ИСТОРИЯ УСПЕХА, ИЛИ «DEV+DEVOPS+OPS»

Сергей Бердников
Артем Каличкин
ЗАО «Золотая Корона», ЦФТ (Центр Финансовых Технологий)

О компании

Банковские карты
Денежные переводы
Кредитные и предоплаченные карты
Транспортные карты
Погашение кредитов

Золотая Корона

ЦФТ ЦЕНТР ФИНАНСОВЫХ ТЕХНОЛОГИЙ

Депозиты
Кредиты
Карты
Учетное ядро
Интеграция
Отчетность

Федеральная система «Город»

Faktura.ru

ДБО Faktura.ru

Подарочные карты

Процессинговый центр «КартСтандарт»

Программы лояльности

Что было, наше наследие, наша техника

- *Собственные ЦОДы,*
- *SPARC Solaris only,*
- *Oracle DataBase, PL\SQL,*
- *Java, много Java, только Java,*
- *«Теплый» резерв*

Наши процессы до DevOps

- ITIL, очень много заявок
- ITIL, очень много регламентов
- ~200 человек на поставку
- IT OPS, работающие по инструкции
- Попытки CMDB

- Исходная формула

Итак, формула:

Delivery team = (DEV) + (ПА + UNIX + DBA);

Предпосылки трансформации

- *Много человеческих ошибок при передаче изменений,*
- *Постоянный накал страстей «найди виноватого»,*
- *Скорость выноса новых модулей до 3 недель,*
- *Единые точки отказа(только вертикальное масштабирование), никакой балансировки,*
- *Плановый простой во время обновлений ~2 часа.*

Цели былинных времен из архива Confluence:

DevOps в ТД ДП. Эталонная целевая модель

Денежные переводы Техническая дирекция

1 Цель которую хочется достичь: 3
2 Состояние, которого хочется достичь: 4
3 Прежде всего блок уже инициированных тем, которые нужно довести до конца. 5
4 Далее, ключевые отличия эталонной модели измененного Dev от сегодняшнего Dev: 6
5 Далее, ключевые отличия модели выпуска версии и обновления боевой, в отличии от сегодняшнего Release: 7
6 Далее, ключевые отличия эталонной модели Ops от сегодняшнего Ops: 8
7 130806 План DevOps в ТД ДП до конца года 9

4 Далее, ключевые отличия эталонной модели измененного Dev от сегодняшнего Dev:

1. У всех видов разработки должен быть инструмент отладки до передачи кода тестировщикам

2. И на разработческой, и на тестовой, и на проде, и вообще везде, ВКЛЮЧАЯ БОЕВУЮ - полностью аналогичная схема развертывания, со всеми нюансами и потрохами. Должна быть реализована на практике парадигма "коммит кода на боевую". Т.е. конечно не на работающую боевую, но код с первой минуты жизни должен жить в окружении, 100% аналогичном боевой конфигурации и схеме развертывания

3. На проде нужно обеспечить 100% эмуляцию всех внешних запросов во всех вариациях (развитие технологии слепков, которая так видимо и не зажила) по всем направлениям, которые представляют ценность для бизнеса. И модель должна быть авто-расширяемая. Прогон всех видов запросов при каждом ночном цикле авторегрессии на боевой.

4. Участие Отдела Эксплуатации в рабочих группах Отдела Производства. Экспертиза решений с точки зрения не функциональных требований, требований к сборкам и пакетам, обновление "рецептов" деплоя сразу при определении пакета. Вообще другая роль ОЭ, ОЭ как источник требований боевой архитектуры и особенностей схем развертывания и автоматического релиза. Т.е. ОЭ со 2л ОС тоже участвуют в разработке РЕШЕНИЯ. Это сейчас трудно представить, но это эталонная модель. Более того - участие ОЭ в рабочих группах решает и задачу изменения культуры - все всегда должны чувствовать себя НА БОЕВОЙ. Нет уютной песочницы, где можно расслабиться и делать ошибки.

5. Динамически и легко расширяемая авторегрессия. Каждый баг на боевой должен закрываться доработкой существующего или разработкой и включением в ночной прогон нового автотеста, проверяющего конкретно эту ситуацию. Организационно вообще после этого нужно вводить правило - повторение дефекта на боевой - нет премии и баста. Ошибаться МОЖНО, повторять ошибки НЕ ДОПУСТИМО

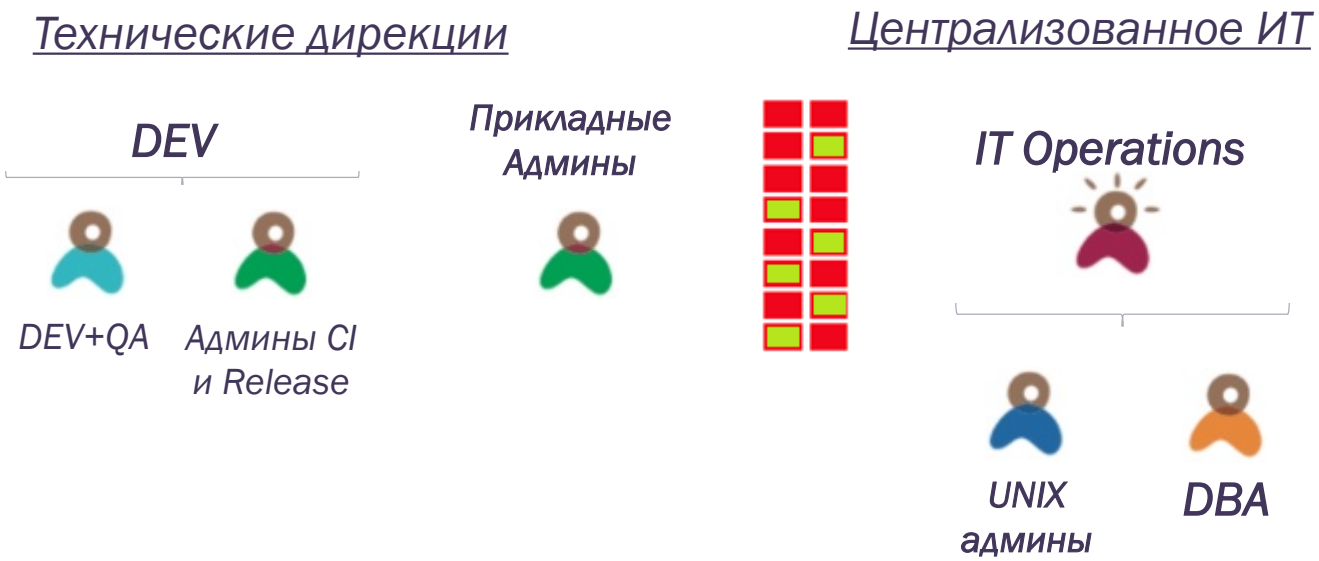
Аргументы против

- *Автоматика все сломает!*
- *Снижается уровень безопасности!*
- *Большой оверхед, нужно писать код программы, а потом еще какой-то код...*

Первые шаги

- *Решены проблемы с передачей конфигурации,*
- *Вынос новых модулей Неделя +,*
- *Доверие между DEV и Прикладными администраторами,*
- *Программные интерфейсы взаимодействия ПА с остальным ИТ.*

- *Формула Beta*

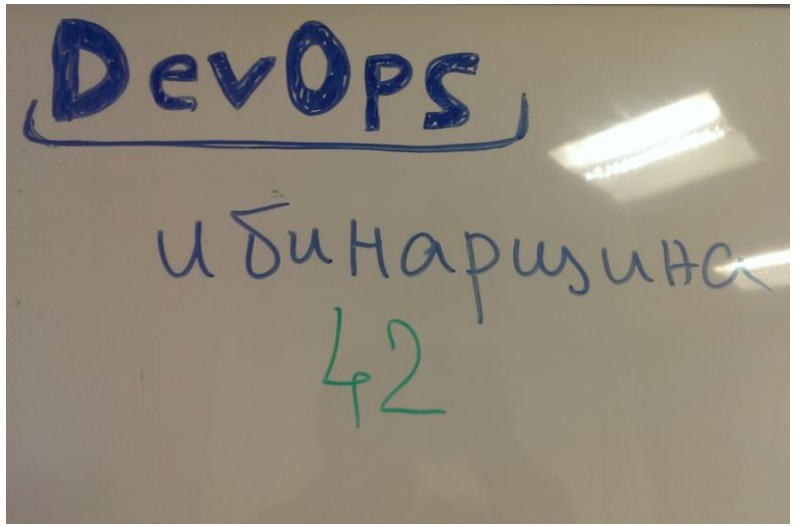


Итак, формула Beta:

Delivery team = [(DEV + devПА) + ПА] + [UNIX + DBA];

Муки выбора, тупняк, техника, стек ПО

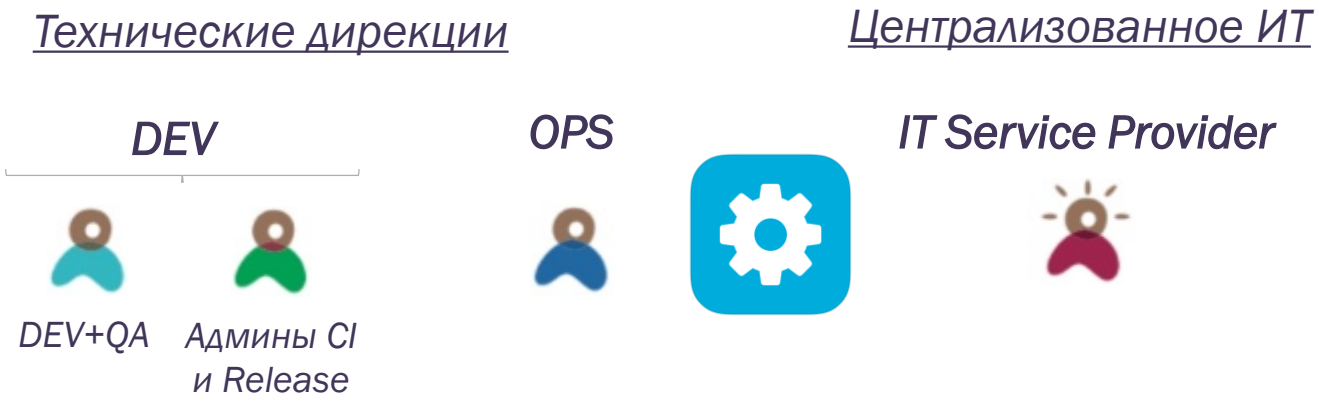
- *Chef? Puppet? Ansible?*
- *А бинарщина?*
- *Zip vs. RPM*
- *CI vs. CD | CI feat. CD ?*
- *PS\SQL + миграция схемы*
- *Stage≡Prod | Stage=Prod | Stage~Prod ?*



Вторая итерация

- Серьезное перераспределение ответственности,
- Вынос новых модулей ~ 3 дня

- Формула 1.0

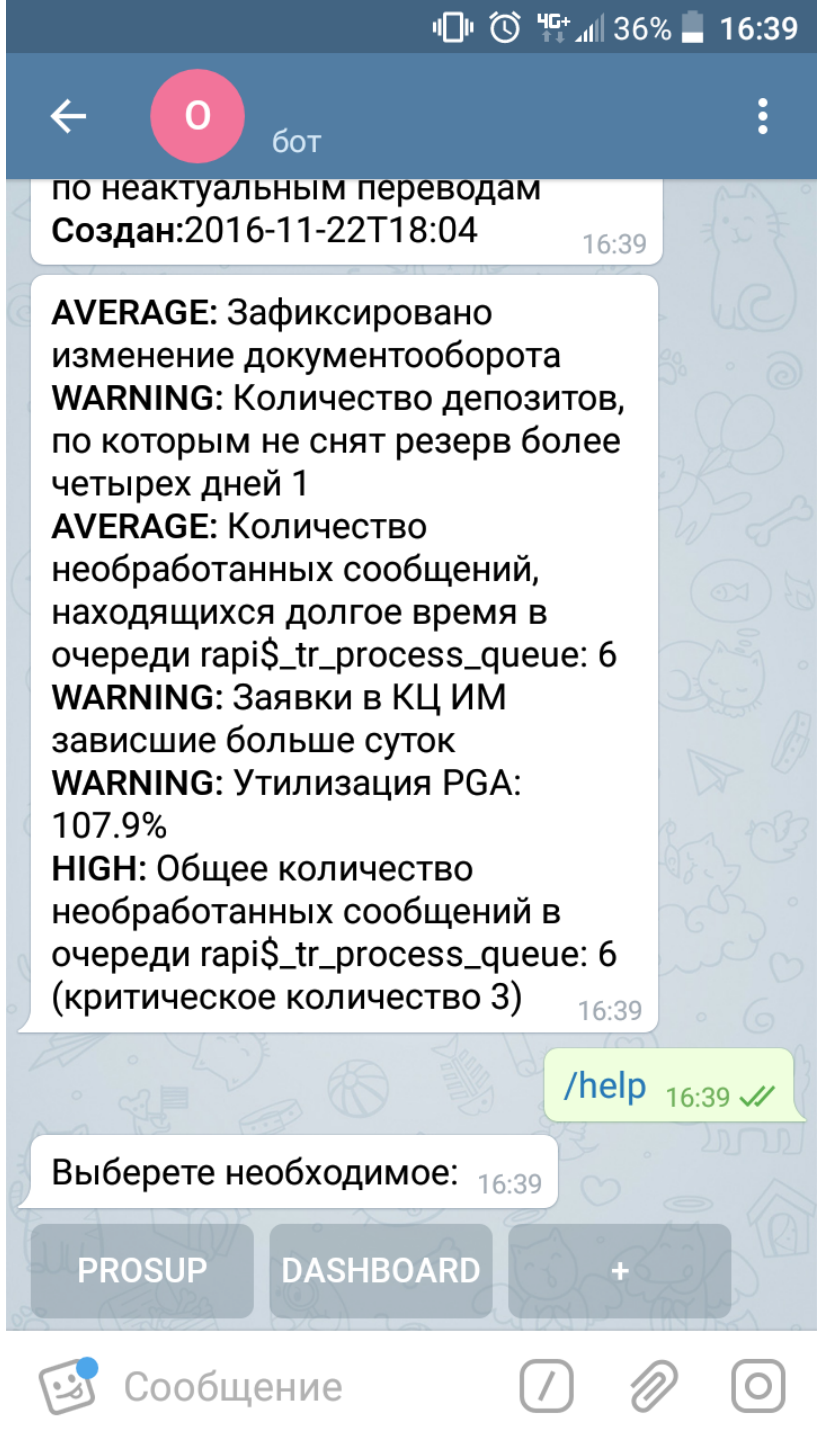


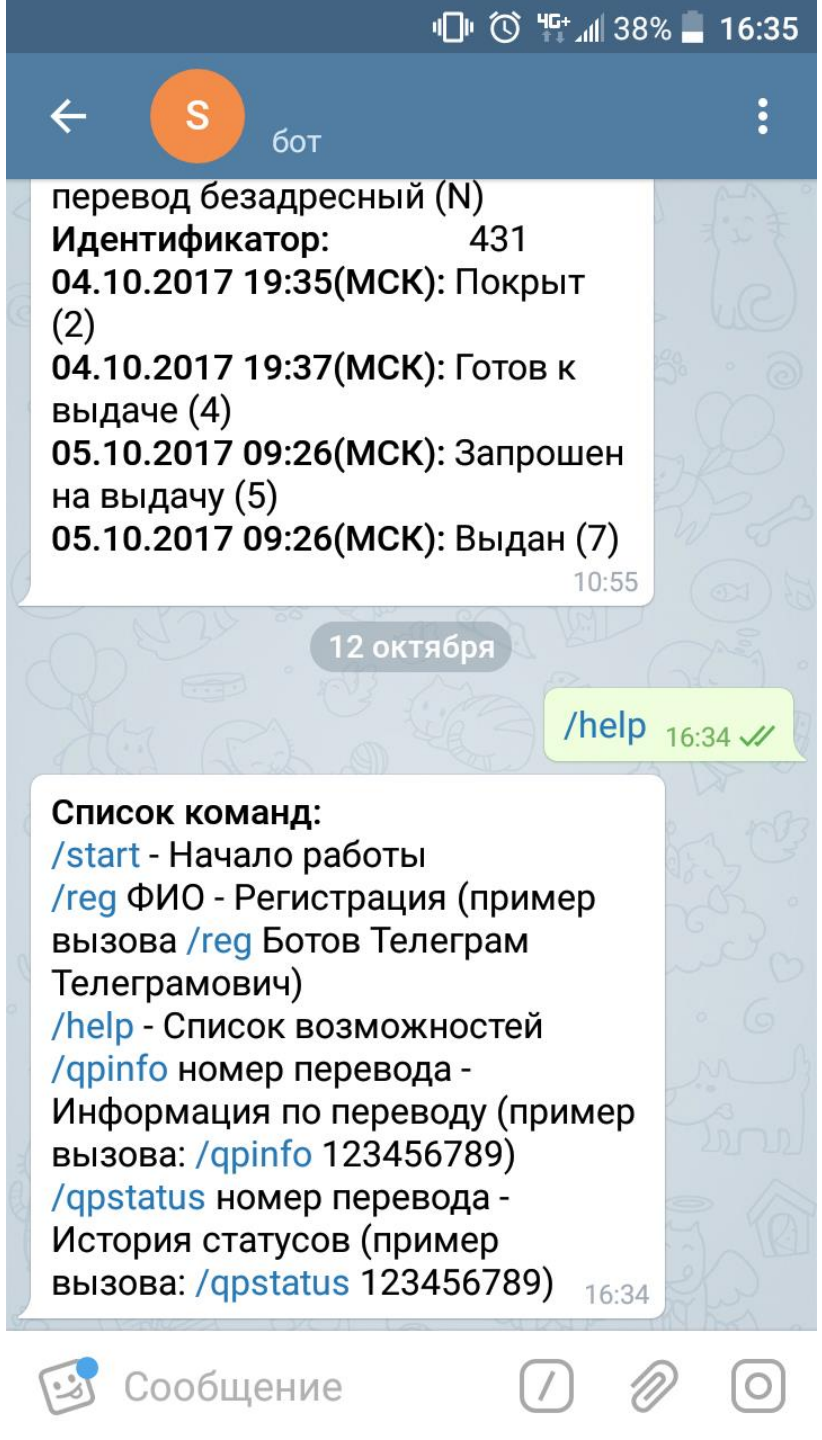
Итак, формула 1.0:

Delivery team = [(DEV + devПА) + OPS] + [IT];

Взаимодействие...
ИТ, безопасность, сопровождение, бизнес

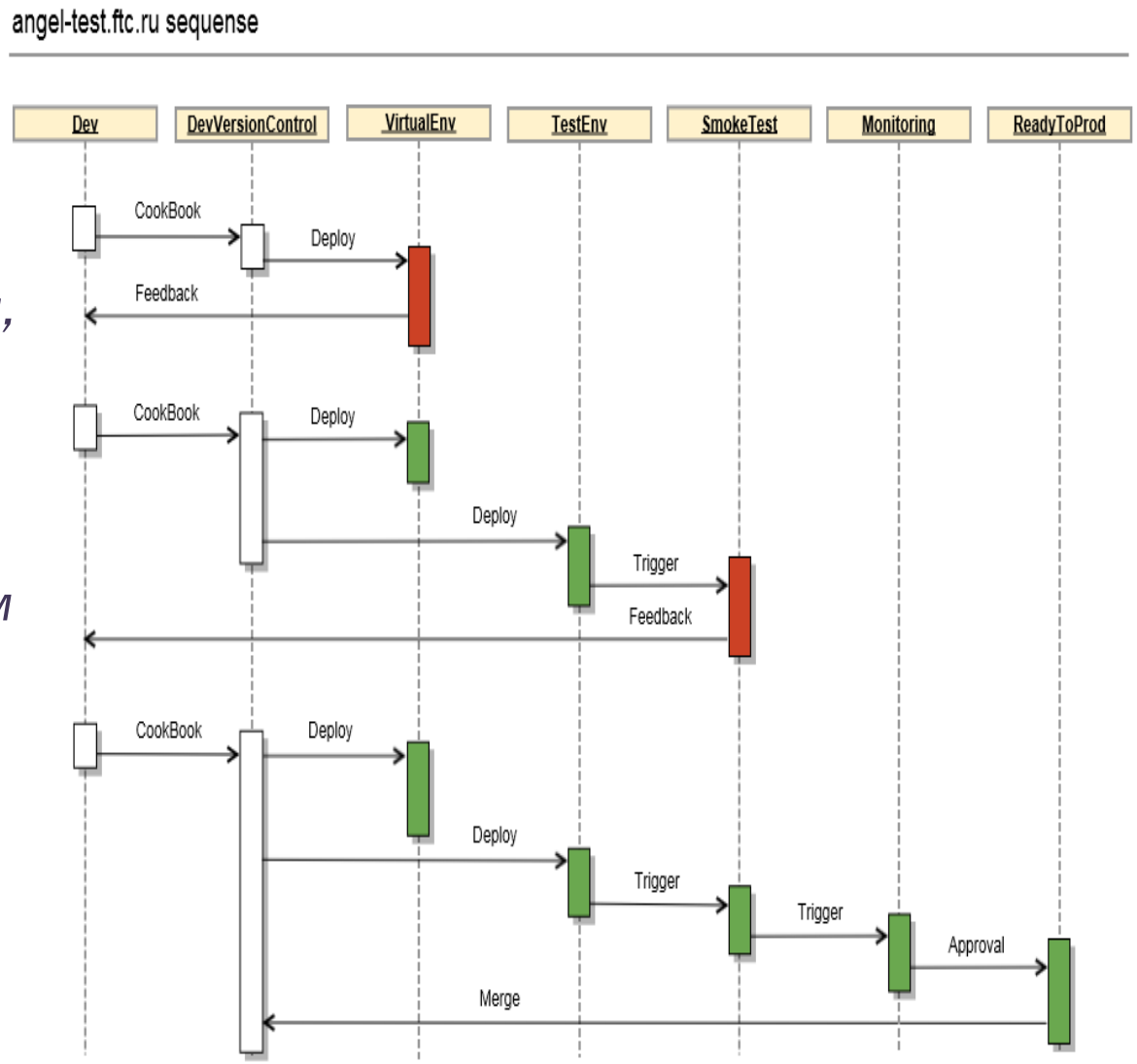
- *Строим программное api (bash и git, боты) для уменьшения кол-ва заявок,*
- *Рассказываем о том что мы делаем, что бы не быть сумасшедшими в их глазах,*
- *Перераспределение обязанностей, если по другому не получается.*


36% 16:39
O
бот
по неактуальным переводам
Создан:2016-11-22T18:04
16:39
AVERAGE: Зафиксировано изменение документооборота
WARNING: Количество депозитов, по которым не снят резерв более четырех дней 1
AVERAGE: Количество необработанных сообщений, находящихся долгое время в очереди rapi$_tr_process_queue: 6
WARNING: Заявки в КЦ ИМ зависшие больше суток
WARNING: Утилизация PGA: 107.9%
HIGH: Общее количество необработанных сообщений в очереди rapi$_tr_process_queue: 6 (критическое количество 3)
16:39
/help 16:39
Выберете необходимое: 16:39
PROSUP
DASHBOARD
+
Сообщение


38% 16:35
S
бот
перевод безадресный (N)
Идентификатор: 431
04.10.2017 19:35(МСК): Покрыт (2)
04.10.2017 19:37(МСК): Готов к выдаче (4)
05.10.2017 09:26(МСК): Запрошен на выдачу (5)
05.10.2017 09:26(МСК): Выдан (7)
10:55
12 октября
/help 16:34
Список команд:
/start - Начало работы
/reg ФИО - Регистрация (пример вызова /reg Ботов Телеграм Телеграмович)
/help - Список возможностей
/qpinfo номер перевода - Информация по переводу (пример вызова: /qpinfo 123456789)
/qpstatus номер перевода - История статусов (пример вызова: /qpstatus 123456789)
16:34
Сообщение

Работа с DEV

- *Рассказываем про то, что мы делаем и зачем,*
- *Учим основам работы с unix, разработчик должен знать как его приложение должно выглядеть на бою реально,*
- *Проходим путь от «мы пишем рецепты -> учим их править -> они пишут рецепты»*
- *Помогаем с CI и сборкой,*
- ***Помогаем строить безопасную инфраструктуру для разработки.***



Где мы сейчас

- Все ПО раскатывается из рецептов(БД, middleware, наше ПО),
- Разработчики сами пишут рецепты,
- Ops'ов привлекают как можно раньше,
- Деплой продукта на бой занимает несколько часов.

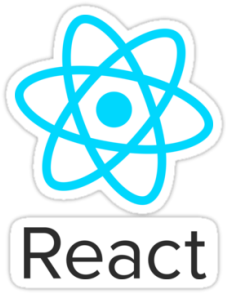
- Формула 1.1

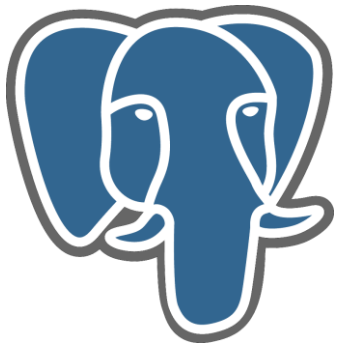
Delivery team = DEV + DEVOPS + IT;

- 5 команд разработки,
- 2 команды пишут рецепты сами,
- 176 человек в DEV,
- 500 000 строк кода в Puppet,
- 20 проектов поставки в GIT,
- **6 DEVOPSов.**

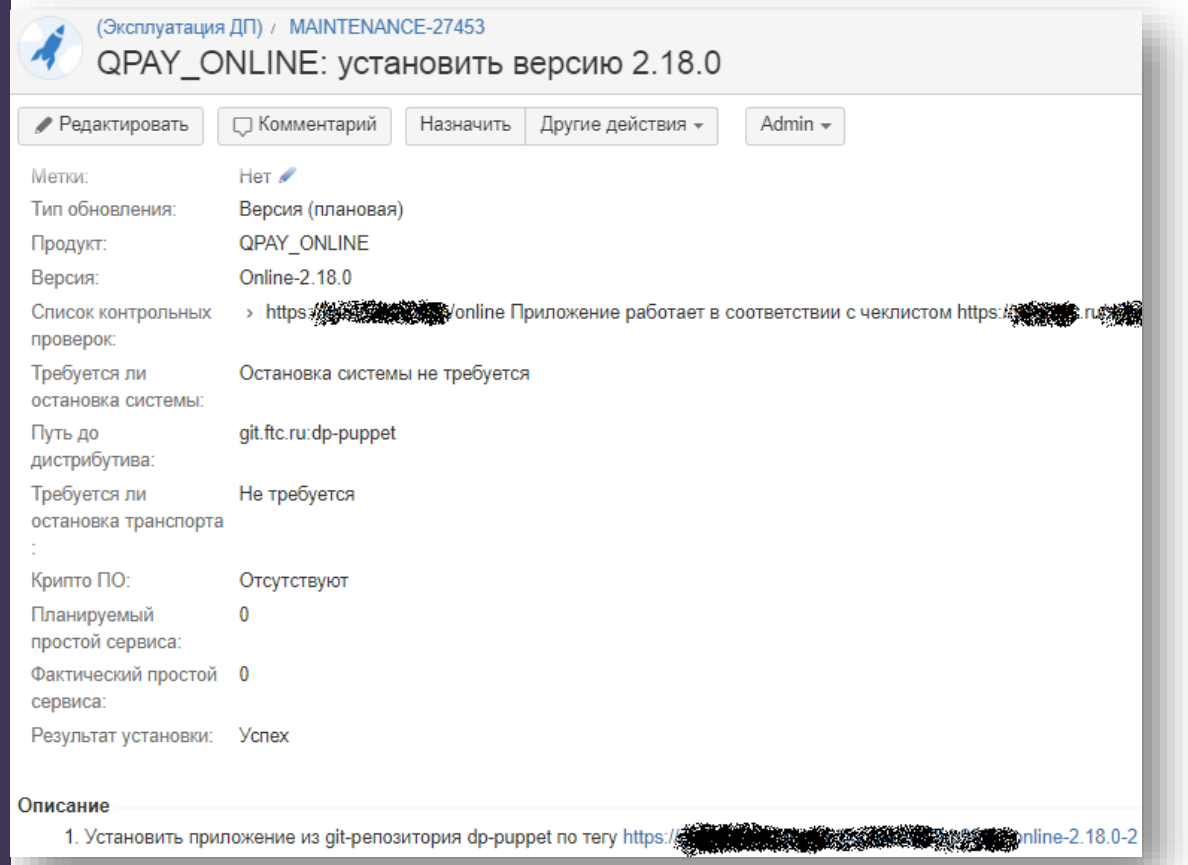
Где еще нам помог DevOps

- DRP становиться проще и актуальнее,
- CMDB с другой стороны PuppetDB,
- Горизонтальное масштабирование… в Enterprise





В цифрах



- ***Деплой новых модулей до 3 часов,***
- *Практически исчезли ошибки при выносах,*
- *Инструкция с 5 страниц А4 сократилась до пары строк…*

Почему мы достигли успеха?

- *Удалось поменять мировоззрение людей и их отношение к задачам. Изменения стали идти снизу, а не сверху,*
- *С уважением отнеслись к выстроенным процессах смежных подразделений, не было попытки все поменять «здесь и сейчас», строили мосты между нашими процессами и общекорпоративными,*
- *Есть процесс непрерывного совершенствования. К каждому новому проекту наш стек улучшается, самое лучшее передаем в ядро.*

Наши планы

- *Финансовому Enterprise нужен devSECops…*
- *Переосмысление всех процессов. Как представить гибкие процессы регулятору и аудиторам?*
- *Docker, Kubernetes…*

ВОПРОСЫ?

Сергей Бердников: s.berdnikov@cft.ru

Артем Каличкин: a.kalichkin@cft.ru