



# Monitoring and Data-Driven Decisions with Prometheus and SUSE Manager

How monitoring of dynamic environments can be made easier with  
SUSE Manager, Prometheus and Grafana

João Cavalheiro and Johannes Renner

# Observability

**“...a measure of how well internal states of a system can be inferred from knowledge of its external outputs.”**

- **Metrics**
- **Logging**
- **Distributed tracing**

# Observability

**“...a measure of how well internal states of a system can be inferred from knowledge of its external outputs.”**

- **Metrics**
- **Logging**
- **Distributed tracing**

**If systems don't adequately externalize their state, monitoring will fall short.**

# Monitoring – Metrics

**Main data source for Alerting and Visualization:**

- **Starting point for troubleshooting**
  - “Something looks wrong on this dashboard”
- **Used as Service Level Indicators**
- **How available are we to the outside world?**
- **What are our customers experiencing?**

# Monitoring – Metrics

**Main data source for Alerting and Visualization:**

- **Starting point for troubleshooting**
  - “Something looks wrong on this dashboard”
- **Used as Service Level Indicators**
- **How available are we to the outside world?**
- **What are our customers experiencing?**

**Good metrics help to eliminate hypotheses before you investigate them.**

# Prometheus and Grafana

# About Prometheus

- Originally built at SoundCloud
- Has its own time-series database
- Data collection via pull model over HTTP
- Targets are set via static configuration or service discovery
- Metrics have a name, a set of labels, a timestamp and a value

Diagram illustrating the structure of a Prometheus metric line:

```
...  
http_requests_total{status="200",method="GET"} @1434317560938 94355
```

The diagram uses brackets to identify the components of the metric line:

- Metric name:** `http_requests_total`
- Labels:** `{status="200",method="GET"}`
- Timestamp:** `@1434317560938`
- Sample Value:** `94355`

# Exposing Metrics

- Each application/system we want to monitor must expose metrics
- Instrumentation vs Exporters
- When the metrics endpoint is embedded into an existing application, it's referred to as instrumentation. When it is part of a stand-alone process, we call it an Exporter.
- Extensive list of Prometheus exporters
  - <https://prometheus.io/docs/instrumenting/exporters/>
- Node exporter is one of the most widely used
- Easy to build your own exporters
- You can monitor almost anything



# Blackbox Monitoring

- Enables probing of endpoints over HTTP(s), DNS, TCP and ICMP
- Blackbox exporter is a helper daemon that can accept commands from Prometheus
- Preferred method to monitor services from the outside
- Can easily measure service latency
- Transparent integration with Prometheus
- The blackbox daemon is yet another scrape target
- Can be useful when you have no access to client internals
- But should not be used as a replacement for proper instrumentation

# Querying Metrics

- Prometheus has its own query language – PromQL
- PromQL is a functional expression language
- Easily filter multi-dimensional time-series
- Example: HTTP internal server errors per second... an hour ago
- `rate(api_http_requests_total{status=500}[5m] offset 1h)`
- Regex matching
- `up{instance=~"web-server-.*"} == 0`
- Used in all interactions with Prometheus (visualization, alerts)

# Alerts

- Prometheus has its own alerting system – Alert Manager
- Takes care of deduplicating, grouping, and routing
- Alerting rules are written in PromQL
- Supports HA setups
- Integration with email, PagerDuty and OpsGenie
- HTTP API and CLI tool: amtool
- Can be “plugged” into your existing scripts

# Grafana

- **Used to query and visualize metrics**
- **Works with Prometheus, but not only...**
- **Grafana supports multiple backends**
- **It is possible to combine data from different sources in the same dashboard**
- **Fully customizable**
- **Each panel has a wide variety of styling and formatting options**
- **Supports templates**
- **Collection of add-ons and pre-built dashboards**

# Putting the Pieces Together

# How to Get Started

- Which Prometheus components will I need?
- How can I configure my clients to expose metrics?
- What can be automated?
- How do I get started with building graphics?
- And, what should I monitor?

# Building Blocks

- **Packaged Prometheus, Grafana and exporters**
- **Integration with SUSE Manager**
- **Monitoring best practices**
- **Future vision**

# SUSE Manager



# What Is SUSE Manager?

- **Systems and infrastructure management (on-prem or cloud)**
- **Based on Spacewalk in the past, now forked into Uyuni:**
  - **<https://www.uyuni-project.org/>**
- **Backend based on Salt Open**
- **Management of diverse client architectures and OSs**
- **Scalability is an ongoing challenge**

# SUSE Manager Meets Monitoring

## Set up and automate Prometheus monitoring with SUSE Manager

- **Packages will be provided via official channels:**
  - **Main packages (Prometheus, Grafana) from SUSE Manager channels**
  - **Specific metrics exporters alongside the respective applications**
- **Enable exporters on managed clients using Salt Formulas**
- **Group systems to define data exporter templates**
- **Integration with Prometheus service discovery**

# SUSE Manager Meets Monitoring

## Self-monitoring of SUSE Manager Server and Proxy

- Easily enable monitoring of SUSE Manager Server
- Pre-installed exporters can expose metrics about:
  - Hardware
  - Java virtual machines
  - Apache and PostgreSQL
  - SUSE Manager's internals
- SUSE Manager Proxy to be monitored via squid exporter

# SUSE Manager Meets Monitoring

## Coming next...

- **Provisioning of Prometheus and Grafana servers**
- **Integration with existing Prometheus setups**
- **Closing the monitoring loop**
  - **Automatically enable monitoring during client onboarding**
  - **Alert templates by system group and integration with AlertManager**
  - **Fully automated integration with Prometheus service discovery**
  - **Optional encryption and authentication on the metrics endpoints**
  - **Automatic provisioning of Grafana dashboard templates**

# Researching

**Grafana Loki: Prometheus-inspired logging**

**Common monitoring building blocks for SUSE products**

**Grafana add-on development**

# Live Demo

# Live Demo

- **SUSE Manager self-monitoring**
- **Install and enable exporters on managed clients**
- **Integration with Prometheus service discovery**
- **Grafana dashboards**
- **Alerts**



We adapt. You succeed.



## **Unpublished Work of SUSE LLC. All Rights Reserved.**

This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE LLC. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

## **General Disclaimer**

This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of SUSE LLC. in the United States and other countries. All third-party trademarks are the property of their respective owners.