

# InfluxDB

## Installation

- [Install InfluxDB | InfluxDB OSS v2 Documentation \(influxdata.com\)](#)
- Download: <https://portal.influxdata.com/downloads/>

### Install Influx DB

```
# Red Hat/CentOS/Fedora
cat <<EOF | sudo tee /etc/yum.repos.d/influxdata.repo
[influxdata]
name = InfluxData Repository - Stable
baseurl = https://repos.influxdata.com/stable/$basearch/main
enabled = 1
gpgcheck = 1
gpgkey = https://repos.influxdata.com/influxdata-archive_compat.key
EOF

yum install influxdb2
```

```
# With Docker
# --reporting-disabled [REDACTED]
mkdir data

docker run \
  --name influxdb \
  -p 8086:8086 \
  --volume $PWD/data:/var/lib/influxdb2 \
  influxdb:2.7.4 --reporting-disabled
```

### Install Influx CLI

? InfluxDB 2.1 ???influx CLI ? influxDB ????????

Download: [Install and use the influx CLI | InfluxDB OSS 2.7 Documentation \(influxdata.com\)](https://docs.influxdata.com/influxdb/v2.7/get-started/install/#installing-the-client)

```
# amd64
wget https://dl.influxdata.com/influxdb/releases/influxdb2-client-2.7.1-linux-amd64.tar.gz
tar xvzf path/to/influxdb2-client-2.7.1-linux-amd64.tar.gz
mv influx /usr/local/bin/
```

## Start the service

```
systemctl start influxdb
systemctl status influxdb
systemctl enable influxdb
```

## Set up and initialize DB (v2.7+)

1. Visit `localhost:8086` in a browser
2. Create a user, bucket and organization names.
  - Initial username
  - Password
  - Initial organization name
  - Initial bucket name
3. The **API Tokens** will be generated.
4. Copy the generated token and store it for safe keeping.

## InfluxDB configuration

file: `/etc/influxdb/config.{toml,yaml,yml,json}`

```
# View the server configurations
influx server-config
```

config.toml:

- `/var/lib/influxdb/engine` ??????

```
bolt-path = "/var/lib/influxdb/influxd.bolt"
engine-path = "/var/lib/influxdb/engine"
```

## Optional: With Docker

```
docker exec -it influxdb influx config create --config-name local-admin --host-url http://localhost:8086 --org
<YOUR-ORG> --token <YOUR-TOKEN --active
```

```
docker cp influxdb:/etc/influxdb2/influx-configs ./

docker exec -it influxdb influx server-config > config.yml

docker run -p 8086:8086 \
  -v $PWD/config.yml:/etc/influxdb2/config.yml \
  -v $PWD/influx-configs:/etc/influxdb2/influx-configs \
  -v $PWD/data:/var/lib/influxdb2 \
  influxdb:2.7.4
```

## Set up the influx CLI (v2.7+)

????????? influx CLI ???????????????? config create ????????????

```
# Create config
influx config create --config-name <config-name> \
  --host-url http://localhost:8086 \
  --org <your-org> \
  --token <your-auth-token> \
  --active

influx config ls

# Enter InfluxQL shell
influx v1 shell
> show databases
> quit

# View the server configuration
influx server-config
```

## Schema Design

official tutorial: [InfluxDB schema design](#)

Data organization

- Bucket
  - Measurement
  - Tags
  - Fields
  - Timestamp

## Data Elements

- timestamp
- measurement
- tag key
- tag value
- field key
- field value

## Where to store data (tag or field)

### tag value

- ??????? string
- ????????
- ?????????????

### field value

- ??????? strings, floats, integers, or boolean
- ??? String???????????????
- ??? Integer????????????? `i`
- ??? GROUP BY ?????
- ????????????
- ?????????????????? field value ????????????????????
- field ?????? index????????????? field?????????

## ?????

# Measurement name with spaces

my\ Measurement fieldKey="string value"

# Double quotes in a string field value

myMeasurement fieldKey="\string\" within a string"

# Tag keys and values with spaces

myMeasurement,tag\ Key1=tag\ Value1,tag\ Key2=tag\ Value2 fieldKey=100

?????

Measurement names, tag keys, and field keys ?????  ???

# Management

Service

```
systemctl start influxdb
systemctl enable influxdb
systemctl stop influxdb
```

## InfluxDB v2.7

```
# InfluxQL shell
influx v1 shell
> show databases
> quit

# User management
influx user ls
## Create user
influx user create -n <username> -p <password> -o <org-name>
## Delete user
influx user delete -i <user-id>

# Bucket management
influx bucket ls
## Create bucket
influx bucket create --name <bucket-name> --org <org-name> --retention <retention-period-duration>
influx bucket delete -n <bucket-name>
## Rename the bucket
influx bucket update -i <bucket-id> --name <new-bucket-name>
## Update the retention
influx bucket update -i <bucket-id> --retention 90d

# Token management
influx auth ls
```

Case: ??????? vmware ? token ????? bucket vmware

```
# Create a user for the Org windbs
influx user create -n vmware -o windbs
influx user password -n vmware

# Create a token for the User vmware and <bucket-id>

influx auth create \
  --org windbs \
  --read-bucket 299f5d260eab27cc \
  --write-bucket 299f5d260eab27cc \
  --user vmware \
  --description "vmware's token"
```

Case: ???? Token ??? Org ?????

```
influx auth create \
  --org my-org \
  --description "my-org-all-access" \
  --all-access
```

Case: ???? Token ??? Org ? Operator ??

```
influx auth create \
  --org my-org \
  --description "my-org-operator" \
  --operator
```

## InfluxDB v1.8

Create a new user and database

```
influx
> CREATE USER admin WITH PASSWORD 'adminpass' WITH ALL PRIVILEGES
> exit

influx -username admin -password adminpass
> set password for admin = 'newpass'
>
```

```
> create database mmap_nmon with duration 180d
> create user mon with password 'thisispassword'
> grant ALL on mmap_nmon to mon
> show GRANTS for mon
```

## Login to InfluxDB

```
# In the shell
influx -username admin -password thisispass

# In the InfluxDB CLI
> auth
username:
password
```

## Logs (v2.7+)

```
journalctl -u influxdb
journalctl -n 50 -f -u influxdb
```

## Configure log level

Level: debug, info (default), error

/etc/influxdb/config.toml:

```
log-level = "info"
```

## Enable the Flux query log

/etc/influxdb/config.toml:

```
flux-log-enabled = true
```

## Flux (v2.x)

2024/1/24 ??: Flux language ?????????????? InfluxQL ? core SQL ??????????

????? Flux language ?????????????? Web ?? Data Explorer?

## List buckets

```
buckets()
```

## List all measurements in a bucket

```
import "influxdata/influxdb/schema"

schema.measurements(bucket: "example-bucket")
```

## List field keys

```
import "influxdata/influxdb/schema"

schema.fieldKeys(bucket: "example-bucket")
```

## List fields in a measurement

```
import "influxdata/influxdb/schema"

schema.measurementFieldKeys(
  bucket: "example-bucket",
  measurement: "example-measurement",
)
```

## List tags in a measurement

```
import "influxdata/influxdb/schema"

schema.measurementTagKeys(
  bucket: "example-bucket",
  measurement: "example-measurement",
)
```

## Filter by fields and tags

```
from(bucket: "example-bucket")
  |> range(start: -1h)
  |> filter(fn: (r) => r._measurement == "example-measurement-name" and r.mytagname == "example-tag-value")
```



```
|> filter(fn: (r) => r._field == "example-field-name")
```

??????

```
|> first()
|> last()
|> limit(n: 3)
```

## InfluxQL (v1.x)

```
> show databases
name: databases
name
----
_internal
nmon_reports
nmon2influxdb_log

> show users
user admin
---- ----
admin true
mon false

> use nmon_reports
Using database nmon_reports
> show measurements
name: measurements
name
----
CPU_ALL
DISKAVGRIO
DISKAVGWIO
DISKBSIZE
DISKBUSY
DISKREAD
DISKREADSERV
DISKRIO
```

```
DISKRXFER
```

```
DISKSERV
```

```
...
```

## Retention Policy

```
# for current database
```

```
> show retention policies
```

```
name    duration  shardGroupDuration replicaN default
```

```
-----
```

```
autogen 8760h0m0s 168h0m0s      1    true
```

```
# for specified database
```

```
> show retention policies on nmon2influxdb_log
```

```
name      duration  shardGroupDuration replicaN default
```

```
-----
```

```
autogen    0s    168h0m0s      1    false
```

```
log_retention 48h0m0s 24h0m0s      1    true
```

```
# Create a policy
```

```
> CREATE RETENTION POLICY "one_day_only" ON "NOAA_water_database" DURATION 1d REPLICATION 1
```

```
# Alter the policy
```

```
> ALTER RETENTION POLICY "what_is_time" ON "NOAA_water_database" DURATION 3w SHARD DURATION 2h  
DEFAULT
```

```
# Delete a policy
```

```
> DROP RETENTION POLICY "what_is_time" ON "NOAA_water_database"
```

## Verify the account

```
curl -G http://localhost:8086/query -u mon:thisispassword --data-urlencode "q=SHOW DATABASES"
```

```
{"results":[{"statement_id":0,"series":[{"name":"databases","columns":["name"],"values":[[{"mmap_nmon"}]]}]}
```

?????

```
SHOW MEASUREMENTS --[ ]
```

```
SHOW FIELD KEYS --[ ]
```

```
show field keys from <measurement-name>
```

```

SHOW series from pay --key
SHOW TAG KEYS FROM "pay" --key tag key
SHOW TAG VALUES FROM "pay" WITH KEY = "merId" --key tag key
SHOW TAG VALUES FROM cpu WITH KEY IN ("region", "host") WHERE service = 'redis'
DROP SERIES FROM WHERE = " --key
SHOW CONTINUOUS QUERIES --
SHOW QUERIES --
KILL QUERY --
SHOW RETENTION POLICIES ON mydb --
show series cardinality on mydb --

```

????

```

SELECT * FROM ./ / LIMIT 1 --
select * from pay order by time desc limit 2
select * from db_name."POLICIES name".measurement_name -- POLICIES name

```

????

```

delete from "query" --
drop MEASUREMENT "query" --delete
DELETE FROM cpu
DELETE FROM cpu WHERE time < '2000-01-01T00:00:00Z'
DELETE WHERE time < '2000-01-01T00:00:00Z'
DROP DATABASE "testDB" --
DROP RETENTION POLICY "dbbak" ON mydb --dbbak
DROP SERIES from pay where tag_key=" --key tag

```

????

- mean: ???
- sum: ??
- min: ???
- max: ???
- count: ???
- difference: ??? (????????)
- non\_negative\_difference: ??? (????????)

```

select * from pay order by time desc limit 2
select mean(allTime) from pay where time >= today() group by time(10m) tz('Asia/Taipei')
select * from pay tz('Asia/Taipei') limit 2

```

```
SELECT sum(allTime) FROM "pay" WHERE time > now() - 10s
select count(allTime) from pay where time > now() - 10m group by time(1s)

select difference("commit_sql") from "snapdb2" where time > now() - 1h limit 10
select non_negative_difference(*) from "snapdb2" where time > now() - 1h limit 10
select non_negative_difference(/commit_sql|rollback_sql/) from "snapdb2" where time > now() - 1h limit 10
```

## User and Privileges

```
> CREATE USER "todd" WITH PASSWORD '123456'

> SHOW GRANTS for "todd"

> SET PASSWORD FOR "todd" = 'newpassword'

> GRANT READ ON "NOAA_water_database" TO "todd"
> GRANT ALL ON "NOAA_water_database" TO "todd"

> REVOKE ALL PRIVILEGES FROM "todd"
> REVOKE ALL ON "NOAA_water_database" FROM "todd"
> REVOKE WRITE ON "NOAA_water_database" FROM "todd"

> DROP USER "todd"
```

## Covert timestamp to normal datetime

```
influx -precision rfc3339
# Or, in CLI
> precision rfc3339
```

## Convert InfluxData to csv format

```
# With CLI
# There is also useful -precision option to set format of timestamp.
influx -database 'database_name' -execute "SELECT * FROM table_name" -format csv > test.csv

influx -username your_user_if_any -password "secret!" -database 'db_name' -host 'localhost' -execute 'SELECT *
FROM "db_name"."your_retention_policy_or_nothing"."your_measurement_name" WHERE time > \'2017-09-
05\' and time < \'2017-09-05T23:59:59Z\' AND other_conditions_if_required' -format 'csv' >
/tmp/your_measurement_name_20170905.csv
```

```
# With HTTP-API
# Samples:
# "q=SELECT * FROM \"mymeasurement\" where time > now() - 130d"
# "q=SELECT * FROM \"mymeasurement\" where (time < now() - 130d) and (time > now() - 260d)"
curl -G 'http://localhost:8086/query' --data-urlencode "db=mydb" --data-urlencode "epoch=#timeunit" --data-urlencode "q=SELECT * FROM \"mymeasurement\" " -H "Accept: application/csv" > mytargetcsv.csv
```

## Tag Query

```
> show tag values from {measurement} with key={key}
> show tag values from {measurement} with key={key} where {tag-key}={tag-value}
```

## Learning

- [\[Influxdb\] influxdb????](#)
- [Use Grafana with InfluxDB v1.8 | InfluxDB OSS 1.8 Documentation \(influxdata.com\)](#)
- [Setting up InfluxDB v2 \(Flux\) with InfluxQL in Grafana | by Ivana Huckova | Medium](#)