DS4300 Spring 2024
Xi Chen, James D'Eila, Kaydence Lin, Kaito Minami, Yidi Wang

Diving into InfluxDB: Time Flow Analysis. Past, Present, and Future!

Abstract

We will cover the current most popular time-series database, InfluxDB with its key principles, diverse use cases, comprehensive tutorial, and critical analysis including strengths, weaknesses, licensing terms, community base, and more. We will also go over its unique UI to answer why InfluxDB's simple yet informative design to specialize in time series has an advantage over its Relational Database counterpart. Our goal for this report is to introduce and educate the idea of InfluxDB for readers to apply it practically in the future.

Introduction

The origin of InfluxDB started way before its first commit in 2013. The founders applied to Y Combinator for an investor with a real-time metric and monitoring SaaS project idea, Errplane. Along with this Errplane, they were asked for a second idea and it was the InfluxDB, the open-source time series database. The first version of Errplane was first launched in 2012 on Cassandra. To go with on-premise development, the second version was rewritten in Go. They improved the Errplane API over the next 10 months and pushed two major APIs: one for data collection with a pub-sub mechanism and one for counting and aggregating and a service for answering queries. Errplane was not as successful, so they dug into its main usage and it turned out they were using it like a time-series database.

In 2013, based on this finding, they made a major shift, which was later officially called InfluxDB. At the time, Graphite and OpenTSDB were the go-to for the open-source time-series database. The most common complaint for them, however, was that it was hard to install and it did not scale well, and neither was pushed at any level closer to Cassandra, Hadoop, Riak, and Mongo despite their popularity. The final realization came when they went to the Monitorama Conference in Berlin in September 2013. They expected to meet potential customers of Errplane, but instead, they found half of the attendees were employees and entrepreneurs attempting the same thing. If so many people are trying the same thing, it creates a barrier to entry and rather there's a bigger chance of creating a standard open-source platform, inviting all of the competitors as customers.

Even before the official project release, InfluxDB gained international support. Members of the growing InfluxDB community have given talks in Kyoto, Sydney, Cologne, and Dublin. They already had 3,000 stars on GitHub and 47 contributors to its core. In the 1.0 release in 2014, the production team focused on stability, scalability, production use, and API to be mature enough to not have major changes for a while. Upon its success, the Errplane company changed its name to InfluxData Inc in 2015. In 2016, the platform introduced its cloud offering. Flux, a purpose-built time series data query language, was introduced in 2018. Now it serves its reputable next-generation monitoring, analytics, and IoT applications to many businesses including Autodesk, Cisco, eBay, and Coupa, and the original InfluxDB GitHub repository has grown its popularity to gain over 27,000 stars and 514 contributors.

Key Principles

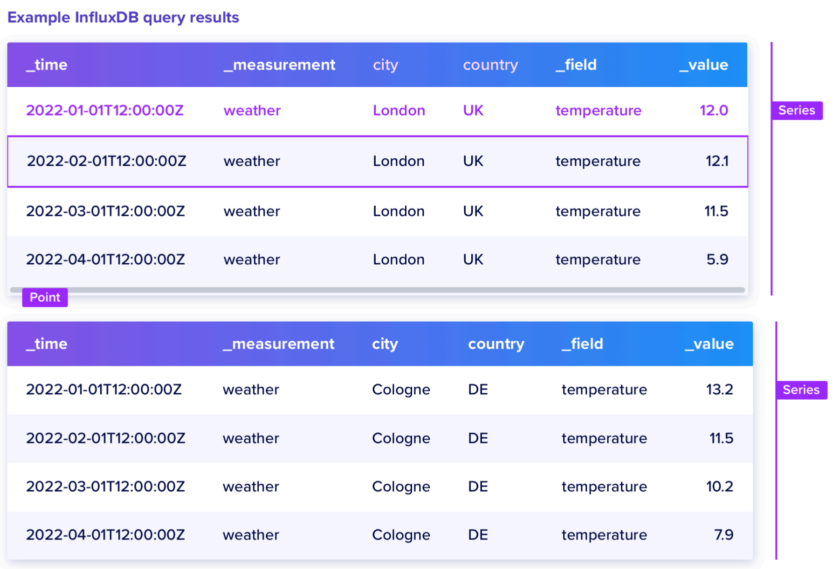
InfluxDB 2.7 is a database platform to collect, store, process, and visualize time-series data. Time series data is a sequence of data points ordered chronologically. We can observe the measurements from the same source to track change over time. Examples of time series data include: Industrial sensor data, Server performance metrics, Heartbeats per minute, Electrical activity in the brain, Rainfall measurements, and Stock prices.

To organize data more concisely, InfluxDB data model has a structure to contain buckets and measurements. A **bucket** can contain multiple measurements. Measurements consist of tags, fields, and timestamps.

- **Tags** are the key value that does not change over time such as location, host, station, and name.
- **Fields** are the key value that does change over time such as temperature, pressure, and stock prices.
- **Timestamp** allows the database to easily sort the data chronologically.

For more essential terminologies:

- **Point** is the single data record identified by its measurement, tag keys, tag values, field key, and timestamp.
- **Series** is a group of points with the same measurement, tag keys, and tag values.


Example InfluxDB query results
Series
Point
Series

_time	_measurement	city	country	_field	_value
2022-01-01T12:00:00Z	weather	London	UK	temperature	12.0
2022-02-01T12:00:00Z	weather	London	UK	temperature	12.1
2022-03-01T12:00:00Z	weather	London	UK	temperature	11.5
2022-04-01T12:00:00Z	weather	London	UK	temperature	5.9

_time	_measurement	city	country	_field	_value
2022-01-01T12:00:00Z	weather	Cologne	DE	temperature	13.2
2022-02-01T12:00:00Z	weather	Cologne	DE	temperature	11.5
2022-03-01T12:00:00Z	weather	Cologne	DE	temperature	10.2
2022-04-01T12:00:00Z	weather	Cologne	DE	temperature	7.9

Liscensing

InfluxDB Cloud Serverless - Open Source Database Plan

This is for customers who use smaller workloads and uses a multi-tenant infrastructure. This has a 30 day free trial and after that, you only pay for what you use. There is no commitment to using this service.

Enterprise Versions

- InfluxDB CloudDedicated: This is for customers who use scaled workloads and uses a dedicated infrastructure.
- InfluxDB Clustered: This is for customers who use large-scale workloads and uses managed on-premise or in your own personal private cloud.
- The enterprise versions are often used by businesses which has maintenance agreements amd special controls. It is also often installed on a business's server.

Marketplace Subscriptions

You can also subscribe to InfluxDB Cloud Serverless from a Cloud Marketplace account to like Amazon Web Services, Microsoft Azure, and Googld Cloud.

Use-Cases

Predictive Analysis

Predictive analysis uses historical data to predict future outcomes through statistical algorithms and machine learning. It is commonly used in finance, network security, healthcare, retail, and marketing. A few ways it is applied are by detecting financial fraud, predicting machine or system breakdowns, or predicting weather. Other tools are often used in combination with InfluxDB to achieve this, such as InfluxDB task, TensorFlow, algorithms, and machine learning.

Stock Market

Stock market data often has an enormous amount of time-series data that needs to be stored because it has data on stock prices that fluctuate throughout the day or the actual stock price of each day. With InfluxDB, you can store and gather stock data more efficiently. You could observe the volume of stocks traded over time, which could help detect stock manipulation using machine learning or algorithms. You could also use time-series algorithms to predict future stock prices.

Healthcare

Healthcare data often needs to be stored over time, such as the height of a growing kid, blood pressure or tumor size over time, or behavioral changes. It is often collected and stored by a patient's healthcare provider when they visit for appointments or check-ups. Smart devices can also often track a person's health, such as heart rate, sleep, or how active a person is. With the help of machine learning models or algorithms, it can help detect unusual patterns or predict changes in a person's health.

Space Industry

The space industry requires many novel technologies to explore the vastly unknown galaxies. Time-series data is used to help track the location and signal changes and quality in satellites over time. It can be used to help predict signal interface and weather such as solar flares or meteor showers. It can be used to help plan space exploration and create the infrastructure needed by determining when it should happen, trajectories and metrics that need to be met, and making sure the machine systems function properly. Earth's environment and climate, such as sea levels or Earth's temperature, can also be collected, analyzed, and predicted more efficiently.

Retail and E-Commerce

As consumers split their time shopping between retail and e-commerce stores, businesses can collect time series data and monitor real-time foot traffic, time spent in-store or online, and items bought. This can help businesses accurately keep track of consumer spending to make business decisions. With this data, businesses can determine how and what products should be marketed, what are the most popular products during a certain time or season, and how to enhance a customer's experience.

Basic tutorial

Install InfluxDB

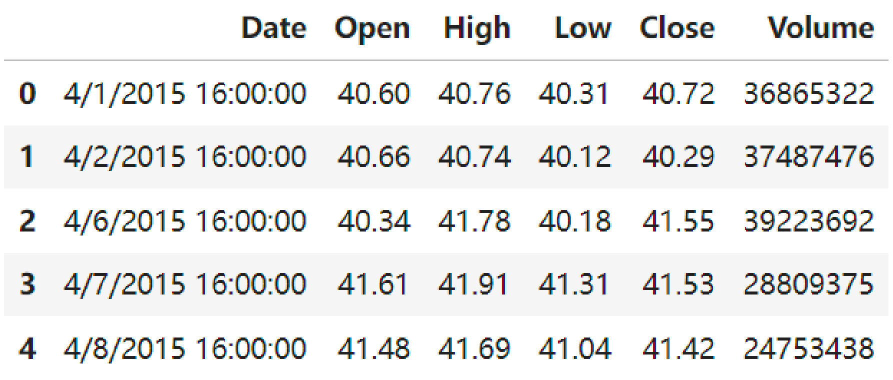
Step 1: Go to https://docs.influxdata.com/influxdb/v2/install/ , choose the corresponding system of your computer, and manually download InfluxDB.

Step 2: After unzipping the downloaded file to your computer, keep following the instructions about starting/running the InfluxDB in the website provided in Step 1.

Step 3: in your browser, go to http://localhost:8086, if successful, you should see the UI of influxDB! Congratulations! If this is the first time you use influxDB, you will also be asked to create an account, which will be used in later login.

How to use InfluxDB

In this tutorial, we will be using Microsoft stock price data from Kaggle, you could download the data from this link:
https://www.kaggle.com/datasets/vijayvvenkitesh/microsoft-stock-time-series-analysis/data .



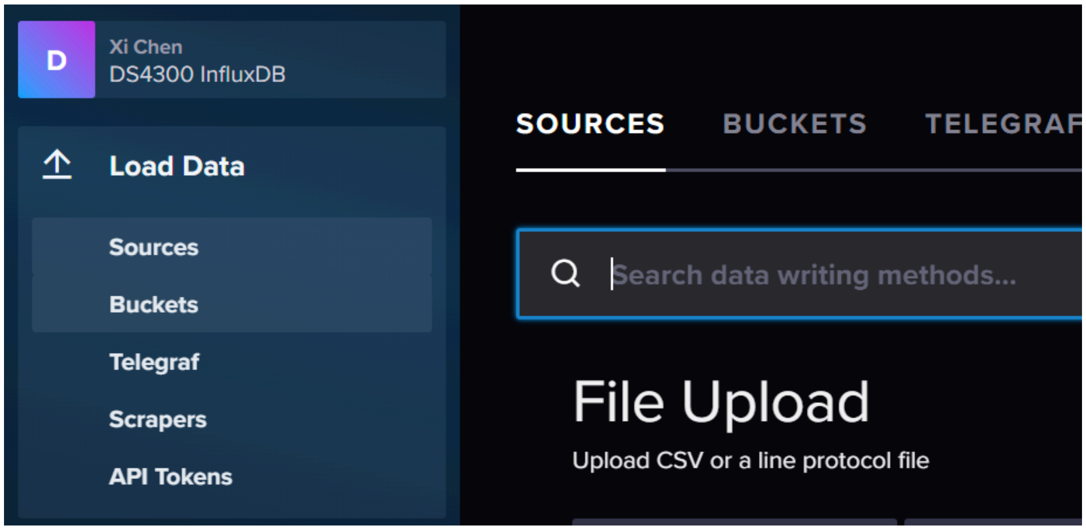
	Date	Open	High	Low	Close	Volume
0	4/1/2015 16:00:00	40.60	40.76	40.31	40.72	36865322
1	4/2/2015 16:00:00	40.66	40.74	40.12	40.29	37487476
2	4/6/2015 16:00:00	40.34	41.78	40.18	41.55	39223692
3	4/7/2015 16:00:00	41.61	41.91	41.31	41.53	28809375
4	4/8/2015 16:00:00	41.48	41.69	41.04	41.42	24753438

Write Data into InfluxDB

There are three ways to write data into InfluxDB:

1. Directly upload the annotated CSV in the InfluxDB UI
2. Use line protocol (learn more about what line protocol is in https://docs.influxdata.com/influxdb/cloud/reference/syntax/line-protocol/)
3. Use InfluxDB CLI (detailed instructions in InfluxDB UI -> Load Data -> InfluxCLI)

You can find these three options after you click "Load Data" in the left side menu of InfluxDB UI.



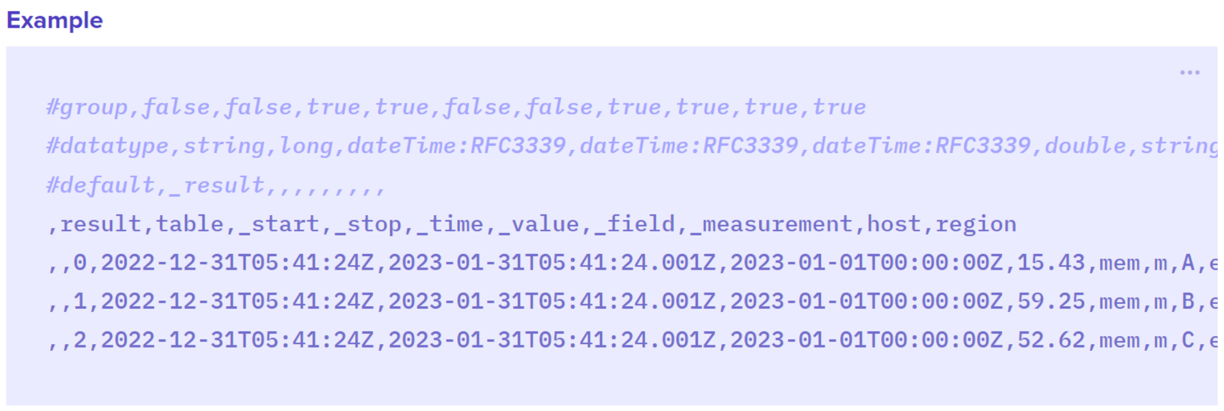
In this tutorial, we will be using the first option, loading annotated csv from InfluxDB UI.

InfluxDB UI does not support raw csv files. In other words, you cannot upload a csv file from Kaggle without annotating the data. Different from raw csv files, annotated csv file not only contains the data itself but also includes information about the data, the key components are:

1. Annotation row: describe the column properties, for instance, data types of column. There are three parts of annotation row, which are:
   a. Group: indicates if a column is a part of the group key, uses true/false

    b. Datatype: type of data of each column, such as long, string, double etc.
    c. Default: the default value of each column
2. Header row, describe the labels of columns. There are **mandatory** labels (without these labels, InfluxDB cannot read it), such as:
    a. _time: time in dateTime:RFC3339 format, such as “2015-04-01T16:00:00Z”
    b. _measurement: what is being measured, for example, stock price, weather, sleep behavior.
    c. _field: category of what we are measuring, for example, if we are measuring weather, then what is humidity, temperature, oxygen level etc.
    d. _value: actual number of fields, for example, what is the actual temperature in that period.
3. Record row: the actual data

Before we manipulate our data, let’s look at the official example of annotated csv:

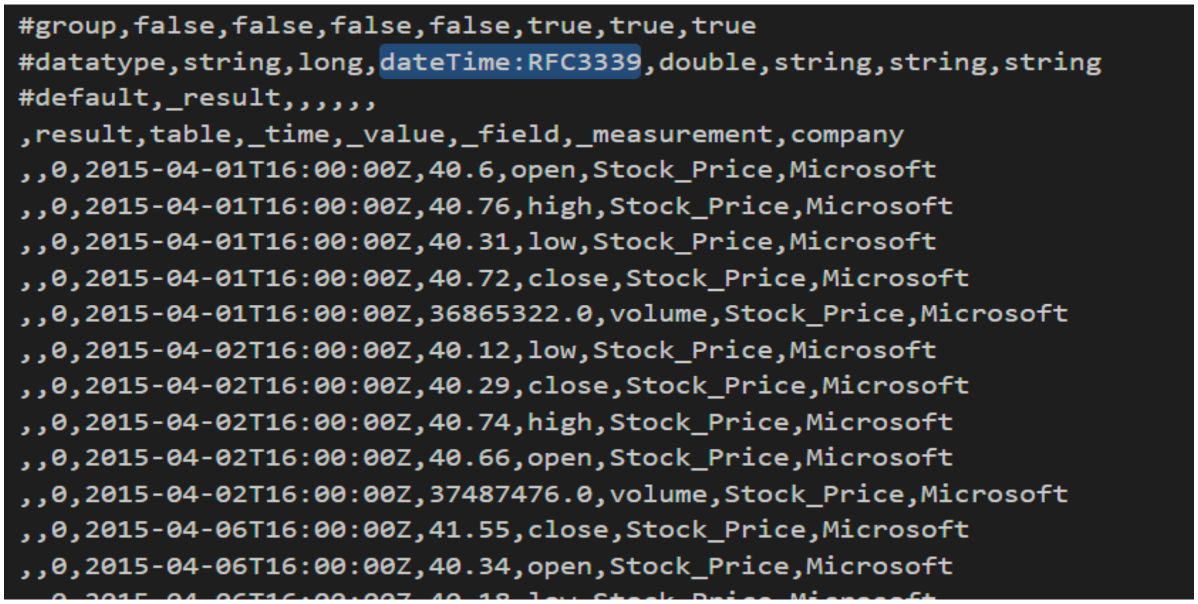


Example

```
#group,false,false,true,true,false,false,true,true,true,true
#datatype,string,long,dateTime:RFC3339,dateTime:RFC3339,dateTime:RFC3339,double,string
#default,_result,,,,,,,,,
,result,table,_start,_stop,_time,_value,_field,_measurement,host,region
,,0,2022-12-31T05:41:24Z,2023-01-31T05:41:24.001Z,2023-01-01T00:00:00Z,15.43,mem,m,A,e
,,1,2022-12-31T05:41:24Z,2023-01-31T05:41:24.001Z,2023-01-01T00:00:00Z,59.25,mem,m,B,e
,,2,2022-12-31T05:41:24Z,2023-01-31T05:41:24.001Z,2023-01-01T00:00:00Z,52.62,mem,m,C,e
```

Our goal is to make our data look like the example above. Let’s start! Open the Colab notebook here:
https://colab.research.google.com/drive/1gDsrrBi_oPSJt6BsgK9-amlJNlAKOA6M?usp=sharing

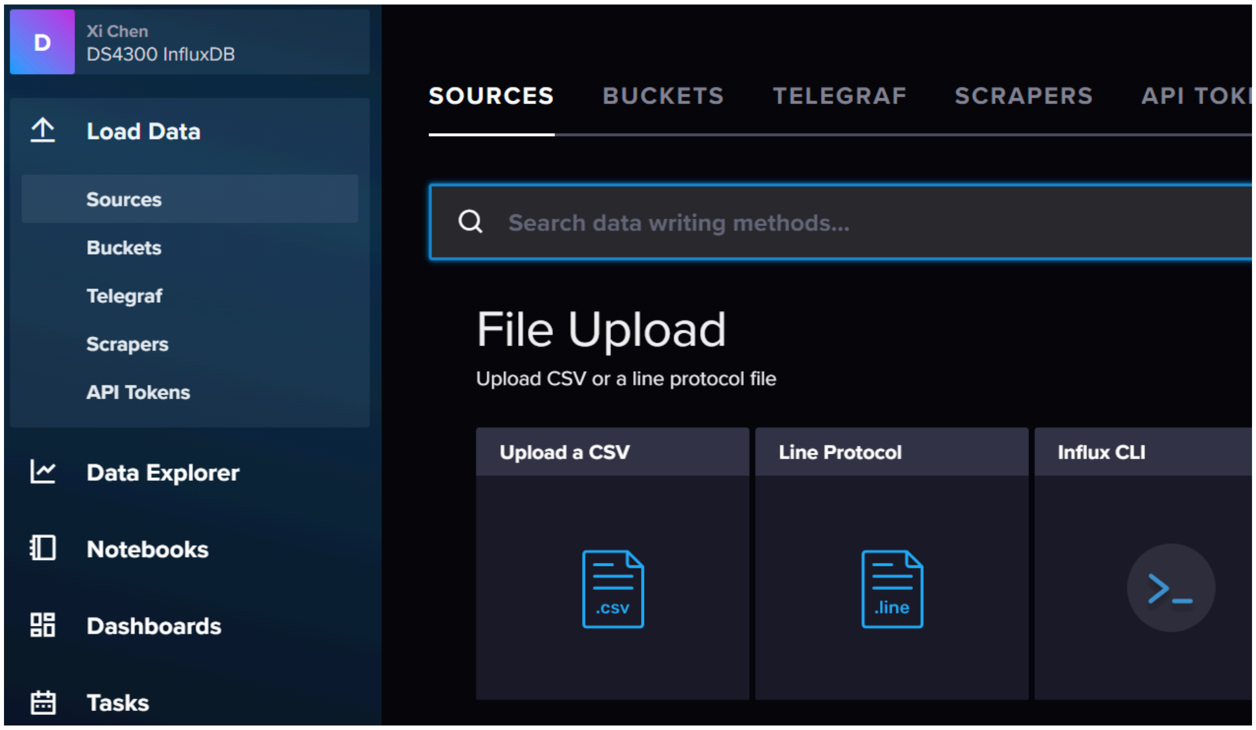
If you finished running the notebook and following the instructions, you should get the following results:



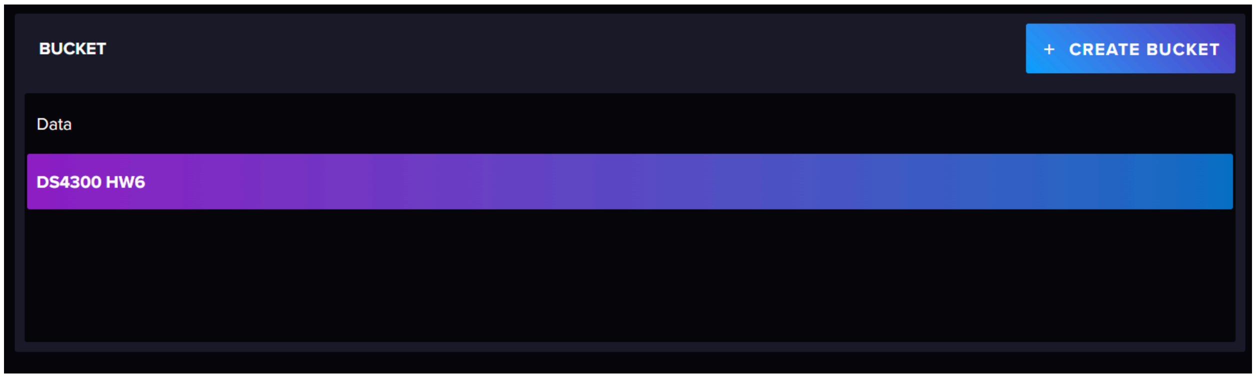
```
#group,false,false,false,false,true,true,true
#datatype,string,long,dateTime:RFC3339,double,string,string,string
#default,_result,,,,,,
,result,table,_time,_value,_field,_measurement,company
,,0,2015-04-01T16:00:00Z,40.6,open,Stock_Price,Microsoft
,,0,2015-04-01T16:00:00Z,40.76,high,Stock_Price,Microsoft
,,0,2015-04-01T16:00:00Z,40.31,low,Stock_Price,Microsoft
,,0,2015-04-01T16:00:00Z,40.72,close,Stock_Price,Microsoft
,,0,2015-04-01T16:00:00Z,36865322.0,volume,Stock_Price,Microsoft
,,0,2015-04-02T16:00:00Z,40.12,low,Stock_Price,Microsoft
,,0,2015-04-02T16:00:00Z,40.29,close,Stock_Price,Microsoft
,,0,2015-04-02T16:00:00Z,40.74,high,Stock_Price,Microsoft
,,0,2015-04-02T16:00:00Z,40.66,open,Stock_Price,Microsoft
,,0,2015-04-02T16:00:00Z,37487476.0,volume,Stock_Price,Microsoft
,,0,2015-04-06T16:00:00Z,41.55,close,Stock_Price,Microsoft
,,0,2015-04-06T16:00:00Z,40.34,open,Stock_Price,Microsoft
```

Now we are ready to import the data in InfluxDB UI!

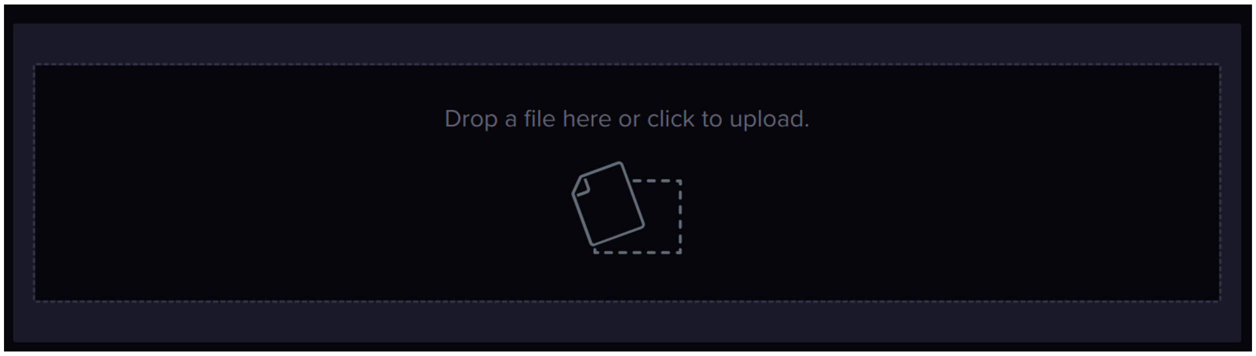
Open up the UI and select "Upload a CSV" from "Load Data":



Then, create a bucket if you don't have one yet. A bucket is a named location where time-series data is stored. Here I created a bucket called "DS4300 HW6".



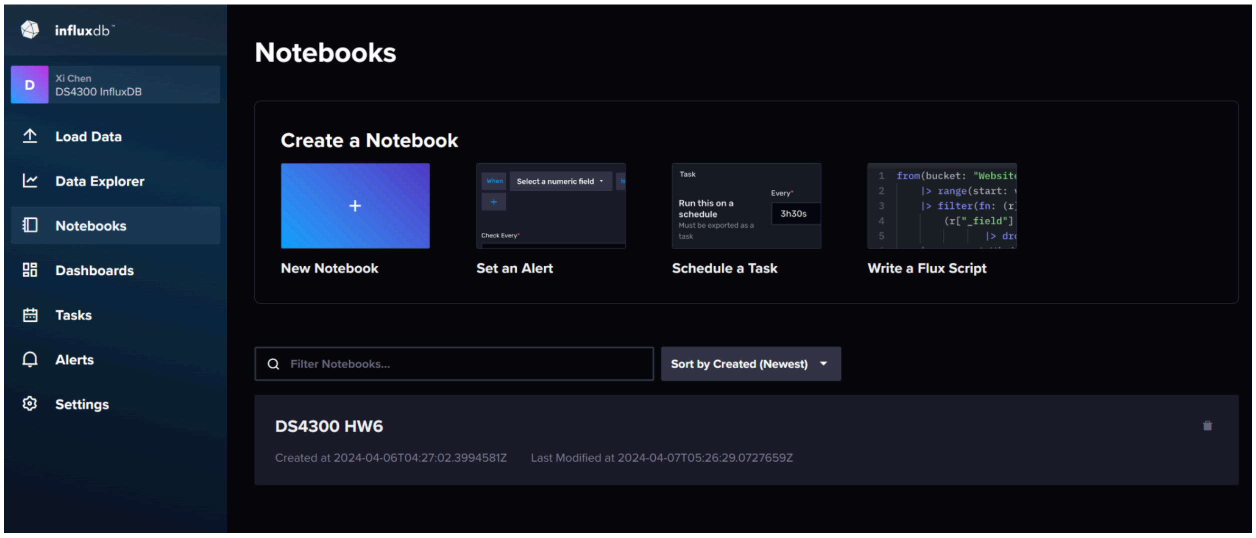
Next step, upload or drop the csv in this area.



You should be able to see "successful" on your screen!

Analyze Time-Series Data in InfluxDB

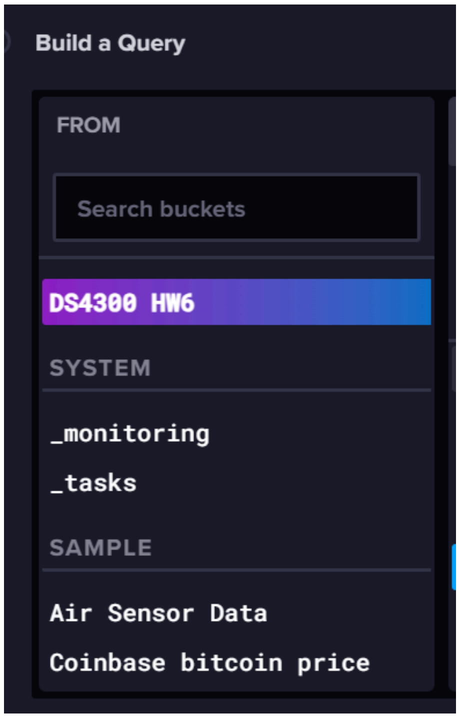
After we imported the data, the next thing we could do is data analysis. On the left menu of InfluxDB UI, click "Notebooks" and create a new notebook.



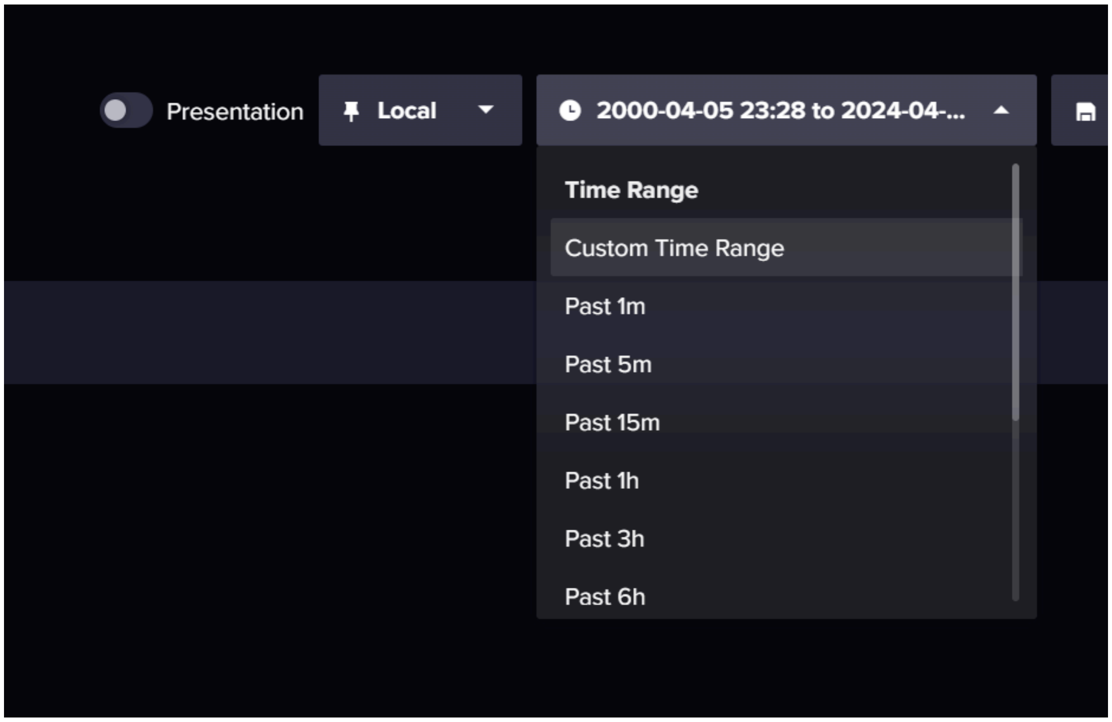
DS4300 HW6

Created at 2024-04-06T04:27:02.3994581Z Last Modified at 2024-04-07T05:26:29.0727659Z

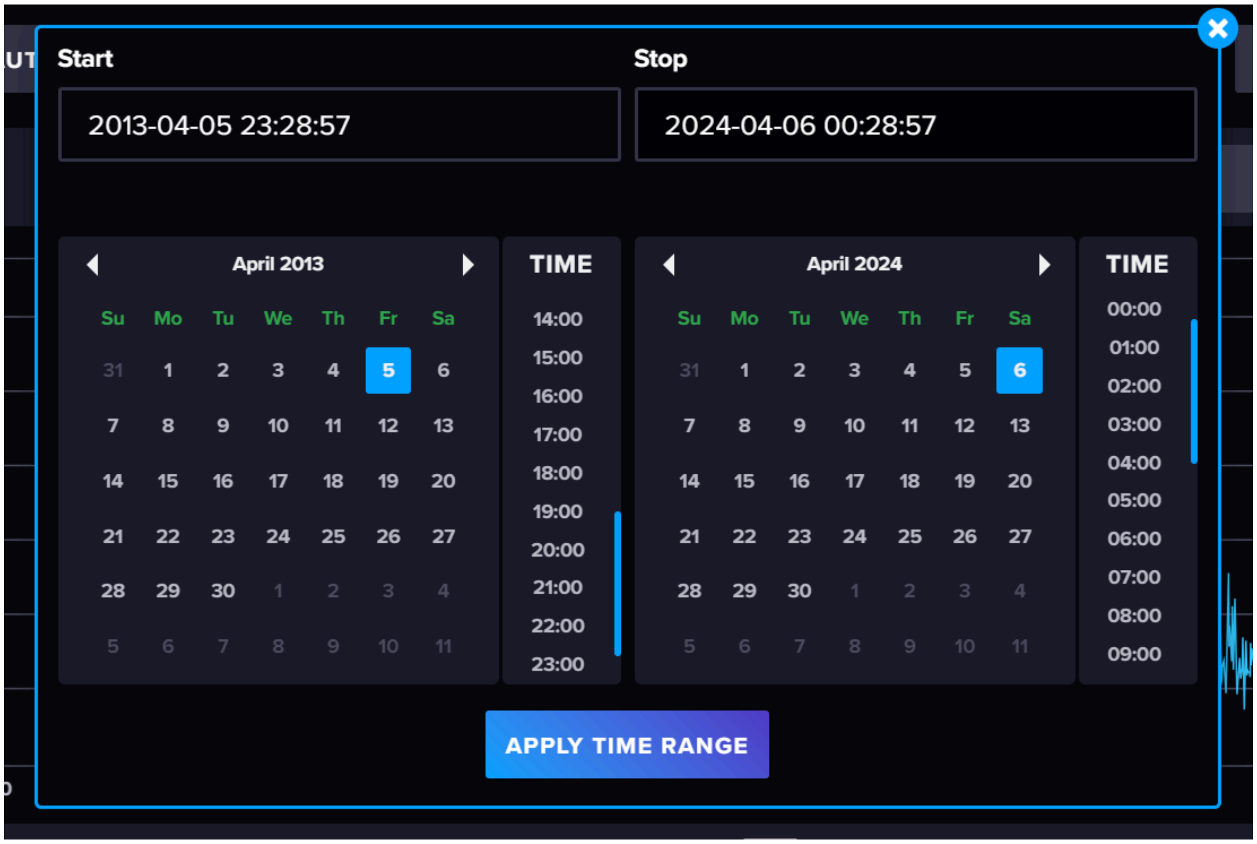
After you enter the notebook, first select the bucket we created in the earlier part of tutorial, which is called “DS4300 HW6”



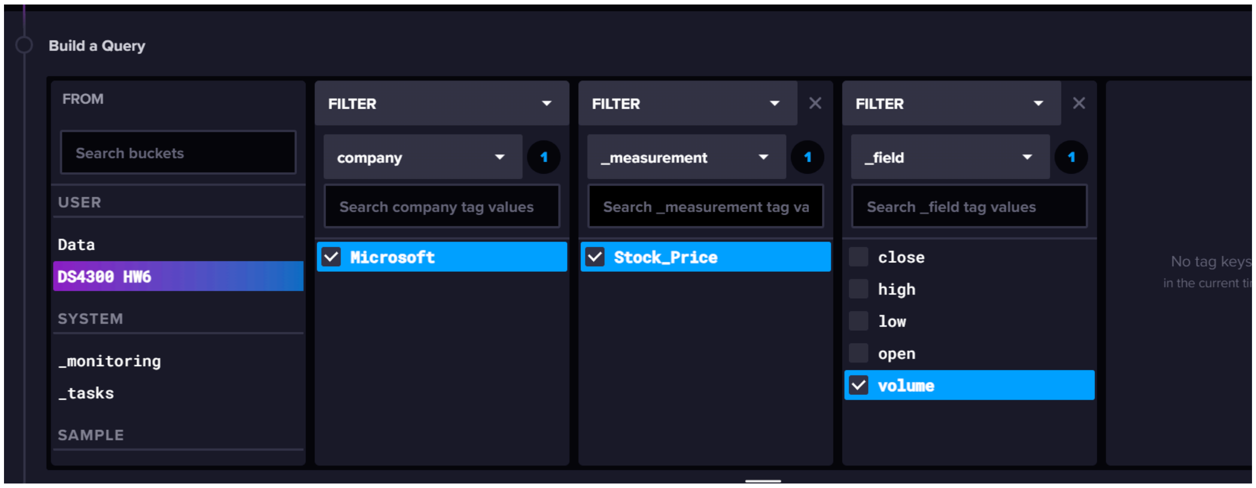
Then, select custom time range in the top right corner



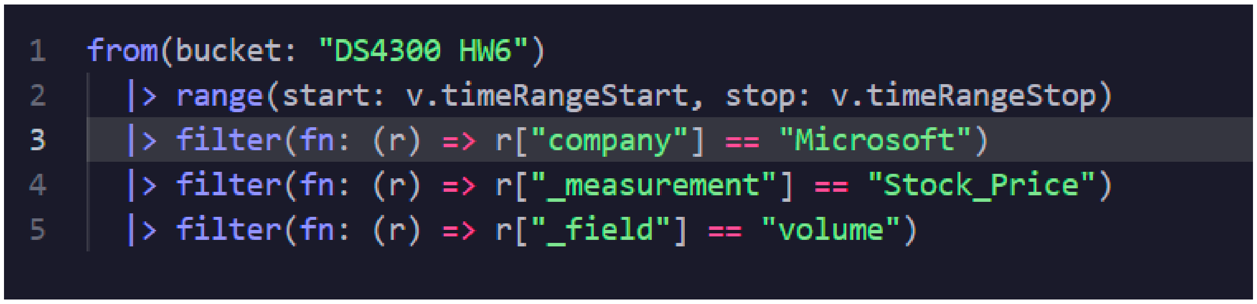
Change the start time from 2024 to 2013. InfluxDB drops data based on a given time range so that if our data doesn't exist in the time range, we could not observe any data points. Since our data starts from some time in 2014, we could set the start time in 2013 to make sure we have all the data showing. After you change the start time, click "APPLY TIME RANGE".



Let’s say we want to investigate how the volume (number of shares traded in that day) changes for Microsoft stock. For the first filter, we choose “company” and check “Microsoft” to identify which company we want to learn from. For the second filter, we chose “_measurement” and “Stock_Prices” because we want to know about Microsoft stock. Finally, choose “_field” and “volume”. During this process, InfluxDB simultaneously writes flux queries for us. After you are done, click “Run”.

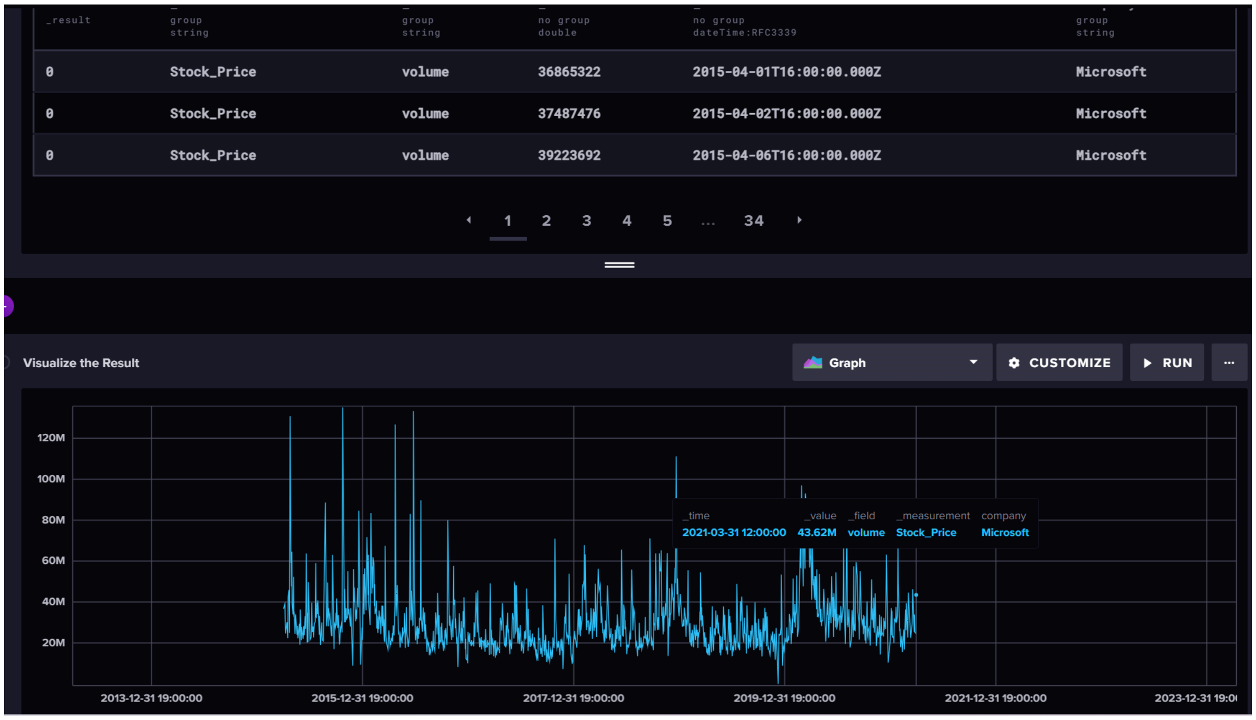


If you want to write the flux query by yourself, you see that it uses v.timeRangeStart as start and v.timeRangeStop as stop because we are using the UI and they refer to the time range we set previously. If we are working in Python, this should be changed to something more intuitive. You can find an example later:



```
from(bucket: "DS4300 HW6")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["company"] == "Microsoft")
  |> filter(fn: (r) => r["_measurement"] == "Stock_Price")
  |> filter(fn: (r) => r["_field"] == "volume")
```

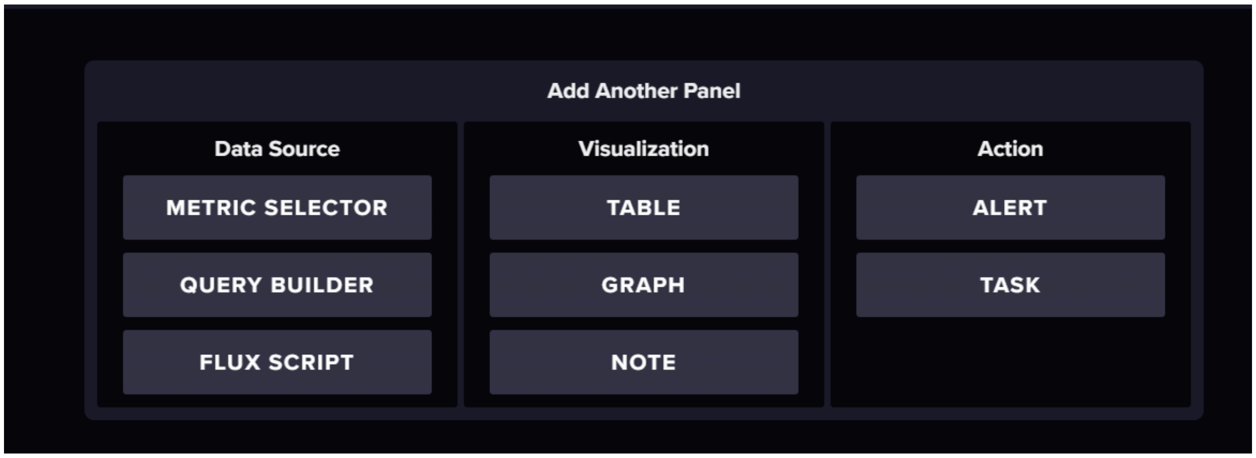
You should see a table and a line plot like this:



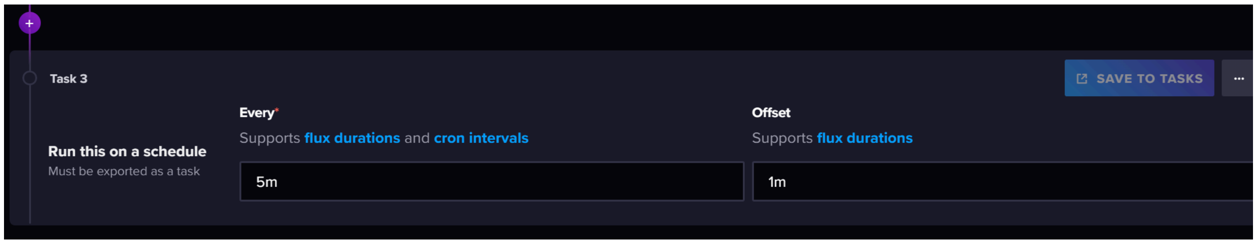
_result	group string	group string	no group double	no group dateTime:RFC3339	group string
0	Stock_Price	volume	36865322	2015-04-01T16:00:00.000Z	Microsoft
0	Stock_Price	volume	37487476	2015-04-02T16:00:00.000Z	Microsoft
0	Stock_Price	volume	39223692	2015-04-06T16:00:00.000Z	Microsoft

_time	_value	_field	_measurement	company
2021-03-31 12:00:00	43.62M	volume	Stock_Price	Microsoft

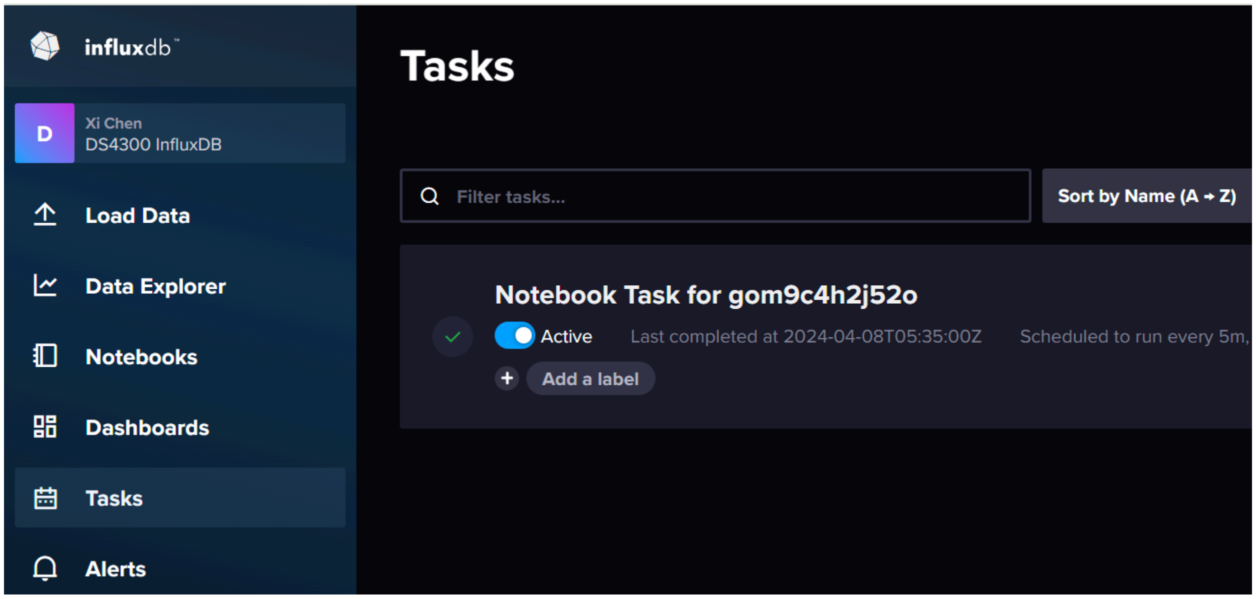
Moreover, what if we want to run this query simultaneously? For instance, we are interested in the close price of Microsoft stock in the last three days, which is helpful for analyzing the performance of their stock. First thing we need to do is to reconstruct the query we previously made, including changing the time range to “past 3d” and “_field” to “close”. However, we want to do this everyday and one click on “run” seems to be complex if we have 1000+ other “runs” to be clicked. InfluxDB offers a convenient option for us, called “task”. You could find it at the bottom of the notebook: Add Another Panel -> Action -> TASK



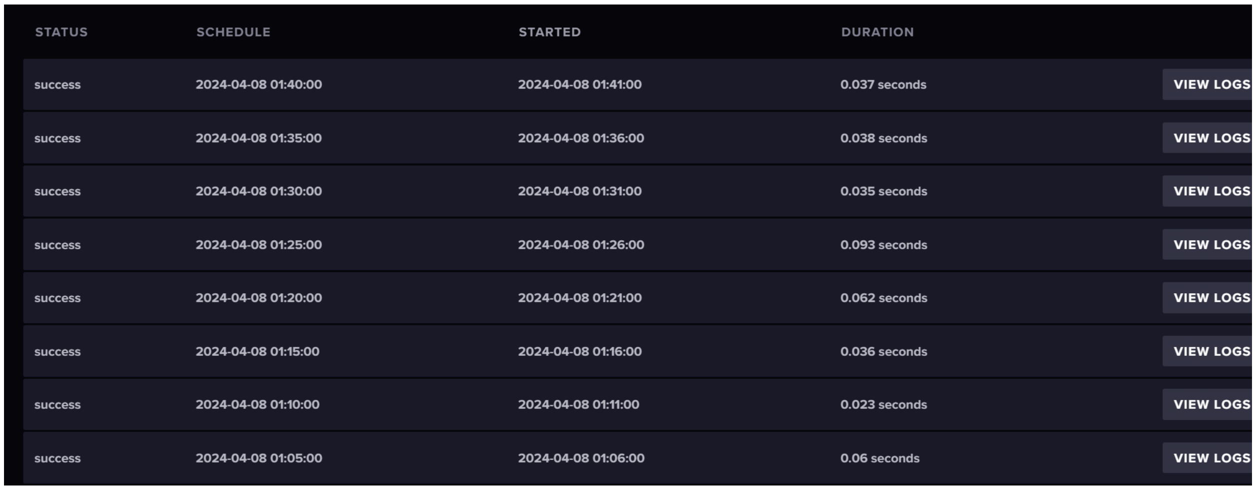
Then you should be able to see this block. There are two values to be filled in, which are “Every” and “Offset”. “Every” corresponds to how frequently you want to run this query and “Offset” serves as an extra wait time, which might be helpful when your data needs more time to be finalized when “Every” is up. In this tutorial, we use “5m” for “Every” and “1m” for “Offset” so that you don’t need to wait a day to see what would happen!



After a while, you could select “Tasks” in the menu of InfluxDB UI and select the task you just created.



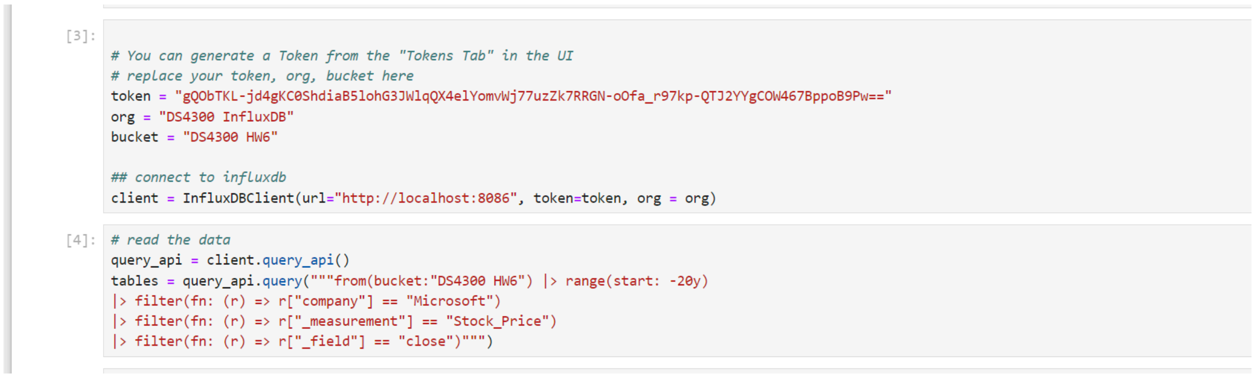
As you can see in the below screenshot, the query automatically runs for every 6 minutes! The “Tasks” tool would be more powerful when we have on-going data. Imagine we have new data automatically added to our database and a query is automatically analyzing it, even more, visualizing the results for you. Although we haven’t covered how to automatically add the data and utilize the result from auto-query, this could be done by “Telegraf” and “Dashboards” in InfluxDB.



STATUS	SCHEDULE	STARTED	DURATION	
success	2024-04-08 01:40:00	2024-04-08 01:41:00	0.037 seconds	VIEW LOGS
success	2024-04-08 01:35:00	2024-04-08 01:36:00	0.038 seconds	VIEW LOGS
success	2024-04-08 01:30:00	2024-04-08 01:31:00	0.035 seconds	VIEW LOGS
success	2024-04-08 01:25:00	2024-04-08 01:26:00	0.093 seconds	VIEW LOGS
success	2024-04-08 01:20:00	2024-04-08 01:21:00	0.062 seconds	VIEW LOGS
success	2024-04-08 01:15:00	2024-04-08 01:16:00	0.036 seconds	VIEW LOGS
success	2024-04-08 01:10:00	2024-04-08 01:11:00	0.023 seconds	VIEW LOGS
success	2024-04-08 01:05:00	2024-04-08 01:06:00	0.06 seconds	VIEW LOGS

Advanced: Microsoft Stock Price forecast with InfluxDB and Pytorch
(https://colab.research.google.com/drive/1AoTlIYlhsCxHam8u6MKzK7IGolbz_Ka-?usp=sharin g)
In short, what we do in this notebook are:

- Load data from InfluxDB using the InfluxDB-client libraries

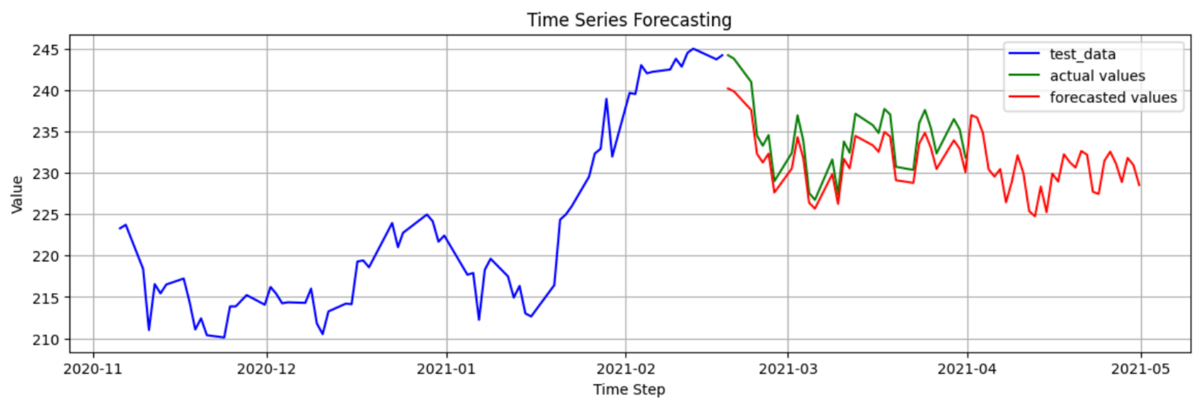


```
[3]:
# You can generate a Token from the "Tokens Tab" in the UI
# replace your token, org, bucket here
token = "gQObTKL-jd4gKC0ShdiaB5lohG3JWlqQX4elYomvWj77uzZk7RRGN-oOfa_r97kp-QTJ2YYgCOW467BppoB9Pw=="
org = "DS4300 InfluxDB"
bucket = "DS4300 HW6"

## connect to influxdb
client = InfluxDBClient(url="http://localhost:8086", token=token, org = org)
```

```
[4]:
# read the data
query_api = client.query_api()
tables = query_api.query("""from(bucket:"DS4300 HW6") |> range(start: -20y)
|> filter(fn: (r) => r["company"] == "Microsoft")
|> filter(fn: (r) => r["_measurement"] == "Stock_Price")
|> filter(fn: (r) => r["_field"] == "close")""")
```

-
- Use Pytorch to create a LSTM model to do time-series forecasting and visualize the actual vs predicted values, at the end, we would get the following result:



-

As you can see, InfluxDB is very convenient for storing time-series data and supporting forecasting on our data, which significantly helps decision making in a fast-moving market.

Analysis, Strengths, and Weaknesses

Because of its straightforward user interface, multiple methods of analysis, ability to automate, and scalability, InfluxDB is a highly useful database for the storage and analysis of time series data. Through the process of querying, it is possible to calculate different values, such as sums and averages over any given amount of time. The advantage of this database is that along with each query, InfluxDB also displays a graphical representation of the output of the query, showcasing the change in the data over time through multiple different types of graphs. The user interface of InfluxDB makes querying quite simple, as selecting a bucket and filtering only requires the use of drop down boxes, and the automatic visualizations streamline the data analysis process. Tasks also allow for the automation of the querying process, which is highly useful for running consistent analysis over time for a large amount of different items being updated continuously (ex. different stocks in a dataset). Because InfluxDB is scalable, it is able to handle this continuously updating data while maintaining performance.

InfluxDB has garnered significant recognition across industries, becoming adopted for many uses rapidly and developing a very strong user community. Some of their business customers include WayFair, IBM, ebay, and Cisco. One reason for this is because InfluxDB offers data portability, data log access, and data anonymization, which many of its competitors do not have. Moreover, InfluxDB is open-source and offers Database as a Service (DBaaS), meaning that users can access a cloud database without having to install or manage their own database software. It is so powerful because it can handle a greater average influx of data without sacrificing disk space, it has horizontal and elastic scalability through clustering, and it is more granular than other database systems. This allows database managers to better handle CPU, RAM, and disk space and easily complete capacity planning.

However, these many advantages do come with some tradeoffs. First, update and delete permissions within InfluxDB are highly restrictive. Also, InfluxDB prioritizes read and write requests over strong consistency, meaning that if there are many writes rapidly while querying, the query's results may not have the most recent data included. For InfluxDB, the entire dataset is prioritized as opposed to individual points, meaning that points do not have IDs and are instead identified by their timestamp and series. This severely restricts the ability to perform cross joins. Possibly the greatest limitation of this database system is that it assumes data sent multiple times is duplicated, thus not storing it twice. This can lead to data being overwritten in rare cases.

Overall, this database has a multitude of advantages with only a few drawbacks due to performance tradeoffs. The performance capabilities and its ease of use make it clear as to why InfluxDB has quickly developed a large user base.

Summary

InfluxDB emerges as a robust, user-friendly platform specifically designed to manage time-series data efficiently. Through its innovative data model structured around buckets, measurements, tags, and fields, it provides a flexible yet powerful schema to store and query time-series data. This schema facilitates the organization, retrieval, and analysis of data based on time, which is invaluable for predictive analysis, stock market analysis, healthcare monitoring, space industry logistics, and retail and e-commerce insights.

The tutorial section provides a practical glimpse into setting up InfluxDB and inserting and analyzing data, demonstrating the database's accessibility to users of varying expertise levels. The use cases highlighted throughout the document illustrate InfluxDB's versatility, showing its application in predictive analytics, monitoring stock market trends, managing healthcare data, aiding space exploration, and optimizing retail strategies.

InfluxDB is undeniably useful for applications requiring time-series data analysis. Its intuitive UI and straightforward query language, Flux, lower the barrier to entry for new users while offering advanced features for complex data analysis.

The platform's scalability is one of its key strengths, allowing it to handle massive influxes of data without significant degradation in performance. This scalability ensures that InfluxDB can grow with your project's needs, from small startups to large enterprises.

While InfluxDB excels in managing time-series data, it is essential to consider the specific needs of your project. The database's specialized focus means it may not be the best fit for every scenario, particularly where relational data management is paramount. However, for projects where time-series data is a core component, InfluxDB’s performance, ease of use, and extensive features make it an excellent choice.

After a thorough evaluation, it is recommended to consider InfluxDB as a potent solution for time-series data management in your development projects. Its ability to provide detailed insights into data over time can unlock significant value, making it a worthy addition to your technological toolkit.

References

Dix, P. (16, September 2014). One year of InfluxDB and the road to 1.0. InfluxData. Retrieved April 9, 2024, from https://www.influxdata.com/blog/one-year-of-influxdb-and-the-road-to-1-0/

Dotis, A. (2023, September 5). *Predictive Analytics Using a Time Series Database*. InfluxData. Retrieved April 7, 2024, from https://www.influxdata.com/blog/predictive-analytics-using-time-series-database/

Get InfluxDB. (n.d.). InfluxData. Retrieved April 9, 2024, from https://www.influxdata.com/get-influxdb/

Get started with influxdb. Get started with InfluxDB | InfluxDB OSS v2 Documentation. (n.d.). https://docs.influxdata.com/influxdb/v2/get-started/

IBM. (n.d.). *What is database as a Service (DBaaS)?*. IBM.com. https://www.ibm.com/topics/dbaas

InfluxData Inc. (2023, October 20). *InfluxDB's strengths and use cases applied in Data Science*. InfluxData.com. https://www.influxdata.com/blog/influxdb-strengths-use-cases-data-science/

influxdata/influxdb: Scalable datastore for metrics, events, and real-time analytics. (2023, September 21). GitHub. Retrieved April 9, 2024, from https://github.com/influxdata/influxdb

InfluxData Inc. (n.d.). *InfluxDB design principles*. InfluxDB design principles | InfluxDB OSS v2 Documentation. https://docs.influxdata.com/influxdb/v2/reference/key-concepts/design-principles/

InfluxDB (InfluxData). (n.d.). Clarify. Retrieved April 9, 2024, from https://www.clarify.io/integrations-browse/influxdb-influxdata

Information Today Inc. (2019, April 1). *DBA corner: Database scalability, elasticity, and availability*. Database Trends and Applications. https://www.dbta.com/Editorial/Think-About-It/DBA-Corner-Database-Scalability-Elasticity-and-Availability-130315.aspx

Install InfluxDB | InfluxDB OSS v2 Documentation. (n.d.). InfluxData Documentation. Retrieved April 8, 2024, from https://docs.influxdata.com/influxdb/v2/install/

Line protocol | InfluxDB Cloud (TSM) Documentation. (n.d.). InfluxData Documentation. Retrieved April 8, 2024, from https://docs.influxdata.com/influxdb/cloud/reference/syntax/line-protocol/

Meyers, J. (2023, August 22). Time Series Is out of This World: Data in the Space Sector. The New Stack. Retrieved April 8, 2024, from https://www.influxdata.com/blog/time-series-out-of-this-world-data-space-sector/

Pricing. (n.d.). InfluxData. Retrieved April 9, 2024, from https://www.influxdata.com/influxdb-pricing/

Time Series Forecasting With TensorFlow and InfluxDB. (2022, August 17). InfluxData. Retrieved April 7, 2024, from https://www.influxdata.com/blog/time-series-forecasting-with-tensorflow-influxdb/with-tensorflow-influxdb/

Time Series Forecasting Using Pytorch. (2023, September 25). Geeksforgeeks. Retrieved April 9, 2024, from https://www.geeksforgeeks.org/time-series-forecasting-using-pytorch/

Venkitesh, V. V. (2021). Microsoft Stock- Time Series Analysis. Kaggle. Retrieved April 8, 2024, from https://www.kaggle.com/datasets/vijayvvenkitesh/microsoft-stock-time-series-analysis/data