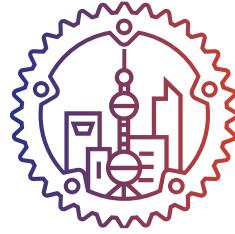




RUST CHINA CONF 2021 - 2022

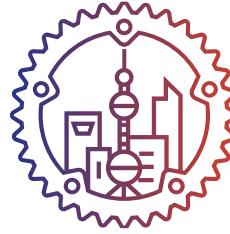
第二届中国Rust开发者大会

2022.07.31 Online

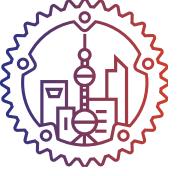


使用 Rust 构建云原生数仓 Databend

尚卓燃 ([PsiACE](#))



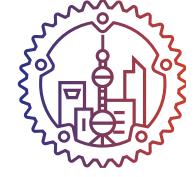
• Databend 简介	1
• 架构和设计	4
• What's New	9
• In Rust Way	13
• 社区	20
• Databend Cloud	24



Databend 简介

Databend 使用 Rust 研发
开源、完全面向云架构的新式数仓

Databend 是什么



Databend 是一个使用 Rust 研发、开源、完全面向云架构的新式数仓，致力于提供极速的弹性扩展能力，使大数据分析门槛进一步降低，让任何人都可以高效率、低成本地挖掘数据价值。



[Databend Cloud \(beta\)](#) | [Documentation](#) | [Benchmarking](#) | [Roadmap \(v0.8\)](#)

Slack [Join Databend](#) build passing Platform Linux, macOS, ARM License Apache 2.0

- website: <https://databend.rs>
- github: <https://github.com/datafuse-labs/databend>

Databend 特性



极致弹性

存算分离，实现资源的精细化调度，能够按需、按量弹性扩展。

卓越性能

数据级并行和指令级并行加持，支持大规模并行处理。

Git-like 存储引擎

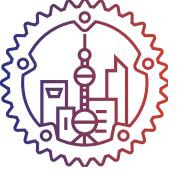
使用快照存储数据。查询、克隆和恢复历史数据轻而易举。

支持半结构化数据

内置 ARRAY, MAP, JSON 数据类型，轻松导入和操作半结构化数据，进一步挖掘数据价值

生态兼容

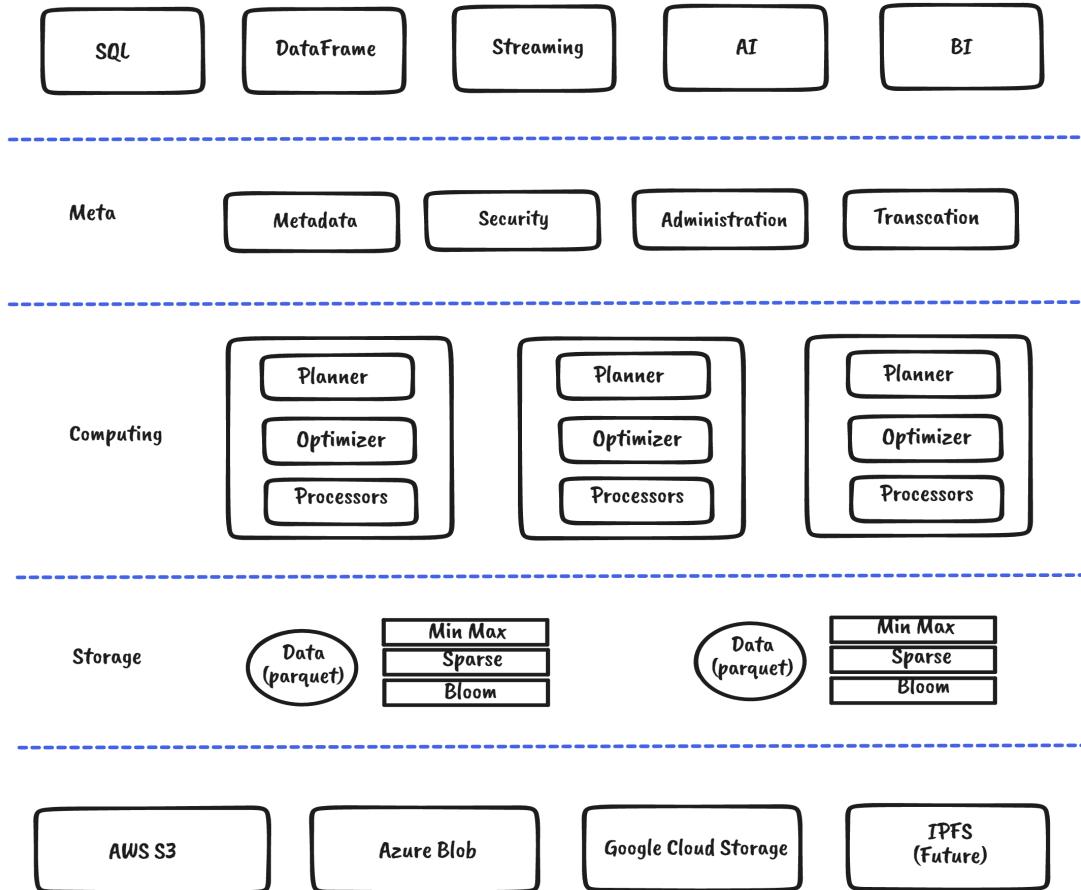
Databend 兼容 ANSI SQL，支持 MySQL 和 ClickHouse 客户端，打通现有工具和 BI 系统。



架构与设计

Databend 可以分为三层：
Meta, Computing, Storage

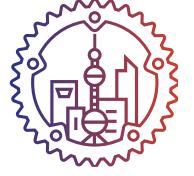
架构和设计



左图展示了 Databend 的整体架构，大致由三部分组成：

- Meta Layer
- Computing Layer
- Storage Layer

架构和设计 - *Meta*



Meta

Metadata

Security

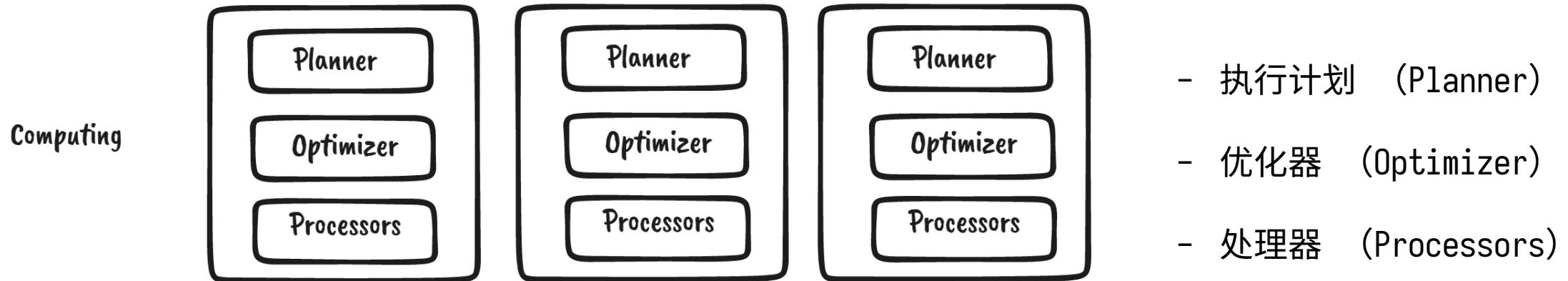
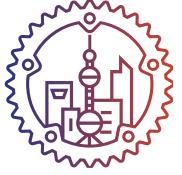
Administration

Transcation

Meta 是一个多租户、高可用的分布式 key-value 存储服务，具备事务能力。

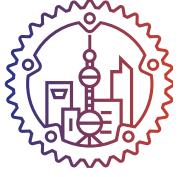
- 表的元信息、索引信息、集群信息、事务信息等。
- 用户系统、用户权限、使用统计等信息。
- 用户登录认证、数据加密等。

架构和设计 - Computing



计算层由多个集群 (cluster) 组成，不同集群可以承担不同的工作负载，每个集群又由多个计算节点 (node) 组成，可以轻松添加、删除节点或集群，做到资源的按需、按量管理。

架构和设计 - Storage

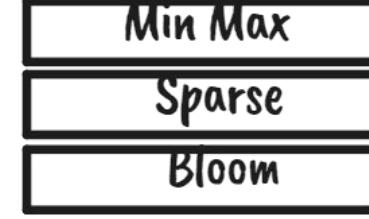


Storage

Data
(parquet)

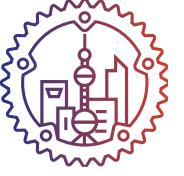


Data
(parquet)



Databend 使用 Parquet 列式存储格式来储存数据，
为了加快查找 (Partition Pruning) ，
Databend 为每个 Parquet 提供了自己的索引：min_max, sparse, bloom 。

存储层能够让数据自由流动，支持对接多种对象存储服务。



What's New

不止于性能，Databend 还关心这些

更加友好的查询体验



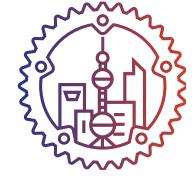
引入严格的语义检查环节，
在查询编译阶段拦截大部分错误。

- 语法错误，例如：
 - 写错了关键字
 - 遗漏了某些子句
- 语义错误，例如：
 - 使用了不存在的 Column
 - Column 具有歧义

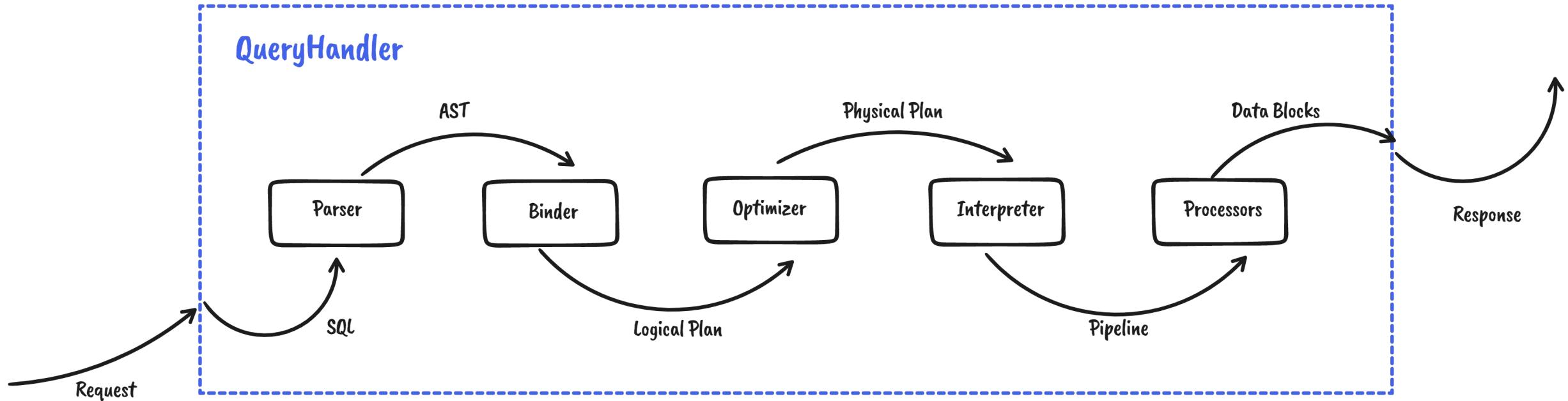
```
MySQL [(none)]> select number from numbers(10) where a = 1;
ERROR 1105 (HY000): Code: 1065, displayText = error:
--> SQL:1:38
|   select number from numbers(10) where a = 1
|                                         ^
|                                         column doesn't exist

MySQL [(none)]> select number from numbers(10) as t inner join numbers(10) as t1 using(number);
ERROR 1105 (HY000): Code: 1065, displayText = error:
--> SQL:1:8
|   select number from numbers(10) as t inner join numbers(10) as t1 using(number)
|                                         ^
|                                         column reference is ambiguous
|                                         .
```

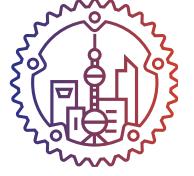
全新 Planner



- [New SQL Planner Framework Design | Databend](#)
- [Databend SQL Planner 全新设计](#)



类型安全的 Expression



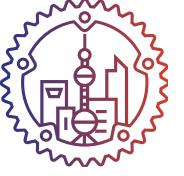
全新的表达式框架，包含一套形式化的类型系统。

- 类型检查。运行时不关注类型信息。
- 类型安全的向下转型（downcast）。
- Enum 分发，减少运行时开销，降低开发难度。
- 泛型，在函数签名中使用，减少手写的重载。

Learn More

- [RFC | Formal Type System](#)
- [Typed Type Exercise in Rust](#)
- [Tracking issue for new expression framework](#)

```
registry.register_2_arg::<BooleanType, BooleanType, BooleanType, _>(
    "and",
    FunctionProperty::default(),
    |lhs, rhs| lhs && rhs,
);
```



In Rust Way

关于使用 Rust 的一些实践和体会

快速迭代

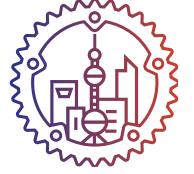


- [周刊（第7期）：一个C系程序员的Rust初体验 - codedump的网络日志](#)

- 6周更新一次 nightly 工具链
- 上游优先并及时更新依赖
- 与 fmt / clippy 等现有工具集成
- 利用生态中的好工具
 - cargo-{edit / udeps / audit}
- 探索可能对开发更友好的新工具
 - mold / cargo-hakari / nextest
 - goldenfiles / insta / enum-dispatch

```
Differences (-left|+right):
----- Output -----
'I'm who I'm.'
----- AST -----
Literal {
    span: [
-     QuotedString(0..18),
+     QuotedString(0..16),
    ],
    lit: String(
        "I'm who I'm.",
    ),
}
.cargo/git/checkouts/rust-goldenfile-6352648ef139d984/16c5783/src/differs.rs:15:5
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
```

测试风格



```
[lib]
doctest = false
test = false

[[bin]]
name = "databend-query"
path = "bin/databend-query.rs"
doctest = false
test = false
```

```
----- Input -----
5 * (a and ) 1
----- Output -----
error:
--> SQL:1:12
1 | 5 * (a and ) 1
| -          ^ expected more tokens for expression
| |
| while parsing expression
```

- [Rust, Databend and the Cloud Warehouse \(4\) Databend 社区如何做测试 \[虎哥的博客 \] \(bohutang.me\)](https://matklad.github.io>Delete Cargo Integration Tests (matklad.github.io)• <a href=)
- [如何为 Databend 添加新的测试 | Databend 内幕大揭秘 \(psiace.github.io\)](https://psiace.github.io/How-to-add-new-tests-to-Databend)

代码演进 - eg. 1



async-trait

```
● ● ●  
#[async_trait]  
pub trait AsyncSource: Send {  
    const NAME: &'static str;  
  
    async fn generate(&mut self) -> Result<Option<DataBlock>>;  
}
```

with GAT

```
● ● ●  
#![feature(generic_associated_types)]  
#![feature(type_alias_impl_trait)]  
  
use futures::Future;  
  
pub trait AsyncSource: Send {  
    const NAME: &'static str;  
    type BlockFuture<'a>: Future<Output = Result<Option<DataBlock>>> + Send  
        where Self: 'a;  
  
    fn generate(&mut self) -> Self::BlockFuture<'_>;  
}
```

unbox async (Now)

```
● ● ●  
#[async_trait::async_trait]  
pub trait AsyncSource: Send {  
    const NAME: &'static str;  
  
    #[async_trait::unboxed_simple]  
    async fn generate(&mut self) -> Result<Option<DataBlock>>;  
}
```

- [Skyzh | 2022-01-31-gat-async-trait/](#)
- [Allow 'async fns' to return 'impl Future'](#)
- [VVVVVVVVV's async-trait fork with unboxed](#)

代码演进 - eg.2



dyn datatype

```
● ● ●  
pub type DataTypePtr = Arc<dyn DataType>;  
  
#[typetag::serde(tag = "type")]  
pub trait DataType: std::fmt::Debug + Sync + Send + DynClone {}
```

enum-dispatch (Now)

```
● ● ●  
#[derive(Clone, Debug, serde::Deserialize, serde::Serialize)]  
#[serde(tag = "type")]  
#[enum_dispatch(DataType)]  
pub enum DataTypeImpl {  
    Null(NullType),  
    Nullable(NullableType),  
    ...  
}  
#[enum_dispatch]  
pub trait DataType: std::fmt::Debug + Sync + Send + DynClone {}
```

enum as inner (Future)

```
● ● ●  
pub type GenericMap<'a> = [DataType];  
  
#[derive(Debug, Clone, PartialEq, Eq, EnumAsInner)]  
pub enum DataType {  
    Boolean,  
    String,  
    UInt8,  
    ...  
}
```

- [enum_dispatch - Rust \(docs.rs\)](#)
- [feat\(expr\): add new crate `common-expression`](#)

踩坑小记 - eg. 1



```
● ● ●  
// Jaeger layer.  
let mut jaeger_layer = None;  
let jaeger_agent_endpoint =  
    env::var("DATABEND_JAEGER_AGENT_ENDPOINT").unwrap_or_else(|_| "".to_string());  
if !jaeger_agent_endpoint.is_empty() {  
    global::set_text_map_propagator(TraceContextPropagator::new());  
  
    let tracer = opentelemetry_jaeger::new_pipeline()  
        .with_service_name(app_name)  
        .with_agent_endpoint(jaeger_agent_endpoint)  
        .with_auto_split_batch(true)  
        .install_batch(opentelemetry::runtime::Tokio)  
        .expect("install");  
  
    jaeger_layer = Some(tracing_opentelemetry::layer().with_tracer(tracer));  
}
```

如果默认一直开着日志发送服务，
但集群里没开日志收集服务

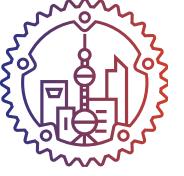
可能会一直 buffer 住这些没发的日志

踩坑小记 - eg.2



```
● ● ●  
  
async fn leader_metrics() -> Result<()> {  
    let span = tracing::debug_span!("leader_metrics");  
    let _ent = span.enter();  
  
    let all_members = btreeset![0, 1, 2, 3, 4];  
    let left_members = btreeset![0, 1, 2, 3];  
    ...  
}
```

- <https://github.com/tokio-rs/tracing#in-asynchronous-code>
- [Fixing Memory Leaks in Rust \(onesignal.com\)](https://onesignal.com)



社区

为共筑更好的 Rust 生态而努力。

开源项目



openraft

基于 tokio 运行时的异步共识算法实现，在 Raft 之上更进一步，旨在成为构建下一代分布式系统的基石。

目前已经应用在 SAP / Azure 的项目中。



opendal

从开源第一天就为连接数据世界做准备。让所有人都可以无痛、高效地访问不同存储服务。

近期的提案包括实现一个 oli 命令行工具，以操作不同服务中存储的数据，并支持数据迁移。



opensrv

为数据库项目提供高性能和高可靠的服务端协议兼容，建立在 tokio 运行时上的异步实现。

支持 MySQL 和 Clickhouse 协议，目前在 CeresDB 中得到应用。

课程

- [Rust 培养计划 \(25 期\)](#)
- [Rust 新手入门系列课程 \(9 期\)](#)

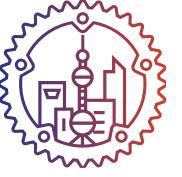
Databend 动态 投稿 76 合集和列表 4 搜索视频、动态

关注数 62	粉丝数 3043	获赞数 1188	播放数 5.6万	阅读数 2441
18-Databend 如何利用 Github 做全球协作开发 370 2021-11-22	17-探讨 Rust 智能指针 databend 1082 2021-11-15	16-通过 Futures 库分析加深对 Rust 异步机制的理解 866 2021-11-8	15-探讨为什么Pin在Rust异步编程中如此重要 Databend 1346 2021-11-4	新一代云原生数仓 Databend 架构及现状 1328 2021-10-25
02:02:14	02:13:31	02:03:33	01:55:19	01:14:41
14-探讨 Rust 异步编程框架 Mio Rust 培养提高计划 2739 2021-10-28	12-初探 Rust 微服务架构 Datafuse 1510 2021-10-18	11-实战：使用 Rust 开发动态链接库并在 Golang 中使用 Vol. 10 2045 2021-10-15	Rust 使用Redis给短链服务加速 Vol. 10 602 2021-10-11	基于Rust Axum-web 和 MySQL 开发短链服务 1888 2021-9-27
01:10:15	02:12:08	02:09:32	01:17:00	57:58
Dive into Rust Closure Rust 培养提高计划 第 7 期 1642 2021-9-9	Tokio 入门之运行时介绍 Rust 培养提高计划 3386 2021-9-6	Rust 异步编程入门-Futures (1) Rust 培养提高计划 2603 2021-8-30	通过 Datafuse 理解全链路跟踪 745 2021-8-22	认识面向基础架构语言Rust Vol 1 3104 2021-8-1
35:04				理解Rust的所有权 Vol 2 1943 2021-8-9

自 21 年 8 月起，Databend 和 Rust 中文社区、知数堂，启动了面向 Rustacean 和数据库开发人员的公开课计划。



上游优先



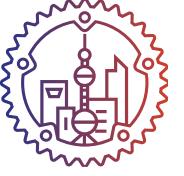
开源协作理所当然地需要将变更反馈给社区。

- 更新
- 沟通
- 贡献

[2022-25: 开源当以上游优先
\(xuanwo.io\)](https://xuanwo.io)

arrow2 是使用 transmute-free 操作重写的 Arrow 实现，Databend 的核心依赖

- 贡献者中有 9 位是 datafuse labs 成员 (9/62)
- 其中有三位是 top 15 贡献者, sundy-li 排第 4
- 实现对 BinaryArray 的计算支持
- 累计提高从 Primitive 到 Binary/String 转换性能近 8 倍



Databend Cloud

专注于云端大数据一站式解决方案



Databend Cloud 是基于开源云原生数仓项目 Databend 打造的一款易用、低成本、高性能的新一代大数据分析平台，让用户无需再关注任何资源，而是专注于数据价值的挖掘，打造的一个真正按需、按量付费的 Data Cloud。

Databend Cloud 专注于云端大数据一站式解决方案，以解决传统大数据项目中运维难，成本高，使用复杂的问题，让你更加专注数据价值的挖掘。

The image shows the official website for Databend Cloud. At the top right is the Databend logo, which consists of a blue whale icon followed by the word "Databend". Below the logo is the slogan "一站式数据分析云平台" (One-stop Data Analysis Cloud Platform). Underneath the slogan is the tagline "按需付费 | 易用 | 极致性能" (Pay-as-you-go | Easy-to-use | Extreme performance). A search bar at the bottom left contains the URL "app.databend.com" and a magnifying glass icon. The main content area displays three screenshots: a smartphone showing the "Start for Free" sign-up form, a laptop displaying a "Worksheet" interface with a list of data sources, and a tablet showing a "Welcome to Databend" screen.

Thanks

Rust China Conf 2021-2022
Online, China