Cassandra Day

2022

Future of Cassandra

Sponsored by DataStax

C* 4.1

C* 4.2

Ecosystem

Community

Upgrading

Apache Cassandra
Built as a Technical Necessity

of Fortune 100 companies have adopted

High Availability
Always On

Every second of downtime translates into lost revenue

Linear Scalability
Hyper-Scalability

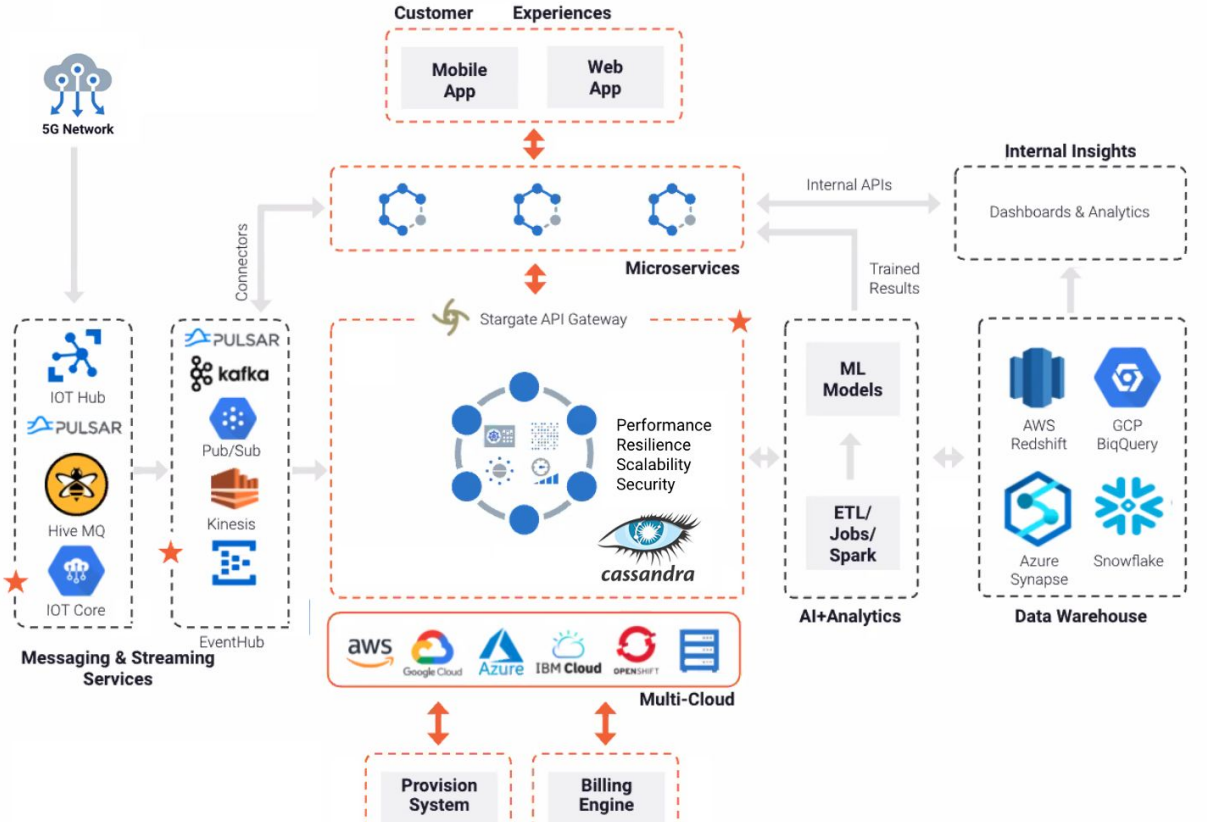
Millions of operations per day, hour, or second

Low Latency
Faster Pace

Every millisecond of latency has consequence

Global Distribution
Data Everywhere

On-premises, hybrid, multi-cloud, centralized, or edge


Customer Experiences
Mobile App
Web App
5G Network
Internal Insights
Dashboards & Analytics
Internal APIs
Trained Results
Connectors
Microservices
Stargate API Gateway
IOT Hub
PULSAR
Hive MQ
IOT Core
PULSAR
kafka
Pub/Sub
Kinesis
EventHub
Messaging & Streaming Services
Performance
Resilience
Scalability
Security
cassandra
ML Models
ETL/ Jobs/ Spark
AI+Analytics
AWS Redshift
GCP BiqQuery
Azure Synapse
Snowflake
Data Warehouse
aws
Google Cloud
Azure
IBM Cloud
OPENSHIFT
Multi-Cloud
Provision System
Billing Engine

- **4.0**
  - 25% faster
  - Faster Big Clusters
  - Incremental Repairs
  - Virtual Tables
  - Transient Replicas
- **4.1**
  - Pluggability
    - Storage
    - Encryption
    - Authentication
  - Guardrails
  - Improved Config format
- ***4.2***
  - ACID Transactions
    - Accord Consensus
  - Storage Attached Indexes
  - Trie Memtables
  - Transactional Cluster Metadata
  - Dynamic Data Masking
  - ...

C* 4.1

C* 4.2

Ecosystem

Community

Upgrading

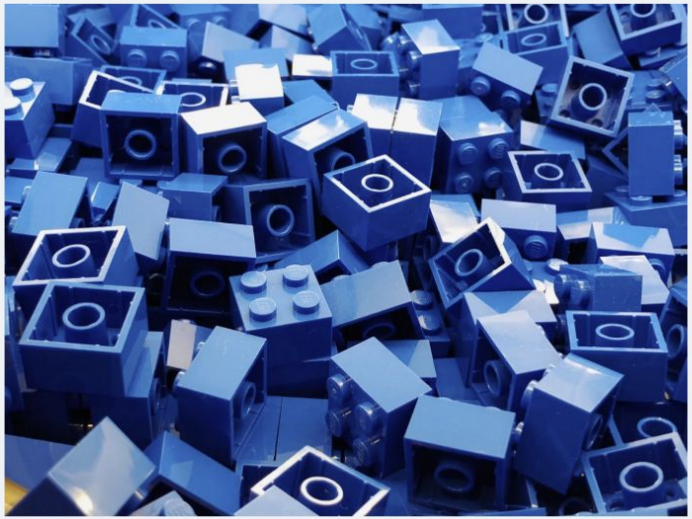
Cassandra 4.1 - What's in it?

- Themes
  - Major yearly releases
  - Usability (Ops safety)
  - Security
  - Pluggability

- Features
  - Configuration Improvements
  - System-wide Guardrails
  - Denylisting Partition Keys
  - Pluggable Extension Points
  - QA building blocks

Apache Cassandra 4.1: Building the Database Your Kids Will Use

Author: Patrick McFadin



https://medium.com/building-the-open-data-stack/apache-cassandra-4-1-building-the-database-your-kids-will-use-431f32210a30

Cassandra 4.1 – Improved Configuration

- Standard and Intuitive names
  - noun_verb
- Units in values

Apache Cassandra 4.1: Configuration Standardization

7 July, 2022 | Ekaterina Dimitrova

« Back to the Apache Cassandra Blog

Create flexibility for end users to provide units of their choice from a predefined list of units for parameters of type data storage, data rate, and duration. For example, this old property:

```
permissions_update_interval_ms: 0
```

can now be written in any of these ways:

```
permissions_update_interval: 0ms
permissions_update_interval: 0s
permissions_update_interval: 0d
permissions_update_interval: 0us
permissions_update_interval: 0µs
```

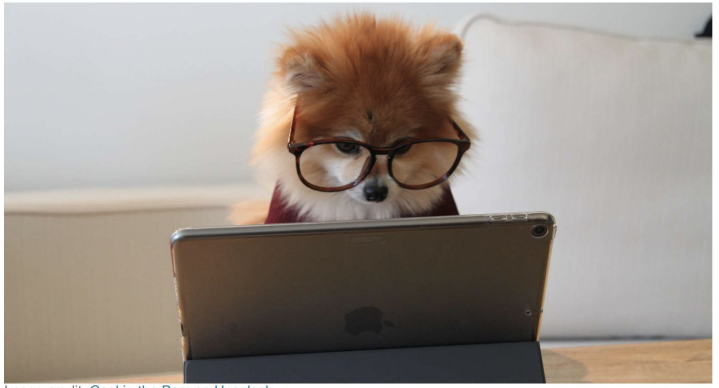
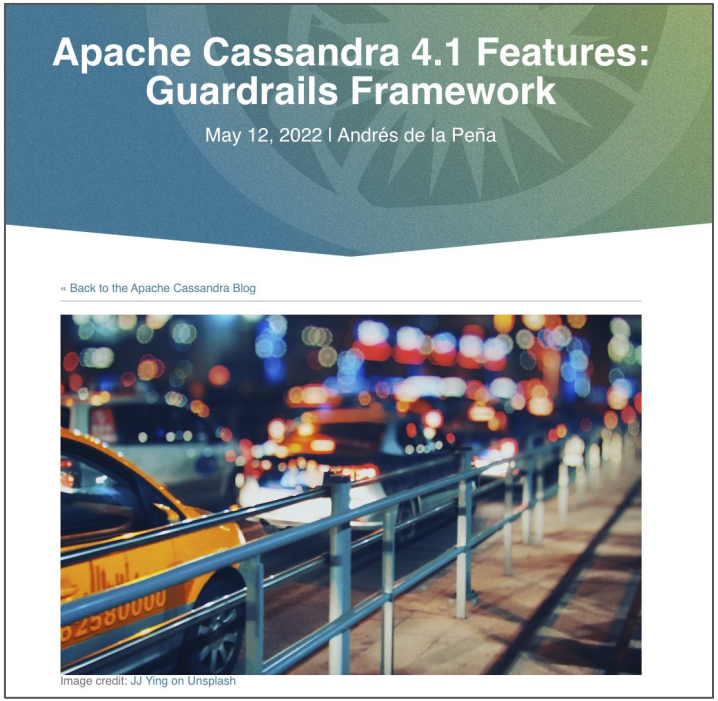


Image credit: Cookie the Pom on Unsplash

Cassandra 4.1 – Guardrails

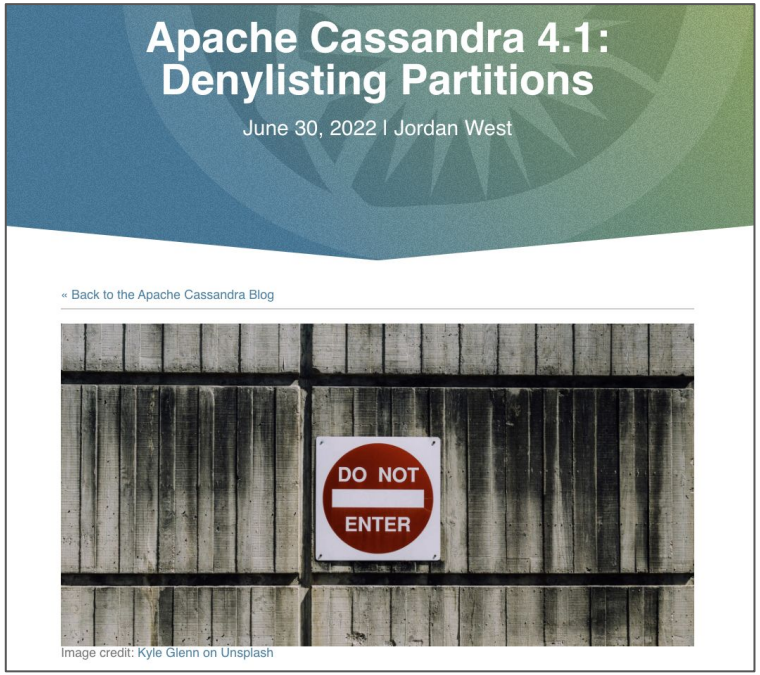
- Hard and soft limits
- Disabling features entirely
- Disallowing specific configuration values
- Administrative users exempted
- Typically checked at the CQL layer
- Based on work in Astra



https://cassandra.apache.org/_/blog/Apache-Cassandra-4.1-Features-Guardrails-Framework.html

Cassandra 4.1 - Denylisting Partition Keys (CEP-13)

- Overloaded partitions break clusters
  - compactions, streaming, GC, etc
  - Bad data modeling choices
  - Unintended usage or attacks
- The solution: operators can identify partition keys for which queries will be disallowed overloaded keys



https://cassandra.apache.org/_/blog/Apache-Cassandra-4.1-Denylisting-Partitions.html

Cassandra 4.1 – Pluggable extension points

- Pluggable memtables
  - Next: SSTables
- Pluggable network encryption
  - External key providers
- Pluggable authentication
  - Extend cqlsh via Python modules

C* 4.1

C* 4.2

Ecosystem

Community

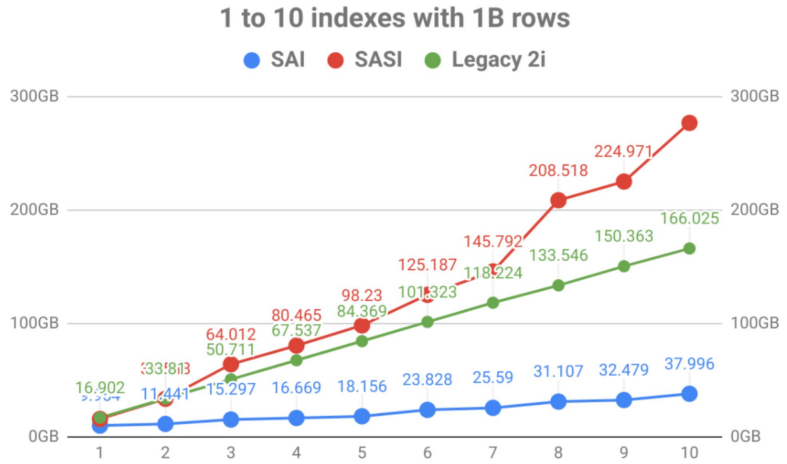
Upgrading

Cassandra 4.2 – What's it looking like?

- Storage Attached Index (SAI)
  - Index multiple columns without fear
- Trie Memtables
- Consensus protocol evolution
  - Paxos -> (E-Paxos ->) Accord
  - ACID transactions
- Transactional Cluster Metadata
- Dynamic Data Masking

Cassandra 4.2 – Storage Attached Indexes

Intended to replace both 2i and SASI



CEP-7: Storage Attached Index

Created by Zhao Yang, last modified by Caleb Rackliffe on Feb 22, 2022

Status

Current state: *Adopted*

Discussion thread: *dev discussion*

JIRA: CASSANDRA-16052

Released: Unreleased

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Scope

Storage Attached Indexing (SAI) is a new secondary index for the Apache Cassandra® distributed database system.

Goals

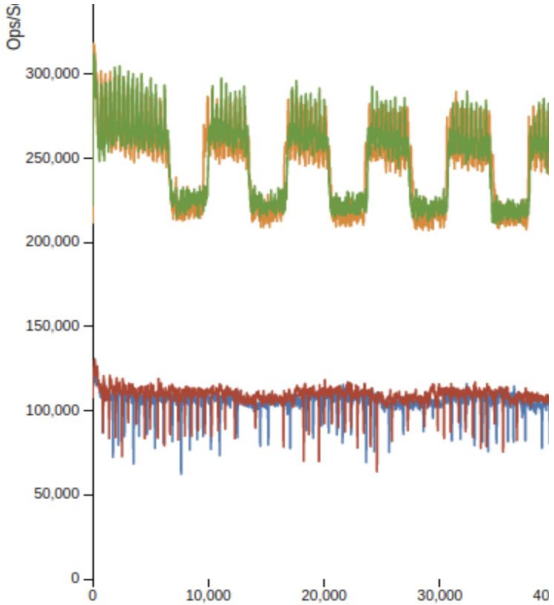
SAI is a new index implementation that builds on the advancements made with SASI. It's goals are roughly the following:

- Provide a solution that enables users to index multiple columns on the same table without suffering scaling problems, especially at write time.
- Achieve feature parity with existing 2i, then eventually extend to SASI features and beyond.
- Perform well enough to replace entirely both legacy 2i and SASI. (This would entail deprecation of and then the complete removal of SASI, which is currently experimental.)

Non-Goals

- Replace a search engine like Elastic or Solr

Cassandra 4.2 – Trie Memtables



CEP-19: Trie memtable implementation

Created by Branimir Lambov, last modified on Feb 25, 2022

Status

Current state: *Accepted*

Discussion thread: *here*

JIRA: CASSANDRA-17240

Released: tbd

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Memtables in Cassandra are big, have complex on-heap structure, and relatively long lifetimes, which makes them a pain point for memory management and garbage collection.

We would like to propose an alternative memtable implementation based on tries. It provides the following advantages:

- More compact storage with shared key prefixes.
- Can be placed off-heap and is garbage-collection-friendly when used on-heap.
- Fast lookups, resulting in faster writes and memtable reads.

Our internal testing has shown this can provide some dramatic performance improvements, including close to doubling of the sustained write throughput combined with drastically reduced tail latencies under load.

The implementation is already battle tested in production as the memtable format in DSE 6.8.

Cassandra 4.2 – Accord ACID Transactions

The example query is this:

```
BEGIN TRANSACTION
  LET car = (SELECT miles_driven, is_running FROM cars WHERE model='pinto');
  LET user = (SELECT miles_driven FROM users WHERE name='Blake');
  SELECT car.is_running, car.miles_driven;
  IF car.is_running THEN
    UPDATE users SET miles_driven = user.miles_driven + 30 WHERE name='blake';
    UPDATE cars SET miles_driven = car.miles_driven + 30 WHERE model='pinto';
  END IF
COMMIT TRANSACTION
```

CEP-15: General Purpose Transactions

Created by Benedict Elliott Smith, last modified on Oct 15, 2021

Audience: All Cassandra Users and Developers
User Impact: Support for fast general purpose transactions
Whitepaper: Accord
Prototype: https://github.com/belliottsmith/accord

Motivation

Users must expend significant effort to modify their database consistently while maintaining scalability. Even simple transactions involving more than one partition may become complex and error prone, as a distributed state machine must be built atop the database. Conversely, packing all of the state into one partition is not scalable.

Performance also remains an issue, despite recent Paxos improvements: latency is still twice its theoretical minimum over the wide area network, and suffers particularly badly under contention.

This work aims to improve Cassandra to support fast general purpose transactions. That is, those that may operate over any set of keys in the database atomically, modifying their contents at-once, with any action conditional on the existing contents of any key.

Goals

- General purpose transactions (may operate over any keys in the database at once)
- Strict-serializable isolation
- Optimal latency: one wide area round-trip for all transactions under normal conditions
- Optimal failure tolerance: latency and performance should be unaffected by any minority of replica failures
- Scalability: there should be no bottleneck introduced

https://cwiki.apache.org/confluence/display/CASSANDRA/CEP-15%3A+General+Purpose+Transactions

Cassandra 4.2 – Transactional Cluster Metadata

Camille Fournier ✓
@skamille

Now this is a beautiful piece of technical proposal writing
cwiki.apache.org/confluence/plu...
It has it all
Goals
Non goals
Migration plans
Test plans
Prior art
😘👌
Beautiful

3:56 AM · Aug 23, 2022 · Twitter Web App

27 Retweets **4** Quote Tweets **197** Likes

CEP-21: Transactional Cluster Metadata

Created by Sam Tunnicliffe, last modified on Aug 24, 2022

Status

Current state: Under Discussion

Discussion thread: https://lists.apache.org/thread/h25skwkbdztz9hj2pxtgh39rnjfzckk7

JIRA:

Released:

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Cluster metadata in Cassandra comprises a number of disparate elements including, but not limited to, distributed schema, topology and token ownership. Following the general design principles of Cassandra, the mechanisms for coordinating updates to cluster state have favoured eventual consistency, with probabilisitic delivery via gossip being a prime example. Undoubtedly, this approach has benefits, not least in terms of resilience, particularly in highly fluid distributed environments. However, this is not the reality of most Cassandra deployments, where the total number of nodes is relatively small (i.e. in the low thousands) and the rate of change tends to be low.

Historically, a significant proportion of issues affecting operators and users of Cassandra have been due, at least in part, to a lack of strongly consistent cluster metadata. In response to this, we propose a design which aims to provide linearizability of metadata changes whilst ensuring that the effects of those changes are made visible to all nodes in a strongly consistent manner. At its core, it is also pluggable, enabling Cassandra-derived projects to supply their own implementations if desired.

Probabilistic propagation of cluster state is suboptimal and not necessary, given the scale that Cassandra typically operates at. Not to mention it is (by definition) unreliable and, since there is no strict order for changes propagated through gossip, there can be no universal "reference" state at any given time. We aim to add a mechanism for establishing the order of state change events, and replace gossip as the primary mechanism of their dissemination in order to enable nodes which are participating in state-changing operations to detect when these intersect with other ongoing operations, such as reads and writes. The first two areas we plan to apply these changes to are Schema and Token Ownership. Establishing global order for the events requires us to add transactional semantics to changes in the state.

C* 4.1

C* 4.2

Ecosystem

Community

Upgrading

Flexibility Increases Developer Velocity

Drivers

APIs

Cassandra Query Language

SQL-Like Table Model

Structured Data

Key-Value Data

Strong Types

Minimal Query Overhead

gRPC

gRPC

CQL over gRPC

Structured Data

Lighter Weight

Native Driver Alternative

Low Query Overhead

GraphQL

Hierarchy of Types & Fields

Structured Data

Key-Value Data

Runtime Flexibility

{REST}

REST

Row Based

Structured Data

Key-Value Data

Weaker Types

Widely-Used & Flexible

Document

JSON Documents

Semi-Structured Data

Weaker Types

Schemless & Flexible

More Performant & More Structured

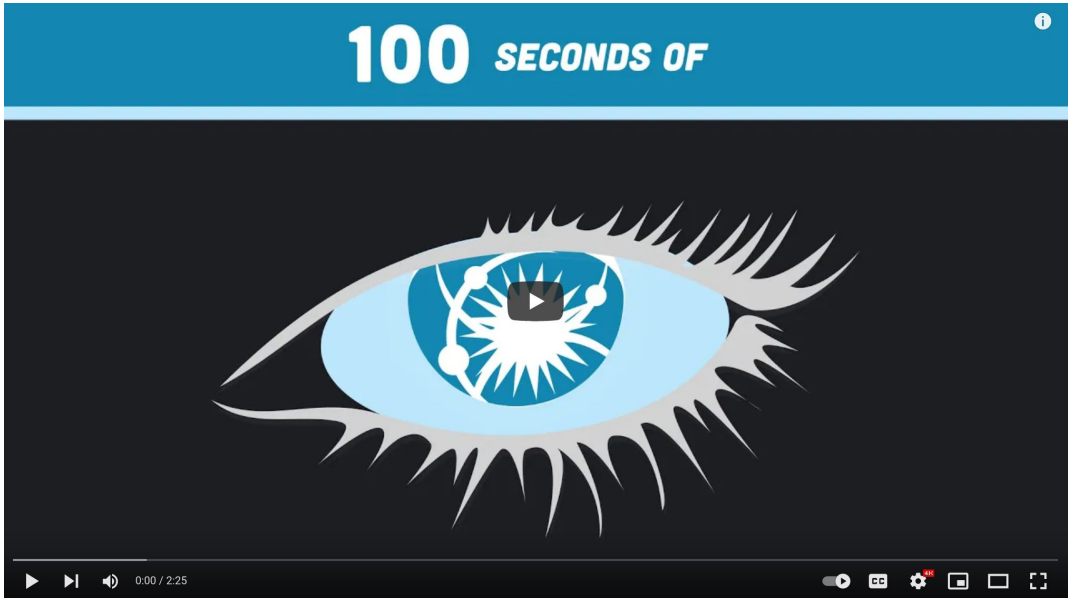
Developer Choices

APIs, Drivers, & Multiple Data Models

More Flexible & Less Structured

100 seconds of Apache Cassandra

www.youtube.com/watch?v=ziq7FUKpCS8



K8SSANDRA

Choice of Data APIs

Multi-Data Model

Any Language

Any Cloud

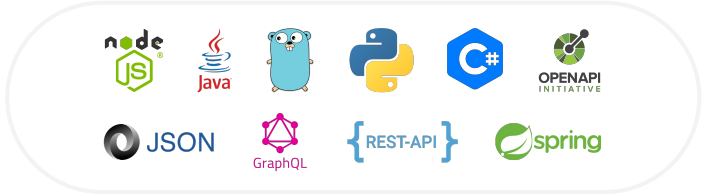
Multi-Region

Zero Downtime

Opinionated Ops

Scale & Elasticity

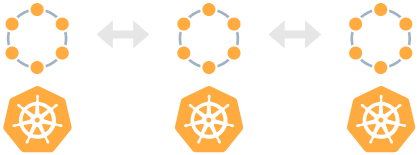
Any API, Any Language



Flexible Data Model

Column-Family | Document | Key-value | Storage Attached Index

Cloud-Native Cassandra Infrastructure Elasticity



Any Cloud

C* 4.1

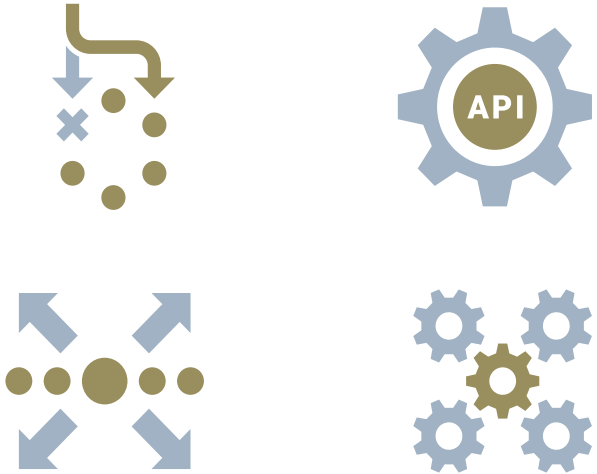
C* 4.2

Ecosystem

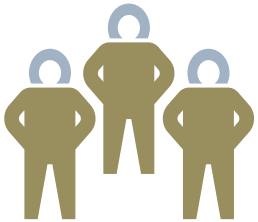
Community

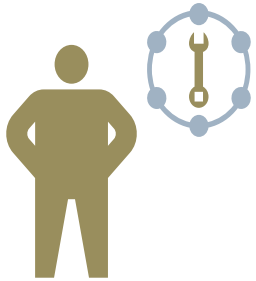
Upgrading

The Choices...

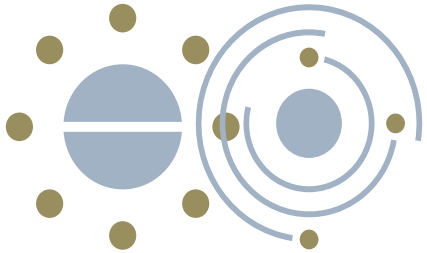


Community





The People...



C* 4.1

C* 4.2

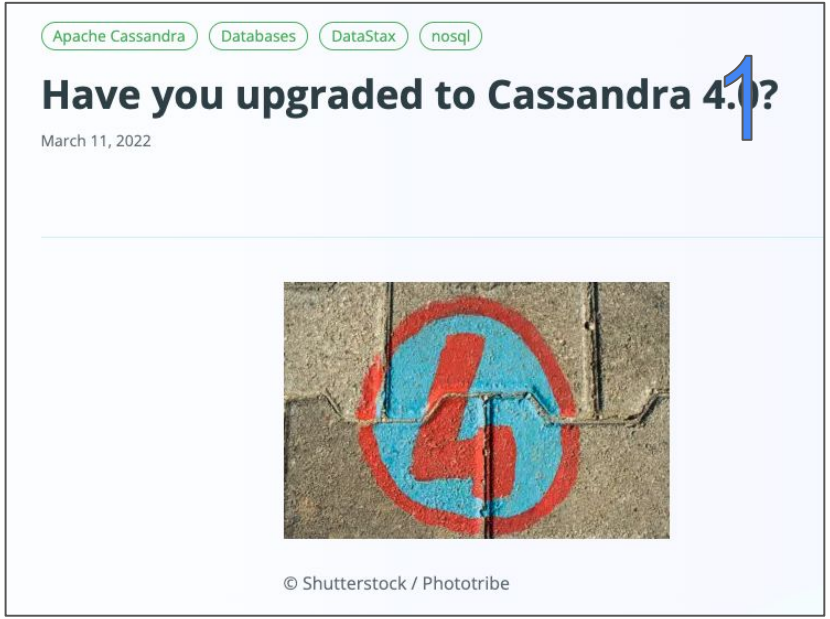
Ecosystem

Community

Upgrading

QA and Upgrades

- What upgrade paths do we support?
  a. Versus recommended paths…
- Simple steps
  a. Carefully check `NEWS.txt` and `CHANGES.txt`
  b. Prepare new config files
  c. Remove deprecated usages
  d. Upgrade JDK
  e. Upgrade drivers, pin native protocol version
  f. Verify your software versions
  g. Perform a cluster-wide snapshot
  h. Have current binaries ready for rollback
  i. Check disk space, no errors or dropped msgs
  j. Disable repairs and backups
  k. Upgrade the first node and verify
  l. Continue to each node in your cluster



https://foojay.io/today/have-you-upgraded-to-cassandra-4-0/

Thank you!

Sponsored by DataStax