Практическая работа 7

«Установка и настройка ELK стека для анализа журналов и логов»

Цель:

Научиться устанавливать и настраивать основные компоненты ELK стека (Elasticsearch, Logstash, Kibana) для анализа и визуализации журналов и логов.

Описание задания

Практическая работа по установке и настройке ELK стека представляет собой важный этап в закреплении навыков, полученных ранее при работе с развертыванием компонентов в изолированной среде WSL с использованием Docker и Docker Compose. Кроме того, развертывание ELK стека, поможет углубить понимание процесса анализа и визуализации логов, что является важным навыком для работы в области разработки и администрирования информационных систем.

Основные этапы работы включают:

- Установку и конфигурирование ELK стека.
- Конфигурирование Filebeat для сбора и отправки логов, генерируемых самописным Python-приложением, в Elasticsearch для последующего анализа.
- Настройку и подключение к интерфейсу Kibana для визуализации данных из Elasticsearch.
- Поиск и анализ логов.

Немного теории

Стек **ELK** — это мощный набор инструментов для эффективного решения широкого спектра задач сбора, хранения и анализа данных:

- **Elasticsearch** – решение для полнотекстового поиска, построенное поверх Apache Lucene и имеющее дополнительные удобства.
- **Logstash** – утилита для сборки, фильтрации и последующего перенаправления в конечное хранилище данных. Этот механизм обеспечивает конвейер в реальном времени. Он может принимать данные из нескольких источников и преобразовывать их в документы JSON.

- **Kibana** — приложение, позволяющее брать и искать данные из Elasticsearch и строить наглядные графики.

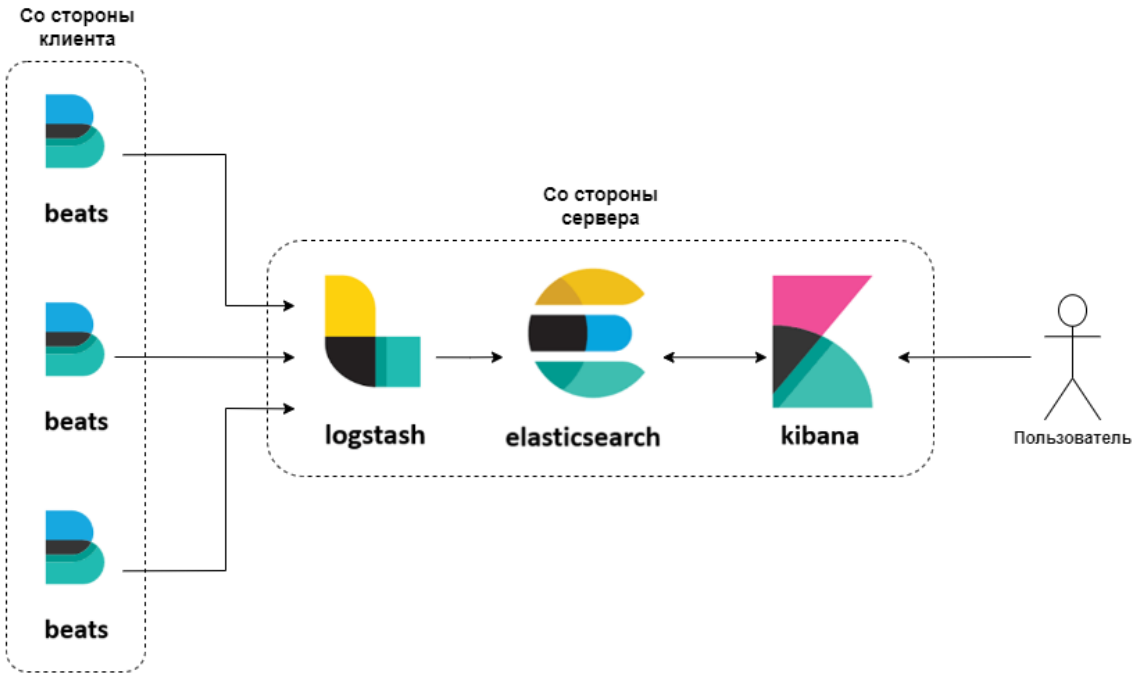

Рисунок 1. Архитектура ELK стека.

Для просто восприятия рисунка 1:

1. **Beat** следит за изменениями логов (**Filebeat** следит за файлами) и пушит логи в **Logstsash**;
2. **Logstash** фильтрует эти логи, производит с ними некоторые манипуляции и кладет их в нужный индекс **Elasticsearch** (таблицу в терминах привычных баз данных);
3. **Kibana** визуализирует эти логи и позволяет вам удобно искать нужные события.

Шаги:

1. **Подготовка к установке**

   Если у вас еще не установлен Docker то необходимо обратиться к официальной документации размещенной по адресу https://docs.docker.com/get-docker/ чтобы выполнить инсталляцию пакета подходящей для вашей операционной системы:

   - Install Docker Desktop on Windows - https://docs.docker.com/desktop/install/windows-install/
   - Install Docker Desktop on Linux - https://docs.docker.com/desktop/install/linux-install/
   - Install Docker Desktop on Mac - https://docs.docker.com/desktop/install/mac-install/

   Перед началом выполнения данного практического задания необходимо склонировать ваш репозиторий и создать новую ветку. Для удобства разворачивания стека ELK мы воспользуемся ранее приобретенным опытом использования Docker Compose. Это мощный инструмент, который позволит нам запустить несколько контейнеров одновременно упростив процесс развертывания и настройки стека ELK запуская все компоненты одной командой. Кроме того, Docker Compose обеспечивает гибкость в настройке нашего окружения. Если потребуется переконфигурировать стек ELK, мы можем легко внести изменения в файл конфигурации Docker Compose, а также решая проблемы с объявлением и конфликтами портов. Это делает процесс развертывания и настройки более простым и удобным, освобождая наше время для более глубокого изучения функциональности ELK стека и анализа полученных данных.

   Для начала подготовим конфигурации соответствующих сервисов. В корне своего проекта создайте структуру сотоящую из следующих файлов и их содержимым:

   configs/elasticsearch/config.yml

```
cluster.name: "elk"
network.host: 0.0.0.0 # Для корректной работы внутри контейнера

xpack.security.enabled: true # Для поддержки функционала,
обеспечивающего безопасность кластера
xpack.license.self_generated.type: basic # Типа лицензии "basic" для
наших нужд хватит с головой
```

configs/logstash/config.yml

```
http.host: "0.0.0.0"
```

configs/kibana/config.yml

```
server.name: kibana
server.host: 0.0.0.0
server.publicBaseUrl: "http://localhost:5601"
monitoring.ui.container.elasticsearch.enabled: true # Для корректного
сбора метрик с elastic search, запущенного в контейнере

elasticsearch.hosts: [ "http://elasticsearch:9200" ]
elasticsearch.username: elastic
elasticsearch.password: MyPw123
```

2. Описание и запуск стека

Описываем конфигурацию нашего будущего стека в *docker-compose.yml*:

```
version: '3.7'

services:
  elasticsearch:
    image: elasticsearch:7.16.1
    volumes:
      -
./configs/elasticsearch/config.yml:/usr/share/elasticsearch/config/elas
ticsearch.yml:ro
      -
./docker_volumes/elasticsearch/data:/usr/share/elasticsearch/data
    environment:
      ES_JAVA_OPTS: "-Xmx512m -Xms512m"
      ELASTIC_USERNAME: "elastic"
      ELASTIC_PASSWORD: "MyPw123"
      discovery.type: single-node
    networks:
      - elk
    ports:
      - "9200:9200"
      - "9300:9300"

  logstash:
    image: logstash:7.16.2
    volumes:
      -
./configs/logstash/config.yml:/usr/share/logstash/config/logstash.yml:r
o
    environment:
      LS_JAVA_OPTS: "-Xmx512m -Xms512m"
```

```
    ports:
      - "5044:5044"
      - "5000:5000"
      - "9600:9600"
    networks:
      - elk
    depends_on:
      - elasticsearch

  kibana:
    image: kibana:7.16.1
    depends_on:
      - elasticsearch
    volumes:
      -
./configs/kibana/config.yml:/usr/share/kibana/config/kibana.yml:ro
    networks:
      - elk
    ports:
      - "5601:5601"

networks:
  elk:
    driver: bridge
```

Теперь поднимаем созданный проект командой:

```
docker-compose up
```

Примечание: предполагается, что ранее вы уже выполняли настройку и конфигурирование Docker и Docker Compose и эти компоненты у вас работают исправно. В ином случае, обратитесь к инструкции по их установке используя официальный ресурс https://docs.docker.com/get-docker/ либо руководствуясь той же инструкцией устраните ошибки чтобы двигаться дальше.

Если предыдущая команда была выполнена спешно, убеждаемся, что мы можем попасть в административный вэб-интерфейс Kibana, для этого переходим по адресу http://localhost:5601 и вводим логин/пароль из docker-compose.yml конфига (секция environment сервиса elasticsearch).

Обратите внимание: сервисы стартуют долго, Kibana может стартовать от 3 до 5 минут, так что не торопитесь паниковать и искать ошибки.

После того, как доступ в админ панель былаполучена на этом этапе можно закрывайте ее и приступать к следующим шагам.

3. Настройка Logstash для сохранения логов в Elasticsearch

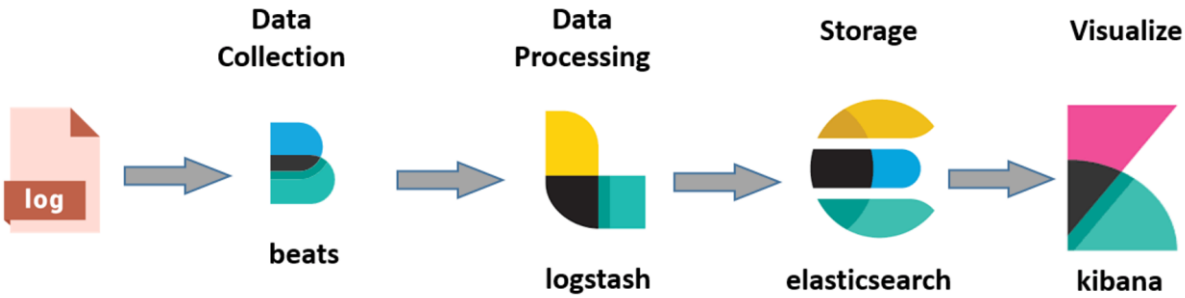

Рисунок 2. Схема сбора логов.

Для этого подготовим конфигурацию configs/logstash/pipelines.yml:

```
- pipeline.id: service_stamped_json_logs
  pipeline.workers: 1
  pipeline.batch.size: 1
  path.config:
"/usr/share/logstash/config/pipelines/service_stamped_json_logs.conf"
```

Мы будем собирать логи в формате json с указанием сервиса. Для этого нужно будет привести логи в json формат. В соответсвии с этим требованием, создадим следующую конфигурацию

configs/logstash/pipelines/service_stamped_json_logs.conf

```
# Логи будут прилетать из beats'ов по порту 5044
input {
  beats {
    port => 5044
  }
}

filter {
  # Дропаем лог, если он пришел от неизвестного нам сервиса (по
желанию)
  # Ниже я два раза указал host_metrics_app в списке - это не опечатка.
Какого-то лешего в условии, в массиве должно быть минимум 2 элемента.
  # Так как приложение у нас одно - просто дублируем
  # Поле service у нас появится благодаря конфигурированию Filebeat
  if [fields][service] not in ["host_metrics_app", "host_metrics_app"]
{
```

```
    drop {}
  }
  # Оригинальный json-лог, который был сгенерирован вашим приложением,
будет лежать по ключу message
  # (из filebeat'а логи прилетают не в чистом виде)
  json {
    source => "message"
  }
  # Говорим logstash'у, чтобы в качестве timestamp'а лога он брал
именно наш timestamp
  # (в моем случае поле asctime в теле сообщения в формате "yyyy-MM-dd
HH:mm:ss.SSS" и часовом поясе UTC)
  # и затем подтирал поле asctime.
  date {
    match => ["asctime", "yyyy-MM-dd HH:mm:ss.SSS"]
    timezone => "UTC"
    target => "@timestamp"
    remove_field => ["asctime"]
  }
}

output {
  # Отображаем лог в stdout (поиграйтесь и удалите данную строку)
  stdout {}
  # Пушим лог в elasticsearch, индекс будет создан автоматически по
названию сервиса и текущей дате
  elasticsearch {
    hosts => "elasticsearch:9200"
    index => "logs_%{[fields][service]}-%{+YYYY.MM.dd}"
    user => "elastic"
    password => "MyPw123"
  }
}
```

Заранее подготовим конфигурацию и для Filebeat:

configs/filebeat/config.yml

```
filebeat.inputs:
  - type: log
    enabled: true
    # Я запущу filebeat в докере и проброшу логи приложения по данному
пути
    paths:
      - /host_metrics_app/host_metrics_app.log
    # В fields мы можем указать дополнительные поля, а затем в logstash
вытаскивать их
    # и делать какую-нибудь дополнительную работу с логами
    fields:
      # Название нашего сервиса
      service: host_metrics_app
```

```
output.logstash:
  # Будьте внимательны при запуске вне докера и вместо logstash укажите
правильный адрес хоста с logstash.
  hosts: ["logstash:5044"]
```

Обновим наш docker-compose конфиг. Секцию volumes сервиса logstash заменим на:

```
volumes:
  - ./configs/logstash/config.yml:/usr/share/logstash/config/logstash.yml:ro
  -
./configs/logstash/pipelines.yml:/usr/share/logstash/config/pipelines.yml:ro
  - ./configs/logstash/pipelines:/usr/share/logstash/config/pipelines:ro
```

Добавим сервис filebeat:

```
beats:
  image: elastic/filebeat:7.16.2
  volumes:
    - ./configs/filebeat/config.yml:/usr/share/filebeat/filebeat.yml:ro
    - ./host_metrics_app/:/host_metrics_app/:ro
  networks:
    - elk
  depends_on:
    - elasticsearch
```

4. Собираем логи и наблюдаем их в Kibana

Для этого нам понадобится написать простое приложение на python, которое лишь собирает некоторые метрики хоста и складывает в лог-файл в формате json. Пример такого host_mertics_app/main.py приложения:

```
import datetime
import json
import os
from time import sleep

import psutil

BASE_DIR = os.path.dirname(os.path.abspath(__file__))

while True:
    load1, load5, load15 = psutil.getloadavg()
    ram_usage_percent = psutil.virtual_memory().percent

    log = {
        'load_avg': {
```

```
                'load1': load1,
                'load5': load5,
                'load15': load15,
            },
            'ram_usage_percent': ram_usage_percent,
            'asctime':
datetime.datetime.utcnow().isoformat(timespec='milliseconds', sep=' ')
        }

        with open(os.path.join(BASE_DIR, 'host_metrics_app.log'), 'a') as
f:
            f.write(json.dumps(log) + '\n')

        sleep(7)
```

Теперь запустим наше приложение и ELK стек (с Filebeat'ом):

```
python host_metrics_app/main.py
docker-compose.yml
```

5. Настройка Kibana на вывод логов

Необходимо настроить Kibana так, чтобы все индексы – Kibana воспринимала как одну большую таблицу с логами. Для этого повторно залогинтесь в вэб интерфейс Kibana и зайдите в меню: Management --> Stack Management --> Index patterns и создайте новый index pattern "logs_host_metrics_app*"
*место где будут собираться ваши логи от приложения собранные Logstash.

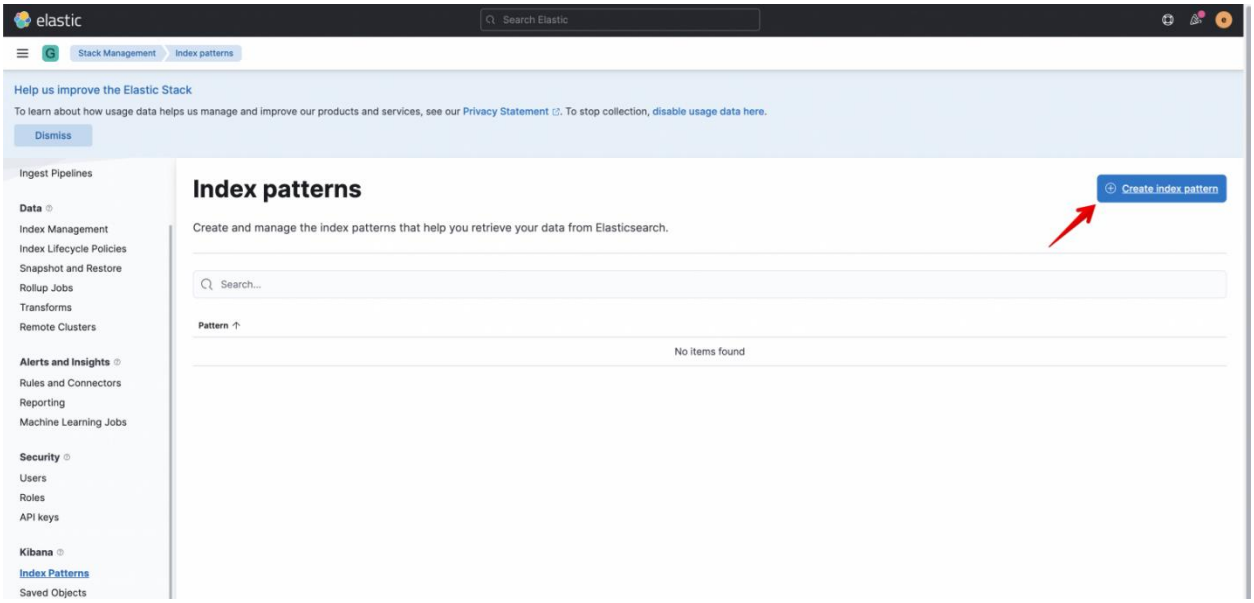

Рисунок 3. Создания шаблона индекса.

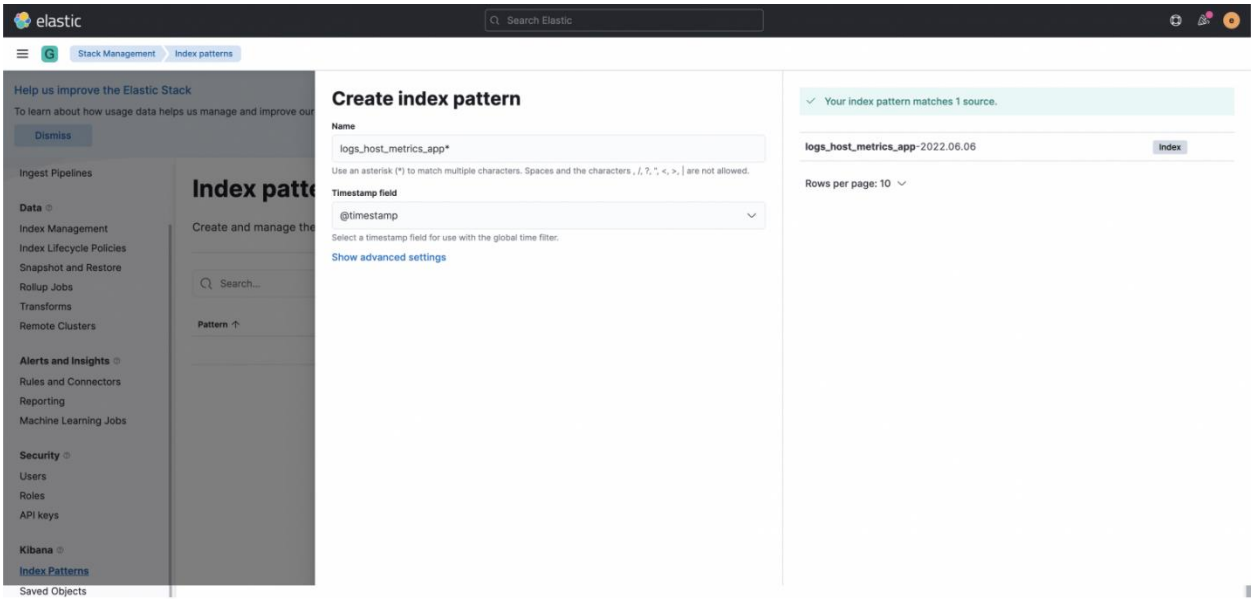

Рисунок 4. Добавление логов в шаблон индекса.

После этого можно будет смотреть и искать наши логи, в меню: Analytics потом в Discover.

6. Проверка результата выполненной работы и задание

Для проверки работы стека **ELK (Elasticsearch, Logstash, Kibana)** можно выполнить следующие действия:

Выполните проверку Elasticsearch:

- Запустите `curl http://localhost:9200` в командной строке. Вы должны увидеть ответ, содержащий информацию о вашем Elasticsearch кластере.
- Проверьте, что индексы успешно создаются, используя

```
curl http://localhost:9200/_cat/indices?v
```

Выполните проверку Logstash:

- Убедитесь, что Logstash успешно получает данные и отправляет их в Elasticsearch. Проверьте логи Logstash, чтобы убедиться, что он работает без ошибок.

- Вы можете использовать команду logstash -t для проверки правильности конфигурации Logstash.

Проверка Kibana:

- Откройте Kibana в браузере (обычно по адресу `http://localhost:5601`).
- Проверьте, что вы можете создавать визуализации и дашборды, используя данные, отправленные в Elasticsearch.
- Убедитесь, что вы можете выполнять поиск по данным в Elasticsearch.

Задание выполните проверку конечной работы стека ELK:

- Отправьте какие-нибудь тестовые данные в Logstash.
- Убедитесь, что эти данные отображаются в Kibana.
- Проверьте, что вы можете выполнять поиск по этим данным и создавать визуализации.
- Эти базовые проверки помогут вам убедиться, что весь стек ELK работает правильно. Если возникают какие-либо проблемы, внимательно проверьте конфигурацию каждого компонента и проанализируйте логи на наличие ошибок.

7. **Сохранение отчет о проделанной работе.**

Создайте директорию reports и разместите в ней результаты проверок в виде скриншотов, подтверждающих выполнения работы. Зафиксируйте изменения с использованием `git add` и `git commit`. Отправьте изменения на GitLab с использованием `git push`.

8. **Создание Pull Request.**

Создайте Pull Request на слияние вашей ветки в main(master) ветку. В запросе укажите пул выполненных работ по заданию. В качестве Rewiewer укажите преподавателя.
Задание считается выполненным, если преподаватель выполнил слияние вашей ветки в основную, иначе смотрите комментарий в вашем запросе с описанием недочетов, исправляйте ошибки и снова выставляйте работу на проверку.

Вопросы и поддержка:

Если у вас возникают трудности или у вас есть вопросы, не стесняйтесь обращаться за помощью к преподавателю или на форум группы. Удачи в выполнении задания!