Prometheus

Если ты не знаешь этого парня -
не пиши мне больше

Что за зверь?

Prometheus - это система мониторинга и оповещения, которая используется для сбора, обработки, хранения и предоставления информации о состоянии различных компонентов IT-систем, таких как серверы, приложения, базы данных и сетевое оборудование.

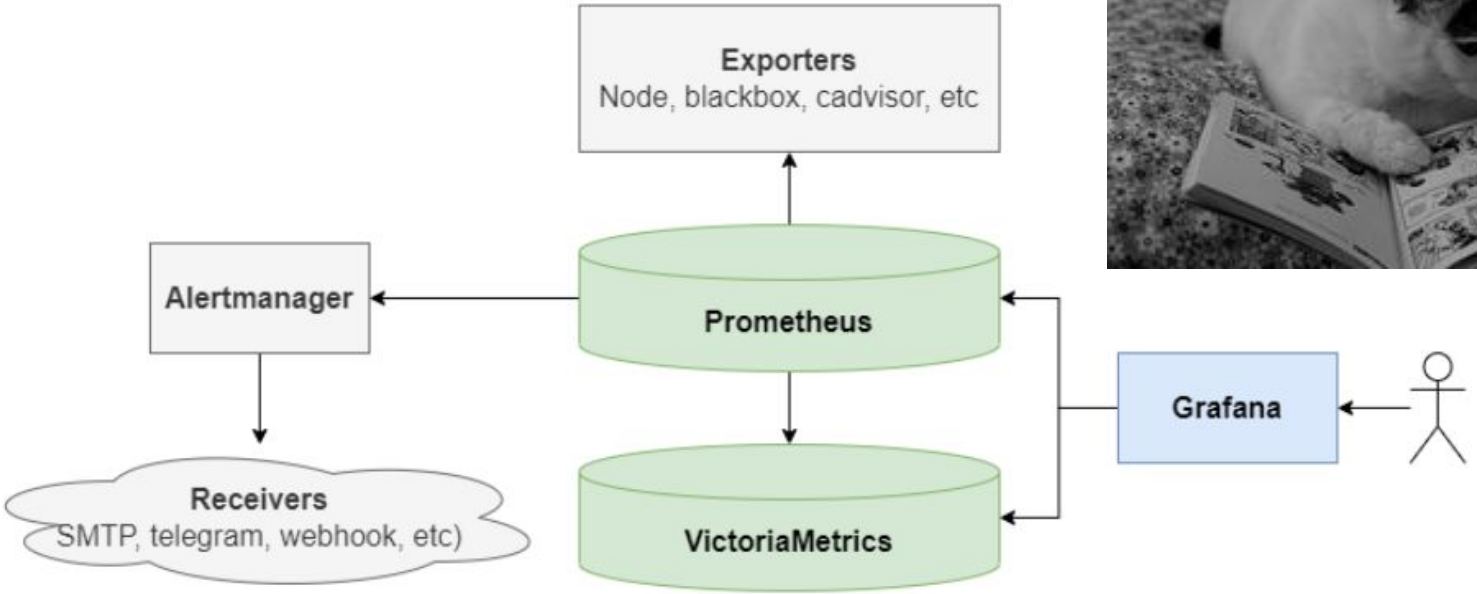
Типовые процессы автоматизации

- ***Сбор метрик***: Prometheus собирает метрики, которые позволяют оценить состояние IT-системы, такие как использование CPU, памяти, дискового пространства и сетевой активности. Эти метрики формируются на основе данных, полученных из экспортеров, которые устанавливаются на мониторируемых устройствах.
- ***Обработка метрик***: Prometheus обрабатывает собранные метрики, выполняя агрегацию, фильтрацию и преобразование данных, чтобы представить их в нужном виде.
- ***Хранение метрик***: Prometheus хранит метрики в своей базе данных, которая позволяет анализировать их на протяжении определенного периода времени. Хранение метрик может быть настроено с помощью параметров хранения данных, таких как время хранения и частота сбора.
- ***Передача метрик***: Prometheus может передавать метрики другим системам мониторинга, используя различные протоколы, такие как HTTP, Alertmanager и другие.
- ***Предоставление информации***: Prometheus предоставляет информацию о состоянии IT-системы с помощью графических интерфейсов и API, что позволяет пользователям быстро и эффективно анализировать и исправлять проблемы в IT-системе.

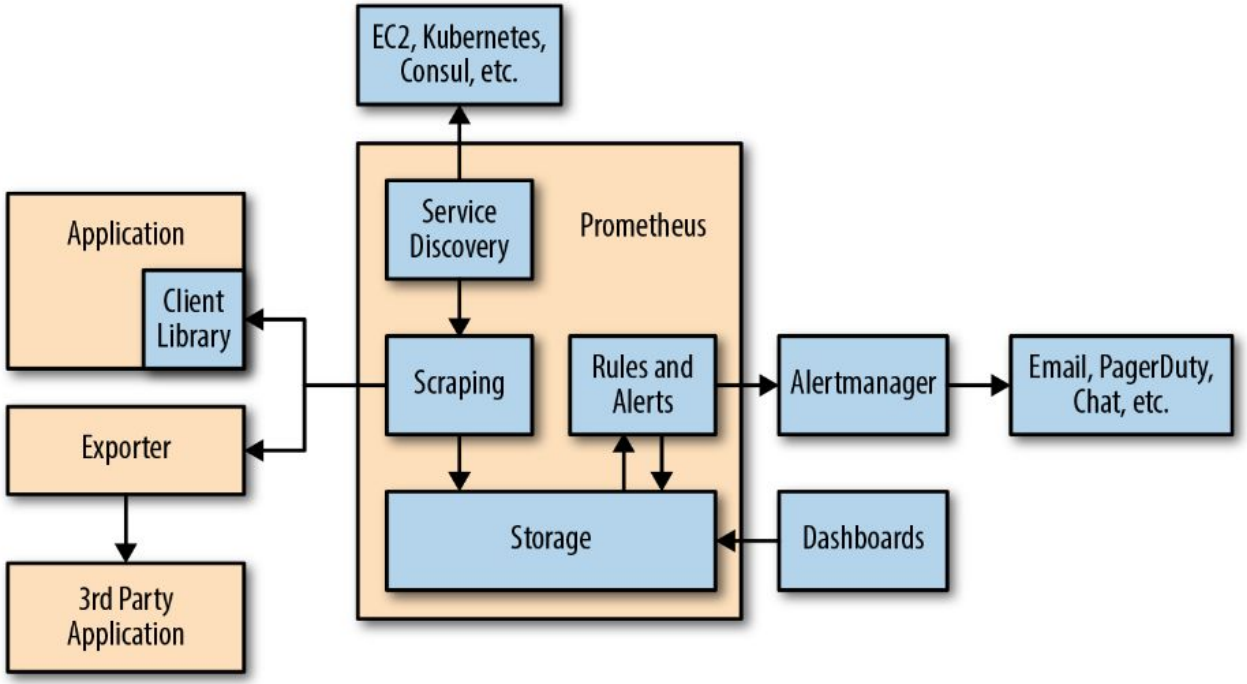
Нефункциональные требования

- Производительность: Prometheus должен иметь высокую производительность для обработки больших объемов данных.
- Масштабируемость: Prometheus должен масштабироваться горизонтально для обеспечения обработки метрик в больших инфраструктурах.
- Надежность: Prometheus должен быть надежным и устойчивым к сбоям, чтобы гарантировать непрерывность мониторинга инфраструктуры.
- Безопасность: Prometheus должен быть безопасным и защищенным от несанкционированного доступа и атак.
- Удобство использования: Prometheus должен быть легким в установке и настройке, иметь понятный интерфейс и предоставлять достаточно документации и руководств для удобства использования.

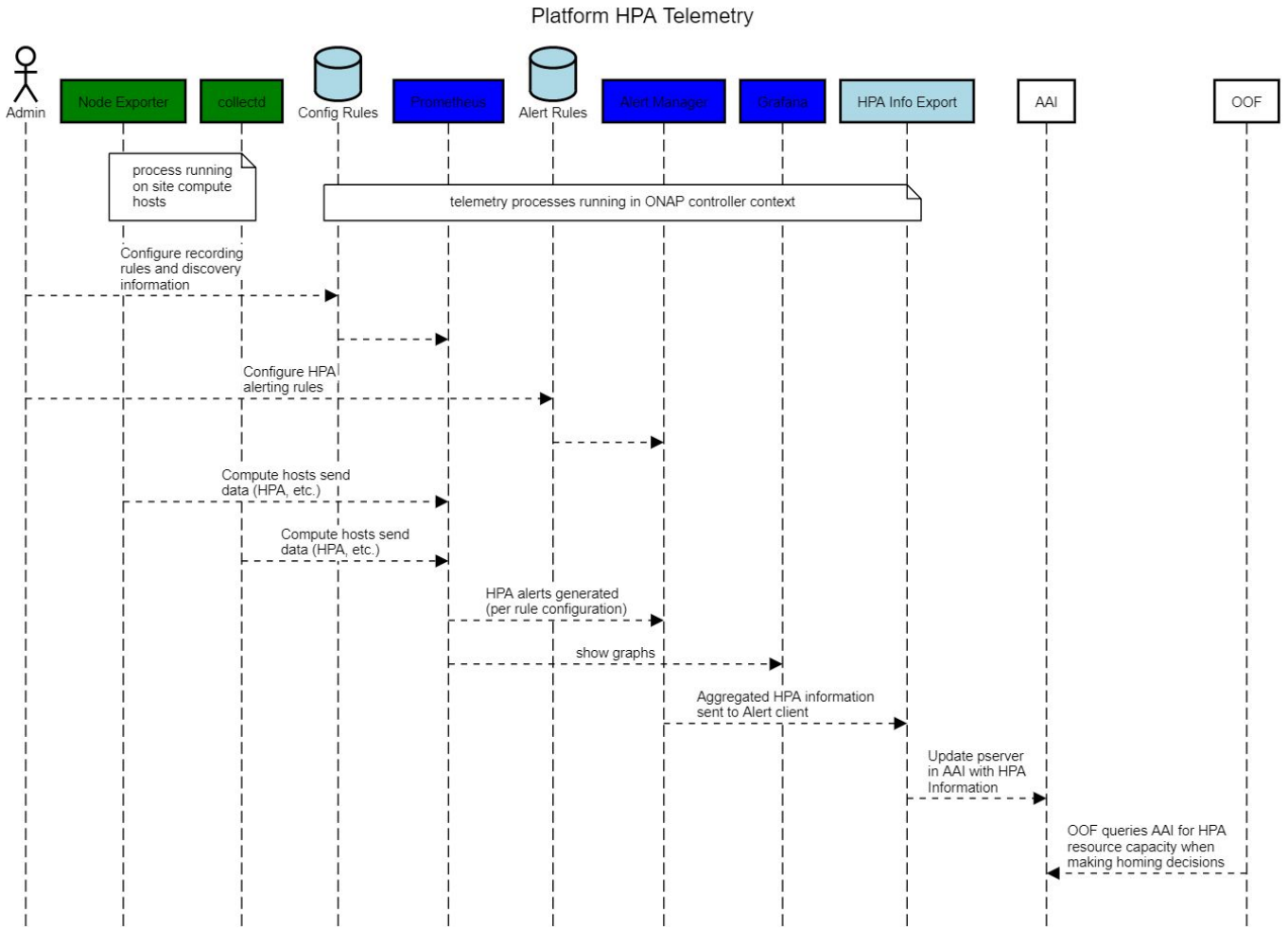
Много букав, можно картинки



Картинка поподробнее



Картинка покрасивше,что даже не влезла на слайд



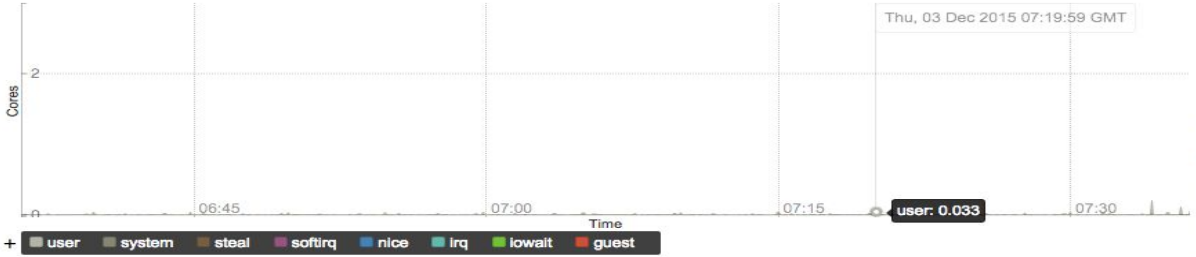
Голый Prometheus

Prometheus Alerts PagerDuty

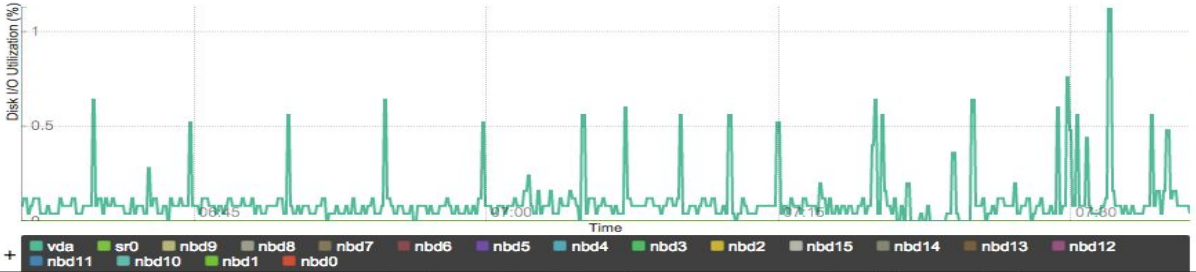
Overview
Node
localhost:9100
CPU
Disk
Prometheus

Node Overview - localhost:9100

CPU Usage



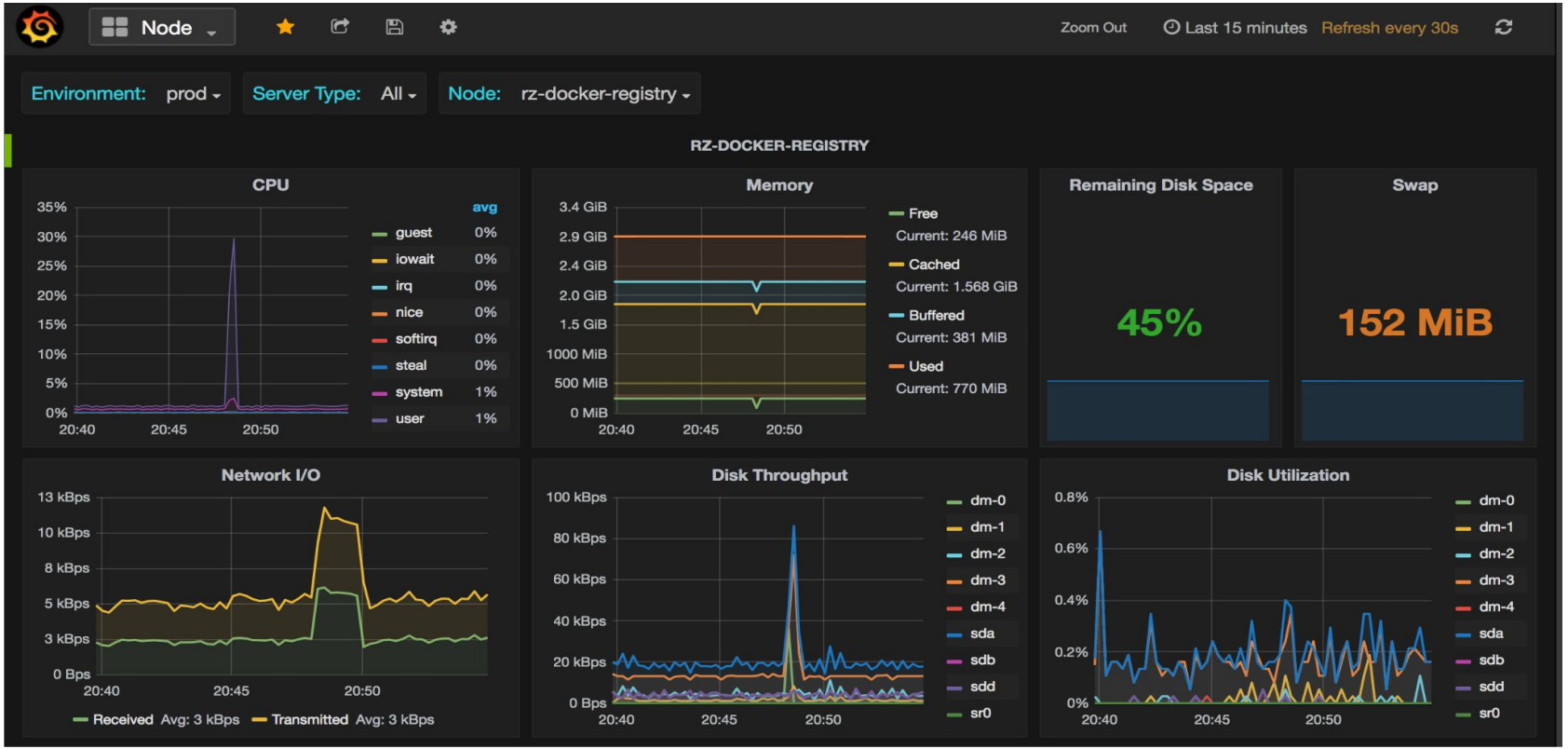
Disk I/O Utilization



Overview	
User CPU	1.8%
System CPU	0.5%
Memory Total	1.955GiB
Memory Free	100.1MiB
Network	
eth0 Received	439.9B/s
eth0 Transmitted	5.444kB/s
lxcbr0 Received	0B/s
lxcbr0 Transmitted	0B/s
vethF967BW Received	0B/s
vethF967BW Transmitted	0B/s
Disks	
nbd0 Utilization	0.0%
nbd1 Utilization	0.0%
nbd10 Utilization	0.0%
nbd11 Utilization	0.0%
nbd12 Utilization	0.0%
nbd13 Utilization	0.0%
nbd14 Utilization	0.0%

1h Until (Off)

Интеграция с Grafana



Информационная архитектура

Метрики - основные информационные объекты в Prometheus, являются числовыми значениями, представляющими состояние каких-либо систем или приложений в конкретный момент времени. Метрики в Prometheus хранятся в виде **временных рядов**, каждый из которых содержит набор значений метрики, связанных с конкретными временными метками.
Оповещение (Alert) - выражения PromQL, которые позволяют задавать условия для метрик и временных рядов. Каждый алерт имеет уникальное имя, описание и набор меток.
Конфигурационные файлы - файлы конфигурации Prometheus определяют, какие метрики будут собираться, какие endpoint'ы следует просматривать, и какие правила следует применять к данным

Архитектура данных

Временные ряды (time series) - последовательности временных меток (timestamp) и соответствующих значения метрики (value)
Datasets - несколько временных рядов, относящихся к одному и тому же приложению или сервису

Функциональная архитектура

Сбор, обработка и хранение метрик. Реализовано с помощью **Prometheus Server**

Управление и отправка оповещений, когда происходят события, которые нужно мониторить. Реализовано с помощью **Alertmanager**

Функционал для отправки приложениями метрик непосредственно в Prometheus. Реализовано с помощью **Pushgateway**

Визуализации данных, собранных Prometheus. Реализовано с помощью **Grafana**

Программная архитектура

- **Prometheus Server** - ядро системы, отвечающее за сбор и хранение метрик, агрегацию их значений и предоставление данных для пользовательских запросов. Сервер написан на языке Go и представляет собой набор бинарных исполняемых файлов.
- **Exporters** - компоненты, предназначенные для сбора метрик с различных систем. Prometheus поддерживает множество экспортеров, включая стандартные экспортеры для сбора метрик из системы мониторинга Nagios, баз данных MySQL и PostgreSQL, а также экспортеры для мониторинга системных ресурсов и приложений, например, Node Exporter, Apache Exporter и др.
- **Alertmanager** - компонент, предназначенный для управления оповещениями о событиях, происходящих в системе мониторинга. Alertmanager отвечает за группировку, фильтрацию и отправку оповещений на различные каналы связи, включая email, Slack, PagerDuty и др.
- **Grafana** – компонент, который обеспечивает визуализацию метрик, собранных Prometheus. Он позволяет пользователям создавать графики и дашборды на основе собранных метрик, а также настраивать оповещения на основе этих метрик.
- **Push Gateway** – компонент, который позволяет отправлять метрики в Prometheus Server без необходимости настройки экспортера. Это может быть полезно для приложений, которые не поддерживают экспорт метрик, но хотели бы отправлять свои метрики в Prometheus.

Технологии, используемые при разработке компонент

- Язык программирования Go для разработки большинства компонентов Prometheus.
- gRPC для обмена сообщениями между компонентами и поддержки удаленных вызовов процедур.
- LevelDB и RocksDB для хранения данных временных рядов метрик в локальном хранилище.
- TSDB (Time Series Database) для эффективного хранения, обработки и агрегации временных рядов метрик.
- Prometheus Query Language (PromQL) для запросов и агрегации метрик в реальном времени.

Как обеспечивается целостность данных

- Механизм записи данных - данные записываются только после того, как их точность была подтверждена. Если данные неверны или не могут быть подтверждены, они не будут записаны.
- Механизм хранения данных - Prometheus использует хранилище данных, которое обеспечивает сохранность данных, их доступность и возможность быстрого поиска и извлечения. Данные хранятся в формате, который обеспечивает непротиворечивость и уникальность идентификаторов.
- Механизм обработки данных - Prometheus использует специальный язык запросов, который обеспечивает непротиворечивость и уникальность идентификаторов при выполнении операций с данными. Этот язык запросов также позволяет извлекать данные из хранилища данных с высокой производительностью
- Механизм резервного копирования данных - Prometheus позволяет создавать резервные копии данных, что обеспечивает сохранность данных в случае сбоев или других проблем.
- Механизм синхронизации данных - Prometheus позволяет синхронизировать данные между несколькими экземплярами, что обеспечивает их непротиворечивость и уникальность идентификаторов.