Р·СЕРВИС

Рекомендации по мониторингу

локально развернутых сред (On-Premise) Р-Сервис в любом из механизмов развертывания.

СОДЕРЖАНИЕ

1 ОБЩЕЕ ОПИСАНИЕ

В данном руководстве описаны рекомендации по мониторингу локально развернутых сред (On-Premise) Р-Сервис в любом из механизмов развертывания.

Р-Сервис рекомендуется мониторить на нескольких уровнях: доступность пользовательского интерфейса и программного интерфейса, состояние очередей, почтовых интеграций и службы формирования PDF[1]-файлов.

Для эксплуатации в Kubernetes базовый контур должен включать:

- проверки состояния/готовности (health/readiness probes);
- централизованные логи;
- метрики инфраструктуры;
- APM[2] и трассировку (tracing[3]);
- синтетические проверки ключевых пользовательских сценариев.

Данное руководство не затрагивает мониторинг нижележащих ресурсов: хостов виртуализации, узлов (нодов) Kubernetes и других компонентов инфраструктуры.

[1] PDF – универсальный формат электронных файлов, который сохраняет точное оформление документа (текст, шрифты, изображения, макет) независимо от компьютера, операционной системы или программы, где он открывается.

[2] APM – комплекс инструментов для отслеживания здоровья, доступности и скорости работы программных приложений.

[3] Tracing – ключевой компонент APM, который отслеживает точный путь и время выполнения конкретного пользовательского запроса по мере его прохождения через всю систему.

2 ДОСТУПНОСТЬ ПРИЛОЖЕНИЯ

2.1 Роль приложения (APP-роль)

До релиза 1.27.0 используются следующие маршруты для регулярных проверок (health-check):

- компонент app: /teapot возвращает строку "I am a teapot". Используйте данную регулярную проверку ТОЛЬКО в случаях, если ваш балансировщик/система мониторинга умеет проверять тело ответа;
- компонент nginx: _.gif.

После релиза 1.27.0 используются стандартизированные маршруты:

- Readiness check - /readyz - проверка готовности приложения, включая доступность базы данных;
- Health check - /healthz – базовая проверка живости процесса.

2.2 JOB-роль[4]

До релиза 1.27.0 для отложенных очередей (delayed-job) и обработчика входящей почты Clacks не доступны маршруты для регулярных проверок. Для следующих компонентов используются следующие маршруты:

- компонент pdf: /teapot возвращает строку "I am a teapot". Используйте данную регулярную проверку ТОЛЬКО в случаях, если ваш балансировщик/система мониторинга умеет проверять тело ответа;
- компонент sidekiq: / на отдельном порту 7433 возвращает строку "Alive!".

После релиза 1.27.0 используются стандартизированные маршруты:

- Readiness check - /readyz - проверка готовности приложения, включая доступность базы данных;
- Health check - /healthz – базовая проверка живости процесса.

2.3 REALTIME-роль[5]

До релиза 1.27.0 для обработчика очереди Sneakers не доступны маршруты для регулярных проверок:

- компонент FAYE: /teapot возвращает строку "I am a teapot". Используйте данную регулярную проверку ТОЛЬКО в случаях, если ваш балансировщик/система мониторинга умеет проверять тело ответа.

После релиза 1.27.0 используются стандартизированные маршруты:

- Readiness check - /readyz - проверка готовности приложения, включая доступность базы данных;
- Health check - /healthz – базовая проверка живости процесса.

2.4 Оповещения («алерты»[6])

Со стороны системы мониторинга рекомендуется настроить следующие оповещения (алерты) на доступность приложения:

- /healthz недоступен более 1-2 минут (у любой роли);

---

[4] JOB-роль – роль фонового обработчика, который выполняет асинхронные задачи из очереди.

[5] REALTIME-роль – роль, которая обеспечивает обслуживание запросов в момент их поступления.

[6] Алерт – мгновенное уведомление о важном событии, изменении или проблеме.

- /readyz возвращает ошибку более 1 минуты;
- внешний URL[7] системы на любое пространство недоступен;
- P95 времени ответа app выше 1-2 секунд в течение 5-10 минут;
- доля HTTP[8] 5xx выше 1-2% за 5 минут;
- рост HTTP 429.

2.5 Синтетические тесты

Рекомендуется проверять следующий сценарий:

1. Открыть страницу входа в пространство.
2. Проверить, что страница входа открывается и отвечает за приемлемое время (целевое время ответа в облачной модели предоставления программного обеспечения (SaaS) - <3с; в средах заказчика данный целевой срок согласно соглашению об уровне обслуживания (SLA) устанавливает сам заказчик внутри).

Расширенный сценарий так же будет включать:

1. Выполнить вход техническим пользователем.
2. Проверить успешный переход в интерфейс специалиста ИЛИ на портал самообслуживания.
3. Открыть список запросов.

[7] URL – адрес интернет-ресурса, который использует браузер, чтобы найти и открыть нужную страницу, файл или сервис.

[8] HTTP – набор правил обмена данными.

3 KUBERNETES[9] И КОНТЕЙНЕРЫ

Необходимо отслеживать базовый статус контейнеров/подов[10], их использование процессором/памятью (CPU/memory).

В Kubernetes необходимо отслеживать события, например, CrashLoopBackOff, ImagePullBackOff, OOMKilled.

3.1 Роль приложения (Роль APP)

Для роли APP важно отслеживать потребление процессора и памяти (CPU/memory) контейнерами app.

Важно понимать, что максимальное количество утилизируемых ядер равно количеству воркеров[11] контейнера APP. Не следует превышать количество воркеров, ведь это плохо скажется на времени ответа и не повысит производительность.

Каждый воркер в среднем использует в районе 1 Гб оперативной памяти. Рекомендуется закладывать 1.5 Гб оперативной памяти на воркер.

В оповещениях (алертах) ниже показаны относительные значения. Необходимо рассчитывать лимиты на основании 1 ядро и 1.5 Гб на воркер, как описано выше.

Оповещения (алерты) APP:

1. Потребление процессора (CPU) контейнером >80% в течение 5 минут.
2. Потребление оперативной памяти (RAM) контейнером >90% в течение 15 минут.
3. Количество желаемых реплик и доступных реплик отличается не более чем на 20%.

3.2 JOB-роль

Для JOB-роли важно отслеживать не столько потребление ресурсов, сколько рост очередей.

Для получения текущих счетчиков очередей можно использовать следующий запрос SQL[12]:

```
SELECT dj.queue, COUNT(*)
FROM delayed_jobs dj
WHERE dj.failed_at IS NULL
  AND dj.run_at <= NOW() + INTERVAL '5 MINUTE'
GROUP BY dj.queue;
```

[9] KUBERNETES – программная платформа с открытым исходным кодом для автоматизации развертывания, масштабирования и управления контейнеризированными приложениями.

[10] Поды – самые маленькие развертываемые вычислительные единицы, которые можно создавать и которыми можно управлять в Kubernetes.

[11] Воркер – отдельный процесс или сервис, который выполняет работу в фоне.

[12] SQL – язык запросов к базам данных.

Оповещения (алерты) JOB:

1. Любая очередь больше 1000 задач.
2. Возраст старой, самой ожидающей задачи:

- очередь срочных (urgent): старше 1-2 минут;
- очередь быстрых (fast): старше 5 минут;
- очередь нормальных (normal): старше 15 минут;
- очередь медленных (slow): старше 30-60 минут.

3. Неудавшиеся фоновые задачи (Failed jobs) растут непрерывно.
4. Ожидающие завершения задачи (Locked jobs) зависли больше, чем на 2 часа (7200 секунд).
5. Потребление процессора (CPU) контейнером >80% в течении 5 минут.
6. Потребление оперативной памяти (RAM) контейнером >80% в течении 15 минут.
7. Количество желаемых реплик и доступных реплик отличается не более чем на 20%.

3.3 REALTIME-роль

Оповещения (алерты) REALTIME:

1. Любая очередь больше 1000 задач.
2. Потребление процессора (CPU) контейнером >80% в течение 5 минут.
3. Потребление оперативной памяти (RAM) контейнером >80% в течение 15 минут.

4 БАЛАНСИРОВКА

Со стороны балансировщика нагрузки рекомендуется иметь следующие оповещения (алерты):

1. Нет здоровых бэкэндов.
2. Здоровых бэкэндов меньше 80% от ожидаемого количества.
3. 5xx от бэкэнда >1-2%.
4. P95 времени ответа APP >1-2 секунд.
5. P99 времени ответа сильно выше базового уровня.
6. Рост 429.
7. Ошибки TLS Handshake[13] растут.

[13] TLS Handshake – начальный этап установления защищённого соединения между клиентом и сервером, в ходе которого они договариваются о параметрах шифрования, проверяют сертификат и создают общие ключи для дальнейшей защиты данных.

5 БАЗА ДАННЫХ

База данных, как и в любом приложении, является критичной зависимостью. Необходимо отслеживать её доступность и производительность, ведь её деградация прямо влияет на доступность приложения.

В данном руководстве не рассматриваются оповещения (алерты) для базы данных, ведь архитектура БД зависит от инсталляции. Рекомендуется прорабатывать список оповещений (алертов) отдельно.

6 REDIS[14] И MEMCACHED[15]

Доступность Redis определяет возможность использования части функций, когда в то же время доступность Memcached определяет состояние кэширования.

Важно отметить, что без кэша может сильно вырасти нагрузка на APP-роль и БД как реляционную, так и аналитическую.

Оповещения (алерты) Redis:

1. Redis недоступен.
2. Потребление процессора (CPU) контейнером >70% в течение 5 минут.
3. Потребление оперативной памяти (RAM) контейнером >80% в течение 5 минут.
4. Вытеснение ключей (Evictions) > 0 на постоянной основе.
5. Рост задержки отклика (redis latency).

Оповещения (алерты) Memcached:

1. Memcached недоступен.
2. Потребление процессора (CPU) контейнером >70% в течение 5 минут.
3. Потребление оперативной памяти (RAM) контейнером >80% в течение 5 минут.

[14] REDIS – нереляционная СУБД с открытым исходным кодом.
[15] MEMCACHED – система кэширования данных в оперативной памяти.

7 RABBITMQ[16]

Доступность RabbitMQ определяет работу функции реального времени (Realtime) в приложении.

Важно отметить, что в связи с реализацией при недоступности rabbitmq app-воркеры приложения могут отвечать на 3 секунды дольше – это обусловлено наличием «автоматического выключателя» (circuit breaker), которому необходимо открыться на первых 5 неудавшихся запросах.

Оповещения (алерты) RabbitMQ:

1. RabbitMQ недоступен.
2. Сработала «тревога о памяти» (Memory alarm).
3. Сработал «дисковый триггер» (Disk alarm).
4. Любая очередь растет непрерывно 5 минут или превышает какой-то базовый порог.
5. Параметр messages_unacknowledged растет 5-10 минут или долго не снижается.

[16] RABBITMQ – распределённый и горизонтально масштабируемый брокер сообщений.

8 OPENSEARCH[17]

Доступность OpenSearch определяет возможность использования большинства функций для работы отображений и поиска. Важно заметить, что без opensearch будут использоваться «резервные запросы» (fallback-запросы) в реляционную БД, что вызовет повышенную нагрузку на неё.

Оповещения (алерты) OpenSearch:

1. Метрика состояния здоровья кластера (Cluster health) не зеленая:

- Желтая дольше 10-15 минут;
- Возникла красная.

2. Дисковые пороги (Disk watermarks) превышены (использование дискового пространства (disk usage) выше 80-85%).

3. Растут отклоненные запросы (Rejected requests).

[17] OPENSEARCH – высокопроизводительная поисковая и аналитическая система с открытым исходным кодом.

9 ЛОГИ И ОШИБКИ ПРИЛОЖЕНИЯ

Рекомендуется централизованно собирать stdout/stderr[18] всех контейнеров. Логи должны коррелироваться по идентификатору трассировки (trace id).

Критичные категории логов:

1. Исключения (Rails exceptions) или прочие HTTP 5xx.
2. Ошибки обработчиков JOB.
3. Ошибки Redis/OpenSearch/PostgreSQL/RabbitMQ.
4. Медленные запросы (Slow SQL (Slow queries)).
5. События лимита запросов (rate limit).

[18] stdout/stderr – стандартные потоки вывода.

10 РАСПРЕДЕЛЕННАЯ ТРАССИРОВКА

С версии 1.26 в приложении доступна распределенная трассировка (Distributed Tracing) на базе OpenTelemetry. Она позволяет собирать информацию о каждом запросе, проходящем через app и задачах delayed job, исходящем из этого запроса.

Из-за стандартизации OpenTelemetry возможна интеграция с балансировщиками нагрузки.

На основе информации распределенной трассировки является правильным наблюдение за:

1. Задержка (Latency) (P50, P95, P99).
2. Общая пропускная способность запросов (request throughput).
3. Доля ошибок (Error rate).
4. Медленные конечные точки (Slow endpoints).

Для информации по настройке работы OpenTelemetry необходимо завести запрос вендору.