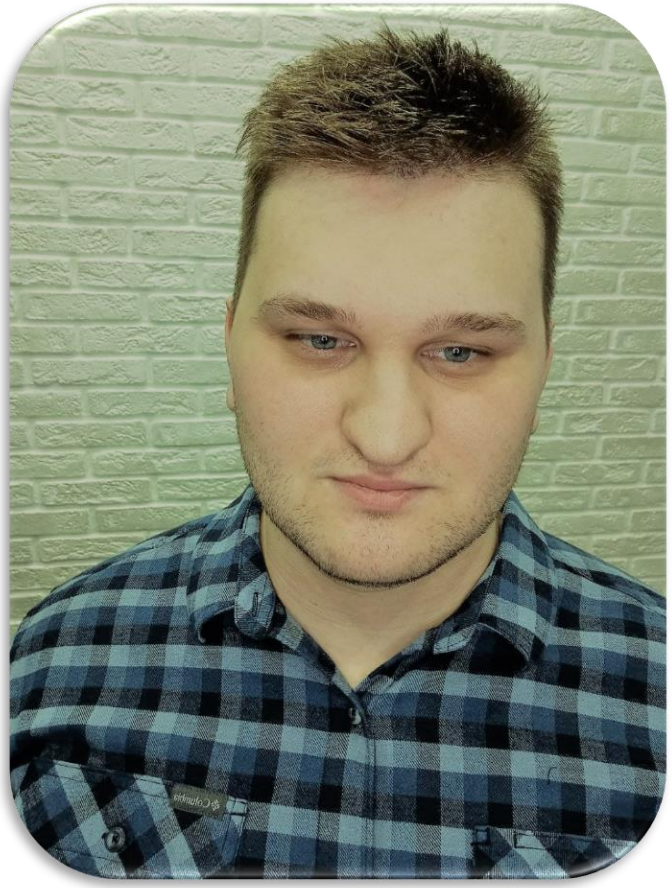
DGTL
Raiffeisen

Перестаньте писать пайплайны, займитесь процессами!

Харченко Евгений
SCL DevOps

Кто говорит



- В университете занимался мелкой автоматизацией, копался с Windows Server
- Работал в организации, представляющей BaaS-решения для UK (United Kingdom) в роли L1-специалиста
- В Райффайзенбанке начинал с инженера техподдержки ServiceDesk
- После в банке работал как Leading Automation Engineer (DevOps)
- Сейчас Senior Community Lead DevOps

План доклада

1

Постановка проблемы – почему тема доклада важная?

2

Обзор постмортемов – ревью ситуаций и их разбор

3

Решения – как все-таки строить процессы?

DGTL Raiffeisen

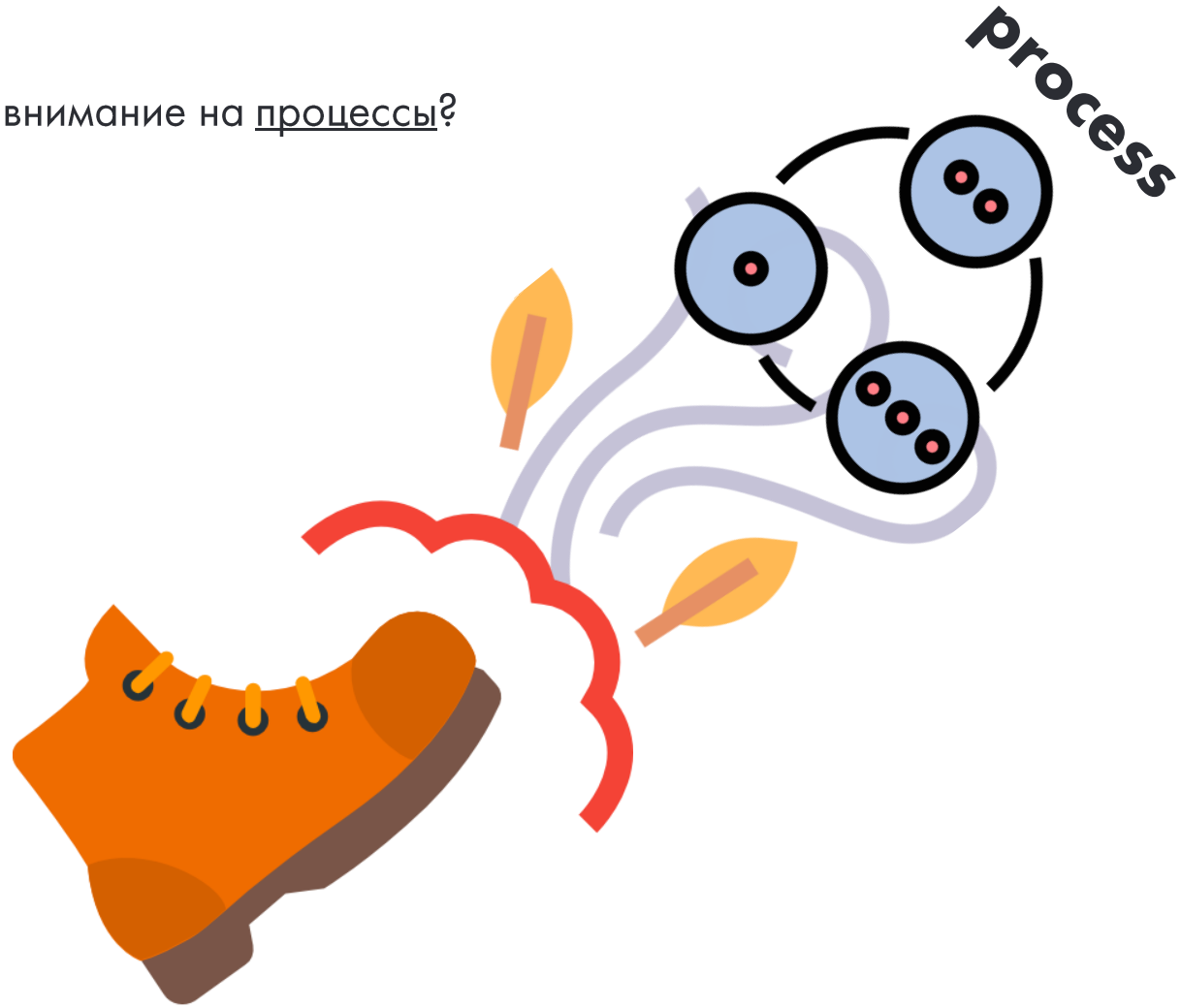
Постановка проблемы

АКТ 1

Почему это важно?

Почему “хватит писать пайплайны” и стоит обратить внимание на процессы?

- Много коммуникаций
- Процесс сложнее, чем пайплайн
- Технология — не серебряная пуля
- Искажение понимания профессии в индустрии
- Профанация



“Очень важно знать, откуда мы пришли, потому что если не знаешь откуда ты, то не знаешь где ты, а если ты не знаешь где ты, то не знаешь и куда идёшь. А если ты не знаешь, куда идёшь, то, скорее всего, идёшь не туда.”

Терри Пратчетт, “Платье цвета полуночи”

Развитие DevOps

Как развивалась индустрия

2007

1

Причина

- Патрик Дебуа в Бельгии переносит крупный центр обработки данных, занимается тестированием.
- Часто взаимодействует с development и operations и видит в этом сложности, прежде всего – в коммуникации

Катализатор

- Agile-конференция - Эндрю Шафер провел сессию под названием «Гибкая инфраструктура»
- Дебуа - единственный участник сессии
- Диалог о преодолении разрыва и формировании группы администрирования гибкой системы

2

2008

2009

3

Рождение DevOps

- Вдохновение докладом Джона Олспоу и Пола Хаммонда «10 развертываний в день: сотрудничество Dev и Ops в Flicker»
- Создание собственной конференции DevOpsDays
- Термин «DevOps» стал вирусным в мире разработки

2014 – первый State of DevOps Report (DORA)

Появление DevOps в России

- 2015 – RootConf ex. DevOpsConf
- 2017 – DevOpsDays Russia
- 2017 – DevOops
- 2018 – DevOpsConf

4

2015- 2018

Что происходит сейчас

- Возникновение родственных практик
- Много мероприятий в индустрии
- Большой спрос на инженеров
- Сформированное сообщество

5

2021

Как инженер стал DevOps'ом?

Эволюция профессии и её понимания

1 - 2007

2 - 2008

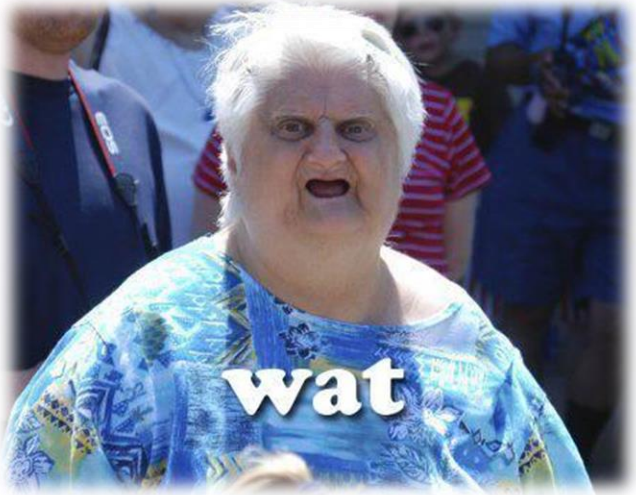
3 - 2009

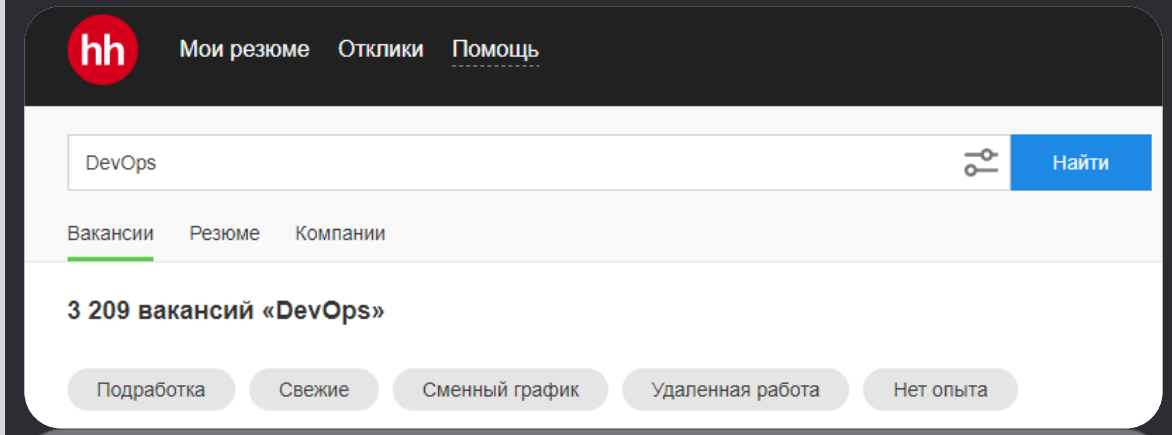
4 - 2015 - 2018

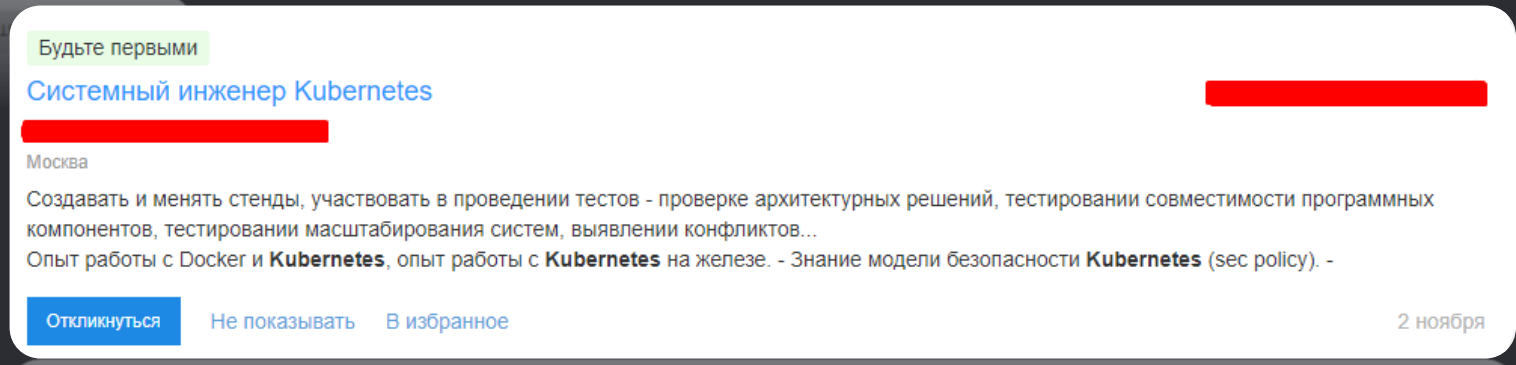
5 - 2021

Ничего не предвещало беды

- Компании начинают называть DevOps профессией
- Смысл DevOps-движения искажается в пользу технологий
- Выстраивание процессов уходит на задний план
- От инженеров ожидают разработки пайплайнов в первую очередь
- Появление DevOps-инженеров
- Появление Kubernetes-инженеров



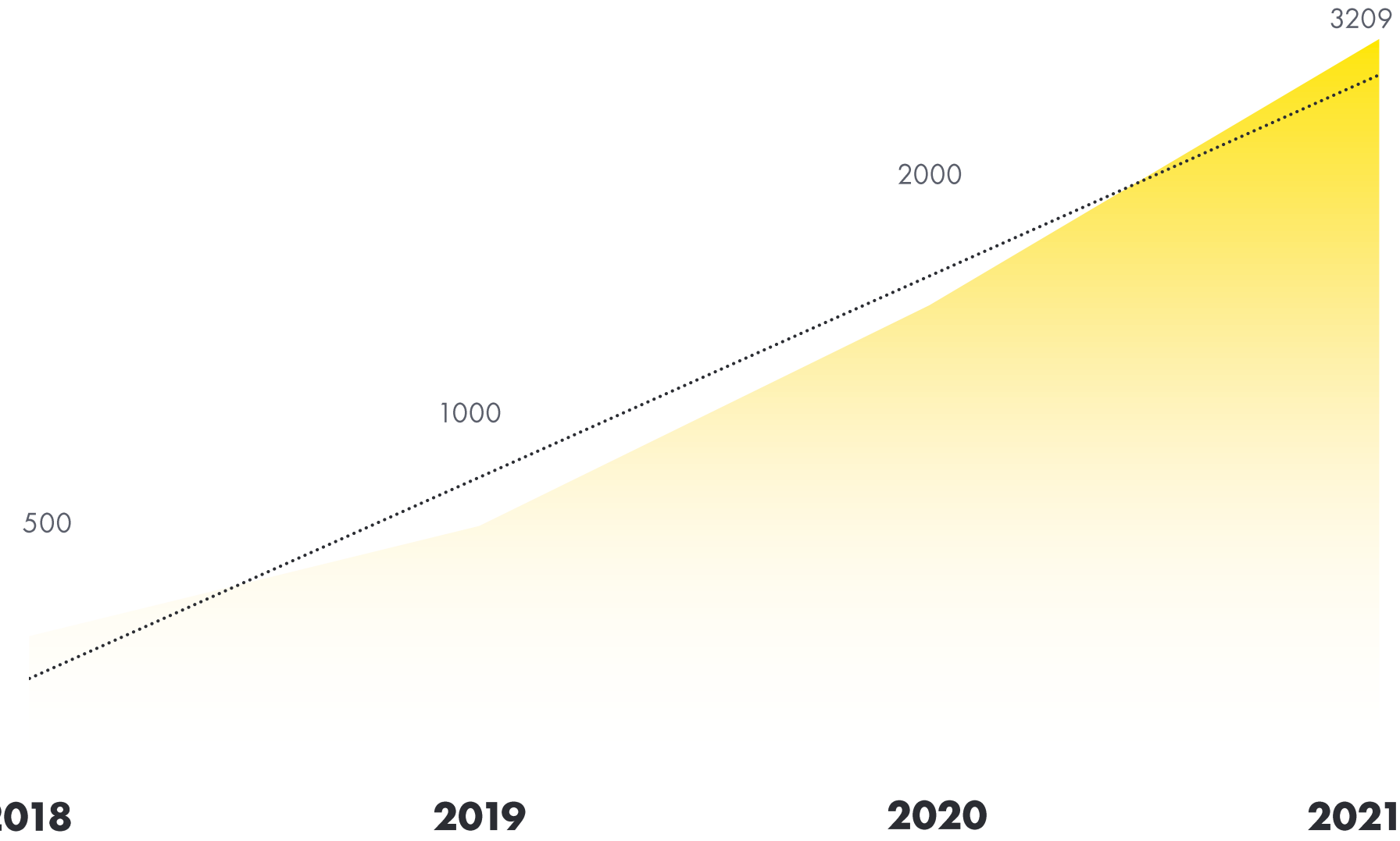




Требования:

- Опыт администрирования OS Linux не менее 3-х лет;
- Опыт эксплуатации систем виртуализации, контейнеризации;
- Знание сетевой архитектуры;
- Опыт работы с веб-серверами;
- Опыт работы с нагруженными БД MySql (настройка, репликация, оптимизация конфигурации);
- Опыт работы с системами управления конфигурациями;
- Написание shell скриптов для вас не должно быть проблемой. Опыт программирования на других скриптовых языках будет плюсом;
- Способность принимать самостоятельные решения и не терять самообладания в критических ситуациях;
- Общение с нашими партнерами (как по e-mail, так и голосом) для решения текущих проблем. Для вас это не должно быть усилием над собой.

Рост спроса на “DevOps’ов”



О процессах

Зачем я должен этим заниматься?
Пусть это делает Scrum Master или Tech/Team Lead!

О роли процессов, и почему командам/организациям следует заниматься процессами?

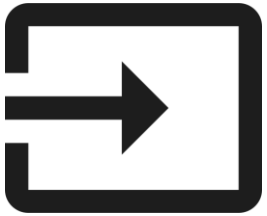
Почему выстраивание процессов важно?

- В современном IT стало больше потребности в том, чтобы договариваться
- Всей команде следует строить процессы вместе
- DevOps как класс – агент изменений
- Инженер, использующий DevOps-практики, несёт изменения
- DevOps is about communication, а общение - это прежде всего люди
- Там, где есть люди, надо строить процессы и упорядочивать хаос
- Помнить: мы делаем это для того, чтобы ускорять доставку ценности клиенту
- Повторить!

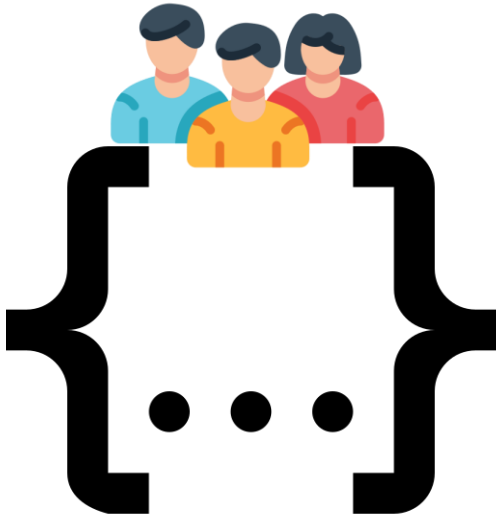
Но что же есть процесс?

Попробуем сравнить процесс с функцией

Функция может что-то принимать

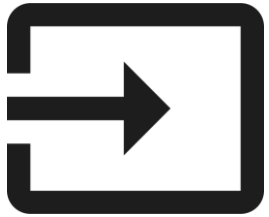


Команда



Совокупность взаимосвязанных действий, которые по определённой технологии преобразуют ввод в вывод в виде ценности

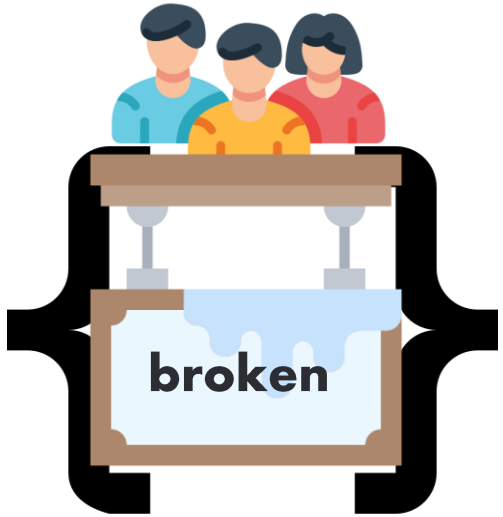
Функция может что-то отдавать



Почему сломанный процесс это плохо?

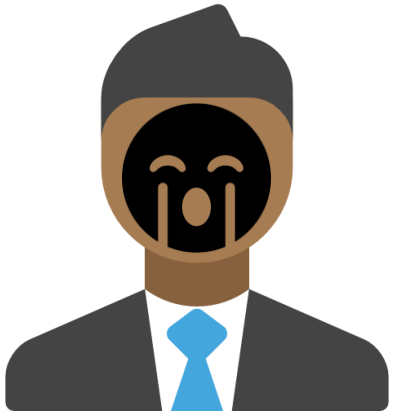
А что если ваша функция не работает?

Команда



Совокупность взаимосвязанных действий, которые по определённой технологии преобразуют ввод в вывод в виде ценности

Бизнес



Не получает ценности

Резюмируем первую часть

Что мы узнали из всего этого?

- Не работающий процесс = Не работающая функция = Потери бизнеса
- Процесс – это важно, и над ним работает вся команда
- DevOps был задуман для стирания барьеров в коммуникациях и ускорения
- DevOps не постигнет тебя, только ты сам можешь постичь его

DGTL Raiffeisen

Postmortem review

AKT 2

Кейс #1

Опыт GitLab – **точечная ситуация**

Сбой

- 31 января 2017 года - сбой онлайн-сервиса GitLab.com

Ситуация

- Случайное удаление данных с основного сервера базы данных

Результат

- В результате этого инцидента сервис GitLab.com был недоступен в течение многих часов

Потери

- Была утрачена часть данных без возможности восстановления

about.gitlab.com/blog/2017/02/10/postmortem-of-database-outage-of-january-31/

Incident timeline

1. 31 января инженер начал настройку нескольких серверов PostgreSQL в тестовой среде. Работа над задачей

2. ± 17:20 UTC: Перед тем, как начать эту работу, инженер сделал снимок LVM production DB и загрузил его в stage среду

3. ± 19:00 UTC: GitLab.com начинает испытывать увеличение нагрузки на базу данных из-за того, что, как команда GitLab подозревает, было увеличение спама

4. ± 23:00 UTC: Из-за увеличения нагрузки процесс репликации slave сервера PostgreSQL начала запаздывать. Репликация завершилась неудачно

   Один из инженеров обратился к slave серверу db и удалил каталог данных, а затем запустил pg_basebackup. К сожалению, pg_basebackup зависал, не давая значимого вывода, несмотря на установленную опцию --verbose

   Чтобы решить эту проблему, инженеры решили временно увеличить max_wal_senders со значения по умолчанию 3 до 32. При применении настроек PostgreSQL отказывался перезапускаться

5. ± 23:30 UTC: один из инженеров считает, что, возможно, pg_basebackup создал какие-то файлы в каталоге данных PostgreSQL slave сервера во время предыдущих попыток его запуска. Хотя обычно pg_basebackup в этом случае выдает ошибку, соответствующий инженер не был уверен, что происходит.

   Позже другой инженер (которого в то время не было рядом) обнаружил, что это нормальное поведение: pg_basebackup будет ждать, пока master сервер начнет отправлять данные репликации. К сожалению, это не было четко задокументировано ни в наших технических модулях, ни в официальном документе pg_basebackup.

   Пытаясь восстановить процесс репликации, инженер приступает к стиранию каталога базы данных PostgreSQL, ошибочно полагая, что они делали это на slave сервере. К сожалению, вместо этого этот процесс был выполнен на основном сервере. Инженер прекратил процесс через секунду или две, заметив свою ошибку, но к этому моменту уже было удалено около 300 ГБ данных.

   Надеясь, что они смогут восстановить базу данных, задействованные инженеры отправились искать резервные копии базы данных и попросили помощи по Slack. К сожалению, процесс поиска и использования резервных копий полностью не удался.

Разбираемся в причинах инцидента

Чтобы проанализировать первопричину этих проблем, мы воспользуемся методом под названием «5 why?».

Мы разделим инцидент на 2 основные проблемы:

GitLab.com не работает и восстановление GitLab.com занимает много времени.

1 Проблема 1: GitLab.com не работал около 18 часов

- Почему был недоступен GitLab.com?
- Почему был удален каталог базы данных?
- Почему остановилась репликация?
- Почему увеличилась нагрузка на базу данных?
- Почему было запланировано увольнение сотрудника GitLab?

2 Проблема 2: восстановление GitLab.com заняло более 18 часов

- Почему восстановление GitLab.com заняло так много времени?
- Зачем была нужна промежуточная база данных для восстановления GitLab.com?
- Почему мы не могли переключиться на вторичный хост базы данных?
- Почему мы не могли использовать стандартную процедуру резервного копирования?
- Почему процедура резервного копирования завершилась незаметно?
- Почему не были включены моментальные снимки дисков Azure?
- Почему процедура резервного копирования не проверялась регулярно?

Ключевые проблемы в процессе работы команды

Разберёмся в ключевых проблемах в процессе при данной аварии

1. Проблема 1: GitLab.com не работал около 18 часов
2. Проблема 2: восстановление GitLab.com заняло более 18 часов

- Автоматизация – потребовалось ручное вмешательство при восстановлении DB
- Observability – не достаточно широкая обзорность происходящего с инфраструктурой
- Проработка DRP – медленные VM с копией промежуточной DB
- Ответственность за резервное копирование – никто не отвечал за это

Кейс #2

Опыт продуктовой команды Raiffeisenbank – **рутинные ситуации**

Описание ситуации

Сложная поддержка

- Deploy приложения и устранение уязвимостей только в нерабочие часы – соответственно, овертайм для Operations

Инфраструктура

- Отсутствие демо-окружения для демонстрации клиентам

Коммуникации

- Токсичное общение в рабочем чате команды

Тех. долг

- Приоритет бизнеса в выпуске фич, перед всем остальным
- Отсутствие документации по эксплуатации ПО (Runbooks)

Симптомы

Основные причины которые привели к проблемам

1 Сложная поддержка

- Архитектура приложения не рассчитана на бесшовное обновление
- Бизнес не выделяет под доработку тех. долга ресурсы – нужно договорится!

2 Инфраструктура

- Необходимо проработать нестандартное решение с отделом безопасности и отделом сетевых технологий, но в продукте у вас нет на это прав
- Конфликт бэклогов платформы и продукта

3 Коммуникации

- Низкая вовлечённость менеджмента команды в поддержание культуры коммуникация

4 Тех. долг

- В продуктовой команде Tech Lead'у необходимо "продавать" овнеру уменьшение тех долга!

Ключевые проблемы в процессе работы команды

Разберёмся в ключевых проблемах в процессе при рутинной работе

1. Проблема 1: Сложно обновлять и поддерживать продукт
2. Проблема 2: Трудно договорится с бизнесом

- Культура "не ищи виновного" – необходимо налаживать соц. связи
- Ты сделал это, ты запускаешь это! – больше вовлечённости всей команды в процесс эксплуатации
- Коммуникации – взаимодействие с платформой для внедрения продукта и создание соглашений в команде

Резюмируем вторую часть

Что мы узнали из всего этого?

- Коммуникации – корень большинства проблем в процессах
- Автоматизация – автоматизируйте всё, человеку свойственно совершать ошибки
- Документация – документировать то, что важно, работу в “узких” местах, командные соглашения
- Один в поле не воин – вся команда ответственна за сервис, который крутится в production, не только ops

DGTL Raiffeisen

Как всё-таки строить процессы?

АКТ 3

DevOps практики – о чем мы позабыли?

Истоки

Культурные

- DevOps – это про коммуникации
- Культура "не ищи виновного"
- Распространяй знания, ломай барьеры

Процессные

- Проверяй гипотезы быстро и отбрасывай их
- Точка истины - VCS
- Сервера – скот, не домашние животные
- Ты сделал это, ты запускаешь это!

Технические

- Инфраструктура как код
- Автоматизируй всё
- Измеряй всё
- Логирование – важно!
- Идемпотентность
- CI/CD

DevOps практики – о чем мы позабыли?

Истоки

Культурные

- DevOps – это про коммуникации
- Культура "не ищи виновного"
- Распространяй знания, ломай барьеры

Процессные

- Проверяй гипотезы быстро и отбрасывай их
- Точка истины - VCS
- Сервера – скот, не домашние животные
- Ты сделал это, ты запускаешь это!

Технические

- Инфраструктура как код
- Автоматизируй всё
- Измеряй всё
- Логирование – важно!
- Идемпотентность
- CI/CD

С чего начать в построении процесса?

Определить, где мы есть, и какие у нас точки роста?

Культурные

- Как и насколько эффективно мы взаимодействуем?
- Что мы делаем, если ошиблись? – Есть ли у нас культура постмортемов?
- Как и насколько эффективно мы обмениваемся опытом внутри?

Процессные

- Насколько быстро мы готовы тестировать новые идеи и готовы ли вообще?
- Где мы храним исходный код?
- Какой у нас подход к управлению средами?
- Готовы ли каждый член команды быть ответственным за то, что он сделал?

Технические

- Где мы храним знания о нашей инфраструктуре?
- Всё ли у нас автоматизировано?
- Измеряется ли всё, что касается моего проекта/продукта/системы?
- Есть ли у нас логирование, а если есть насколько оно информативно?
- Идемпотентны ли наши внедрения/изменения ?
- Реализован ли CI/CD и удовлетворят ли потребности команды?

Что может помочь при выстраивании процесса?

Какие технологи, методики, фрэймворки могут нам помочь для построения процесса?

Культура + Процесс

- Agile ритуалы
- Тренинги + Обучения для организации/команды
- Культура постмортемов
- Starmap
- IT Standards
- Разрабатываем исключительно через VCS
- Работа над mindset'ом

Техника

- Всевозможные инструменты, можно подсмотреть в:
  - CNCF Landscape - landscape.cncf.io
  - Technology radar - thoughtworks.com/radar
  - digital.ai - digital.ai/periodic-table-of-devops-tools
  - devops.com

Roadmap внедрения DevOps в процессы с нуля

Разработайте DevOps-стратегию

Интеграция технологического ландшафта + CI/CD

Мониторинг производительности приложения

Презентуйте идею DevOps

Используйте контейнеризацию

Больше QA и выравнивания с Dev

Итого

Немного выводов и советов

- Знать откуда вы и ради чего вы это делаете?
- Всегда ищите точки роста, у вашей команды, продукта/проекта!
- Помнить чужие ошибки и делится своим опытом!
- DevOps про доверие и тесное взаимодействие!
- Каждая игра, часть большей игры, думайте о продукте целиком, а не только о своей зоне ответственности!
- Процессы важнее инструментов!

DGTL Raiffeisen

Перестаньте писать пайплайны, займитесь процессами!

LinkedIn

Telegram

Raiffeisen