub and Maven.



Riley Hodge • Virginia Beach, VA
Sr. Big Data/ Scala Developer
FedEx, 2023 - Present

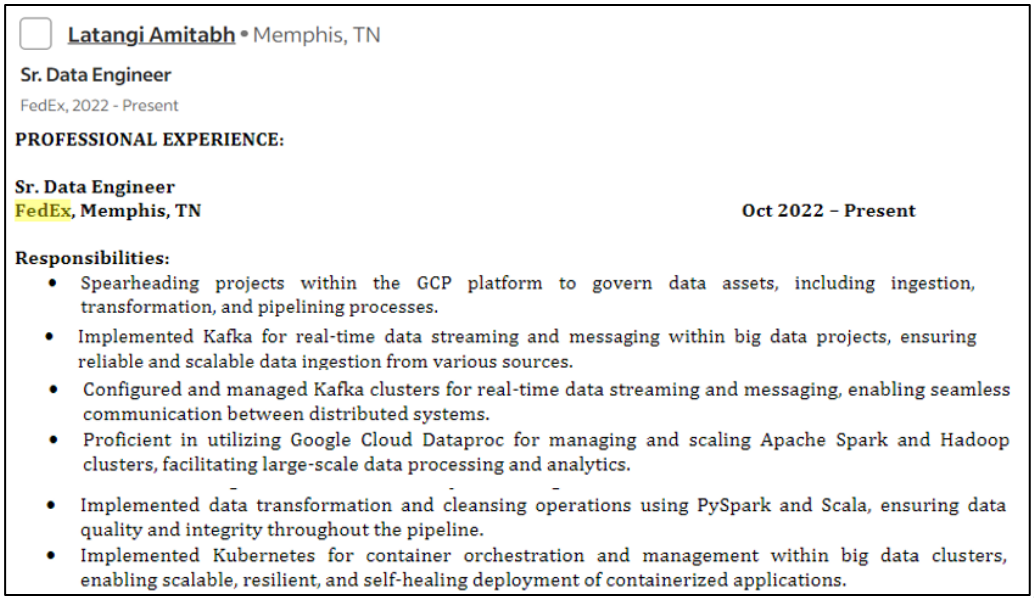
Work Experience

Sr. Big Data/ Scala Developer
FedEx - Memphis, TN
April 2023 to Present

Responsibilities:
· Responsible for building scalable distributed data solutions using Hadoop and migrate legacy Retail applications ETL to Hadoop.
· Wrote the Spark code in Scala to connect to HBASE and read/write data to the HBASE table.
· Extracted data from different databases and to copy into HDFS using Sqoop and have an expertise in using compression techniques to optimize the data storage.
· Implemented Flume, Spark framework for real time data processing. Developed simple to complex Map Reduce jobs using Hive and Pig for analyzing the data.
· Used different Serdes for converting JSON data into pipe separated data.
· Optimized Map/Reduce Jobs to use HDFS efficiently by using various compression mechanisms.
· Built analytics for structured and unstructured data and managing large data ingestion by using Avro, Flume, Thrift, Kafka and Sqoop.
· Worked on scalable distributed computing systems, software architecture, data structures and algorithms using Hadoop, Apache Sparkand Apache Storm etc.
· Ingested streaming data into Hadoop using Spark, Storm Framework and Scala.
· Developed Bankers Rounding UDF for Hive/Pig or Implemented Teradata Rounding in Hive/Pig.
· Continuously monitored and managed the Hadoop Cluster using ClouderaManager.
Environment: Hadoop, Map Reducer, Aws, S3, Redshift, HDFS, Hive, Pig, Spark, Storm, Flume, Thrift, Kafka, Sqoop, Oozie, Impala, SQL, Scala, Java (JDK 1.6), Hadoop (Cloudera) and Eclipse.

[34]

[34] resumes.indeed.com – Smart Search (Login Required).



Latangi Amitabh • Memphis, TN

Sr. Data Engineer

FedEx, 2022 - Present

PROFESSIONAL EXPERIENCE:

Sr. Data Engineer
FedEx, Memphis, TN **Oct 2022 – Present**

Responsibilities:

- Spearheading projects within the GCP platform to govern data assets, including ingestion, transformation, and pipelining processes.
- Implemented Kafka for real-time data streaming and messaging within big data projects, ensuring reliable and scalable data ingestion from various sources.
- Configured and managed Kafka clusters for real-time data streaming and messaging, enabling seamless communication between distributed systems.
- Proficient in utilizing Google Cloud Dataproc for managing and scaling Apache Spark and Hadoop clusters, facilitating large-scale data processing and analytics.
- Implemented data transformation and cleansing operations using PySpark and Scala, ensuring data quality and integrity throughout the pipeline.
- Implemented Kubernetes for container orchestration and management within big data clusters, enabling scalable, resilient, and self-healing deployment of containerized applications.

[35]



Anuradha Modali

Sr. Data Engineer

Patient Pay Inc., 2021 - Present

Data Engineer

FedEx, USA, 2019 - 2021

FedEx, USA **Jul 2019 – Aug 2021**
Data Engineer

Responsibilities:

- Translating customer requirement with HealthCare data (Claims, Membership and Eligibility), FHIR for analysis and implementation.
- Set up and worked on Kerberos authentication principles to establish secure network communication on cluster and testing of HDFS, Hive, Pig and MapReduce to access cluster for new users.
- Build data pipelines in airflow in GCP for ETL related jobs using different airflow operators.
- Experience in GCP Dataproc, GCS, Cloud functions, Big Query
- Developing complex ETL jobs using Jupyter notebooks, PySpark and Python.

[36]

43. For example, Defendants have infringed, and continue to directly infringe, at least claim 1 of the ’686 Patent by making, using, offering to sell, selling, and/or importing into the United States products that include HDFS.

44. HDFS performs a method of providing data security for use with a file system. For example, the Apache Hadoop framework provides security features to protect data present in

[35] resumes.indeed.com – Smart Search (Login Required).
[36] resumes.indeed.com – Smart Search (Login Required).

Hadoop clusters. Hadoop uses the HDFS (a file system) to store data. HDFS comprises a name node and multiple data nodes.

> Hadoop, an open-source framework used for distributed storage and processing of big data, provides various security features to protect data and ensure secure access to Hadoop clusters. Here are some of the ways Hadoop is secured:

> 2. Authorization: Hadoop uses Access Control Lists (ACLs) to manage permissions for users and groups accessing data on the Hadoop Distributed File System (HDFS). Additionally, Hadoop provides fine-grained authorization through the use of Apache Ranger, which allows administrators to create policies that define access control for specific resources and actions.

[37]

> **NameNode and DataNodes**
>
> HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

[38]

> **The Communication Protocols**
>
> All HDFS communication protocols are layered on top of the TCP/IP protocol. A client establishes a connection to a configurable TCP port on the NameNode machine. It talks the ClientProtocol with the NameNode. The DataNodes talk to the NameNode using the DataNode Protocol. A Remote Procedure Call (RPC) abstraction wraps both the Client Protocol and the DataNode Protocol. By design, the NameNode never initiates any RPCs. Instead, it only responds to RPC requests issued by DataNodes or clients.

[39]

45. HDFS performs the step of applying a mapping function to data block numbers that are associated with a file, wherein the data block numbers are contained in an index node

[37] https://medium.com/@big_data_landscape/hadoop-security-5dac0ffe209e

[38] http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html#NameNode_and_DataNodes

[39] https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

associated with said file. For example, NameNode performs various operations, such as opening and closing of files, and mapping data block locations (e.g., data block number). Mapping (e.g., mapping function) is the process of determining which data block is stored at which location. By way of further example, data blocks are associated with a file. The data block locations are stored in the NameNode's (e.g., index node) memory.

If you take another look at Figure 4-3, you can see the NameNode daemon running on a master node server. All mapping information dealing with the data blocks and their corresponding files is stored in a file named `fsimage`. HDFS is a journaling file system, which means that any data changes are logged in an edit journal that tracks events since the last *checkpoint* — the last time when the edit log was merged with `fsimage`. In HDFS, the edit journal is maintained in a file named `edits` that's stored on the NameNode.

The client opens the file it wishes to read by calling `open()` on the `FileSystem` object, which for HDFS is an instance of `DistributedFileSystem` (step 1 in Figure 3-2). `DistributedFileSystem` calls the namenode, using remote procedure calls (RPCs), to determine the locations of the first few blocks in the file (step 2). For each block, the namenode returns the addresses of the datanodes that have a copy of that block. Furthermore, the datanodes are sorted according to their proximity to the client (according to the topology of the cluster's network; see "Network Topology and Hadoop" on page 70). If the client is itself a datanode (in the case of a MapReduce task, for instance), the client will read from the local datanode if that datanode hosts a copy of the block (see also Figure 2-2 and "Short-circuit local reads" on page 308). [40]

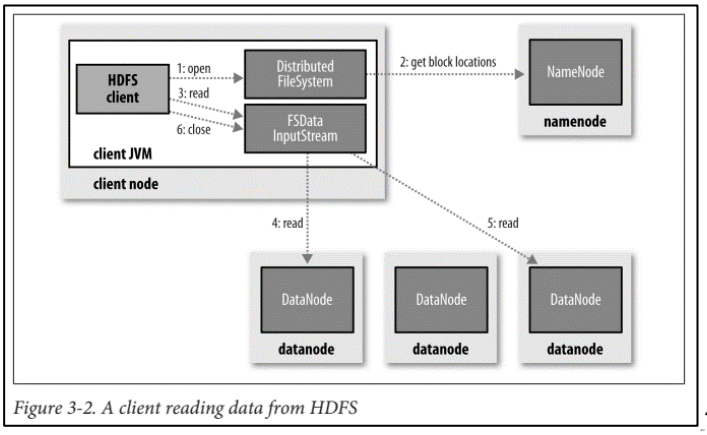


Figure 3-2. A client reading data from HDFS [41]

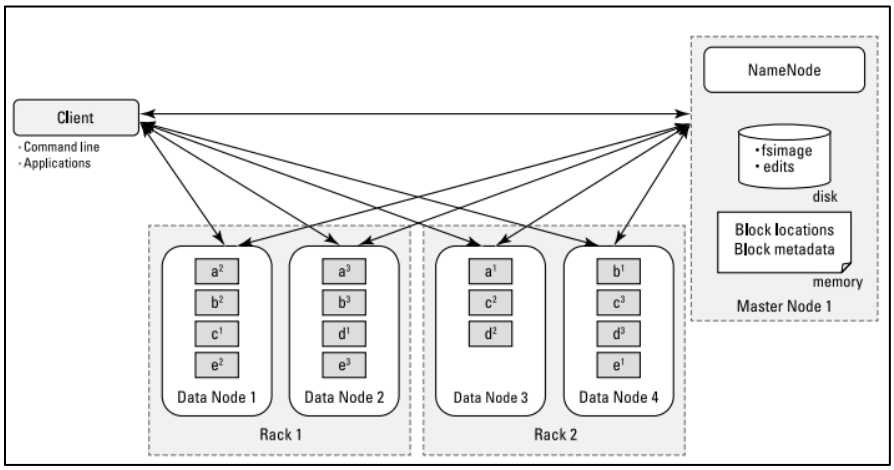
40 https://ia800201.us.archive.org/2/items/HadoopForDummiesDirkDeRoos2014/Hadoop%20For%20Dummies%20-%20%20Dirk%20deRoos%202014.pdf, at Page 74 of 411

41 https://grut-computing.com/HadoopBook.pdf, at Page 97 of 756.

NameNode and DataNodes

HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode. [42]

After the startup process is completed, the NameNode has a complete picture of all the data stored in HDFS, and it's ready to receive application requests from Hadoop clients. As data files are added and removed based on client requests, the changes are written to the slave node's disk volumes, journal updates are made to the `edits` file, and the changes are reflected in the block locations and metadata stored in the NameNode's memory (see Figure 4-4).



Throughout the life of the cluster, the DataNode daemons send the NameNode heartbeats (a quick signal) every three seconds, indicating they're active. (This default value is configurable.) Every six hours (again, a configurable default), the DataNodes send the NameNode a block report outlining which file blocks are on their nodes. This way, the NameNode always has a current view of the available resources in the cluster. [43]

[42] https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

[43] https://ia800201.us.archive.org/2/items/HadoopForDummiesDirkDeRoos2014/Hadoop%20For%20Dummies%20-%20%20Dirk%20deRoos%202014.pdf, at Page 75 of 411

46. HDFS performs the step of obtaining mapped data block numbers after applying the mapping function, wherein the mapped data block numbers are addresses of data of the file in a storage device. For example, NameNode determines "the addresses of the data nodes" (e.g., mapped data block numbers) on which the data block is stored. Data nodes (e.g., storage device(s)) store the data blocks of a file.

> The client opens the file it wishes to read by calling `open()` on the `FileSystem` object, which for HDFS is an instance of `DistributedFileSystem` (step 1 in Figure 3-2). `DistributedFileSystem` calls the namenode, using remote procedure calls (RPCs), to determine the locations of the first few blocks in the file (step 2). For each block, the namenode returns the addresses of the datanodes that have a copy of that block. Furthermore, the datanodes are sorted according to their proximity to the client (according to the topology of the cluster's network; see "Network Topology and Hadoop" on page 70). If the client is itself a datanode (in the case of a MapReduce task, for instance), the client will read from the local datanode if that datanode hosts a copy of the block (see also Figure 2-2 and "Short-circuit local reads" on page 308).

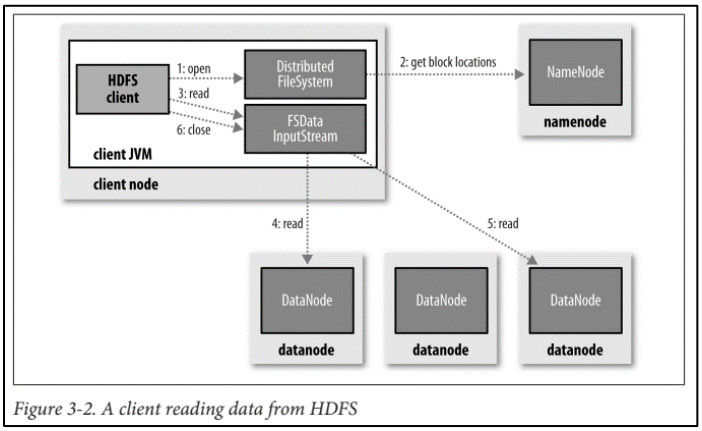


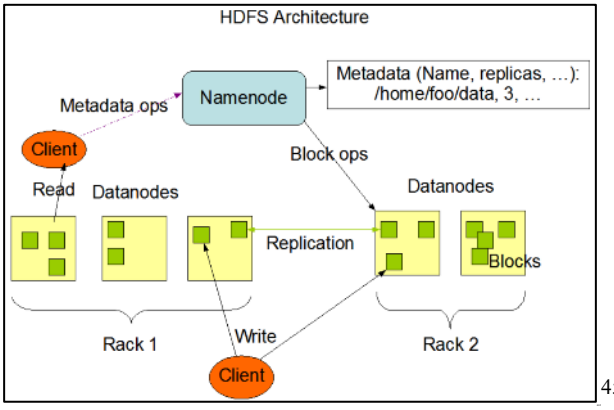
Figure 3-2. A client reading data from HDFS

> One important aspect of this design is that the client contacts datanodes directly to retrieve data and is guided by the namenode to the best datanode for each block. This design allows HDFS to scale to a large number of concurrent clients because the data traffic is spread across all the datanodes in the cluster. Meanwhile, the namenode merely has to service block location requests (which it stores in memory, making them very efficient) and does not, for example, serve data, which would quickly become a bottleneck as the number of clients grew.[44]

[44] https://grut-computing.com/HadoopBook.pdf, at Pages 97-98 of 756.

NameNode and DataNodes

HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.



[45]

47. Defendants directly infringe the '686 Patent through their implementation and use of HDFS, performing at least the steps of claim 1.

48. Defendants have indirectly infringed and continue to indirectly infringe one or more claims of the '686 Patent by knowingly and intentionally inducing others, including FedEx customers and end-users, to directly infringe, either literally or under the doctrine of equivalents, by making, using, offering to sell, selling, and/or importing into the United States products that include the infringing technology.

[45] https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

49. Defendants, with knowledge that these products, or the use thereof, infringe the '686 Patent at least as of October 16, 2024,[46] knowingly and intentionally induced, and continue to knowingly and intentionally induce, direct infringement of the '686 Patent by providing these products to end-users for use in an infringing manner. Additionally, on information and belief, Defendants have adopted a policy of not reviewing the patents of others, including specifically those related to Defendants' specific industry, thereby remaining willfully blind to the '686 Patent at least as early as the issuance of the '686 Patent.

50. Defendants have induced infringement by others, including end-users, with the intent to cause infringing acts by others or, in the alternative, with the belief that there was a high probability that others, including end-users, infringe the '686 Patent, but while remaining willfully blind to the infringement. Defendants have and continue to induce infringement by their customers and end-users by supplying them with instructions on how to operate the infringing technology in an infringing manner, while also making publicly available information on the infringing technology via Defendants' website, product literature and packaging, and other publications.[47]

51. Plaintiffs have suffered damages as a result of Defendants' direct and indirect infringement of the '686 Patent in an amount to be proven at trial.

52. Plaintiffs have suffered, and will continue to suffer, irreparable harm as a result of Defendants' infringement of the '686 Patent, for which there is no adequate remedy at law, unless Defendants' infringement is enjoined by this Court.

---

[46] Plaintiffs sent Defendant FedEx Corporation correspondence on October 8, 2024, providing notice of infringement of the '686 Patent. Therefore, Defendants had knowledge of the '686 Patent as of at least October 2024.

[47] *See, e.g.,* https://www.fedex.com/en-us/developer/get-support.html#:~:text=1.800.GoFedEx%201.800.463.3339%20(,say%20%E2%80%9CInternational%20Services.%E2%80%9D); https://www.fedex.com/en-us/customer-support.html

53. Defendants have committed and continue to commit acts of infringement that Defendants actually knew or should have known constituted an unjustifiably high risk of infringement of at least one valid and enforceable claim of the ’686 Patent. Defendants’ direct and indirect infringement of the ’686 Patent has been and continue to be willful, intentional, deliberate, and/or in conscious disregard of rights under the patent. Plaintiffs are entitled to an award of treble damages, reasonable attorney fees, and costs in bringing this action.

COUNT III
(Infringement of the ’332 Patent)

54. Paragraphs 1 through 22 are incorporated by reference as if fully set forth herein.

55. Plaintiffs have not licensed or otherwise authorized Defendants to make, use, offer for sale, sell, or import any products that embody the inventions of the ’332 Patent.

56. Defendants have infringed, and continue to directly infringe, the ’332 Patent, either literally or under the doctrine of equivalents, without authority and in violation of 35 U.S.C. § 271, by making, using, offering to sell, selling, and/or importing into the United States products that satisfy each and every limitation of one or more claims of the ’332 Patent. Such products include Kubernetes.

57. For example, Defendants have infringed, and continue to directly infringe, at least claim 1 of the ’332 Patent by making, using, offering to sell, selling, and/or importing into the United States products that include Kubernetes.

58. On information and belief, FedEx has used Kubernetes in the U.S. since at least 2019, as set forth in Paragraph 27.

59. Kubernetes performs the method for hot deployment and/or redeployment in grid computing environment, wherein the grid computing environment includes one or more grid nodes. For example, Kubernetes uses a method of rolling updates to allow for hot deployment and

IPR2026-00060 EX2010 032

redeployment (rollouts and rollbacks). Using rolling updates, application containers (e.g., pods) are updated and scheduled on worker nodes of the cluster.

> Kubernetes progressively rolls out changes to your application or its configuration, while monitoring application health to ensure it doesn't kill all your instances at the same time. If something goes wrong, Kubernetes will rollback the change for you. Take advantage of a growing ecosystem of deployment solutions.

[48]

Core Components

A Kubernetes cluster consists of a control plane and one or more worker nodes. Here's a brief overview of the main components:

Control Plane Components

Manage the overall state of the cluster:

kube-apiserver

The core component server that exposes the Kubernetes HTTP API

etcd

Consistent and highly-available key value store for all API server data

kube-scheduler

Looks for Pods not yet bound to a node, and assigns each Pod to a suitable node.

kube-controller-manager

Runs controllers to implement Kubernetes API behavior.

cloud-controller-manager (optional)

Integrates with underlying cloud provider(s).

[48] https://kubernetes.io/

Node Components

Run on every node, maintaining running pods and providing the Kubernetes runtime environment:

kubelet

Ensures that Pods are running, including their containers.

kube-proxy (optional)

Maintains network rules on nodes to implement Services.

Container runtime

Software responsible for running containers. Read Container Runtimes to learn more.[49]

Users expect applications to be available all the time, and developers are expected to deploy new versions of them several times a day. In Kubernetes this is done with rolling updates. A **rolling update** allows a Deployment update to take place with zero downtime. It does this by incrementally replacing the current Pods with new ones. The new Pods are scheduled on Nodes with available resources, and Kubernetes waits for those new Pods to start before removing the old Pods.[50]

60. Kubernetes performs the step of adding a new version of an application release bundle in a repository server. For example, container images are software bundles shared in a registry. To rollout a new version of the software, a container image with the new version is first formed.

Once you have a running Kubernetes cluster, you can deploy your containerized applications on top of it. To do so, you create a Kubernetes **Deployment**. The Deployment instructs Kubernetes how to create and update instances of your application. Once you've created a Deployment, the Kubernetes control plane schedules the application instances included in that Deployment to run on individual Nodes in the cluster.

[49] https://kubernetes.io/docs/concepts/overview/components/
[50] https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/

When you create a Deployment, you'll need to specify the container image for your application and the number of replicas that you want to run. You can change that information later by updating your Deployment; Module 5 and Module 6 of the bootcamp discuss how you can scale and update your Deployments. [51]

Images

A container image represents binary data that encapsulates an application and all its software dependencies. Container images are executable software bundles that can run standalone and that make very well defined assumptions about their runtime environment.

You typically create a container image of your application and push it to a registry before referring to it in a Pod.

If you don't specify a registry hostname, Kubernetes assumes that you mean the Docker public registry. You can change this behaviour by setting default image registry in container runtime configuration. [52]

61. Kubernetes performs the step of determining by a discovery services module which of the one or more grid nodes are running an application associated with the added new version of the application release bundle upon adding the new version of the application release bundle in the repository server. For example, a control plane has a deployment controller that ensures that the new container image is deployed all across the ReplicaSet of the cluster.

Updating an application

Users expect applications to be available all the time, and developers are expected to deploy new versions of them several times a day. In Kubernetes this is done with rolling updates. A **rolling update** allows a Deployment update to take place with zero downtime. It does this by incrementally replacing the current Pods with new ones. The new Pods are scheduled on Nodes with available resources, and Kubernetes waits for those new Pods to start before removing the old Pods. [53]

[51] https://kubernetes.io/docs/tutorials/kubernetes-basics/deploy-app/deploy-intro/
[52] https://kubernetes.io/docs/concepts/containers/images/
[53] https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/

Initially, the pods run the first version of your application—let's suppose its image is tagged as `v1`. You then develop a newer version of the app and push it to an image repository as a new image, tagged as `v2`. You'd next like to replace all the pods with this new version. Because you can't change an existing pod's image after the pod is created, you need to remove the old pods and replace them with new ones running the new image.[54]

Deployments

A Deployment manages a set of Pods to run an application workload, usually one that doesn't maintain state.

A *Deployment* provides declarative updates for Pods and ReplicaSets.

You describe a *desired state* in a Deployment, and the Deployment Controller changes the actual state to the desired state at a controlled rate. You can define Deployments to create new ReplicaSets, or to remove existing Deployments and adopt all their resources with new Deployments.

Each time a new Deployment is observed by the Deployment controller, a ReplicaSet is created to bring up the desired Pods. If the Deployment is updated, the existing ReplicaSet that controls Pods whose labels match `.spec.selector` but whose template does not match `.spec.template` are scaled down. Eventually, the new ReplicaSet is scaled to `.spec.replicas` and all old ReplicaSets is scaled to 0.[55]

Once you have a running Kubernetes cluster, you can deploy your containerized applications on top of it. To do so, you create a Kubernetes **Deployment**. The Deployment instructs Kubernetes how to create and update instances of your application. Once you've created a Deployment, the Kubernetes control plane schedules the application instances included in that Deployment to run on individual Nodes in the cluster.

Once the application instances are created, a Kubernetes Deployment controller continuously monitors those instances. If the Node hosting an instance goes down or is deleted, the Deployment controller replaces the instance with an instance on another Node in the cluster. **This provides a self-healing mechanism to address machine failure or maintenance.**[56]

[54] https://livebook.manning.com/book/kubernetes-in-action/chapter-9/13
[55] https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
[56] https://kubernetes.io/docs/tutorials/kubernetes-basics/deploy-app/deploy-intro/

IPR2026-00060 EX2010 036

62. Kubernetes performs the step of notifying a client application manager associated with one or more of the determined grid nodes about adding the new version of the application release bundle along with a type of data transfer protocol to use. For example, the Deployment controller notifies Kubelet processes in worker nodes. Each Kubelet process is therefore notified of the desired configuration including the new application and associated protocol information.

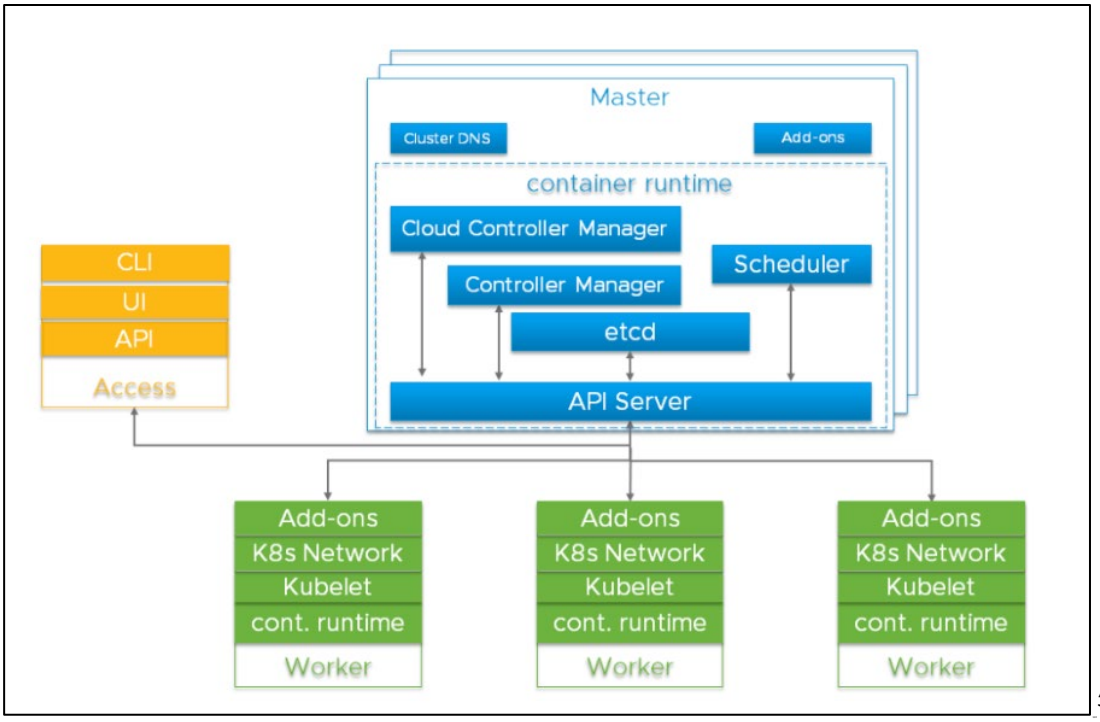
> Each time a new Deployment is observed by the Deployment controller, a ReplicaSet is created to bring up the desired Pods. If the Deployment is updated, the existing ReplicaSet that controls Pods whose labels match `.spec.selector` but whose template does not match `.spec.template` are scaled down. Eventually, the new ReplicaSet is scaled to `.spec.replicas` and all old ReplicaSets is scaled to 0.

[57]

> The kubelet is the primary "node agent" that runs on each node. It can register the node with the apiserver using one of: the hostname; a flag to override the hostname; or specific logic for a cloud provider.
>
> The kubelet works in terms of a PodSpec. A PodSpec is a YAML or JSON object that describes a pod. The kubelet takes a set of PodSpecs that are provided through various mechanisms (primarily through the apiserver) and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

[58]

[57] https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
[58] https://kubernetes.io/docs/concepts/overview/components/#kubelet



[59]

[59] https://kubernetes.io/blog/2018/08/03/out-of-the-clouds-onto-the-ground-how-to-make-kubernetes-production-grade-anywhere/

```
Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Thu, 30 Nov 2017 10:56:25 +0000
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision=2
Selector:               app=nginx
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
   Containers:
    nginx:
      Image:        nginx:1.16.1
      Port:         80/TCP
      Environment:  <none>
      Mounts:       <none>
    Volumes:        <none>
  Conditions:
    Type           Status  Reason
    ----           ------  ------
    Available      True    MinimumReplicasAvailable
    Progressing    True    NewReplicaSetAvailable
  OldReplicaSets:  <none>
  NewReplicaSet:   nginx-deployment-1564180365 (3/3 replicas created)
  Events:
    Type    Reason             Age   From                   Message
    ----    ------             ----  ----                   -------
    Normal  ScalingReplicaSet  2m    deployment-controller  Scaled up replica set nginx-deployment-2035384211 to 3
    Normal  ScalingReplicaSet  24s   deployment-controller  Scaled up replica set nginx-deployment-1564180365 to 1
    Normal  ScalingReplicaSet  22s   deployment-controller  Scaled down replica set nginx-deployment-2035384211 to 2
    Normal  ScalingReplicaSet  22s   deployment-controller  Scaled up replica set nginx-deployment-1564180365 to 2
    Normal  ScalingReplicaSet  19s   deployment-controller  Scaled down replica set nginx-deployment-2035384211 to 1
    Normal  ScalingReplicaSet  19s   deployment-controller  Scaled up replica set nginx-deployment-1564180365 to 3
    Normal  ScalingReplicaSet  14s   deployment-controller  Scaled down replica set nginx-deployment-2035384211 to 0
```
[60]

63. Kubernetes performs the step of hot deploying/redeploying the new version of the application release bundle on running one or more application servers in an associated grid node using an appropriate hot deployment plug-in based on the data transfer protocol by a respective one of the client application managers. For example, rolling updates of the new software are made with the rollout command of the kubectl interface. The image is deployed in the worker nodes according to the desired configuration by the Kubelet process.

[60] https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

Updating an application

> *Rolling updates allow Deployments' update to take place with zero downtime by incrementally updating Pods instances with new ones.*

Users expect applications to be available all the time, and developers are expected to deploy new versions of them several times a day. In Kubernetes this is done with rolling updates. A **rolling update** allows a Deployment update to take place with zero downtime. It does this by incrementally replacing the current Pods with new ones. The new Pods are scheduled on Nodes with available resources, and Kubernetes waits for those new Pods to start before removing the old Pods. [61]

Deployments

A Deployment manages a set of Pods to run an application workload, usually one that doesn't maintain state.

A *Deployment* provides declarative updates for Pods and ReplicaSets.

You describe a *desired state* in a Deployment, and the Deployment Controller changes the actual state to the desired state at a controlled rate. You can define Deployments to create new ReplicaSets, or to remove existing Deployments and adopt all their resources with new Deployments.

- Declare the new state of the Pods by updating the PodTemplateSpec of the Deployment. A new ReplicaSet is created and the Deployment manages moving the Pods from the old ReplicaSet to the new one at a controlled rate. Each new ReplicaSet updates the revision of the Deployment.

[61] https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/

- The `.spec.template` field contains the following sub-fields:
  - The Pods are labeled `app: nginx` using the `.metadata.labels` field.
  - The Pod template's specification, or `.spec` field, indicates that the Pods run one container, `nginx`, which runs the `nginx` Docker Hub image at version 1.14.2.
  - Create one container and name it `nginx` using the `.spec.containers[0].name` field.

[62]

2. To see the rollout status, run:

```
kubectl rollout status deployment/nginx-deployment
```

The output is similar to this:

```
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
```

or

```
deployment "nginx-deployment" successfully rolled out
```

Get more details on your updated Deployment:

- After the rollout succeeds, you can view the Deployment by running `kubectl get deployments`. The output is similar to this:

```
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           36s
```

[62] https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

IPR2026-00060 EX2010 041

The output is similar to this:

```
NAME                              DESIRED   CURRENT   READY   AGE
nginx-deployment-1564180365   3         3         3       6s
nginx-deployment-2035384211   0         0         0       36s
```

[63]

```
Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Thu, 30 Nov 2017 10:56:25 +0000
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision=2
Selector:               app=nginx
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
   Containers:
    nginx:
      Image:        nginx:1.16.1
      Port:         80/TCP
      Environment:  <none>
      Mounts:       <none>
    Volumes:        <none>
  Conditions:
    Type           Status  Reason
    ----           ------  ------
    Available      True    MinimumReplicasAvailable
    Progressing    True    NewReplicaSetAvailable
  OldReplicaSets:  <none>
  NewReplicaSet:   nginx-deployment-1564180365 (3/3 replicas created)
  Events:
    Type    Reason             Age   From                   Message
    ----    ------             ----  ----                   -------
    Normal  ScalingReplicaSet  2m    deployment-controller  Scaled up replica set nginx-deployment-2035384211 to 3
    Normal  ScalingReplicaSet  24s   deployment-controller  Scaled up replica set nginx-deployment-1564180365 to 1
    Normal  ScalingReplicaSet  22s   deployment-controller  Scaled down replica set nginx-deployment-2035384211 to 2
    Normal  ScalingReplicaSet  22s   deployment-controller  Scaled up replica set nginx-deployment-1564180365 to 2
    Normal  ScalingReplicaSet  19s   deployment-controller  Scaled down replica set nginx-deployment-2035384211 to 1
    Normal  ScalingReplicaSet  19s   deployment-controller  Scaled up replica set nginx-deployment-1564180365 to 3
    Normal  ScalingReplicaSet  14s   deployment-controller  Scaled down replica set nginx-deployment-2035384211 to 0
```

[64]

64. Defendants directly infringe the ’332 Patent through their implementation and use of Kubernetes for rolling updates, performing at least the steps of claim 1.

65. Defendants have indirectly infringed and continue to indirectly infringe one or more claims of the ’332 Patent by knowingly and intentionally inducing others, including FedEx customers and end-users, to directly infringe, either literally or under the doctrine of equivalents,

[63] https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
[64] https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

by making, using, offering to sell, selling, and/or importing into the United States products that include the infringing technology.

66. Defendants, with knowledge that these products, or the use thereof, infringe the '332 Patent at least as of October 16, 2024,[65] knowingly and intentionally induced, and continue to knowingly and intentionally induce, direct infringement of the '332 Patent by providing these products to end-users for use in an infringing manner. Additionally, on information and belief, Defendants have adopted a policy of not reviewing the patents of others, including specifically those related to Defendants' specific industry, thereby remaining willfully blind to the '332 Patent at least as early as the issuance of the '332 Patent.

67. Defendants have induced infringement by others, including end-users, with the intent to cause infringing acts by others or, in the alternative, with the belief that there was a high probability that others, including end-users, infringe the '332 Patent, but while remaining willfully blind to the infringement. Defendants have and continue to induce infringement by their customers and end-users by supplying them with instructions on how to operate the infringing technology in an infringing manner, while also making publicly available information on the infringing technology via Defendants' website, product literature and packaging, and other publications.[66]

68. Plaintiffs have suffered damages as a result of Defendants' direct and indirect infringement of the '332 Patent in an amount to be proven at trial.

[65] Plaintiffs sent Defendant FedEx Corporation correspondence on October 16, 2024, providing notice of infringement of the '332 Patent. Therefore, Defendants had knowledge of the '332 Patent as of at least October 2024.

[66] *See, e.g.,* https://www.fedex.com/en-us/developer/get-support.html#:~:text=1.800.GoFedEx%201.800.463.3339%20(,say%20%E2%80%9CInternational%20Services.%E2%80%9D); https://www.fedex.com/en-us/customer-support.html

69. Plaintiffs have suffered, and will continue to suffer, irreparable harm as a result of Defendants' infringement of the '332 Patent, for which there is no adequate remedy at law, unless Defendants' infringement is enjoined by this Court.

70. Defendants have committed and continue to commit acts of infringement that Defendants knew or should have known constituted an unjustifiably high risk of infringement of at least one valid and enforceable claim of the '332 Patent. Defendants' direct and indirect infringement of the '332 Patent has been and continue to be willful, intentional, deliberate, and/or in conscious disregard of rights under the patent. Plaintiffs are entitled to an award of treble damages, reasonable attorney fees, and costs in bringing this action.

COUNT IV
(Infringement of the '832 Patent)

71. Paragraphs 1 through 22 are incorporated by reference as if fully set forth herein.

72. Plaintiffs have not licensed or otherwise authorized Defendants to make, use, offer for sale, sell, or import any products that embody the inventions of the '832 Patent.

73. Defendants have directly infringed the '832 Patent, either literally or under the doctrine of equivalents, without authority and in violation of 35 U.S.C. § 271, by making, using, offering to sell, selling, and/or importing into the United States products that satisfy each and every limitation of one or more claims of the '832 Patent. Such products include Apache Kafka Streams using RocksDB Universal Compaction, RocksDB Universal Compaction, Apache Cassandra, and Apache HBase.

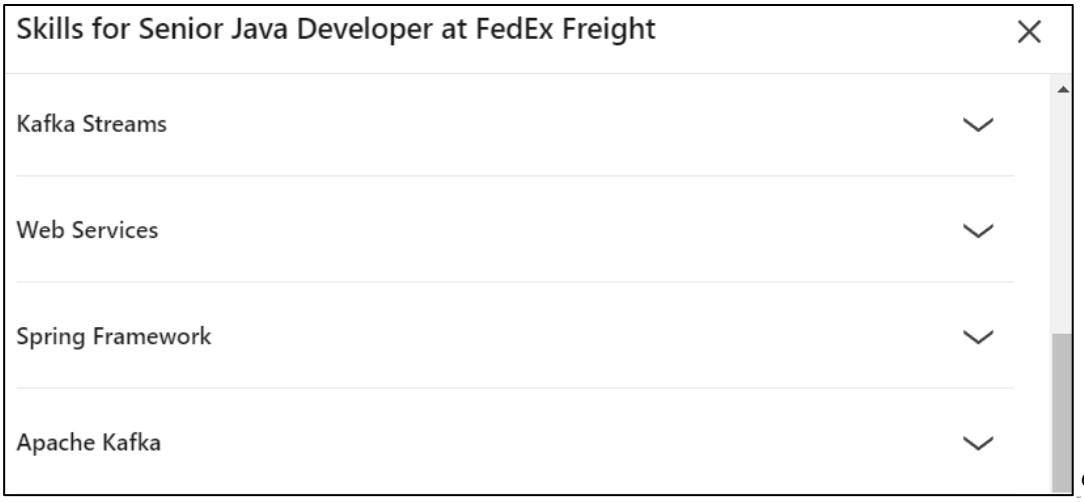
74. For example, Defendants have directly infringed at least claim 25 of the '832 Patent by making, using, offering to sell, selling, and/or importing into the United States products that include Apache Kafka Streams using RocksDB Universal Compaction, RocksDB Universal Style Compaction, Apache Cassandra, and Apache HBase.

IPR2026-00060 EX2010 044

Apache Kafka Streams using RocksDB Universal Compaction

75. On information and belief, FedEx has been using Apache Kafka Streams using RocksDB Universal Compaction in the U.S. since at least 2020. The default state store type of Apache Kafka Streams is RocksDB. Apache Kafka Streams also uses universal compaction by default for its state stores.



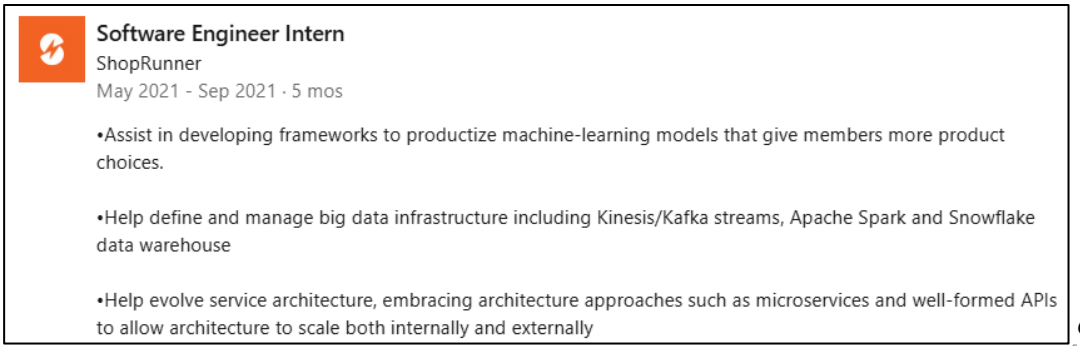


[67]



[68]

[67] https://www.linkedin.com/in/e-marie-i-m-80b41b264/
[68] https://newsroom.fedex.com/newsroom/global/fedex-completes-acquisition-of-shoprunner-expanding-its-e-commerce-capabilities



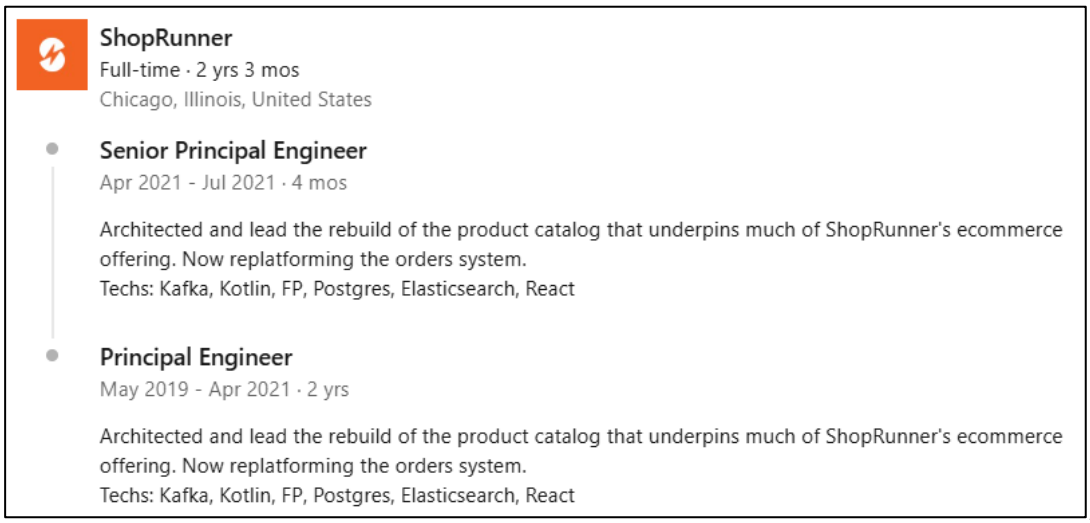
Software Engineer Intern
ShopRunner
May 2021 - Sep 2021 · 5 mos

•Assist in developing frameworks to productize machine-learning models that give members more product choices.

•Help define and manage big data infrastructure including Kinesis/Kafka streams, Apache Spark and Snowflake data warehouse

•Help evolve service architecture, embracing architecture approaches such as microservices and well-formed APIs to allow architecture to scale both internally and externally

[69]



ShopRunner
Full-time · 2 yrs 3 mos
Chicago, Illinois, United States

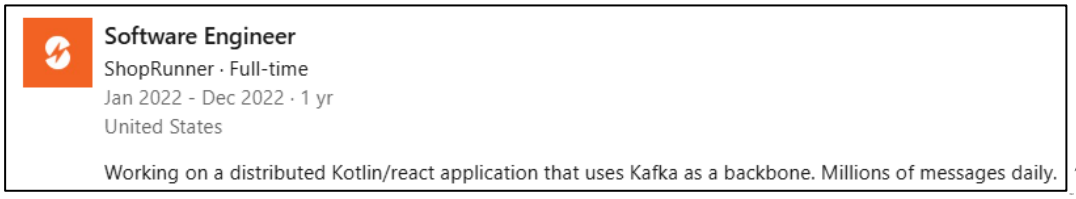
Senior Principal Engineer
Apr 2021 - Jul 2021 · 4 mos

Architected and lead the rebuild of the product catalog that underpins much of ShopRunner's ecommerce offering. Now replatforming the orders system.
Techs: Kafka, Kotlin, FP, Postgres, Elasticsearch, React

Principal Engineer
May 2019 - Apr 2021 · 2 yrs

Architected and lead the rebuild of the product catalog that underpins much of ShopRunner's ecommerce offering. Now replatforming the orders system.
Techs: Kafka, Kotlin, FP, Postgres, Elasticsearch, React

[70]



Software Engineer
ShopRunner · Full-time
Jan 2022 - Dec 2022 · 1 yr
United States

Working on a distributed Kotlin/react application that uses Kafka as a backbone. Millions of messages daily.

[71]

[69] https://www.linkedin.com/in/cristian-baeza-01/
[70] https://www.linkedin.com/in/sksamuel/
[71] https://www.linkedin.com/in/corlaez/

Software Engineer
ShopRunner · Full-time
Jun 2019 - Mar 2022 · 2 yrs 10 mos
Kraków Area, Poland

Part of the team building Marketplace as a Service (MaaS) that connect brands to online shopping destinations.

- Creating multiple solutions for products publishing, both batch and streaming (Kafka, Pyspark, Cadence workflows)
- Building internal tools that are supporting operational tasks for the product (React, typescript)
- Maintaining order processing applications (Kotlin)

Apart from Engineering work I've also had an opportunity to participate in Product Manager's responsibilities:

- Gathering, evaluating and analyzing business and user requirements
- Cooperating on a daily basis with the other team members to help set up the vision and the strategy of the product
- Defining the product scope, release phases and planning the roadmap
- Facilitating Scrum ceremonies

[72]

Kafka: Exploring the Enigmatic Realms of Existence

Which domains use Kafka

Apache Kafka is being widely used across various industries and domains due to its *robust features and real-time streaming capabilities.*

8. Transportation and Logistics:

- Real-time tracking and management of vehicles and shipments.
- Supply chain visibility and optimization.
- Traffic monitoring and route optimization.
- Companies — Uber, FedEx, DHL

[73]

[72] https://www.linkedin.com/in/renata-gargas/
[73] https://sayantansamanta098.medium.com/kafka-exploring-the-enigmatic-realms-of-existence-a9d102ae2ff

Unlocking Industry Success: The Power of Apache Kafka for Real-time Data Solutions

8. **Transportation and Logistics:**

- **FedEx**: Relies on Kafka for real-time package tracking, route optimization, and supply chain visibility in logistics operations.

[74]

FedEx Corporation Technology Stack

Find prospects by the technologies they use. Use this section to learn more about key technologies and tools, tracked by 6sense, used by FedEx Corporation

Apache Kafka

Apache Kafka is an open-source message broker project developed by the Apache Software Fou...

[75]

Latangi Amitabh • Memphis, TN

Sr. Data Engineer

FedEx, 2022 - Present

PROFESSIONAL EXPERIENCE:

Sr. Data Engineer
FedEx, Memphis, TN **Oct 2022 – Present**

Responsibilities:

- Spearheading projects within the GCP platform to govern data assets, including ingestion, transformation, and pipelining processes.
- Implemented Kafka for real-time data streaming and messaging within big data projects, ensuring reliable and scalable data ingestion from various sources.
- Configured and managed Kafka clusters for real-time data streaming and messaging, enabling seamless communication between distributed systems.
- Proficient in utilizing Google Cloud Dataproc for managing and scaling Apache Spark and Hadoop clusters, facilitating large-scale data processing and analytics.

[76]

[74] https://www.linkedin.com/pulse/mastering-apache-kafka-empowering-industries-data-mohammad-fa-alfard-heonf/
[75] https://6sense.com/company/fedex-corporation/5c3affffd55ae49f1b75e38d
[76] resumes.indeed.com – Smart Search (Login Required).

default.dsl.store

The default state store type used by DSL operators.

Type: string
Default: rocksDB
Valid Values: [rocksDB, in_memory]
Importance: low

[77]

Use level compaction

Kafka Streams uses universal compaction by default for its state stores. However, level compaction has lower space amplification than universal compaction at the cost of lower write rates. The corresponding option that you can tune in your `RocksDBConfigSetter` implementation is `options.setCompactionStyle(CompactionStyle.LEVEL)`.

[78]

76. Apache Kafka Streams using RocksDB Universal Compaction performs a method for monitoring performance of a storage device. For example, Apache Kafka Streams using RocksDB Universal Compaction uses a method of optimizing read and write rates to allow for monitoring the performance of a storage device.

Optimized for Fast Storage

RocksDB is optimized for fast, low latency storage such as flash drives and high-speed disk drives. RocksDB exploits the full potential of high read/write rates offered by flash or RAM.

RocksDB provides several APIs to read KV pairs from a database, including Get and MultiGet for point lookups and Iterator for sequential scanning. These APIs may result in RocksDB reading blocks from SST files on disk storage. The types of blocks and the frequency with which they are read from storage is workload dependent. Some workloads may have a small working set and thus may be able to cache most of the data required, while others may have large working sets and have to read from disk more often. In the latter case, the latency would be much higher and throughput would be lower than the former. They would also be dependent on the characteristics of the underlying storage media, making it difficult to migrate from one medium to another, for example, local flash to disaggregated flash.

[79]

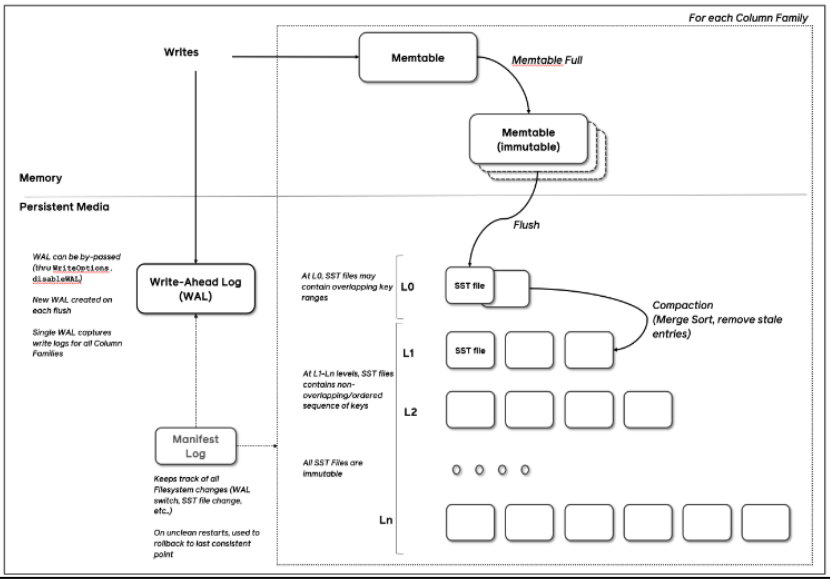
[77] https://kafka.apache.org/documentation/#introduction
[78] https://www.confluent.io/blog/how-to-tune-rocksdb-kafka-streams-state-stores-performance/
[79] https://rocksdb.org/blog/

3. High Level Architecture

RocksDB is a storage engine library of key-value store interface where keys and values are arbitrary byte streams. RocksDB organizes all data in sorted order and the common operations are `Get(key)`, `NewIterator()`, `Put(key, val)`, `Delete(key)`, and `SingleDelete(key)`.

The three basic constructs of RocksDB are **memtable**, **sstfile** and **logfile**. The memtable is an in-memory data structure - new writes are inserted into the *memtable* and are optionally written to the **logfile** (aka. Write Ahead Log(WAL)). The logfile is a sequentially-written file on storage. When the memtable fills up, it is flushed to a **sstfile** on storage and the corresponding logfile can be safely deleted. The data in an sstfile is sorted to facilitate easy lookup of keys.

The default format of sstfile is described in more details here.



[80]

Compaction Styles

Both Level Style Compaction and Universal Style Compaction store data in a fixed number of logical *levels* in the database. More recent data is stored in Level-0 (L0) and older data in higher-numbered levels, up to Lmax. Files in L0 may have overlapping keys, but files in other levels generally form a single sorted run per level.

Level Style Compaction (default) typically optimizes disk footprint vs. logical database size (space amplification) by minimizing the files involved in each compaction step: merging one file in Ln with all its overlapping files in Ln+1 and replacing them with new files in Ln+1.

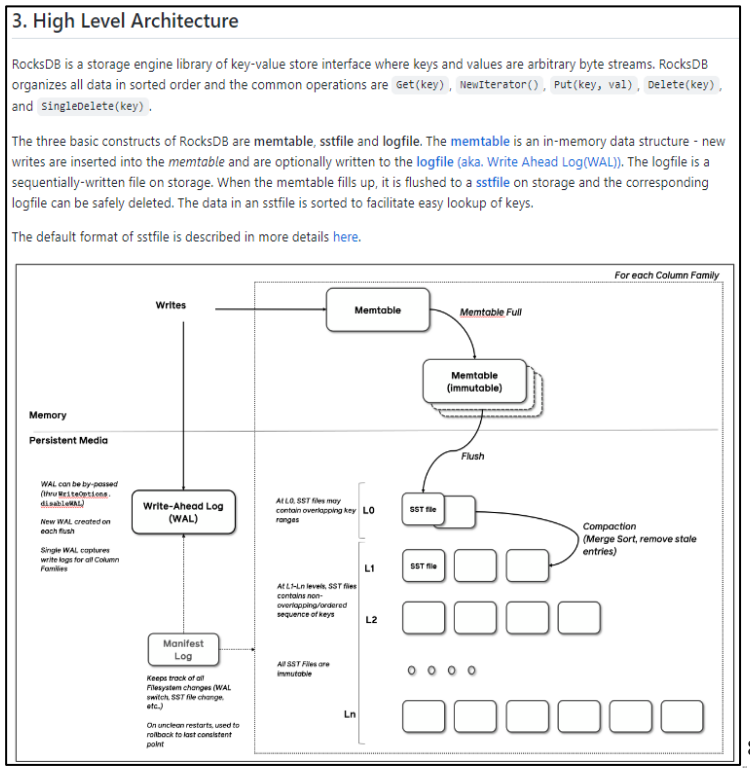
Universal Style Compaction typically optimizes total bytes written to disk vs. logical database size (write amplification) by merging potentially many files and levels at once, requiring more temporary space. Universal typically results in lower write-amplification but higher space- and read-amplification than Level Style Compaction.

[81]

77. Apache Kafka Streams using RocksDB Universal Compaction performs the step of

[80] https://github.com/facebook/rocksdb/wiki/RocksDB-Overview
[81] https://github.com/facebook/rocksdb/wiki/RocksDB-Overview

intercepting communications between a computer system and said storage device. For example, Apache Kafka Streams using RocksDB Universal Compaction, in response to previously captured performance captured by intercepting communications, inserts new writes into a memtable which are optionally written to the logfile.



[82]

78. Apache Kafka Streams using RocksDB Universal Compaction performs the step of analyzing said intercepted communications. For example, when the memtable fills up, it is flushed to a sstfile in storage and sorted to facilitate easy lookup of keys. For example, when certain conditions are met, such as the age of data, space amplification, the number of sorted runs while respecting size_ratio, and the number of sorted runs without respecting size_ratio, compaction is triggered and scheduled.

[82] https://github.com/facebook/rocksdb/wiki/RocksDB-Overview

3. High Level Architecture

RocksDB is a storage engine library of key-value store interface where keys and values are arbitrary byte streams. RocksDB organizes all data in sorted order and the common operations are `Get(key)`, `NewIterator()`, `Put(key, val)`, `Delete(key)`, and `SingleDelete(key)`.

The three basic constructs of RocksDB are **memtable**, **sstfile** and **logfile**. The **memtable** is an in-memory data structure - new writes are inserted into the *memtable* and are optionally written to the **logfile** (aka. Write Ahead Log(WAL)). The logfile is a sequentially-written file on storage. When the memtable fills up, it is flushed to a **sstfile** on storage and the corresponding logfile can be safely deleted. The data in an sstfile is sorted to facilitate easy lookup of keys.[83]

Precondition: n >= options.level0_file_num_compaction_trigger

Unless number of sorted runs reaches this threshold, no compaction will be triggered at all.

If the pre-condition is satisfied, there are four conditions. Each of them can trigger a compaction. They are evaluated in order and the selection stops once a compaction has been scheduled. Before going into detail, the overview for picking a compaction is:

1. Try to schedule a compaction by age of data
2. Else if the space amplification constraint has been violated, try to schedule a major compaction
3. Else try to schedule a compaction (minor or major) while respecting size_ratio
4. Else try to schedule a minor compaction without respecting size_ratio[84]

1. Compaction triggered by age of data

For universal style compaction, the aging-based triggering criterion has the highest priority since it is a hard requirement. When trying to pick a compaction, this condition is checked first.

If the condition meets (there are files older than options.periodic_compaction_seconds), then RocksDB proceeds to pick sorted runs for compaction. RocksDB picks sorted runs from oldest to youngest until encountering a sorted run that is being compacted by another compaction. These files will be compacted to bottommost level unless bottommost level is reserved for ingestion behind. In this case, files will be compacted to second bottommost level.

2. Compaction Triggered by Space Amplification

If the estimated *size amplification ratio* is larger than options.compaction_options_universal.max_size_amplification_percent / 100, all files will be compacted to one sorted run.

3. Compaction Triggered by number of sorted runs while respecting size_ratio

If a compaction is not triggered by the space amplification constraint as described above then a compaction can be triggered by the number of sorted runs. This step respects *size ratio trigger* when selecting the sorted runs to compact.

We calculated a value of *size ratio trigger* as

```
size_ratio_trigger = (100 + options.compaction_options_universal.size_ratio) / 100
```

[83] https://github.com/facebook/rocksdb/wiki/Universal-Compaction
[84] https://github.com/facebook/rocksdb/wiki/Terminology

4. Compaction Triggered by number of sorted runs without respecting size_ratio

If none of the previous conditions cause a compaction to be scheduled, then this condition might schedule one. This step does not respect *size ratio trigger* but the logic is otherwise similar to what was described above for condition 3. If we need to compact more than options.compaction_options_universal.max_merge_width number of sorted runs, we cap it to options.compaction_options_universal.max_merge_width.[85]

79. Apache Kafka Streams using RocksDB Universal Compaction performs the step of reallocating at least some of said data on said storage device to enhance the performance of said storage device based on said analyzed communications. For example, Apache Kafka Streams using RocksDB Universal Compaction will trigger and schedule compactions based on various conditions, which efficiently allocate space.

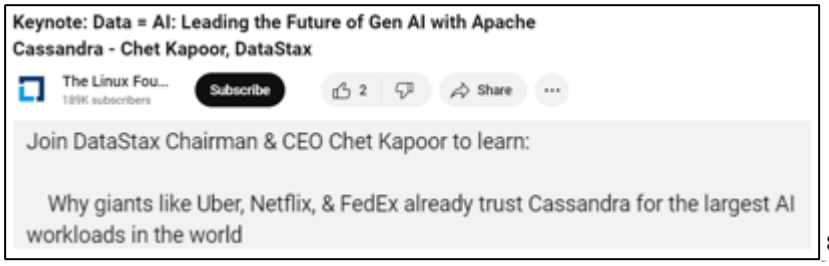
In the presence of ongoing writes, compactions are needed for space efficiency, read (query) efficiency, and timely data deletion. Compaction removes key-value bindings that have been deleted or overwritten, and re-organizes data for query efficiency. Compactions may occur in multiple threads if configured.

The entire database is stored in a set of *sstfiles*. When a *memtable* is full, its content is written out to a file in Level-0 (L0) of the LSM tree. RocksDB removes duplicate and overwritten keys in the memtable when it is flushed to a file in L0. In *compaction*, some files are periodically read in and merged to form larger files, often going into the next LSM level (such as L1, up to Lmax).

The overall write throughput of an LSM database directly depends on the speed at which compactions can occur, especially when the data is stored in fast storage like SSD or RAM. RocksDB may be configured to issue concurrent compaction requests from multiple threads. It is observed that sustained write rates may increase by as much as a factor of 10 with multi-threaded compaction when the database is on SSDs, as compared to single-threaded compactions.[86]

Apache Cassandra

80. On information and belief, FedEx has used Apache Cassandra in the U.S. since at least 2018.



[87]

[85] https://github.com/facebook/rocksdb/wiki/Universal-Compaction
[86] https://github.com/facebook/rocksdb/wiki/RocksDB-Overview
[87] https://www.youtube.com/watch?v=flaFAjXXuK8

Why Ryzeo Relies on Real-Time Data for E-Commerce Behavioral Marketing | DataStax

DataStax
11.8K subscribers
Subscribe

About DataStax:
DataStax is the real-time data company. With DataStax, any enterprise can mobilize real-time data and quickly build the smart, high-scale applications required to become a true data-driven business. DataStax Astra DB uniquely combines the power of Apache Cassandra™—the world's most scalable database—with the advanced Apache Pulsar™ streaming technology in a unified stack delivered as a service on any cloud. Hundreds of the world's leading enterprises, including Verizon, Audi, FedEx, ESL Gaming and many more rely on DataStax to unleash the power of real-time data to deliver the in-the-moment digital experiences that can win new markets and change industries. Learn more at DataStax.com.

[88]

Real-time machine learning:

"Uber uses Cassandra, Netflix uses Cassandra, FedEx uses Cassandra because it can handle limitless scale and all these events you need to store, and that's what ML needs. These customers have very large amounts of data on the internet but they need some tools to make it usable within the ML algorithm. We looked around and that's what Kaskada specialises in. It extracts the most relevant data so you only compute what's necessary, then it maps it very efficiently and that reduces your cost significantly," he tells *The Stack*.

[89]

FedEx Corporation Tech Stack

FedEx Corporation uses 8 technology products and services including Hyperion, Apache Cassandra, Artifactory, and more. Explore FedEx Corporation's tech stack below.

Hyperion
Corporate Performance Management

Apache Cassandra
Database

[90]

[88] https://www.youtube.com/watch?v=6B6EuTFAMIg
[89] https://www.thestack.technology/real-time-machine-learning-datastax-kaskada//
[90] https://leadiq.com/c/fedex-corporation/636463d1d4454f2aa3835a80

Shailaja vemula · 3rd
Looking for Full stack developer roles, available to work only on C2C
Orlando, Florida, United States · Contact info

Experience

Full-stack Developer
Contract
Jul 2020 - Present · 4 yrs 3 mos

Developing the User Interface Screens for presentation using Bootstrap, J Query, AJAX, JSP, HTML, CSS, and Java Script created multi-module Maven project to integrate Spring 5.0, Rest API, Cassandra & micro services and Jenkins for continuous integration.
Moving existing Angular JS controllers and services to Angular7 components for more responsive and less dependency developed RESTAPI's using Spring MVC and spring boot, hosted all Micro services on Pivotal Cloud Foundry (PCF)

[91]

Lucas Oppersdorff • Arlington, TX

Tech Lead/Developer

FedEx, 2021 - Present

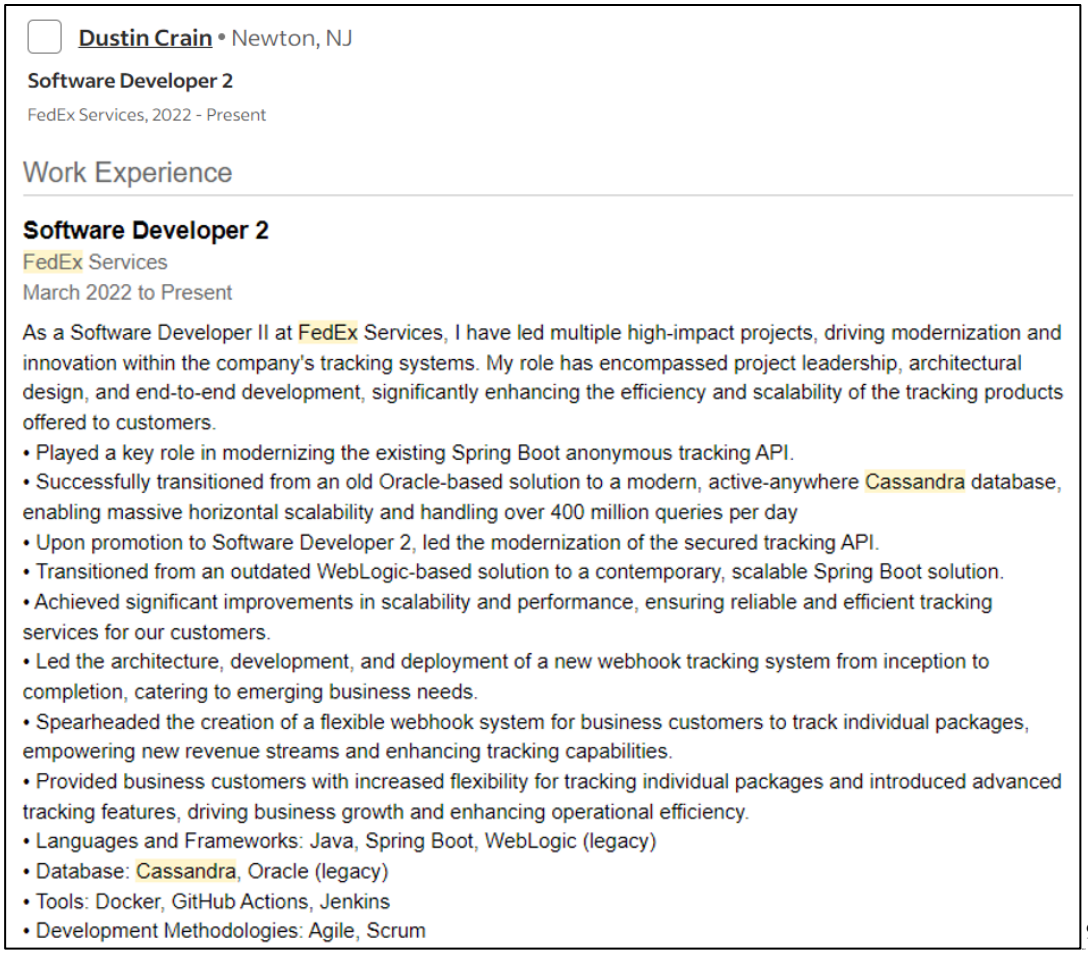
Work Experience

Tech Lead/Developer
FedEx - Remote
August 2021 to Present

- (Contract) | Python, Java, SQL, NoSQL, Azure
- Lead engineer on Operations/Planning/Logistics team on last mile delivery project
- Designed and implemented Ingestion, Estimation, Training and Prediction systems
- Implemented ETA and Battery Capacity training and prediction services
- Built Planning service for estimation
- Built Itinerary scheduling service and alt flow Api's
- Azure and Open Source components, AKS, Azure Functions, Azure ServiceBus
- Kubernetes, Jenkins and Azure DevOps CI/CD
- Additional technologies: Cassandra, Redis, RabbitMQ

[92]

[91] https://www.linkedin.com/in/shailaja-vemula-a7770526a/
[92] resumes.indeed.com – Smart Search (Login Required).



Dustin Crain • Newton, NJ

Software Developer 2

FedEx Services, 2022 - Present

Work Experience

Software Developer 2

FedEx Services

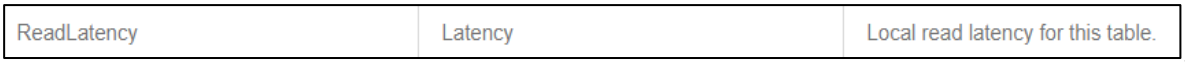
March 2022 to Present

As a Software Developer II at FedEx Services, I have led multiple high-impact projects, driving modernization and innovation within the company's tracking systems. My role has encompassed project leadership, architectural design, and end-to-end development, significantly enhancing the efficiency and scalability of the tracking products offered to customers.

- Played a key role in modernizing the existing Spring Boot anonymous tracking API.
- Successfully transitioned from an old Oracle-based solution to a modern, active-anywhere Cassandra database, enabling massive horizontal scalability and handling over 400 million queries per day
- Upon promotion to Software Developer 2, led the modernization of the secured tracking API.
- Transitioned from an outdated WebLogic-based solution to a contemporary, scalable Spring Boot solution.
- Achieved significant improvements in scalability and performance, ensuring reliable and efficient tracking services for our customers.
- Led the architecture, development, and deployment of a new webhook tracking system from inception to completion, catering to emerging business needs.
- Spearheaded the creation of a flexible webhook system for business customers to track individual packages, empowering new revenue streams and enhancing tracking capabilities.
- Provided business customers with increased flexibility for tracking individual packages and introduced advanced tracking features, driving business growth and enhancing operational efficiency.
- Languages and Frameworks: Java, Spring Boot, WebLogic (legacy)
- Database: Cassandra, Oracle (legacy)
- Tools: Docker, GitHub Actions, Jenkins
- Development Methodologies: Agile, Scrum

[93]

81. Apache Cassandra performs a method for monitoring performance of a storage device. For example, Apache Cassandra monitors performance metrics, such as through "ReadLatency." By way of further example, Apache Cassandra performs compaction, which results in reclaiming disk space.



ReadLatency	Latency	Local read latency for this table.

[94]

[93] resumes.indeed.com – Smart Search (Login Required).

[94] https://cassandra.apache.org/doc/latest/cassandra/operating/metrics.html

Cassandra Compaction is a process of reconciling various copies of data spread across distinct SSTables. Cassandra performs compaction of SSTables as a background activity. Cassandra has to maintain fewer SSTables and fewer copies of each data row due to compactions improving its read performance. Compaction is a crucial area of Cassandra performance and maintenance.

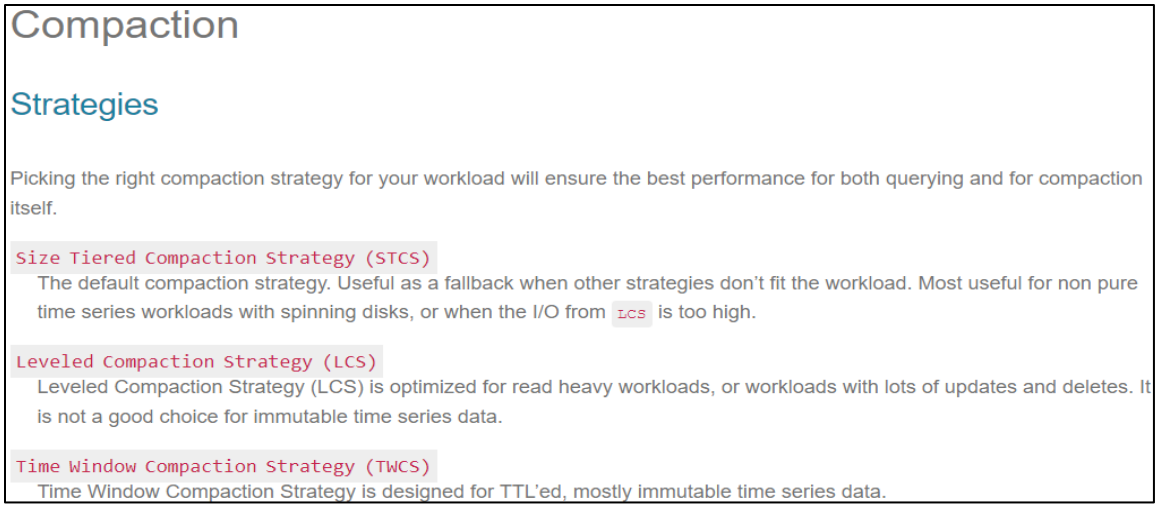
Cassandra's Write Path

To understand the importance of compactions in Cassandra, you must first understand how Cassandra writes data to a disk. The Cassandra write path in a nutshell:

1. Cassandra stores recent writes in memory (in a structure called the Memtable).
2. When enough writes have been made, Cassandra flushes the Memtable to disk. Data on disk is stored in relatively simple data structures called Sorted String Tables (SSTable). At the most simplified level, an SSTable could be described as a sorted array of strings.
3. Before writing a new SSTable, Cassandra merges and pre-sorts the data in the Memtable according to Primary Key. In Cassandra, a Primary Key consists of a Partition Key (the unique key that determines which node the data is stored on) and any Clustering Keys that have been defined.
4. The SSTable is written to disk as a single contiguous write operation. SStables are immutable. Once they are written to disk they are not modified. Any updates to data or deletion of data within an SSTable is written to a new SSTable. If data is updated regularly, Cassandra may need to read from multiple SSTables to retrieve a single row.
5. Compaction operations occur periodically to re-write and combine SSTables. This is required because SSTables are immutable (no modifications once written to disk). Compactions prune deleted data and merge disparate row data into new SSTables in order to reclaim disk space and keep read operations optimized.

If you are unfamiliar with Cassandra's write path, please read **The write path to compaction** from Cassandra Wiki.[95]

[95] https://www.instaclustr.com/blog/apache-cassandra-compaction/



Compaction

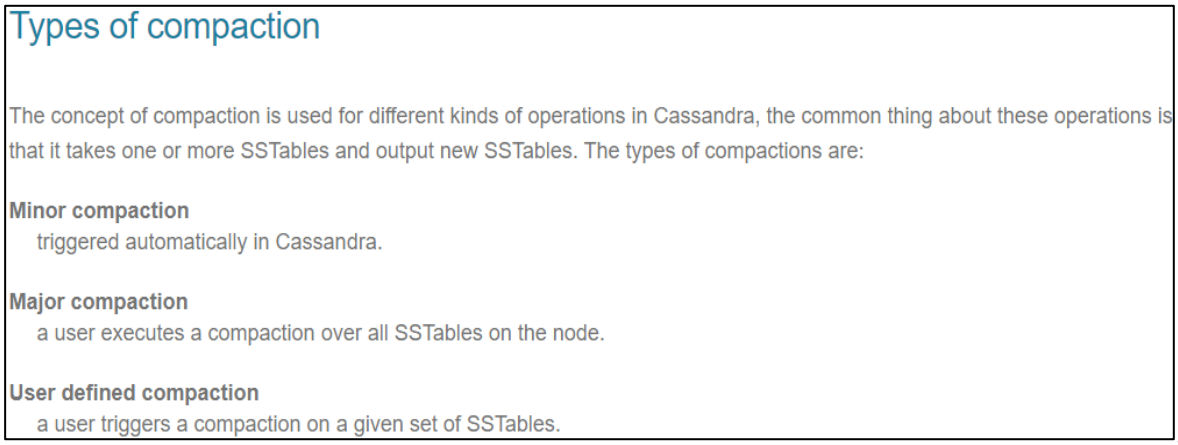
Strategies

Picking the right compaction strategy for your workload will ensure the best performance for both querying and for compaction itself.

`Size Tiered Compaction Strategy (STCS)`
The default compaction strategy. Useful as a fallback when other strategies don't fit the workload. Most useful for non pure time series workloads with spinning disks, or when the I/O from `LCS` is too high.

`Leveled Compaction Strategy (LCS)`
Leveled Compaction Strategy (LCS) is optimized for read heavy workloads, or workloads with lots of updates and deletes. It is not a good choice for immutable time series data.

`Time Window Compaction Strategy (TWCS)`
Time Window Compaction Strategy is designed for TTL'ed, mostly immutable time series data.



Types of compaction

The concept of compaction is used for different kinds of operations in Cassandra, the common thing about these operations is that it takes one or more SSTables and output new SSTables. The types of compactions are:

Minor compaction
triggered automatically in Cassandra.

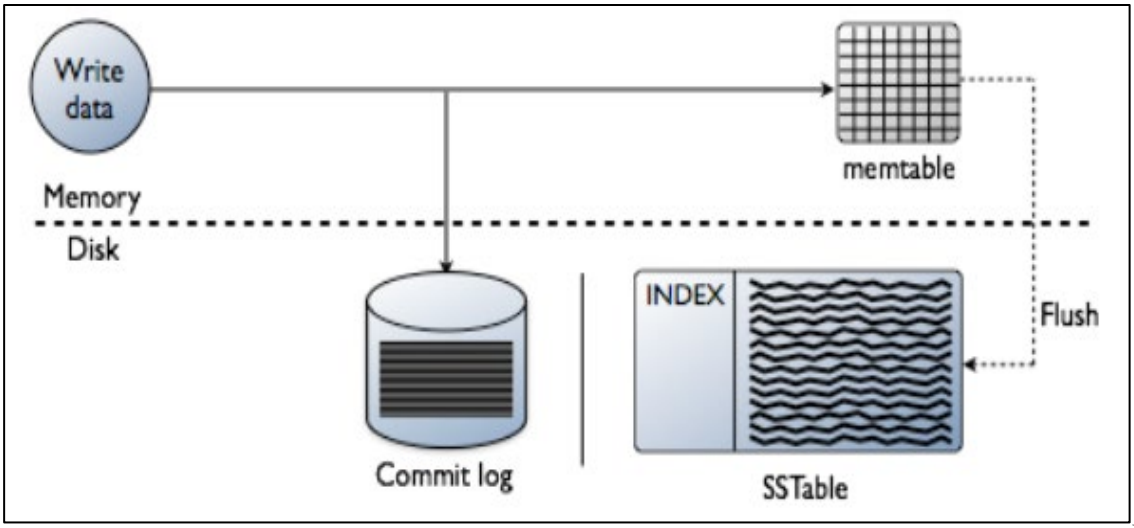
Major compaction
a user executes a compaction over all SSTables on the node.

User defined compaction
a user triggers a compaction on a given set of SSTables.

[96]

82. Apache Cassandra performs the step of intercepting communications between a computer system (e.g., a Cassandra node) and said storage device (e.g., a disk associated with a Cassandra node).

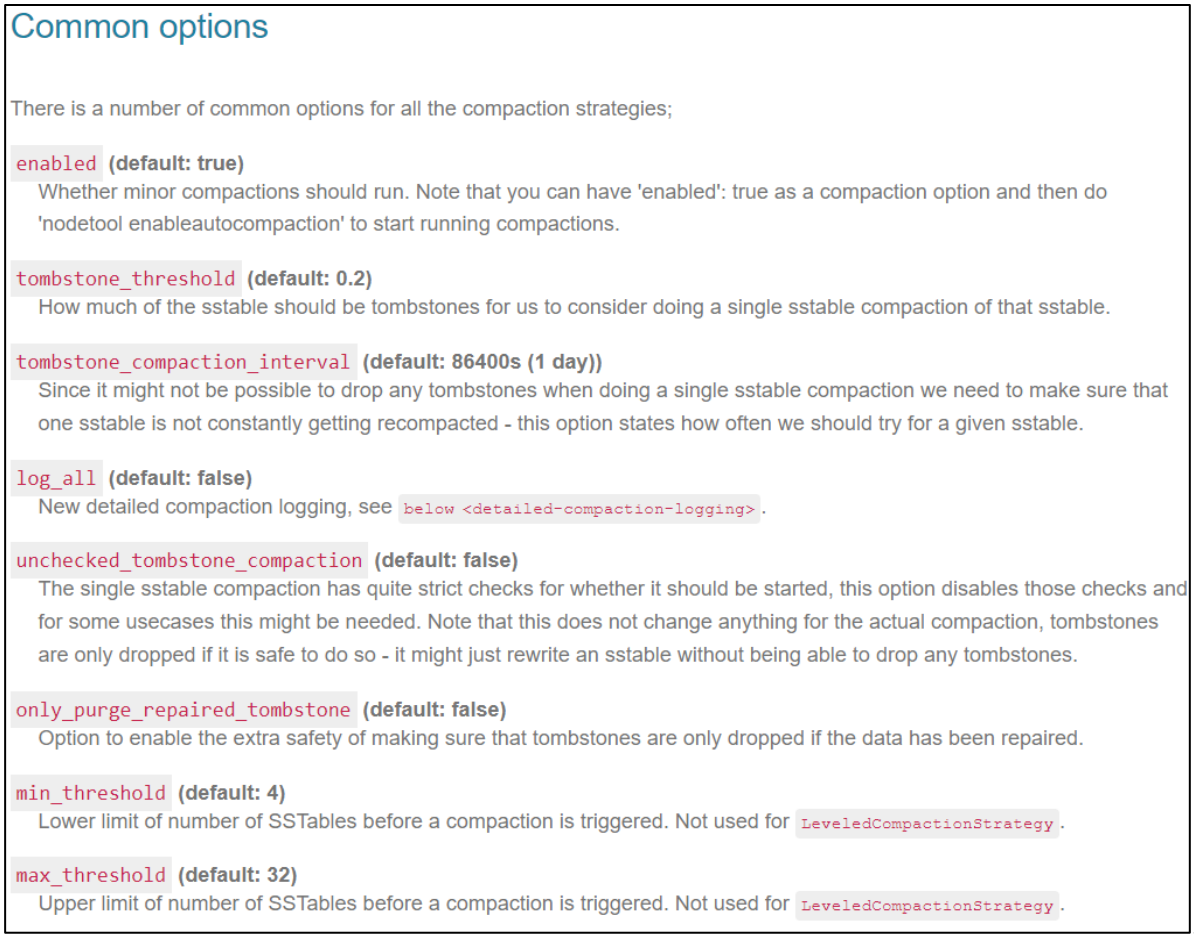
[96] https://cassandra.apache.org/doc/latest/cassandra/operating/compaction/



[97]

83. Apache Cassandra performs the step of analyzing said intercepted communications. For example, the step of analyzing can entail monitoring the memtable and tracking the number of SSTables flushed to a disk and comparing it to the parameters min_threshold and max_threshold to determine whether to perform a minor compaction.

[97] https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/dml/dmlHowDataWritten.html



Common options

There is a number of common options for all the compaction strategies;

`enabled` **(default: true)**
Whether minor compactions should run. Note that you can have 'enabled': true as a compaction option and then do 'nodetool enableautocompaction' to start running compactions.

`tombstone_threshold` **(default: 0.2)**
How much of the sstable should be tombstones for us to consider doing a single sstable compaction of that sstable.

`tombstone_compaction_interval` **(default: 86400s (1 day))**
Since it might not be possible to drop any tombstones when doing a single sstable compaction we need to make sure that one sstable is not constantly getting recompacted - this option states how often we should try for a given sstable.

`log_all` **(default: false)**
New detailed compaction logging, see `below <detailed-compaction-logging>`.

`unchecked_tombstone_compaction` **(default: false)**
The single sstable compaction has quite strict checks for whether it should be started, this option disables those checks and for some usecases this might be needed. Note that this does not change anything for the actual compaction, tombstones are only dropped if it is safe to do so - it might just rewrite an sstable without being able to drop any tombstones.

`only_purge_repaired_tombstone` **(default: false)**
Option to enable the extra safety of making sure that tombstones are only dropped if the data has been repaired.

`min_threshold` **(default: 4)**
Lower limit of number of SSTables before a compaction is triggered. Not used for `LeveledCompactionStrategy`.

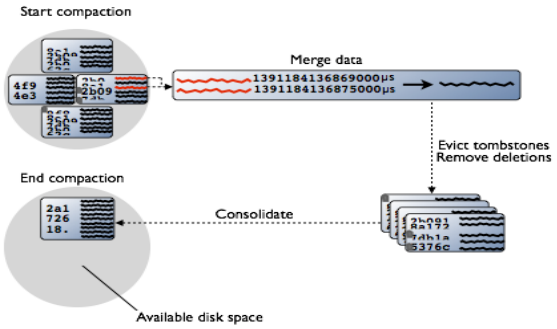
`max_threshold` **(default: 32)**
Upper limit of number of SSTables before a compaction is triggered. Not used for `LeveledCompactionStrategy`.

[98]

84. Apache Cassandra performs the step of reallocating at least some of said data on said storage device to enhance the performance of said storage device based on said analyzed communications. For example, the step of reallocating can occur during minor compaction and entails merging SSTables in response to analysis of previous performance. Continuing with the example, the step of relocating enhances performance of the storage device by: (1) reducing access ("read") time; and (2) making additional disk space available for reuse.

[98] https://cassandra.apache.org/doc/latest/cassandra/operating/compaction/

During compaction, there is a temporary spike in disk space usage and disk I/O because the old and new SSTables co-exist. This diagram depicts the compaction process:



Compaction merges the data in each SSTable by partition key, selecting the latest data for storage based on its timestamp. Cassandra can merge the data performantly, without random IO, because rows are sorted by partition key within each SSTable. After evicting tombstones and removing deleted data, columns, and rows, the compaction process consolidates SSTables into a single file. The old SSTable files are deleted as soon as any pending reads finish using the files. Disk space occupied by old SSTables becomes available for reuse.

Cassandra 2.1 improves read performance after compaction by performing an incremental replacement of compacted SSTables. Instead of waiting for the entire compaction to finish and then throwing away the old SSTable (and cache), Cassandra can read data directly from the new SSTable even before it finishes writing.

As data is written to the new SSTable and reads are directed to it, the corresponding data in the old SSTables is no longer accessed and is evicted from the page cache. Thus begins an incremental process of caching the new SSTable, while directing reads away from the old one. The dramatic cache miss is gone. Cassandra provides predictable high performance even under heavy load.

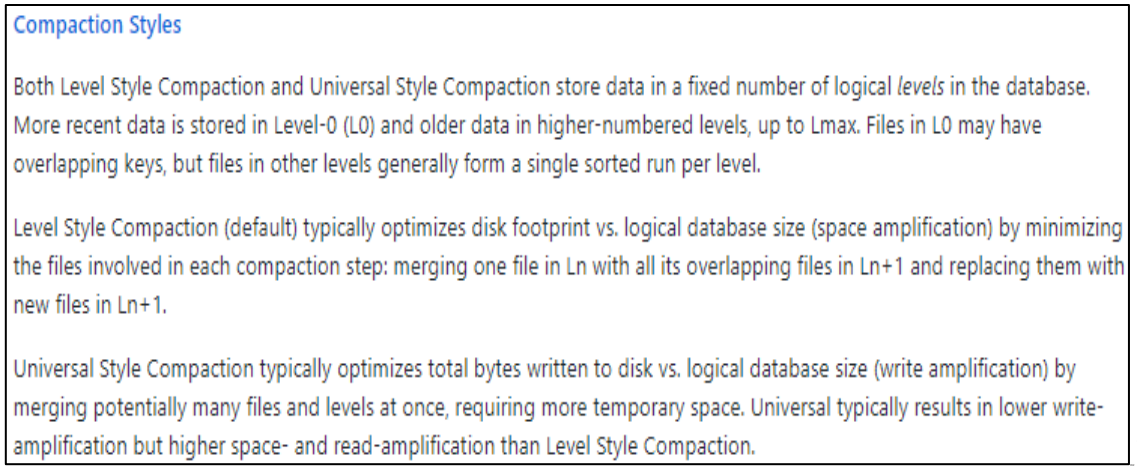
[99]

RocksDB Universal Style Compaction

85. On information and belief, FedEx has used RocksDB Universal Style Compaction in the U.S. since at least 2020, as set forth in Paragraph 75.

86. RocksDB Universal Style Compaction performs the method of monitoring performance of a storage device. For example, RocksDB uses a process called compaction. Two compaction style available with RocksDB are Level Style Compaction and Universal Style Compaction ("USC").

[99] https://docs.datastax.com/en/cassandra-oss/2.1/cassandra/dml/dml_write_path_c.html



Compaction Styles

Both Level Style Compaction and Universal Style Compaction store data in a fixed number of logical *levels* in the database. More recent data is stored in Level-0 (L0) and older data in higher-numbered levels, up to Lmax. Files in L0 may have overlapping keys, but files in other levels generally form a single sorted run per level.

Level Style Compaction (default) typically optimizes disk footprint vs. logical database size (space amplification) by minimizing the files involved in each compaction step: merging one file in Ln with all its overlapping files in Ln+1 and replacing them with new files in Ln+1.

Universal Style Compaction typically optimizes total bytes written to disk vs. logical database size (write amplification) by merging potentially many files and levels at once, requiring more temporary space. Universal typically results in lower write-amplification but higher space- and read-amplification than Level Style Compaction.

[100]

87. RocksDB USC performs the step of intercepting communications between a computer system and said storage device. For example, a storage device can be a disk associated with a RocksDB node. The computer system is a client that functions, e.g., to send write requests that are to be stored on disk. When writing data using RocksDB, the step of intercepting communications between a computer (e.g., a RocksDB node) and a storage device (e.g., a disk associated with the RocksDB node) entails populating data into a memtable.

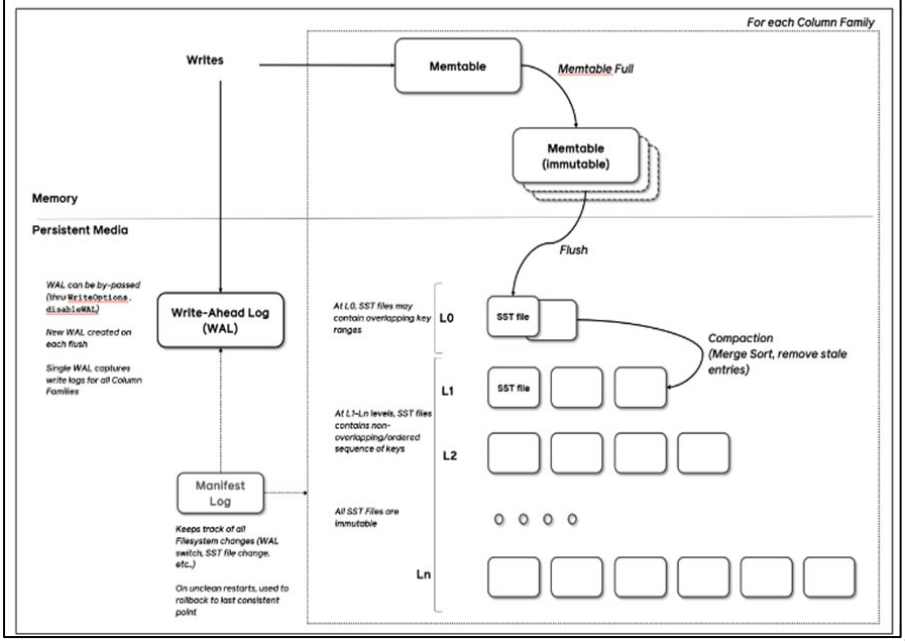
[100] https://github.com/facebook/rocksdb/wiki/RocksDB-Overview

3. High Level Architecture

RocksDB is a storage engine library of key-value store interface where keys and values are arbitrary byte streams. RocksDB organizes all data in sorted order and the common operations are `Get(key)`, `NewIterator()`, `Put(key, val)`, `Delete(key)`, and `SingleDelete(key)`.

The three basic constructs of RocksDB are **memtable**, **sstfile** and **logfile**. The **memtable** is an in-memory data structure - new writes are inserted into the *memtable* and are optionally written to the **logfile** (aka. Write Ahead Log(WAL)). The logfile is a sequentially-written file on storage. When the memtable fills up, it is flushed to a **sstfile** on storage and the corresponding logfile can be safely deleted. The data in an sstfile is sorted to facilitate easy lookup of keys.

The default format of sstfile is described in more details here.



[101]

88. RocksDB USC performs the step of analyzing said intercepted communications. For example, RocksDB can determine if the memtable size is large enough to be flushed.

3. High Level Architecture

RocksDB is a storage engine library of key-value store interface where keys and values are arbitrary byte streams. RocksDB organizes all data in sorted order and the common operations are `Get(key)`, `NewIterator()`, `Put(key, val)`, `Delete(key)`, and `SingleDelete(key)`.

The three basic constructs of RocksDB are **memtable**, **sstfile** and **logfile**. The **memtable** is an in-memory data structure - new writes are inserted into the *memtable* and are optionally written to the **logfile** (aka. Write Ahead Log(WAL)). The logfile is a sequentially-written file on storage. When the memtable fills up, it is flushed to a **sstfile** on storage and the corresponding logfile can be safely deleted. The data in an sstfile is sorted to facilitate easy lookup of keys.

[102]

By way of further example, unless the threshold specified by

[101] https://github.com/facebook/rocksdb/wiki/RocksDB-Overview
[102] https://github.com/facebook/rocksdb/wiki/RocksDB-Overview

options.level0_file_num_compaction+trigger, no compaction will be triggered at all.

Precondition: n >= options.level0_file_num_compaction_trigger

Unless number of sorted runs reaches this threshold, no compaction will be triggered at all. [103]

By way of further example, if the precondition is satisfied, then any one of four conditions can trigger a compaction. They are evaluated in order and the selection stops once a compaction has been scheduled. More specifically and as shown in the bottom screenshot, with the precondition satisfied, USC may be triggered by: (1) the age of data; (2) Space Amplification; (3) the number of sorted runs while respecting size_ratio; or (4) number of sorted runs without respecting size_ratio.

If the pre-condition is satisfied, there are four conditions. Each of them can trigger a compaction. They are evaluated in order and the selection stops once a compaction has been scheduled. Before going into detail, the overview for picking a compaction is:

1. Try to schedule a compaction by age of data
2. Else if the space amplification constraint has been violated, try to schedule a major compaction
3. Else try to schedule a compaction (minor or major) while respecting size_ratio
4. Else try to schedule a minor compaction without respecting size_ratio

[104]

1. Compaction triggered by age of data

For universal style compaction, the aging-based triggering criterion has the highest priority since it is a hard requirement. When trying to pick a compaction, this condition is checked first.

If the condition meets (there are files older than options.periodic_compaction_seconds), then RocksDB proceeds to pick sorted runs for compaction. RocksDB picks sorted runs from oldest to youngest until encountering a sorted run that is being compacted by another compaction. These files will be compacted to bottommost level unless bottommost level is reserved for ingestion behind. In this case, files will be compacted to second bottommost level.

2. Compaction Triggered by Space Amplification

If the estimated *size amplification ratio* is larger than options.compaction_options_universal.max_size_amplification_percent / 100, all files will be compacted to one sorted run.

3. Compaction Triggered by number of sorted runs while respecting size_ratio

If a compaction is not triggered by the space amplification constraint as described above then a compaction can be triggered by the number of sorted runs. This step respects *size ratio trigger* when selecting the sorted runs to compact.

We calculated a value of *size ratio trigger* as

```
size_ratio_trigger = (100 + options.compaction_options_universal.size_ratio) / 100
```

[103] https://github.com/facebook/rocksdb/wiki/universal-compaction
[104] https://github.com/facebook/rocksdb/wiki/universal-compaction

4. Compaction Triggered by number of sorted runs without respecting size_ratio

If none of the previous conditions cause a compaction to be scheduled, then this condition might schedule one. This step does not respect *size ratio trigger* but the logic is otherwise similar to what was described above for condition 3. If we need to compact more than options.compaction_options_universal.max_merge_width number of sorted runs, we cap it to options.compaction_options_universal.max_merge_width.[105]

89. RocksDB USC performs the step of reallocating at least some of said data on said storage device to enhance the performance of said storage device based on said analyzed communications. For example, reallocation occurs during USCs.

1. Compaction triggered by age of data

For universal style compaction, the aging-based triggering criterion has the highest priority since it is a hard requirement. When trying to pick a compaction, this condition is checked first.

If the condition meets (there are files older than options.periodic_compaction_seconds), then RocksDB proceeds to pick sorted runs for compaction. RocksDB picks sorted runs from oldest to youngest until encountering a sorted run that is being compacted by another compaction. These files will be compacted to bottommost level unless bottommost level is reserved for ingestion behind. In this case, files will be compacted to second bottommost level.

2. Compaction Triggered by Space Amplification

If the estimated *size amplification ratio* is larger than options.compaction_options_universal.max_size_amplification_percent / 100, all files will be compacted to one sorted run.

3. Compaction Triggered by number of sorted runs while respecting size_ratio

If a compaction is not triggered by the space amplification constraint as described above then a compaction can be triggered by the number of sorted runs. This step respects *size ratio trigger* when selecting the sorted runs to compact.

We calculated a value of *size ratio trigger* as

```
size_ratio_trigger = (100 + options.compaction_options_universal.size_ratio) / 100
```

[106]

90. Continuing the example, USCs enhance performance by, among other things, using space more efficiently and increasing write throughput.

4. Compaction Triggered by number of sorted runs without respecting size_ratio

If none of the previous conditions cause a compaction to be scheduled, then this condition might schedule one. This step does not respect *size ratio trigger* but the logic is otherwise similar to what was described above for condition 3. If we need to compact more than options.compaction_options_universal.max_merge_width number of sorted runs, we cap it to options.compaction_options_universal.max_merge_width.[107]

[105] https://github.com/facebook/rocksdb/wiki/Universal-Compaction
[106] https://github.com/facebook/rocksdb/wiki/Universal-Compaction
[107] https://github.com/facebook/rocksdb/wiki/Universal-Compaction

In the presence of ongoing writes, compactions are needed for space efficiency, read (query) efficiency, and timely data deletion. Compaction removes key-value bindings that have been deleted or overwritten, and re-organizes data for query efficiency. Compactions may occur in multiple threads if configured.

The entire database is stored in a set of *sstfiles*. When a *memtable* is full, its content is written out to a file in Level-0 (L0) of the LSM tree. RocksDB removes duplicate and overwritten keys in the memtable when it is flushed to a file in L0. In *compaction*, some files are periodically read in and merged to form larger files, often going into the next LSM level (such as L1, up to Lmax).

The overall write throughput of an LSM database directly depends on the speed at which compactions can occur, especially when the data is stored in fast storage like SSD or RAM. RocksDB may be configured to issue concurrent compaction requests from multiple threads. It is observed that sustained write rates may increase by as much as a factor of 10 with multi-threaded compaction when the database is on SSDs, as compared to single-threaded compactions.[108]

Apache HBase

91. On information and belief, FedEx has used Apache HBase is the U.S. since at least 2021.

Riley Hodge • Virginia Beach, VA

Sr. Big Data/ Scala Developer

FedEx, 2023 - Present

Work Experience

Sr. Big Data/ Scala Developer

FedEx - Memphis, TN

April 2023 to Present

Responsibilities:

· Responsible for building scalable distributed data solutions using Hadoop and migrate legacy Retail applications ETL to Hadoop.

· Wrote the Spark code in Scala to connect to HBASE and read/write data to the HBASE table.

· Extracted data from different databases and to copy into HDFS using Sqoop and have an expertise in using compression techniques to optimize the data storage.

· Implemented Flume, Spark framework for real time data processing. Developed simple to complex Map Reduce jobs using Hive and Pig for analyzing the data.

· Used different Serdes for converting JSON data into pipe separated data.

· Optimized Map/Reduce Jobs to use HDFS efficiently by using various compression mechanisms.

· Built analytics for structured and unstructured data and managing large data ingestion by using Avro, Flume, Thrift, Kafka and Sqoop.

· Worked on scalable distributed computing systems, software architecture, data structures and algorithms using Hadoop, Apache Sparkand Apache Storm etc.

· Ingested streaming data into Hadoop using Spark, Storm Framework and Scala.

· Developed Bankers Rounding UDF for Hive/Pig or Implemented Teradata Rounding in Hive/Pig.

· Continuously monitored and managed the Hadoop Cluster using ClouderaManager.

Environment: Hadoop, Map Reducer, Aws, S3, Redshift, HDFS, Hive, Pig, Spark, Storm, Flume, Thrift, Kafka, Sqoop, Oozie, Impala, SQL, Scala, Java (JDK 1.6), Hadoop (Cloudera) and Eclipse.[109]

[108] https://github.com/facebook/rocksdb/wiki/RocksDB-Overview

[109] resumes.indeed.com – Smart Search (Login Required)

FedEx Completes Acquisition of ShopRunner, Expanding its E-Commerce Capabilities

December 28, 2020 07:00 AM
3 min read

110

Big Data/Spark Engineer
ShopRunner · Contract
Dec 2018 - Mar 2020 · 1 yr 4 mos
Remote

Devised and documented a novel architecture and development process to shift the existing ETL pipeline into Hadoop-based systems, with a focus on scalability and maintainability.
Leverage Spark PySpark extensively to transform data for creating models to support the Data Science team in their analytics and modeling work.
Constructed Streaming data pipelines with Spark PySpark and HBase, enabling real-time processing of large data streams.
Engineered a reusable Spark-based framework for processing and flattening JSON data that multiple enterprise applications can employ.
Devised a generic framework for preparing data to be fed into machine learning models, ensuring the quality and consistency of the data input.

111

[110] https://newsroom.fedex.com/newsroom/global/fedex-completes-acquisition-of-shoprunner-expanding-its-e-commerce-capabilities
[111] https://www.linkedin.com/in/supratik-das-403a1519b/

92. Apache HBase performs the method for monitoring performance of a storage device. For example, the performance of a storage device (e.g., a disk that stores HFiles and/or StoreFiles) is monitored through the use of metrics including the number of read and write requests received.

164. HBase Monitoring

164.1. Overview

The following metrics are arguably the most important to monitor for each RegionServer for "macro monitoring", preferably with a system like OpenTSDB. If your cluster is having performance issues it's likely that you'll see something unusual with this group.

HBase

- See rs metrics

[112]

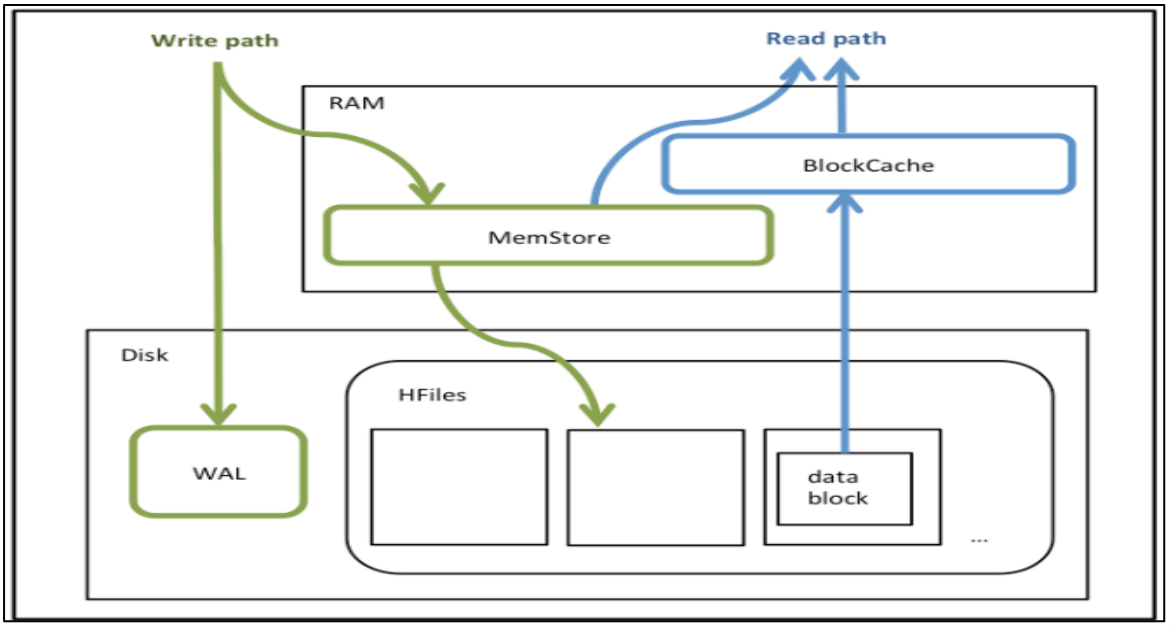
1. An Overview of HBase I/O

The following table describes several concepts related to HBase file operations and memory (RAM) caching.

HBase Component	Description
HFile	An HFile contains table data, indexes over that data, and metadata about the data.
Block	An HBase block is the smallest unit of data that can be read from an HFile. Each HFile consists of a series of blocks. (Note: an HBase block is different than an HDFS block or other underlying file system blocks.)
BlockCache	BlockCache is the main HBase mechanism for low-latency random read operations. BlockCache is one of two memory cache structures maintained by HBase. When a block is read from HDFS, it is cached in BlockCache. Frequent access to rows in a block cause the block to be kept in cache, improving read performance.
MemStore	MemStore ("memory store") is the second of two cache structures maintained by HBase. MemStore improves write performance. It accumulates data until it is full, and then writes ("flushes") the data to a new HFile on disk. MemStore serves two purposes: it increases the total amount of data written to disk in a single operation, and it retains recently-written data in memory for subsequent low-latency reads.
Write Ahead Log (WAL)	The WAL is a log file that records all changes to data until the data is successfully written to disk (MemStore is flushed). This protects against data loss in the event of a failure before MemStore contents are written to disk.

[113]

[112] https://hbase.apache.org/book.html#ops.monitoring
[113] https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.2.0/bk_Sys_Admin_Guides/content/sysadmin_HBase_io_overview.html



114

93. Apache HBase performs the step of intercepting communications between a computer system and said storage device. For example, a computer system is a client that functions, e.g., to send write requests, that are to be stored on a disk. When writing data, the communications between a computer system (e.g., a client) and a storage device (e.g., a disk), are intercepted by a RegionServer populating data into a MemStore prior to it being written to disk.

BlockCache and MemStore reside in random-access memory (RAM); HFiles and the Write Ahead Log are persisted to HDFS.

Figure 1 shows write and read paths (simplified):

- During write operations, HBase writes to WAL and MemStore. Data is flushed from MemStore to disk according to size limits and flush interval.
- During read operations, HBase reads the block from BlockCache or MemStore if it is available in those caches. Otherwise it reads from disk and stores a copy in BlockCache.

115

[114] https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.2.0/bk_Sys_Admin_Guides/content/sysadmin_HBase_io_overview.html
[115] https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.2.0/bk_Sys_Admin_Guides/content/sysadmin_HBase_io_overview.html

Figure 1 shows write and read paths (simplified):

- During write operations, HBase writes to WAL and MemStore. Data is flushed from MemStore to disk according to size limits and flush interval.
- During read operations, HBase reads the block from BlockCache or MemStore if it is available in those caches. Otherwise it reads from disk and stores a copy in BlockCache.

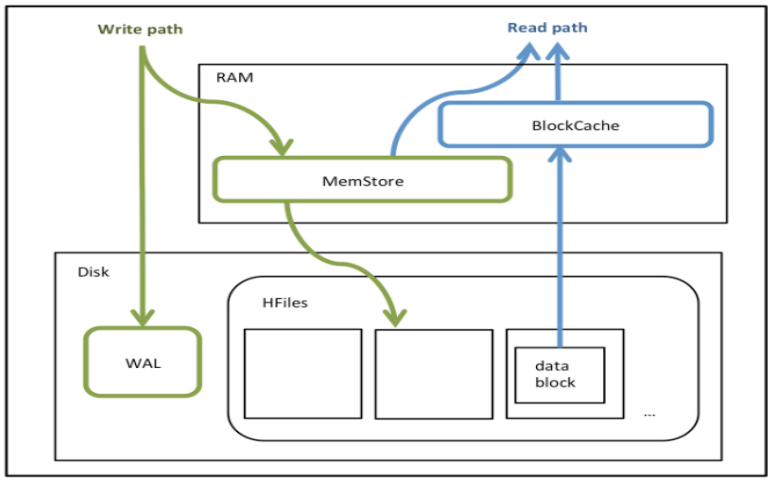


Figure 1. HBase read and write operations

116

94. Apache HBase performs the step of analyzing said intercepted communications. For example, Apache HBase analyzes the intercepted communications by: (1) monitoring the MemStore size; and (2) tracking the number of StoreFiles flushed to disk and comparing it to the value of the parameter hbase.hstore.compaction.min to determine whether to perform a compaction.

`hbase.regionserver.global.memstore.size`

Description

Maximum size of all memstores in a region server before new updates are blocked and flushes are forced. Defaults to 40% of heap (0.4). Updates are blocked and flushes are forced until size of all memstores in a region server hits hbase.regionserver.global.memstore.size.lower.limit. The default value in this configuration has been intentionally left empty in order to honor the old hbase.regionserver.global.memstore.upperLimit property if present.

Default

none

[116] https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.2.0/bk_Sys_Admin_Guides/content/sysadmin_HBase_io_overview.html

`hbase.hstore.compaction.min`

Description

The minimum number of StoreFiles which must be eligible for compaction before compaction can run. The goal of tuning hbase.hstore.compaction.min is to avoid ending up with too many tiny StoreFiles to compact. Setting this value to 2 would cause a minor compaction each time you have two StoreFiles in a Store, and this is probably not appropriate. If you set this value too high, all the other values will need to be adjusted accordingly. For most cases, the default value is appropriate (empty value here, results in 3 by code logic). In previous versions of HBase, the parameter hbase.hstore.compaction.min was named hbase.hstore.compactionThreshold.

Default

none [117]

95. Apache HBase performs the step of reallocating at least some of said data on said storage device to enhance the performance of said storage device based on said analyzed communications. For example, compaction reallocates at least some of the data (e.g., StoreFiles) on the storage device (e.g., disk) to enhance performance of the storage device by, for example, increasing performance on read operations. The reallocating is based upon the analyzed communications including use of various parameters such as hbase.regionserver.global.memstore.size and hbase.hstore.compaction.min.

72.7.7. Compaction

Ambiguous Terminology

- A *StoreFile* is a facade of HFile. In terms of compaction, use of StoreFile seems to have prevailed in the past.
- A *Store* is the same thing as a ColumnFamily. StoreFiles are related to a Store, or ColumnFamily.
- If you want to read more about StoreFiles versus HFiles and Stores versus ColumnFamilies, see HBASE-11316.

When the MemStore reaches a given size (`hbase.hregion.memstore.flush.size`), it flushes its contents to a StoreFile. The number of StoreFiles in a Store increases over time. *Compaction* is an operation which reduces the number of StoreFiles in a Store, by merging them together, in order to increase performance on read operations. Compactions can be resource-intensive to perform, and can either help or hinder performance depending on many factors. [118]

96. Defendants have directly infringed the ’832 Patent through their use, promotional demonstrations, testing, repairs, and instructional guidance of Apache Kafka Streams using RocksDB Universal Style Compaction, RocksDB Universal Style Compaction, Apache Cassandra, and Apache HBase, performing at least the steps of claim 25.

[117] https://hbase.apache.org/book.html
[118] https://hbase.apache.org/2.2/book.html

97. Plaintiffs have suffered damages as a result of Defendants' direct infringement of the '832 Patent in an amount to be proven at trial.

DEMAND FOR JURY TRIAL

Plaintiffs hereby demand a jury for all issues so triable.

PRAYER FOR RELIEF

WHEREFORE, Plaintiffs pray for relief against Defendants as follows:

a. Entry of judgment declaring that Defendants have directly and/or indirectly infringed one or more claims of each of the Patents-in-Suit;

b. Entry of judgment declaring that Defendants' infringement of the '416 Patent, the '686 Patent, and the '332 Patent is willful;

c. An order pursuant to 35 U.S.C. § 283 permanently enjoining Defendants, their officers, agents, servants, employees, attorneys, and those persons in active concert or participation with them, from further acts of infringement of the '416 Patent, the '686 Patent, and the'332 Patent;

d. An order awarding damages sufficient to compensate Plaintiffs for Defendants' infringement of the Patents-in-Suit, but in no event less than a reasonable royalty, together with interest and costs;

e. Entry of judgment declaring that this case is exceptional and awarding Plaintiffs their costs and reasonable attorney fees under 35 U.S.C. § 285; and

f. Such other and further relief as the Court deems just and proper.

IPR2026-00060 EX2010 073

Dated: May 9, 2025

Respectfully submitted,

/s/ Vincent J. Rubino, III

Alfred R. Fabricant
NY Bar No. 2219392
Email: ffabricant@fabricantllp.com
Peter Lambrianakos
NY Bar No. 2894392
Email: plambrianakos@fabricantllp.com
Vincent J. Rubino, III
NY Bar No. 4557435
Email: vrubino@fabricantllp.com
FABRICANT LLP
411 Theodore Fremd Avenue
Suite 206 South
Rye, New York 10580
Telephone: (212) 257-5797
Facsimile: (212) 257-5796

ATTORNEYS FOR PLAINTIFFS VALTRUS INNOVATIONS LTD. and KEY PATENT INNOVATIONS LTD.