# Black Box Monitoring in Go

## An introduction to Building Black Box monitors Go

**Grant Griffiths**
**Software Engineer**
**Platform Cloud Engineering**
**GE Digital**

PREDIX

# Who am I?

GE Digital - Platform Cloud Engineering

Sr. Software Engineer

San Francisco, CA

GE Go User Group

Founder & Organizer

@ggriffiths

@griffithsgrant

## Climbing + Mountaineering

1. **Engineering @ GE**
2. Introduction to Black Box Monitoring
3. Black Box monitoring @ GE
4. Sample black box monitor
5. Takeaways

# Who here has owned a GE Fridge, Lightbulb, or Microwave?


Actually, we make
HUGE Industrial assets...


2029
GECX

# Fun facts about GE

- GE Power generates roughly 33% of the world's electricity
- Every two seconds, an aircraft powered by GE takes off
- 35,000 wind turbines globally
- 25% of all global hydropower

# Some customers

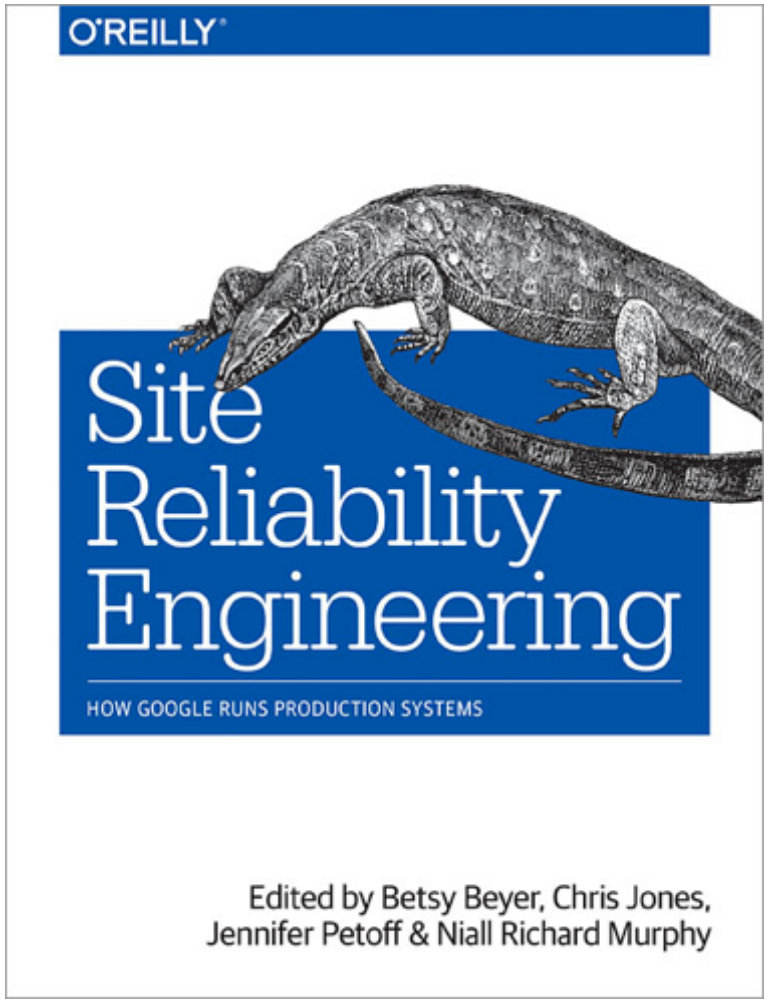
- Schindler
- Exelon
- Rosneft
- BP
- GE Power
- GE Aviation
- GE Renewables

# Industrial Internet of Things (IIoT)

- GE Assets produce petabytes of useful and mission critical data
- Store and analyzed in Predix, our platform for IIoT
- Downtime can have huge impact – lives lost and million of dollars
- We must focus on reliability and availability

1. Engineering @ GE
2. **Introduction to Black Box Monitoring**
3. Black Box monitoring @ GE
4. Sample black box monitor
5. Takeaways

# Google SRE book



# Site Reliability Engineering

- “Fundamentally, it’s what happens when you ask a software engineer to design an operations function.”

  *- Ben Treynor, VP of Engineering @ Google*

- Using software expertise to solve operations problems through automation
- Google SRE Book - landing.google.com/sre/book
  - Additionally, (**Site Reliability workbook free** to download until **Aug 23**)
  - Covers monitoring amongst many other topics

# What is Black Box Monitoring?

- Testing externally visible behavior as a user would see it
- Questions that it answers
  - Is my service up?
  - Is my service consumable?
  - Is a dependency of my service down?
- Node uptime != service uptime

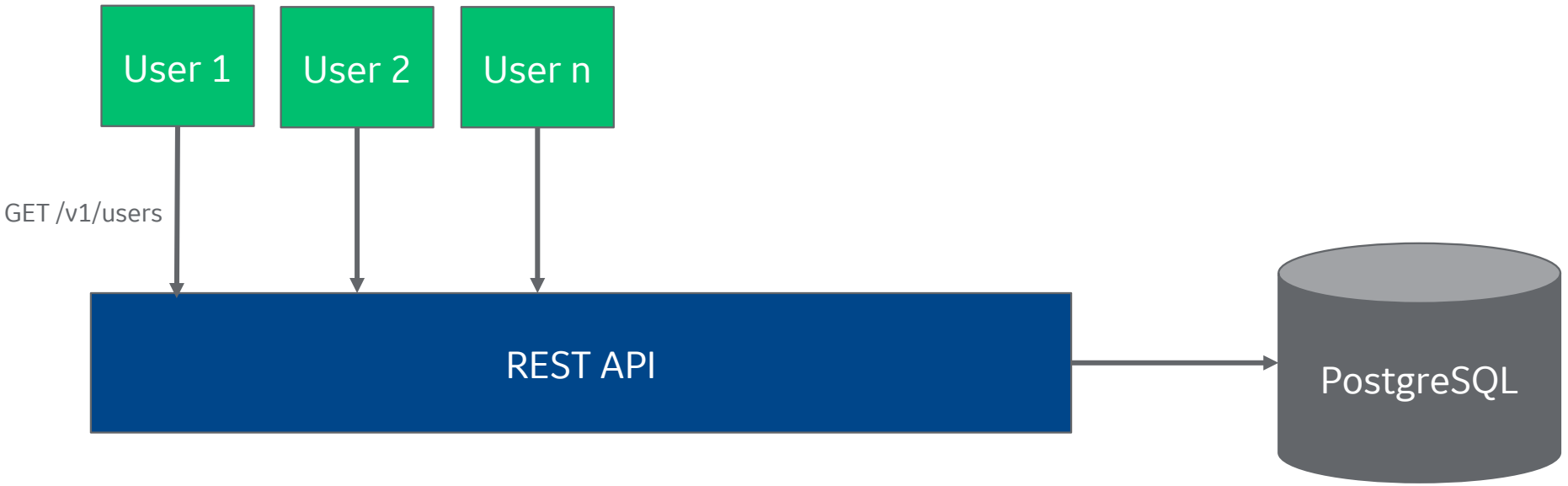
# What is NOT Black Box monitoring?

- White box monitoring
  - CPU/RAM/Disc utilization
  - JVM stats
  - pprof/tracing
  - Logging
  - System internals
- Individual component health checks
- Anything the user cannot find out

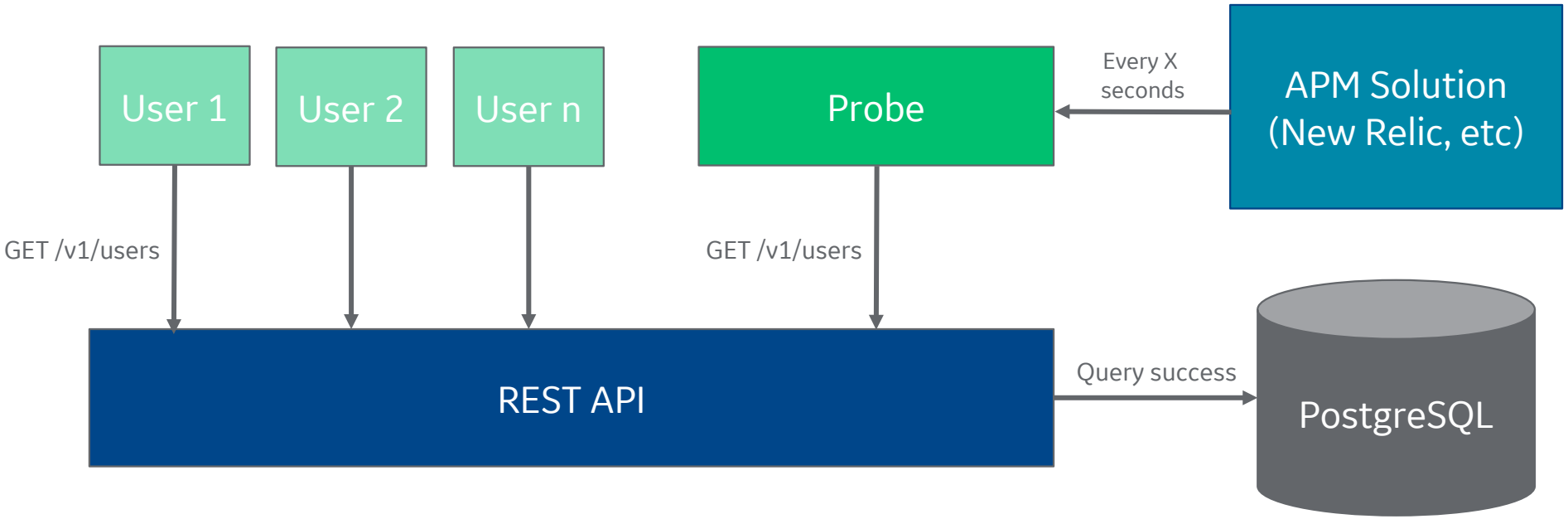
# What are probes?

- Small applications that measure service uptime
- Knows nothing about system internals
- Simulates an end user (black box)
- Automated, pages engineers upon failures

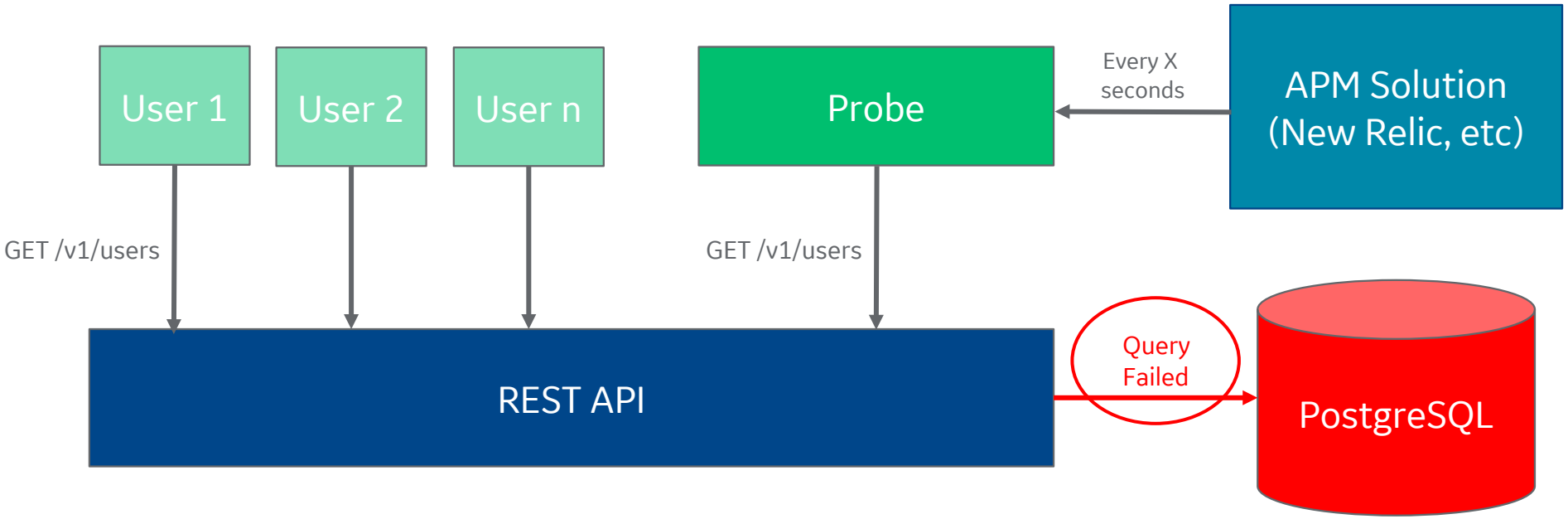
# Simple REST API Architecture



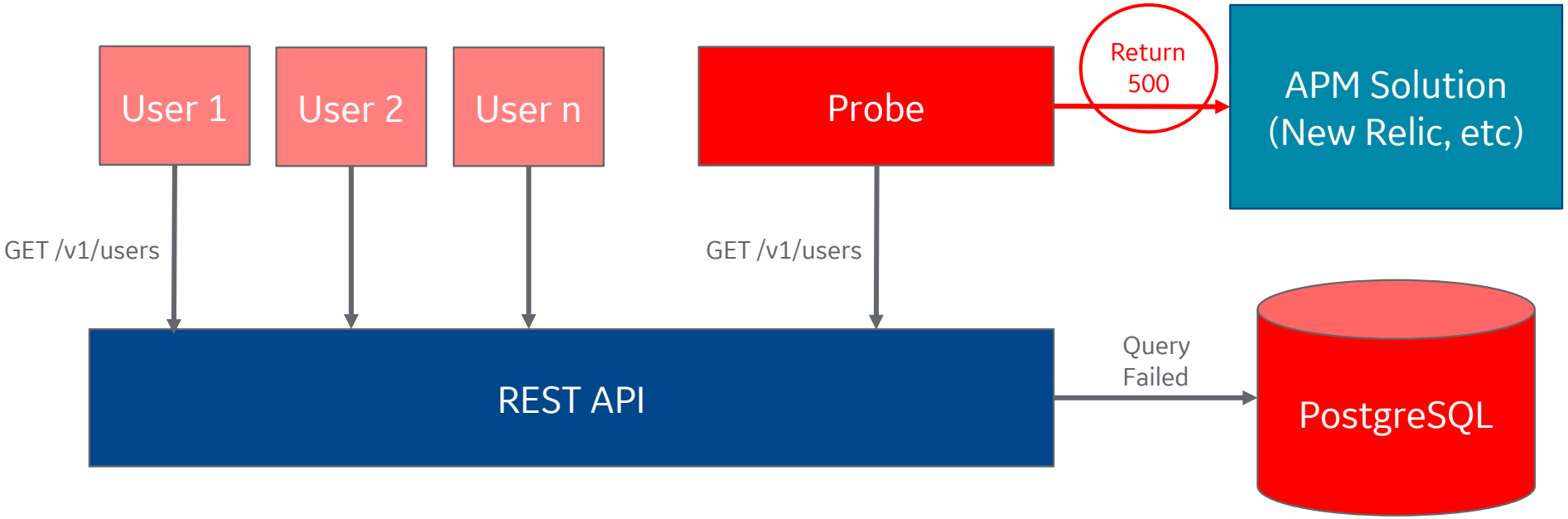
# Let’s add a Probe to monitor it



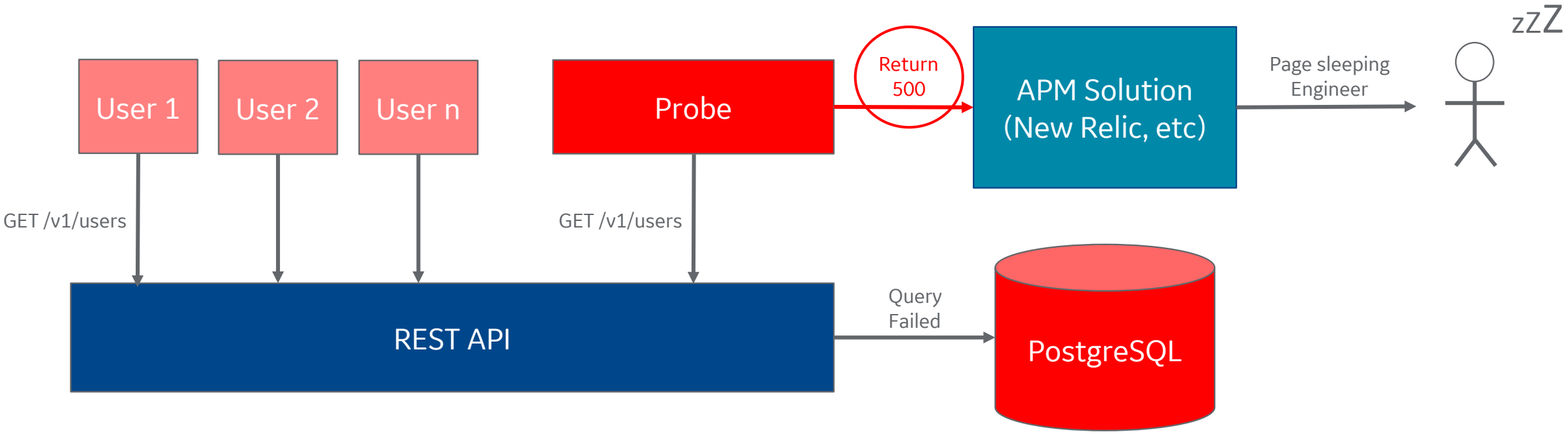
# Service queries DB



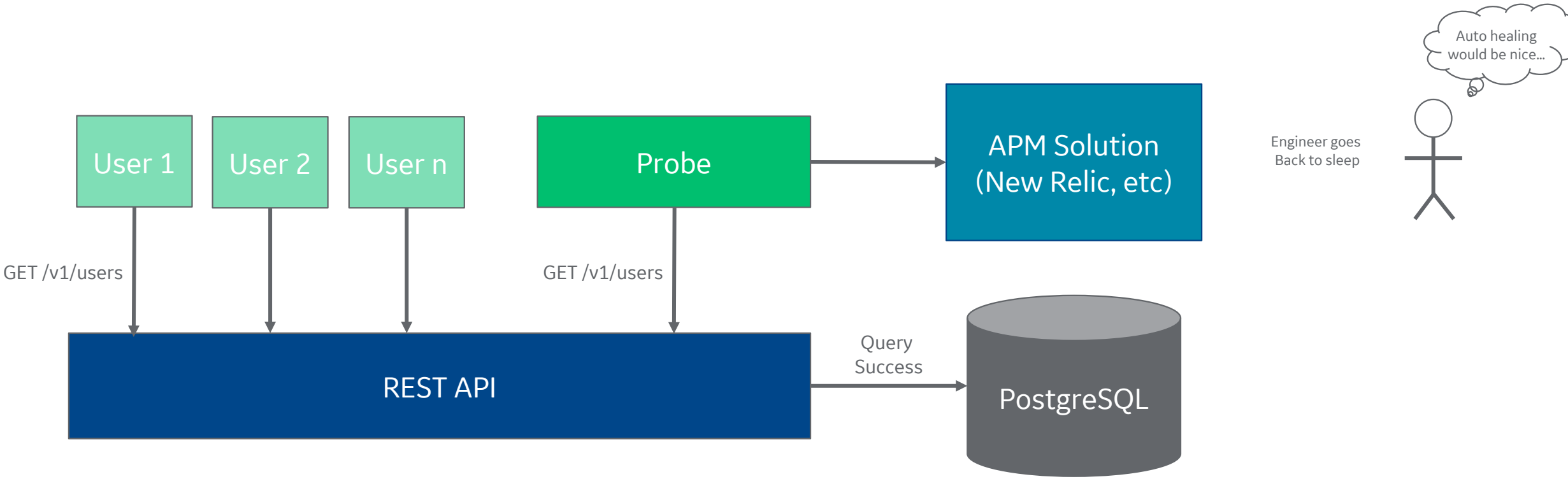
# Probe returns 500



# Page engineer on PostgreSQL team



# Monitoring a simple REST API – Problem fixed by engineer



# Why Go for these probes?

- Easily simulating end users w/ Time package
- Simple and concise HTTP package for interacting with services
- Create HTTP endpoints for new relic or other services
- Fits with cloud & SRE ecosystem

# Probe best practices

- Check components that will provide the most visibility
- Should be lightweight in deployment and operational cost
- Probe status should be binary – your service is either up or down

# Advanced probe features

- Service SLA measurements
- Detailed status on service components
- Load spikes and behavior
- Auto healing systems
- Randomized user activity

1. Engineering @ GE
2. Introduction to Black Box Monitoring
3. **Black Box monitoring @ GE Digital**
4. Sample black box monitor
5. Takeaways

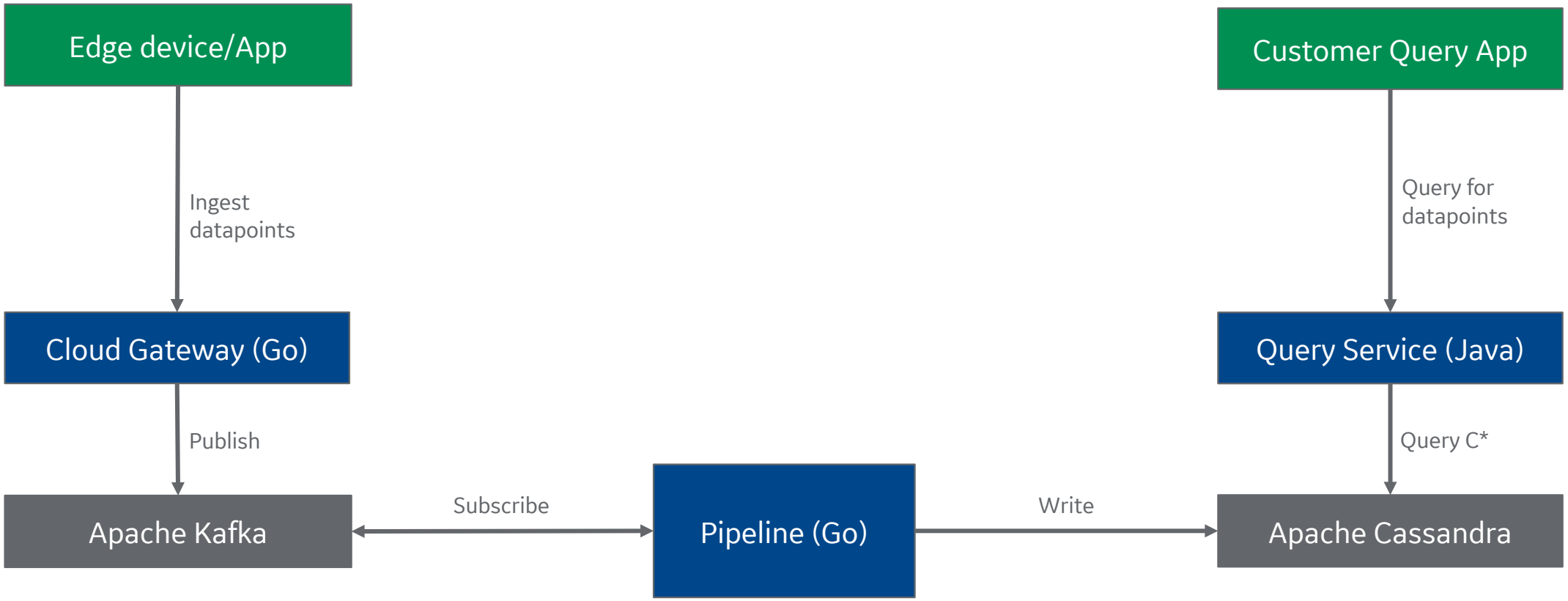
# Monitoring and Diagnostics service **(M&D)**

- Stores time series data
- Store datapoints via WSS
- Query datapoints via REST
- Internal components for parsing, writing, etc
  - Many moving parts
  - Black box monitor covers all components

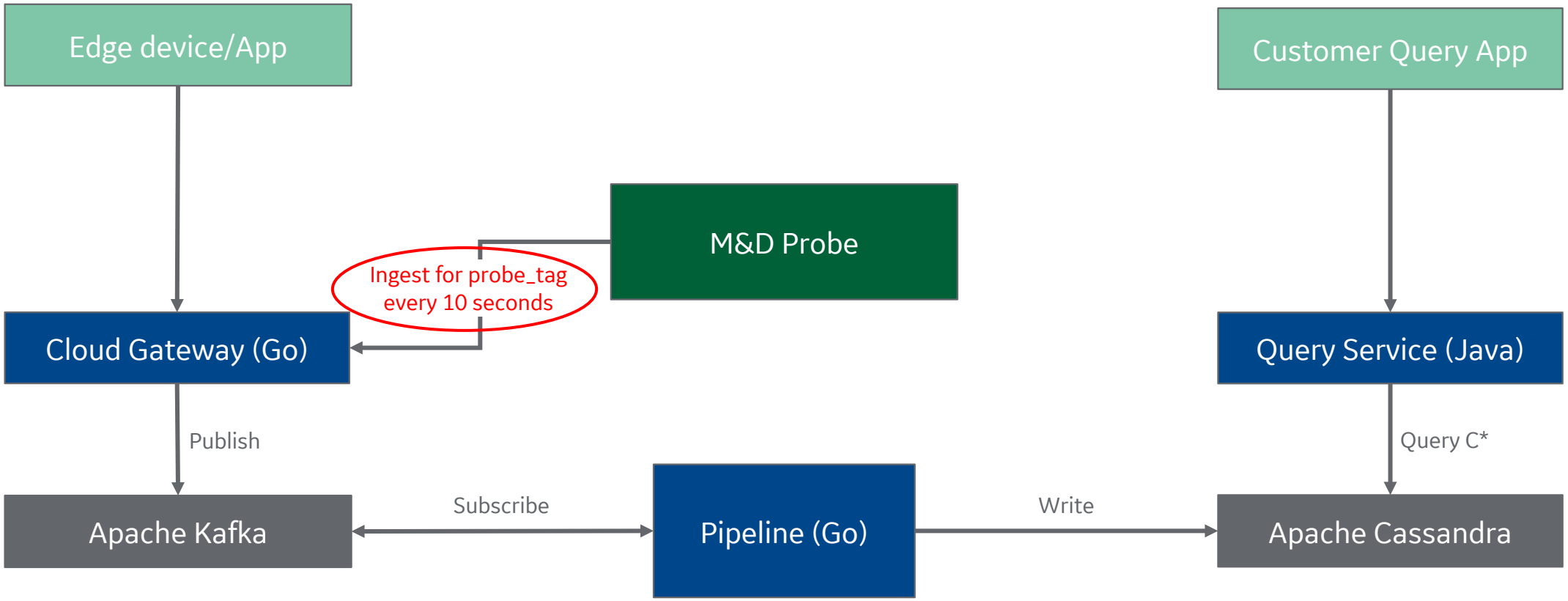
**M&D**
**Ingestion Message**

```
{
  "messageId": "<MessageID>",
  "body":[{
    "name":"<TagName>",
    "datapoints":[
       [ <Timestamp>, <Value>, <Quality> ],
       [ <Timestamp>, <Value>, <Quality> ],
       [ <Timestamp>, <Value>, <Quality> ]
    ]
  }]
}
```

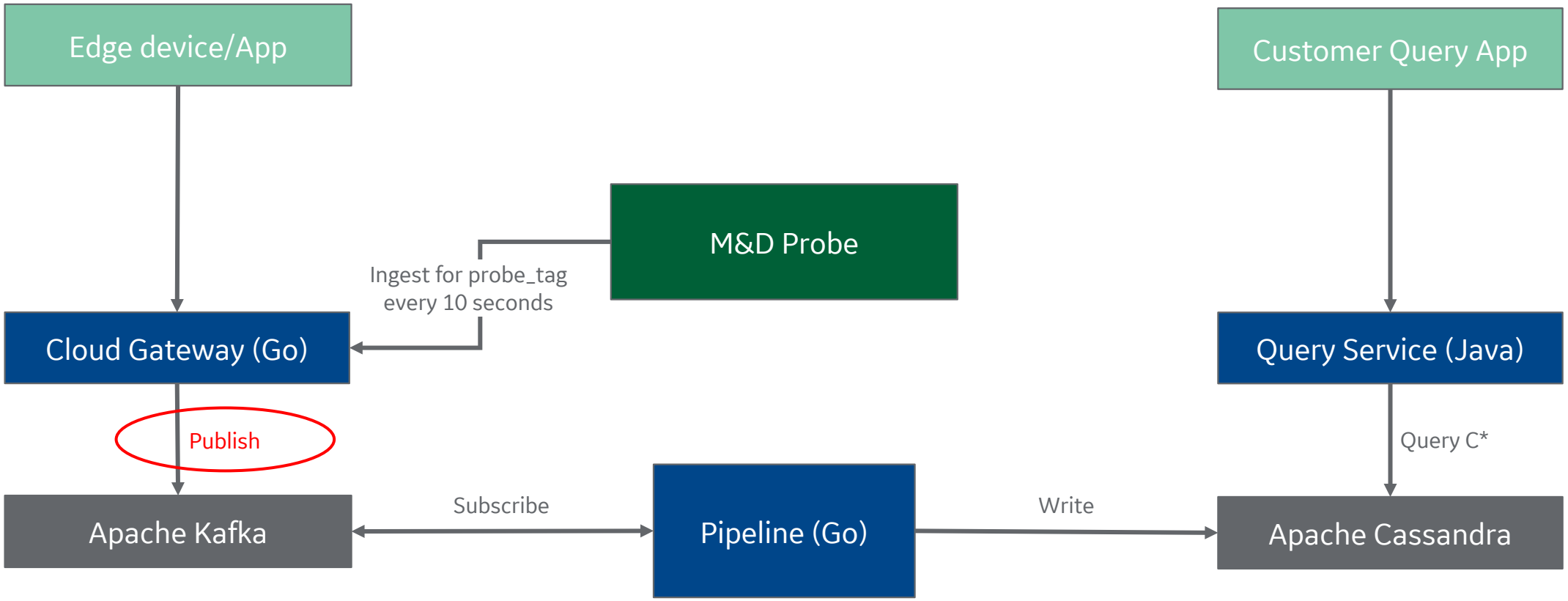
# Monitoring and Diagnostics service **(M&D)**



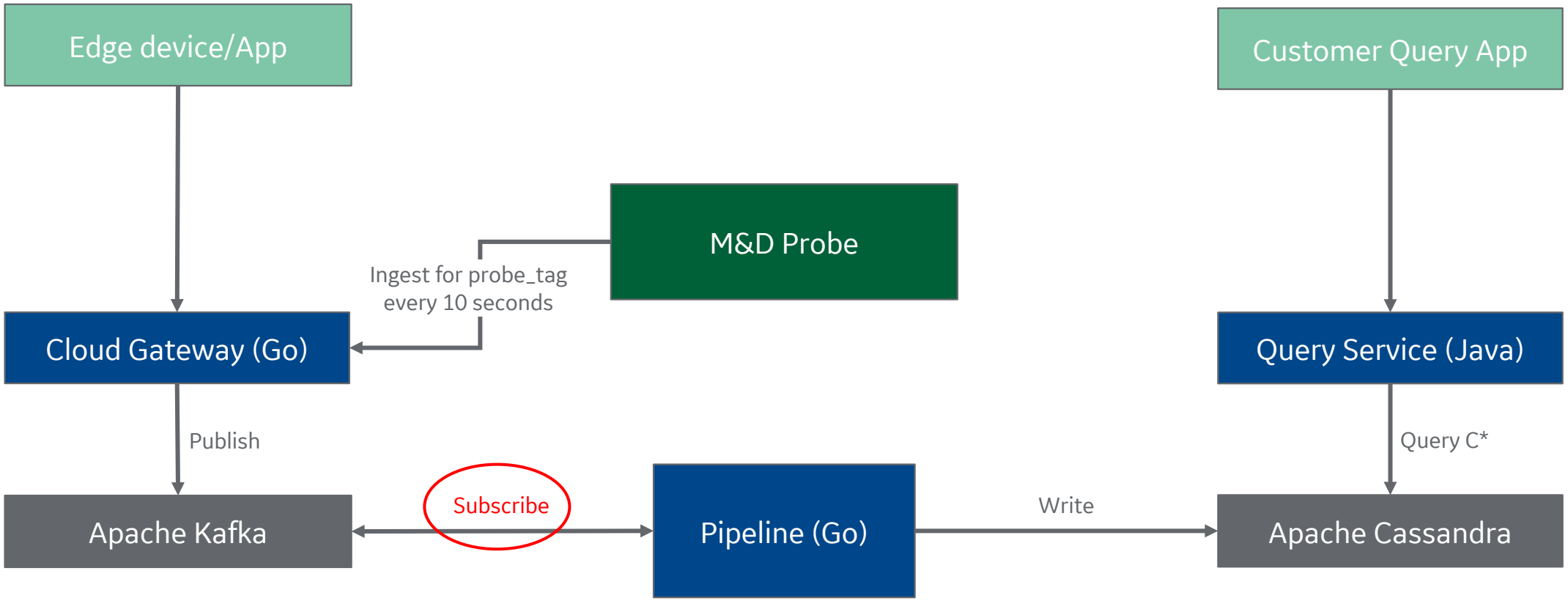
# Probe ingests to cloud gateway every 10 seconds



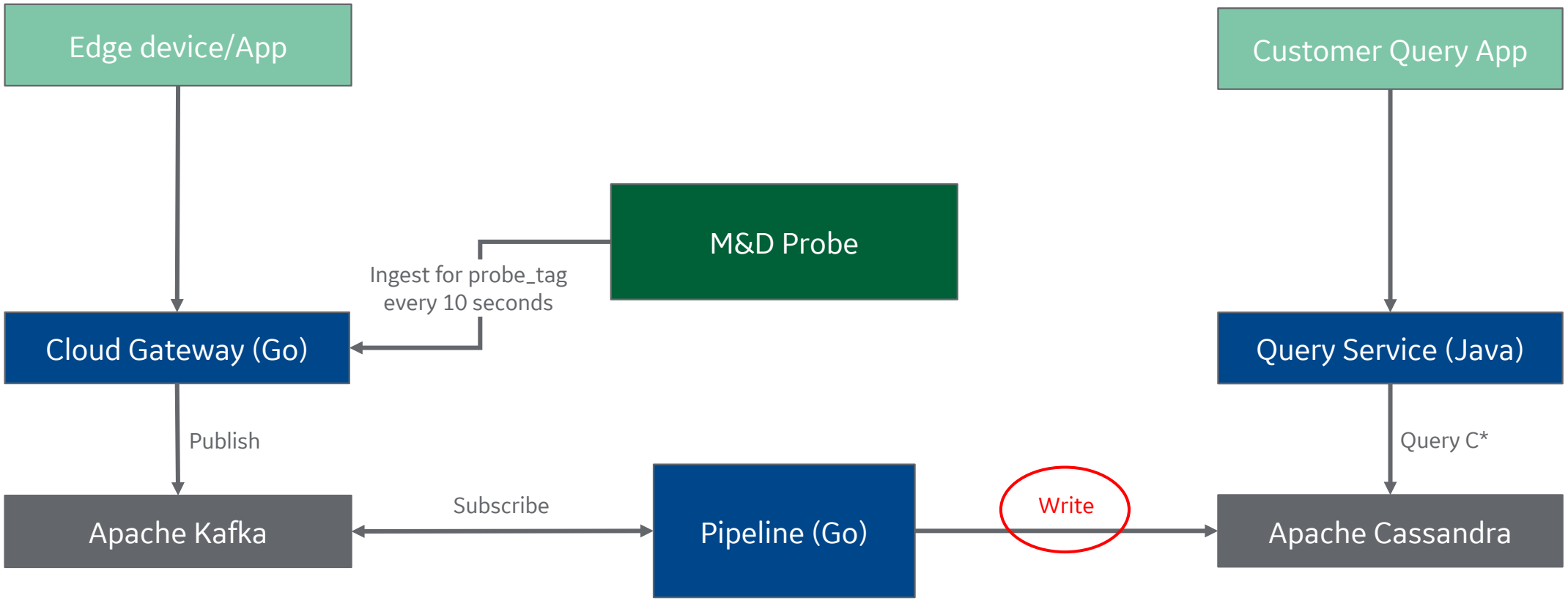
# Cloud gateway publishes to Kafka



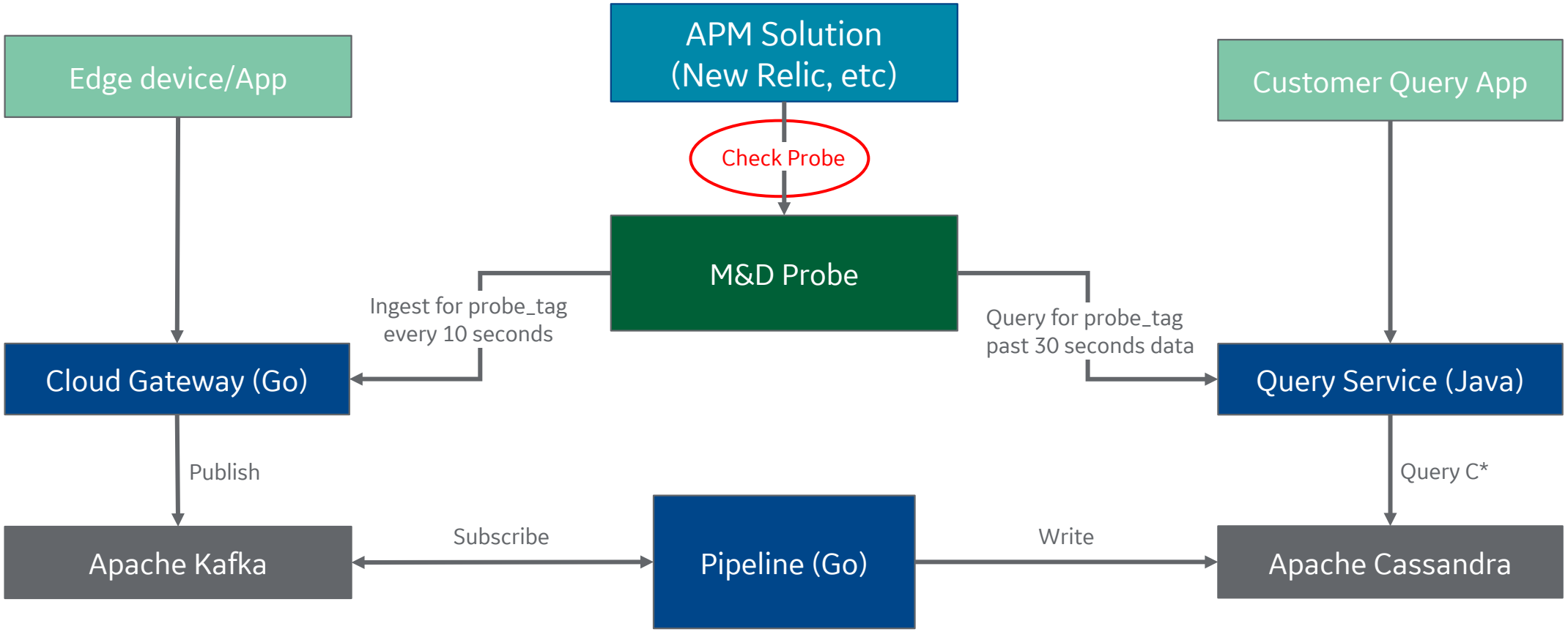
# Pipeline parses data from Kafka



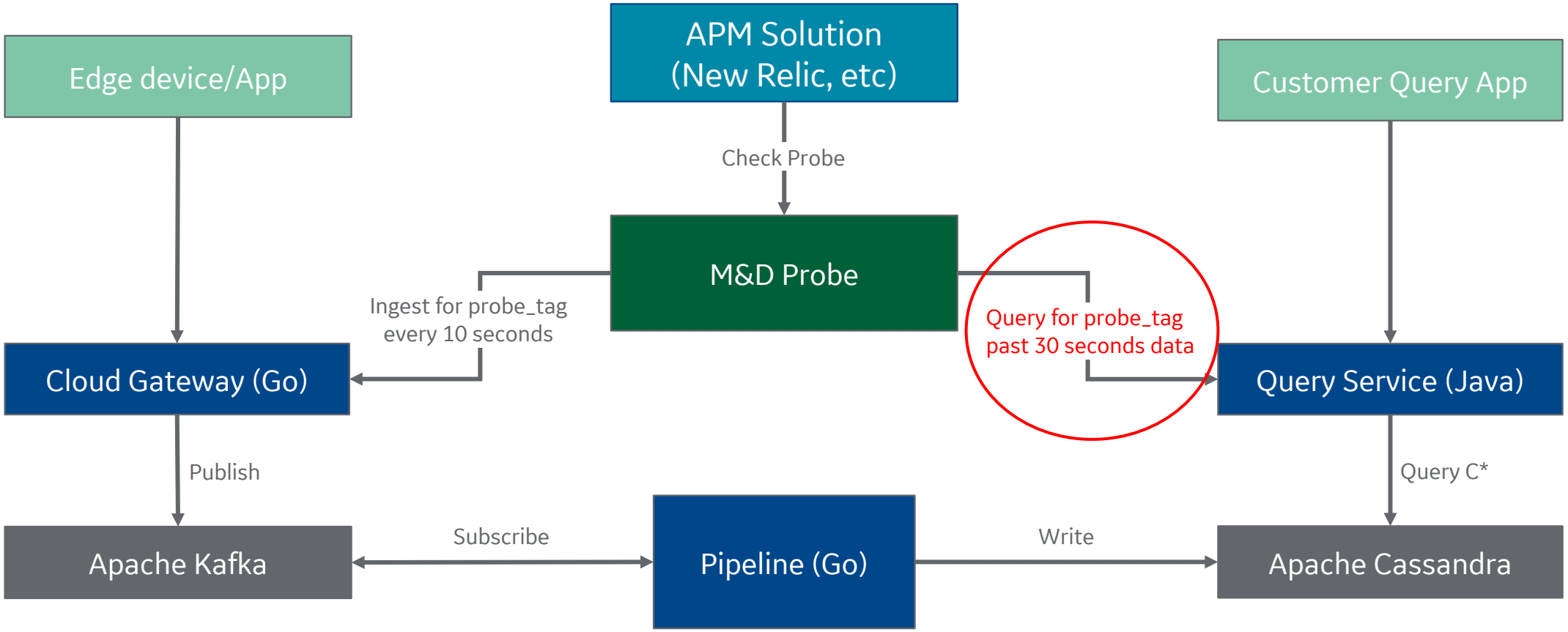
# Pipeline writes to C*



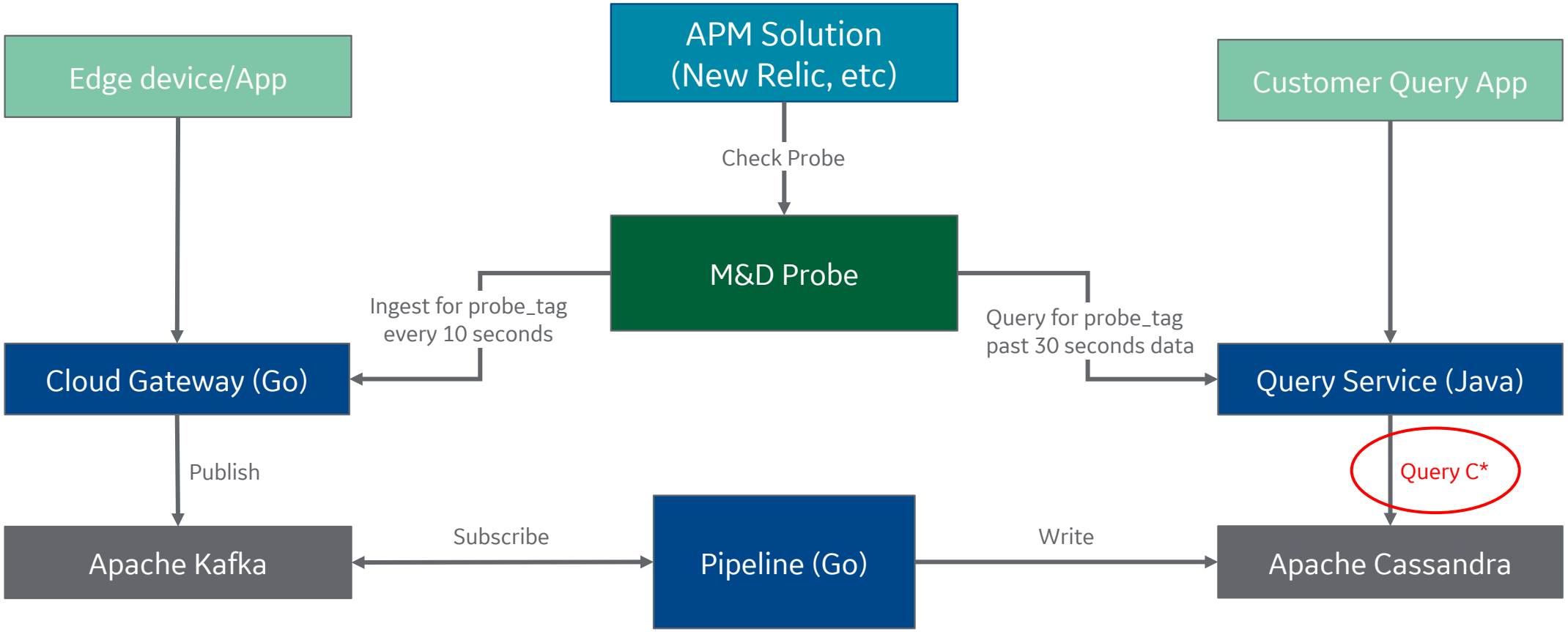
# Every X seconds, APM solution checks the probe



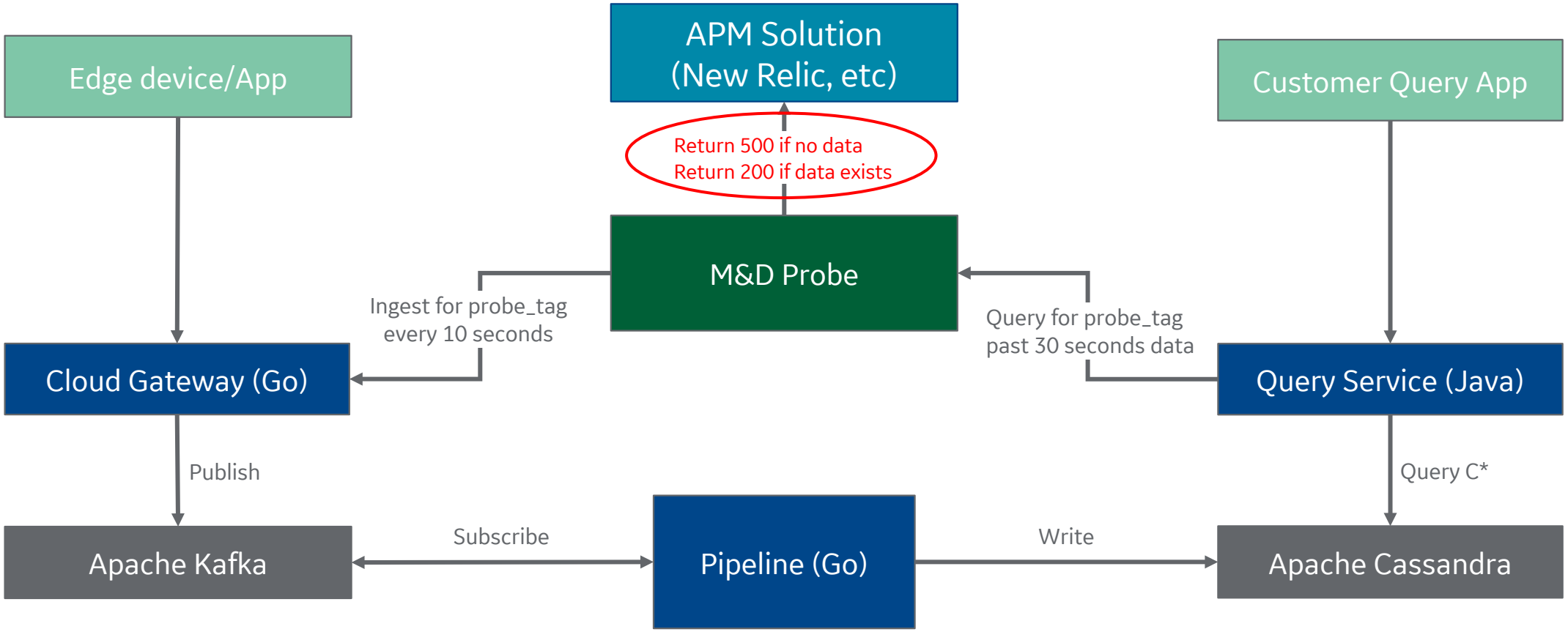
# Probe queries for canary_tag data in the past 30 seconds



# Query service checks C*



# Returned to APM solution



# Other probes at GE Digital

- Monitors our big data scheduling & execution service (**Predix insights**)
  - Submit Airflow DAG, Execute analytic at an interval
  - Check for analytic success
- Monitors our service instance creation
  - Create, Bind, Unbind, Delete service instances
- Predix Event Hub, Blobstore, Columnar datastore

1. Engineering @ GE
2. Introduction to Black Box Monitoring
3. Black Box monitoring @ GE
4. **Sample black box monitor**
5. Takeaways

# Hello world service

```
package main

import (
    "fmt"
    "net/http"
    "os"
)

func main() {
    http.HandleFunc("/hello", func(w http.ResponseWriter, r *http.Request) {
        fmt.Fprintf(w, "Hello world")
    })

    http.ListenAndServe(os.Getenv("PORT"), nil)
}
```

# The probe interface

```
// Probe provides a common interface for building service probe health checks
type Probe interface {
    Start(username, password string) error
    Heartbeat()
    Check() error
}
```

# Sample probe implementation – high level flow

```go
// ApplicationProbe logic goes here.
type ApplicationProbe struct{}

// NewApplication creates a new probe for your application
func NewApplication() *ApplicationProbe {
    return &ApplicationProbe{}
}

// Start will start up the probe monitoring
// application with a basic auth protected endpoint
func (a *ApplicationProbe) Start(username, password string) error {
    logger.Debug("Probe is taking off!", "function", "main()")

    go a.Heartbeat()

    healthcheckHandler := handler.NewHealthChecker(a.Check, true)
    basicauthHandler := handler.NewBasicAuthenticator(username, password)
    http.Handle("/application/probe/status", basicauthHandler.Wrap(healthcheckHandler))
    return http.ListenAndServe(":"+os.Getenv("PORT"), nil)
}
```

# Sample probe implementation – checks

```go
// Heartbeat will start up a goroutine that contantly
// performs a given action every sp often
func (a *ApplicationProbe) Heartbeat() {

}

// Check checks that LeaveNest has been running successfully
func (a *ApplicationProbe) Check() error {
    resp, err := http.Get("localhost:8080/hello")
    if err != nil {
        return err
    }

    if resp.StatusCode != http.StatusOK {
        return errors.New("Health check failed")
    }

    return nil
}
```

1. Engineering @ GE
2. Introduction to Black Box Monitoring
3. Black Box monitoring @ GE
4. **Sample black box monitor**
5. Takeaways

# Takeaways

- Node uptime != service uptime
- Use Go to build black box monitors
- Simulate end user traffic
- GE uses this concept for many services in prod
- Catches many production issues early on that white box monitoring might not

# The “we’re hiring” slide

PREDIX

# Questions?

PREDIX