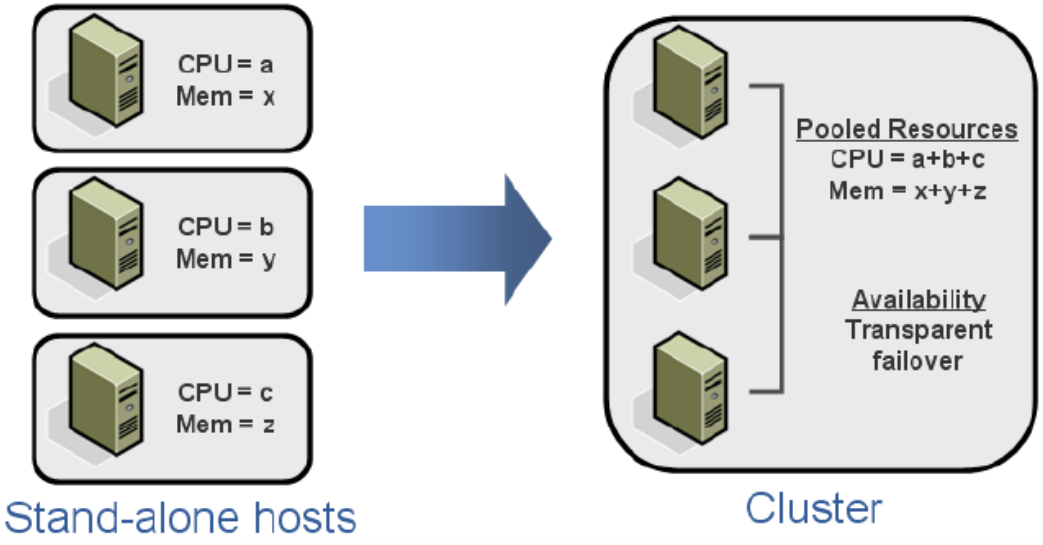
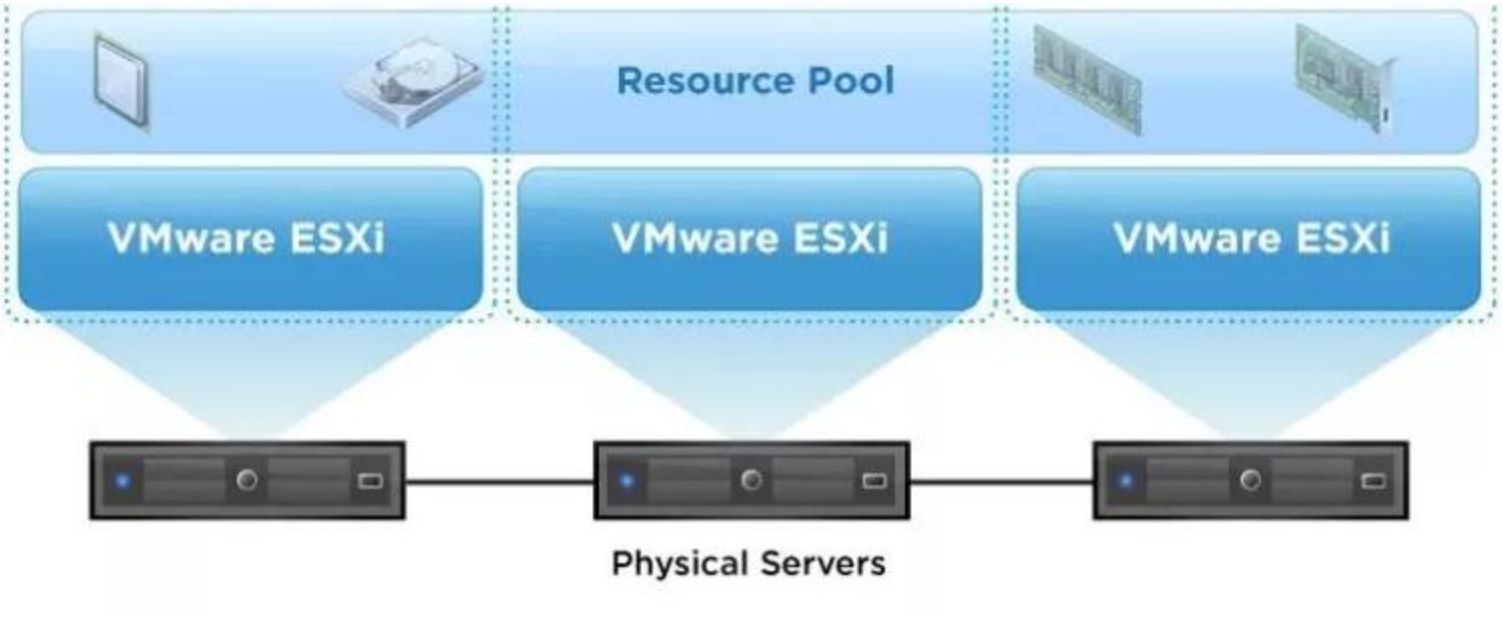
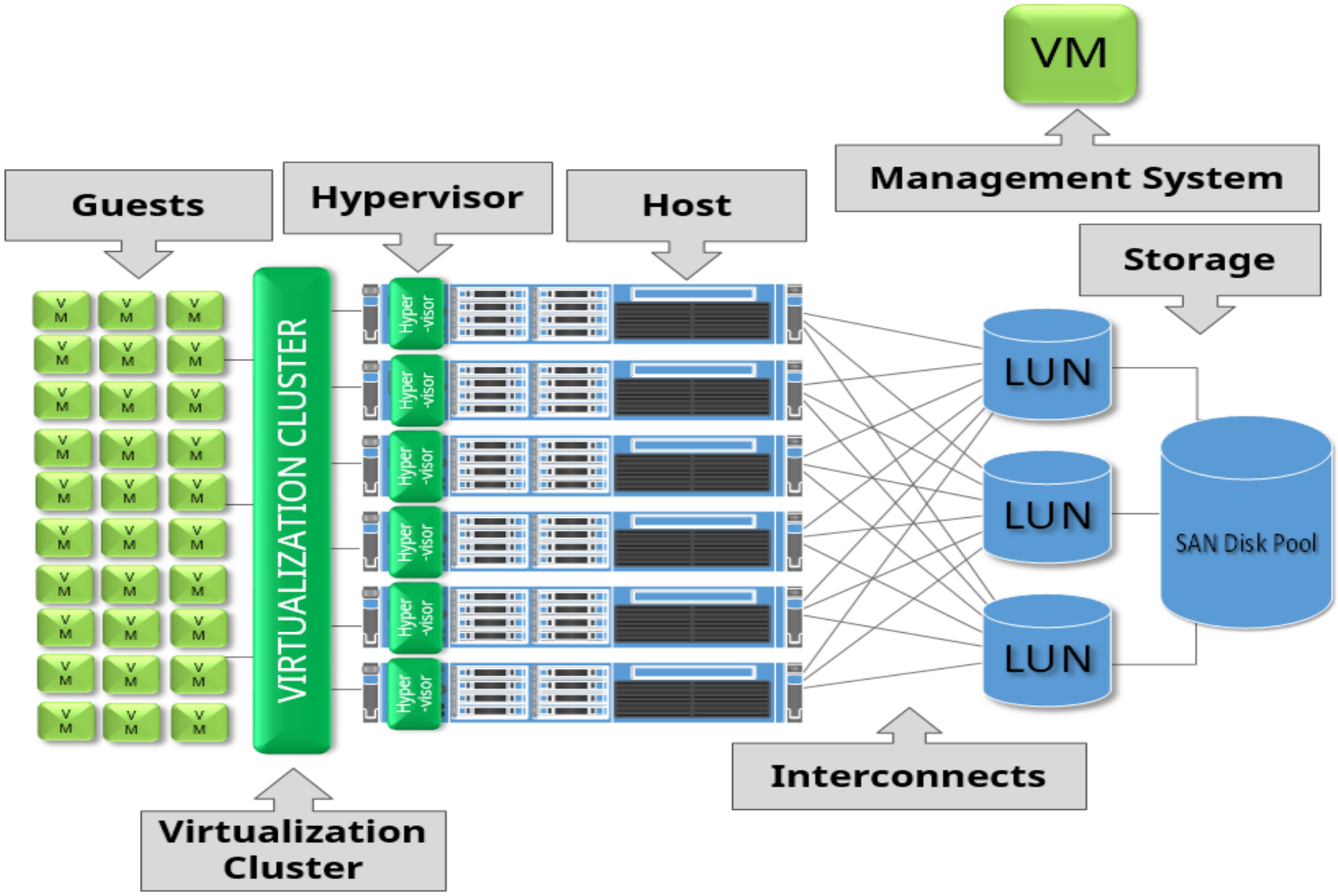
Тема 10
Настройка, мониторинг и оценка эффективности виртуальной инфраструктуры

Формирование списка ресурсов





Настройка инфраструктуры



Модель «Инфраструктура как код (IaC)», которую иногда называют «программируемой инфраструктурой», — это модель, по которой процесс настройки инфраструктуры аналогичен процессу программирования ПО. По сути, она положила начало устранению границ между написанием приложений и созданием сред для этих приложений. Приложения могут содержать скрипты, которые создают свои собственные виртуальные машины и управляют ими. Это основа облачных вычислений и неотъемлемая часть DevOps.

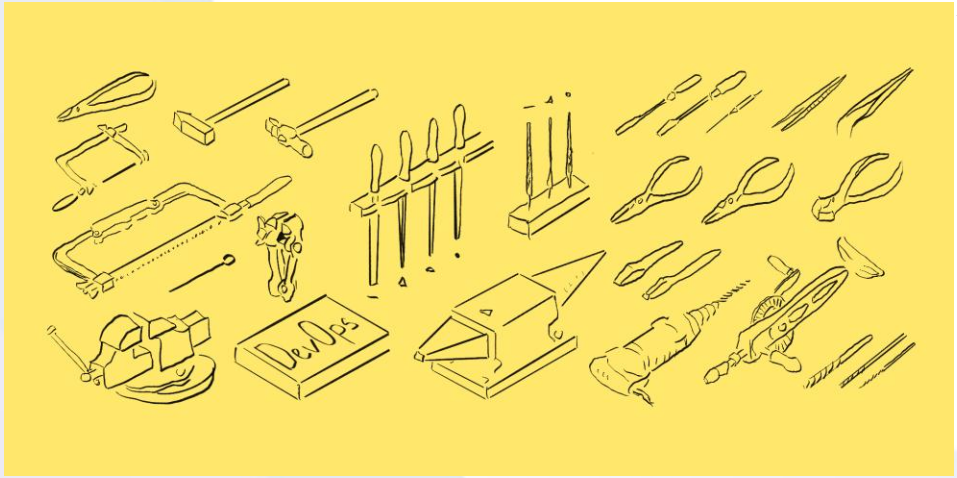
Инфраструктура как код позволяет управлять виртуальными машинами на программном уровне. Это исключает необходимость ручной настройки и обновлений для отдельных компонентов оборудования. Инфраструктура становится чрезвычайно "эластичной", то есть воспроизводимой и масштабируемой. Одни оператор может выполнять развертывание и управление как одной, так и 1000 машинами, используя один и тот же набор кода. Среди гарантированных преимуществ инфраструктуры как кода — скорость, экономичность и уменьшение риска.

Модель IaC послужила истоком развития концепции DevOps. Все более тонкая грань между кодом, обеспечивающим работу приложений, и кодом, настраивающим инфраструктуру, означает, что у разработчиков и специалистов по эксплуатации становится все больше общих обязанностей и рабочих задач.

12 инструментов DevOps-инженера для мониторинга архитектуры. Общая информация

- Управление конфигурациями, инструменты метрики, оповещений, визуализации. Преимущества инструментов мониторинга архитектуры DevOps, ссылки на сами инструменты и публикации о них.

Почему важен мониторинг? Команды разработчиков кодят быстрее, продукт нужно тестировать оперативнее, делать релизы скорее и не проседать в уровне качества. Но частые изменения кода влекут лавинообразный рост ошибок. Для получения полной картины необходимо иметь проактивный мониторинг ПО.



Инструменты для мониторинга

- Хорошая мониторинговая платформа позволит отследить производительность всей системы и приложений локально, в облаке или в контейнерных средах. Эффективный набор инструментов повысит производительность и поможет сократить или даже устранить время простоя. Вы сможете планировать обновления и новые проекты, лучше распределять время и другие ресурсы, а самое важное – обнаруживать и решать проблемы до того, как их увидит пользователь.

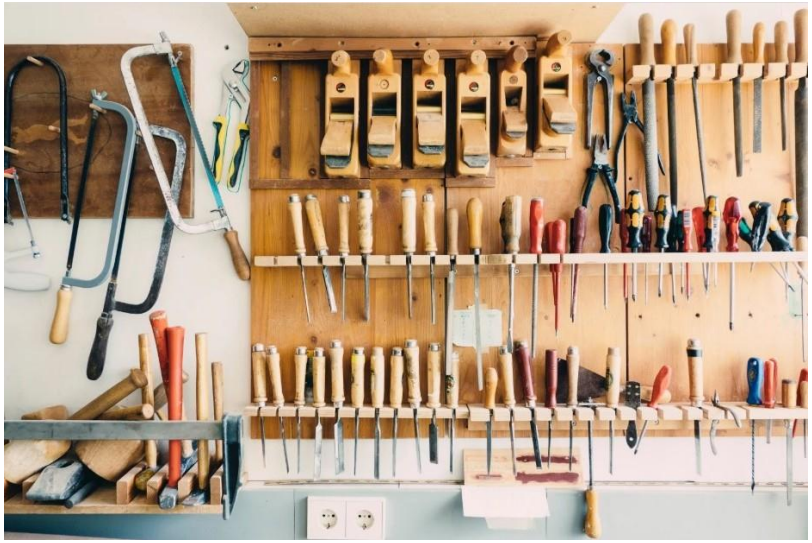
- Sensu – гибкое и масштабируемое решение для проверки работоспособности телеметрии и служб. Используется для мониторинга серверов, контейнеров, приложений, функций и подключенных устройств.
- Prometheus – инструмент со встроенной базой данных, использующий pull-метод для сбора информации.
- Nagios – старый надежный инструмент для мониторинга компьютерных систем и сетей с открытым исходным кодом, задавший моду на мониторинг у целого поколения инженеров.

Управление конфигурациями

- Средства управления конфигурациями позволяют автоматизировать подготовку и развертывание систем, принудительно применять требуемые конфигурации и устранять баги. Моделируя инфраструктуру кода, вы можете применять контроль версий, автоматизированное тестирование и непрерывную доставку в приложения. Автоматизация рутинной, подверженной ошибкам работы приводит к повышению скорости, предсказуемости и масштабируемости. Также обеспечиваются стандартизированные конфигураций в test, dev и prod средах.

Инструменты

- Ansible – инструмент на Python. Применяется без агента и использует императивный подход.
- Chef – инструмент на Ruby, императивный подход управления конфигурациями.
- Puppet – декларативный подход, использует специфичный «доменный» язык и архитектуру агент/мастер.



Инструменты оповещения

- Система оповещения должна быть достаточно чувствительной, чтобы реагировать на перебои, но не настолько чувствительной, чтобы спамить вас сообщениями о мелких падениях, которые не заметят пользователи. Инструменты оповещения помогают заложить основу для политик безопасности, а также настроить: кого уведомлять, как мониторить и как устанавливать приоритеты восстановления.

Инструменты

- PagerDuty – ситуационная платформа управления с дополнениями для аналитики, анализа событий и автоматического реагирования на инциденты.
- ServiceNow – использует автоматизированные рабочие процессы для ITSM (IT Service Management), обслуживания клиентов и бизнес-процессов.
- Slack – можно повесить оповещения на ту же платформу, которую вы используете для групповых чатов и совместной работы.

Инструменты

- InfluxDB – база данных для работы с временными рядами, подходящая для долгосрочного хранения данных.
- Splunk – использует модель базы данных, как в поисковых системах.

Визуализация

- Инструменты визуализации могут рассматриваться в качестве основы для набора инструментов, используемых при мониторинге. Вы можете объединять данные, сортировать и визуализировать их в различных панельках. Тонкая настройка позволит команде создавать и совместно использовать собственные наработки в области мониторинга.

Инструменты

- Grafana – может использоваться поверх различных хранилищ данных, включая Graphite, InfluxDB и Elasticsearch.

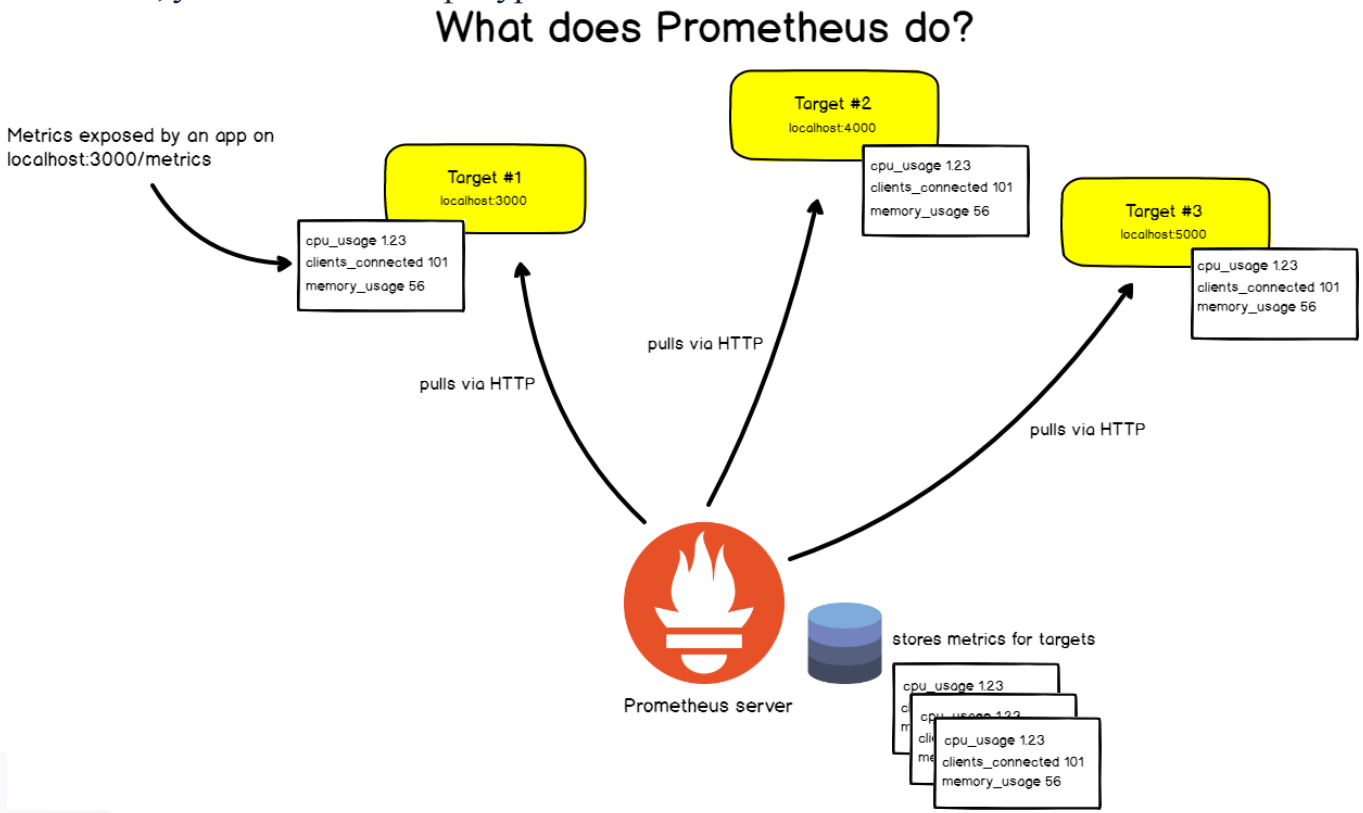
Инструмент для мониторинга Prometheus

- Prometheus был создан на SoundCloud в 2012 году и с тех пор стал стандартом для мониторинга систем. У него полностью открытый исходный код, он предоставляет десятки разных экспортеров, с помощью которых можно за считанные минуты настроить мониторинг всей инфраструктуры.

- Prometheus обладает очевидной ценностью и уже используется новаторами в отрасли, вроде DigitalOcean или Docker, как часть системы полного мониторинга.

Что такое Prometheus

- Prometheus – это база данных временных рядов, но не обычная. К нему можно присоединить целую экосистему инструментов, чтобы расширить функционал. Prometheus мониторит самые разные системы: серверы, базы данных, отдельные виртуальные машины, да почти что угодно. Для этого Prometheus периодически скрейпит свои целевые объекты. Prometheus извлекает метрики через HTTP-вызовы к определенным конечным точкам, указанным в конфигурации Prometheus.



- Возьмем, например, веб-приложение, расположенное по адресу http://localhost:3000. Приложение передает метрики в текстовом формате на некоторый URL. Допустим, http://localhost:3000/metrics.

- По этому адресу Prometheus с определенными интервалами **извлекает данные** из целевого объекта.

Как работает Prometheus

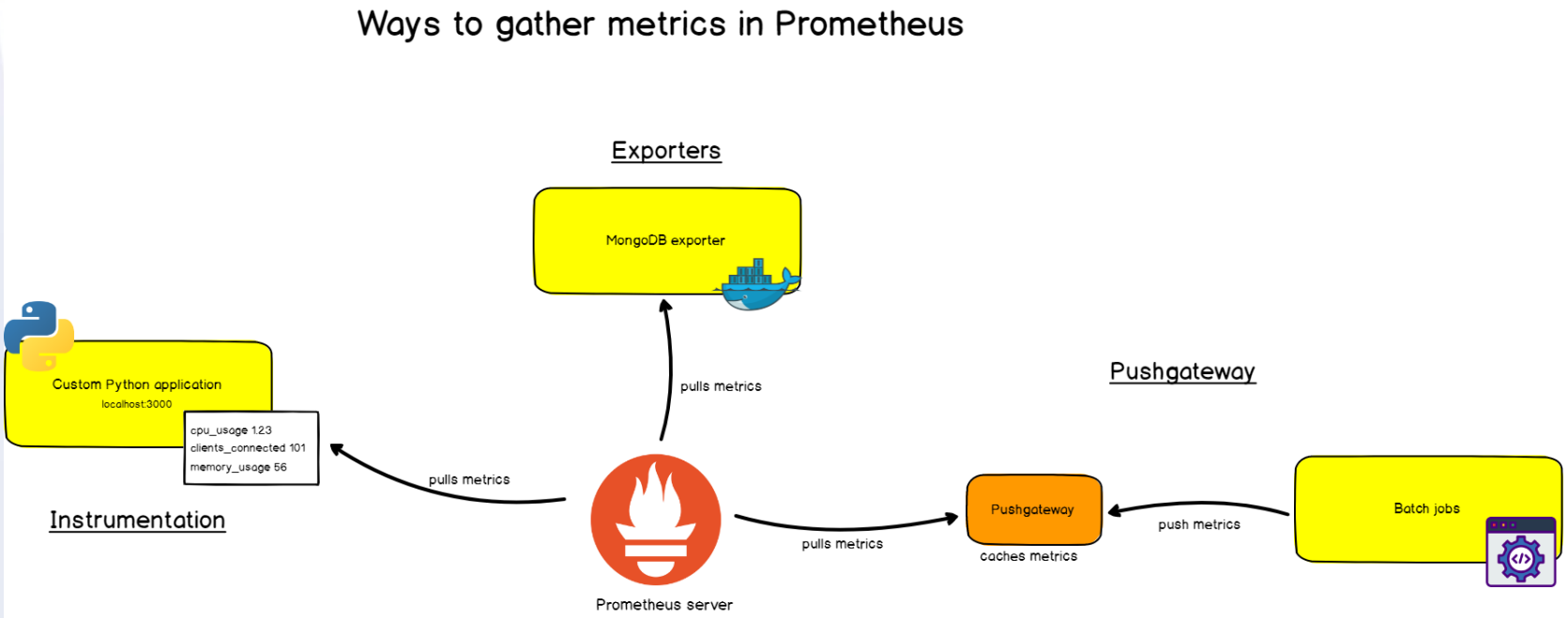
- Во-первых, вам нужно, чтобы он извлекал метрики из ваших систем. Тут есть разные способы:

- **Инструментирование** приложения, то есть ваше приложение будет предоставлять совместимые с Prometheus метрики по заданному URL. Prometheus определит его как целевой объект и будет скрейпить с указанным интервалом.

- Использование **готовых экспортеров**. В Prometheus есть целая коллекция экспортеров для существующих технологий. Например, готовые экспортеры для мониторинга машин Linux (Node Exporter), для распространенных баз данных (SQL Exporter или MongoDB Exporter) и даже для балансировщиков нагрузки HTTP (например, HAProxy Exporter).

- Использование **Pushgateway**. Иногда приложения или задания не предоставляют метрики напрямую. Они могут быть не предназначены для этого (например, пакетные задания) или вы сами решили не предоставлять метрики напрямую через приложение.

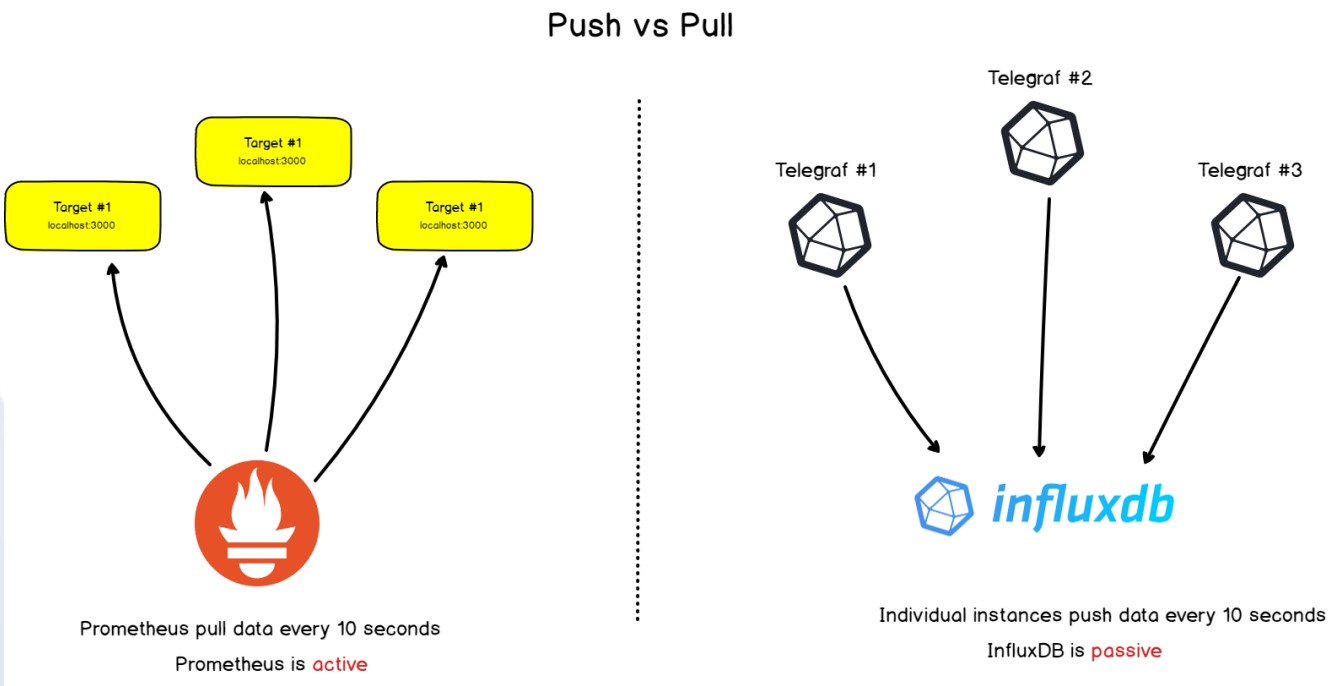
- Как вы уже поняли, Prometheus сам собирает данные (исключая редкие случаи, когда мы используем Pushgateway).



Сбор vs. отправка

- У Prometheus есть заметное отличие от других баз данных временных рядов: он активно сканирует целевые объекты, чтобы получить у них метрики.

- InfluxDB, например, работает иначе: вы сами напрямую отправляете ему данные.



- Оба подхода имеют свои плюсы и минусы. На основе доступной документации составлен список причин, по которым создатели Prometheus выбрали такую архитектуру:

- **Централизованный контроль.** Если Prometheus отправляет запросы целевым объектам, всю настройку мы выполняем на стороне Prometheus, а не отдельных систем.

- Prometheus сам решает, где и как часто проводить скрейпинг. Если объекты сами отправляют данные, есть риск, что таких данных будет слишком много, и на сервере произойдет сбой. Когда система собирает данные, можно контролировать частоту сбора и создавать несколько конфигураций скрейпинга, чтобы выбирать разную частоту для разных объектов.

- Prometheus хранит **агрегированные метрики.**

- Prometheus не основан на событиях и этим сильно отличается от других баз данных временных рядов. Он не перехватывает отдельные события с привязкой ко времени (например, перебои с сервисом), а собирает предварительно агрегированные метрики о ваших сервисах.

- Если конкретно, веб-сервис не отправляет сообщение об ошибке 404 и сообщение с причиной ошибки. Отправляется сообщение о факте, что сервис получил сообщение об ошибке 404 за последние пять минут.

- Это главное различие между базами данных временных рядов, которые собирают агрегированные метрики, и теми, что собирают необработанные метрики.

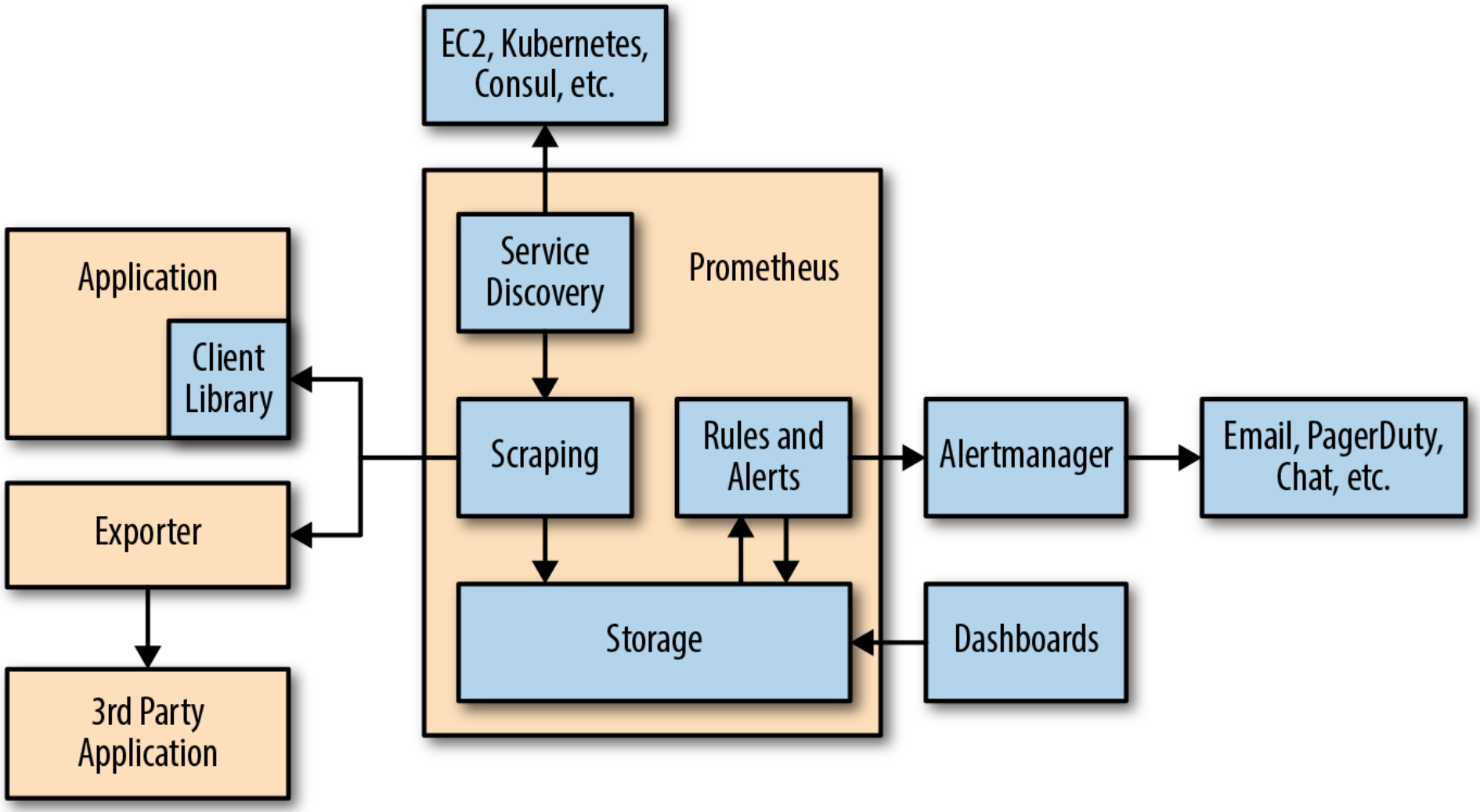
Prometheus

Развитая экосистема Prometheus

- По сути Prometheus – база данных временных рядов. Но при работе с такими базами данных часто нужно **визуализировать** данные, **анализировать** их и настраивать по ним **оповещения.**

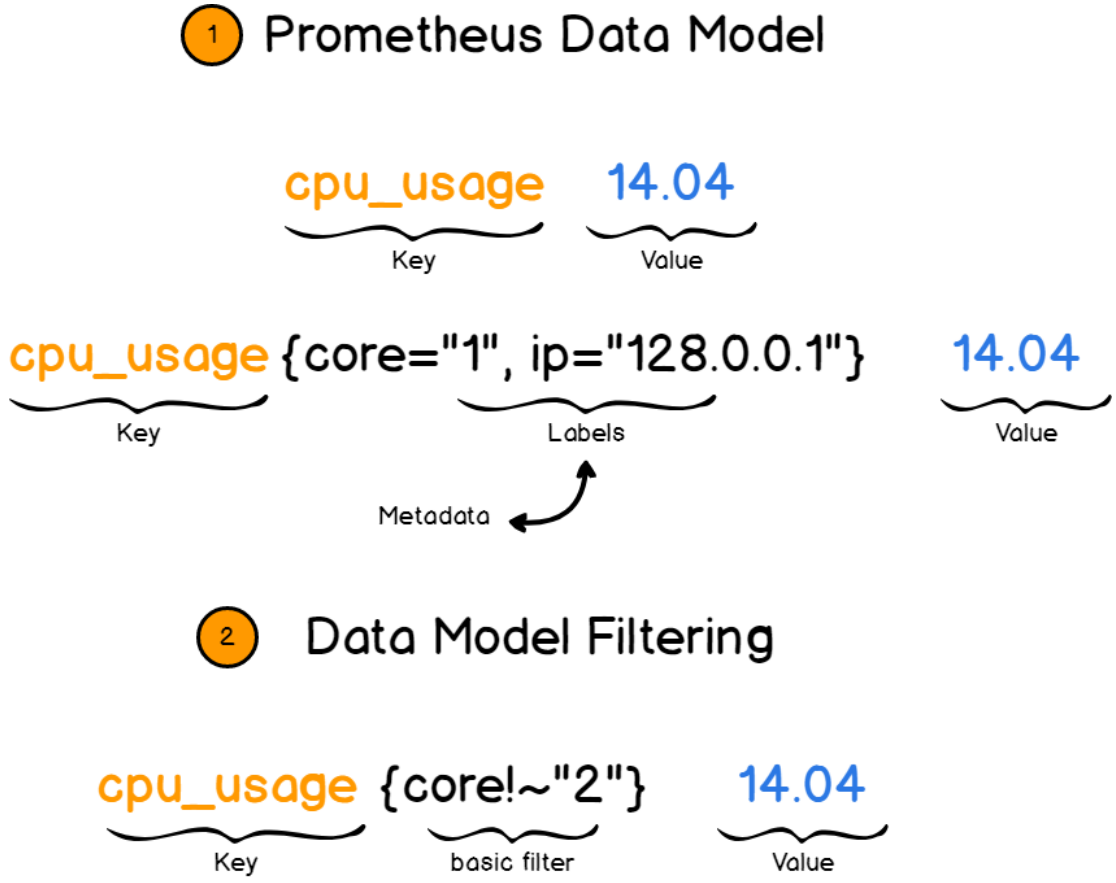
- Prometheus поддерживает следующие инструменты, расширяющие его функционал:

- **Alertmanager**. Prometheus отправляет оповещения в Alertmanager на основе кастомных правил, определенных в файлах конфигурации. Оттуда их можно экспортировать в разные конечные точки (например, Pagerduty или Slack).
- **Визуализация данных**. Как и в Grafana, вы можете визуализировать временные ряды прямо в пользовательском веб-интерфейсе Prometheus. Вы можете фильтровать данные и составлять конкретные обзоры происходящего в разных целевых объектах.
- **Обнаружение сервисов.** Prometheus динамически обнаруживает целевые объекты и автоматически скрейпит новые цели по запросу. Это особенно удобно, если вы работаете с контейнерами, которые динамически меняют адреса в зависимости от спроса.


EC2, Kubernetes, Consul, etc.
Service Discovery
Prometheus
Application
Client Library
Scraping
Rules and Alerts
Alertmanager
Email, PagerDuty, Chat, etc.
Exporter
Storage
Dashboards
3rd Party Application

Концепции Prometheus. Модель данных «ключ-значение»

- Prometheus работает с парами «ключ-значение». Ключ описывает, что мы измеряем, а значение хранит фактическую величину в виде числа.
- Ключом в данном случае называется метрика. Это, например, скорость процессора или занятый объем памяти.
- Если нужно больше деталей о метрике на помощь приходят ярлыки. Ярлыки дают больше сведений о метриках, добавляя дополнительные поля. Например, вы описываете не просто скорость процессора, а скорость одного ядра по определенному IP. Потом вы сможете фильтровать метрики по ярлыкам и просматривать только нужную информацию.



Метрики

- Как только вы автоматизируете управление конфигурацией, настроите оповещение и мониторинг, в вашем распоряжении будут данные, которые можно анализировать. Встает вопрос: «как надежно хранить и анализировать информацию?». Нужна система хранения, которая позволит объединять и изучать показатели системы, поведение пользователей, уровень обслуживания и риски.

- Информация, получаемая из метрик, помогает принимать решения на всех уровнях бизнес-процесса, улучшая вашу способность соответствовать SLA (Service Level Agreement), удовлетворять ожидания клиентов и обосновывать потребность в инвестициях.

Типы метрик Prometheus

- При мониторинге с Prometheus метрики можно описать четырьмя способами.

- Счетчик. Это самый простой тип метрик. Счетчик, как понятно из названия, **считает элементы за период времени**. Если вы хотите посчитать, например, ошибки HTTP на серверах или посещения веб-сайта, используйте **счетчик**. И по логике, разумеется, **счетчик может только увеличивать или обнулять число**, поэтому не подходит для значений, которые могут уменьшаться, или для отрицательных значений. С его помощью особенно удобно считать **количество наступлений** определенного события за период времени, т. е. показатель изменения метрики со временем.

- Измерители. Измерители имеют дело со значениями, которые **со временем могут уменьшаться.** Их можно сравнить с термометрами – если посмотреть на термометр, увидим текущую температуру. Но если измерители могут увеличиваться и уменьшаться и принимать положительные и отрицательные значения, то выходит, они лучше счетчиков, а значит счетчики бесполезны, но это не так. Измерители идеально подходят для измерения текущего значения метрики, которое со временем может уменьшиться. Вот тут-то и кроются подводные камни: измеритель не показывает развитие метрики за период времени. Используя измерители, **можно упустить нерегулярные изменения метрики со временем.** Если система отправляет метрики каждые 5 секунд, а Prometheus скрейпит целевой объект каждые 15, в процессе можно потерять некоторые метрики. Если выполнять дополнительные вычисления с этими метриками, точность результатов окажется еще ниже. У счетчика каждое значение агрегировано. Когда Prometheus собирает его, он понимает, что значение было отправлено в определенный интервал.

- Гистограмма – это более сложный тип метрики. Она предоставляет дополнительную информацию. Например, сумму измерений и их количество. Значения собираются в области с настраиваемой верхней границей. Поэтому гистограмма может:
- **Рассчитывать средние значения**, то есть сумму значений, поделенную на количество значений.
- **Рассчитывать относительные измерения значений**, и это очень удобно, если нужно узнать, сколько значений в определенной области соответствуют заданным критериям. Особенно это полезно, если нужно отслеживать пропорции или установить индикаторы качества.

  Если вы имеете дело с пропорциями, **вам нужны гистограммы.**

- Сводки – **это расширенные гистограммы**. Они тоже показывают сумму и количество измерений, а еще **квантили за скользящий период**. Квантили – это деление плотности вероятности на отрезки равной вероятности.

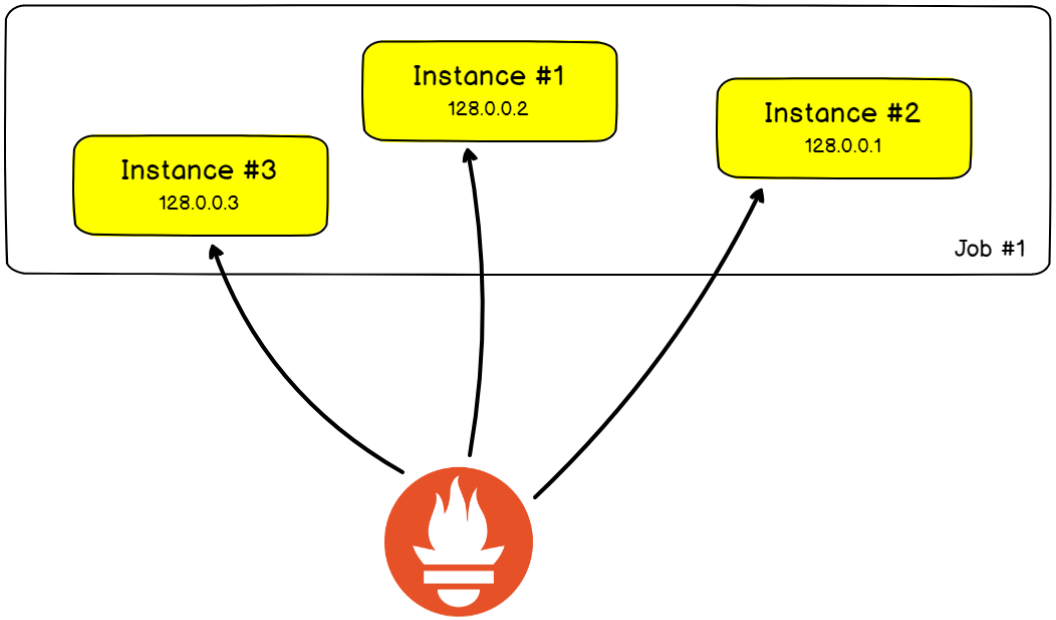
Что лучше: гистограммы или сводки?

Все зависит от **намерения**. Гистограммы объединяют значения за период времени, предоставляя сумму и количество, по которым можно отследить развитие определенной метрики.

Сводки, с другой стороны, показывают квантили за скользящий период (т. е. непрерывное развитие во времени). Это особенно удобно, если вам нужно узнать значение, которое представляет 95% значений, записанных за период.

Задания и экземпляры

- **Серверы реплицируются и распределяются по всему миру.**
- Чтобы это проиллюстрировать, давайте рассмотрим классическую архитектуру из двух серверов HAProxy, которые перераспределяют нагрузку по девяти бэкенд-веб-серверам.
  В этом примере из реальной жизни мы отследим **число ошибок HTTP, возвращенных веб-серверами.**
- На языке Prometheus один веб-сервер называется **экземпляром**. **Заданием** будет тот факт, что вы измеряете число ошибок HTTP на всех экземплярах.



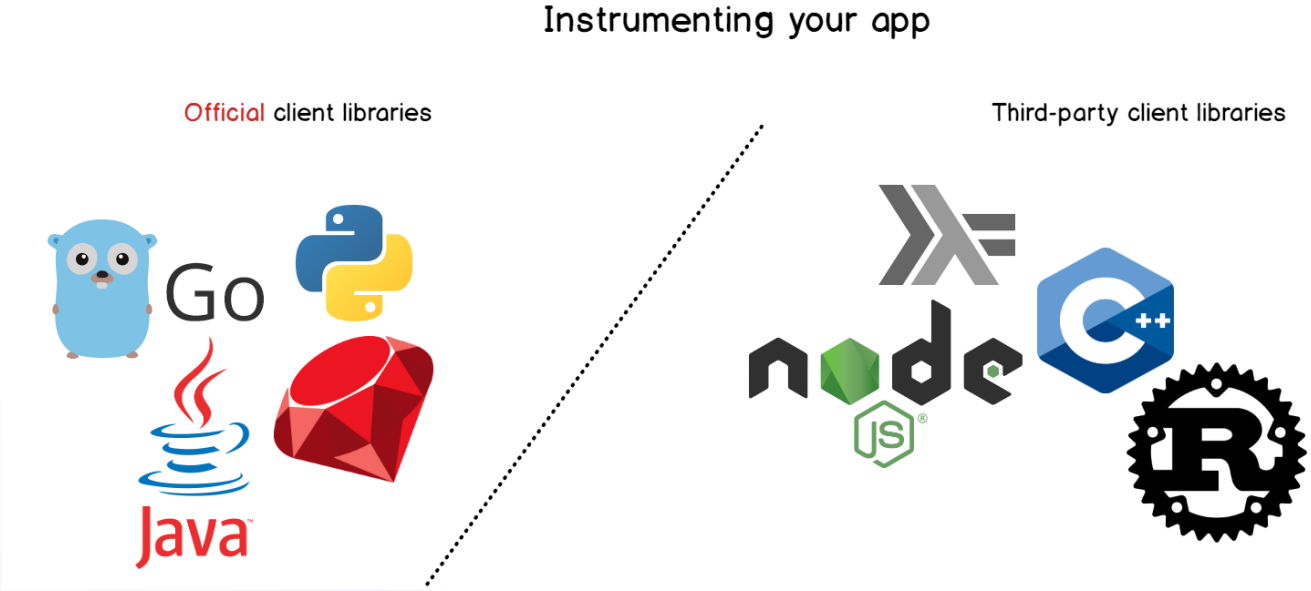
- Прелесть в том, что задания и экземпляры — это поля в ярлыках, и вы можете фильтровать результаты по определенному экземпляру или заданию.

```
cpu_usage {job="1", instance="128.0.0.1"}  14.04
cpu_usage {job="1", instance="128.0.0.2"}  12.01
cpu_usage {job="1", instance="128.0.0.3"}  16.03
```

Инструментирование

- **Инструментирование** – это еще одна важная часть Prometheus. Вы инструментируете приложения, прежде чем извлекать из них данные. На языке Prometheus инструментирование означает добавление клиентских библиотек в приложение, чтобы они предоставляли метрики Prometheus.

- Инструментирование доступно для большинства распространенных языков программирования: например, **Python, Java, Ruby, Go и даже Node или C#**. По сути, вы создаете объекты памяти (например, измерители или счетчики), которые будут динамически увеличивать или уменьшать значение. Потом вы выбираете, где предоставлять метрики. Prometheus заберет их оттуда и сохранит в свою базу данных временных рядов.



Примеры использования Prometheus. DevOps

- Со всеми этими экспортерами для разных систем, баз данных и серверов очевидно, что Prometheus предназначен, в основном, для **сферы DevOps**.

- **Prometheus идеально подходит для DevOps.** Для настройки и запуска экземпляров почти не требуется усилий, и можно легко активировать и настроить любой вспомогательный инструмент. Благодаря **обнаружению целевых объектов** – например, через файловый экспортер, – это отличное решение для стеков, где широко используются контейнеры и распределенные архитектуры. В среде, где экземпляры то и дело создаются и удаляются, ни один стек DevOps не обойдется без **обнаружения сервисов.**

- Помимо DevOps, Prometheus используется в здравоохранении (Спрос растет, и ИТ-архитектуры обязаны ему соответствовать. Если нет надежного инструмента для мониторинга всей инфраструктуры, **есть риск столкнуться с серьезными перебоями в обслуживании**. В сфере здравоохранения такую опасность точно надо свести к минимуму) и финансовых услугах.

Сравним три наиболее широко используемых альтернативы (Prometheus, Grafana и Graphite)

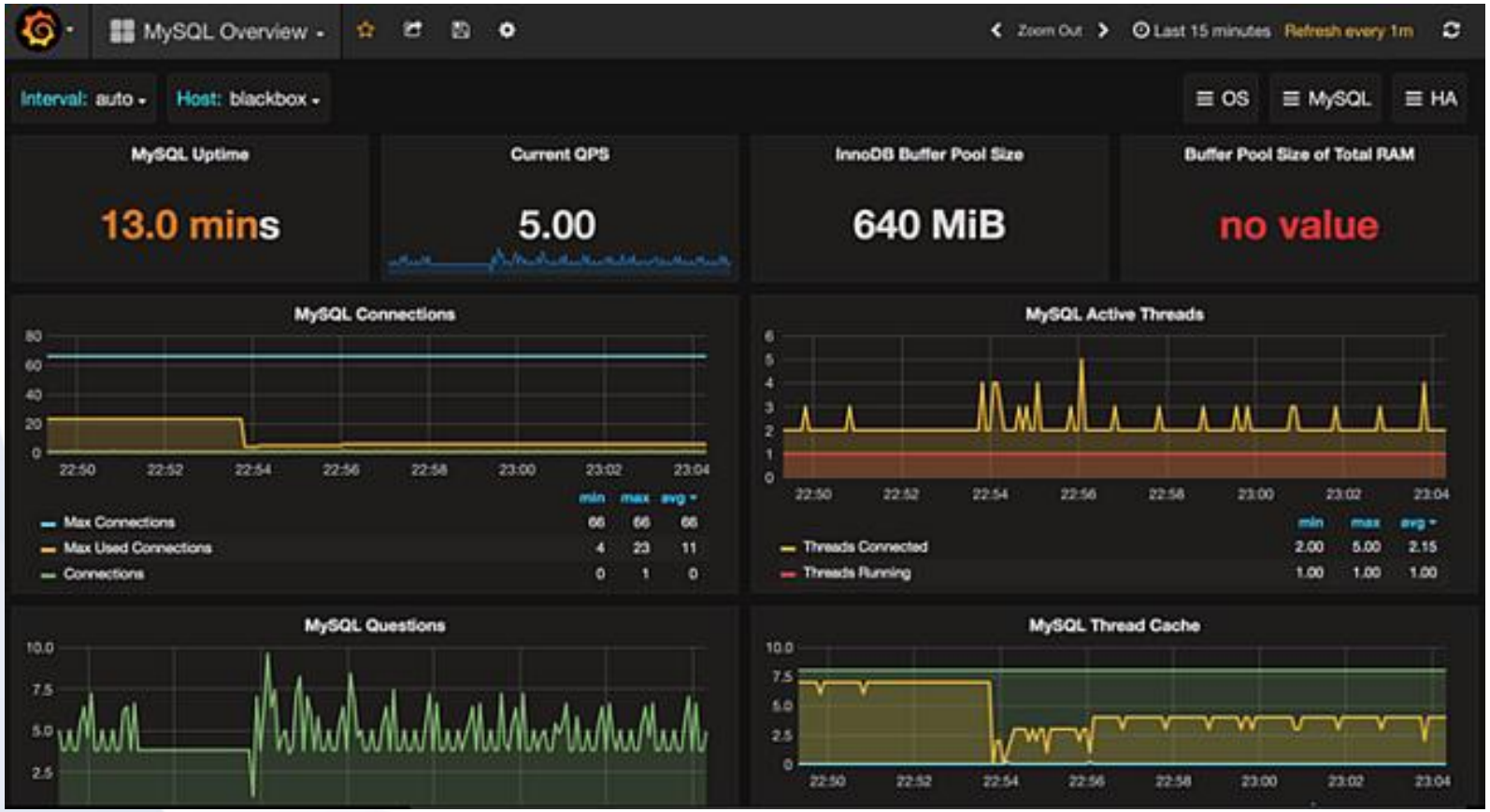
- Чтобы определить их сильные и слабые стороны. Мы будем использовать следующие параметры в качестве общих точек сравнения между всеми тремя вариантами:
  - Визуализация и редактирование панели;
  - Место хранения;
  - Сбор информации;
  - Плагин, архитектура и расширяемость;
  - Тревога и отслеживание событий;
  - Совместимость облачного мониторинга;
  - Open Source против коммерческих предложения.

Визуализация и редактирование панели инструментов

- Это та часть, где вы проектируете и строите графики метрик/временных рядов и упорядочиваете их на инструментальных панелях. Обзор преимуществ систем мониторинга.

- Grafana: С точки зрения визуализации, создания и настройки панели мониторинга, Grafana – лучший из всех вариантов. Он многофункциональный, простой в использовании и очень гибкий.
- Graphite: Хорошие параметры визуализации, но в его основные функции не входит редактирование приборной панели. В реальном мире Graphite используется в сочетании с Grafana; Graphite хранит данные, а Grafana – визуализацию.
- Prometheus: Отлично, но в целом сложно использовать функции редактирования графиков и приборной панели. Prometheus использует консольные шаблоны для визуализации и редактирования панели, но поначалу обучение этим консольным шаблонам может оказаться трудным. В реальном мире я рекомендую начать с использования Grafana для редактирования графиков и панелей, а затем (когда достигнут уровень) перейти к шаблонам консоли Prometheus.

- **Победитель:** Grafana побеждает здесь с большим отрывом, в то время как Prometheus занимает второе место.



Место хранения

- Визуализация – это часть задачи, но мы не можем визуализировать временные ряды из воздуха. Нам нужно получить их из источника, и этот источник должен каким-то образом хранить все временные ряды и предоставлять возможность их запроса:

  - Grafana: Нет поддержки хранения временных рядов. Grafana – это только решение для визуализации. Хранение временных рядов не является частью его основной функциональности.
  - Graphite: На этот раз Graphite побеждает Grafana. Graphite может хранить временные ряды, полученные из других источников (обычно это инструменты прямого мониторинга), и предоставлять язык запросов для получения хранимых данных. Опять же, Grafana может использоваться с Graphite для визуализации данных, хранящихся на его внутренней стороне хранилища.
  - Prometheus: Способ, которым Prometheus хранит временные ряды, является лучшим на сегодняшний день (благодаря своей многомерной модели, которая использует теги ключ-значение вдоль временных рядов для лучшей организации данных и предоставления мощных возможностей запросов). Как упоминалось ранее, Grafana может использоваться с языком запросов Prometheus для создания графиков и информационных панелей.

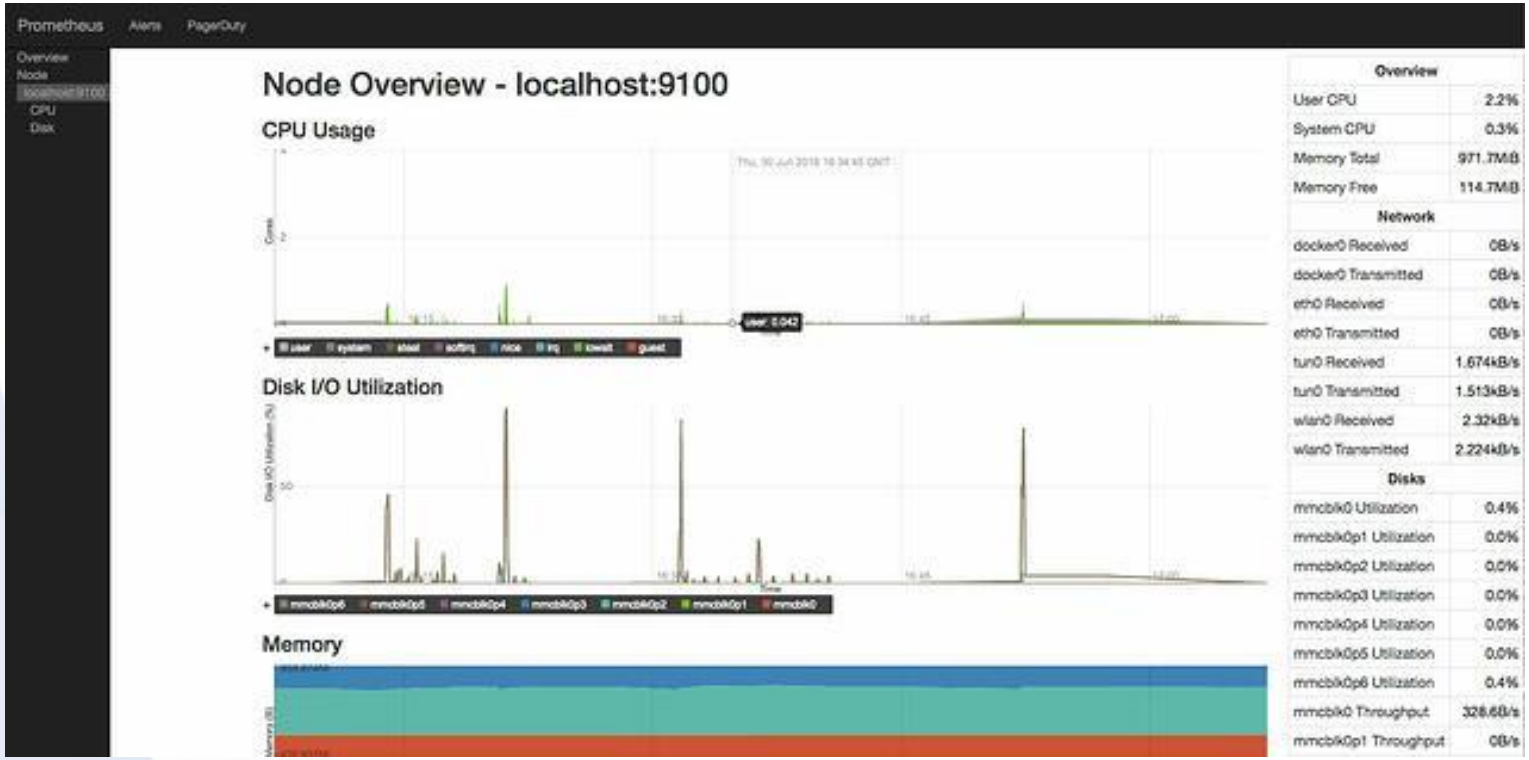
- **Победитель:** Prometheus выделяется здесь с Graphite, занимающим тут второе место, и Grafana как абсолютный неудачник.

Сбор информации

- Допустим, у вас есть и хранилище, и визуализация, но вам необходимо получить данные из ваших служб. И вот тут нам поможет прямой мониторинг. Используя старые методы (SNMP) или новые (агенты), вам нужен способ получения метрик, которые в конечном итоге будут храниться в виде временных рядов:

- Grafana: Нет поддержки сбора данных. Опять же, Grafana – это всего лишь решение для визуализации. Ни хранение временных рядов, ни сбор временных рядов не являются частью его основной функциональности.
- Graphite: Сбор данных также не поддерживается, по крайней мере, напрямую. Вам необходимо включить такие решения, как statd, collectd и другие, чтобы сделать часть сбора данных функциональной. Graphite получит все данные из этих источников и сохранит эти данные в виде временных рядов в своей внутренней памяти.
- Prometheus: Да, Prometheus может выполнять сбор данных вместе с хранением и визуализацией. Это очень полное решение, как и другие инструменты мониторинга (Cacti, Nagios и Zabbix).

- **Победитель:** Prometheus снова побеждает, а Graphite и Grafana оба проигрывают в этой гонке.



Архитектура и расширяемость плагинами

- Одним из сильных сторон всех современных программных решений является возможность расширения за счет использования плагинов или других подобных средств. Таким образом, вы можете расширить уже имеющиеся основные функции и включить в свое решение набор совершенно новых функций:

  - Grafana: Да, поддерживается и с большим набором плагинов, применяемых к источникам данных, приложениям и редактированию панели мониторинга.
  - Graphite: Да, определенным образом. Graphite на самом деле не предоставляет и не имеет подключаемой библиотеки. Вместо этого есть много инструментов, которые уже совместимы с Graphite.
  - Prometheus: Опять да, определенным образом. Prometheus называет их экспортерами. Экспортеры позволяют сторонним инструментам экспортировать свои данные в Prometheus. Кроме того, некоторые программные компоненты в мире с открытым исходным кодом уже совместимы с Prometheus.

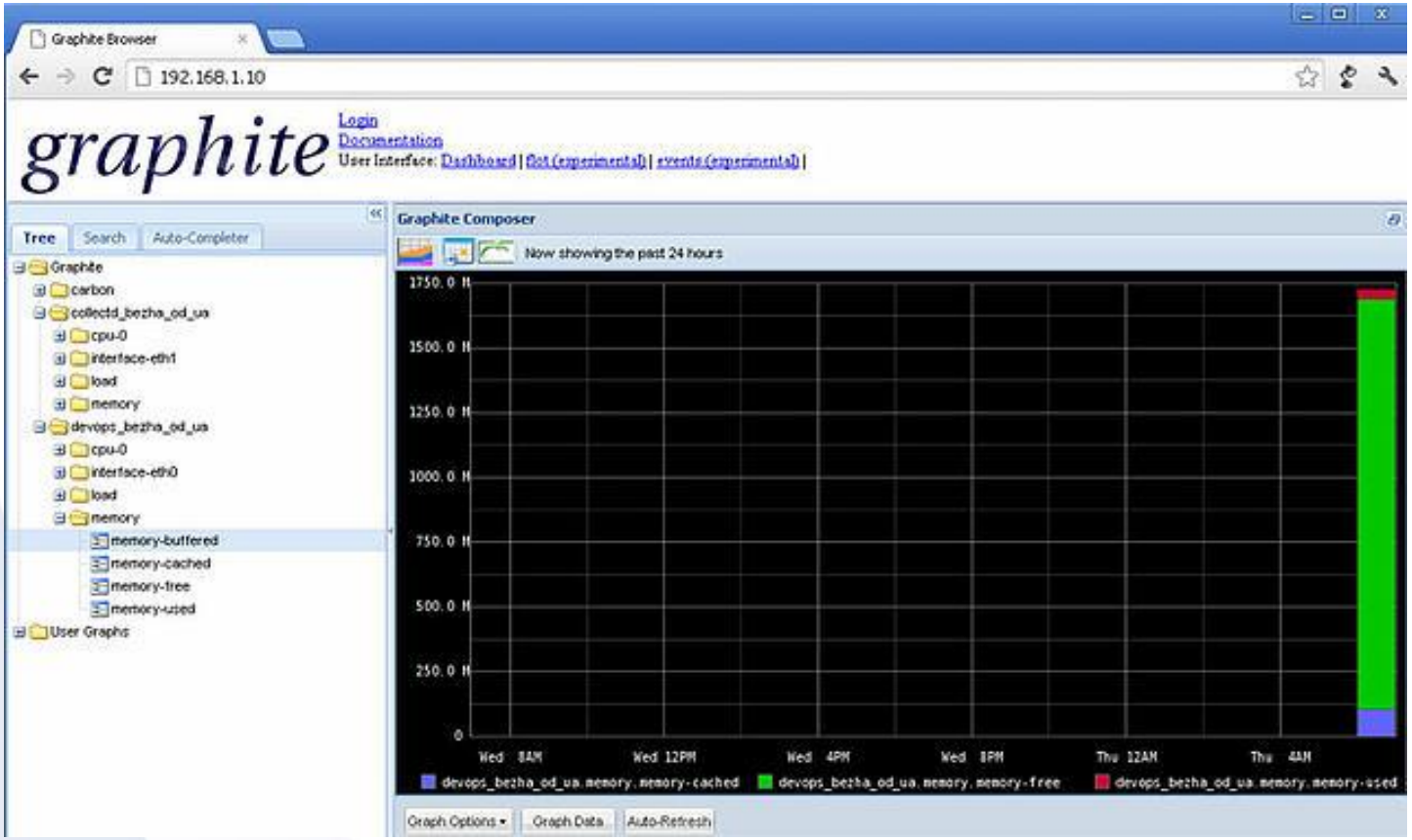
- **Победитель:** Все они. Правда у Grafana есть настоящие плагины, которые расширяют его основные функциональные возможности, но есть много инструментов, которые так или иначе совместимы как с Graphite, так и с Prometheus.

Тревога и отслеживание событий

- Решение по мониторингу не является полным, если только вы не включите способ генерировать сигналы тревоги, когда какой-либо показатель начинает действовать забавно. Кроме того, отслеживание событий – это хороший способ связать повторяющиеся события, которые могут помочь вам лучше диагностировать проблемы в вашей инфраструктуре:

- Grafana: Нет, или, по крайней мере, не напрямую. Grafana может только визуализировать временные ряды, и в этой задаче она превосходит все остальные, но ни управление тревогами, ни отслеживание событий не являются частью его основных функций. Косвенным образом существуют способы преобразования журналов вхождений в числа, что позволяет отслеживать события.
- Graphite: Он может выполнять отслеживание событий, но не может напрямую выполнять сигнализацию.
- Prometheus: Полностью поддерживает сигнализацию. Несмотря на то, что прямое отслеживание событий не включено, очень мощный язык запросов Prometheus позволяет вам выполнять некоторые действия, чтобы косвенно выполнять часть отслеживания событий.

- **Победитель:** Беспорный победитель Prometheus. Graphite финиширует на втором месте, а Grafana даже не доходит до финиша.



Совместимость облачного мониторинга

- Некоторые облачные сервисы, такие как AWS и OpenStack, включают свою собственную инфраструктуру мониторинга, которая собирает и хранит временные ряды, а в некоторых случаях также предоставляет базовые возможности редактирования графиков и инструментальных панелей. Мы не хотим вдаваться в подробности в этой части статьи, поэтому мы просто поговорим об общедоступных облаках с использованием AWS и частных облаках с использованием OpenStack.
- Сервис мониторинга AWS называется Cloudwatch, который включает в себя не только хранилище данных для всех его метрик, основанных на временных рядах, но также включает в себя базовое редактирование графиков и приборной панели.
- OpenStack (особенно в его последних выпусках) включает Gnocchi, который представляет собой решение «Временные ряды как услуга», в котором пока нет прямого графика и компонента редактирования. Давайте посмотрим, как наши три соперника могут интегрироваться как с AWS, так и с OpenStack.

- Grafana: Лучшее решение на данный момент. В связи с тем, что оба облачных решения (AWS и OpenStack) уже осуществляют сбор данных, хранение данных и даже управление аварийными сигналами, единственное, что вам действительно нужно, — это визуализация и создание панели мониторинга. Grafana включает поддержку (через плагин) для AWS Cloudwatch и OpenStack Gnocchi. Если ваше развертывание полностью основано на облаке, и в него включено решение для мониторинга (Cloudwatch или Gnocchi), не используйте ничего, кроме Grafana.
- Graphite: В GitHub уже доступны некоторые компоненты, которые можно использовать для передачи данных AWS Cloudwatch в Graphite, но, опять же, в этом нет необходимости, и трату времени, учитывая, что Cloudwatch уже доступен для функций, которые будет охватывать Graphite. Для OpenStack сообщество рекомендует использовать Gnocchi только с Grafana. Я бы порекомендовал вам не тратить свое время и придерживаться уже доступных параметров мониторинга в облаке.
- Prometheus: Есть официальный экспортер AWS Cloudwatch, так что вы можете контролировать все свои облачные компоненты AWS с помощью Prometheus, но пока нет поддержки для OpenStack Gnocchi. Важно отметить, что, хотя Gnocchi поддерживает как collectd, так и statsd (опции с экспортерами в Prometheus), поддержка является однонаправленной, то есть вы можете отправлять метрики collectd / statsd в Gnocchi, но не наоборот.

- **Победитель:** Grafana – единственный победитель, а остальные претенденты на второе место. В идеале, вы должны придерживаться предложения по мониторингу, уже доступного в облаке, и дополнять только там, где это необходимо. Вот почему Grafana является лучшим вариантом здесь. Сбор и хранение временных рядов уже охвачены как Cloudwatch, так и Gnocchi.

Открытый исходный код против Enterprise

- Во многих проектах с открытым исходным кодом общепринятая практика заключается в том, чтобы включать какое-то корпоративное/коммерческое предложение с дополнительными опциями. Давайте рассмотрим, что могут предложить в качестве дополнения наши три претендента:

  - Grafana: Модель с открытым исходным кодом полностью функциональна и готова к работе. Специальной коммерческой версии не существует, но есть хостинговое решение, предоставляемое Grafana и управляемое ими.
  - Graphite: Модель с открытым исходным кодом полностью функциональна и готова для предприятий.
  - Prometheus: Как и две другие, модель с открытым исходным кодом полностью функциональна и готова для предприятий.

- **Победитель:** Grafana может быть объявлен как победитель, потому что он предлагает вариант размещения. Но, если учесть, что все варианты полнофункциональны в предложениях с открытым исходным кодом, то все они получают первое место.

Выводы

- **Облачные сервисы, такие как AWS и OpenStack:** Если ваша инфраструктура полностью основана на облаке, и у вас уже есть доступные метрики из таких вариантов, как Cloudwatch или Gnocchi, выберите Grafana. Вам не нужно хранить временные ряды или определять аварийные сигналы. Все, что вам нужно, это преодолеть графические ограничения, унаследованные от Cloudwatch и Gnocchi, и отобразить ваши метрики в умном, удобном и многофункциональном виде. Именно здесь Grafana превосходит все остальные варианты.
- **Классическая инфраструктура с базовыми решениями для сбора данных:** Если ваша инфраструктура использует такие вещи, как collectd, statd или другие подобные инструменты только для сбора данных, и при условии, что они могут использоваться Graphite, то используйте Graphite для выполнения части хранения временных рядов на централизованном сервере и добавьте Grafana в ваш микс, чтобы правильно отобразить эти показатели. Обратите внимание, что Graphite может отслеживать события, но это не то же самое, что генерация аварийных сигналов, поэтому для выполнения этой задачи вам понадобится что-то еще.
- **Любая инфраструктура без какого-либо мониторинга:** Если вы начинаете с нуля и у вас нет других доступных вариантов мониторинга, то выбирайте Prometheus. Изначально вы можете добавить Grafana, чтобы упростить редактирование графиков и информационных панелей, пока вы не овладеете навыками использования шаблонов консоли Prometheus.

Используемые источники

- Сравнение Prometheus, Grafana, Graphite – https://dataenginer.ru/?p=1454
- Инструменты DevOps – https://proglib.io/p/12-instrumentov-devops-inzhenera-dlya-monitoringa-arhitektury-2020-02-13
- Руководство по Prometheus – https://habr.com/ru/company/southbridge/blog/455290/
- Мониторинг программ – https://ru.wikipedia.org/wiki/Мониторинг_программ

Спасибо за внимание