перезапустить. После чтения конфигурации происходит соединение с первой точкой отправки данных и отправка данных о агенте. Затем происходит проверка поддерживаемых «Типов мониторов» в данной операционной ситеме. О неподдерживаемых типах выдаются предупреждения. После этого происходит чтение конфигурации запусков мониторов, происходит проверка типа монитора. Если тип не поддерживается, выдаётся предупреждение и данный монитор пропускается. К оставшимся мониторам привязываются соответствующие триггеры. Далее программа ожидает времени запуска монитора, запускает его в отдельном процессе и ожидает дальше.

В данной программе используется собственная реализация запуска процессов по cron расписанию. Для каждого монитора просчитывается определённое количество отметок времени, когда монитор должен запуститься. Основной процесс программы периодически (раз в минуту) проверяет ближайшие отметки времени. Когда текущее время проходит отметку времени выполняется запуск отдельного под-процесса для монитора.

Основной процесс практически не задерживается запуском под-процессов мониторов. Каждый под-процесс запускает команду монитора, ожидает её завершения, обрабатывает выходные данные команды и отправляет данные другому агенту или БД в зависимости от конфигурации. Если выходные данные команды соответствуют условию связанного с монитором триггера, то запускается ещё один под-процесс, где выполняются те же действия для триггера. Агент имеет настраиваемый лимит на количество одновременно запущенных процессов, если количество процессов достигает лимита, то запуск монитора откладывается до следующей проверки отметок времени. Также присутствует настраиваемый таймаут для мониторов.

С. М. Климов
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **А. И. Кучеров,** ст. преподаватель

ПЕРЕДАЧА ДАННЫХ В ПРОГРАММЕ АВТОМАТИЗАЦИИ СОСТОЯНИЯ ОПЕРАЦИОННОЙ СИСТЕМЫ НА УЗЛЕ ЛВС

Программа автоматизации состояния ОС на узле ЛВС запускает команды по расписанию и обрабатывает их вывод. Обработка вывода осуществляется следующим путём:

1. Вывод разбивается на массив объектов согласно шаблонам разделения. Далее каждый объект вывода обрабатывается отдельно.

2. Из объекта вывода достаются нужные поля согласно указанной схеме. Если схема не указана, то сохраняется весь объект.

3. Если указан один и более фильтров, объект проверяется на соответствие фильтрам. Если не соответствует – объект отбрасывается.

4. Все оставшиеся объекты обрабатываются настроенным и активным эндпоинтом.

Шаблон разделения состоит из комбинации трёх полей. Каждое из полей может быть не задано, содержать текст или регулярное выражение.

В зависимости от того какие полей заданы, разбивка может выбирать только определённые куски из вывода. Регулярные выражения называются следующим образом:

– begin – определяет где начинается новый объект;

– end – определяет где заканчивается объект;

– separator – разделитель объектов, не включается.

Схема объекта вывода описывает, какие поля должны быть сохранены, их имена, типы, откуда брать значение, значения по умолчанию. Поддерживается обработка следующих форматов: text (сохранить объект вывода целиком), json, yaml, csv (можно поменять разделитель) и regexp (можно сохранить поля, указанные в круглых скобках в регулярном выражении).

Фильтры позволяют отсечь ненужные объекты, используя шаблон включения, шаблон исключения или по порядковому номеру. Можно задать любое количество фильтров, если объект не соответствует фильтру, объект отбрасывается, последующие фильтры не проверяются.

В приложении можно задать один и более эндпоинтов – точек, куда нужно отправить данные. На данный момент доступны только 3 эндпоинта: другой агент (выступит в качестве прокси), elasticsearh, лог агента.

Программа будет отправлять данные по доступному эндпоинту с самым низким номером в списке. Если все эндпоинты недоступны, программы выдаст предупреждение в лог и будет откладывать данные в буфер и ждать пока один из эндпоинтов не станет доступен, затем отправит всё из буфера по доступному эндпоинту.

А. В. Ковалев
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **А. В. Воруев**, канд. физ.-мат. наук, доцент

ЛОГИРОВАНИЕ МИКРОСЕРВИСНЫХ ПРИЛОЖЕНИЙ

Логирование данных приложений является одной из важнейших функций в работе приложения и возможности просматривать значимые данные. Существует большое множество сборщиков логов, например: Logwatch, syslog-ng, Fluentd, Fluentd.

Для ограничения входящего потока данных необходима система фильтрации собранных данных, на данный момент существует большое обилие систем фильтрации, такие как: Elasticsearch, Algolia, Sphinx, Apache Solr и другие.

После выбора системы сбора и фильтрации собранных данных нужна удобная система визуализации, будет использоваться Kibana, которая хорошо взаимодействует с elasticsearch, проста в настройке, имеет удобный интерфейс и информативно отображает собранные данные с возможностью отображать интересующую информацию в полном объеме.

Порядок работы стека сервисов следующий: Fluentd обрабатывает файлы журналов приложений на основе установленных нами критериев фильтрации и отправляет эти журналы в Elasticsearch. Через Kibana просматривают и анализируют журналы, когда это необходимо (рисунок 1).

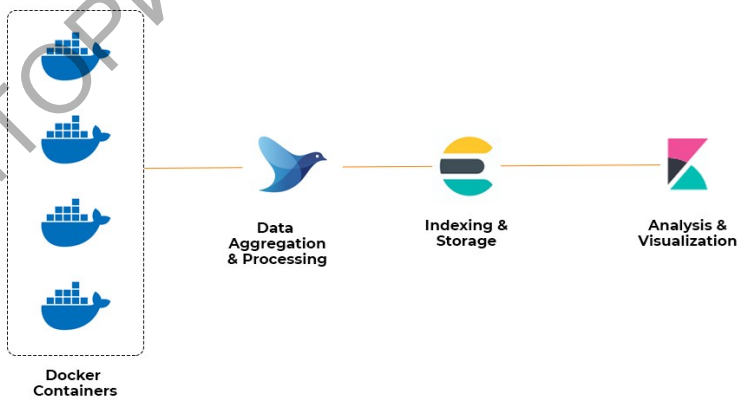

Рисунок 1 – Схема взаимодействия в EFK

Все три инструмента основаны на JVM и требуют установки JDK 1.8.