

Постоянный мониторинг и совершенствование

Ведение постоянного мониторинга существенно для осознания того, каким образом работа приложений и системы могут повлиять на конечного пользователя. Изучая логи и информацию, становится возможным уловить, каким образом модификации кода воздействуют на клиентов. В эру круглосуточного доступа к услугам и постоянных обновлений, как приложений, так и систем, активное наблюдение является неотъемлемым. Это позволяет заблаговременно реагировать, формируя уведомления и проводя анализ в реальном времени.

Ваше стремление к усовершенствованию практик DevOps может быть улучшено отслеживанием различных метрик. Вот несколько примеров метрик, связанных с DevOps:

- **Масштаб изменений:** Это отражает количество разработанных задач, объем нового кода и количество исправленных дефектов.
- **Частота развертываний:** Этот параметр отражает, насколько часто команда выполняет развертывание приложений. Он должен быть стабилен или показывать тенденцию к увеличению.
- **Время от разработки до развертывания:** Временной промежуток между началом разработки и окончанием развертывания может выявить затруднения на различных этапах процесса реализации.
- **Процент неудачных развертываний:** Эта метрика, включая количество неудачных развертываний, должна быть минимальной. Она следует рассматривать совместно с масштабом изменений. Проверьте потенциальные уязвимые места, если масштаб изменений небольшой, а число неудачных развертываний высоко.
- **Доступность:** Измерьте, сколько релизов вызвали сбои, которые могли привести к нарушению SLA. Каково среднее время недоступности приложения.
- **Объем жалоб клиентов:** Количество поданных клиентами жалоб может отражать качество вашего продукта.
- **Процент изменения объема пользователей:** Количество новых пользователей и сопутствующее увеличение трафика может быть индикатором для масштабирования вашей инфраструктуры в соответствии с нагрузкой.

После того как сборка была развернута в производственной среде, непрерывный контроль за ее эффективностью становится обязательным.

Пример мониторинга

1. Настройка и деплоймент:

Допустим, у нас есть веб-приложение, развернутое в Kubernetes, используя CI/CD Pipeline через Jenkins. После деплоймента, мы хотим настроить мониторинг приложения.

2. Интеграция с Prometheus и Grafana:

Для мониторинга, мы будем использовать Prometheus для сбора метрик и Grafana для визуализации данных.

- **Prometheus** конфигурируется для сбора метрик от веб-приложения и Kubernetes.
- **Grafana** настраивается для отображения дашбордов с ключевыми показателями производительности приложения.

3. Постоянный мониторинг:

С Prometheus и Grafana, команда DevOps может мониторить:

- Загрузку CPU и использование памяти контейнеров.
- Количество запросов, ошибок и времени ответа от веб-приложения.
- Состояние и доступность сервисов Kubernetes.

4. Анализ и оптимизация:

На основе данных, собранных через мониторинг, команда может:

- Идентифицировать узкие места и оптимизировать производительность.
- Определить поведение системы под нагрузкой и внести коррективы при необходимости.
- Исправлять обнаруженные ошибки и улучшать код приложения.

5. Алертинг:

С Prometheus, можно настроить алерты для уведомления команды о любых аномалиях или проблемах, таких как:

- Высокая загрузка CPU.

- Недостаток памяти.
- Ошибки в приложении.

6. Обратная связь и совершенствование:

На основе обратной связи от мониторинга, команда DevOps может реализовывать изменения для устранения проблем производительности и стабильности, а также проводить непрерывное совершенствование инфраструктуры и приложения.

7. Собрание обратной связи от пользователей:

Особенно важно собирать обратную связь от пользователей о проблемах, багах или улучшениях, которые они хотели бы видеть. Это может включать в себя системы отслеживания ошибок, аналитику использования и опросы пользователей.