Умный мониторинг

Установка и запуск системы мониторинга

Содержание

1 Список принятых сокращений

1.1 Сокращения

Наименование	Описание сокращения/аббревиатуры
АРМ	Автоматизированное рабочее место
БД	База данных
БС	Информационные системы – ПО для автоматизации бизнес-процессов компании на технологической платформе «1С:Предприятие 8»
ВМ	Виртуальная машина
ЕЦМ	Единый центр мониторинга окружений и системы сбора и анализа журналов
ЖР	Журнал регистрации
ИБ	Информационная база
КЕ	Конфигурационная единица, виртуальная машина с встроенным агентом, обеспечивающая поток данных (ТЖ 1С, СЖС) для проверки функциональности системы мониторинга
Актуатор	Скрипт (или иное запрограммированное заранее действие), выполняемый на КЕ с целью реализации какого-либо воздействия на возникшую там ситуацию
НТ	Нагрузочное тестирование
ОИБ	Отдел информационной безопасности
ОС	Операционная система
ПМИ	Программа и методика испытаний
ПО	Программное обеспечение
ПЭ	Подэтап
СЖС	Системный журнал событий
СУБД	Система управления базами данных
ТЖ 1С	Технологический журнал 1С
ТЗ	Техническое задание
ЦУ	Центр управления
CMDB	База данных управления конфигурации (Configuration management database)
ELK	Elasticsearch, Kibana
Панель	Сущность представления Grafana. Базовый блок визуализации.
Дашборд	Сущность представления Grafana (по-английски – Dashboard). Логическая группировка наборов панелей, как правило, по определенным критериям

2 Введение

2.1 Назначение документа

Описание программного продукта «Умный мониторинг» и порядка его развертывания (установки).

Развертывание серверного уровня комплекса (транспортный слой, СУБД и т.п.) без Центра управления осуществляется с помощью сценария Ansible для Astra Linux (поддерживаются варианты «Орел»; «Смоленск», версии ядра 4.15.3-1-generic и 4.15.3-141-generic). Центр управления устанавливается отдельно.

Операционная система на КЕ: для примера взята Microsoft Windows 10.

Для других операционных систем потребуется адаптация сценариев Ansible под целевые операционные системы или ручная установка используемых компонент.

3 Общие положения

3.1 Описание программного решения

Решение «**Умный мониторинг**» базируется на идеях AIOps (Artificial Intelligence for IT Operations) платформе для автоматизированной систематизации и обработки больших данных с целью повышения продуктивности ИТ-службы организации.

Реализует три основных направления:

- Мониторинг – сбор и хранение данных
- Анализ и Визуализация данных
- Автоматизация реагирования на инциденты

Концептуальная схема решения, представленная на рис. 1, состоит, по сути, из трех блоков: Транспортный слой, Обработка и хранение данных, Центр управления.

Транспортный слой реализован на базе java-приложения Apache Kafka Confluent распределённой, горизонтально масштабируемой системы, обеспечивающей наращивание пропускной способности как при росте числа и нагрузки со стороны источников, так и количества систем-подписчиков.

Обработка и хранение данных реализованы на базе компонент с открытым исходным кодом FluentD, TimescaleDB (PostgreSQL), Elasticsearch, что позволяет обрабатывать данные/журналы из разных источников, хранить множество данных и осуществлять по ним поиск.

Центр управления реализован на базе Grafana, Kibana, Git и 1С, позволяет отображать сведения о конфигурационных единицах и осуществлять управляющее воздействие через систему актуаторов.

Схема демонстрирует сбор данных с конфигурационных единиц (КЕ), которые представляют из себя ВМ с развернутым системным и прикладным ПО, а также встроенным агентом влияния (Java-агент) – неотъемлемой частью системы «Умный мониторинг».

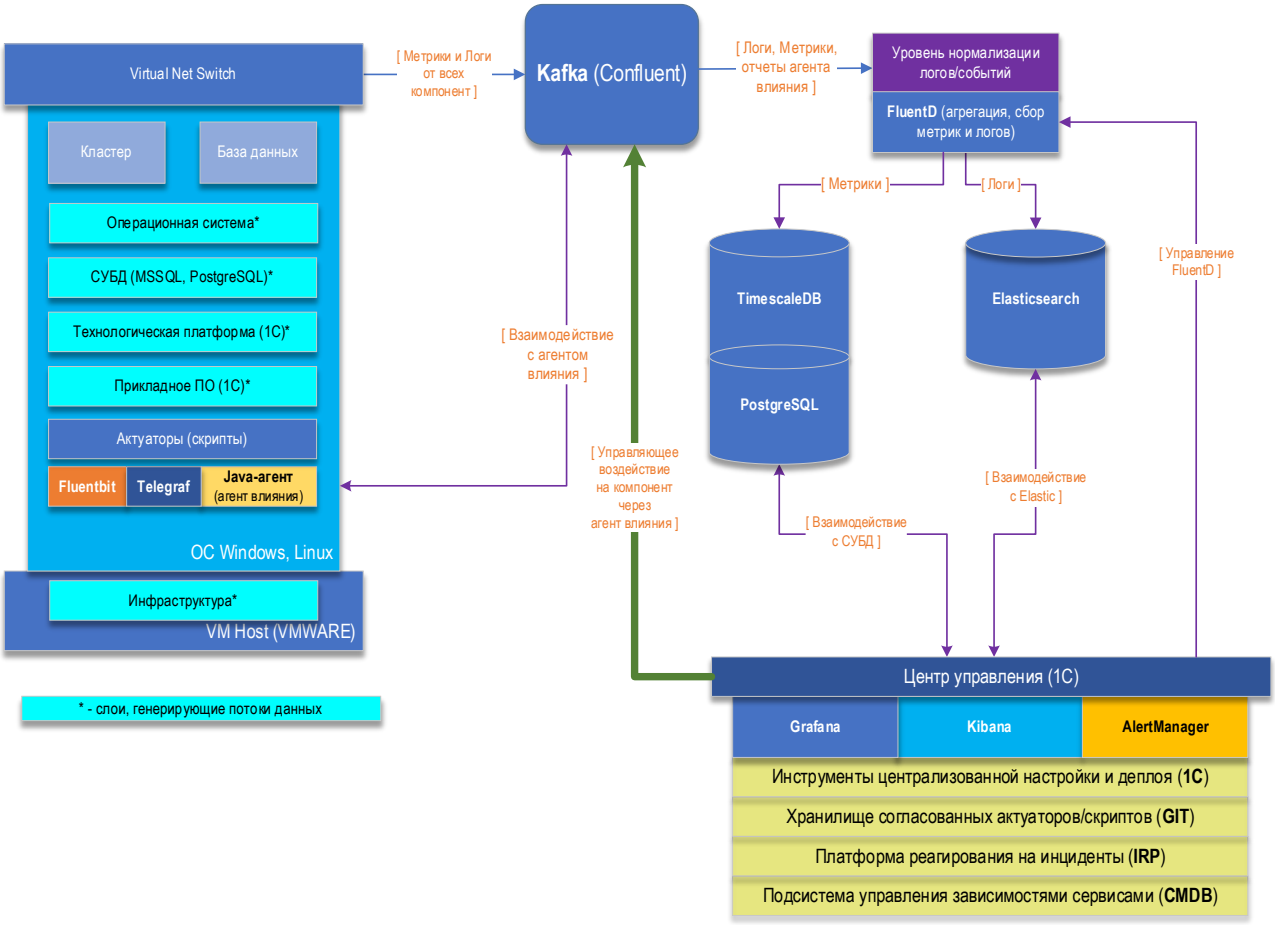


Рисунок 1 – Концептуальная схема решения

3.2 Предварительные условия развертывания решения

Перед запуском «Умного мониторинга» требуется подготовить серверный и клиентский уровни системы. Все программное обеспечение устанавливается из заранее полученных актуальных (если не указано иное) дистрибутивов. Для удобства настройки рекомендуется установить пакет GNU Midnight Commander (mc) на узлы серверного уровня, либо аналог. Все команды следует выполнять от имени суперпользователя. После установки возможно потребуется ручное изменение конфигурационных файлов под нужды системы (например, для расшифровки журнала регистраций требуется вручную указать пути в фильтрах по предоставленному образцу по количеству первоначальных КЕ).

3.3 Подготовка узлов серверного уровня

Для всех узлов серверного уровня требуется открыть доступ по протоколу SSH (TCP/22). Подключение используется для удаленного управления и установки компонент умного мониторинга с помощью Ansible, авторизация может выполняться по паре имя пользователя-пароль или по ключу (сертификату). Способ зависит от политики безопасности, принятой в организации.

3.3.1 Предварительная подготовка к установке серверного слоя

Для установки серверного уровня «Умного мониторинга» используется программное обеспечение Ansible, поэтому требуется провести первоначальную настройку для обеспечения работы установщика:

1. Установить на каждый узел серверного уровня python версии 2.7 или 3.6.
   apt install python2.7

Если в ходе запуска Ansible будет выдана ошибка об отсутствии python нужной версии на целевом узле, то значит необходимо поставить пакет python командой **apt install python**.

2. Создать на каждом узле контура пользователя с правами sudo, с помощью которого будет происходить подключение Ansible к целевому узлу и производиться установка и настройка.
   adduser username
   passwd username
   usermod -aG astra-admin,astra-console username

На ОС Astralinux SE «Смоленск» так же требуется выдать соответствующий уровень мандатного доступа созданному пользователю командой **pdpl-user -i 63 username**.

3. Скопировать дистрибутив на узел, где будет устанавливаться Ansible (например, в домашнюю папку пользователя user).
4. Установить на любой узел, имеющий доступ к контуру, Ansible. Наиболее простым решением в данном случае будет использование одного из узлов, входящих в контур, кроме узла для fluentd (связано с особенностями установки). Для удобства пользователя в комплект поставки включены нужные пакеты, однако можно использовать и свои пакеты. **Важно**: требуемая версия Ansible 2.9+.
   dpkg -i sshpass_1.06-1_amd64.deb
   dpkg -i ansible_2.9.9-1ppa~trusty_all.deb

5. Сделать резервную копию исходного конфигурационного файла ansible и заменить его на поставляемый с дистрибутивом.
   mv /etc/ansible/ansible.cfg /etc/ansible/ansible.cfg.bak
   cp /home/user/ansible/ansible.cfg /etc/ansible/ansible.cfg

3.3.2 Подготовка сценариев Ansible и установка

В составе дистрибутива включено два инвентаря ansible:

- Staging - для тестового развертывания
- Production - для полноценного развертывания

Сделано это для удобства развертывания (дает возможность разворачивать тестовый контур и продуктивный из одного дистрибутива простой сменой входного инвентарного файла), фактической разницы после установки не будет. Для установки надо заполнить следующие данные в одном из инвентарей (для примера будем использовать инвентарь staging, все пути написаны относительно файла install.yml):

1. Файл inventories/staging/hosts.

В данном файле требуется указать следующие переменные:

- IP адреса узлов вместо «заглушек» для переменных ansible_host (узлы разбиты на группы: kafkaservers – для серверов kafka; dbservers – для серверов Timescaledb; elasticservers – для серверов elasticsearch; monitorservers – для центра управления, Grafana и Kibana; fluentdservers – для серверов FluentD);
- ansible_user – логин пользователя, который создавался в разделе 3.3.1
- ansible_password – пароль пользователя, который создавался в разделе 3.3.1
- ansible_become_password– пароль пользователя, который создавался в разделе 3.3.1

2. Файл inventories/staging/group_vars/all.yml

В данном файле требуется указать следующие переменные:

- IP адрес узла Elasticsearch
- IP адрес узла TimescaleDB
- fluent_dbpassword– пароль, используемый пользователем базы данных fluent, отправляющим метрики в TimescaleDB из FluentD. Данный пользователь будет создан при установке серверного слоя.
- telegraf_dbpassword– пароль, используемый пользователем базы данных telegraf, отправляющим метрики в TimescaleDB от Telegraf серверного уровня. Данный пользователь будет создан при установке серверного слоя.

3. Файл inventories/staging/group_vars/dbservers.yml

В данном файле требуется указать следующие переменные:

- tsadmin_dbpassword– пароль, используемый пользователем базы данных tsadmin, данный пользователь является суперпользователем, необходим на случай кластеризации TimescaleDB. Данный пользователь будет создан при установке серверного слоя.
- grafanareader_dbpassword– пароль, используемый пользователем базы данных grafanareader, читающим данные для Grafana. Данный пользователь будет создан при установке серверного слоя.

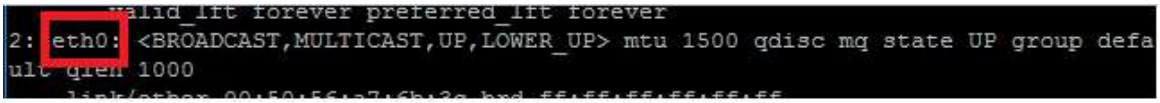
4. Файл inventories/staging/group_vars/fluentdservers.yml

В данном файле требуется указать следующие переменные:

- IP адрес узла Kafka

5. Файл inventories/staging/host_vars/*.yml

Во всех файлах внутри папки необходимо указать название сетевого интерфейса, для которого будут собираться метрики. Чтобы узнать название сетевого интерфейса необходимо воспользоваться командой **ip a** на целевом узле.



После настройки инвентаря необходимо настроить роли (набор действий и объектов, выполняющий определенное действие, например, установку Kafka). В

обязательном порядке требуется указать **имя узла Kafka** в файле **roles/fluentd/vars/main.yml**. IP адрес сервера git указывается в **roles/git/vars/main.yml**. Там же можно указать пароль пользователя git. По умолчанию git ставится в режиме сервера, если этого не требуется, то необходимо изменить переменную git_server в **roles/git/vars/main.yml**. Остальные роли настраиваются по необходимости, для этого требуется раскомментировать в соответствующей роли в файле roles/<название роли>/vars/main.yml нужную переменную и изменить ее значение.

После внесения всех необходимых изменений желательно выполнить команду **ansible -i /home/user/sm/inventories/staging/hosts -m ping all**, чтобы убедиться, что инвентарь настроен верно. Если ошибок не возникло, то установка выполняется командой **ansible-playbook -i /home/user/ansible/inventories/staging/hosts /home/user/sm/install.yml**

3.3.3 Дополнительные действия после установки

После установки через Ansible требуются дополнительные действия для обеспечения полной работы «Умного мониторинга» (сразу после установки обрабатываются только журнал событий Windows, технический журнал 1С и журнал регистраций без расшифровки). В начале каждого пункта указано на каком узле требуется производить действия.

1. **Узел TimescaleDB**. В базе metrics требуется создать гипертаблицы для хранения метрик. В приложении А приведен пример команд для создания таблиц согласно настройкам Telegraf для КЕ (приложение Б).
2. **Узел FluentD**. После создания таблиц, необходимо раскомментировать блоки source, filter и match, отвечающие за пересылку метрик (для ориентирования следует пользоваться лейблом METRICS и METRIC1C), в настроечном файле /etc/fluentd/fluent.conf (настройки обработки соответствуют настройкам Telegraf из Приложения Б).
3. **Узел FluentD**. Для обеспечения расшифровки журнала регистраций необходимо скопировать с КЕ словарь в формате lgf и преобразовать его в json файл с помощью скрипта lgf-to-json.sh, который устанавливается в папку **/etc/fluentd/dictionaries** во время инсталляции fluentd**.** Далее полученные файлы необходимо поместить в папку по имени КЕ и внести изменения в /etc/fluentd/fluent.conf: в нем необходимо найти блоки <filter jr.example> и, скопировав их, актуализировать под целевой КЕ. Например, пусть КЕ имеет имя host1, тогда блоки надо переписать следующим образом (изменения выделены полужирным):

```
<filter jr.host1>
  @type dict_map
  @id "filter_dict_map_userid1##{worker_id}"
  key_name UserId
  dictionary_path /etc/fluentd/dictionaries/host1/UserId.json
</filter>
```

Остальные блоки изменяются аналогично. Данную процедуру требуется повторить для каждой КЕ, с которой предполагается снимать журнал регистраций. Важно отметить, что поле @id должно быть уникальным, таким образом, если узлов несколько (например, имеется host2), то для другого узла блок будет выглядеть так:

```
<filter jr.host2>
  @type dict_map
  @id "filter_dict_map_userid2##{worker_id}"
  key_name UserId
  dictionary_path /etc/fluentd/dictionaries/host2/UserId.json
</filter>
```

4. **Узел TimescaleDB и FluentD**. Если предполагается сбор Apdex, то в базе metrics требуется создать гипертаблицу apdex (см. Приложение А) и раскомментировать блоки source, filter и match, отвечающие за обработку Apdex, в настроечном файле /etc/fluentd/fluent.conf.
5. **Узел TimescaleDB и FluentD**. Если предполагается сбор метрик виртуальной машины VMWare, то необходимо создать гипертаблицу таблицу metric_vmware (см. Приложение А) в базе metrics и раскомментировать блоки source, filter и match, отвечающие за обработку метрик виртуальной машины, в настроечном файле /etc/fluentd/fluent.conf.
6. **Узел FluentD**. После всех необходимых изменений конфигурационного файла fluentd требуется выполнить его перезагрузку командами
   sudo systemctl stop fluentd
   sudo systemctl start fluentd
7. **Узел Центра управления**. В Kibana необходимо импортировать файлы 1c.ndjson и winlog.ndjson из папки **for elastic** в разделе Stack Management->Saved Objects главного меню (открывается по кнопке ☰). А так же в разделе Stack Management->Index Management во вкладке Index Templates создать шаблоны индексов технического журнала и журнала регистраций (Приложение В) для индексов fluentd_tj1c* и fluentd_jr* соответственно.
8. **Узел Центра управления и elasticsearch**. При необходимости включения системы аутентификации на elasticsearch необходимо следовать официальной инструкции (Ссылка для версии 7.13: https://www.elastic.co/guide/en/elasticsearch/reference/7.13/security-minimal-setup.html)
9. **Узел Центра управления**. В Grafana необходимо указать в настройках два источника: тип Postgresql, IP адрес TimescaleDB, пользователь для подключения grafanareader, базы для подключения metrics и sysmetrics. В случае, если SSL не используется (это вариант по умолчанию), то необходимо выставить SSL Mode в disable. В составе дистрибутива так же поставляется набор дашбордов (с отображением информации серверного слоя, а также примеры возможных дашбордов для КЕ), которые при желании можно импортировать. После импорта необходимо актуализировать дашборды в части имен узлов (непосредственно в самих графиках и в разделе Variables). Пароль и логин для доступа к Grafana по умолчанию admin/admin.
   Примечание: если возникает ошибка доступа к таблицам при просмотре дашборда, то необходимо на узле **TimescaleDB** ввести следующие команды:
   echo "GRANT CONNECT ON DATABASE sysmetrics TO grafanareader;GRANT USAGE ON SCHEMA public TO grafanareader;GRANT SELECT ON ALL TABLES IN SCHEMA public TO grafanareader;" | sudo -u postgres psql -d sysmetrics

```
echo "GRANT CONNECT ON DATABASE metrics TO grafanareader;GRANT USAGE ON SCHEMA public TO grafanareader;GRANT SELECT ON ALL TABLES IN SCHEMA public TO grafanareader;" | sudo -u postgres psql -d metrics
```

3.4 Подготовка КЕ

Следующие действия надо производить на каждой КЕ, подключенной к системе мониторинга.

3.4.1 Установка и настройка Telegraf

Для установки Telegraf необходимо:

1. Распаковать архив с telegraf в каталог C:\Program Files\InfluxData\telegraf\
2. Настроить конфигурационный файл Telegraf (Приложение Б). Обязательно требуется указать IP адрес сервера Kafka.
3. Создать сервис командой telegraf.exe --service install --config "C:\Program Files\InfluxData\telegraf\telegraf.conf".
4. Запустить службу telegraf.

3.4.2 Установка и настройка Fluentbit

На КЕ используется версия 1.8.10 с поддержкой вывода информации в Kafka. Fluentbit занимается сбором логов и отправкой их в транспортный слой.

1. Создать папку C:\Program Files\FluentBit
2. Распаковать архив fluentbit-1.8.10-win64.zip в созданную папку
3. Настроить конфигурационный файл FluentBit, указав IP адреса Kafka и путь до журнала регистраций. (Приложение Г)
4. В файле hosts (расположение файла в Windows – C:\Windows\System32\drivers\etc) указать имя узла Kafka и его IP, например

   192.168.1.1 kafka
5. Создать сервис командой

   ```
   sc.exe create fluent-bit binpath= "C:\Program Files\FluentBit\bin\fluent-bit.exe" -c "C:\Program Files\FluentBit\conf\fluent-bit.conf"
   ```
6. Запустить созданную службу fluent-bit

3.4.3 Установка и настройка агента влияния (java-агента)

Для работы агента требуется установить Java не ниже версии 17.

1. Создать папку C:\SM\ite-agent\
2. Скопировать содержимое папки дистрибутива agent в созданный каталог.

3. Создать папку C:\SM\ite-agent\datafiles
4. Запустить агент через bat файл. Содержание файла ite_mon.bat:

   cd c:\sm\ite-agent

   java -jar ite_smagent-full.jar .\datafiles

3.5 Подготовка ЦУ

Для установки ЦУ требуется скачать дистрибутив 1С для Linux. ЦУ устанавливается на ту же машину, куда разворачивались Grafana и Kibana. Для корректной работы приложения должна быть установлена любая графическая оболочка (для Astralinux это оболочка fly, можно установить с помощью пакета fly-all-main). Кроме того, следует убедиться, что в системе присутствуют пакеты libunwind и geoclue-2.0.

3.5.1 Установка и настройка 1С для ЦУ

1. Запустить установку 1С:Предприятия от имени суперпользователя. Внимание! Требуется версия технологической платформы 1С:Предприятия не ниже 8.3.14, рекомендуется использовать 8.3.19.
2. Во время установки включить все флажки по всем компонентам, за исключением серверной части. После установки убедится, что в процессе установки не было ошибок на нехватку каких-либо пакетов. Если такие ошибки были, то требуется переустановить 1с заново, предварительно установив вручную отсутствующие зависимости.
3. Переписать в каталог пользователя содержимое папки control center
4. Запустить от имени пользователя, который будет работать с ЦУ, скрипт base_importer.sh со следующими параметрами:
   a. Имя группы (будет создана, если отсутствует), в которую необходимо помещать всех пользователей, работающих с ЦУ.
   b. Каталог, в который будет импортирована база (будет создан, если отсутствует).
   c. Полный путь до бинарного файла 1cestart (включая само имя файла).
5. Запустить 1С.
6. В стартовом окне нажать «Добавить» и выбрать пункт «Добавление в список существующей информационной базы».
7. Название конфигурации указать произвольное, каталог базы указать такой же, как на шаге 4. Нажать «Далее» и «Готово».

Запускать ЦУ, следует выбирая созданную конфигурацию в п.7 стартового окна 1С. Все последующие настройки мониторинга вашей ИТ-инфраструктуры осуществляются непосредственно через ЦУ.

4 Материально-техническое обеспечение системы «Умного мониторинга»

Таблица 1 Прикладное ПО, на котором тестирование системы «Умного мониторинга» было завершено успешно

Прикладное ПО	Версия, релиз	Назначение	Примечание
Java JDK	17+	Java JDK	
1С:Предприятие 32-бита или 64-бита	8.3.19+	Обеспечение работоспособности Центра управления	x64 и x32 одной версии
Google Chrome	Последняя версия	Веб-интерфейс компонент стека	
ite_smagent_full.jar	Актуальная версия	Агент мониторинга	Выполняемое ПО в JVM
Influx telegraf	1.8+	Агент сбора метрик Telegraf	
fluent-bit	1.8.10+	Агент сбора логов FluentBit	Специальная сборка с включенной поддержкой Kafka
Far manager	Последняя версия	Файловый менеджер	Позволяет читать файлы любого размера без требований к памяти. Важен для анализа ТЖ
Offset Explorer	2.1	GUI для Kafka	Визуальный анализ топиков брокера
PgAdmin	5.6	GUI СУБД Postgres (Timescale)	Управление и анализ СУБД из GUI
Java jre (linux)	8u232+	Java jre	Требуется для работы kafka и elasticsearch
Kafka (linux)	6.1.2+	Брокер сообщений	Используется сборка Confluent community edition
build-essential (linux)	последняя версия для astralinux	Набор пакетов для сборки и разработки	Требуется для сборки ruby
libssl-dev (linux)	последняя версия для astralinux	Библиотека ssl	Требуется для сборки ruby
zlib1g-dev (linux)	последняя версия для astralinux	Библиотека сжатия	Требуется для сборки ruby
libpq-dev (linux)	последняя версия для astralinux	Библиотека для работы с СУБД Postgres (Timescale)	Требуется для плагина fluentd sql
ruby (linux)	2.7.4	Интерпретатор языка Ruby	Требуется для работы fluentd
FluentD (linux)	1.14.0+	Сборщик данных	Требует для своей работы ruby

PostgreSQL (linux)	13+	СУБД	Минорная версия последняя, требуется для работы timescaledb
Timescaledb (linux)	2.3.0+	Фреймворк для работы с временными рядами	
Elasticsearch (linux)	7.13.2+	Масштабируемая утилита полнотекстового поиска и аналитики	Требует для своей работы Java
Kibana (linux)	7.13.2+	Панель визуализации данных из Elasticsearch	
Grafana (linux)	8.0.4+	Веб-приложение для аналитики и интерактивной визуализации	
Telegraf (linux)	1.13.0	Агент сбора метрик Telegraf	Специальная сборка с включенной поддержкой PostgreSQL

Приложение А
Пример команд для создания таблиц в СУБД

```
CREATE TABLE public.metric_cpu (
    "timestamp" bigint NOT NULL,
    name text,
    host text,
    instance text,
    objectname text,
    "Percent_Interrupt_Time" real,
    "Percent_Privileged_Time" real,
    "Percent_User_Time" real,
    "Percent_Processor_Time" real
);
ALTER TABLE public.metric_cpu OWNER TO fluent;
SELECT create_hypertable('metric_cpu', 'timestamp', chunk_time_interval => 864000);

CREATE TABLE public.metric_memory (
    "timestamp" bigint NOT NULL,
    name text,
    host text,
    instance text,
    objectname text,
    "Available_Bytes" bigint,
    "Cache_Bytes" bigint,
    "Committed_Bytes" bigint,
    "Free_&_Zero_Page_List_Bytes" bigint,
    "Page_Faults_persec" real,
    "Pages_persec" real,
    "Percent_Usage" real
);
ALTER TABLE public.metric_memory OWNER TO fluent;
SELECT create_hypertable('metric_memory', 'timestamp', chunk_time_interval =>
864000);

CREATE TABLE public.metric_system (
    "timestamp" bigint NOT NULL,
    name text,
    host text,
    instance text,
    objectname text,
    "Processor_Queue_Length" bigint,
    "Threads" bigint,
    "Context_Switches_persec" real
);
ALTER TABLE public.metric_system OWNER TO fluent;
```

```
SELECT create_hypertable('metric_system', 'timestamp', chunk_time_interval => 864000);

CREATE TABLE public.metric_network (
    "timestamp" bigint NOT NULL,
    name text,
    host text,
    instance text,
    objectname text,
    "Bytes_Total_persec" bigint,
    "Output_Queue_Length" bigint
);
ALTER TABLE public.metric_network OWNER TO fluent;
SELECT create_hypertable('metric_network', 'timestamp', chunk_time_interval => 864000);

CREATE TABLE public.metric_disk (
    "timestamp" bigint NOT NULL,
    name text,
    host text,
    instance text,
    objectname text,
    "Percent_Free_Space" real,
    "Free_Megabytes" bigint,
    "Avg._Disk_Read_Queue_Length" bigint,
    "Avg._Disk_Write_Queue_Length" bigint,
    "Avg._Disk_sec/Read" real,
    "Avg._Disk_sec/Write" real,
    "Avg._Disk_Bytes/Read" real,
    "Avg._Disk_Bytes/Write" real
);
ALTER TABLE public.metric_disk OWNER TO fluent;
SELECT create_hypertable('metric_disk', 'timestamp', chunk_time_interval => 864000);

CREATE TABLE public.metric_mssql (
    "timestamp" bigint NOT NULL,
    name text,
    host text,
    instance text,
    objectname text,
    "Percent_Processor_Time" real,
    "Private_Bytes" bigint,
    "Virtual_Bytes" bigint,
    "ID_Process" bigint,
    "Thread_Count" bigint,
    "Full_Scans_persec" real,
    "Table_Lock_Escalations_persec" real,
    "Buffer_cache_hit_ratio" bigint,
```

"Free_List_Stalls_persec" real,
"Free_Pages" bigint,
"Lazy_Writes_persec" real,
"Page_life_expectancy" bigint,
"Data_Files_Size_KB" bigint,
"Log_Files_Size_KB" bigint,
"Log_Files_Used_Size_KB" bigint,
"Transactions_persec" real,
"Active_Transactions" bigint,
"Active_Temp_Tables" bigint,
"Temp_Tables_Creation_Rate" bigint,
"Temp_Tables_For_Destruction" bigint,
"Processes_blocked" bigint,
"Lock_Timeouts_timeout_>_0_persec" real,
"Number_of_Deadlocks_persec" real,
"Average_Wait_Time_ms" bigint,
"Lock_Requests_persec" real,
"Lock_Timeouts_persec" real,
"Lock_Wait_Time_ms" bigint,
"Database_Cache_Memory_KB" bigint,
"Free_Memory_KB" bigint,
"Granted_Workspace_Memory_KB" bigint,
"Lock_Blocks" bigint,
"Lock_Memory_KB" bigint,
"Memory_Grants_Outstanding" bigint,
"Memory_Grants_Pending" bigint,
"Target_Server_Memory_KB" bigint,
"Total_Server_Memory_KB" bigint,
"Cache_Hit_Ratio" bigint,
"Auto-Param_Attempts_persec" real,
"Batch_Requests_persec" real,
"Failed_Auto-params_persec" real,
"SQL_Compilations_persec" real,
"SQL_Re-Compilations_persec" real,
"Free_Space_in_tempdb_KB" bigint,
"Longest_Transaction_Running_Time" bigint,
"Transactions" bigint,
"Lock_waits" bigint,
"Log_buffer_waits" bigint,
"Log_write_waits" bigint,
"Memory_grant_queue_waits" bigint,
"Network_IO_waits" bigint,
"Non-Page_latch_waits" bigint,
"Page_IO_latch_waits" bigint,
"Page_latch_waits" bigint,
"Thread-safe_memory_objects_waits" bigint,
"Transaction_ownership_waits" bigint,
"Number_of_SuperLatches" bigint,

```
    "Total_Latch_Wait_Time_ms" bigint,
    "Latch_Waits_persec" real,
    "SuperLatch_Promotions_persec" real,
    "SuperLatch_Demotions_persec" real,
    "Average_Latch_Wait_Time_ms" real
);
ALTER TABLE public.metric_mssql OWNER TO fluent;
SELECT create_hypertable('metric_mssql', 'timestamp', chunk_time_interval => 864000);

CREATE TABLE public.metric_proc1c (
    "timestamp" bigint NOT NULL,
    name text,
    host text,
    instance text,
    objectname text,
    "Percent_Processor_Time" real,
    "Private_Bytes" bigint,
    "Virtual_Bytes" bigint,
    "ID_Process" bigint,
    "Thread_Count" bigint
);
ALTER TABLE public.metric_proc1c OWNER TO fluent;
SELECT create_hypertable('metric_proc1c', 'timestamp', chunk_time_interval => 864000);

CREATE TABLE public.metric_1c (
    "timestamp" bigint NOT NULL,
    name text,
    host text,
    "BlockedByDbms" real,
    "LongestCall" real,
    "BlockedByLs" real,
    "AppID 1CV8" real,
    "AppID 1CV8C" real,
    "AppID BackgroundJob" real,
    "AppID Designer" real,
    "AppID COMConsole" real,
    "AppID HTTPServiceConnection" real,
    "AppID JobScheduler" real,
    "AppID RAS" real,
    "AppID SrvrConsole" real,
    "AppID WSConnection" real,
    "Infobase" real
);
ALTER TABLE public.metric_1c OWNER TO fluent;
SELECT create_hypertable('metric_1c', 'timestamp', chunk_time_interval => 864000);
```

```
CREATE TABLE public.metric_apdex (
    "timestamp" bigint NOT NULL,
    name text,
    host text,
    operation text,
    session int,
    "user" text,
    operation_time real,
    weight real
);
ALTER TABLE public.metric_apdex OWNER TO fluent;
SELECT create_hypertable('metric_apdex', 'timestamp', chunk_time_interval => 864000);

CREATE TABLE public.metric_vmware (
    "timestamp" bigint NOT NULL,
    name text,
    host text,
    instance text,
    objectname text,
    "CPU_stolen_time" real,
    "Effective_VM_Speed_in_MHz" real,
    "Host_processor_speed_in_MHz" real,
    "Memory_Active_in_MB" int,
    "Memory_Ballooned_in_MB" int,
    "Memory_Swapped_in_MB" int
);
ALTER TABLE public.metric_vmware OWNER TO fluent;
SELECT create_hypertable('metric_vmware', 'timestamp', chunk_time_interval =>
864000);
```

Приложение Б
Настройка агента telegraf.conf

```
debug = true
quiet = false
logtarget = "file"
logfile = "c:\\1c\\agent-logs\\telegraf.log"
logfile_rotation_interval = "1d"
logfile_rotation_max_size = "100MB"
logfile_rotation_max_archives = 5

###############################################################################
#                            OUTPUT PLUGINS                                   #
###############################################################################
[[outputs.kafka]]
  ## URLs of kafka brokers
  brokers = ["ip адрес:9092"]
  topic = "metric"
  data_format = "json"
   [outputs.kafka.topic_suffix]
     method = "measurement"
     separator = "-"

[[inputs.win_perf_counters]]
#  UseWildcardsExpansion = true
#  PrintValid = true
[[inputs.win_perf_counters.object]]
 ObjectName = "Processor"
  Instances = ["_Total"]
  Counters = ["% Interrupt Time", "% Privileged Time", "% User Time", "% Processor
Time"]
  Measurement = "cpu"
[[inputs.win_perf_counters.object]]
  ObjectName = "Processor Information"
  Instances = ["_Total"]
  Counters = ["% Processor Time", "% User Time", "% Privileged Time"]
  Measurement = "cpu"
```

```
[[inputs.win_perf_counters.object]]
  ObjectName = "System"
  Instances = ["_Total"]
  Counters = ["Processor Queue Length", "Threads", "Context Switches/sec"]
  Measurement = "system"
[[inputs.win_perf_counters.object]]
  ObjectName = "Memory"
  Instances = ["_Total"]
  Counters = ["Available Bytes", "Cache Bytes", "Committed Bytes", "Free & Zero Page
List Bytes", "Page Faults/sec", "Pages/sec"]
  Measurement = "memory"


[[inputs.win_perf_counters.object]]
  ObjectName = "Paging File"
  Instances = ["_Total"]
  Counters = ["% Usage"]
  Measurement = "memory"


[[inputs.win_perf_counters.object]]
  ObjectName = "Network Interface"
  Instances = ["*"]
  Counters = ["Bytes Total/sec","Output Queue Length"]
  Measurement = "network"



[[inputs.win_perf_counters.object]]
  ObjectName = "LogicalDisk"
  Instances = ["*"]
  Counters = ["% Free Space","Free Megabytes","Avg. Disk Read Queue Length","Avg.
Disk Write Queue Length","Avg. Disk sec/Read","Avg. Disk sec/Write","Avg. Disk
Bytes/Read","Avg. Disk Bytes/Write"]
  Measurement = "disk"


[[inputs.win_perf_counters.object]]
  ObjectName = "Process"
  Instances = ["ragent*"]
  Counters = ["% Processor Time","Private Bytes","Virtual Bytes","ID Process","Thread
Count"]
  Measurement = "proc1c"
```

```
[[inputs.win_perf_counters.object]]
  ObjectName = "Process"
  Instances = ["rmngr*"]
  Counters = ["% Processor Time","Private Bytes","Virtual Bytes","ID Process","Thread
Count"]
  Measurement = "proc1c"

[[inputs.win_perf_counters.object]]
  ObjectName = "Process"
  Instances = ["rphost*"]
  Counters = ["% Processor Time","Private Bytes","Virtual Bytes","ID Process","Thread
Count"]
  Measurement = "proc1c"

[[inputs.win_perf_counters.object]]
  ObjectName = "Process"
  Instances = ["sqlservr*"]
  Counters = ["% Processor Time","Private Bytes","Virtual Bytes","ID Process","Thread
Count"]
  Measurement = "mssql"

[[inputs.win_perf_counters.object]]
  ObjectName = "SQLServer:Access Methods"
  Instances = [""]
  Counters = ["Full Scans/sec","Table Lock Escalations/sec"]
  Measurement = "mssql"

[[inputs.win_perf_counters.object]]
  ObjectName = "SQLServer:Buffer Manager"
  Instances = [""]
  Counters = ["Buffer cache hit ratio","Free List Stalls/sec","Free Pages","Lazy
Writes/sec"]
  Measurement = "mssql"

[[inputs.win_perf_counters.object]]
  ObjectName = "SQLServer:Buffer Node"
  Instances = ["*"]
  Counters = ["Page life expectancy"]
  Measurement = "mssql"
```

```
[[inputs.win_perf_counters.object]]
  ObjectName = "SQLServer:Databases"
  Instances = ["*"]
  Counters = ["Data File(s) Size (KB)","Log File(s) Size (KB)","Log File(s) Used Size (KB)"]
  Measurement = "mssql"

[[inputs.win_perf_counters.object]]
  ObjectName = "SQLServer:Databases"
  Instances = ["_Total"]
  Counters = ["Transactions/sec"]
  Measurement = "mssql"

[[inputs.win_perf_counters.object]]
  ObjectName = "SQLServer:Databases"
  Instances = [""]
  Counters = ["Active Transactions"]
  Measurement = "mssql"

[[inputs.win_perf_counters.object]]
  ObjectName = "SQLServer:General Statistics"
  Instances = [""]
  Counters = ["Active Temp Tables","Temp Tables Creation Rate","Temp Tables For Destruction","Processes blocked","Transactions"]
  Measurement = "mssql"

[[inputs.win_perf_counters.object]]
  ObjectName = "SQLServer:Locks"
  Instances = ["_Total"]
  Counters = ["Lock Timeouts (timeout > 0)/sec","Number of Deadlocks/sec","Average Wait Time (ms)","Lock Requests/sec","Lock Timeouts/sec","Lock Wait Time (ms)"]
  Measurement = "mssql"

[[inputs.win_perf_counters.object]]
  ObjectName = "SQLServer:Memory Manager"
  Instances = [""]
  Counters = ["Database Cache Memory (KB)","Free Memory (KB)","Granted Workspace Memory (KB)","Lock Blocks","Lock Memory (KB)","Memory Grants Outstanding","Memory Grants Pending","Target Server Memory (KB)","Total Server Memory (KB)"]
  Measurement = "mssql"
```

```
[[inputs.win_perf_counters.object]]
  ObjectName = "SQLServer:Plan Cache(SQL Plans)"
  Instances = [""]
  Counters = ["Cache Hit Ratio"]
  Measurement = "mssql"


[[inputs.win_perf_counters.object]]
  ObjectName = "SQLServer:SQL Statistics"
  Instances = [""]
  Counters = ["Auto-Param Attempts/sec","Batch Requests/sec","Failed Auto-
params/sec","SQL Compilations/sec","SQL Re-Compilations/sec"]
  Measurement = "mssql"


[[inputs.win_perf_counters.object]]
  ObjectName = "SQLServer:Transactions"
  Instances = [""]
  Counters = ["Free Space in tempdb (KB)","Longest Transaction Running
Time","Transactions"]
  Measurement = "mssql"


[[inputs.win_perf_counters.object]]
  ObjectName = "SQLServer:Wait Statistics(Waits in progress)"
  Instances = [""]
  Counters = ["Lock waits","Log buffer waits","Log write waits","Memory grant queue
waits","Network IO waits","Non-Page latch waits","Page IO latch waits","Page latch
waits","Thread-safe memory objects waits","Transaction ownership waits"]
  Measurement = "mssql"


[[inputs.win_perf_counters.object]]
  ObjectName = "SQLServer:Latches"
  Instances = [""]
  Counters = ["Number of SuperLatches","Total Latch Wait Time (ms)","Latch
Waits/sec","SuperLatch Promotions/sec","SuperLatch Demotions/sec","Average Latch Wait
Time (ms)"]
  Measurement = "mssql"
```

Приложение В
Шаблоны индексов технического журнала и журнала регистраций

Примечание: при настройке шаблонов требуется скопировать данные из блоков settings и mappings приведенных ниже JSON для настройки соответствующих шагов. Mappings импортируется в Kibana через кнопку Load JSON.

Технический журнал:

```
{
  "template": {
    "settings": {
      "index": {
        "number_of_shards": "2",
        "number_of_replicas": "0",
        "refresh_interval": "30s"
      }
    },
    "mappings": {
      "dynamic_templates": [],
      "properties": {
        "ConnectString": {
          "type": "text"
        },
        "Context": {
          "type": "text"
        },
        "DataBase": {
          "type": "keyword"
        },
        "Descr": {
          "type": "text"
        },
        "Func": {
          "type": "keyword"
        },
        "IB": {
          "type": "text"
        },
        "IName": {
          "type": "text"
        },
        "InBytes": {
          "type": "long"
        },
        "Interface": {
          "type": "text"
        },
```

```
"Locks": {
  "type": "text"
},
"MName": {
  "type": "text"
},
"Memory": {
  "type": "long"
},
"MemoryPeak": {
  "type": "long"
},
"OSThread": {
  "type": "text"
},
"OutBytes": {
  "type": "long"
},
"Regions": {
  "type": "keyword"
},
"RetExcp": {
  "type": "text"
},
"Rows": {
  "type": "long"
},
"Sdbl": {
  "type": "text"
},
"SessionID": {
  "type": "text"
},
"Sql": {
  "type": "text"
},
"SrcName": {
  "type": "keyword"
},
"Trans": {
  "type": "keyword"
},
"Txt": {
  "type": "text"
},
"Usr": {
  "type": "keyword"
},
```

```
"WaitConnections": {
  "type": "text"
},
"applicationName": {
  "type": "keyword"
},
"callWait": {
  "type": "text"
},
"clientID": {
  "type": "keyword"
},
"computerName": {
  "type": "keyword"
},
"connectID": {
  "type": "text"
},
"dbpid": {
  "type": "text"
},
"detachSeances": {
  "type": "text"
},
"duration": {
  "type": "long"
},
"event": {
  "type": "keyword"
},
"excp_txt": {
  "type": "text"
},
"first": {
  "type": "text"
},
"formatted_date": {
  "type": "date"
},
"hostname": {
  "type": "keyword"
},
"isAttached": {
  "type": "text"
},
"level": {
  "type": "keyword"
},
```

```
"lic_txt": {
  "type": "text"
},
"log": {
  "type": "text"
},
"message": {
  "type": "text"
},
"p:processName": {
  "type": "text"
},
"proc_txt": {
  "type": "text"
},
"process": {
  "type": "keyword"
},
"processName": {
  "type": "keyword"
},
"product": {
  "type": "text"
},
"res": {
  "type": "text"
},
"rphostAlive": {
  "type": "text"
},
"t:applicationName": {
  "type": "text"
},
"t:clientID": {
  "type": "text"
},
"t:computerName": {
  "type": "text"
},
"t:connectID": {
  "type": "text"
},
"tj_path": {
  "type": "text"
},
"tz": {
  "type": "integer"
}
```

```
      }
    },
    "aliases": {}
  }
}
```


Журнал регистраций:


```
{
  "template": {
    "settings": {
      "index": {
        "number_of_shards": "1",
        "number_of_replicas": "0",
        "refresh_interval": "30s"
      }
    },
    "mappings": {
      "dynamic_templates": [],
      "properties": {
        "MoreMetadata": {
          "type": "text"
        },
        "filename": {
          "type": "text"
        },
        "formatted_date": {
          "type": "date"
        },
        "log": {
          "type": "text"
        },
        "БазаДанных": {
          "type": "keyword"
        },
        "ВажностьСобытия": {
          "type": "keyword"
        },
        "Комментарий": {
          "type": "keyword"
        },
        "Компьютер": {
          "type": "keyword"
        },
        "НомерТранзакции": {
          "type": "text"
        },
        "ОбъектМетаданных": {
          "type": "keyword"
```

```
        },
        "Пользователь": {
          "type": "keyword"
        },
        "ПредставлениеДанных": {
          "type": "text"
        },
        "Приложение": {
          "type": "keyword"
        },
        "Сеанс": {
          "type": "keyword"
        },
        "Сервер": {
          "type": "keyword"
        },
        "Событие": {
          "type": "keyword"
        },
        "Соединение": {
          "type": "text"
        },
        "СтатусТранзакции": {
          "type": "keyword"
        },
        "СтрокаДанных1": {
          "type": "text"
        },
        "СтрокаДанных2": {
          "type": "text"
        },
        "СтруктураДанных": {
          "type": "keyword"
        },
        "Транзакция": {
          "type": "text"
        }
      }
    },
    "aliases": {}
  }
}
```

Приложение Г
Настройка fluent-bit.conf

Примечание: Указаны только измененные и дополнительные параметры

fluent-bit.conf

```
[SERVICE]
   log_file     C:\1C\agents-logs\fluent-bit.log
# Перед проверкой установить в debug, перед использованием вернуть info
    log_level    debug

    # Parsers File
    # ============
    # specify an optional 'Parsers' configuration file
    parsers_file parsers.conf

[INPUT]
    name tail
    path_key tj_path
    path C:\1C\tj\*\*.log
    db C:\ProgramData\FluentBit\db\tj1c.db
    multiline.parser tj1c
    Buffer_Chunk_Size 500KB
    Buffer_Max_Size 5MB
    Tag tj_1c

[INPUT]
    Name         winlog
    Channels     System
    Interval_Sec 5
    DB           C:\ProgramData\FluentBit\db\winlog.db
    Tag winlog

[INPUT]
    name tail
```

```
    path_key jr_path
    path C:\Путь\до\*.lgp
    db C:\ProgramData\FluentBit\db\rj.db
    multiline.parser jr1c
    Buffer_Chunk_Size 500KB
    Buffer_Max_Size 5MB
    Tag jr

[FILTER]
    name record_modifier
    match tj_1c
    record hostname ${HOSTNAME}
    record product tj_1c
    record tz 3

[FILTER]
    name record_modifier
    match winlog
    record hostname ${HOSTNAME}
    record product winlog
    record tz 3

[FILTER]
    name record_modifier
    match jr
    record hostname ${HOSTNAME}
    record product jr_1c
    record tz 3

[OUTPUT]
    Name        Kafka
    Match       winlog
    Brokers     192.168.1.1:9092
    Topics      winlog

[OUTPUT]
    Name        Kafka
```

```
    Match       tj_1c
    Brokers     192.168.1.1:9092
    Topics      tj1c

[OUTPUT]
    Name        Kafka
    Match       jr
    Brokers     192.168.1.1:9092
    Topics      jr
```

В конфигурационный файл parsers.conf добавить

```
[MULTILINE_PARSER]
    name tj1c
    type regex
    flush_timeout 1000
    rule "start_state" "/^\d{2}:\d{2}\.\d{6}-/" "cont"
    rule "cont" "/^(?!\d{2}:\d{2}\.)/" "cont"

[MULTILINE_PARSER]
    name jr1c
    type regex
    flush_timeout 1000
    rule "start_state" "/^{\d{14},\w,$/" "cont"
    rule "cont" "/^(?!{\d{14})/" "cont"
```