ClickHouse at DoorDash

PATRICK ROGERS, SOFTWARE ENGINEER, OBSERVABILITY

December 12, 2024

Introduction

CONTENTS

- About Me & My Team
- Some ClickHouse Use Cases at DoorDash
  - TCP Connection & DNS Query Data
  - Service Map Connections
- Some Lessons Learned
- What's Next?

About Me

- Patrick Rogers 👋
  - hellopatrick on Github
  - in/patrick-rogers-here on LinkedIn
- Staff Engineer on the Observability team at DoorDash
- Before DoorDash, I worked on GIFs.


DOORDASH
Wolt

About Observability @ DoorDash

- Part of our Core Infrastructure organization.
- Help provide insights into our software systems.
- Develop & manage the metrics, logs, and tracing infrastructure.

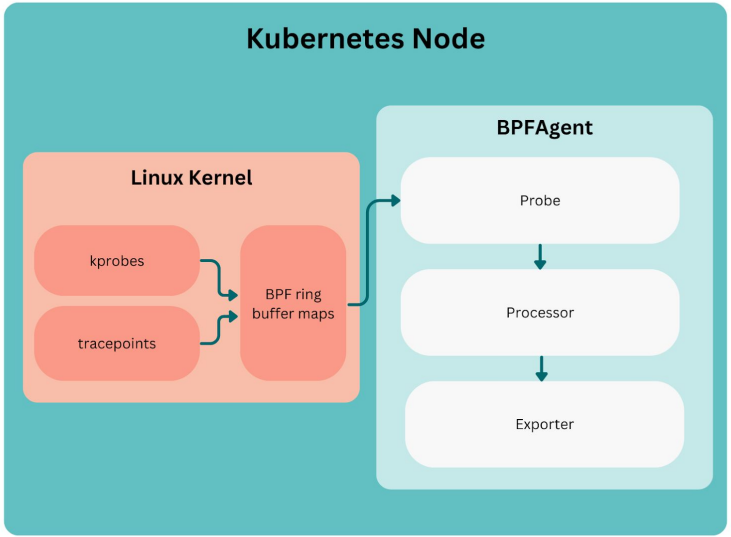
TCP & DNS Monitoring

NETWORK TOPOLOGY

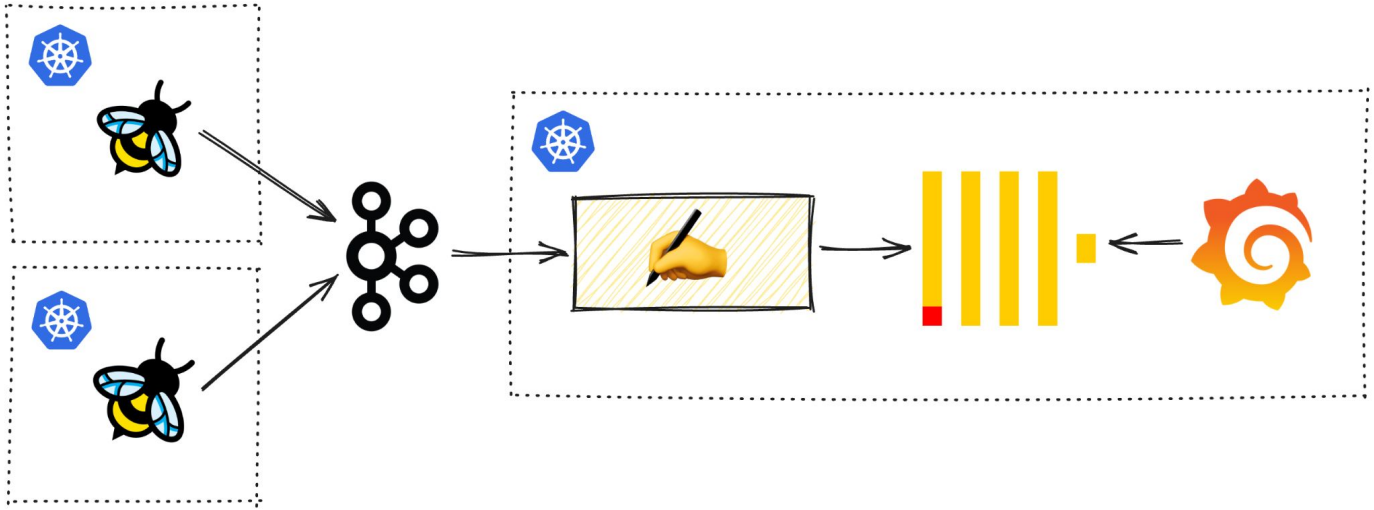
🔎 eBPF-based Monitoring

TCP & DNS

- Wrote BPFAgent to monitor TCP connections & DNS traffic in our Kubernetes clusters.
- Uses eBPF probes to monitor **tcp_connect**, **tcp_close**, and **skb_consume_udp**.
- Initially attempted to extract insights via Prometheus 😵 & ElasticSearch 💸.



High Level Architecture



📜 Chronicle

Our internal codename & repo for Clickhouse-related software.

scribe

- Kafka to Clickhouse Ingestion Pipelines
  - Built upon ch-go & franz-go.
- Wraps developer-written mappers of Kafka records to columns of data.

```
type Mapper interface {     Patrick Rogers, 2 years ago
  Table() string
  Map(records []*kgo.Record) (chproto.Input, error)
}
```

mappergen

- DoorDash uses protobufs heavily internally.
- Built `mappergen` to scaffold most mappers.
  - `protoreflect` types mapped to sensible choices of columns.
    - e.g. Enum is LowCardinality(String)
  - Will also generate a basic schema.

Schema

```
CREATE TABLE IF NOT EXISTS bpfagent.tcp_local ON CLUSTER primary
(
  `start_time` DateTime64(3) CODEC(DoubleDelta, LZ4),
  `end_time` DateTime64(3) CODEC(DoubleDelta, LZ4),
  `src_addr` IPv6,
  `dst_addr` IPv6,
  `dst_port` UInt16,
  `dst_dns` String,
  `exepath` String,
  `bytes_sent` UInt64,
  `bytes_recv` UInt64,
  `namespace` LowCardinality(String),
  `service` LowCardinality(String),
  `app` LowCardinality(String),
  `pod` String,
  `container` LowCardinality(String),
  `k8s_cluster_id` LowCardinality(String),
  `host` String CODEC(ZSTD(1))
)
ENGINE = MergeTree
ORDER BY (`k8s_cluster_id`, `namespace`, `pod`, `dst_port`, `dst_dns`, `end_time`)
```

TCP Mapper

```
type Mapper struct {
  age          prometheus.Histogram
  input        chproto.Input
  startCol     *chproto.ColDateTime64
  endCol       *chproto.ColDateTime64
  srcAddrCol   *chproto.ColIPv6
  dstAddrCol   *chproto.ColIPv6
  dstPortCol   *chproto.ColUInt16
  dstDNSCol    *chproto.ColStr
  exeCol       *chproto.ColStr
  sentCol      *chproto.ColUInt64
  recvCol      *chproto.ColUInt64
  namespaceCol *chproto.ColLowCardinality[string]
  serviceCol   *chproto.ColLowCardinality[string]
  appCol       *chproto.ColLowCardinality[string]
  podCol       *chproto.ColStr
  containerCol *chproto.ColLowCardinality[string]
  clusterCol   *chproto.ColLowCardinality[string]
  hostCol      *chproto.ColStr
}
```

```
func (m *Mapper) Map(records []*kgo.Record) (chproto.Input, error) {
  var pb bpfagent.TcpConnectionEvent

  m.input.Reset()

  for _, r := range records {
    if err := proto.Unmarshal(r.Value, &pb); err ≠ nil {
      return nil, err
    }

    start := pb.StartedAt.AsTime()
    end := pb.EndedAt.AsTime()
    m.age.Observe(time.Since(end).Seconds())

    m.startCol.Append(start)
    m.endCol.Append(end)

    var srcArr, dstArr [16]byte

    if src := net.ParseIP(pb.SourceAddr); src ≠ nil {
      copy(srcArr[:], src[:16])
    }

    m.srcAddrCol.Append(srcArr)
```

Downstream Dependencies

destination	total
sibyl-prediction-service-web-cng.service.prod.ddsd:50051	9.06 Mil
promotion-service-web.service.prod.ddsd:50051	4.13 Mil
sibyl-prediction-service-web-feed-hp.service.prod.ddsd:50051	3.50 Mil
order-service-cart-grpc.service.prod.ddsd:50051	2.43 Mil
dbmesh-discovery-experience.service.prod.ddsd:50051	2.05 Mil
menu-data-service-consumer.service.prod.ddsd:50051	2.03 Mil
geo-intelligence-web.service.prod.ddsd:50051	1.99 Mil
geo-intelligence-web-eta.service.prod.ddsd:50051	1.95 Mil
sibyl-prediction-service-web-search.service.prod.ddsd:50051	1.45 Mil
consumer-profile-service-web.service.prod.ddsd:50051	1.38 Mil

Connections Tracked For sibyl-prediction-service-web-cng.service.prod.ddsd

column	type	data_compressed_bytes	data_uncompressed_bytes	compression_ratio	codec
pod	String	28.6 GB	6.46 TB	225	
exepath	String	49.1 GB	4.39 TB	89.3	
dst_dns	String	26.2 GB	4.26 TB	163	
host	String	16.4 GB	3.14 TB	192	
dst_addr	IPv6	323 GB	2.37 TB	7.35	
src_addr	IPv6	11.8 GB	2.37 TB	201	
bytes_sent	UInt64	455 GB	1.19 TB	2.61	
start_time	DateTime64(3)	327 GB	1.19 TB	3.62	CODEC(DoubleDelta, LZ4)
bytes_recv	UInt64	298 GB	1.19 TB	3.98	
end_time	DateTime64(3)	255 GB	1.19 TB	4.65	CODEC(DoubleDelta, LZ4)
dst_port	UInt16	2.66 GB			
container	LowCardinality(String)	16.2 GB			
app	LowCardinality(String)	1.48 GB			
service	LowCardinality(String)	1.46 GB			
namespace	LowCardinality(String)	1.36 GB			
k8s_cluster_id	LowCardinality(String)	697 MB			

```
SELECT
    name as column,
    type,
    sum(data_compressed_bytes) as data_compressed_bytes,
    sum(data_uncompressed_bytes) as data_uncompressed_bytes,
    data_uncompressed_bytes / data_compressed_bytes AS compression_ratio,
    compression_codec as codec
FROM cluster(primary, system, columns)
WHERE database = '${db}' and table = '${table}'
GROUP BY column, type, codec
ORDER BY compression_ratio DESC
```

Service Map

SERVICE TOPOLOGY

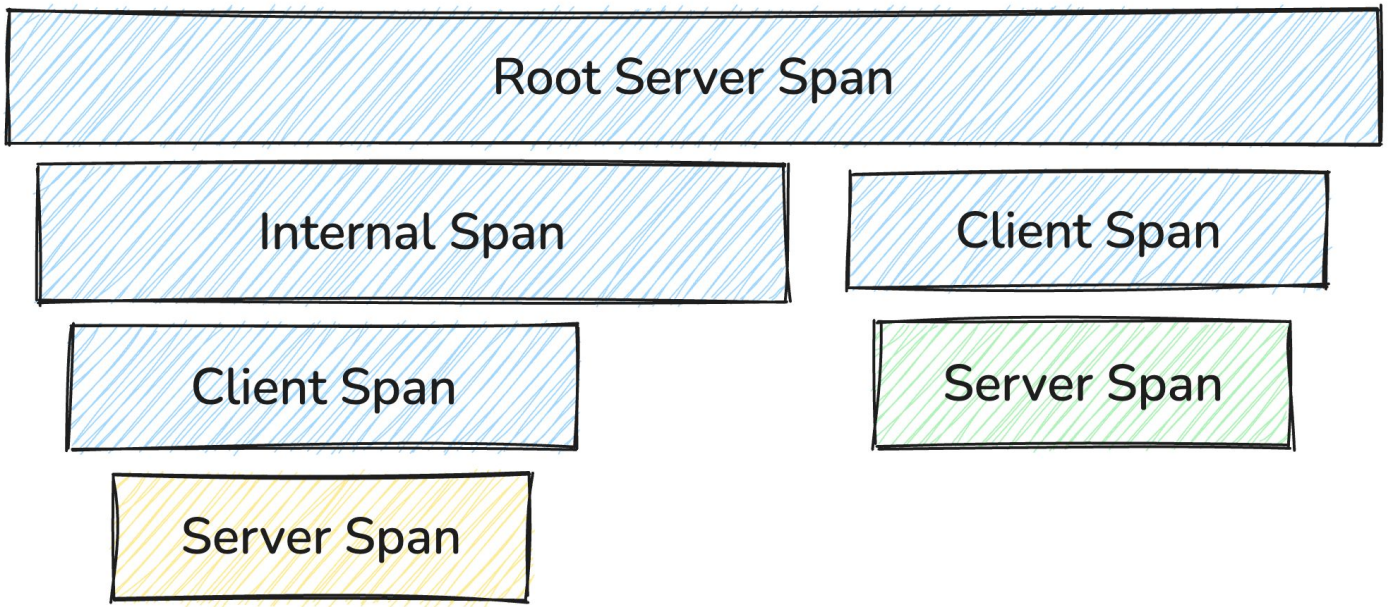
Endpoint-scoped Dependencies

- eBPF data resolves at the service/container-level.
- How can we help teams know their dependencies at the endpoint level?

🔭 OpenTelemetry

- Fully adopted at DoorDash.
- If you store the span data in ClickHouse, it'll likely look something like this.
- But this is difficult to analyze connections without expensive recursive CTEs.

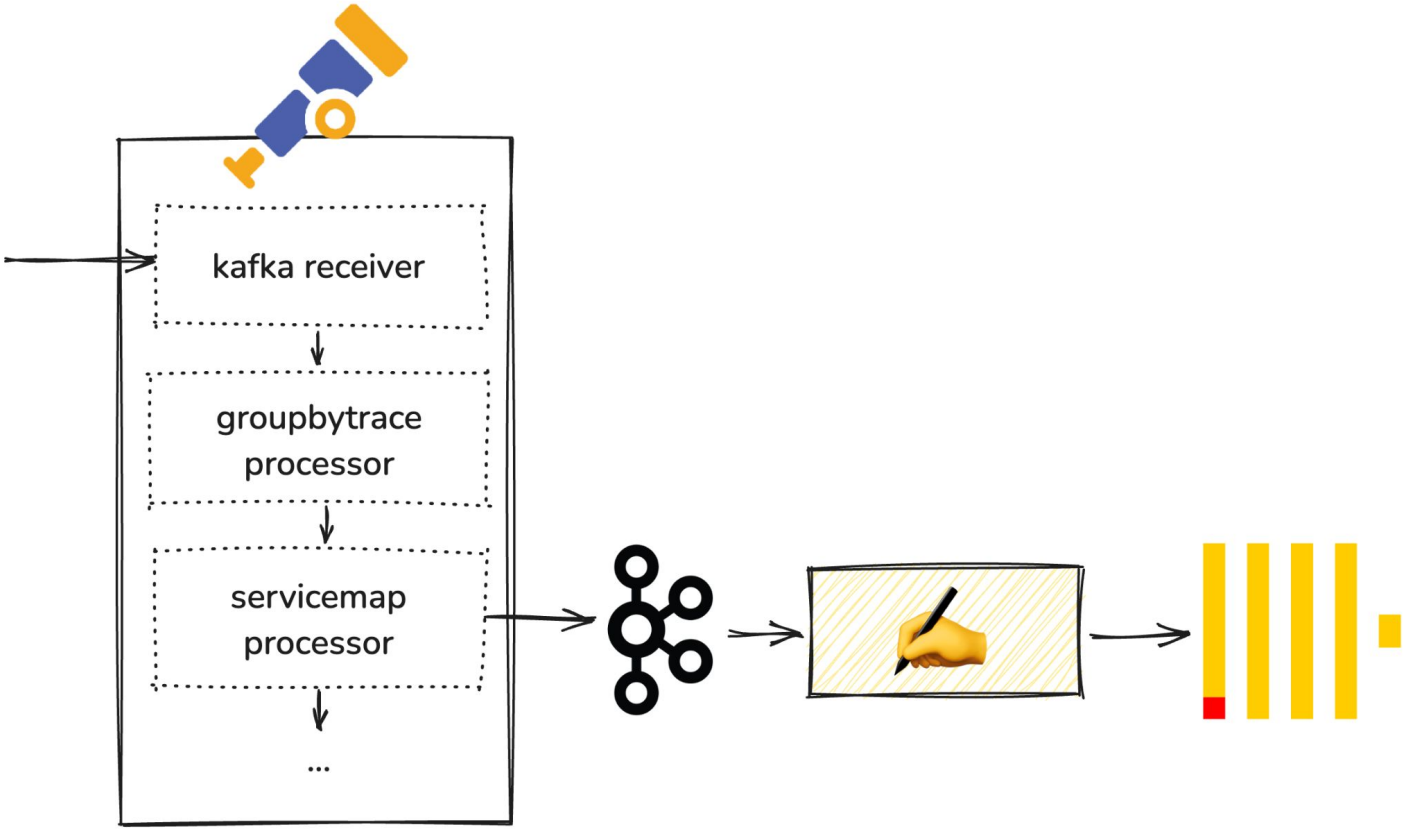
```
CREATE TABLE IF NOT EXISTS otel.traces_local ON CLUSTER primary
(
    `timestamp`      DateTime64(9) CODEC (DoubleDelta, ZSTD(1)),
    `trace_id`       FixedString(16) CODEC (ZSTD(1)),
    `span_id`        FixedString(8) CODEC (ZSTD(1)),
    `parent_span_id` FixedString(8) CODEC (ZSTD(1)),
    `trace_state`    String CODEC (ZSTD(1)),
    `span_name`      LowCardinality(String) CODEC (ZSTD(1)),
    `span_kind`      LowCardinality(String) CODEC (ZSTD(1)),
    `service_name`   LowCardinality(String) CODEC (ZSTD(1)),

    `resource_attributes` Map(LowCardinality(String), String) CODEC (ZSTD(1)),
    `span_attributes`     Map(LowCardinality(String), String) CODEC (ZSTD(1)),

    `duration`       Int64 CODEC (T64, ZSTD(1)),
    `status_code`    LowCardinality(String) CODEC (ZSTD(1)),
    `status_message` String CODEC (ZSTD(1)),

    `events` Nested (
        `timestamp`        DateTime64(9),
        `name`             LowCardinality(String),
        `attributes`       Map(LowCardinality(String), String)
    ) CODEC(ZSTD(1)),

    INDEX idx_trace_id trace_id TYPE bloom_filter(0.001) GRANULARITY 1,
    INDEX idx_res_attr_keys mapKeys(resource_attributes) TYPE bloom_filter(0.01) GRANULARITY 1,
    INDEX idx_res_attr_values mapValues(resource_attributes) TYPE bloom_filter(0.01) GRANULARITY 1,
    INDEX idx_span_attr_keys mapKeys(span_attributes) TYPE bloom_filter(0.01) GRANULARITY 1,
    INDEX idx_span_attr_values mapValues(span_attributes) TYPE bloom_filter(0.01) GRANULARITY 1,
    INDEX idx_duration duration TYPE minmax GRANULARITY 1
)
ENGINE = MergeTree
ORDER BY (toStartOfHour(`timestamp`), `service_name`, `span_kind`, `span_name`, `status_code`, `timestamp`)
```


Root Server Span
Internal Span
Client Span
Client Span
Server Span
Server Span

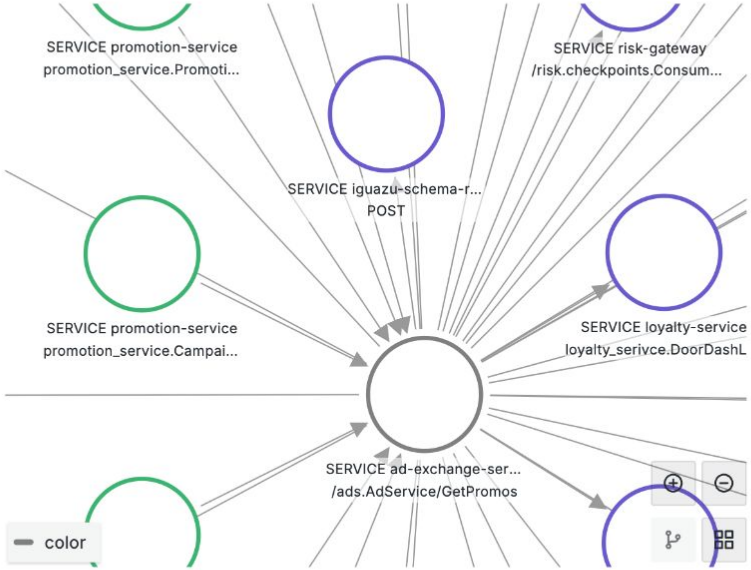
Service Map

- We want to store sources & targets.
- Still include some information about these edges, such as trace and span id.

```
CREATE TABLE IF NOT EXISTS servicemap.raw_edges_local ON CLUSTER primary
(
    `timestamp`         DateTime CODEC (DoubleDelta, ZSTD(1)),

    `source_type`       LowCardinality(String) CODEC (ZSTD(1)),
    `source_name`       String CODEC (ZSTD(1)),
    `source_context`    String CODEC (ZSTD(1)),
    `source_tier`       Int8 CODEC (T64, LZ4),
    `source_error`      Bool,
    `source_duration`   UInt64 CODEC (T64, LZ4),
    `source_span_id`    FixedString(8) CODEC (LZ4),

    `target_type`       LowCardinality(String) CODEC (ZSTD(1)),
    `target_name`       String CODEC (ZSTD(1)),
    `target_context`    String CODEC (ZSTD(1)),
    `target_tier`       Int8 CODEC (T64, LZ4),
    `target_error`      Bool,
    `target_duration`   UInt64 CODEC (T64, LZ4),
    `target_span_id`    FixedString(8) CODEC (LZ4),

    `trace_id`          FixedString(16) CODEC (LZ4)
)
ENGINE = MergeTree
ORDER BY (
  `source_type`, `source_name`, `source_context`,
  `target_type`, `target_name`, `target_context`,
  `timestamp`
)
```


kafka receiver
groupbytrace processor
servicemap processor
...

Service Graph



Clients

source_name	source_context
promotion-service	promotion_service.PromotionService/(
promotion-service	promotion_service.PromotionService/(
promotion-service	promotion_service.PromotionService/(
promotion-service	promotion_service.CampaignServingS

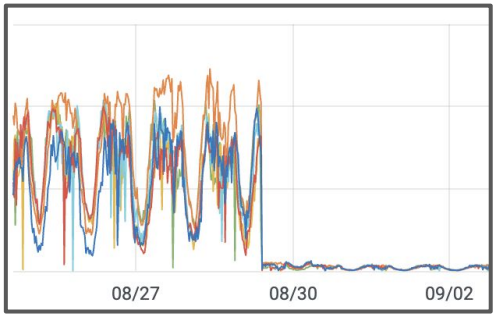
Dependencies

target_type	target_name	target_context	appears	fanout
SERVICE	ad-billing-service	/ad_billing_service.v1	100%	2
SERVICE	argo-search-ads-sup	/argo_search.broker.v	100%	2
SERVICE	janus-service	/janus_service.v1.Not	100%	1
SERVICE	audience-service	/doordash.audience.s	100%	1
SERVICE	consumer-service	doordash_consumer_	100%	1
SERVICE	loyalty-service	/loyalty_serivce.Door	100%	1
SERVICE	audience-service	/doordash.audience.s	100%	1
SERVICE	content-management	GET /v1/contents/:id	20%	3
SERVICE	entity-cache-web	POST /risk.checkpoir	20%	1
SERVICE	merchant-data-servic	/composite_store_ser	20%	1
SERVICE	iguazu-kafka-rest-pr	POST /topics/{topic}	10%	5
SERVICE	iguazu-schema-regis	POST	10%	2
SERVICE	content-management	GET /v1/contents/{id}	10%	2
SERVICE	sibyl-prediction-serv	SibylPredictionServic	10%	1
MESSAGE_BUS	kafka	promotion-streaming	10%	1
SERVICE	merchant-data-servic	composite_store_serv	10%	1

📚 Some Lessons Learned

Experiences

- Avoid distributed table inserts.
  - We're already writing our own ingestors, so we turned to relying purely on client-side load balancing.
  - Reduced number of merges/inserts that any node has to deal with by an order-of-magnitude.
- Monitor disk throughput.
  - OpenTelemetry cluster would come to halt at some times.
    - During S3 uploads, we would hit our EBS volume throughput limits.
  - Split load over multiple disks with fastest one used for ingestion.
- To use **ch-go** with Nested, model as Array columns.




```
eventsTimestamp  *chproto.ColArr[time.Time]
eventsName       *chproto.ColArr[string]
eventsAttributes *chproto.ColArr[map[string]string]
```

🔮 What's Next For Us?

Patiently Awaiting non-experimental JSON!

- Adopting structured wide events as the Observability pillar.
  - Context-rich events per network hop in internal processes.
    - Modelled as OpenTelemetry spans.
    - "*That log line should have been an attribute*".
  - Semantic conventions shared across the organization.
- Stored in ClickHouse.
  - The new experimental JSON type is *very* exciting.

One Last Thing...

Advent of Code*!*

adventofcode.com

- Daily programming puzzles throughout the month of December.
  - Typically people do them in Python… or OCaml… or C++.
- But what about ClickHouse?
  - ClickHouse SQL is quite featureful & expressive.
  - `clickhouse local` as our “runtime”.

🫣 Day 2 Spoilers

adventofcode.com/2024/day/2

- **Parsing** Input is a text file, one sequence of integers per line.
    - splitByChar, arrayMap, toInt*
- **Part 1** How many sequences are monotonically increasing or decreasing with a bounded difference between elements?
    - arrayDifference, arrayExists, arrayAll
- **Part 2** What if you are allowed to delete one element from the sequence, how many can be made valid?
    - arrayEnumerate, arrayFilter, arrayMap

```
-- clickhouse local --queries-file bonus/2024-02.sql
with (
    --- input
    arrayMap(x -> toInt16(x), splitByChar(' ', c1)) as
report, part1
    arrayPopFront(arrayDifference(report)) as deltas,
    not arrayExists(x -> abs(x) > 3, deltas)
    and (
        arrayAll(x -> x > 0, deltas)
        or arrayAll(x -> x < 0, deltas)
    ) as valid,
    --- part2
    arrayEnumerate(report) as idxs,
    arrayMap(
        i -> arrayFilter((_, j) -> i != j, report, idxs),
        idxs
    ) as possibleReports,
    arrayMap(
        report -> arrayPopFront(arrayDifference(report)),
        possibleReports
    ) as possibleDeltas,
    arrayExists(
        deltas -> (
            not arrayExists(x -> abs(x) > 3, deltas)
            and (
                arrayAll(x -> x > 0, deltas)
                or arrayAll(x -> x < 0, deltas)
            )
        ),
        possibleDeltas
    ) as fixable
)
select
    countIf(valid) as part1,
    countIf(valid or fixable) as part2
from
    file('testdata/2024-02.txt', 'LineAsString', 'c1
String');
```


Thank You!
Questions?