Network Monitoring System with SNMP

This presentation provides a comprehensive overview of network monitoring solutions using SNMP, highlighting their benefits and implementation strategies.

Importance of Network Monitoring

Understanding the Necessity of Monitoring Networks

- **Ensures Network Reliability**

  Network monitoring guarantees consistent uptime and minimizes disruptions.

- **Enhances Performance**

  Proactive monitoring helps in identifying and resolving performance bottlenecks efficiently.

- **Reduces Manual Monitoring Efforts**

  Automation in monitoring eliminates the need for constant manual oversight, saving time.

- **Facilitates Quick Issue Detection**

  Real-time alerts enable rapid response to issues, minimizing downtime and losses.

- **Supports Business Continuity**

  Ensures that critical business operations remain uninterrupted through consistent oversight.

Innovative Solutions for Network Monitoring

Addressing Network Monitoring Challenges Effectively

- **Utilizes SNMP for Monitoring**

  The system leverages Simple Network Management Protocol to automate device monitoring effectively.

- **Reduces Manual Effort**

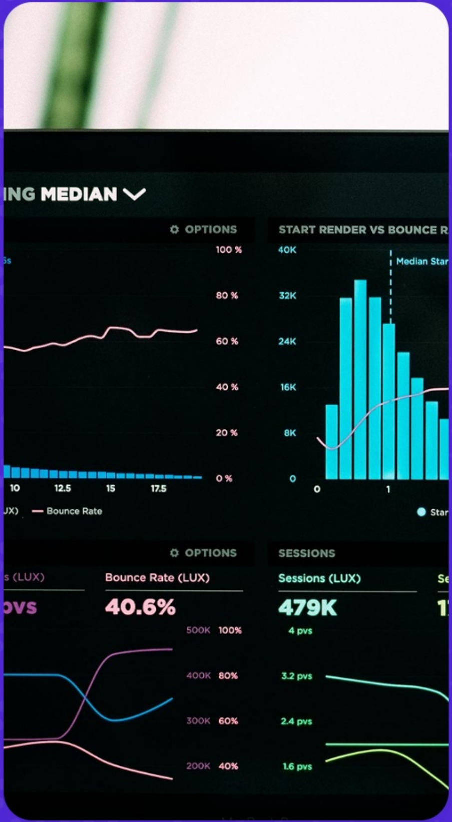
  Automated monitoring minimizes the need for manual checks, saving time and resources.

- **Faster Issue Detection**

  Real-time insights enable quicker identification of network issues, enhancing response time.

- **Enhances Automation Capabilities**

  The system supports advanced automation features, streamlining network management tasks.



Comprehensive Technology Stack Overview

Exploring the components of our network monitoring system

- **Go Programming Language**

  Utilized for the polling engine and data collection processes, ensuring efficient performance.

- **PostgreSQL Database**

  Selected for its robust storage capabilities, particularly effective with JSON indexing.

- **Vert.x Framework**

  Employed for managing API requests and enabling event-driven communication in the system.

- **ZeroMQ (ZMQ)**

  Facilitates rapid brokerless communication between Go and Vert.x, enhancing system responsiveness.

User Workflow Steps for Device Polling

A detailed overview of the user steps

- **Create Credential Profile**

  Establish a reusable profile with login details for SNMP devices.

- **Create Discovery Profile & Run Discovery**

  User initiates the discovery process to identify available devices.

- **Start Polling (if successful)**

  Begin data collection from discovered devices if the discovery was successful.

- **View Data via Provisioning Job ID**

  Users can fetch real-time metrics using the job ID from provisioning.

Enhanced System Features Overview

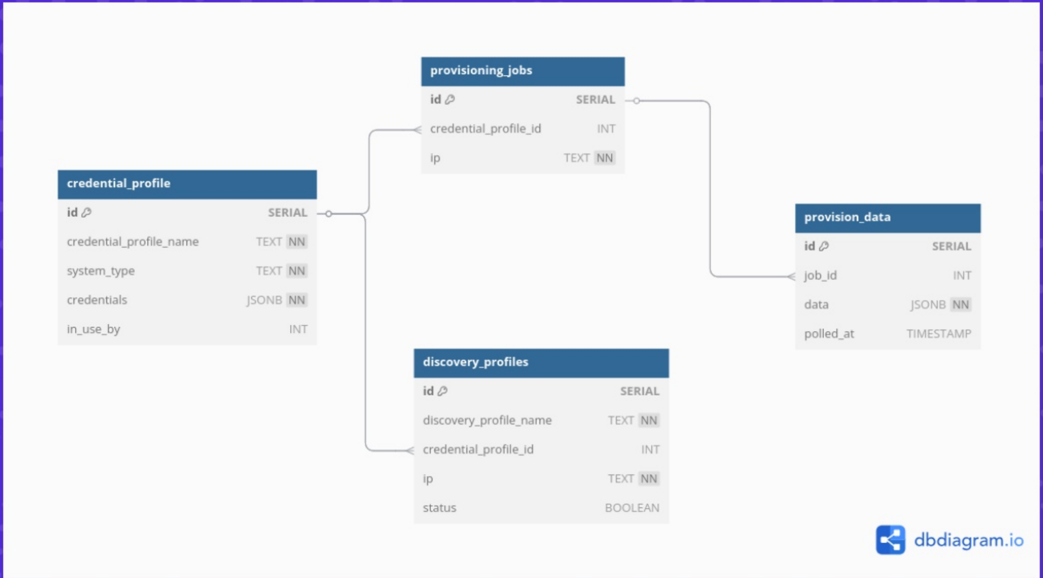
Explore the System's Key Metrics and Features

- Top Interfaces by Error Rate
  - Identify interfaces with the highest error rates to target improvement efforts.
- Top Interfaces by Speed
  - Highlight interfaces with optimal speed for better performance assessment.
- Top Interfaces with Most Restarts
  - Analyze interfaces that frequently restart to enhance system stability.
- Collected Metrics Overview
  - Comprehensive metrics collected for each device to support performance monitoring.
- System Name & Description
  - Details about each system's identification and function for clarity.
- System Location
  - Geographical data on system locations to facilitate network management.
- System Uptime
  - Monitoring system uptime to ensure reliability and availability.
- Total Packets Received/Sent
  - Track the total packets to evaluate data flow and performance.
- Interface Speed Metrics
  - Measure the speed of each interface to ensure optimal performance.
- Error Packets Analysis
  - Assess the error packets sent and received for troubleshooting.
- Discarded Packets Overview
  - Review discarded packets to identify potential network issues.

Schema Diagram

Database Schema Diagram

Click here to edit subtitle



provisioning_jobs		
id	SERIAL	
credential_profile_id	INT	
ip	TEXT	NN

credential_profile		
id	SERIAL	
credential_profile_name	TEXT	NN
system_type	TEXT	NN
credentials	JSONB	NN
in_use_by	INT	

provision_data		
id	SERIAL	
job_id	INT	
data	JSONB	NN
polled_at	TIMESTAMP	

discovery_profiles		
id	SERIAL	
discovery_profile_name	TEXT	NN
credential_profile_id	INT	
ip	TEXT	NN
status	BOOLEAN	

Plugin Engine Flow (Go)

Understanding the Plugin Engine Workflow

1. **ZMQ Router Listens for Requests**

   The ZMQ Router acts as the entry point for incoming requests to the Plugin Engine, ensuring efficient communication.

2. **Router Sends to Dealer**

   Upon receiving a request, the Router forwards it to the Dealer for processing and distribution to Workers.

3. **Dealer Distributes to Workers**

   The Dealer is responsible for distributing the received requests to multiple Worker REP Sockets for parallel processing.

4. **Workers Fetch Data from SNMP**

   Workers communicate with SNMP devices to fetch the required data, ensuring accurate and timely responses.

5. **Responses Sent Back to Dealer**

   Once the Workers have fetched the data, they send the responses back to the Dealer for aggregation.

6. **Dealer Forwards to Router**

   The Dealer aggregates the responses and forwards them back to the Router for the final delivery.

7. **Router Sends to Vert.x**

   Finally, the Router sends the aggregated responses back to the Vert.x (ZMQ Messenger) for further processing.

Enhancing Query Performance with Indexing

Leveraging Indexing for Optimal Data Management

- **Importance of Indexing**

  Indexing is crucial for improving query performance, especially with large datasets.

- **SNMP Data Challenges**

  Large-scale SNMP data presents challenges in query efficiency without proper indexing.

- **Role of JSONB Indexing**

  JSONB indexing is particularly effective for querying complex data structures.

- **Performance Enhancement**

  Utilizing indexes can significantly reduce query response times and resource consumption.

- **Practical Applications**

  Implementing indexing strategies can lead to better data retrieval in real-time applications.

Justifying Our Technology Stack Choices

Exploring the Rationale Behind Our Choices

- **Why Go?**

  Utilizes efficient worker-based processing with goroutines, enhancing performance.

- **Faster Execution**

  Go outperforms thread-based execution, especially for network requests.

- **Why ZMQ?**

  ZeroMQ is fast, lightweight, and brokerless, perfect for efficient messaging.

- **Why PostgreSQL?**

  Supports JSONB for flexible SNMP data storage and high-performance indexing.

- **Event-Driven with Vert.x**

  Vert.x employs an event-driven architecture for handling multiple concurrent requests.

- **GitHub Copilot's Assistance**

  GitHub Copilot aids in debugging, suggesting SQL queries, and generating code snippets in Go & Vert.x.

- **Claude & DeepSeek Support**

  These tools help understand complex implementation issues and structure API endpoints efficiently.

- **Optimizing Development Processes**

  AI contributes to optimizing processes such as better indexing strategies for PostgreSQL.

- **Streamlining Communication Pipelines**

  AI recommends efficient structuring of ZMQ communication pipelines to enhance performance.

- **Handling Concurrent Requests**

  AI provides alternative approaches for efficiently managing concurrent SNMP requests.

Harnessing AI in Software Development

Exploring AI's Impact on Development Efficiency

Thank You

purchase
no purchase
Maximize Revenue with Sophisticated Lead Nurturing