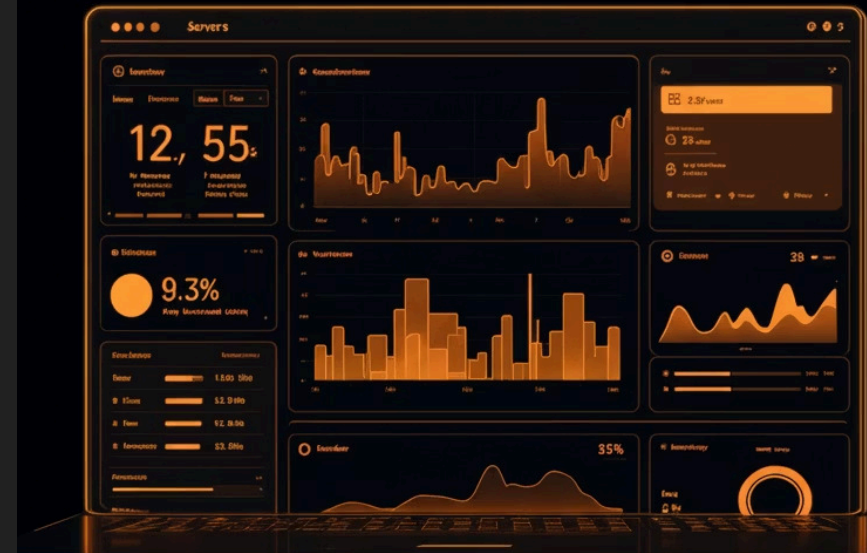
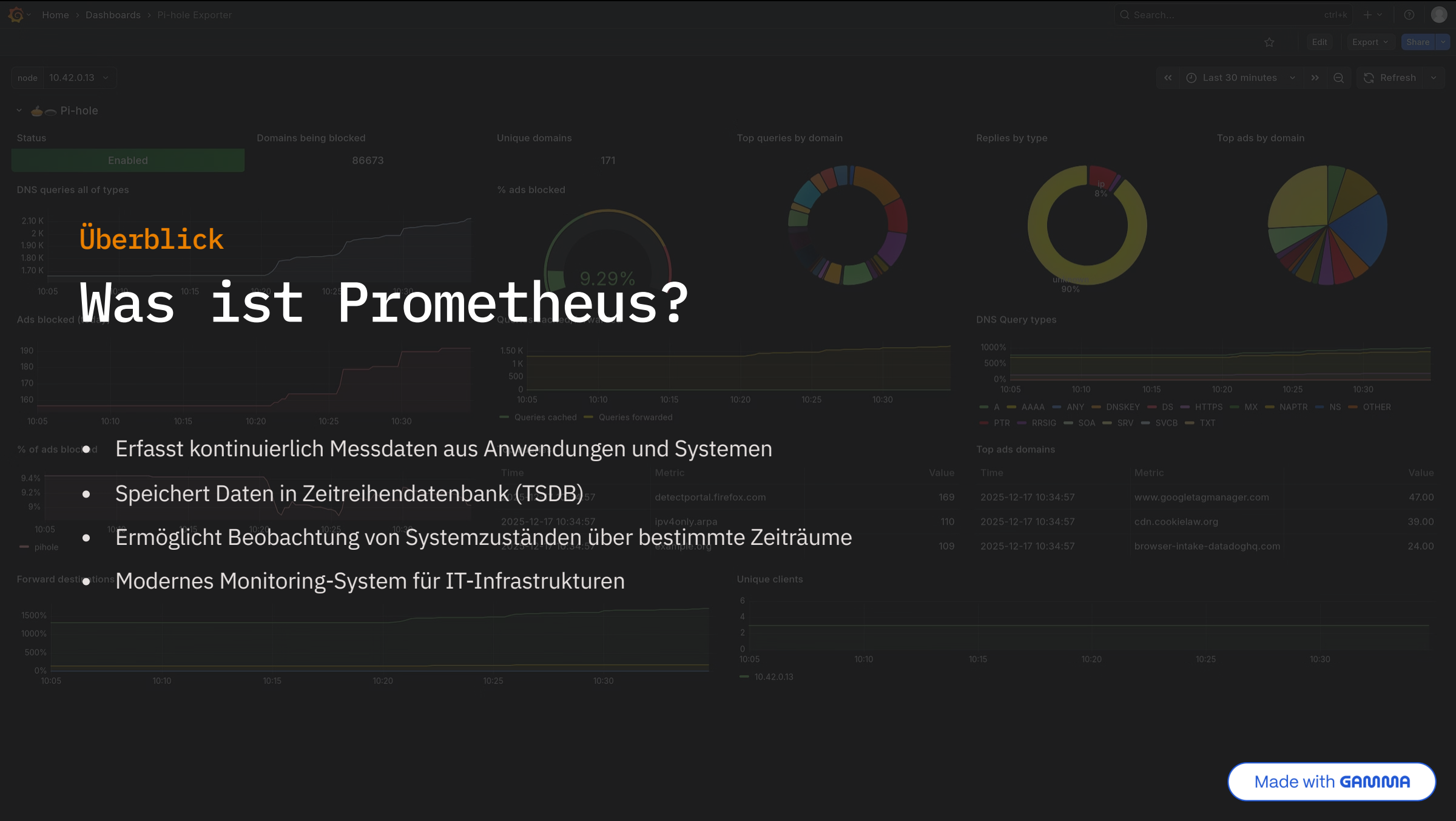


# Prometheus

Time-Series Datenbank für Monitoring-Systeme

Jonathan Al Kass & Johannes Olzem





# Systemarchitektur



## Time Series Database

Speicherung der Zeitreihen-Daten in optimierter TSDB



## HTTP-Server

Eigener Webserver mit Oberfläche für Abfragen und Konfiguration



## Retrieval Komponente

Sammelt Metrik-Daten und speichert sie in der TSDB

📄 Installation möglich auf dem System selbst oder in Containern (z.B. Docker). Keine spezifischen Systemanforderungen von Entwicklern angegeben.

# Prometheus vs. Relationale Datenbanken

## Zweck

**Prometheus:** Monitoring & Observability

**Relational:** Persistente Datenspeicherung

## Datenmodell

**Prometheus:** Zeitreihen, keine Joins

**Relational:** Tabellen mit Relationen & Joins

## Abfragesprache

**Prometheus:** PromQL

**Relational:** SQL

## Datenfluss

**Prometheus:** Pull-Modell (fragt ab)

**Relational:** Push-Modell (schreibt)

## Datenintegrität

**Prometheus:** Kleine Inkonsistenzen akzeptabel

**Relational:** ACID-Prinzip, hohe Integrität



# Praxisbeispiel: Heimserver-Monitoring



Client

**10.42.0.12** überwacht mit SNMP



Heimserver

**10.42.0.13** mit Prometheus, Grafana & PiHole



Containerbasiert

Services laufen mit Docker

Prometheus sammelt Metriken mit node-exporter (Host), pi-hole-exporter (DNS-Blocker) und snmp\_exporter (Netzwerkstatistiken). Verwaltung erfolgt über docker-compose.yml.

# API-Schnittstellen

## Querying API

Endpunkt: `/api/v1`

### Beispielabfrage:

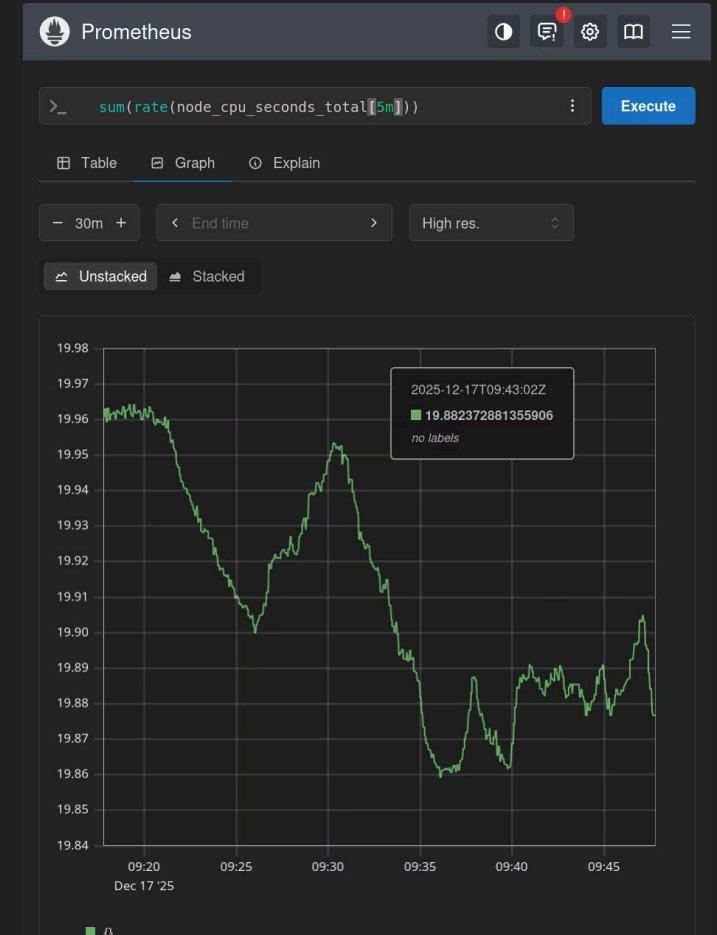
```
GET /api/v1/query?query=go_threads{instance="localhost:9090"}
```

### Antwort-Keys:

- `status`: "success" oder "error"
- `data`: Datentyp und abgefragte Daten

## Management API

- `GET /-/healthy`: Statusprüfung
- `GET /-/ready`: Bereitschaftsprüfung
- `PUT /-/reload`: Konfiguration neu laden
- `PUT /-/quit`: Ordnungsgemäßes Herunterfahren



# Administration & Monitoring

## Konfiguration

Erfolgt mit YAML-Dateien. Globale Optionen wie `scrape_interval` und `Targets` werden in `prometheus.yml` definiert.

## Self-Monitoring

Prometheus kann sich selbst überwachen. Metriken wie HTTP-Requests, Threads und Speicherverbrauch unter `/metrics` verfügbar.

## Backup & Recovery

Snapshots per API-Call: `POST /api/v1/admin/tsdb/snapshot`.

# Vor- und Nachteile

## Vorteile

- Schnelle Datenerfassung
- Geringe Latenz
- Komplexe Analysen mit PromQL
- Viele Exporter
- Skalierbar

## Nachteile

- Steile Lernkurve
- Komplexe Konfiguration
- Hoher Ressourcenverbrauch bei großen Datenmengen
- Keine Langzeitarchivierung
- Keine Authentifizierung





# Fazit

- Modernes Monitoring-System zur Erfassung von Messdaten
- Zeitreihen-Datenmodell ermöglicht übersichtliche Darstellung
- Komplexe Einrichtung und Abfragesprache
- Besonders geeignet für IT-Systemen mit ein oder mehreren Services
- Gute Integration mit vielen Tools wie Grafana
- Insgesamt sehr hilfreich für Systemüberwachung

# Persönliche Bewertung

“

„Prometheus ist ein sehr gutes Werkzeug zur Überwachung von IT-Infrastrukturen. Besonders positiv ist die automatische Erfassung der Daten und die vielen Erweiterungsmöglichkeiten durch Exporter.“

”

Der Einstieg ist zwar nicht ganz einfach, aber nach kurzer Einarbeitung ist das System gut nutzbar und sehr hilfreich.

Nach dieser Einführung werden wir Prometheus wahrscheinlich auch in unserem privaten Umfeld einrichten und verwenden.