ZABBIX

Monitoring

на примере Zabbix

Tinkoff.ru

Роман Николаев

О чем пойдет речь?

1. Что такое мониторинг?
2. Существующие системы мониторинга
3. Архитектура Zabbix
4. Zabbix Items
5. User Parameters
6. Zabbix Triggers
7. Zabbix Actions
8. Low-level Discovery
9. Zabbix + Grafana

1

Что такое мониторинг?

Мониторинг?

Основные задачи:

- Оперативный мониторинг (обнаружение и диагностика проблем)
- Анализ исторических данных («разбор полетов», планирование ресурсов, ...)

Типы метрик:

— Системные метрики:
CPU, RAM, Disk Space, I/O operations, ...

— Метрики приложения/сервиса:
наличие HTTP ошибок, доступность БД, скорость выполнения метода...

— Бизнес-метрики:
кол-во активных клиентов, суммарные показатели выручки, частота покупки товара, ...

2

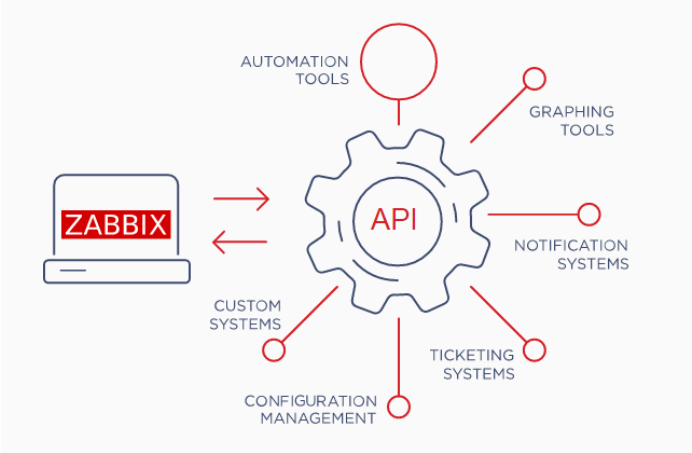
Существующие системы мониторинга

Существующие системы мониторинга

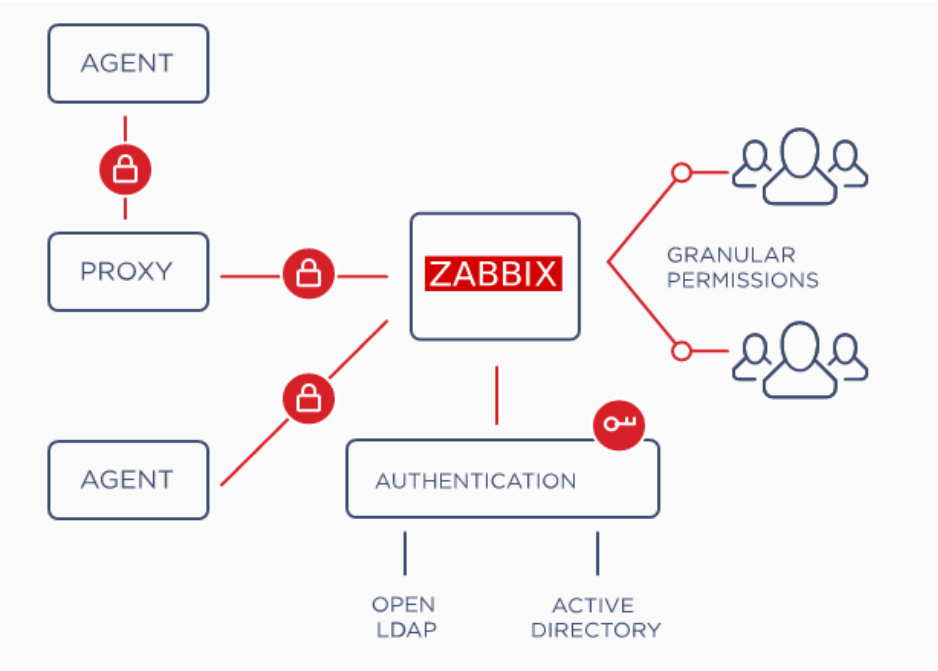
Nagios

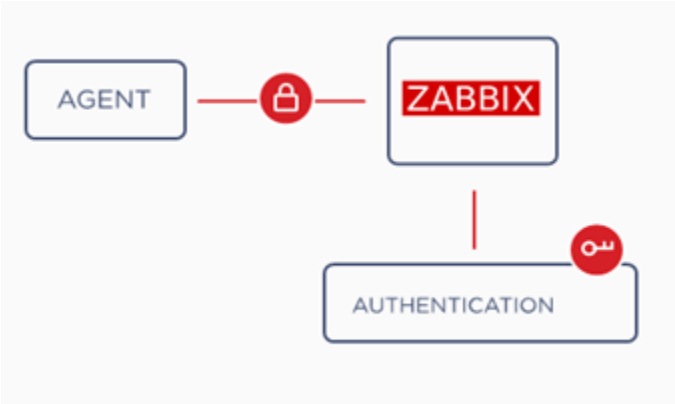
3

Архитектура Zabbix



Архитектура Zabbix



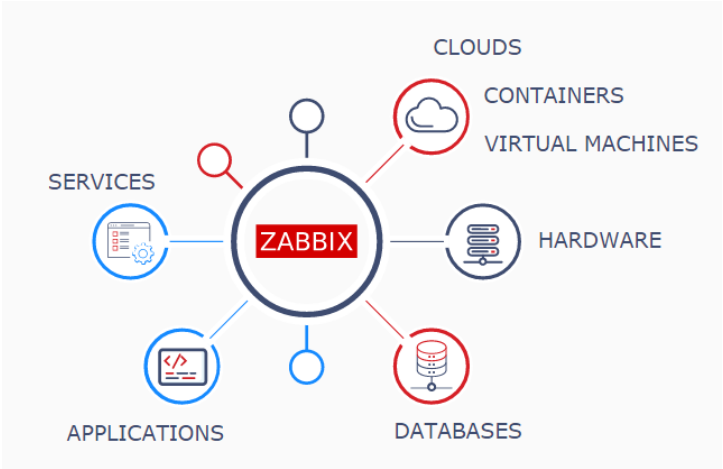


Ключевые особенности

- Frontend: веб-интерфейс на php
- Backend: реляционная БД (MySQL / PostgreSQL)
- Сбор метрик по моделям pull/push
- Добавление новых метрик через веб-интерфейс или API
- Возможность построить распределенную систему мониторинга
- Масштабирование системы возможно с помощью Zabbix Proxy
- Отказоустойчивость системы строится за счет возможностей соответствующей подсистемы (Linux HA-Cluster, Virtual IP, DB Replication)

4

Zabbix Items

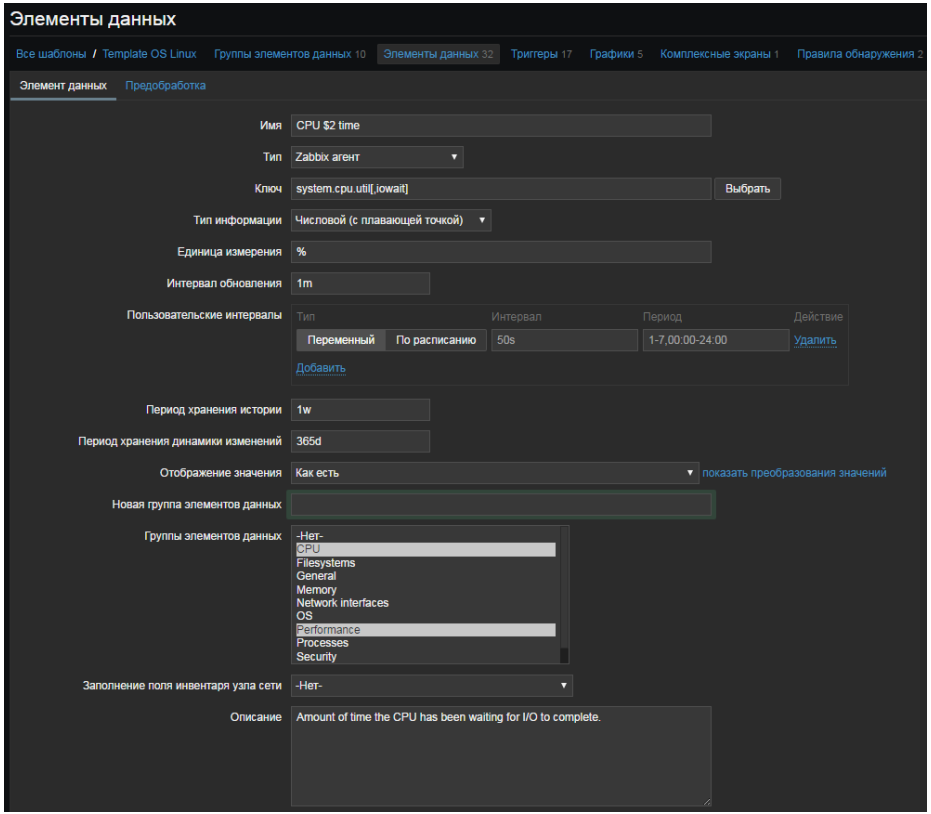


Zabbix Items (элементы данных)

Сущность, которая собирает данные (метрика).

Типы:

- Zabbix агент
- SNMP агент
- SNMP трапы
- IPMI проверки
- Простые проверки
- Журнал (лог)
- Вычисляемые элементы данных
- Внутренние проверки Zabbix
- SSH проверки
- Telnet проверки
- Внешние проверки (скрипты)
- Агрегированные проверки
- Траппер
- JMX
- ODBC
- Зависимые элементы данных



Zabbix Items (элементы данных)

Zabbix агент — для сбора данных используется Zabbix агент.

SNMP агент — опрос хостов по SNMP (протокол UDP, порт 161): сетевуха, сервера, принтеры...

SNMP трапы — «ловушка» для сообщений от SNMP-устройств (протокол UDP, порт 162).

IPMI проверки — Intelligent Platform Management Interface, позволяет наблюдать за состоянием аппаратного обеспечения напрямую с так называемых карт управления “out-of-band”, независимо от ОС или же от наличия питания на хосте (HP iLO, DELL DRAC, IBM RSA...).

Простые проверки — не требуется Zabbix агент, удаленные проверки выполняются с Zabbix сервера/прокси (доступность порта, ping).

Журнал (лог) — мониторинг логов а предмет появления ключевой фразы (точная строка / регулярка). Требуется Zabbix агент.

Вычисляемые элементы данных — подсчеты на основании других элементов данных (виртуальный источник данных).

Внутренние проверки Zabbix — наблюдение за внутренним процессами Zabbix (queue, items, hosts, requiredperformance, ...)

SSH проверки — результат выполнения произвольной команды/скрипта по ssh (логин + пароль / файл ключа), Zabbix агент **не** нужен.

Telnet проверки — результат выполнения произвольной команды/скрипта по telnet (логин + пароль), Zabbix агент **не** нужен.

Внешние проверки (скрипты) — выполнение произвольного скрипта (с параметрами) с Zabbix сервера/прокси, Zabbix агент **не** нужен.

Агрегированные проверки — совокупная инфа по элементам данных на основе прямых запросов в БД («температура по больнице»).

Траппер — отправка данных (метод “push”) с узла вместо запроса данных (метод “pull”). Например, с помощью zabbix_sender.

JMX — получение данных со счетчиков JMX (Java Management Extensions) из Java-приложений.

ODBC — запросы в БД напрямую с Zabbix сервера с помощью ODBC-драйвера (прослойка для работы с СУБД).

Зависимые элементы данных — основной элемент данных инициирует сбор зависимого элемента (когда нужна синхронность сбора).

5

User Parameters

{code}

User Parameters (пользовательские параметры)

Пользовательские параметры – возможность выполнять любые команды непосредственно на целевом узле, получая тем самым необходимые данные по метрикам.

Синтаксис:
```
UserParameter=<ключ>,<команда>
```

Пример:
```
UserParameter=mysql.ping,mysqladmin -uroot ping | grep -c alive
```

Агент будет возвращать '1', если MySQL сервер доступен, '0' - в противном случае.

Гибкие пользовательские параметры – допускают параметры с указанным ключем. В этом случае гибкие пользовательские параметры могут быть основой для создания нескольких элементов данных.

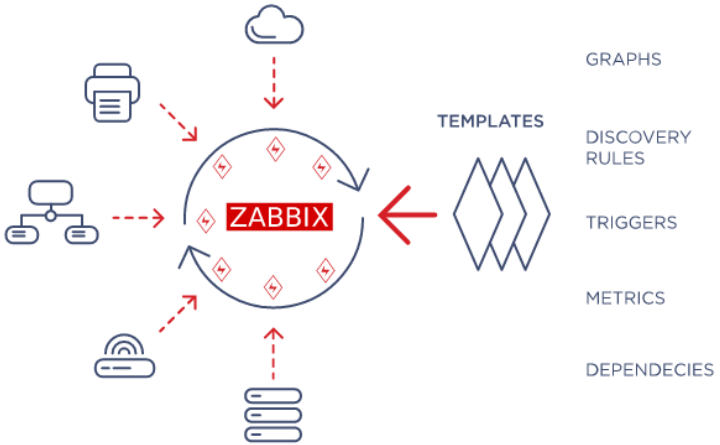
Синтаксис:
```
UserParameter=ключ[*],команда
```

Пример:
```
UserParameter=wc[*],grep -c "$2" $1
```

Этот параметр можно использовать для подсчета количества строк в файле.
```
wc[/etc/passwd,root]
wc[/etc/services,zabbix]
```

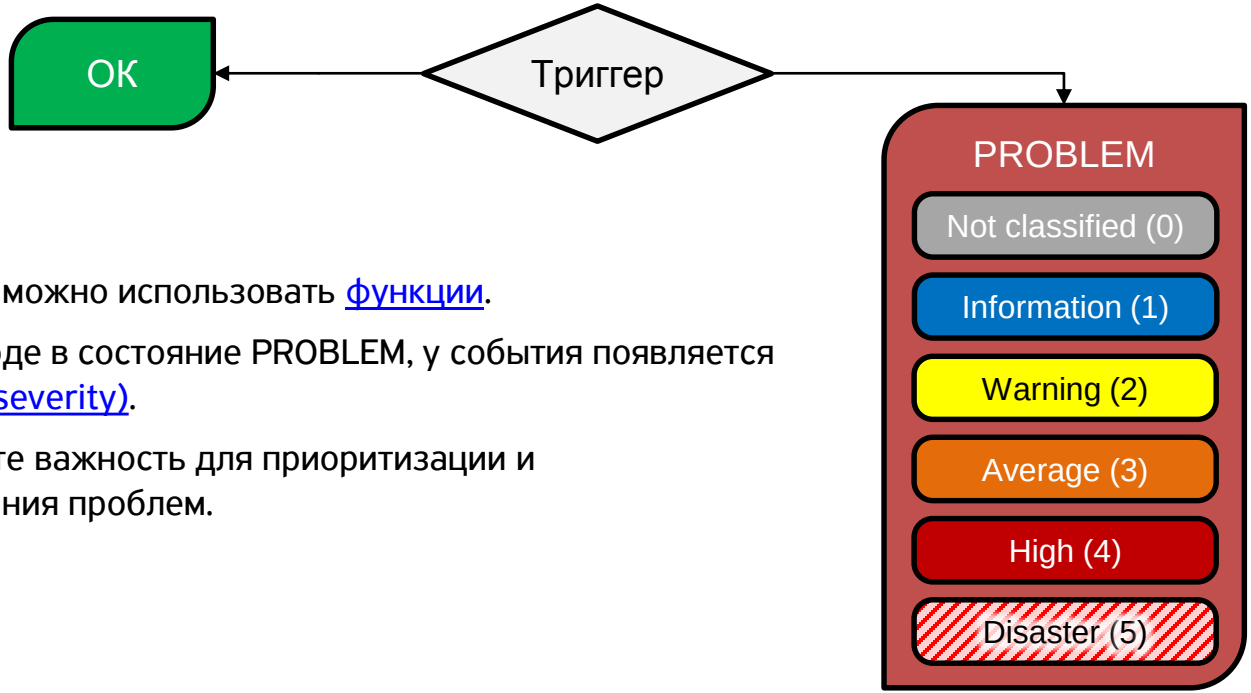
6

Zabbix Triggers



Zabbix Triggers (триггеры)

Триггер – логическое выражение, которое определяет какую-либо ситуацию/состояние.



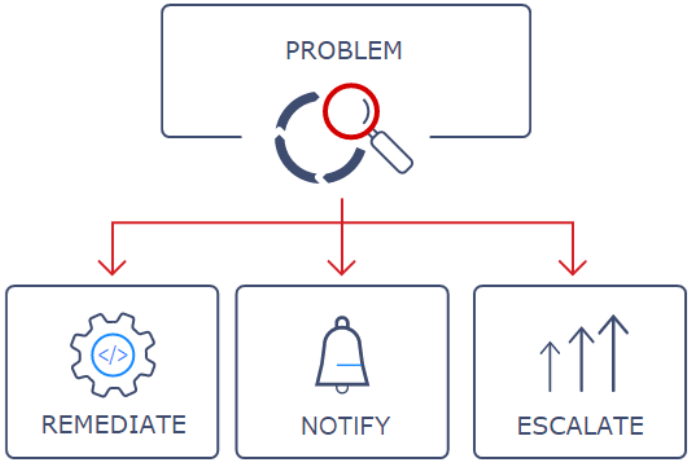
В триггере можно использовать функции.

При переходе в состояние PROBLEM, у события появляется важность (severity).

Используйте важность для приоритизации и эскалирования проблем.

7

Zabbix Actions



Zabbix Actions (действия)

Action (действия) – это реакция на какое-либо событие.

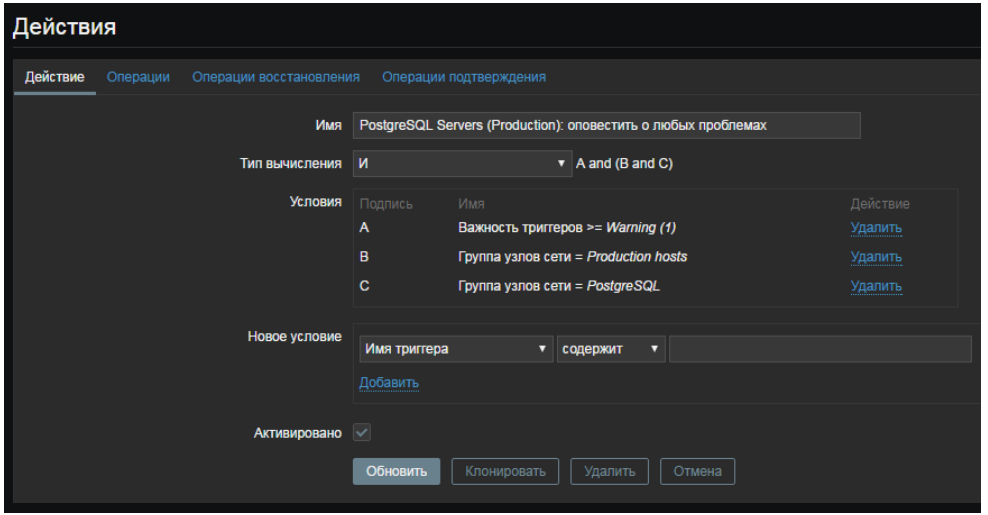
Triggers (Триггеры)
В основном используется для оповещения по сработавшим триггерам (alerts).

Auto Registration (Авторегистрация)
Удобная возможность автоматически добавлять узлы в мониторинг.

Network Discovery (Обнаружение)
Сканирование сети в указанном диапазоне и добавление найденных узлов в мониторинг.

Internal events (Внутренние события)
Возникают, когда:

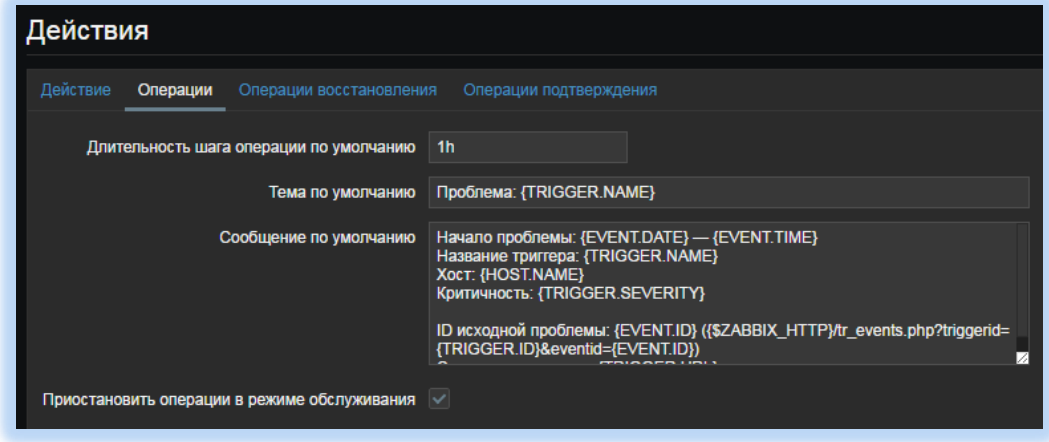
- элемент данных меняет свое состояние с “нормального” на “не поддерживается” или обратно
- правило низкоуровневого обнаружения меняет свое состояние с “нормального” на “не поддерживается” или обратно
- триггер меняет свое состояние с “нормального” на “неизвестное” и обратно

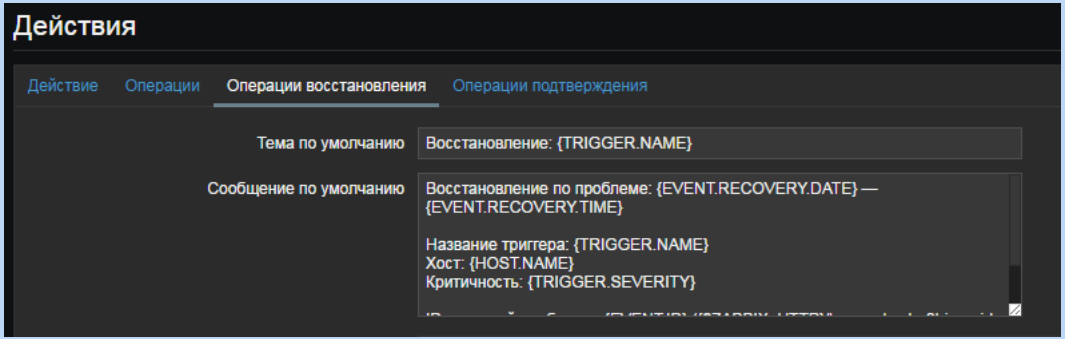


События в Zabbix генерируются несколькими источниками:

- события на триггеры – всякий раз, когда триггер меняет свое состояние (OK → ПРОБЛЕМА → OK)
- события на обнаружение – при обнаружении узлов сети или сервисов
- события на авторегистрацию – когда активные агенты автоматически регистрируются сервером
- внутренние события – когда элементы данных/правила низкоуровневого обнаружения становятся не поддерживаемыми или триггер переходит в состояние неизвестно.

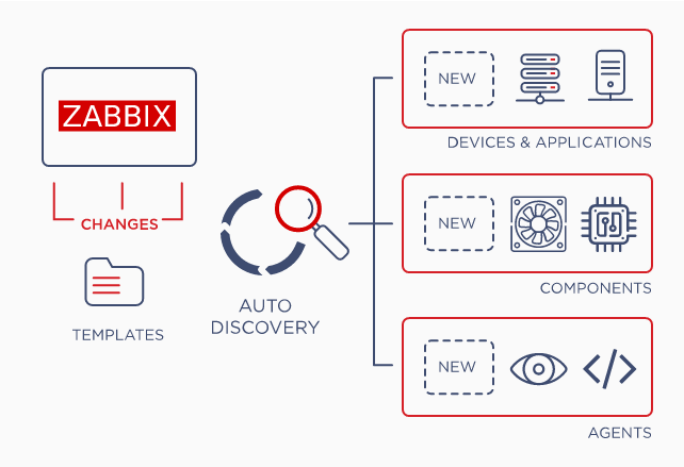
Zabbix Actions (действия)





8

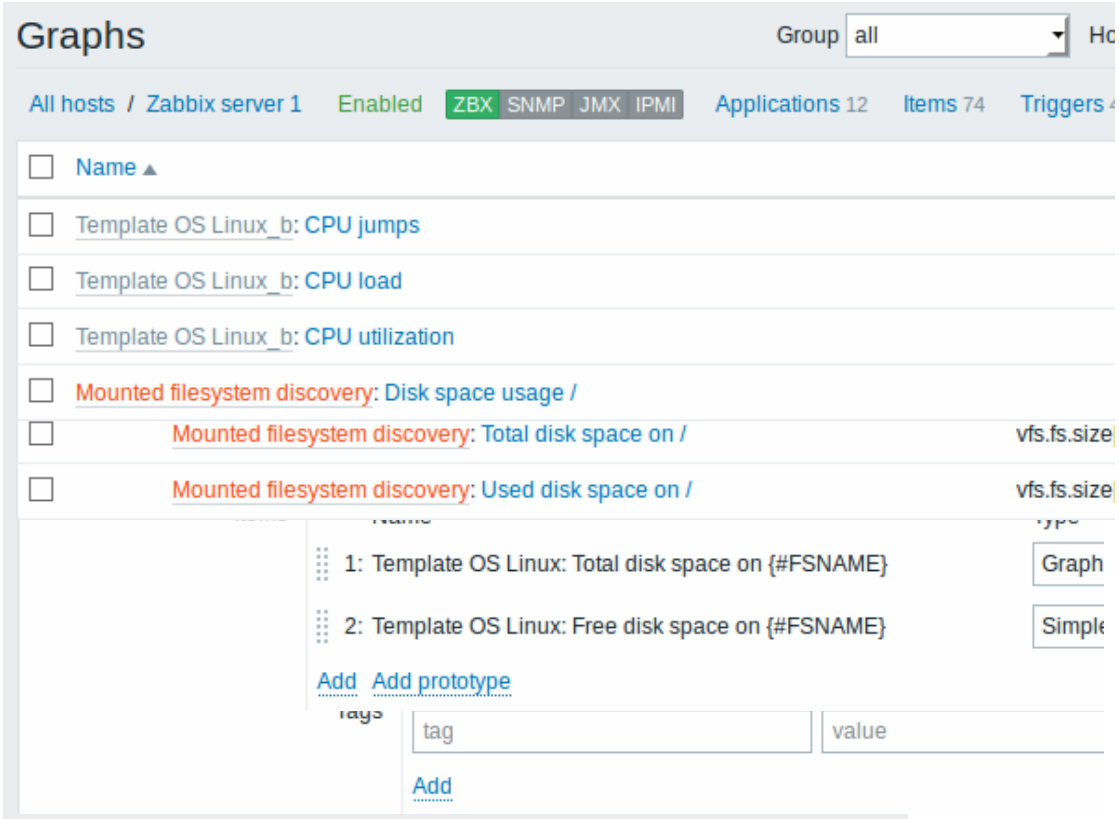
LLD


ZABBIX
CHANGES
TEMPLATES
AUTO
DISCOVERY
NEW
DEVICES & APPLICATIONS
NEW
COMPONENTS
NEW
AGENTS

Low Level Discovery (низкоуровневое обнаружение)

Низкоуровневое обнаружение – это автоматическое создание элементов данных, триггеров и графиков на основе правил обнаружения.

1. Создаем правило обнаружения
2. Создаем прототипы элементов данных
3. Создаем прототипы триггеров
4. Создаем прототипы графиков
5. Профит!



Low Level Discovery (низкоуровневое обнаружение)

Пример вывода кастомного правила низкоуровнего обнаружения (JSON):

```
{
  "data":[

  { "{#FSNAME}":"/",                          "{#FSTYPE}":"rootfs"   },
  { "{#FSNAME}":"/sys",                       "{#FSTYPE}":"sysfs"    },
  { "{#FSNAME}":"/proc",                      "{#FSTYPE}":"proc"     },
  { "{#FSNAME}":"/dev",                       "{#FSTYPE}":"devtmpfs" },
  { "{#FSNAME}":"/dev/pts",                   "{#FSTYPE}":"devpts"   },
  { "{#FSNAME}":"/lib/init/rw",               "{#FSTYPE}":"tmpfs"    },
  { "{#FSNAME}":"/dev/shm",                   "{#FSTYPE}":"tmpfs"    },
  { "{#FSNAME}":"/home",                      "{#FSTYPE}":"ext3"     },
  { "{#FSNAME}":"/tmp",                       "{#FSTYPE}":"ext3"     },
  { "{#FSNAME}":"/usr",                       "{#FSTYPE}":"ext3"     },
  { "{#FSNAME}":"/var",                       "{#FSTYPE}":"ext3"     },
  { "{#FSNAME}":"/sys/fs/fuse/connections",   "{#FSTYPE}":"fusectl"  }

  ]
}
```

9

ZABBIX + Grafana

Grafana: Zabbix Data Source

1. Переходим на http://grafana.fintech-admin.m1.tinkoff.cloud
2. Переключаемся на организацию: Main Org → Switch
3. Добавляем ваш Zabbix для сбора данных: Configuration -> Data Sources -> Add data source
4. Name: Zabbix, Type: Zabbix, URL: you-instance.fintech-admin.m1.tinkoff.cloud:8888
5. Save & Test

Домашнее задание

1. Установить Zabbix Server локально у себя на хосте
2. Установить Zabbix Agent локально у себя на хосте + зарегистрировать агента на Zabbix Server
3. Создать правило обнаружения (LLD): элементы данных + триггеры
4. Добавить кастомную метрику на агента + новый элемент данных + триггер
5. Настроить оповещение на триггеры (почта + Telegram)
6. Настроить дашборд в Grafana под системные метрики + добавить туда панель под кастомную метрику

Полезные ссылки / советы

- Официальная документация:
  https://www.zabbix.com/documentation/
- Best practices:
  https://www.zabbix.com/documentation/3.4/manual/installation/requirements/best_practices
- Планируйте вашу инсталляцию, особенно размер БД:
  https://www.zabbix.com/documentation/3.4/manual/installation/requirements#database_size
- Используйте прокси для масштабирования инсталляции.
- Не собирайте данные слишком часто (нагрузка на Zabbix сервер и БД), если этого не требуется.
- Разносите метрики и триггеры для различных систем/сервисов по шаблонам.
- Собирайте хосты в группы по системам и окружениям (MySQL servers + Production servers + MyServiceName + ...).
- Не храните исторические (неагрегированные) данные слишком долго, храните из в трендах (max/min/avg/count, агрегированные за час).
- Используйте теги событий.
- Дублируйте каналы доставки алертов (почта + СМС/Telegram), не допускайте большого количества «шума».

Спасибо за внимание!

Роман Николаев
Email: r.nikolaev@tinkoff.ru
Telegram: @Nikolaev_RD