CENTENE® Corporation

# Building a Enterprise Eventing Platform

Bryan Zelle

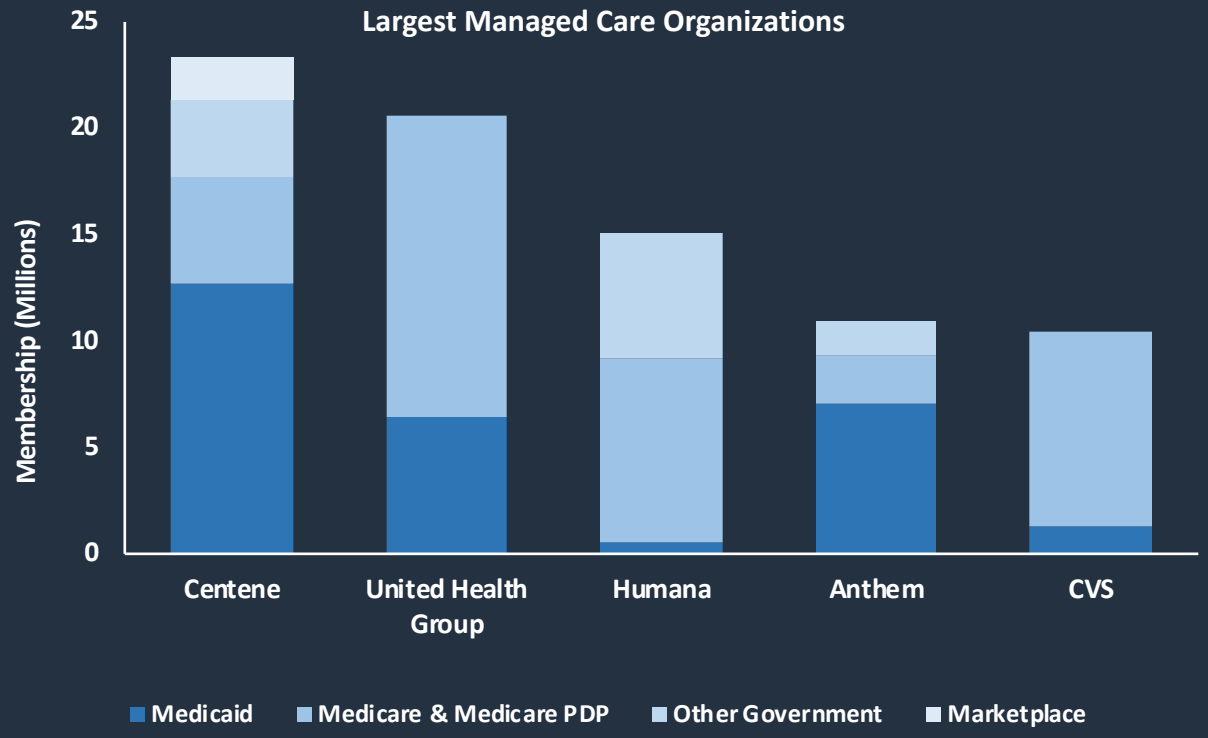
# Centene Introduction

***Industry:***

*Largest Medicaid and Medicare Managed Care Provider*

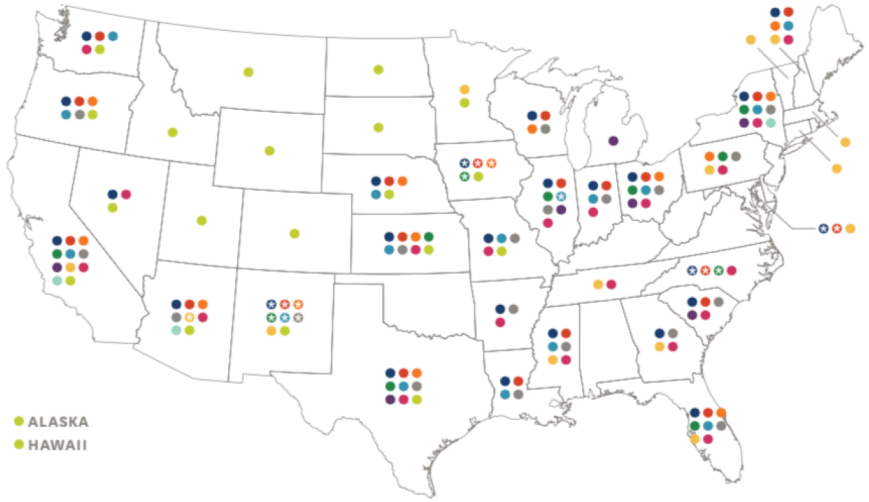
***Mission Statement:***

*Transforming the health of the community, one person at a time*



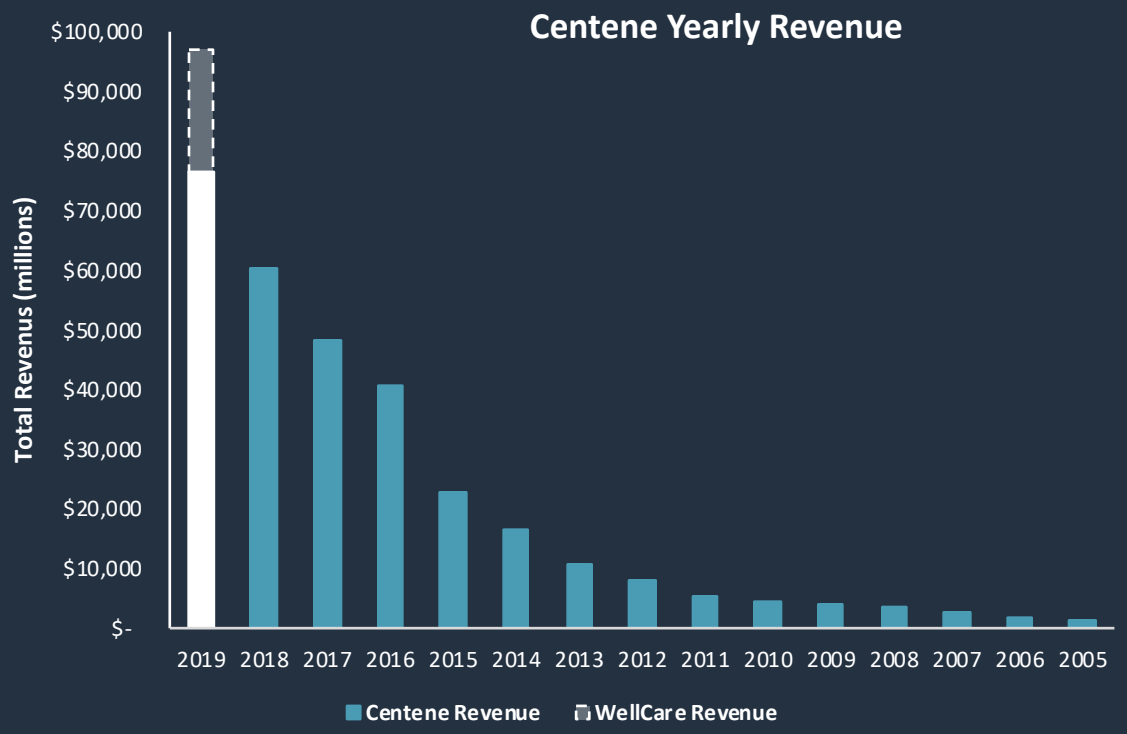

## Membership Composition:

| | | |
|---|---|---|
| Medicaid: | 12,700,000 | 30 States |
| Medicare (Part D): | 4,000,000 | 50 States |
| Marketplace: | 2,000,000 | 21 States |
| Medicare: | 1,000,000 | 28 States |
| Other: | 3,700,000 | 33 States |
| **Total:** | **23,400,000** | **50 States** |



# Summary of Centene's Key Challenges in one word…

# Growth




## By the numbers:

$4.1 Billion Revenue to $96.9 Billion in 10 Years

$80.4 Billion in growth in past 5 years

$48.6 Billion in growth in past 2½ years

## Cause of the growth…

# Mergers & Acquisitions

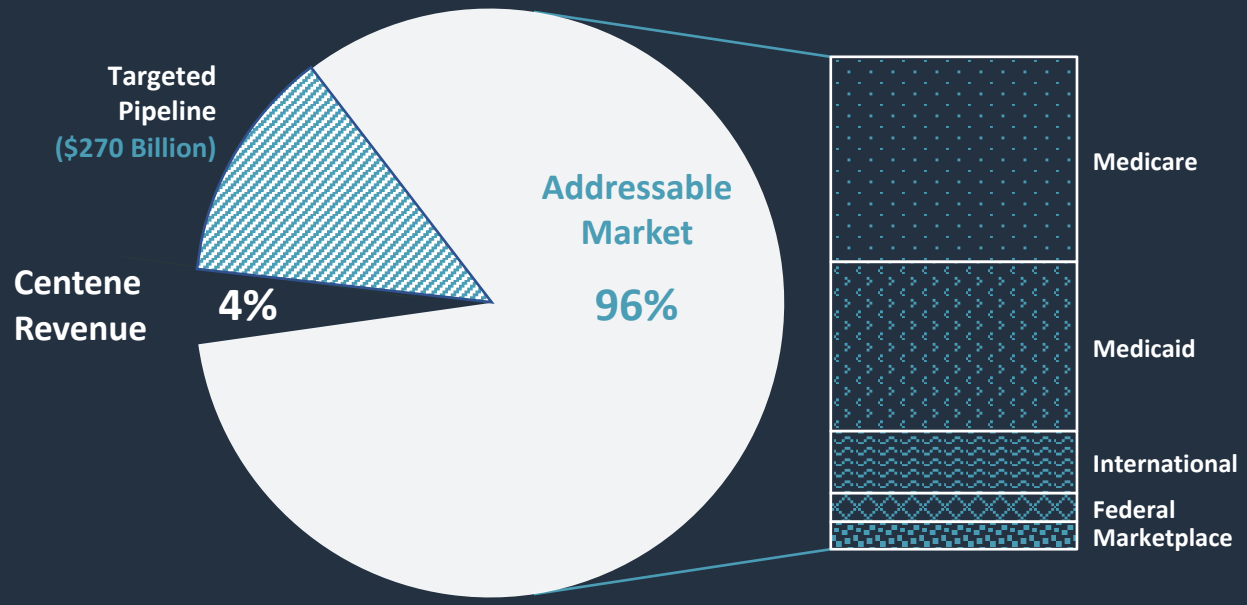
# Centene Growth Outlook

**Addressable Market**

**$2,000,000,000,000 +**

**Centene Revenue**

**$97,000,000,000 +**




# Additional Growth Opportunities

- **$860 B** 40% — **Federal Medicare**
- **$710 B** 33% — **State Medicaid**
- **$260 B** 12% — **International Market**
- **$120B** 6% — **Federal Services**
- **$115 B** 5% — **Health Insurance Marketplace**

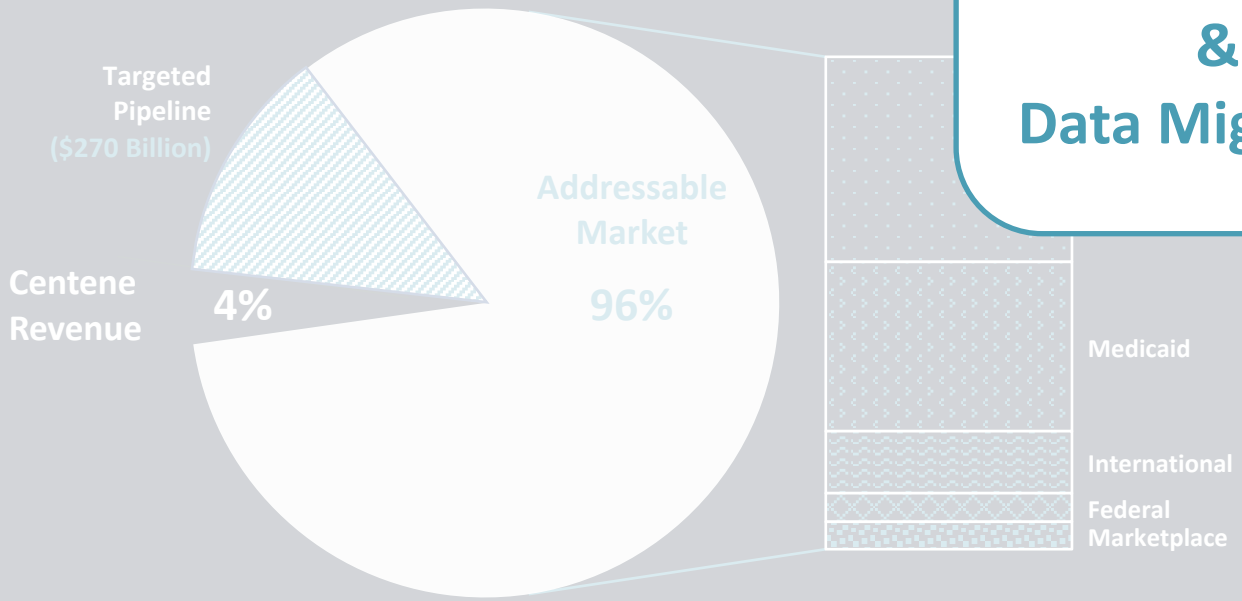
# Centene Growth Outlook

**Addressable Market**

**$2,000,000,000,000 +**

**Centene Revenue**

**$97,000,000,000 +**




## Mergers & Acquisitions

**Data Integration & Data Migration**

# Additional Growth Opportunities

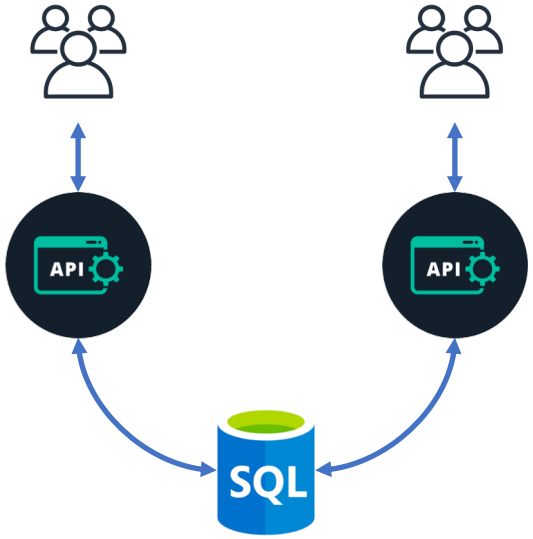
- Federal Medicare
- State Medicaid
- International Market
- **$120B** 6% — Federal Services
- **$115 B** 5% — Health Insurance Marketplace

# Data Integration & Data Migration

1

## Shared Database

- Application Refactor
- Direct Schema Coupling
- Scaling Challenges
- Single Point of Failure



# Data Integration & Data Migration

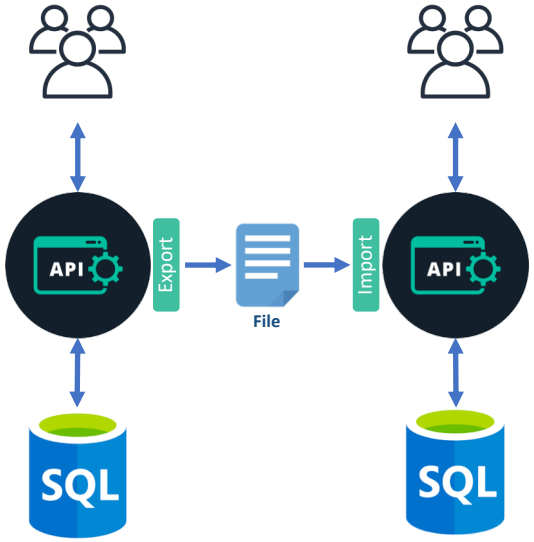
## Shared Database

- Application Refactor
- Direct Schema Coupling
- Scaling Challenges
- Single Point of Failure




## 2 File Transfer (Batch ETL)

- Latent Data
- Direct Database Load
- Consistency Challenges



# Data Integration & Data Migration

## Shared Database

- Application Refactor
- Direct Schema Coupling
- Scaling Challenges
- Single Point of Failure



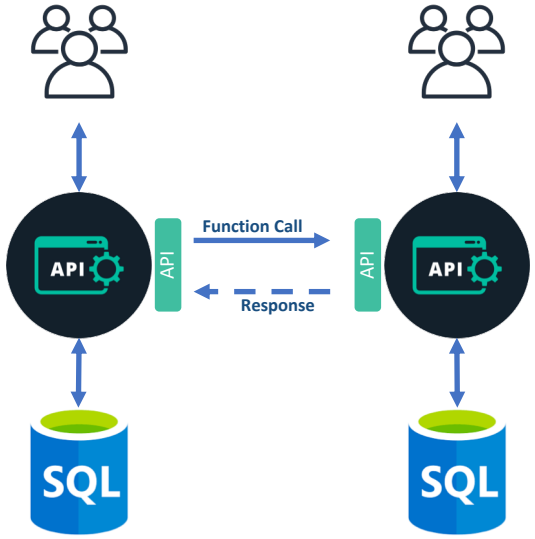

## File Transfer (Batch ETL)

- Latent Data
- Direct Database Load
- Consistency Challenges




## 3 Remote Procedure Invocation

- Direct Coupling
- Application Refactor
- Availability Concerns
- Scaling Concerns



# Data Integration & Data Migration

## Shared Database

- Application Refactor
- Direct Schema Coupling
- Scaling Challenges
- Single Point of Failure




## File Transfer (Batch ETL)

- Latent Data
- Direct Database Load
- Consistency Challenges



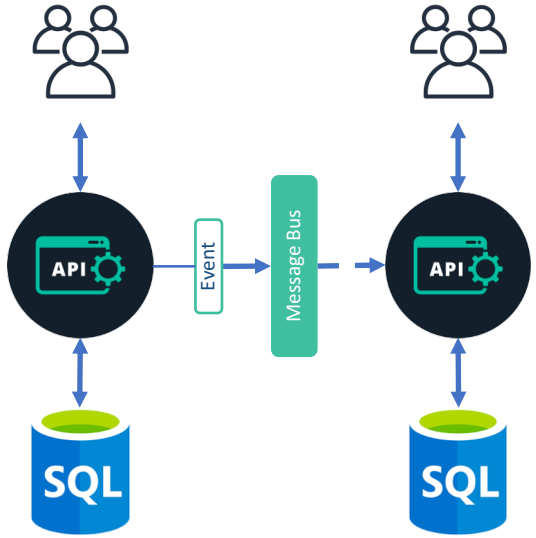

## Remote Procedure Invocation

- Direct Coupling
- Application Refactor
- Availability Concerns
- Scaling Concerns




## 4 Pub / Sub Messaging (Streaming ETL)

- Loosely Coupled
- No Application Refactor
- Highly Availability
- Highly Scalable
- Real-Time Data



# Data Integration & Data Migration

## Shared Database

- Application Refactor
- Direct Schema Coupling
- Scaling Challenges
- Single Point of Failure




## What is a Event?

**Definition: "A significant change in state"**

- Statement of fact (immutable)
- Expects no response (or call to action)
- Has a defined "timepoint"

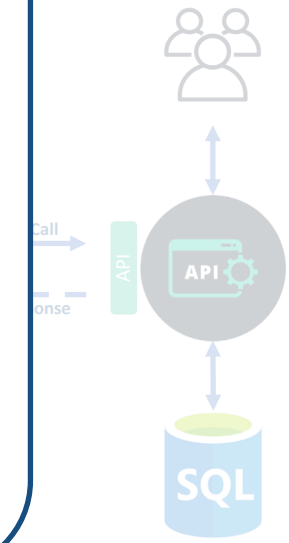
**Persistence**

- Stateless: Notification Event
- Stateful: Event-Carried State Transfer

**Synthesized / Composite Events**

- Combination of Events

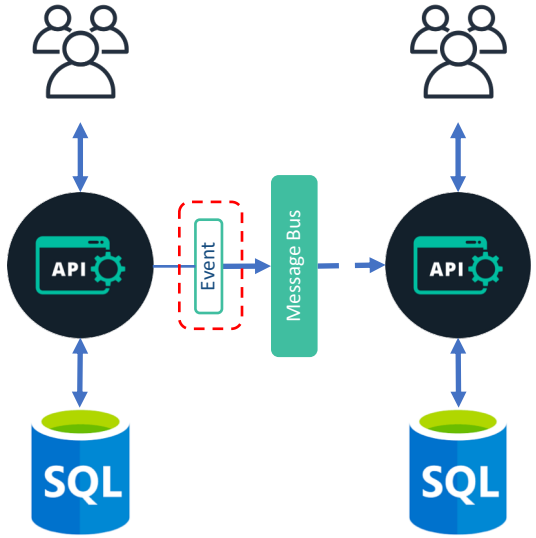

- Absence of an Event


## Pub / Sub Messaging

**(Streaming ETL)**

- Loosely Coupled
- No Application Refactor*
- Highly Availability
- Highly Scalable
- Real-Time Data



# Data Integration & Data Migration

## What is a Event?

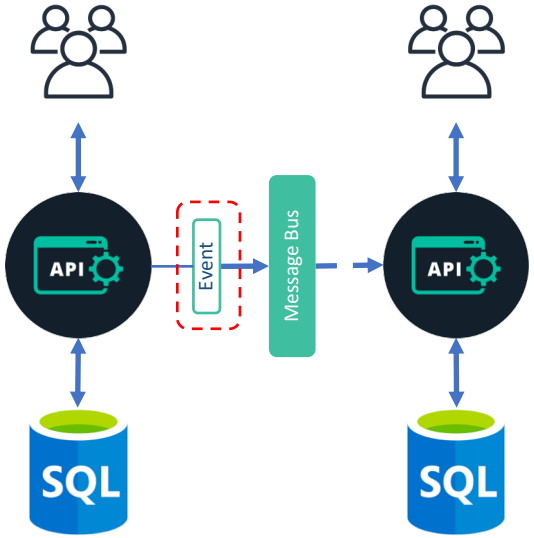
**Definition: "A significant change in state"**

- Statement of fact (immutable)
- Expects no response (no call to action)
- Has a defined "timepoint"

**How do we publish / consume meaningful events?**

### Pub / Sub Messaging
**(Streaming ETL)**

- Loosely Coupled
- No Application Refactor*
- Highly Availability
- Highly Scalable
- Real-Time Data



# Change Data Capture (CDC)

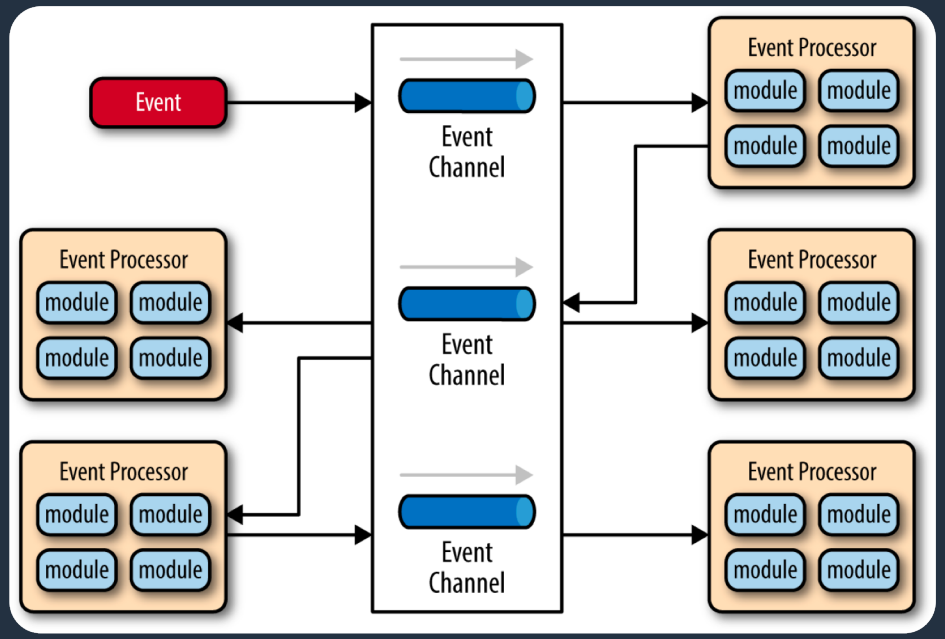
## Broker Topology

**Subscribe to Event Channels (Topics)**

**Self-Defined Event Routing**

**Partial Coupling of Event Channels**

**Reduced Complexity at cost of reduced coordinating of event execution**




### Advantages:

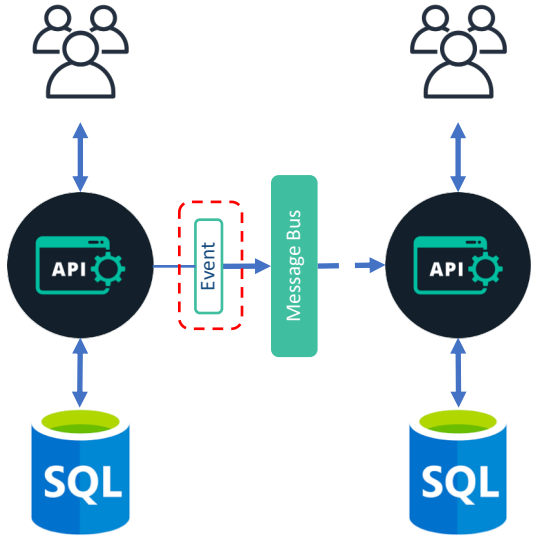
- Mature 3rd Party Products / Tooling
- Limited Database Load
- Fast Implementation
- No refactoring of source system

### Disadvantages:

- No consistent data structure
- No data governance
- Direct technology coupling

## Pub / Sub Messaging
**(Streaming ETL)**

- Loosely Coupled
- No Application Refactor*
- Highly Availability
- Highly Scalable
- Real-Time Data



# Change Data Capture (CDC)

**Transaction Metadata**

**Event Body**

## Example Event Payload (Informatica CDC Publisher v1.2)

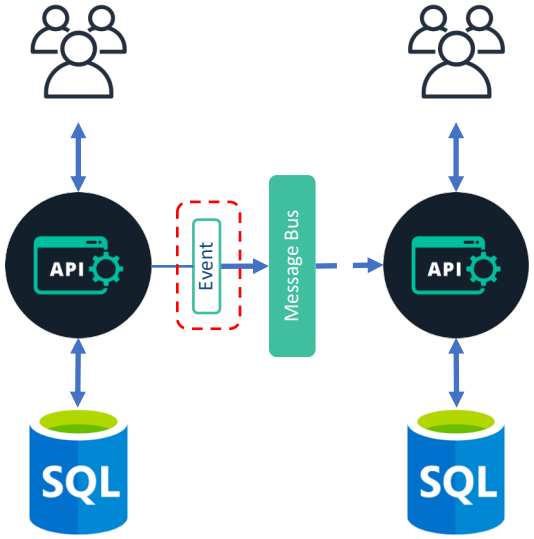

```
{
"INFA_SEQUENCE":{"string":"2,PWX_GENERIC,1,,2,3,C7084816514A5D260"}
,\"DTL__CAPXUSER\":{\"string\":\"USER1\"}
,\"DTL__CAPXTIMESTAMP\":{\"string\":\"20180305131540000000000\"}
,\"INFA_OP_TYPE\":{\"string\":\"UPDATE_EVENT\"}
,\"INFA_TABLE_NAME\":{\"string\":\"d8amisou6p.MEMBER_CONTACT\"}

,\"MEMBER_PCP\":{\"string\":\"Dr. Bryan Zelle\"}
,\"MEMBER_PCP_Present\":true
,\"MEMBER_PCP_BeforeImage\":{\"string\":\"Dr. John Smith\"}
,\"MEMBER_PCP_BeforeImage_Present\":true
}
```


**What Event information are we capturing?**

- **Who** - Who changed the data ?*
- **What** - What data changed ?
- **When** - When the data changed ?
- **Where** - Where was the data changed ?

## Pub / Sub Messaging
**(Streaming ETL)**

- Loosely Coupled
- No Application Refactor*
- Highly Availability
- Highly Scalable
- Real-Time Data



# Mediated (Orchestrated) Eventing

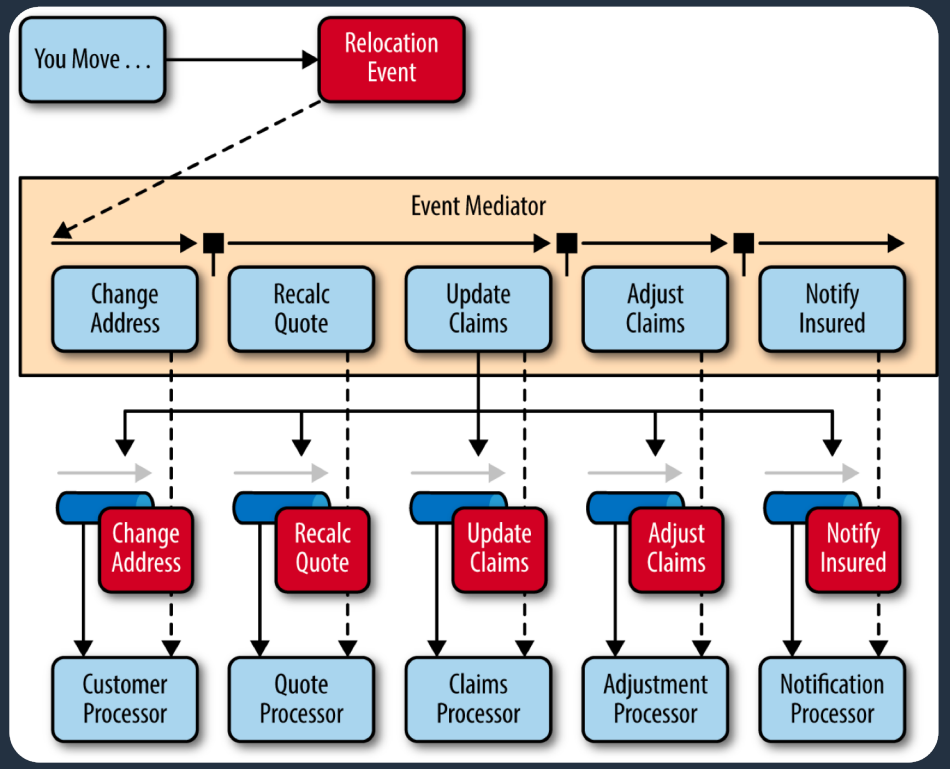
## Mediator Topology

**Mediator transfers events to assigned event channel (Topic)**

**Centrally Coordinated Event Routing**

**Complete Decoupling of Event Channels**

**Increased Complexity at cost of increased coordination of event execution**




**Advantages:**

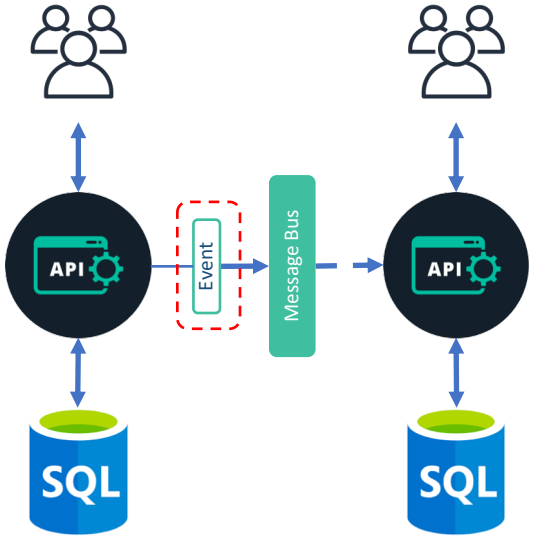
- Consistent / Common Framework
- Enforce Data governance
- Economy of Scale Advantage
- Technology abstraction / decoupling

**Disadvantages:**

- External bottleneck (Mediator Owner)
- Single Point of Failure
- Duplicative data storage

## Pub / Sub Messaging
**(Streaming ETL)**

- Loosely Coupled
- No Application Refactor*
- Highly Availability
- Highly Scalable
- Real-Time Data



# Mediated (Orchestrated) Eventing

## Example Event Payload (JSON vis REST)

**Transaction Metadata**

```
   "Metadata" : {
"Transaction ID" : "C7084816514A5D260",
"User ID" : "USER1",
"Time Stamp" : "20180305131540000000000",
"Transaction Type" : "UPDATE",
"Source System" : "d8amisou6p.MEMBER_CONTACT" } ,
```

**Event Body**

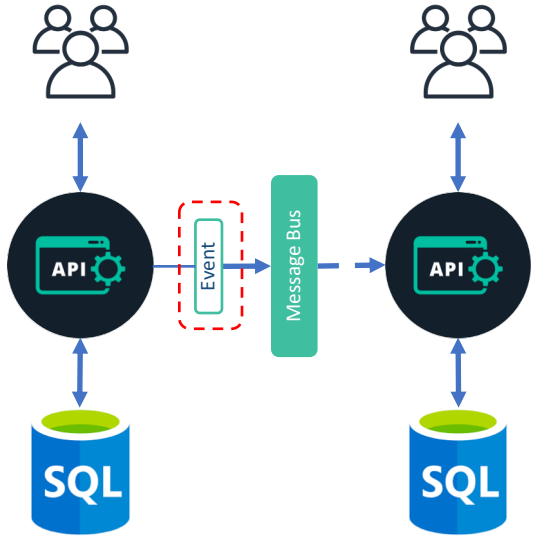
```
   "Event Body" : {
"Event Type" : "Member-PCP-Change",
"Previous Value" : "Dr. John Smith",
"Updated Value" : "Dr. Bryan Zelle",
"Event Source" : "Inbound-Member-Call",
"Caller Information" : {
"Name" : "Jane Doe",
"Inbound Number" : "1-614-847-0982",
"Call Resolution Status" : "5 - Highly Satisfied",
"First Call Resolution" : "Success",
"Internal Representative" : "CN-10238381",*
"Call Duration (Minutes)" : "8:19" }
```

**What Event information are we capturing?**

- **Who** - Who changed the data ?*
- **What** - What data changed ?
- **When** - When the data changed ?
- **Where** - Where was the data changed ?
- **Why** - Why was the data changed ?

## Pub / Sub Messaging
**(Streaming ETL)**

- Loosely Coupled
- No Application Refactor*
- Highly Availability
- Highly Scalable
- Real-Time Data



# Why is the "Why" so important?

"Event Body" : {
"Event Type" : "Member-PCP-Change",
"Previous Value" : "Dr. John Smith",
"Updated Value" : "Dr. Bryan Zelle",
"Event Source" : "Inbound-Member-Call",
"Caller Information" : {
"Name" : "Jane Doe",
"Inbound Number" : "1-614-847-0982",
"Internal Representative" : "CN-10238381",
"Call Resolution Status" : "5 - Highly Satisfied",
"First Call Resolution" : "Success",
"Call Duration (Minutes)" : "8:19"
}

**New Events can be Created or Derived**

**If two separate "Member-PCP-Change" events happen within 2 weeks window -> Create event for Case Manager to Review**

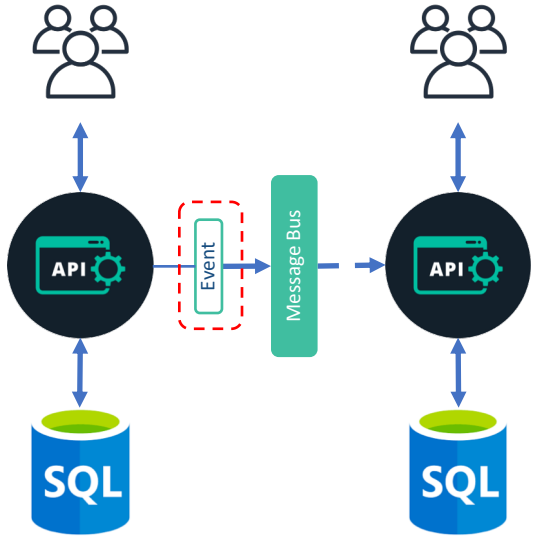
**If Inbound Number not currently associated with Member -> Create event to add it to the Member's Profile**

**If Resolution Status >= 4 -> Create event to assign call rep to member for call-back**

**If first call resolution < > "Success" and call duration > 15 min -> Create event to escalate call to Supervisor for Audit**

## Pub / Sub Messaging
**(Streaming ETL)**

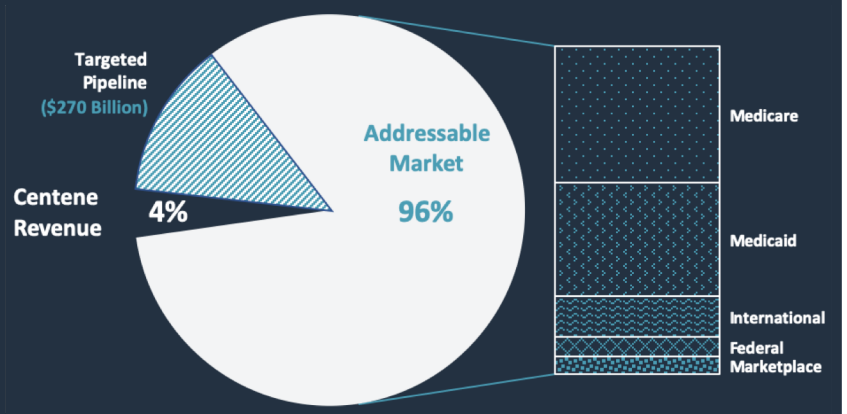
- Loosely Coupled
- No Application Refactor*
- Highly Availability
- Highly Scalable
- Real-Time Data



# Recap

1

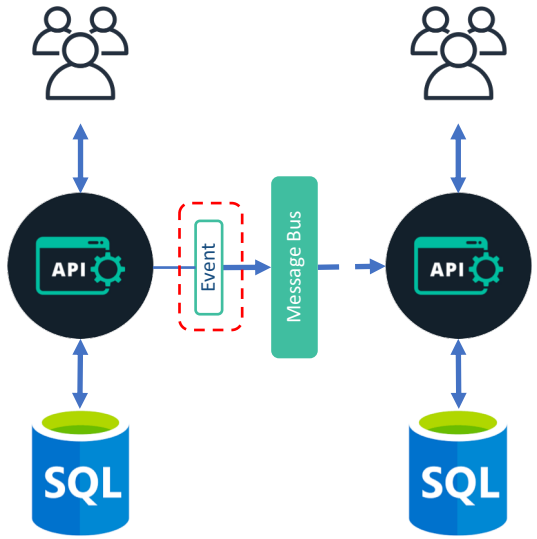
Centene's Core Challenge is Growth cause by Mergers & Acquisitions; causing us to revaluate our Enterprise Data Integration and Data Migration Strategies…




2

Async Pub / Sub Eventing through Kafka provides us valuable capabilities:

- Highly Scalable
- High Autonomy / Decoupling
- High Availability & Data Resiliency
- Real Time Data Transfer
- Complex Steam Processing




3

Leveraging a Mediator Topology enables the creation of meaningful events; which provide insight into why things are happening, so we can react to them in real time…


"Metadata" : {
"Transaction ID" : "C7084816514A5D260",
"User ID" : "USER1",
"Time Stamp" : "20180305131540000000000",
"Transaction Type" : "UPDATE",
"Source System" : "d8amisou6p.MEMBER_CONTACT" } ,
"Event Body" : {
"Event Type" : "Member-PCP-Change",
"Previous Value" : "Dr. John Smith",
"Updated Value" : "Dr. Bryan Zelle",
"Event Source" : "Inbound-Member-Call",
"Caller Information" : {
"Name" : "Jane Doe",
"Inbound Number" : "1-614-847-0982",
"Call Resolution Status" : "5 - Highly Satisfied",
"First Call Resolution" : "Success",
"Internal Representative" : "CN-10238381",
"Call Duration (Minutes)" : "8:19" }

# Common Core Architecture:

1) Event Source
2) Event Intake
3) Event Channel
4) Event Router
5) Event Subscription
6) Event Destination

# Mediator Alternatives?

Event Bridge

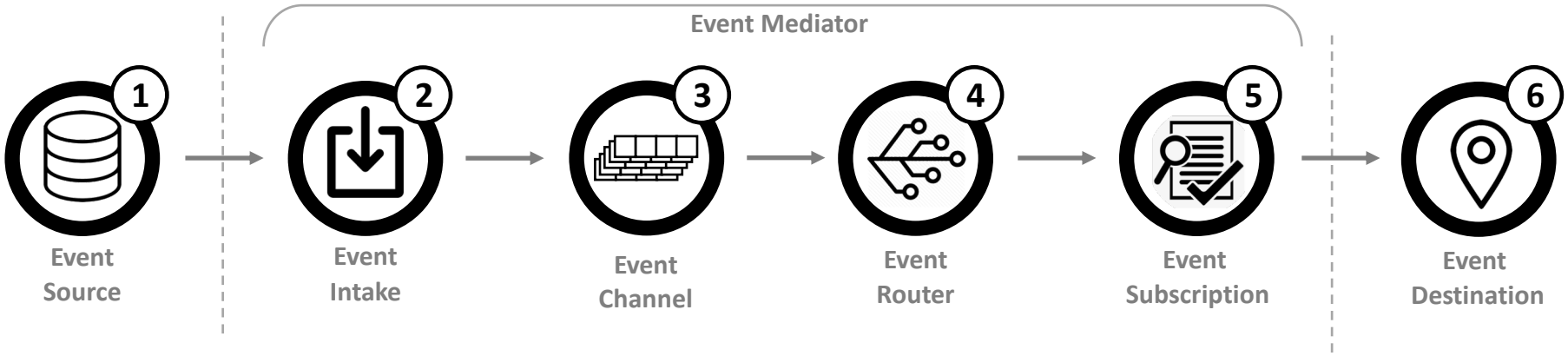

Mule ESB


Event Grid


Apache Camel


Knative Eventing


# Generic Event Mediator



# Required Features & Functionality

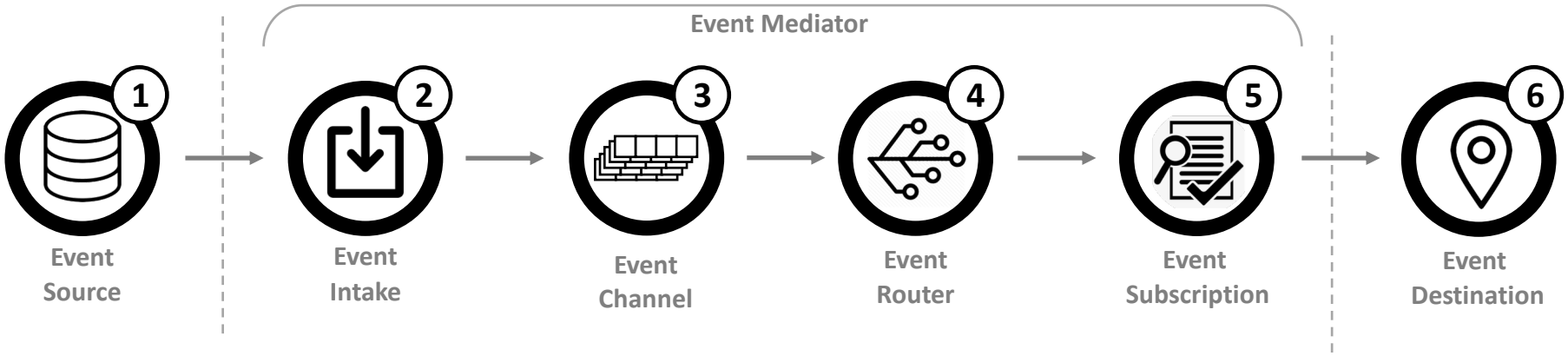
## Design Criteria

1) AVRO Event Serialization
2) JSON Validation of Event Body
3) Centralized Event Registry
4) Distributed Tracing of Events
5) Sensitive Data Redaction
6) Turn / Key Self-Service
7) Cloud Agnostic
8) Permanent Event Storage
9) Flexible Ingestion Intake
10) Pre-built Monitoring / Dashboards
11) Synthetic Events

## Business Value

Reduced Message Size -> Reduced Storage Cost in Cloud
Data Validation -> Clean Data
Easily Find Events -> Prevents Event Duplication & Increases Adoption
Tracing -> Provides Event Lineage and Auditability
Data Restriction -> Protects HIPPA data (including PHI/PII)
Automated Configuration -> Reduced manual administrative burden
Multi-Cloud Strategy -> No Reliance on Single Cloud Provider
Event Persistence -> DR Strategy + Event Sourcing / Hydration
Legacy Systems Limitations -> Offer REST, gRPC, SOAP Interfaces & API's
Universal Metrics -> Consistent / Granular Event Visibility
Fictitious Event -> Blue/Green Deployments, Prod Smoke Testing, Etc.

# Generic Event Mediator



# Required Features & Functionality

## Design Criteria

1) AVRO Event Serialization
2) JSON Validation of Event Body*
3) Centralized Event Registry
4) Distributed Tracing of Events*
5) Sensitive Data Redaction*
6) Turn / Key Self-Service
7) Cloud Agnostic*
8) Permanent Event Storage
9) Flexible Ingestion Intake*
10) Pre-built Monitoring / Dashboards
11) Synthetic Events*

## Leverage 3rd Party Frameworks or Build Custom?

**Assessment:**

Majority of frameworks focused engineering effort on how to get data into framework as easily as possible

- Higher Data Ingest = Increased Billables (SaaS)
- Too many gaps with current features*

**Decision:**

Build Centralized Eventing Framework for Enterprise use across all Centene Domains

n Cloud

Increases Adoption

PHI/PII)

ministrative burden

ud Provider

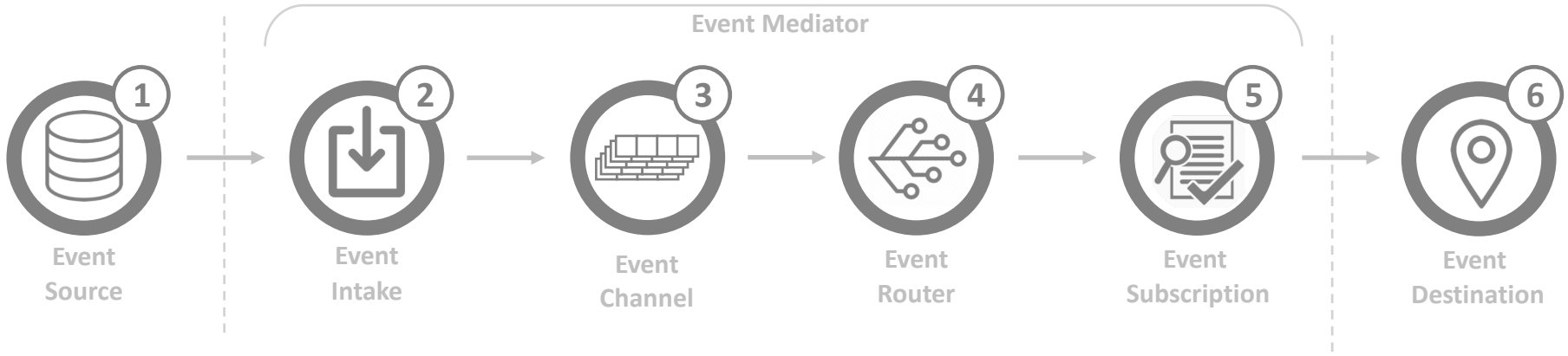
/ Hydration

OAP Interfaces & API's

Visibility

od Smoke Testing, Etc.

## Generic Event Mediator



# CentEvent Architecture

Intake Application

Axway Gateway

{REST}

GRPC

Kubernetes

Docker Container

Java

Firehose Topic

Router Application

Kubernetes

Docker Container

Java

Consumer Topics

mongoDB

Authorization Tokens

Event Types

Routing Rules

Routing HASH

Caffeine Cache

Confluent Schema Registry

Serialize

Deserializer

Schema Registry

UI

Event Discovery UI

Client UI

Admin UI

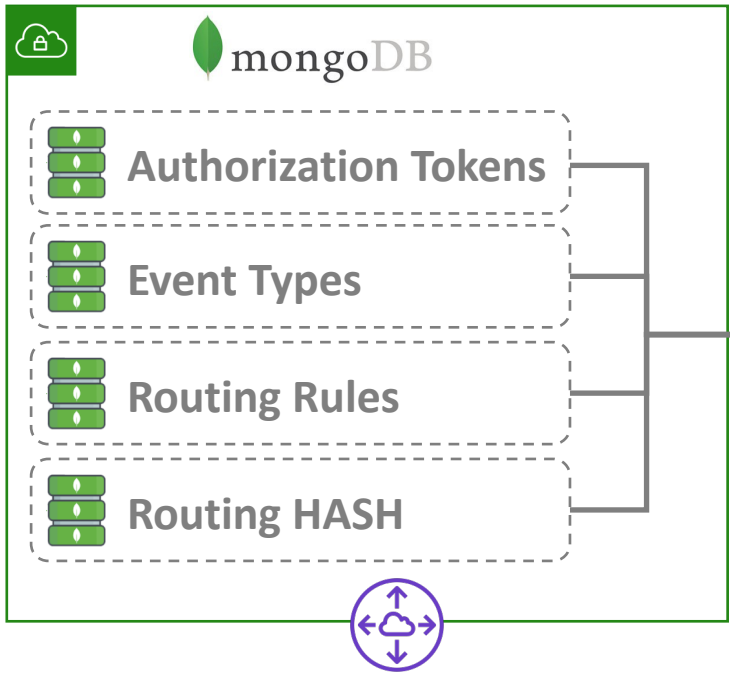
Kubernetes

Docker Container

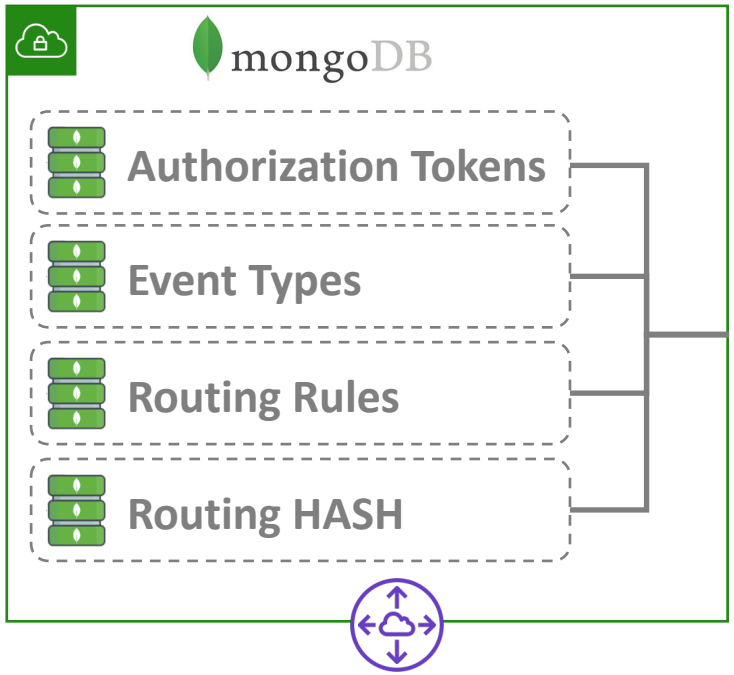
Java

Admin API

# Configuration / Admin Collections



# Configuration / Admin Collections




**Event Types**

```
{
    "key": "CENPAS:OPEN:CASE",
    "noun": "CENPAS",
    "verb": "OPEN",
    "object": "CASE"
},
```

**Routing Rules**

```
"id": "5cf6c54206e6450001c3781c",
"name": "cenpas-all",
"effective": {
    "start": 0,
    "end": 9223372036854775807
},
"topic": "kafk.centene.centevent.eventframework.cenpas.routing",
"criteria": [
    {
        "eventType": "CENPAS:OPEN:CASE",
        "tags": []
    },
    {
        "eventType": "CENPAS:INPROGRESS:CASE",
        "tags": []
```

**Routing HASH**

```
"id": "5cf12fee6126dc0001db38d4",
"key": "CENPAS:OPEN:CASE",
"routes": [
    {
        "ruleId": "5cf19a826126dc0001db38e0",
        "topic": "kafk.centene.centevent.eventframework.cenpas.claims.correspondence",
        "effective": {
            "start": 0,
            "end": 9999999999999999
        },
        "tags": []
    },
    {
        "ruleId": "5cf589bdadf7b000011587a8",
        "topic": "kafk.centene.centevent.eventframework.cenpas.routing",
        "effective": {
            "start": 0,
            "end": 9223372036854775807
        },
        "tags": []
    }
```

# K-Streams Application Code

*1) Fetch the Event Key --> Key = N:V:O*

*2) Join Hashed Routing Rules: maps N:V:O to Consumers that have Subscribed*

*3) Discard Events if no Consumer has Subscribed*

*4) Generate Duplicate Event message; one for each subscribed Consumers*

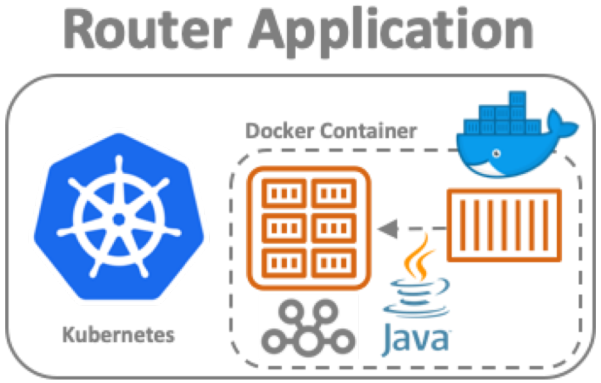
*5) Validate Consumers effective date is within Event date*

*6) Validate the Event contains tags the Consumer specifies*

*7) Validate the Consumers PHI permissions are appropriate*

*8) Place the Event destination topic onto the Kafka message header (metadata)*

*9) Send the event to the destination topic specified in the message header*



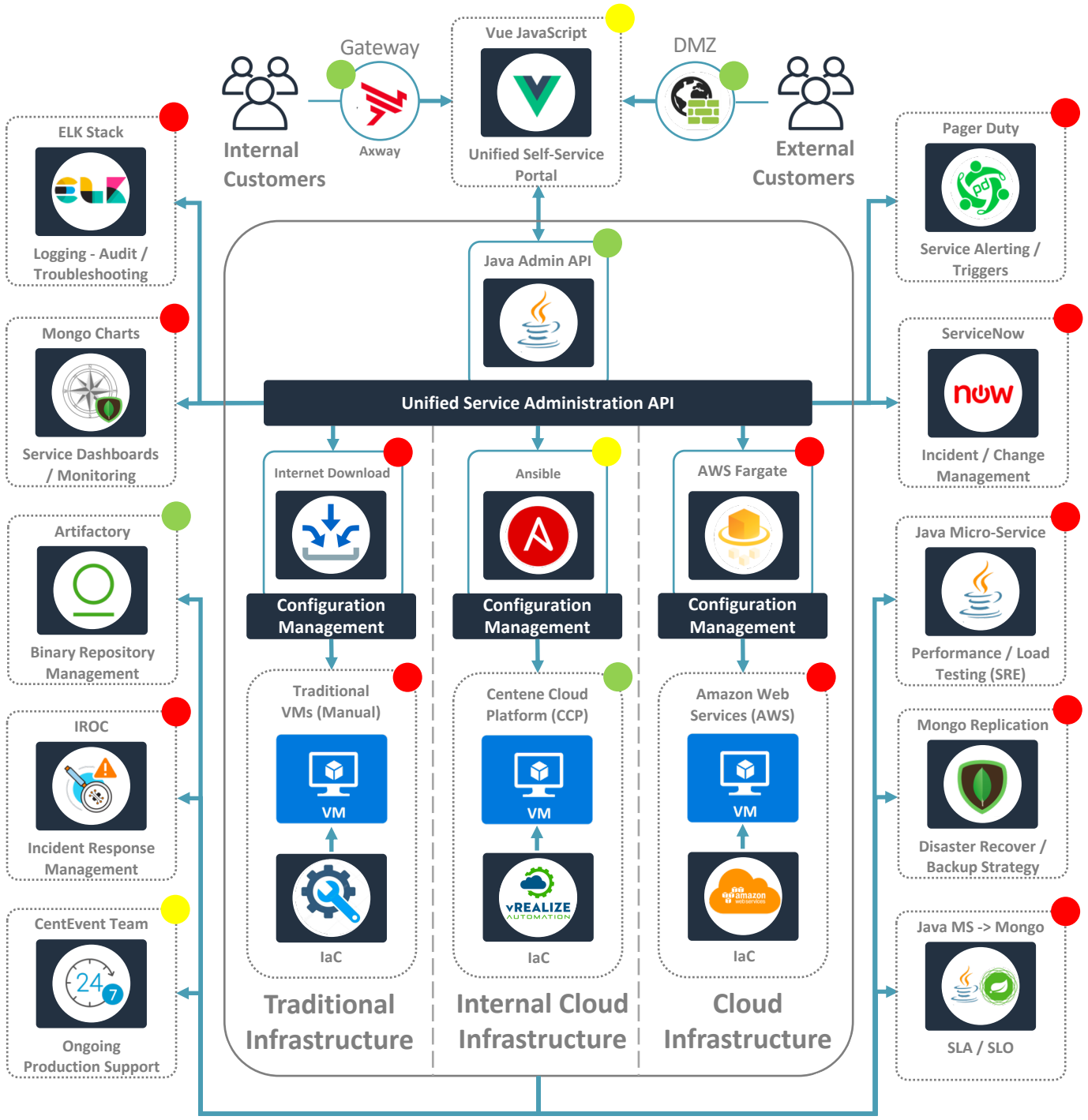

```
public void routeEvent(KStream<String, Event> stream) {
    stream.selectKey((id, event) -> event.getMetadata().getType()) KStream<EventType, Event>
            .map((key, value) -> new KeyValue<>(value, eventTypeRoutingService.findEventTypeRouting(EventTypeUtil.toString(key)).getRoutes())) KStream<Event, List<EventRouteDocument>>
            .peek((k, v) -> log.debug("key: {}, routes: {}", k, v)) KStream<Event, List<EventRouteDocument>>
            .filter((key, value) -> !value.isEmpty()) KStream<Event, List<EventRouteDocument>>
            .flatMapValues((readOnlyKey, value) -> value) KStream<Event, EventRouteDocument>
            .filter(this::effectiveDateRangeFilter) KStream<Event, EventRouteDocument>
            .filter(this::tagFilter) KStream<Event, EventRouteDocument>
            .filter(this::phiFilter) KStream<Event, EventRouteDocument>
            .map(this::updateRoutedTopic) KStream<capture of ? extends EventKey, capture of ? extends Event>
            .peek((k, v) -> log.debug("routing complete, sending to {}", v.getMetadata().getRoutedTopic())) KStream<capture of ? extends EventKey, capture of ? extends Event>
            .to((key, value, recordContext) -> value.getMetadata().getRoutedTopic());
}
```

CENTENE Corporation

- Complete
- In-progress
- Incomplete

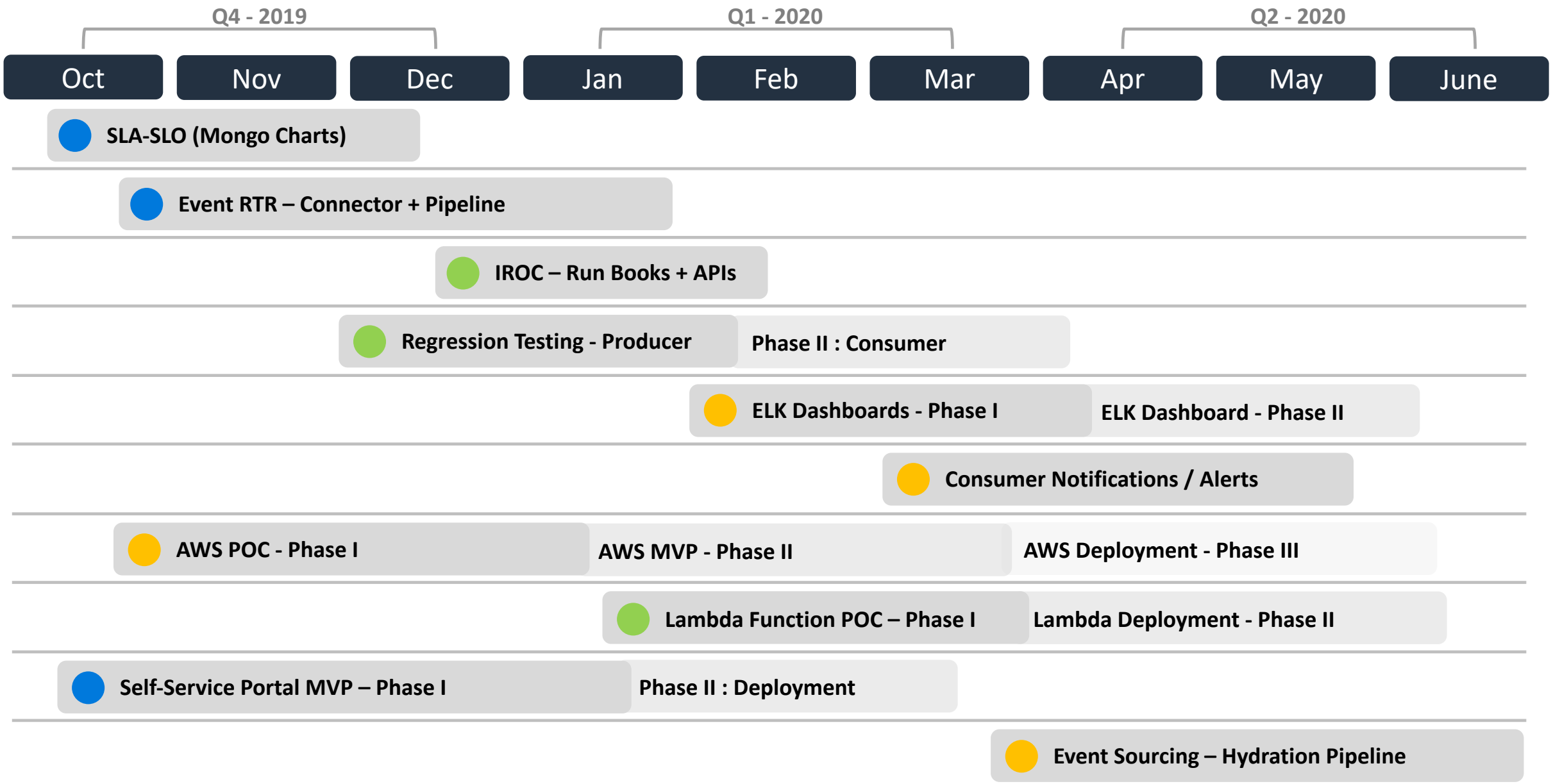
# Eventing as a Service - CentEvent

- VM Provisioning / Management
- Infrastructure as Code (IaC)
- Configuration Management
- Service Administration API
- Service Dashboards / Monitoring
- Incident / Change Management
- Service Alerting / Triggers
- Service Logging (Audit / Troubleshooting)
- Self-Service Portal
- Automated Performance / Load Tests (SRE)
- Disaster Recovery / Backup Strategy
- SLA / SLO (Defined and Captured)
- Binary Repository Management
- Incident Response Management (IROC)
- Ongoing Production Support



# CentEvent Roadmap

Operations | Product Maturity | Demand



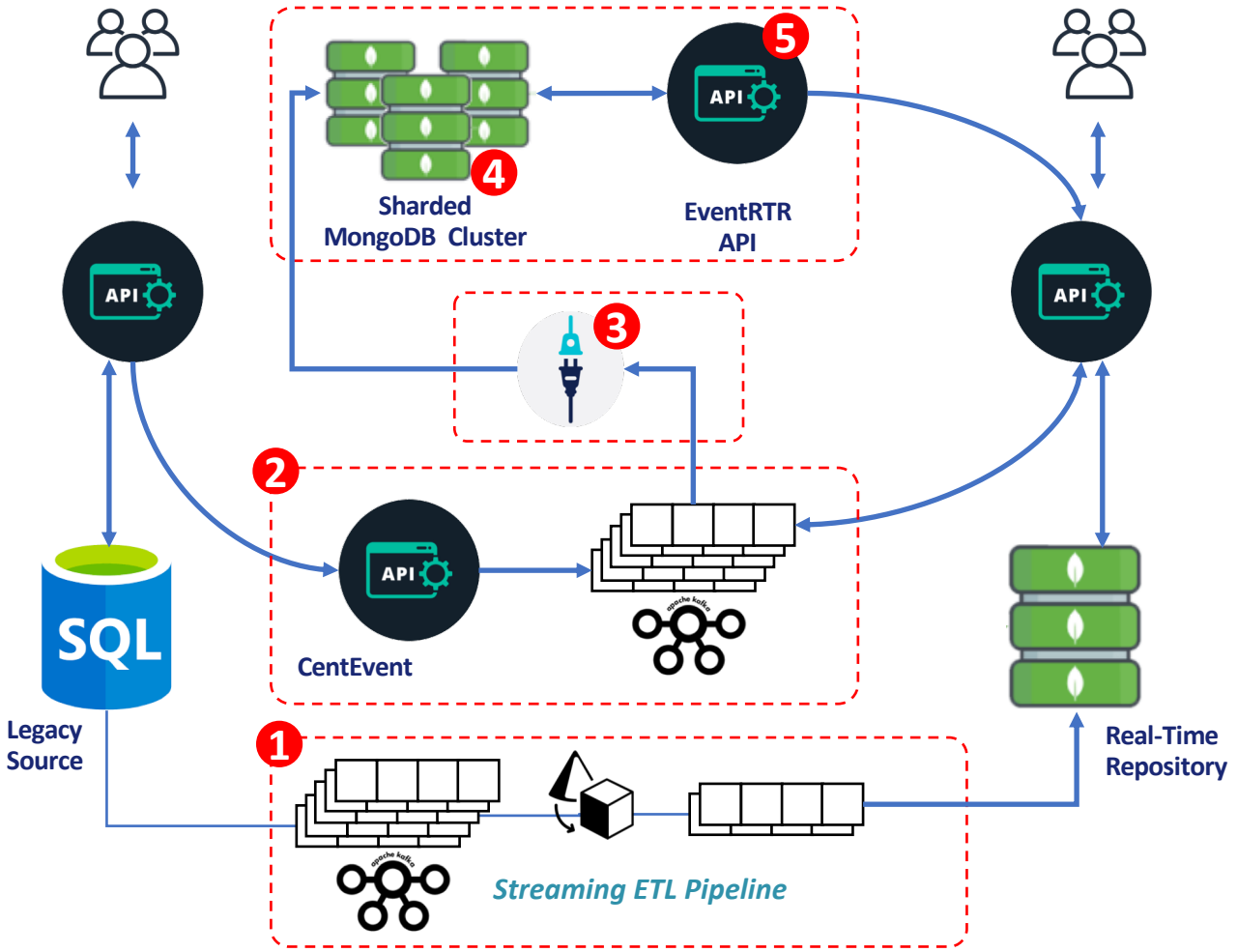
# Universal Centene Integration Pattern

## *Key Components:*

1) CDC Streaming ETL Pipelines
2) CentEvent (Event Mediator)
3) MongoDB Connector (modified)
4) MongoDB Cluster
5) Event RTR API

## *Framework Features:*

- Synthetic Events
- PHI Filtering
- Event Tagging
- Distributed Tracing
- Event Routing
- ELK Monitoring
- Self-Service Portal
- Event Discovery UI





# Application Refactor Use Case

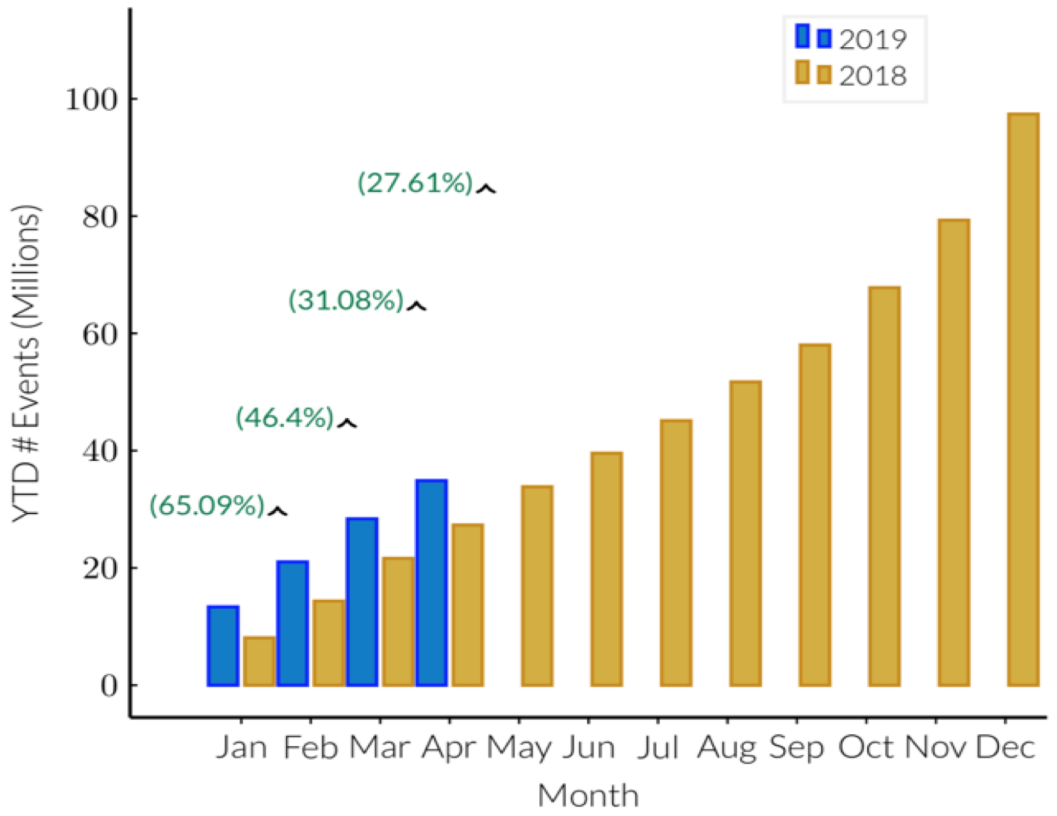
## *Member Insight Platform Migration*

- *Refactor from self-managed Kafka to KaaS*
- *Refactor from Member Insight API to CentEvent*

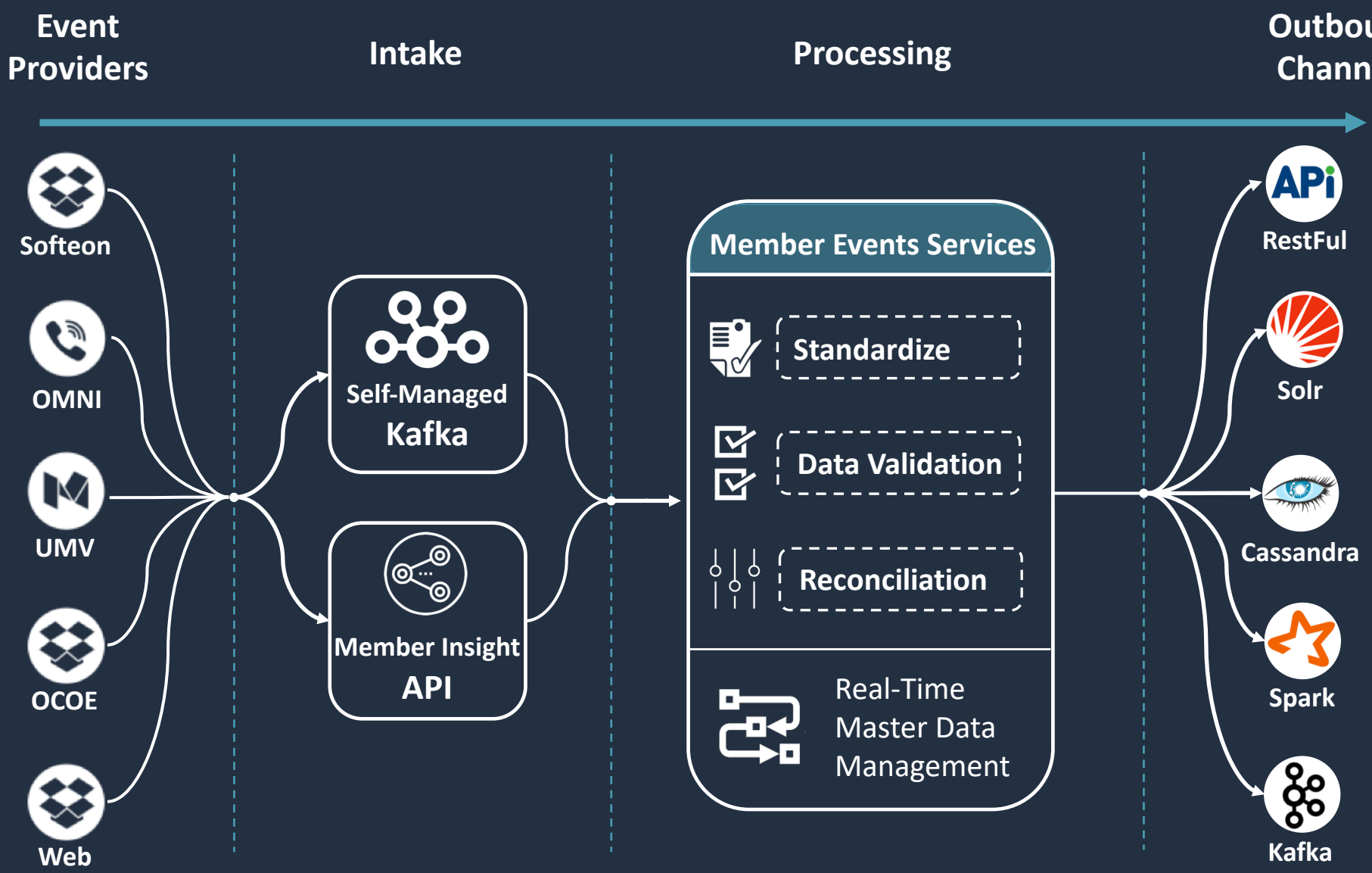
| Total Events Volume | YTD Inbound Calls | YTD Correspondence |
|---|---|---|
| 191,000,000 | 16,000,000 | 10,250,000 |

| YTD Member Services | YTD Outbound Calls |
|---|---|
| 4,350,000 | 4,070,000 |

***Member Event Volume continues to grow significantly - increasing the importance of a stable and scalable underlying infrastructure of services***



# Member Insight Process Flow: Before



# Member Insight Process Flow: After