



# Understanding the New InfluxDB Storage Engine

Anais Dotis-Georgiou

January 2023



# Anais Dotis-Georgiou

## Developer Advocate

---



**LinkedIn**




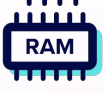



# Agenda

- What is the new InfluxDB Engine?
- What requirements does the new InfluxDB Engine meet?
  - Understanding the Apache Ecosystem
- Offerings and Release Timeline
- New InfluxDB Cloud Features
- SQL support
- Interoperability plans
- Survey
- Resources

---

The new storage engine that  
powers InfluxDB Cloud

# InfluxDB's new storage engine is built on

-  Rust
-   Apache Arrow
-  Apache Parquet
- Arrow Flight
-  DataFusion

# Release Details

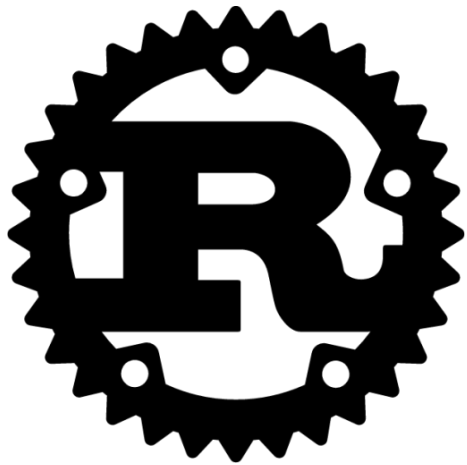
- Release Jan 31st
- Available in InfluxDB Cloud on AWS in:
  - AWS us-east-1
  - AWS eu-central-1

# Requirements for the new storage engine

Requirement/Feature	Rust	Arrow	DataFusion	Parquet
<b>1.</b> No limits on cardinality. Write any kind of event data and don't worry about what a tag or field is.	X	X	X	X
<b>2.</b> Best-in-class performance on analytics queries in addition to our already well-served metrics queries.	X	X	X	X
<b>3.</b> Separate compute from storage and tiered data storage. The DB should use cheaper object storage as its long-term durable store.			X	X
<b>4.</b> Operator control over memory usage. The operator should be able to define how much memory is used for each buffering, caching, and query processing.			X	
<b>5.</b> Bulk data export and import.				X
<b>6.</b> Broader ecosystem compatibility. Where possible, we should aim to use and embrace emerging standards in the data and analytics ecosystem.	X	X	X	X
<b>7.</b> Run at the edge and in the datacenter. Federated by design.	X			X

---

# Rust and InfluxDB Requirements





# Rust and Requirement 1.

**Requirement:** No limits on cardinality. Write any kind of event data and don't worry about what a tag or field is.

## **Rust Contributions:**

- InfluxDB's new storage engine is built on the Rust implementation of Apache Arrow which contributes heavily to meeting this requirement.
- Handling unlimited cardinality use cases requires non-trivial CPU during query processing. Rust supports optimizing resources for increased performance.

# Rust and Requirement 2.

**Requirement:** Best-in-class performance on analytics queries in addition to our already well-served metrics queries.

## **Rust Contributions:**

- Arrow and DataFusion are built on Rust.

# Rust and Requirement 4.

**Requirement:** Operator control over memory usage. The operator should be able to define how much memory is used for each buffering, caching, and query processing.

## **Rust Contributions:**

- Rust is used for memory control.

# Rust and Requirement 6.

**Requirement:** Broader ecosystem compatibility. Where possible, we should aim to use and embrace emerging standards in the data and analytics ecosystem.

## **Rust Contributions:**

- Rust helps support the implementation of Arrow, DataFusion, and Parquet.

# Rust and Requirement 7.

**Requirement:** Run at the edge and in the datacenter. Federated by design.

## **Rust Contributions:**

- Optimizing your memory usage with Rust, means that InfluxDB Cloud will also contain these memory optimizations at the edge or in the datacenter.

# Rust and Requirement 7.

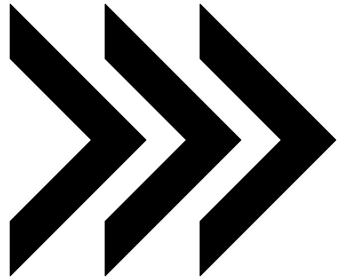
**Requirement:** Run at the edge and in the datacenter. Federated by design.

## **Rust Contributions:**

- Optimizing your memory usage with Rust, means that InfluxDB Cloud will also contain these memory optimizations at the edge or in the datacenter.

---

# Arrow and InfluxDB Requirements



# Arrow and Requirement 1.

**Requirement:** No limits on cardinality. Write any kind of event data and don't worry about what a tag or field is.

## **Arrow Contributions:**

- Apache Arrow overcomes memory challenges associated with large-cardinality use cases by providing efficient columnar data exchange.



---

# Advantages of Columnar Data Storage (sidebar)

# Sidebar—Advantages of Columnar Data Storage

```
measurement1,tag1=tagvalue1 field1=1i timestamp1  
measurement1,tag1=tagvalue2 field1=2i timestamp2  
measurement1,tag2=tagvalue3 field1=3i timestamp3  
measurement1,tag1=tagvalue1,tag2=tagvalue3 field1=4i,field2=true timestamp4  
measurement1, field1=1i timestamp5
```

# Sidebar–Advantages of Columnar Data Storage

Name: measurement1					
field1	field2	tag1	tag2	tag3	time
1i	null	tagvalue1	null	null	timestamp1
2i	null	tagvalue2	null	null	timestamp2
3i	null	null	tagvalue3	null	timestamp3
4i	true	tagvalue1	tagvalue3	tagvalue4	timestamp4
1i	null	null	null	null	timestamp5

# Sidebar–Advantages of Columnar Data Storage

1i	2i	3i	4i	1i
null	null	null	true	null
tagvalue1	tagvalue2	null	tagvalue1	null
null	null	tagvalue3	tagvalue3	null
null	null	null	tagvalue4	null
timestamp1	timestamp2	timestamp3	timestamp4	timestamp5

```
1i, 2i, 3i, 4i, 1i;  
null, null, null, true, null;  
tagvalue1, tagvalue2, null, tagvalue1, null;  
null, null, null, tagvalue3, tagvalue3, null;  
null, null, null, tagvalue4, null;  
timestamp1, timestamp2, timestamp3, timestamp4, timestamp5.
```

# Arrow and Requirement 2.

**Requirement:** Best-in-class performance on analytics queries in addition to our already well-served metrics queries.

## **Arrow Contributions:**

- Arrow offers best in class performance on analytics through the memory optimizations and efficient data exchange.

# Arrow and Requirement 3.

**Requirement:** Separate compute from storage and tiered data storage. The DB should use cheaper object storage as its long-term durable store.

## **Arrow Contributions:**

- Arrow provides the in-memory columnar storage while Parquet will provide the column-oriented data file format on disk.

# Arrow and Requirement 4.

**Requirement:** Operator control over memory usage. The operator should be able to define how much memory is used for each buffering, caching, and query processing

## **Arrow Contributions:**

- The Rust implementation of Apache Arrow provides fine grained memory control.

# Arrow and Requirement 6.

**Requirement:** Broader ecosystem compatibility. Where possible, we should aim to use and embrace emerging standards in the data and analytics ecosystem.

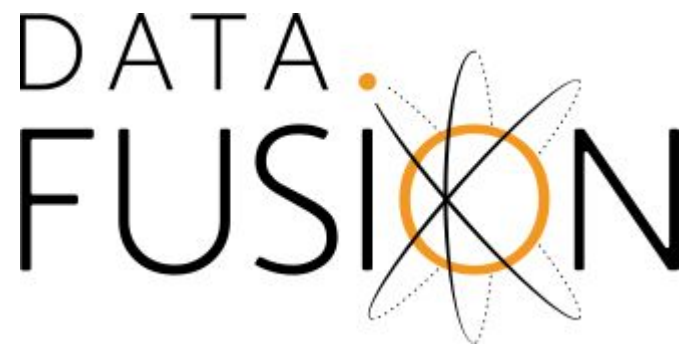
## **Arrow Contributions:**

- Leveraging Arrow is easier with the 12 libraries it supports for C, C++, Java, JavaScript, Python, Ruby, and more.



---

# DataFusion and InfluxDB Requirements



# DataFusion and Requirement 1.

**Requirement:** No limits on cardinality. Write any kind of event data and don't worry about what a tag or field is.

## DataFusion Contributions:

- What use is unlimited cardinality data if you can't query it? DataFusion provides the query, processing, and transformation of this data.

# DataFusion and Requirement 2.

**Requirement:** Best-in-class performance on analytics queries in addition to our already well-served metrics queries.

## **DataFusion Contributions:**

- What use is unlimited cardinality data if you can't query it? DataFusion provides the query, processing, and transformation of this data.

# DataFusion and Requirement 3.

**Requirement:** Separate compute from storage and tiered data storage. The DB should use cheaper object storage as its long-term durable store.

## DataFusion Contributions:

- DataFusion enables fast query against data stored on cheaper object store and separate compute.

# DataFusion and Requirement 6.

**Requirement:** Broader ecosystem compatibility. Where possible, we should aim to use and embrace emerging standards in the data and analytics ecosystem.

## **DataFusion Contributions:**

- DataFusion supports both a postgres compatible SQL and DataFrame API.

---

# Parquet and InfluxDB Requirements



**Parquet**



# Parquet and Requirement 2.

**Requirement:** Best-in-class performance on analytics queries in addition to our already well-served metrics queries.

## **Parquet Contributions:**

- Efficient compression and interoperability with ML and analytics tooling.

# Parquet and Requirement 3.

**Requirement:** Separate compute from storage and tiered data storage. The DB should use cheaper object storage as its long-term durable store.

## **Parquet Contributions:**

- Parquet files take up little disk space and are fast to scan.



# Parquet and Requirement 5.

**Requirement:** Bulk data export and import.

## **Parquet Contributions:**

- Parquet files enable bulk data export and import.

# Parquet and Requirement 5.

**Requirement:** Broader ecosystem compatibility. Where possible, we should aim to use and embrace emerging standards in the data and analytics ecosystem.

## **Parquet Contributions:**

- Parquet offers interoperability with modern ML and analytics tools.

# Parquet and Requirement 5.

**Requirement:** Run at the edge and in the datacenter. Federated by design.

## **Parquet Contributions:**

- Because Parquet files are so efficient, they will facilitate and increase the capacity for data storage at the edge.

# New Data Explorer

The screenshot displays the Data Explorer interface with several key components:

- Header:** "Data Explorer" title, navigation buttons for "+ New Script", "OPEN", and "SAVE", and a "Try New Script Editor" toggle switch (currently on).
- Left Panel:** "Schema Browser" with a "Bucket" dropdown menu and a "Flux Sync" toggle.
- Center Panel:** "Script Editor" with a text area containing a Flux query: `1 // Start by selecting data from the schema browser or typing flux here`. Below the editor are buttons for "CSV", "Past 1h", and "RUN".
- Right Panel:** "Flux library" with a search bar and a list of functions categorized by type (ARRAY, BITWISE).
- Bottom Panel:** "Query Results" section with a "Graph" view selector and a "CUSTOMIZE" button.

Annotations in the image include:

- A blue arrow pointing to the "+ New Script" button with the text "Create new script, select language, and save!".
- A blue arrow pointing to the "Try New Script Editor" toggle with the text "Toggle On".
- The text "Query Builder" is overlaid on the left side of the interface.
- The text "Script Editor" is overlaid in the center of the interface.

# New Data Explorer

The screenshot displays the Data Explorer interface with several key components highlighted by pink boxes:

- Top Bar:** Contains buttons for '+ NEW SCRIPT', 'OPEN', and 'SAVE'.
- Schema Browser:** On the left, it shows a tree view with 'air' selected under 'Bucket' and 'airSensors' under 'Measurement'. Under 'Fields', 'humidity' is selected. Under 'Tag Keys', 'sensor\_id' is expanded and 'TLM0100' is selected.
- Script Editor:** The central area contains a Flux script:

```
1 from(bucket: "air")
2
3   |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
4   |> filter(fn: (r) => r._measurement == "airSensors")
5   |> filter(fn: (r) => r._field == "humidity")
6   |> filter(fn: (r) => r.sensor_id == "TLM0100")
7
```
- Flux Library:** On the right, a 'Flux library' panel is open, showing a search bar and a list of functions under categories like 'ARRAY' and 'BITWISE'.
- Execution Controls:** Below the script editor, there is a 'Ready (1133ms)' status, a 'CSV' download button, a time range selector set to 'Past 1h', and a 'RUN' button.
- Graph View:** At the bottom, a 'Graph' view is active, showing a line chart with multiple data series over time. The x-axis represents time from 2023-01-06 10:30:00 to 2023-01-06 11:15:00, and the y-axis represents values from 0 to 70.

# iox.from() vs from()

```
1 import "experimental/iox"
2 iox.from(bucket: "air" , measurement: "airSensors")
3 // from(bucket: "air")
4 |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
5 // |> filter(fn: (r) => r._measurement == "airSensors")
6 |> filter(fn: (r) => r._field == "humidity")
7 |> filter(fn: (r) => r.sensor_id == "TLM0100")
8
```

# iox.from() vs from()

```
from(bucket: "anais-iox")
  |> range(start: 2022-12-01T19:05:41.000Z, stop: now())
  |> filter(fn: (r) => r._measurement == "airSensors")
  |> filter(fn: (r) => r._field == "temperature")
  |> filter(fn: (r) => r.sensor_id == "TLM0100")
```

Search results... 1 tables 361 rows

TABLE GRAPH

table	_measurement	_field	_value	_time	sensor_id
_result	group string	group string	no group double	no group dateTime:RFC3339	group string
0	airSensors	temperature	71.15167602106759	2022-12-01T19:05:41.000Z	TLM0100
0	airSensors	temperature	71.18581354517336	2022-12-01T19:05:51.000Z	TLM0100
0	airSensors	temperature	71.21025333073534	2022-12-01T19:06:01.000Z	TLM0100
0	airSensors	temperature	71.22533850921944	2022-12-01T19:06:11.000Z	TLM0100

1 2 3 4 5 ... 91

# iox.from() vs from()

```
import "experimental/iox"  
data = iox.from(bucket: "anais-iox", measurement: "airSensors")  
|> range(start: 2022-12-01T19:05:41.000Z, stop: now())  
|> filter(fn: (r) => r.sensor_id == "TLM0100")  
|> yield()
```



Search results... 1 tables 361 rows

TABLE GRAPH

table	_time	co	humidity	sensor_id	temperature
_result	no group dateTime:RFC3339	no group double	no group double	group string	no group double
0	2022-12-01T19:05:41.000Z	0.4929023294920129	35.05279937127679	TLM0100	71.15167602106759
0	2022-12-01T19:05:51.000Z	0.5118084177179043	35.08715659531805	TLM0100	71.18581354517336
0	2022-12-01T19:06:01.000Z	0.5303692714445265	35.050490323714186	TLM0100	71.21025333073534
0	2022-12-01T19:06:11.000Z	0.5402717872886482	35.03133016707075	TLM0100	71.22533850921944

1 2 3 4 5 ... 91



# SQL Support

The screenshot displays a SQL interface with the following components:

- Header:** "Basic SQL" and "Try New Script Editor" (toggleed on).
- Navigation:** "+ New Script", "OPEN", "SAVE", "EDIT", and "Provide Feedback".
- Left Panel:** "Bucket" (anis-iox) and "Measurement" (Select measurement...).
- Code Editor:** SQL query: 

```
1 SELECT "temperature", "sensor_id", "time" FROM "airSensors"  
2 where time >= ('2022-12-01 19:05:41.000')::TIMESTAMP  
3 and time < now()::TIMESTAMP and sensor_id = 'TLM0100'
```
- Execution:** "Ready (143ms)", "CSV", and "RUN" buttons.
- Graph:** A line graph showing temperature over time. The Y-axis ranges from 71.2 to 72.2. The X-axis shows timestamps from 2022-12-01 13:15:00 to 2022-12-01 14:00:00. The graph is titled "Graph" and has a "CUSTOMIZE" button.
- Right Panel (Customize):** "Data" section with "X Column" (time) and "Y Column" (temperature). "Adaptive Zoom" is checked. "Options" section includes "Time Format" (YYYY-MM-DD HH:mm:ss), "Interpolation" (Linear), "Line Colors" (Nineteen Eighty Four), "Hover Dimension" (auto), and "Shade area below graph" (unchecked). "X-Axis" section includes "Generate X-Axis Tick Marks".

# SQL Functions Supported by Jan 31

---

- **Supported Statements:** SELECT, FROM, WHERE, GROUP BY, ORDER BY, JOIN (left and inner), WITH clause, HAVING, UNION, LIMIT, OVER
- **Subqueries:** EXISTS, NOT EXISTS, IN, NOT IN
- **Agg Functions:** COUNT(), AVG(), MEAN(), SUM(), MIN(), MAX()
- **Time Series Functions:** TIME\_BUCKET\_GAPFILL(), DATEBIN(), NOW()
- **Other:** EXPLAIN, EXPLAIN ANALYZE

# Interoperability plans

- FlightSQL plugins (Timeline TBD):
  - Apache Superset
  - Tableau
  - PowerBI
  - Grafana

# Survey

- Please take 5 minutes to fill out this anonymous survey. Your feedback is extremely valuable to us.
- <https://bit.ly/3imWP7Y>



# Get Started



## Want to see how the new InfluxDB Engine works?

Sign up to get notified about the new InfluxDB Cloud Beta program today and stay up to date on our newest features.

Yes. I'm Excited.

[influxdata.com/influxdb-engine-beta/](https://influxdata.com/influxdb-engine-beta/)

# InfluxDB Community Slack workspace



Please join us in the InfluxDB  
Community Slack at  
[www.influxdata.com/slack](http://www.influxdata.com/slack).

To participate in conversations,  
join the `#influxdb_iox` channel.

**influxdata**® Products ▾ Developers Customers Company ▾

Pricing Contact Us Login ▾ **Get InfluxDB** 🔍

# It's About Time. Build on InfluxDB.

The Time Series Data Platform where developers build IoT, analytics, and cloud applications.

**Get InfluxDB** Find the right product →

[influxdata.com](https://influxdata.com)

## Influx Community

A place for Influx Community projects to be hosted

Overview Repositories **23** Packages People **7** Teams Projects Settings

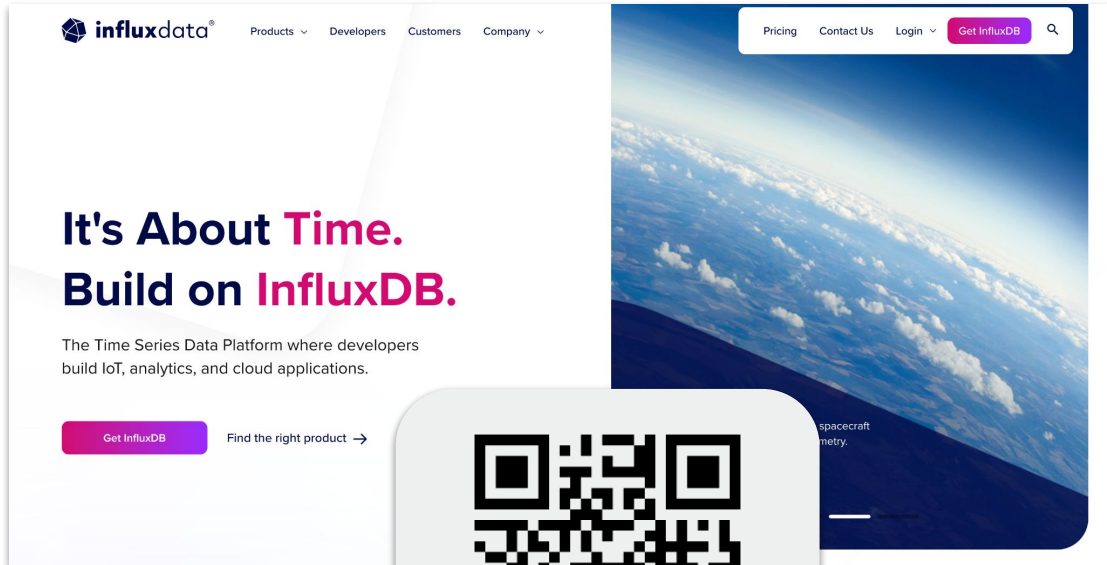
Pinned Customize your public profile

- plant\_buddy** (Public) Python ☆ 1 🍴 4
- tigitto** (Public)
- postman\_influx\_v2\_api** (Public) ☆ 1 🍴 1

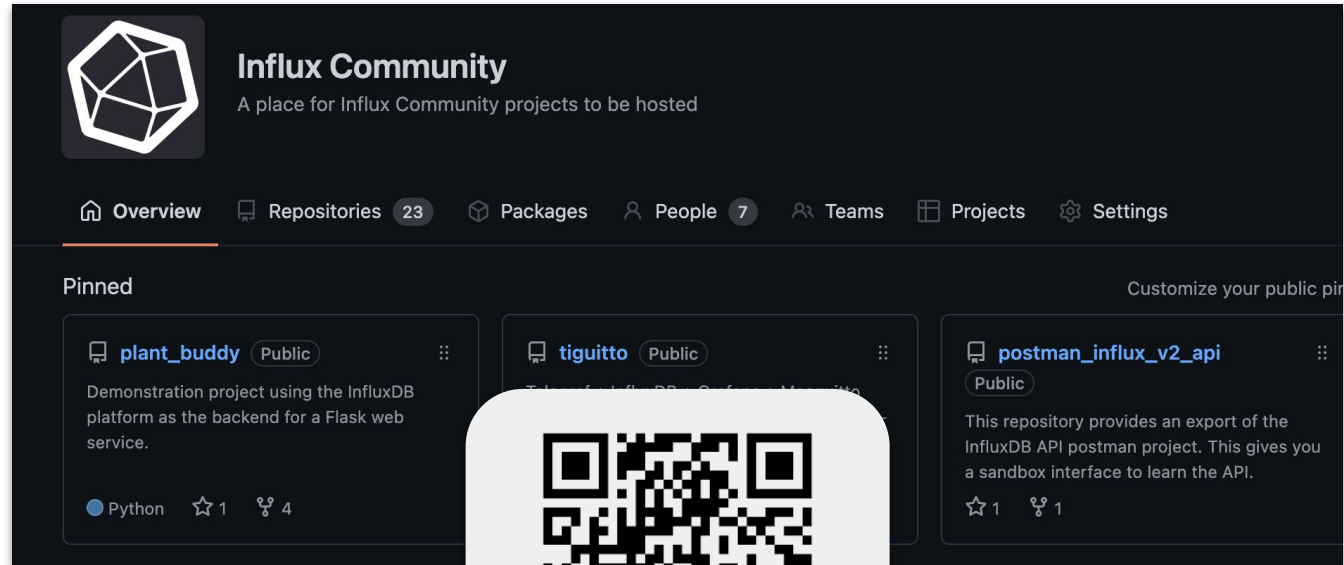
<https://github.com/InfluxCommunity>

# Try it yourself

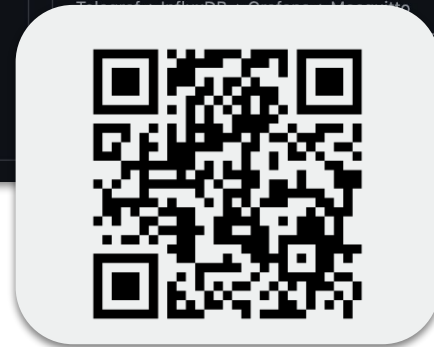
# Get Started



The screenshot shows the InfluxData website homepage. At the top left is the InfluxData logo. Navigation links include Products, Developers, Customers, and Company. On the right, there are links for Pricing, Contact Us, Login, and a prominent 'Get InfluxDB' button. The main content area features the headline 'It's About Time. Build on InfluxDB.' followed by a sub-headline 'The Time Series Data Platform where developers build IoT, analytics, and cloud applications.' Below this is another 'Get InfluxDB' button and a link 'Find the right product ->'. The background image shows a view of Earth from space.



The screenshot shows the Influx Community page on GitHub. It features the Influx logo and the title 'Influx Community' with the tagline 'A place for Influx Community projects to be hosted'. The navigation bar includes Overview, Repositories (23), Packages, People (7), Teams, Projects, and Settings. The 'Pinned' section displays three repositories: 'plant\_buddy' (Public), 'tigitto' (Public), and 'postman\_influx\_v2\_api' (Public). Each repository card includes a brief description and statistics for stars and forks.





# Related Blogs

- [Understanding InfluxDB IOx and the Commitment to Open Source](#)
- [Querying Data in InfluxDB using Flux and SQL](#)
- [Intro to InfluxDB IOx](#)
- [Welcome to InfluxDB IOx: InfluxData's New Storage Engine](#)
- [The Journey of InfluxDB by Paul Dix](#)
- [InfluxData Deploys Next-Generation InfluxDB Time Series Engine with Unlimited Scale](#)
- [Announcing InfluxDB IOx - The Future Core of InfluxDB Built with Rust and Arrow](#)
- [Evolving InfluxDB into the Smart Data Platform for Time Series](#)
- [InfluxData is Building a Fast Implementation of Apache Arrow in Go Using c2goasm and SIMD](#)
- [On InfluxData's New Storage Engine. Q&A with Andrew Lamb](#)
- [Apache Arrow, Parquet, Flight and Their Ecosystem are a Game Changer for OLAP](#)

# Get Help + Resources!

**Forums:** [community.influxdata.com](https://community.influxdata.com)

**Slack:** [influxcommunity.slack.com](https://influxcommunity.slack.com)

**GH:** [github.com/InfluxCommunity](https://github.com/InfluxCommunity)

**Book:** [awesome.influxdata.com](https://awesome.influxdata.com)

**Docs:** [docs.influxdata.com](https://docs.influxdata.com)

**Blogs:** [influxdata.com/blog](https://influxdata.com/blog)

**InfluxDB University:** [influxdata.com/university](https://influxdata.com/university)



# Questions?



---

THANK YOU