influxdata®

Using the InfluxDB 3.0 Go Client Library

Anais Dotis-Georgiou

April 2024

Anais Dotis-Georgiou
Developer Advocate

LinkedIn

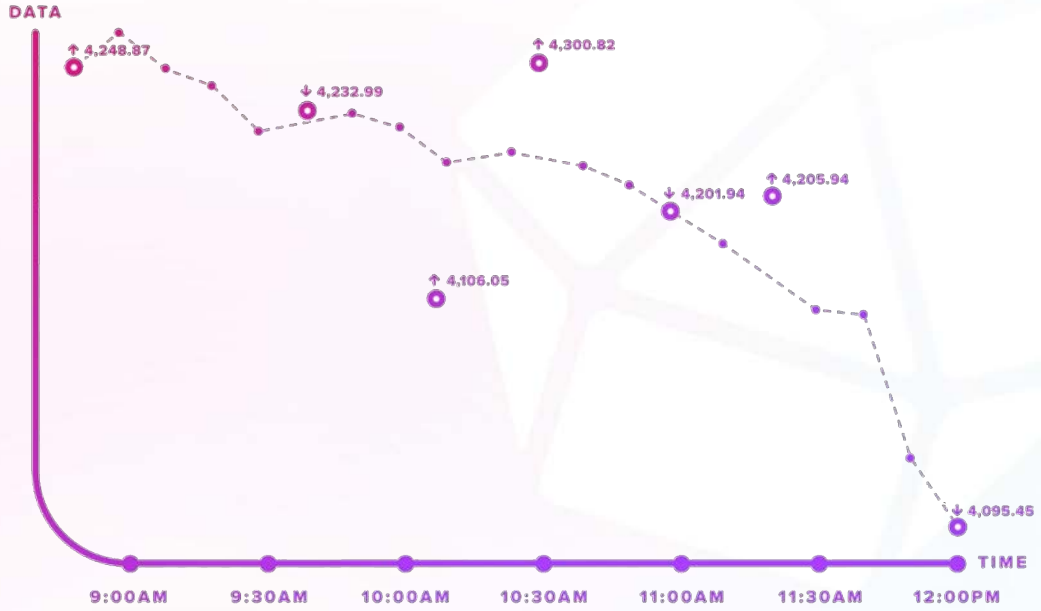
influxdata®

Brought to you by InfluxDB University

InfluxDB University offers free live and self-paced training on:

- InfluxDB v3
- Client Libraries
- Data Science tools and InfluxDB v3
- and more!

A Critical Component of Modern Data Pipelines



Time Series Data

influxdata®

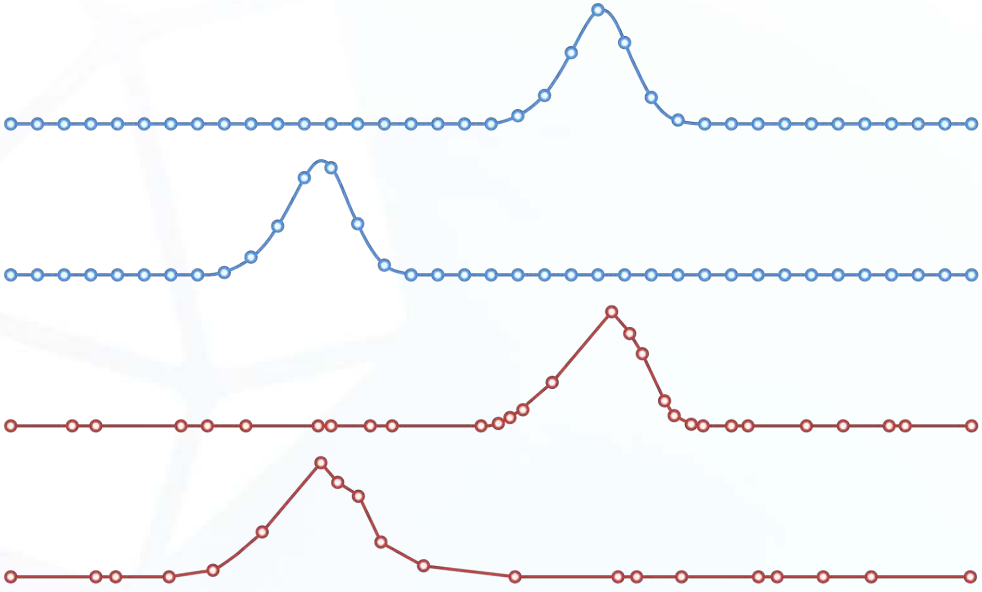
Time Series Data Types

Metrics

Measurements at **regular** time intervals

Events

Measurements at **irregular** time intervals



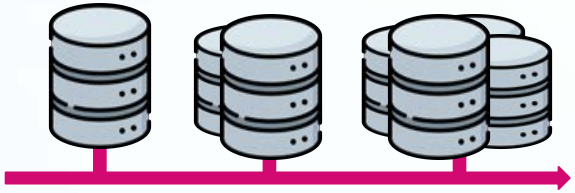
influxdata®

Time Series Databases

Time Series Data

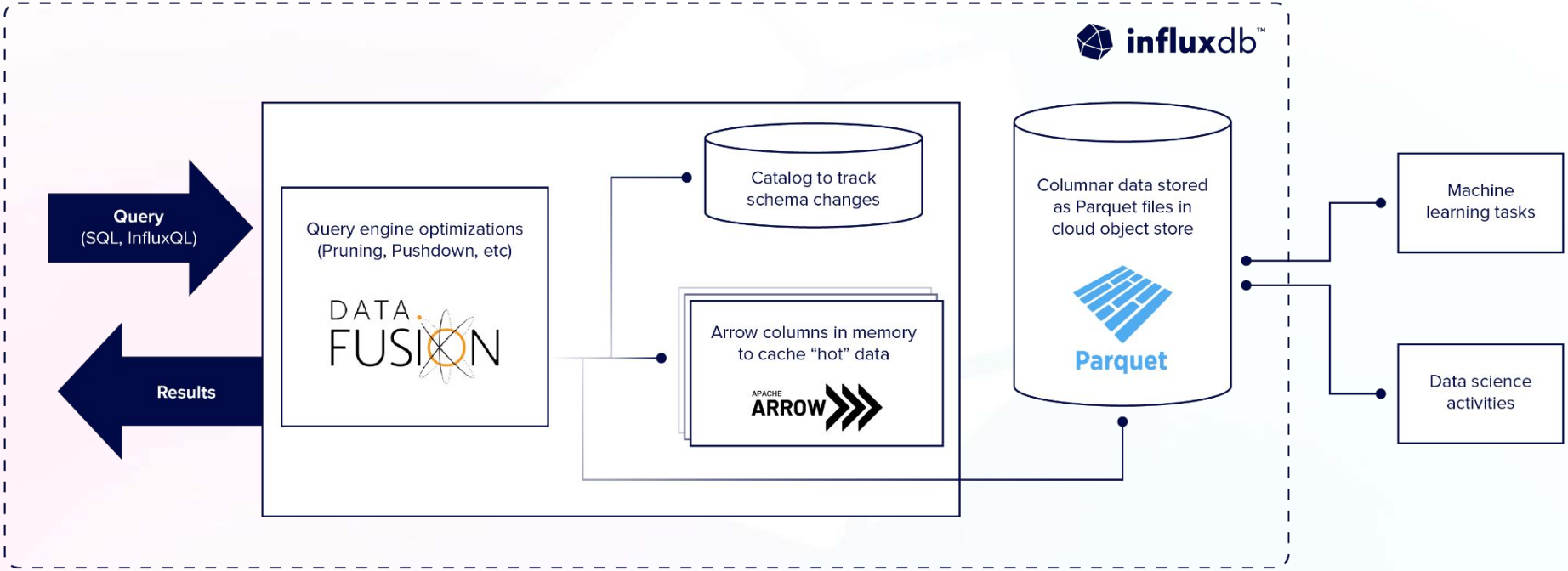
High write throughput

Efficient Queries Over Time Ranges



Scalability and Performance

InfluxDB 3.0



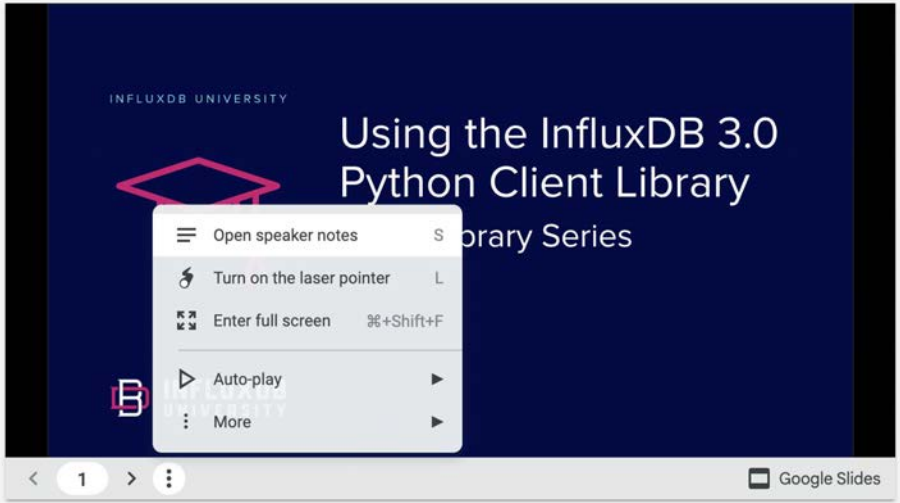
influxdata®

PENDING PUTTING IN VIDEO HERE
ON HOW TO USE

Before we begin...

Please note you can click on blue links for more information on topics and to see further examples.

Click on the three dots below to see the speaker notes to copy and paste code examples and see more detail.



Agenda

- What is the InfluxDB v3 Go Client Library
- Requirements
- Writing data to InfluxDB v3 with the Go Client Library
- Query data with the Go Client Library
- Resources + Help

What is the InfluxDB v3 Go Client Library

InfluxDB v3 Go Client Library

- The InfluxDB Go client library is a software package that provides a set of tools and functions for interacting with InfluxDB using the Go programming language.
- It allows developers to efficiently query and write time-series data from/to InfluxDB, simplifying the integration of InfluxDB into Go applications.

Query Advantages

- The influxdb3-Go Go client library wraps the Apache Arrow client in a convenient InfluxDB v3 interface for executing SQL and InfluxQL queries, requesting server metadata, and retrieving data from InfluxDB Cloud Dedicated using the Flight protocol with gRPC.
- The Apache Arrow Flight Go Client enables transport of large datasets over network interface.
- The Flight protocol with gRPC provides efficient serialization and deserialization and bidirectional streaming.

How the Go Client Library works

- Writes are implemented via the /write API endpoint.
- Queries are implemented via the Apache Arrow Flight Client Libraries and utilize the Arrow Format and Flight gRPC protocol.

Requirements

Requirements

- InfluxDB Cloud 3.0 account
- Database Name (sometimes referred to as a Bucket)
- Authentication token

Resource Center

What would you like to do?

Manage Databases & Security

Create and manage your buckets (databases) & access tokens.

Add Data

Write data into your database with our reporting agent, programmatically, API, CLI, or upload a CSV or Line Protocol File.

Query Data

Query your data with the UI, programmatically, or integrate with 3rd party tools.

Visualize & Alert

Integrate with 3rd party tools to visualize your data or set up alerts.

Create a Database (or Bucket)

What's New

Now Available: The Flight SQL Plugin for Grafana

Grafana now has a community plugin that enables communication with Flight SQL-compatible databases.

What does that mean for you?

- **InfluxDB 3.0 Support and Compatibility**
- **Easy Setup with Grafana Cloud**
- **Enhanced Data Querying and Visualization**

LEARN MORE

Latest Blogs

DevSecOps and DevOps: Key Differences

JUL 24

This post was written by Vincent Chosen. Scroll down for the author's bio. DevOps and DevSecOps have gained more attention in recent years in the world of software development. While both of these methodologies emphasize the agile development process and team... read more

GitOps vs. DevOps: What's the Difference?

JUL 21

This post was written by Damilola Ezekiel, a Software Engineer and a Technical Writer who enjoys learning and sharing new things through writing. She is also an avid open source contributor. In software development, GitOps and DevOps are prominent techniques for... read more

MORE BLOGS

Installation

Installation

Add the latest version of the client package to your project dependencies (go.mod):

```
go get github.com/InfluxCommunity/influxdb3-go/influxdb3
```

Write Data

Instantiate the Client–Description

1. Import your packages.
2. Instantiate the client to write and query InfluxDB v3 by providing your credentials.

Instantiate the Client–Code

```
package main

import (
    "context"
    "fmt"
    "os"
    "time"

    "github.com/InfluxCommunity/influxdb3-go/influxdb3"
)
```

Instantiate the Client–Code

```
func main() {
	// Use env variables to initialize client
	url := os.Getenv("INFLUXDB_URL")
	token := os.Getenv("INFLUXDB_TOKEN")
	database := os.Getenv("INFLUXDB_DATABASE")

	// Create a new client using an InfluxDB server base URL and an authentication
token
	client, err := influx.New(influx.Configs{
		HostURL:   url,
		AuthToken: token,
	})
```

Write Data–Line Protocol–Description

1. To write data to InfluxDB, call `client.Write` with data in line protocol format and the database (or bucket) name.
2. Line protocol is the ingest format for InfluxDB. Line Protocol consists of:
   - Measurements
   - Tags
   - Fields
   - Timestamp
3. When writing a point to InfluxDB, tags are used to store metadata to your instance. Fields are used to contain your actual time series values. However both fields and tags convert to columns in a table in InfluxDB. In practice they are identical, so this distinction is solely for organizational purposes for the user.
4. Data is written synchronously.

Write Data–Line Protocol–Code

```
line := "stat,unit=temperature avg=23.5,max=45.0"
err = client.Write(context.Background(), database,
[]byte(line))
if err != nil {
      panic(err)
   }
```

Write Data–Points–Description

1. This is a code example for how to write a single point to InfluxDB v3. With the `WritePoints` Method.
2. Use the `NewPoint` method to create a Point.
3. You can also append points to an array and write an array of Points to InfluxDB by passing in the array into the `client.WritePoints` method.
4. Data is written synchronously.

Write Data–Points–Code

```
// Create point using full params constructor
 p := influx.NewPoint("stat",
     map[string]string{"unit": "temperature"},
     map[string]interface{}{"avg": 24.5, "max": 45.0},
     time.Now())
 // write point synchronously
 err = client.WritePoints(context.Background(), database, p)
 if err != nil {
     panic(err)
 }
```

Write Data–Upserts

Important Note: You can upsert a field but not a tag. For example if you add a second point (notice the addition of “2” to field values) you would upsert those field values and your previous values will be overwritten with the new field values:

```
# Adding first point
line := "stat,unit=temperature avg=23.5,max=45.0 1690218372000000000"
err = client.Write(context.Background(), database, []byte(line))
if err != nil {
         panic(err)
     }

# Adding second point
line := "stat,unit=temperature avg=23.52,max=45.2 1690218378000000000"
err = client.Write(context.Background(), database, []byte(line))
if err != nil {
         panic(err)
     }
```

Write Data–Upserts

Important Note: However, if you add a second point (notice the addition of “2” to the tags) you would not upsert those values. You would simply add more tag values.

```
# Adding first point
line := "stat,unit=temperature avg=23.5,max=45.0 1690218372000000000"
err = client.Write(context.Background(), database, []byte(line))
if err != nil {
         panic(err)
     }

# Adding second point
line := "stat,unit=temperature2 avg=23.52,max=45.2 1690218378000000000"
err = client.Write(context.Background(), database, []byte(line))
if err != nil {
         panic(err)
     }
```

Query Data

Query Data–SQL

Here we use SQL to query InfluxDB

```
query := `
                    SELECT *
                    FROM "stat"
                    WHERE
                    time >= now() - interval '5 minute'
                    AND
                    "unit" IN ('temperature')
      `

      iterator, err := client.Query(context.Background(), query)

      if err != nil {
            panic(err)
      }

      for iterator.Next() {
            value := iterator.Value()

            fmt.Printf("avg is %f\n", value["avg"])
            fmt.Printf("max is %f\n", value["max"])
      }
```

`Query` Method–Parameters

`ctx`	The context.Context to use for the request. Use the default `context.Background()`.
`database`	The database to be used for InfluxDB operations.
`query`	The SQL query string to execute.

Query Data–SQL explicitly

Here we use SQL to query InfluxDB

```
//Specify querying with SQL explicitly
      options := influxdb3.QueryOptions{
            QueryType: influxdb3.FlightSQL,
      }

      iterator, err := client.QueryWithOptions(context.Background(), &options, query)

      if err != nil {
            panic(err)
      }

      for iterator.Next() {
            value := iterator.Value()
            fmt.Printf("avg is %f\n", value["avg"])
            fmt.Printf("max is %f\n", value["max"])
            fmt.Printf("time is", value["time"])
      }

      }
```

Query Data–InfluxQL

Here we use InfluxQL to query InfluxDB

```
influxQLQuery := "SHOW MEASUREMENTS"

options := influxdb3.QueryOptions{
     QueryType: influxdb3.InfluxQL,
}

iterator, err := client.QueryWithOptions(context.Background(), &options,
influxQLQuery)

if err != nil {
     panic(err)
}

for iterator.Next() {
     value := iterator.Value()
     fmt.Printf("measurement is:", value["name"])
}
```

`QueryWithOptions` Method–Parameters

`ctx`	The context.Context to use for the request. Use the default `context.Background()`.
`database`	The database to be used for InfluxDB operations.
`options`	Query options (query type, optional database). query type can be `FlightSQL` or `InfluxQL`

Full Code Examples

See this code example writing different record types (Line Protocol, Points, and Records) and querying with SQL.

github.com/InfluxCommunity/influxdb3-go/blob/main/example/main.go

Resources + Help

Resources

1. InfluxDB v3 Go Client Library Repository

github.com/InfluxCommunity/influxdb3-python

2. InfluxDB v3 Go Client Library Documentation

docs.influxdata.com/influxdb/cloud-dedicated/reference/client-libraries/v3/python/

InfluxDB Community Slack workspace

Please join us in the InfluxDB Community Slack at www.influxdata.com/slack.

To participate in conversations, join the #influxdb_iox channel.

Get Help + Resources!

Forums: community.influxdata.com

Slack: influxcommunity.slack.com

InfluxCommunity: github.com/InfluxCommunity

Docs: docs.influxdata.com

Blogs: influxdata.com/blog

InfluxCommunity is a GH org where you can find a collection of examples and demos for using and building solutions with InfluxDB.

influxdata®

Try it yourself

Get Started

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 →

uses InfluxDB to manage spacecraft with large volumes of telemetry.

influxdata®

INFLUXDB UNIVERSITY

www.influxdbu.com