编译源码

https://github.com/influxdata/influxdb/commit/5f6a17ffd0fd21b4ed83607628aef19abd36671b

依赖

rust

从 https://rustup.rs/ 获取脚本或下载工具安装。

其他安装方式见：
https://rust-lang.github.io/rustup/installation/other.html

安装完，运行 `cargo --version` 没报错就表示你已进入 rustlang 的大门。

protoc

安装 protobuf compiler 见 https://protobuf.dev/installation/

python3

ubuntu 系统：

```
sudo apt-get install python3 python3-pip python3-venv python3-dev
```

如果你安装了多个 python 版本，则需要使用 `export PYO3_PYTHON=/usr/bin/python3` 指定 python 的具体位置。

编译

```
git clone https://github.com/influxdata/influxdb.git
cd influxdb
cargo build
```

保证好你有个好网络，然后等待你的 CPU 发热，内存飙升。

hello influxdb

在源码目录运行以下命令启动服务端程序：

```
cargo run

# 或者

LOG_FILTER=debug cargo run
```

然后按以下步骤连接数据库，并执行 SQL 命令：

- 创建认证码并保存

```
cargo run -- create token --admin
export INFLUXDB3_AUTH_TOKEN="xxxxxx"
```

- 创建数据库

```
cargo run -- create database idb
```

- 执行 SQL

```
cargo run -- query -d idb 'select 1'
```

SQL 参考手册

探寻源码

Debug in VS Code

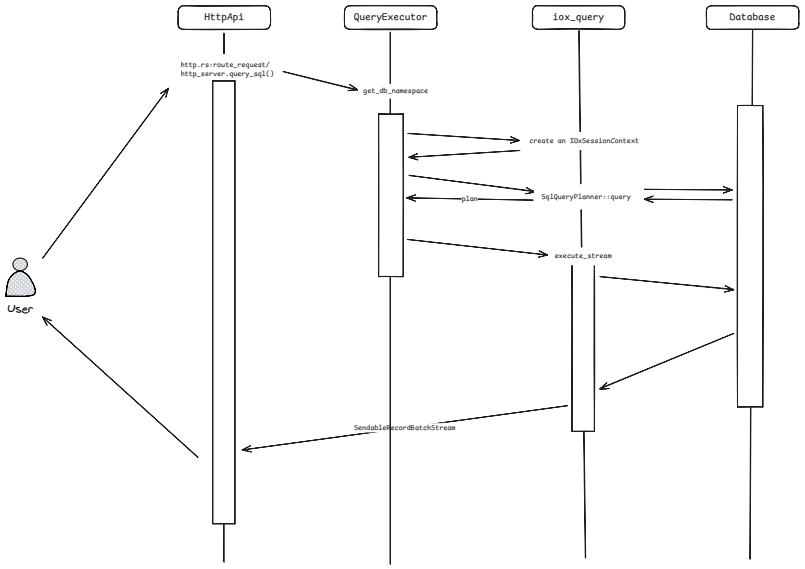
安装插件：rust-analyzer, codelldb

可以使用 attach 和 launch 两种方式调试。launch.json 如下：

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "attach to influxdb3",
            "type": "lldb",
            "request": "attach",
            "pid": "${command:pickProcess}",
            "stopOnEntry": false,
        },
        {
            "name": "Debug executable 'influxdb3'",
            "type": "lldb",
            "request": "launch",
            "cargo": {
                "args": [
                    "run",
                    "--bin=influxdb3",
                    "--package=influxdb3"
                ]
            },
            "args": []
        }
    ]
}
```

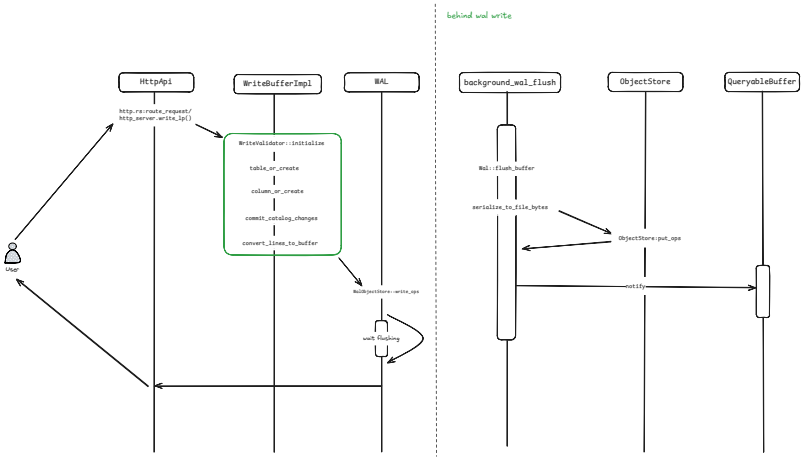
跟踪 select 语句的执行

```
target/debug/influxdb3 create database idb
target/debug/influxdb3 query -d idb 'select 1'
```

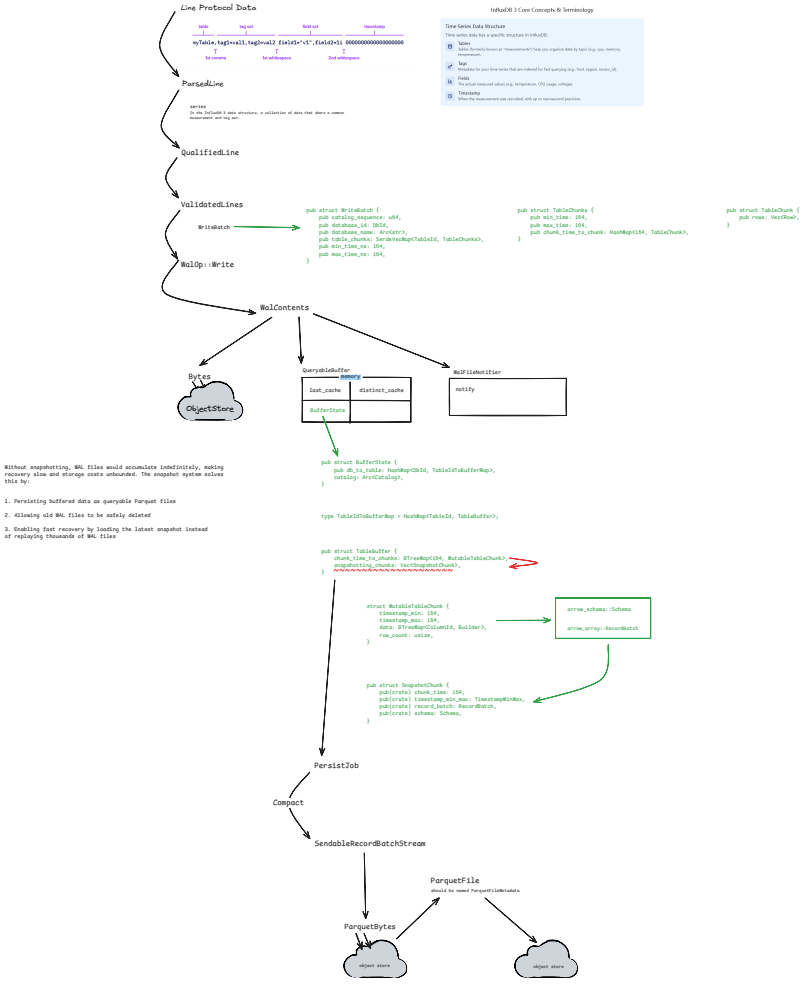


跟踪写流程

```
target/debug/influxdb3 write \
  --database idb \
  'home,room=LivingRoom temp=21.1,hum=35.9,co=0i 1761724800'
```



数据流图



Test

https://insta.rs/docs/quickstart/

> **snapshot test** >
>
> A snapshot test is basically a “freeze-frame” test. Instead of asserting a bunch of tiny details by hand, you capture the entire output of some code once, save it to a file, and from then on your test checks that the output hasn’t unexpectedly changed.
>
> The insta crate is Rust’s most popular tool for this.
>
> Here’s the idea in plain terms:
>
> You run your test the first time → it records the output into a snapshot file.
> Later runs → it compares the new output to the stored snapshot.
> If they differ → the test fails, and you can inspect the diff.
>
> It’s like version control, but for test expectations.

install tools:

```
cargo install nextest
curl -LsSf https://insta.rs/install.sh | sh
```

run all tests:

```
cargo nextest run --workspace
```

run single test:

```
cargo nextest list --workspace
cargo nextest run -E 'test(show_system)'
```

use `cargo insta review` to check failed tests.

参考

- https://docs.influxdata.com/influxdb3/core/
- https://deepwiki.com/influxdata/influxdb
- https://github.com/influxdata/influxdb/blob/main/CONTRIBUTING.md
- https://pyo3.rs/main/building-and-distribution.html