СЛЁРМ Образование для инженеров и технических лидеров

RabbitMQ - Мониторинг

Спикер: Алексей Барабанов

Цель урока

Узнать о механизмах мониторинга и уметь их настраивать на практике

План урока

- **Логирование** - совсем немного про логирование
- **Мониторинг** - наблюдаем и реагируем
- Встроенный **prometheus exporter**
- **Telegraf** - альтернативный exporter
- **Полезные метрики** (втч для кластера)
- Настройка **rabbitmq→telegraf→prometheus→grafana→telegram**
- **Практическое задание**

Логирование

- В docker - используется обычный **log.console**
- При использовании log.file можно настроить ротацию
- Можно настроить **логирование в JSON**
- Можно менять уровень на лету **rabbitmqctl set_log_level debug**
- RABBITMQ_SERVER_ADDITIONAL_ERL_ARGS: -rabbit log_levels [{connection,error},{default,error}] - **не работает с 3.8**

Уровни логирования

- **debug:** самое детальное логирование
- **info:** средний уровень логирования, по умолчанию
- **warning:** предупреждения, ошибки
- **error:** только ошибки
- **critical:** только критические уведомления
- **none:** ничего не пишет кроме стартовой плашки

Категории логирования

- **connection:** всё что связано с соединениями всех протоколов
- **channel:** логи каналов - только для AMQP
- **queue:** логи очередей
- **mirroring:** логи репликации классических очередей в кластере
- **federation:** логи федерации
- **upgrade:** логирование процедуры апгрейда
- **default:** все логи

Документация по логированию

Настройка логирования

- На лету - без перезагрузки можно сменить уровень логирования

```
rabbitmqctl set_log_level debug
```

Настройка логирования

- Монтируем файл в папку **/etc/rabbitmq/conf.d/**

```
rabbitmq:
  image: rabbitmq:3.10.7-management
  hostname: rabbitmq
  environment:
    RABBITMQ_DEFAULT_USER: rmuser
    RABBITMQ_DEFAULT_PASS: rmpassword
  volumes:
    - ./rabbitmq:/var/lib/rabbitmq
    - ./logging.conf:/etc/rabbitmq/conf.d/logging.conf
  ports:
    - 15672:15672
```

Настройка логирования

- Содержимое файла **logging.conf**

```
log.console.level = debug
```

Настройка JSON логирования

- Содержимое файла **logging.conf**

```
log.console.level = debug
log.console.formatter = json
```

Мониторинг

- Нужен всем
- Стеки мониторинга
- Уведомления
- Доступность сервиса, нод, метрики железа
- Внутренние метрики:
  - Встроенный prometheus exporter
  - Внешний exporter - telegraf
  - Другие exporter’ы

Алеринг - на что реагировать

- Недоступность узлов кластера (в частности AMQP портов)
- Длина очередей
- Отсутствие консьюмеров на очередях
- Ределиверы
- Пересоздание каналов, соединений
- Настройка порога свободного места, его срабатывание
- *Память, место, CPU

Встроенный prometheus exporter

- Работает нативно, **“из коробки”**
- Метрики доступны по двум URL (порт **15692**)
  - **http://127.0.0.1:15692/metrics** - общие метрики
  - **http://127.0.0.1:15692/metrics/per-object** - “по объектам”
- **Не умеет в счётчики паблиша, доставки, ределиверов**
- Нужно ходить во все узлы кластера чтобы собрать всю картину
- Содержит много внутренних метрик процессов erlang

Встроенный prometheus exporter

- Содержимое файла **enabled_plugins**

```
[rabbitmq_prometheus,rabbitmq_management].
```

Встроенный prometheus exporter

- Открываем порт 15692 **docker-compose.yml**:

```
rabbitmq:
  image: rabbitmq:3.10.7-management
  hostname: rabbitmq
  restart: always
  environment:
    RABBITMQ_DEFAULT_USER: rmuser
    RABBITMQ_DEFAULT_PASS: rmpassword
    RABBITMQ_SERVER_ADDITIONAL_ERL_ARGS: -rabbit disk_free_limit 2147483648
  volumes:
    - ./rabbitmq:/var/lib/rabbitmq
    - ./enabled_plugins:/etc/rabbitmq/enabled_plugins
  ports:
    - 15672:15672
    - 15692:15692
```

Встроенный prometheus exporter

- Общие метрики **http://127.0.0.1:15692/metrics**

```
rabbitmq_alarms_free_disk_space_watermark 0
rabbitmq_disk_space_available_limit_bytes 2147483648

rabbitmq_connections_opened_total 1
rabbitmq_connections_closed_total 1
rabbitmq_channels_opened_total 1
rabbitmq_channels_closed_total 1
rabbitmq_queues_declared_total 1

erlang_vm_dist_recv_bytes{peer="rabbit@rabbitmq2"} 784631
erlang_vm_dist_recv_bytes{peer="rabbit@rabbitmq3"} 914168

erlang_vm_dist_send_bytes{peer="rabbit@rabbitmq2"} 371742
erlang_vm_dist_send_bytes{peer="rabbit@rabbitmq3"} 421456
```

Встроенный prometheus exporter

- По объектам **http://127.0.0.1:15692/metrics/per-object**

```
rabbitmq_queue_messages{vhost="/",queue="test1"} 1

rabbitmq_queue_consumers{vhost="/",queue="test1"} 0
```

Внешний prometheus exporter - telegraf

- **Легковесный**, на Golang
- Умеет в массу других сервисов
- **Умеет в счётчики паблиша, доставки, ределиверов**
- Собирает необходимые данные через любую ноду кластера
- Работает с RabbitMQ по **restapi (15672)**

Внешний prometheus exporter - telegraf

- Добавляем в **docker-compose.yml**:

```
telegraf:
  image: telegraf:latest
  hostname: telegraf-slerm
  restart: always
  volumes:
    - ./telegraf.conf:/etc/telegraf/telegraf.conf
  ports:
    - 9126:9126
  links:
    - rabbitmq
```

Внешний prometheus exporter - telegraf

- Содержимое **telegraf.conf**:

```
[global_tags]
[agent]
  interval = "10s"
  round_interval = true
[[outputs.prometheus_client]]
  listen = ":9126"
  expiration_interval = "60s"
[[inputs.rabbitmq]]
  url = "http://rabbitmq:15672"
  username = "rmuser"
  password = "rmpassword"
```

Внешний prometheus exporter - telegraf

- Несколько input, не обязательно

```
[[inputs.rabbitmq]]
  url = "http://rabbitmq1:15672"
  username = "rmuser"
  password = "rmpassword"
[[inputs.rabbitmq]]
  url = "http://rabbitmq2:15672"
  username = "rmuser"
  password = "rmpassword"
[[inputs.rabbitmq]]
  url = "http://rabbitmq3:15672"
  username = "rmuser"
  password = "rmpassword"
```

Внешний prometheus exporter - telegraf

- Интересные метрики **http://127.0.0.1:9126/metrics**

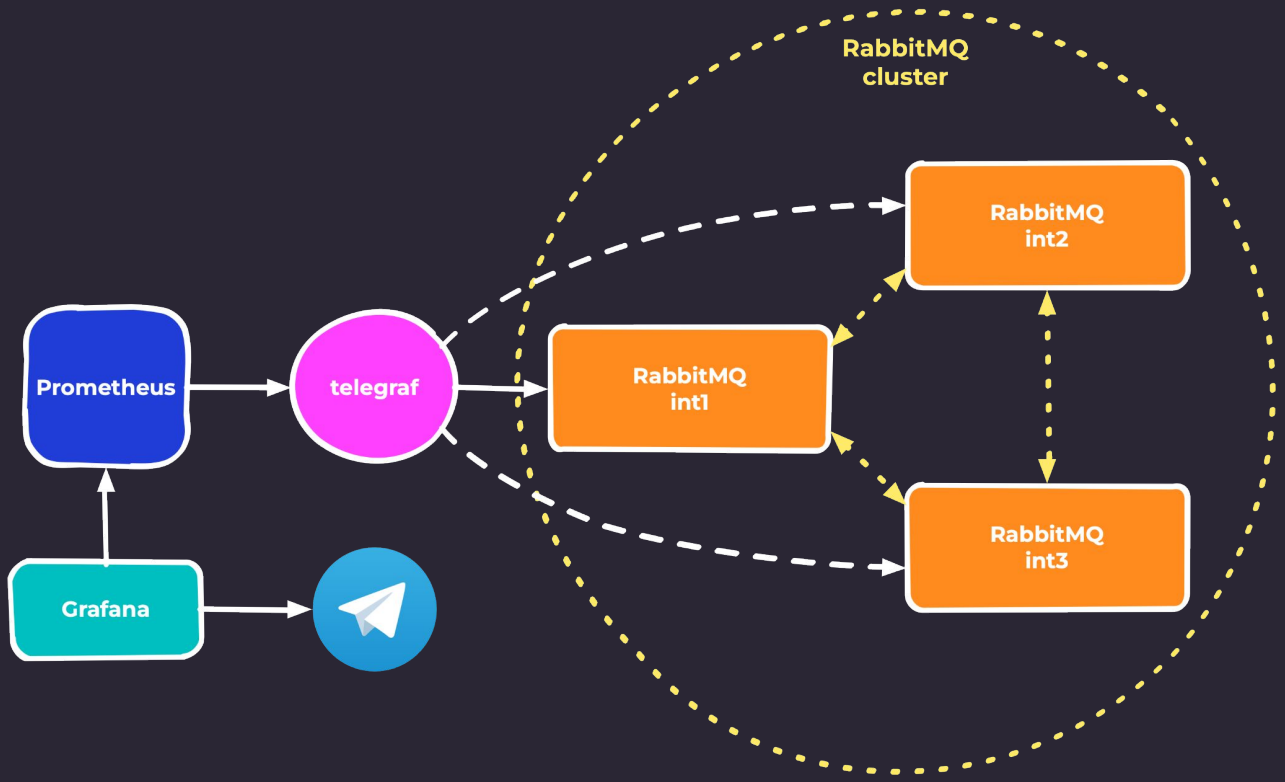
```
rabbitmq_node_disk_free_alarm - статус alarm для свободного места всех нод в кластере
rabbitmq_node_disk_free_limit - настроенный лимит порога свобоного места всех нод в кластере

rabbitmq_node_uptime - аптайм всех нод в кластере
rabbitmq_node_running - статус нод в кластере

rabbitmq_queue_slave_nodes - количество слейвов для каждой классической очереди
rabbitmq_queue_synchronised_slave_nodes - количество синхронизированных слейвов для каждой
классической очереди
```

Внешний prometheus exporter - telegraf

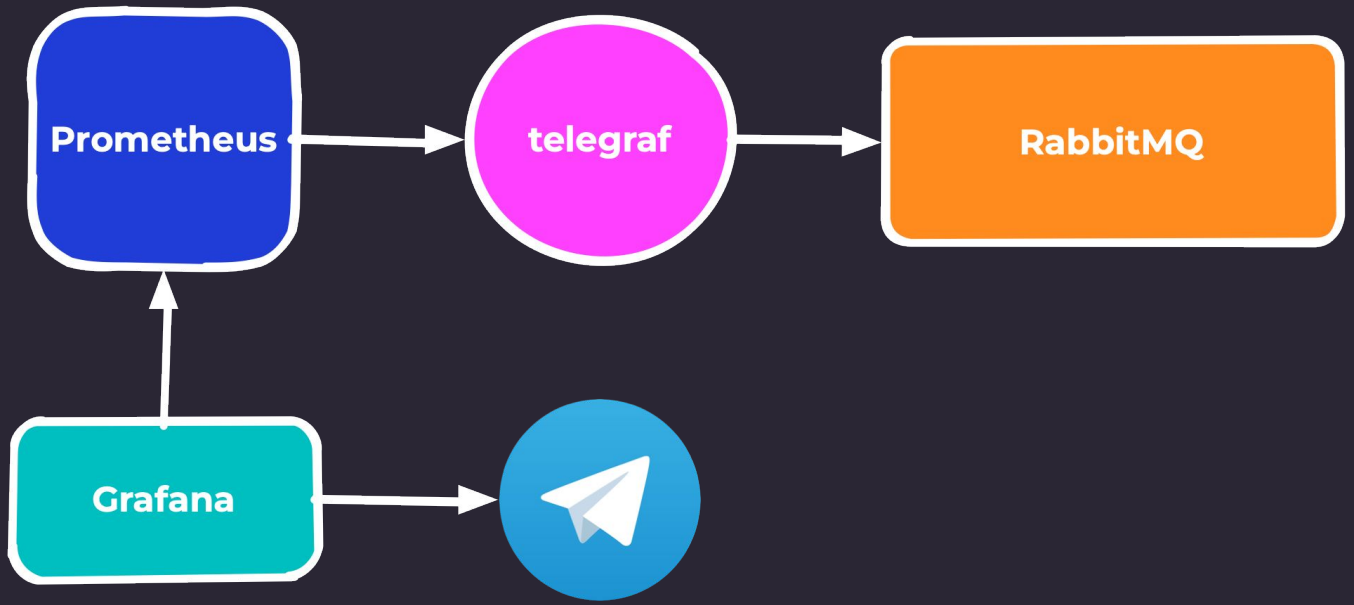
- Интересные метрики **http://127.0.0.1:9126/metrics**

```
rabbitmq_exchange_messages_publish_in - счетчик входящих для каждого exchange
rabbitmq_exchange_messages_publish_out - счетчик исходящих для каждого exchange

rabbitmq_queue_consumers - кол-во консьюмеров для каждой очереди
rabbitmq_queue_messages - кол-во сообщений для каждой очереди

rabbitmq_queue_messages_ack - счетчик ACK для каждой очереди
rabbitmq_queue_messages_redeliver - счетчик переделиверов для каждой очереди
rabbitmq_queue_messages_deliver_get - счетчик деливеров для каждой очереди
rabbitmq_queue_messages_publish - счетчик publish для каждой очереди
```

Стек мониторинга



Стек мониторинга



Демонстрация

Практическое задание

- Запустить **RabbitMQ** с сохранением стейта
- Настроить уровень логирования **debug**
- Добавить в окружение **telegraf**, настроить его.
- Проверить появление **метрик RabbitMQ** в telegraf
- Добавить в окружение **prometheus**, подключить к telegraf
- Добавить в окружение **grafana**, подключить к prometheus
- Создать дашборд, несколько полезных **графиков по метрикам**
- Настроить **алерты в телеграм**

Конец

А кто слушал - молодец