нетология

# Системы для мониторинга

Сергей
Бывшев

# Сергей Бывшев

## Ведущий инженер автоматизации в "Метр квадратный"

Сергей Бывшев

# План занятия

1. Введение
2. TICK
3. Prometheus
4. Zabbix
5. Итоги
6. Домашнее задание

# Введение

# Введение

На текущий момент существует множество систем мониторинга IT-систем.

В данной лекции мы рассмотрим наиболее популярные решения, которые применяются в подавляющем большинстве организаций:

- **TICK** (Telegraf/Influxdb/Chronograf/Kapacitor)
- **Prometheus**
- **Zabbix**

Существует также множество других систем мониторинга, но они все в той или иной степени схожи с представленными в данной лекции.

# Введение

Системы мониторинга можно разделить на 2 подтипа:
**Push-модель** и **Pull-модель**.

Данные подтипы характеризуют процесс сбора метрик внутри системы мониторинга.

- **Push-модель** подразумевает отправку данных с агентов (рабочих машин, с которых собираем мониторинг) в систему мониторинга, посредством вспомогательных служб или программ (обычно посредством UDP).
- **Pull-модель** подразумевает последовательный или параллельный сбор системой мониторинга с агентов накопленной информации из вспомогательных служб.

# Введение

**Плюсы Push-модели:**

- Упрощение репликации данных в разные системы мониторинга или их резервные копии
  (на клиенте настраивается конечная точка отправки или набор таких точек)
- Более гибкая настройка отправки пакетов данных с  метриками
  (на каждом клиенте задается объем данных и частоту отправки)
- UDP является менее затратным способом передачи данных, вследствии чего может вырости производительность сбора метрик
  (обратной стороной медали является гарантия доставки пакетов)

# Введение

**Плюсы Pull-модели:**

- Легче контролировать подлинность данных
  (гарантия опроса только тех агентов, которые настроены в системе мониторинга)
- Можно настроить единый proxy-server до всех агентов с TLS
  (таким образом мы можем разнести систему мониторинга и агенты, с гарантией безопасности их взаимодействия)
- Упрощенная отладка получения данных с агентов
  (так как данные запрашиваются посредством HTTP, можно самостоятельно запрашивать эти данные, используя ПО вне системы мониторинга)

# Введение

**TSDB** - (time-series database) база данных, чаще всего используемая в системах мониторинга. Данные системы характеризуются возможностью эффективного хранения пар “Метка времени - значение”.

Такие базы данных характеризуются эффективным встроенными алгоритмами сжатия высокочастотных временных рядов и алгоритмами индексации временных данных.

В качестве базы данных для мониторинга можно использовать также и “более традиционные БД”, такие как MySQL или PostgreSQL. Но в этом случае возможны потери производительности, что является критичным для системы мониторинга.

**Важно!** Некоторые системы мониторинга (например Zabbix) используют только реляционные БД в качестве системы хранения метрик.

# TICK

# TICK

TICK - представляет из себя набор компонентов для эффективного мониторинга систем.

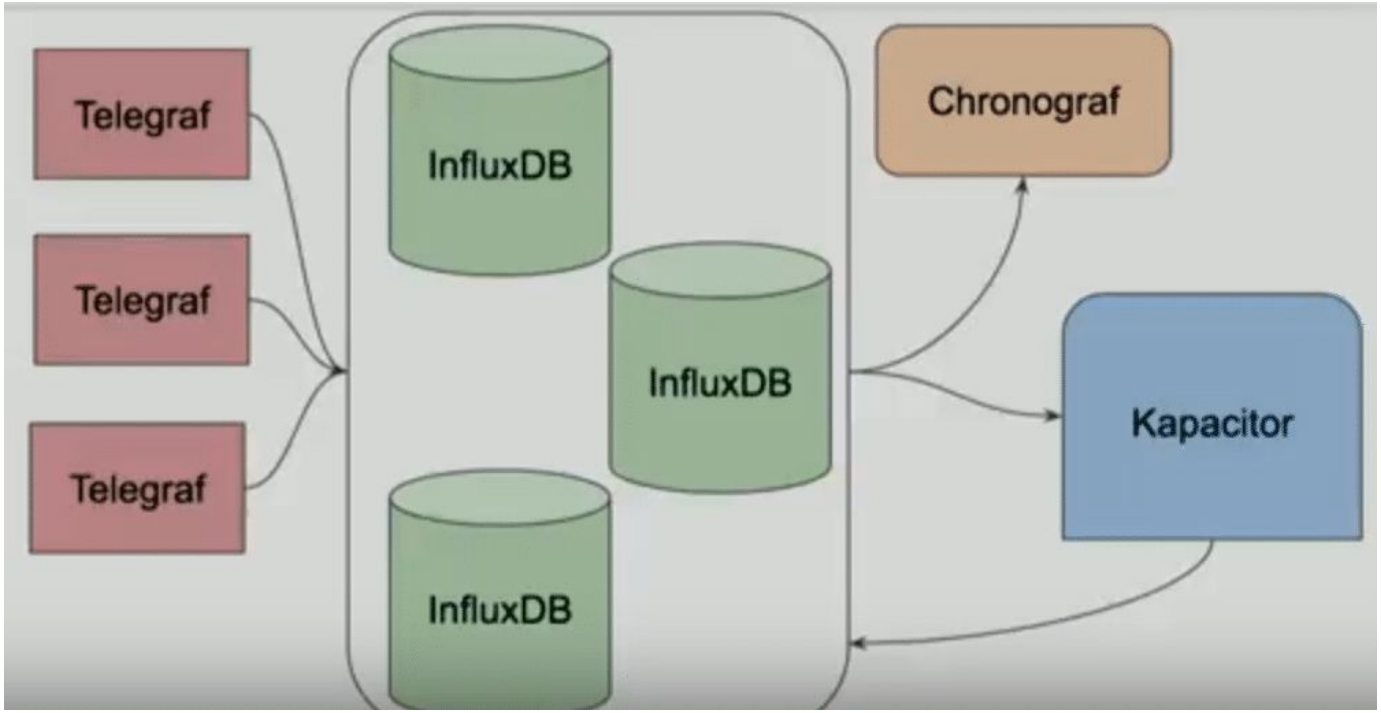
TICK работает в соответствии с **push-моделью**.

Набор компонентов TICK стэка:

- Telegraf (агент для сборки метрик с хост-машин и отправки в TSDB)
- InfluxDB (TSDB для хранения метрик)
- Chronograf (Компонент для визуализации и настройки данных TSDB)
- Kapacitor (Система для настройки правил оповещения)

Данный стэк технологий представляет из себя “коробочную” версию системы мониторинга, которую можно использовать, без дополнительных инструментов.

# TICK




Взято с сайта: influxdata.com

# TICK

**Telegraf** представляет из себя golang-приложение, которое производит сбор метрик и отправляет их в TSDB согласно конфигурации.

Конфигурирование Telegraf осуществляется через conf файл, где указываются входные параметры для сбора и конечные системы для отправки данных.

**Важно!** Telegraf не ограничивается только InfluxDB как конечной точкой для передачи данных. Также Telegraf можно настроить для работы в соответствии с Pull-моделью

# TICK

Пример конфигурации для сборки “железных метрик” сервера и передачи их в InfluxDB:

```
[[inputs.cpu]]
    percpu = true
    totalcpu = true
    collect_cpu_time = false
    report_active = false
[[inputs.mem]]
[[inputs.disk]]
[[inputs.kernel]]
[[outputs.influxdb]]
    urls = ["udp://1.2.3.4:8089"]
```

Взято с сайта: influxdata.com

# TICK

**InfluxDB** - time-series database, осуществляющая взаимодействия с данными через SQL подобный синтаксис (InfluxQL).

Конфигурирование InfluxDB осуществляется через conf файл, где указываются входные параметры хранения данных, производительности БД и сетевой интерфейс для доступа к данным.

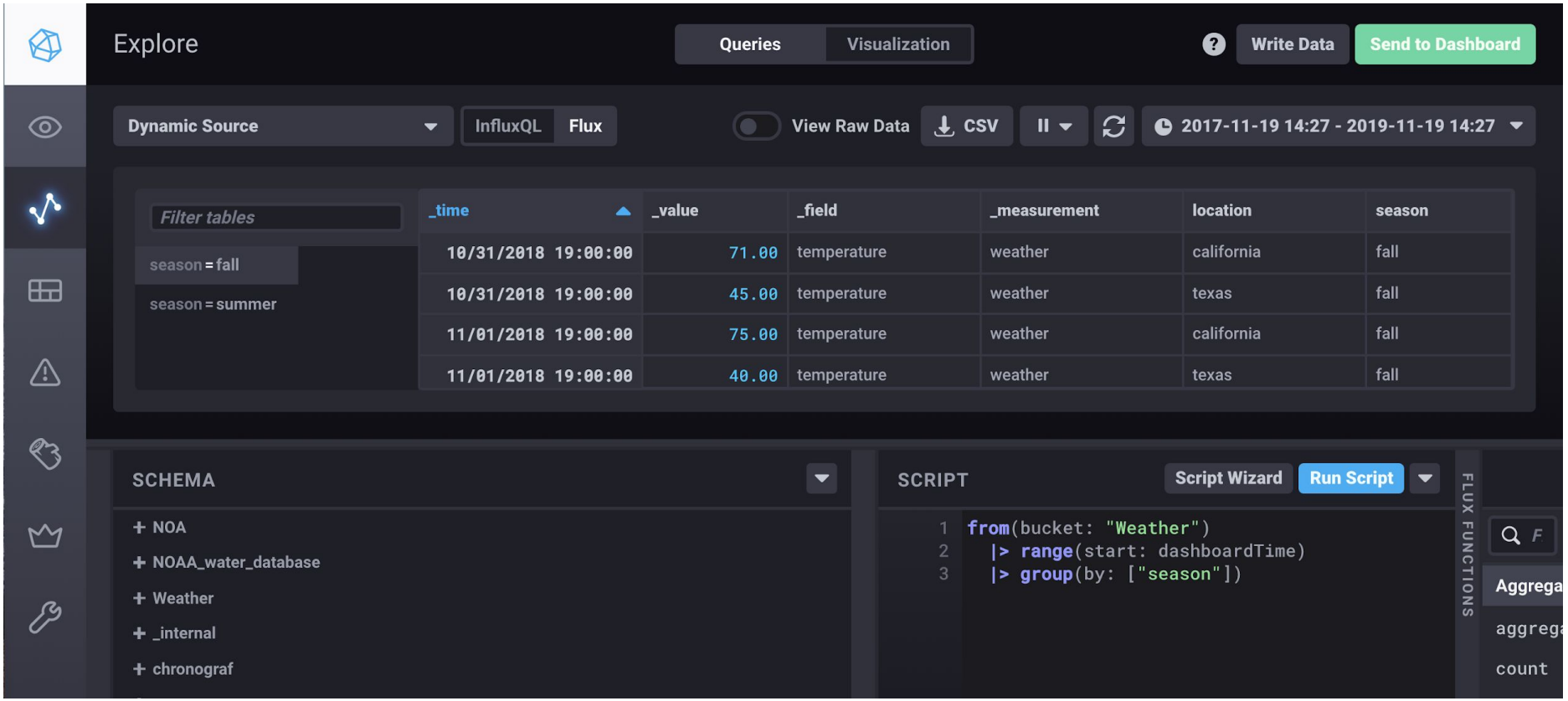

Взято с сайта: influxdata.com

# TICK

**Chronograf** - веб интерфейс, предоставляющий доступ к данным InfluxDB и повзоляющий производить настройки времени жизни данных и правил оповещения Kapacitor.


Взято с сайта: influxdata.com

# TICK




Взято с сайта: github.com

# TICK

**Kapacitor** - компонент для настройки правил оповещения из системы мониторинга.

Пример настроенного правила оповещения:

```
stream
    |from()
        .measurement('cpu_usage_idle')
        .groupBy('host')
    |window()
        .period(1m)
        .every(1m)
    |mean('value')
    |eval(lambda: 100.0 - "mean")
        .as('used')
    |alert()
        .message('{{ .Level}}: {{ .Name }}/{{ index .Tags "host" }}
        .warn(lambda: "used" > 70.0)
        .crit(lambda: "used" > 85.0)

        // Slack
        .slack()
        .channel('#alerts')
```

Взято с сайта: ageropeek.fr

# Prometheus

# Prometheus

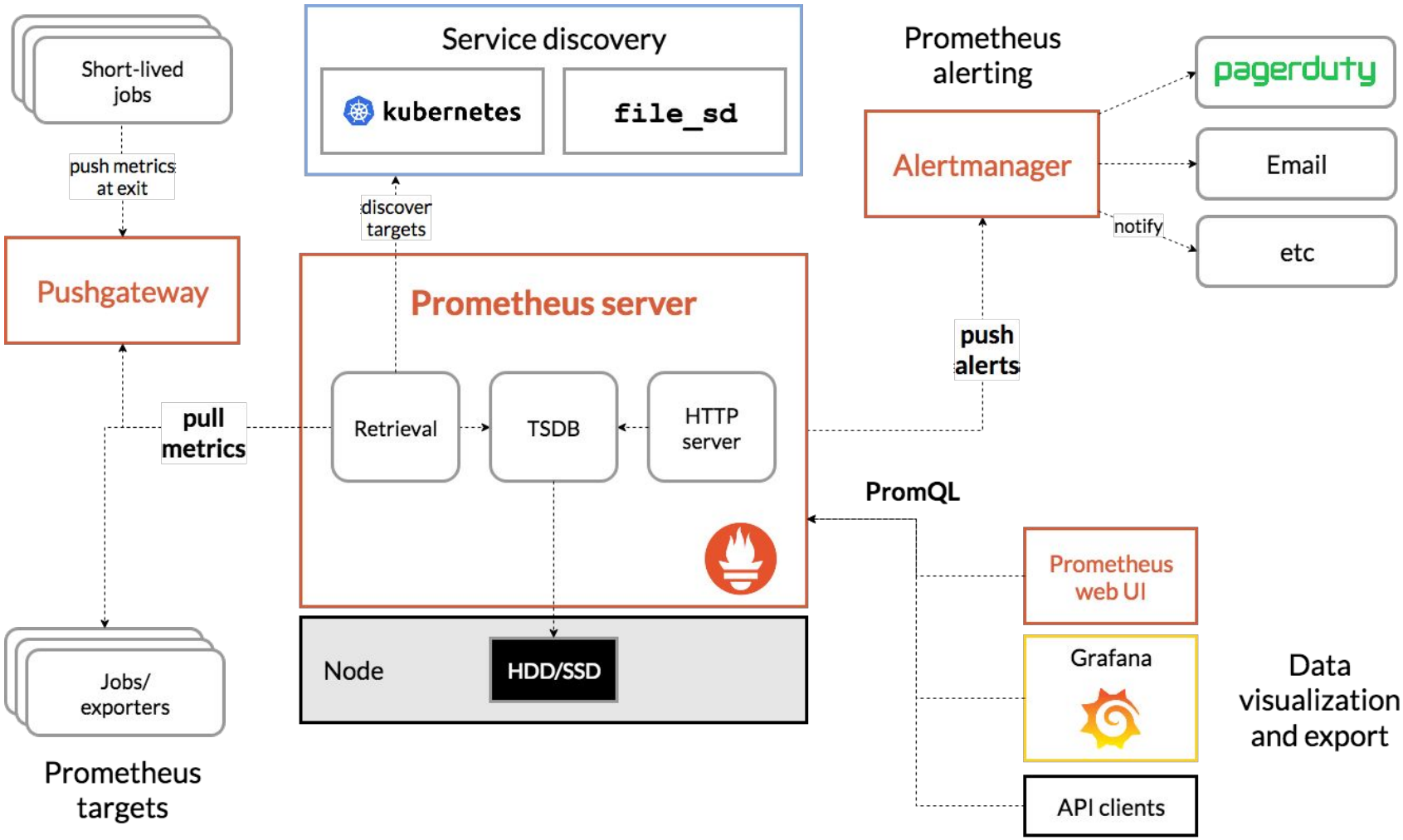
**Prometheus** - также представляет из себя набор компонентов для эффективного мониторинга систем. Prometheus работает в соответствии с **pull-моделью**.

Набор компонентов Prometheus стэка:

- Exporter (агент для сборки метрик с хост-машин и хранения до сбора системой мониторинга)
- Server (хранение данных и их менеджмент)
- Web UI (веб интерфейс для доступа к данным и конфигурирования системы)
- Alert Manager (система оповещения)

Данный стэк технологий представляет из себя “коробочную” версию системы мониторинга, которую можно использовать, без дополнительных инструментов.

# Prometheus




Взято с сайта: prometheus.io

# Prometheus

Почти весь набор компонентов prometheus не требует дополнительных настроек и работает “из коробки”.

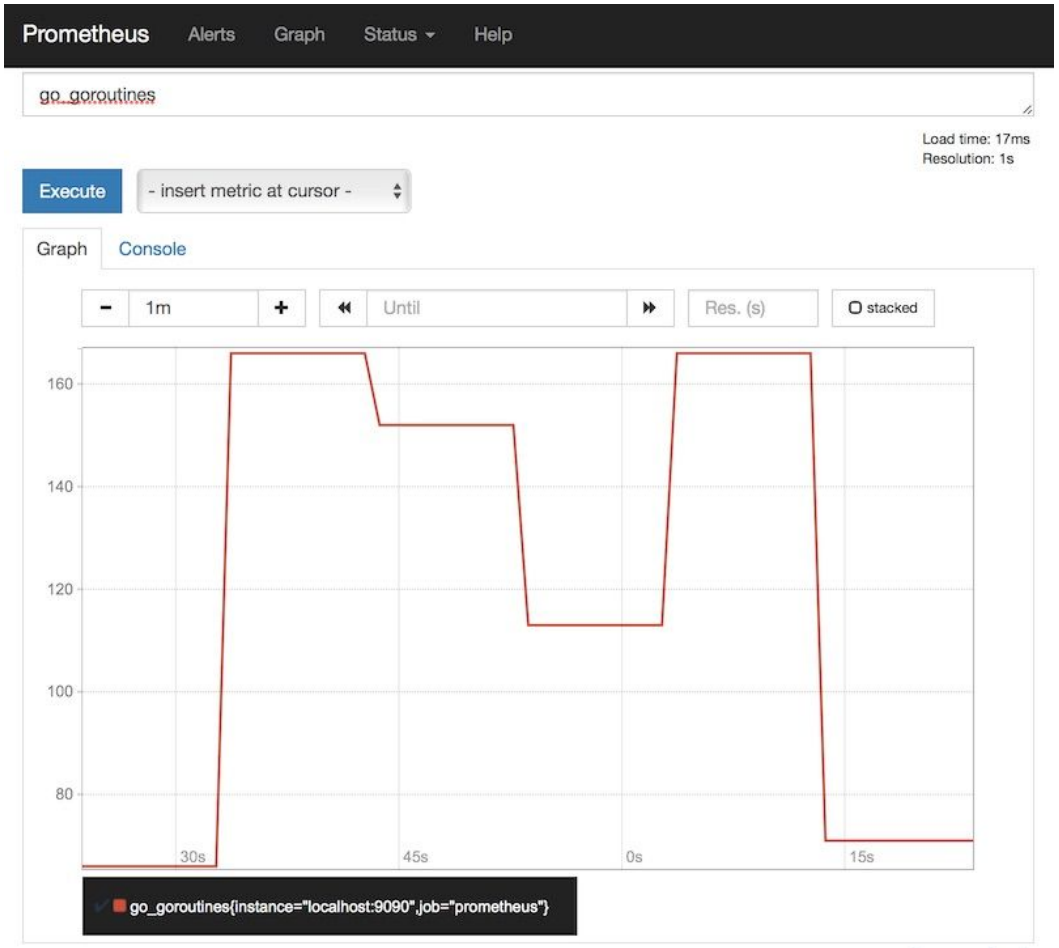
В соответствии с характеристиками работы pull-модели нам нужно лишь указать для Server компонента узлы для сбора метрик. Это производится в конфигурационном файле (yml формат):

```
global:
  scrape_interval: 15s

scrape_configs:
- job_name: node
  static_configs:
  - targets: ['localhost:9100']
```

Взято с сайта: prometheus.io

# Prometheus




Взято с сайта: habr.com

# Zabbix

# Zabbix

**Zabbix** - эффективная, зарекомендовавшая себя система мониторинга. Zabbix работает в соответствии с **push и pull моделью**.

Набор компонентов Zabbix стэка:

- **Agent** (агент для сборки метрик с хост-машин и передачи в систему хранения)
- **Server** (хранение данных конфигурации стека и статистики)
- **Proxy** (средство оптимизации нагрузки на server компонент)
- **Web UI** (веб интерфейс для менеджмента данных, их визуализации, конфигурирования стека и настроек правил оповещения)
- **Database** (хранения метрик мониторинга приложений)

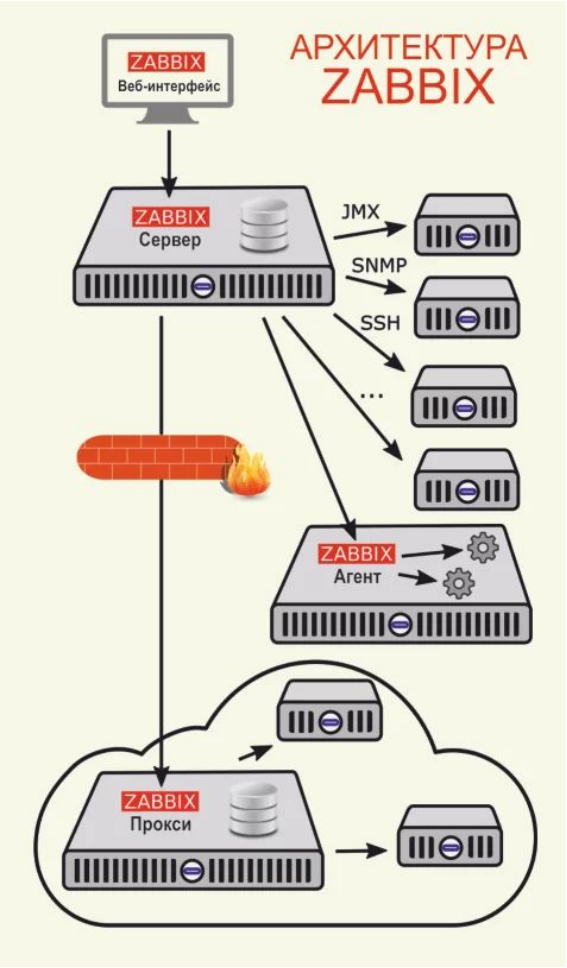
Данный стэк технологий представляет из себя “коробочную” версию системы мониторинга, которую можно использовать, без дополнительных инструментов.

# Zabbix

Механизм дуализма pull и push моделей в Zabbix заключается в настраиваемых “Активных и Пассивных” проверках.

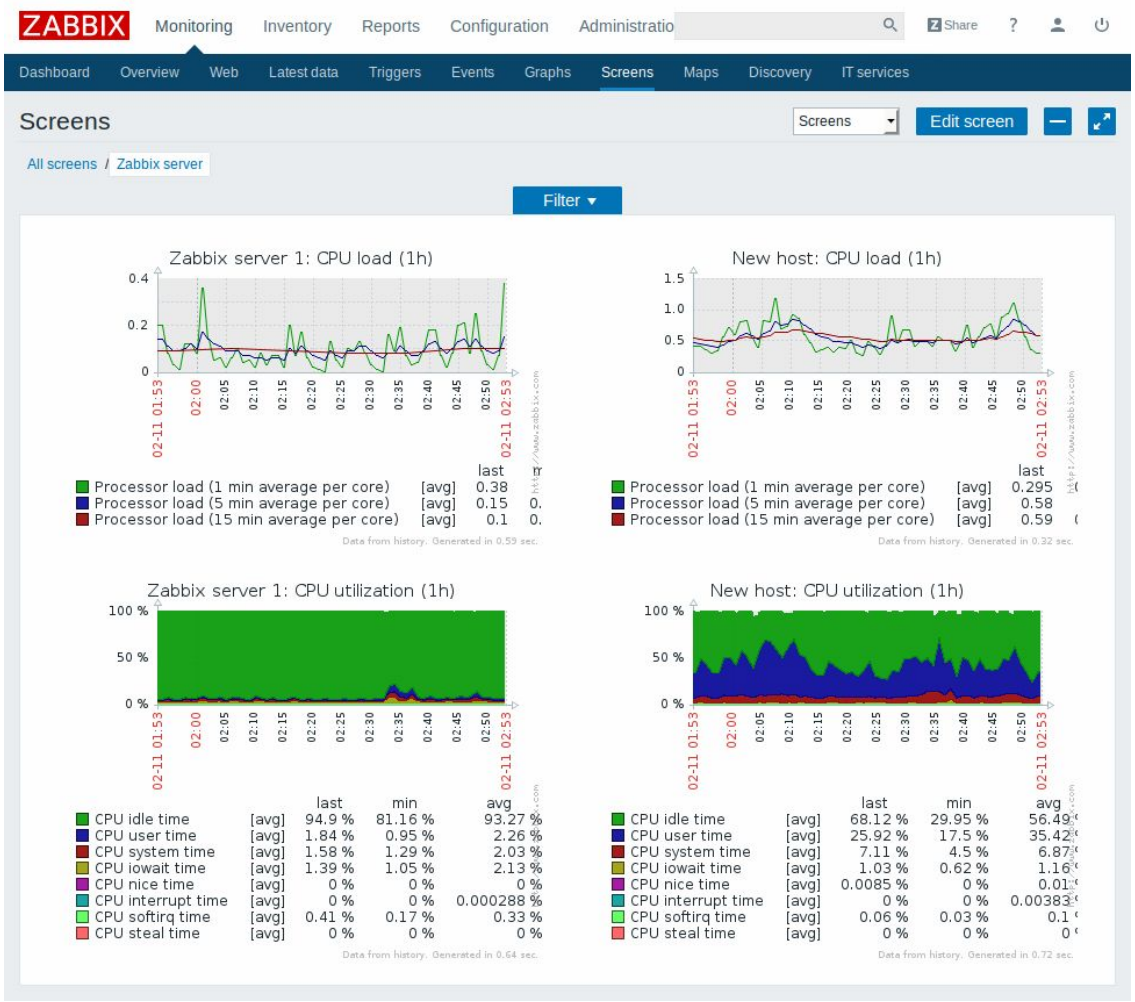
**Активные проверки** - указываются данные для непрерывного наблюдения, которые перенаправляются агентом серверу.

**Пассивные проверки** - набор данных, которые собираются и хранятся на агенте, но отправляются на сервер только по запросу.




Взято с сайта: etrnalhost.net

# Zabbix




Взято с сайта: zabbix.com

# Итоги

# Итоги

В данной лекции мы:

- Узнали что такое Pull и Push модели мониторинга, а также их сильные стороны
- Ознакомились с понятием TSDB
- Рассмотрели наиболее популярные системы мониторинга, а также их верхнеуровневую архитектуру

# Домашнее задание

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

нетология

# Задавайте вопросы и пишите отзыв о лекции!

Сергей Бывшев

Сергей Бывшев