Bimeister

DevOps Governance в аудите продукта

Крылов Александр TL DevOps

Единая цифровая платформа

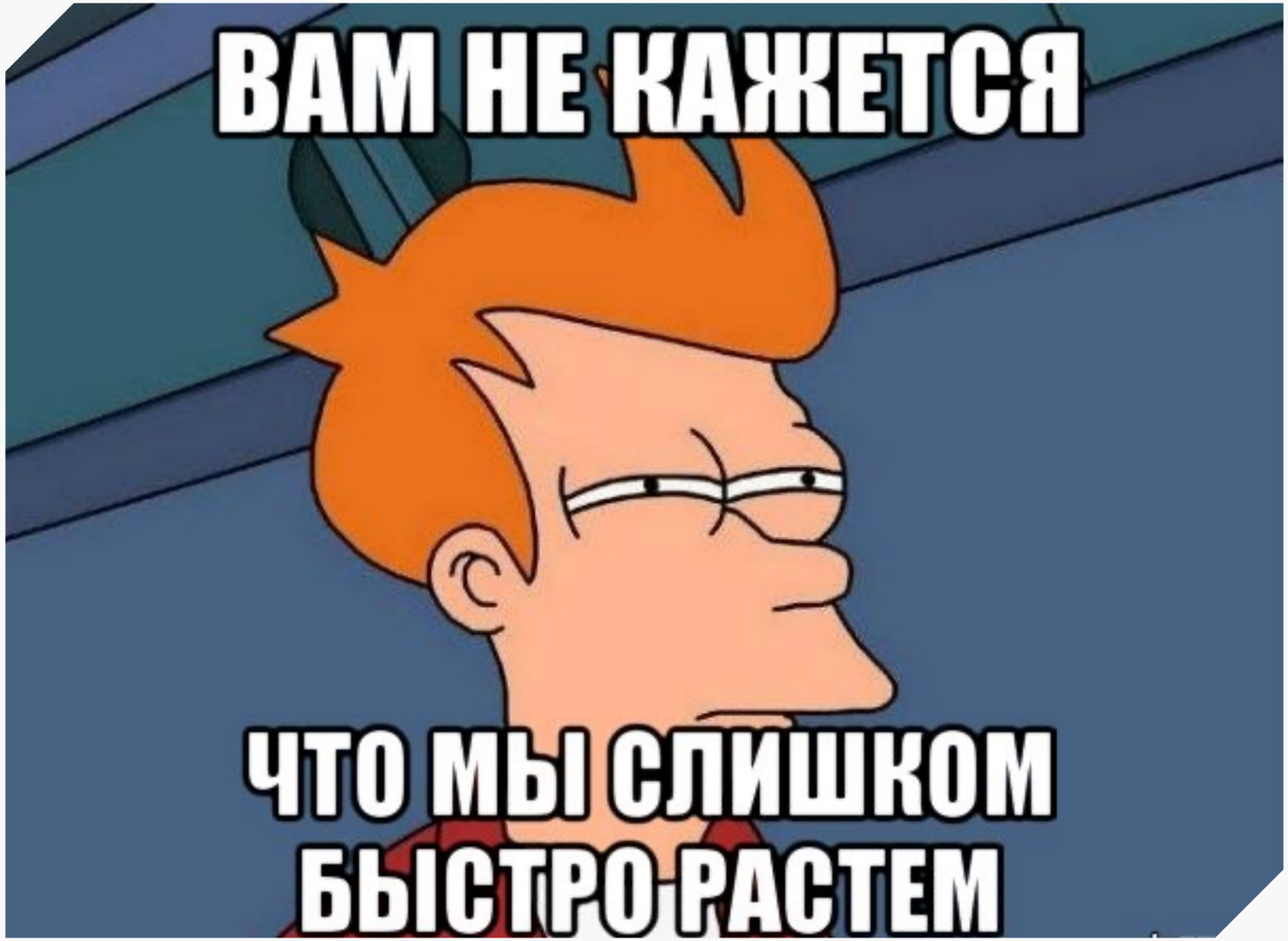
Полезность

Кому будет полезно?

1. CIO, CTO;
2. Middle management: TL DevOps/QA/Dev;
3. Project/product manager;
4. DevOps, SRE, Cloud engineer, QA, Dev, Support.

Какие проблемы решим?

1. Уменьшение T2M;
2. Улучшение разработки теми же ресурсами;
3. Управление быстрого роста компании, но не процессов;
4. Управляемая трансформация.


ВАМ НЕ КАЖЕТСЯ
ЧТО МЫ СЛИШКОМ
БЫСТРО РАСТЕМ

Спикер

- Магистр технических наук
- Спикер DevOpsConf, HighLoad++, Team Lead Conf, Слёрм
- Автор курса «DevOps-инженер» в рамках проекта «Цифровые кафедры» вузов РФ (МИРЭА, МГТУ, МГМУ)
- Автор курса по haproxy и постоянный спикер направления DevOps в Rebrain
- Методист модуля MlOps профессии data-engineer и постоянный спикер направления DevOps в Нетологии
- Соавтор и соведущий подкаста ProITStand, Brainstorm
- Автор ИТ-сообщества по мануалам @devopsforlove

Крылов Александр, TL DevOps в Bimeister

Содержание

Уровень зрелости		0 - нулевой	Первый - базовый	Второй - средний	Третий - продвинутый	Четвёртый - максимальный	#	Метрика
Разработка	Кодирование (1,5)	Код хранится локально (не используется репозиторий)	Код хранится в системе контроля версий gitlab	Приемка кода в ручном режиме (при merge/rebase PR)	Приемка кода в авто режиме (стандарты кодирования)	Документация на код собирается вместе со сборкой	1	% покрытия модульными тестами
		Отсутствуют правила кодирования	Есть стандарты кодирования Стайлгады: • Git Commit Guidelines • Rebase flow • Frontend Style Guide Линтер • front: ◦ eslint, ◦ pretier, • back ◦ валидатор resharper	Код документируется в ручном режиме Документирования Api	Код документируется в авто режиме BIM-19635 - Генерация автодокументации при написании кода/ автосоздание Release notes В РАБОТЕ	Вынос кода в прод происходит без простоя системы (наличие CD)	2	время на сборку при непрерывной интеграции
		Отсутствует практика приемки кода	Все изменения идут через отколотые (feature/hotfix*name task) ветки с указанием номера задачи в коммитах.	Код мерджится только после успешного прогона build пайплайна с последующей автоматической выкладкой на dev или test контур (+ - Автосборка есть Авторазвёртывания нет)	Код автоматически раскатывается после merge PR на тестовые контура	Новый функционал пишется с учетом практики Feature Flags (Feature Toggles)	3	% успешных сборок CI

Bimeister

DEVOPS for love

ИСТОРИЧЕСКАЯ СПРАВКА

Группа компаний Bimeister

Разработчик инженерного ПО для управления объектом на всех стадиях жизненного цикла:
Проектирования, Строительства и Эксплуатации

5 +
Лет на рынке

442
Сотрудника на 25.04.2023 г.

27 x
Рост компании за 2 года

2,7 млрд
Прогноз выручки на 2023 г.

14 млрд
Пайплайн проектов до 2026 г.

26%
Маржинальность по EBITDA 23 г.
(прогноз роста до 42% к 2028 г.)

Сделано в РФ

Полностью российское ПО
+ 2 запатентованных продукта

Патентные решения

3D-движок, система СУИД
+ в процессе получения патента на систему Управления строительством

Структура Группы компаний Bimeister

Bimeister

Разработчик ПО

Экосистема Bimeister

Методики управления промышленными активами

Bimeister Engineering

Информационное моделирование

Разработка информационных и конструкторских моделей

Оцифровка технических архивов

Лазерное сканирование и фотограмметрия

Bimeister Academy

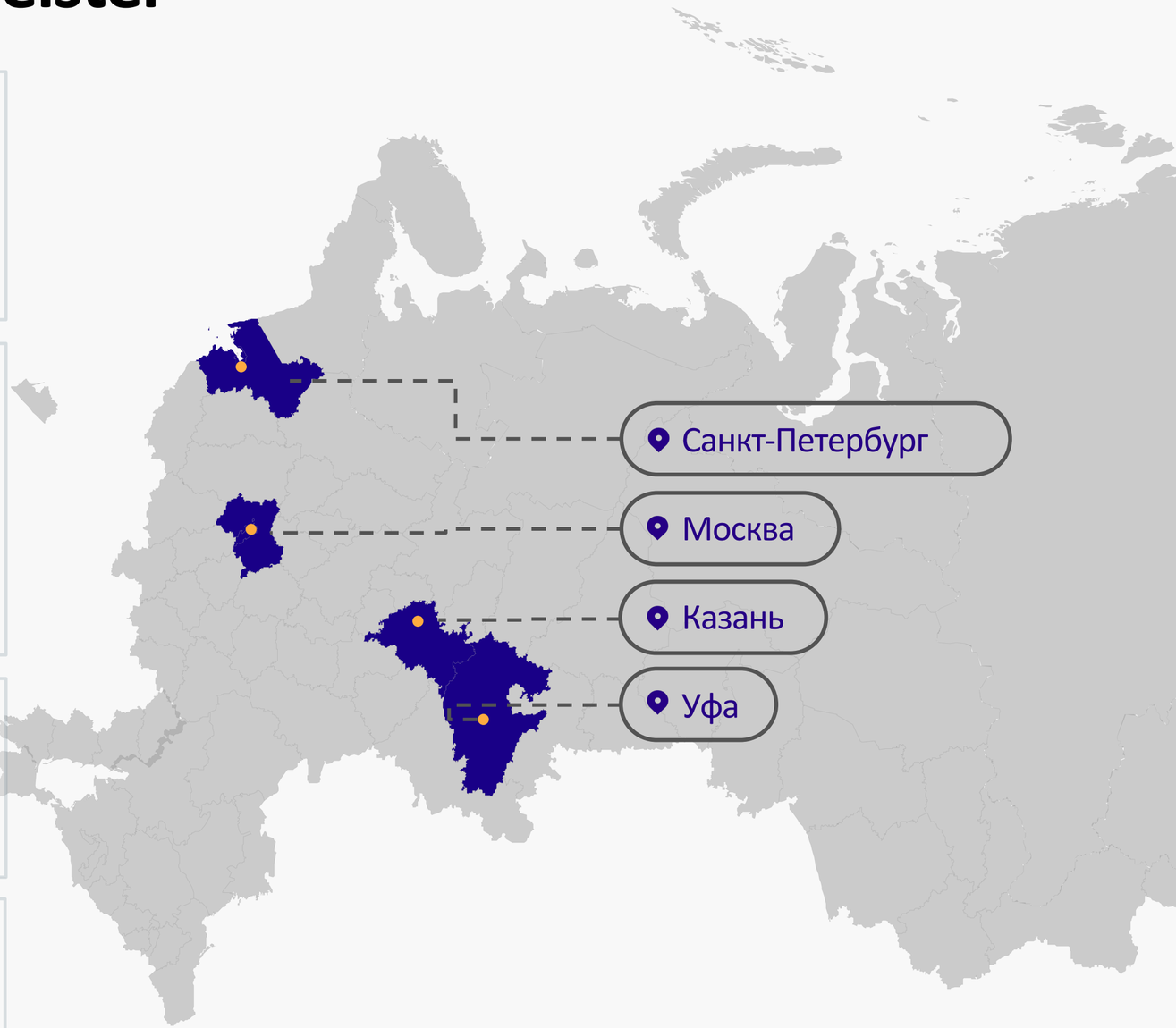
Учебный центр

Базовое и углубленное обучение BIM-технологиям

Bimeister Management

Sales & Marketing

Продажи / маркетинг / управление проектами



Экосистема решений

Bimeister – это экосистема российских решений для цифровизации процессов на всех стадиях жизненного цикла промышленного объекта.

Bimeister Standard

Проверка качества инженерных данных –

информационная система, определяющая требования и стандарты наполнения цифрового паспорта инженерными данными, предназначенная для проверки их качества, полноты и соответствия

Bimeister Data

Структурированное хранение и эффективная работа с инженерными данными –

автоматизированное формирование единой информационной модели из различных источников и форматов данных, собираемая на всех стадиях жизненного цикла объекта

СУИД

Bimeister Engineering

инжиниринговая компания, предоставляющая цифровой сервис для промышленных активов и внедрения комплексных решений

Bimeister Flow

позволяет владельцу объекта осуществить детализированный контроль сроков разработки проектной продукции и повысить ее качество

Bimeister Construct

позволяет владельцу объекта повысить прозрачность фактического исполнения работ на строительной площадке

Bimeister Control

позволяет владельцу объекта сократить время простоя оборудования, вызванное проведением ремонтных работ

Bimeister Right

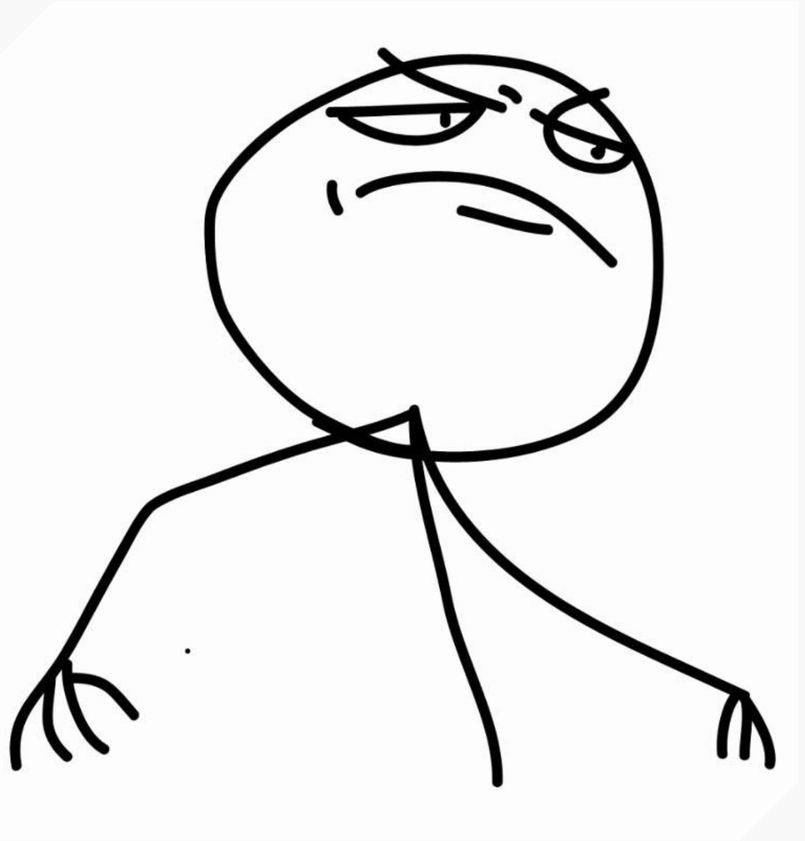
позволяет владельцу объекта сократить эксплуатационные затраты при сохранении или повышении требуемого уровня надежности

МЫ - УЖЕ НЕ СТАРТАП

- Нам 5 лет
- У нас есть CTO
- Есть постоянные заказчики
- Могут быть TOP — down решения
- Внедряется – cost center vs profit center
- Build vs buy-решения
- Согласование ресурсов на активности
- У нас есть transparency meetups

МЫ - ИТ КОМПАНИЯ

- Прививаем DevOps-культуру
- Гильдия DevOps
- Использование Agile, Scrum, Sprint, Kanban практик
- Наличие внутренней разработки
- У нас проходят демо продуктов
- Использование новейших технологий
- Почти 500 человек и все ИТ



ГЛОССАРИЙ

- Команды
- Alfa-Omega
- Sprint (3н)
- Куратор работ

- Product owner
- Тамада (фасилитатор)
- Scrum master
- CTO

МЫ

- Продукты
- Scrum of scrum
- FTE (Full-Time Employee)

- Бизнес
- DevOps as services
- Гильдия DevOps

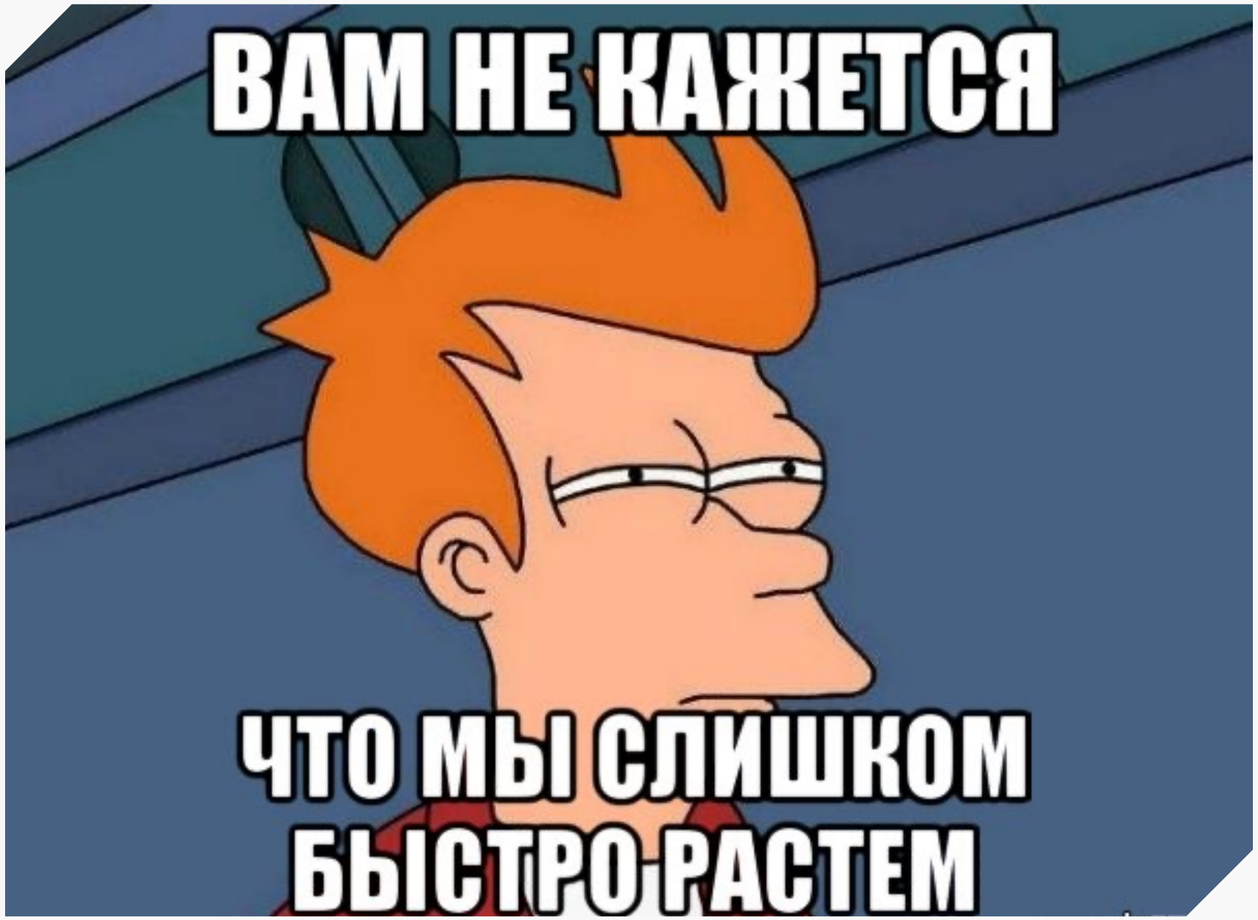
1.1 ПРОБЛЕМЫ

- Отсутствие аудита и реаудита систем
- Отсутствие плана/контроля развития технической архитектуры продукта
- Недостаточно прозрачные процессы разработки
- Нет понимания ценообразования проектов

1.2 ПУТИ РЕШЕНИЯ ПРОБЛЕМ

Что-то, что нам надо

1.3. ВРЕМЯ ДЕЙСТВОВАТЬ



Bimeister

DEVOPS for love

RND-РЕШЕНИЯ

2. ВРЕМЯ ДЕЙСТВОВАТЬ

2.1 ПОИСК РЕШЕНИЙ

- **Модель DevOps от Gartner: технологии, люди, процессы, культура**;

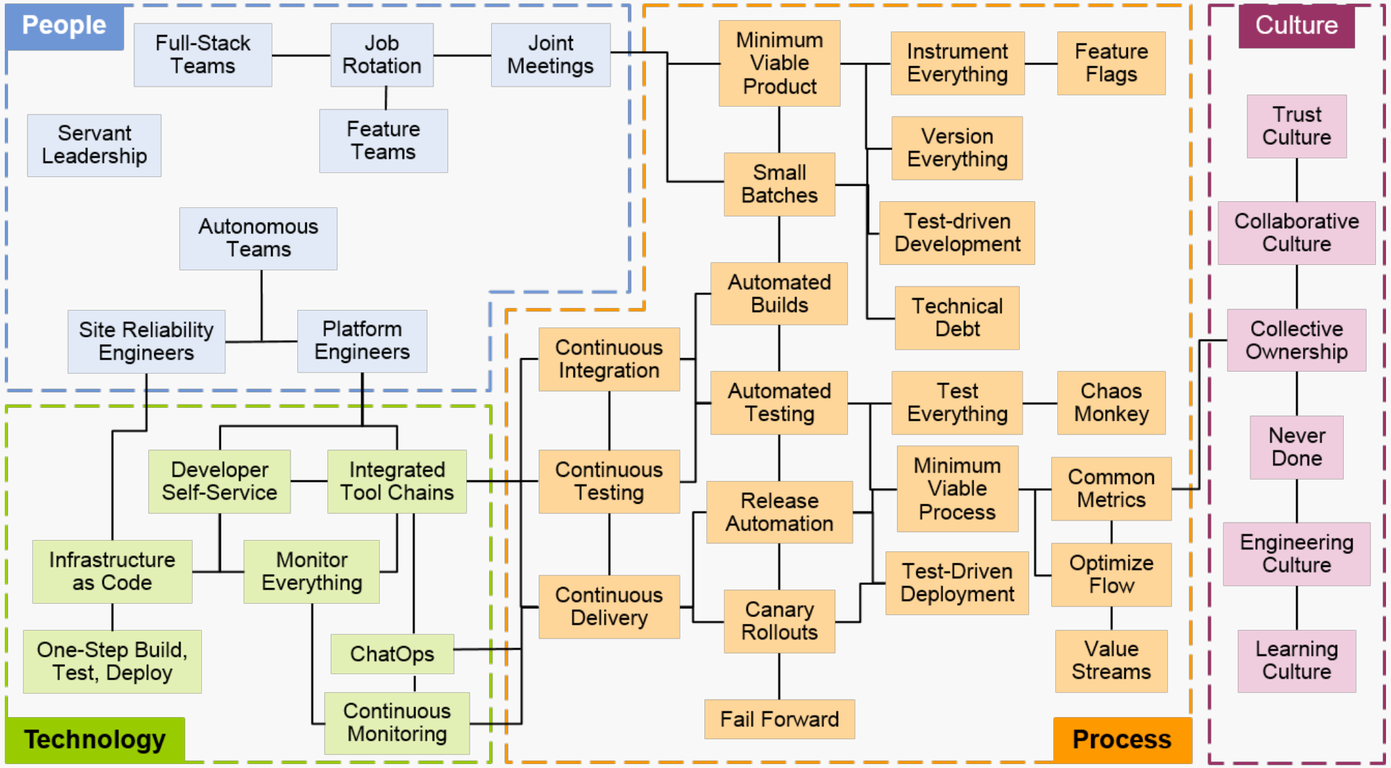


Схема blogs.gartner.com

2.1 ПОИСК РЕШЕНИЙ

- Модель DevOps от Gartner: технологии, люди, процессы, культура;
- **DORA capabilities**;

Aspect of Software Delivery Performance*	Elite	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per day and once per week	Between once per week and once per month	Between once per month and once every six months
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one day	Between one day and one week	Between one week and one month	Between one month and six months
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day[a]	Less than one day[a]	Between one week and one month
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0-15%[b,c]	0-15%[b,d]	0-15%[c,d]	46-60%

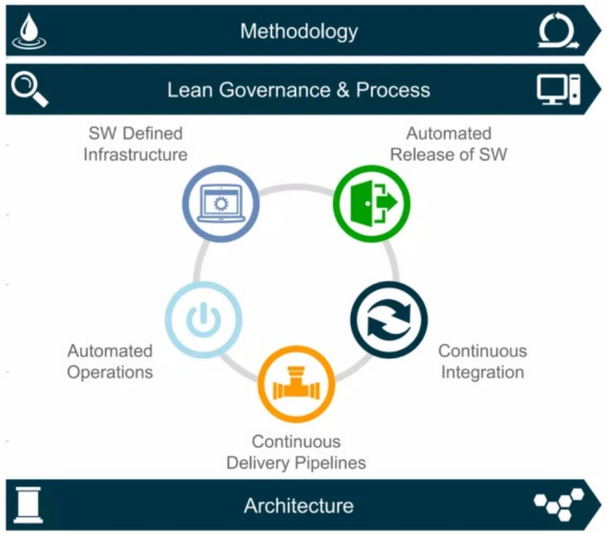
https://habr.com/ru/post/583268/

2.1 ПОИСК РЕШЕНИЙ

- Модель DevOps от Gartner: технологии, люди, процессы, культура;
- DORA capabilities;
- **DevOps maturity model Accenture**;

DevOps Maturity Model – Framework

DELIVERY ORGANISATION AND METHODOLOGY	• Release Frequency • Delivery Organisation • Delivery Process & Methodology • Metrics & Governance
LEAN DELIVERY GOVERNANCE & PROCESSES	• Governance Approach • Continuous Improvement • Product Management • Delivery & Change Control
AUTOMATED RELEASE OF SOFTWARE	• Source Code Version Control • Deployment Process • Deployment Artefact Management
CONTINUOUS INTEGRATION	• Traceability • Continuous Integration • Code Quality & Unit Test
CONTINUOUS DELIVERY (INCLUDING AUTOMATED QA)	• Test Strategy • Test Automation • Test Data Management • Deployment Pipelines
AUTOMATED OPERATIONS	• Monitoring • Organisational Considerations • Resilience
SOFTWARE DEFINED INFRASTRUCTURE & CLOUD	• Environments • Infrastructure Suitability • Responsiveness
PLATFORM/APPLICATION ARCHITECTURE	• Architecture • Impact of Component Failure • Business Enablement



DevOps maturity model

2.1 ПОИСК РЕШЕНИЙ

- Модель DevOps от Gartner: технологии, люди, процессы, культура;
- DORA capabilities;
- DevOps maturity model Accenture;
- **DASA DevOps**;

DASA DevOps v3

2.1 ПОИСК РЕШЕНИЙ

- Модель DevOps от Gartner: технологии, люди, процессы, культура;
- DORA capabilities;
- DevOps maturity model Accenture;
- DASA DevOps;
- **Оценка аудита со стороны**;

2.2 ПОИСК РЕШЕНИЙ И ИТОГИ

- Модель DevOps от Gartner: технологии, люди, процессы, культура;
- DORA capabilities;
- DevOps maturity model Accenture;
- DASA DevOps;
- Оценка аудита со стороны;
- **Опыт коллег из прошлых компаний:**

  - делали ли они аудит систем?

  - делали ли они это сами?

  - как делали?

  - какие метрики использовали?

- Консалтинг — дорого;
- Готового решения на рынке под нас — нет;
- **Выгоднее идти своим путём**.

2.3 РЕШЕНИЕ

Идём своим путём!

Переиспользуем «матрицу зрелости систем» на базе существующего опыта для внедрения матрицы зрелости продукта

Старт – август 2023 года

Bimeister

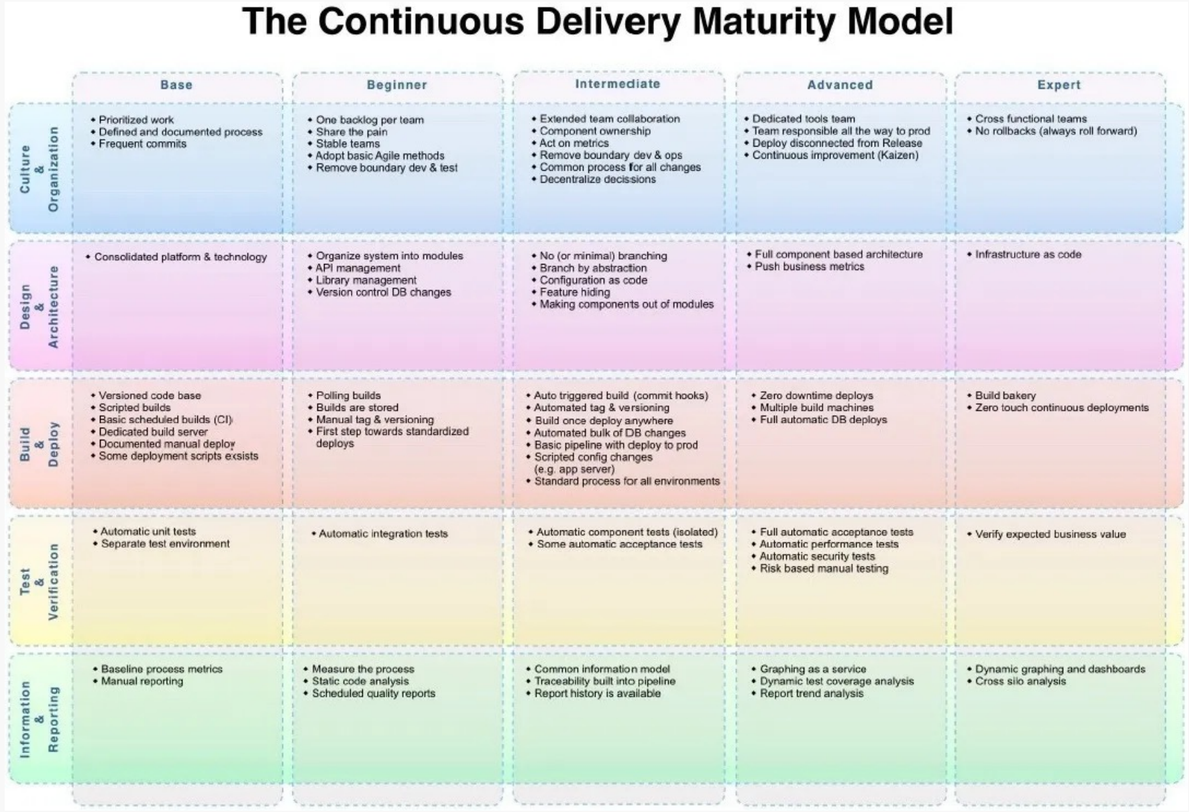
DEVOPS for love

ОПРЕДЕЛЕНИЕ МАТРИЦЫ ЗРЕЛОСТИ ПРОДУКТА

3. ОПРЕДЕЛЕНИЕ МАТРИЦЫ ЗРЕЛОСТИ ПРОДУКТА

Матрица зрелости продукта – прозрачный процесс аудита и реаудита продуктов компании с использованием:

- практик гибкой разработки;
- культуры DevOps;
- Концептов DORA, DASA, DevOps maturity;
- современных подходов к автоматизации;
- современных инструментов CI/CD.



Bimeister

DEVOPS for love

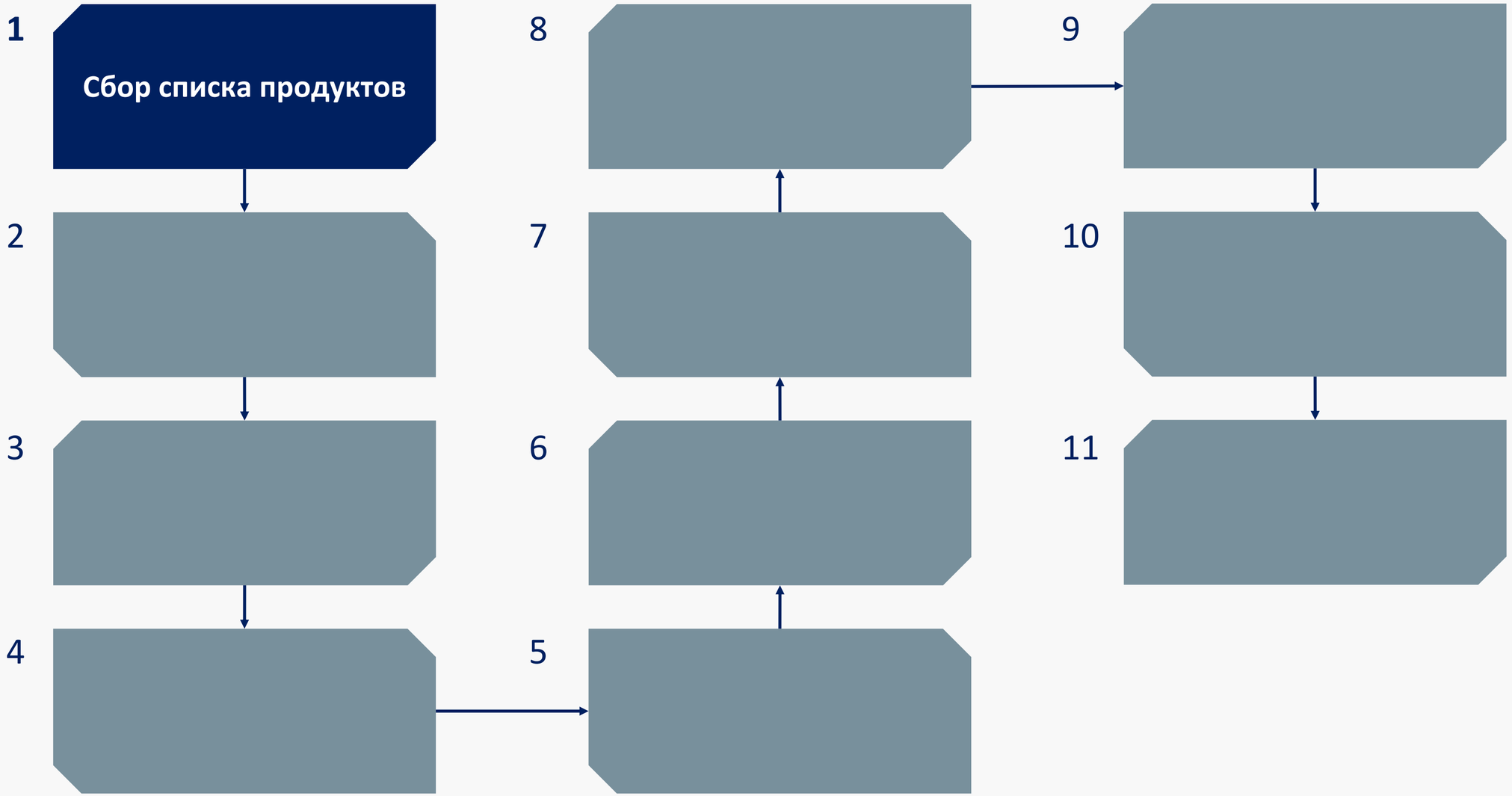
ЭТАПЫ ВНЕДРЕНИЯ МАТРИЦЫ

4.1 ЭТАПЫ ВНЕДРЕНИЯ

1. Сбор списка продуктов
2. Определение ресурсов команды
3. Мутация блоков и методологии
4. Мутация глоссария
5. Согласование количества уровней зрелости
6. Пересмотр первичных метрик
7. Аудит по согласованным метрикам
8. Планирование и согласование работ
9. Реализация
10. Контроль работ
11. Реаудит раз в квант времени

В нашем случае этапов 11

4.1 ЭТАПЫ ВНЕДРЕНИЯ. ЭТАП 1

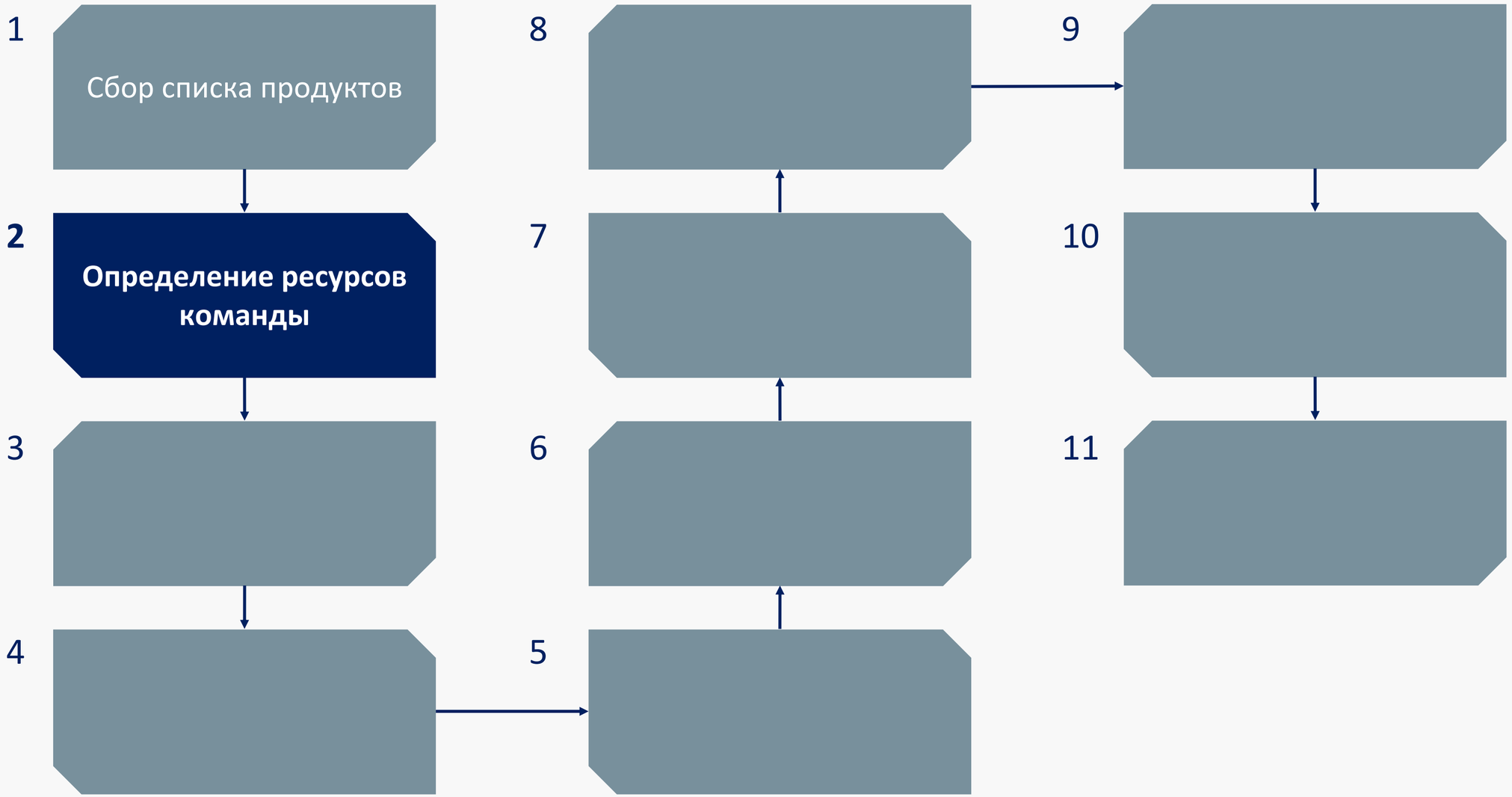


4.1 ЭТАП 1. СБОР СПИСКА ПРОДУКТОВ

- 4 продукта;
- Аудируем их как одну платформу, без разделений;
- Перечень команд по продуктам от Alfa до Omega.

Продукты	Команд внутри
4 (как один)	11

4.2 ЭТАПЫ ВНЕДРЕНИЯ. ЭТАП 2

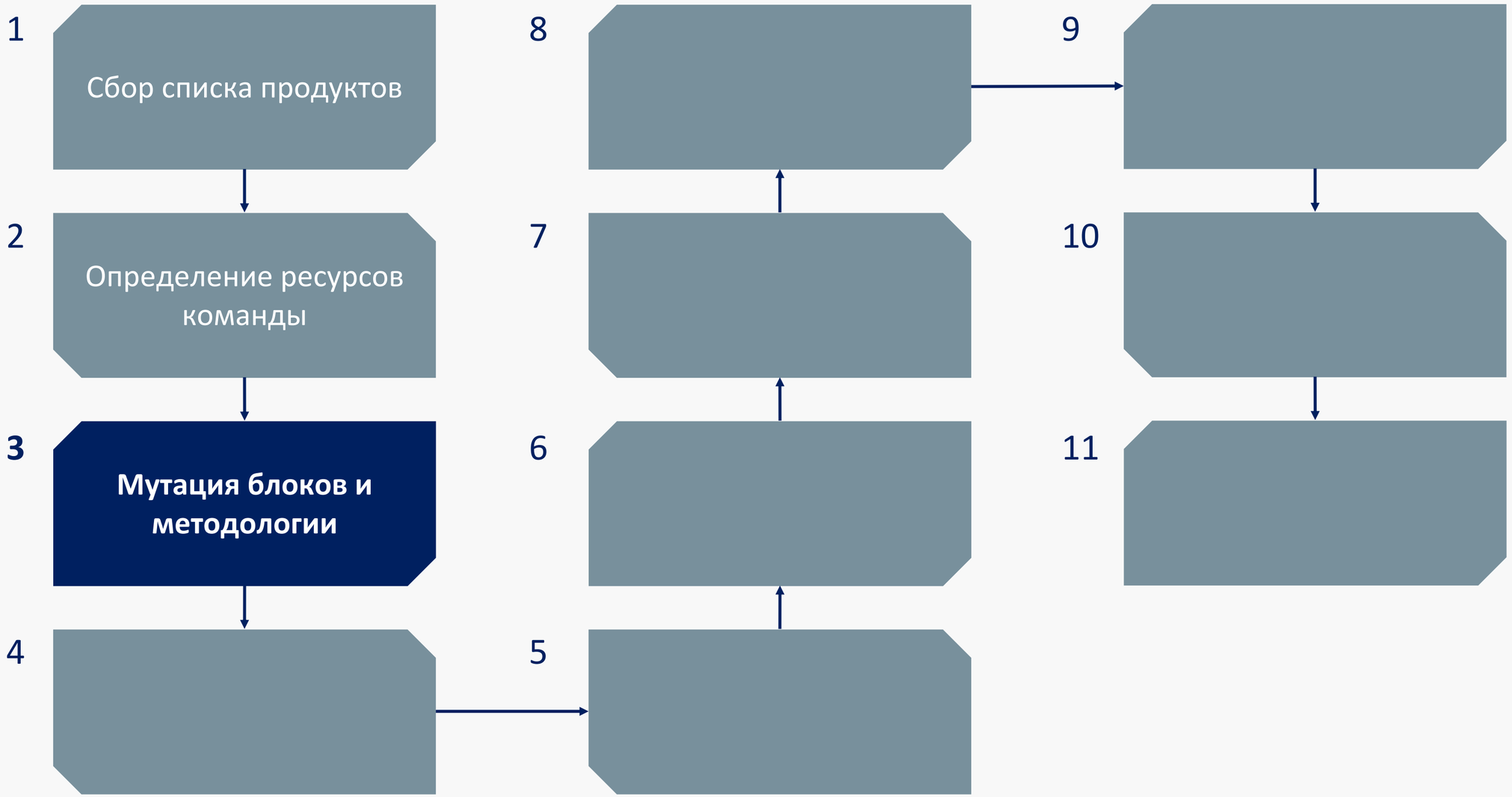


4.2 ЭТАП 2. ОПРЕДЕЛЕНИЕ РЕСУРСОВ КОМАНДЫ

Аллокация FTE:

- DevOps;
- Команды внутри;
- Где есть ресурсы у команд?

4.3 ЭТАПЫ ВНЕДРЕНИЯ. ЭТАП 3



4.3 ЭТАП 3. Мутация блоков и методологии

Практики:

- Разработка:
  - кодирование;
  - модульное/интеграционное тестирование;
  - статический анализ;
  - сборка;
  - CI/CD.

4.3 ЭТАП 3. Мутация блоков и методологии

Практики:

- Разработка;
- **Тестирование:**
  - **методология;**
  - **инструменты;**
  - **команда.**

4.3 ЭТАП 3. Мутация блоков и методологии

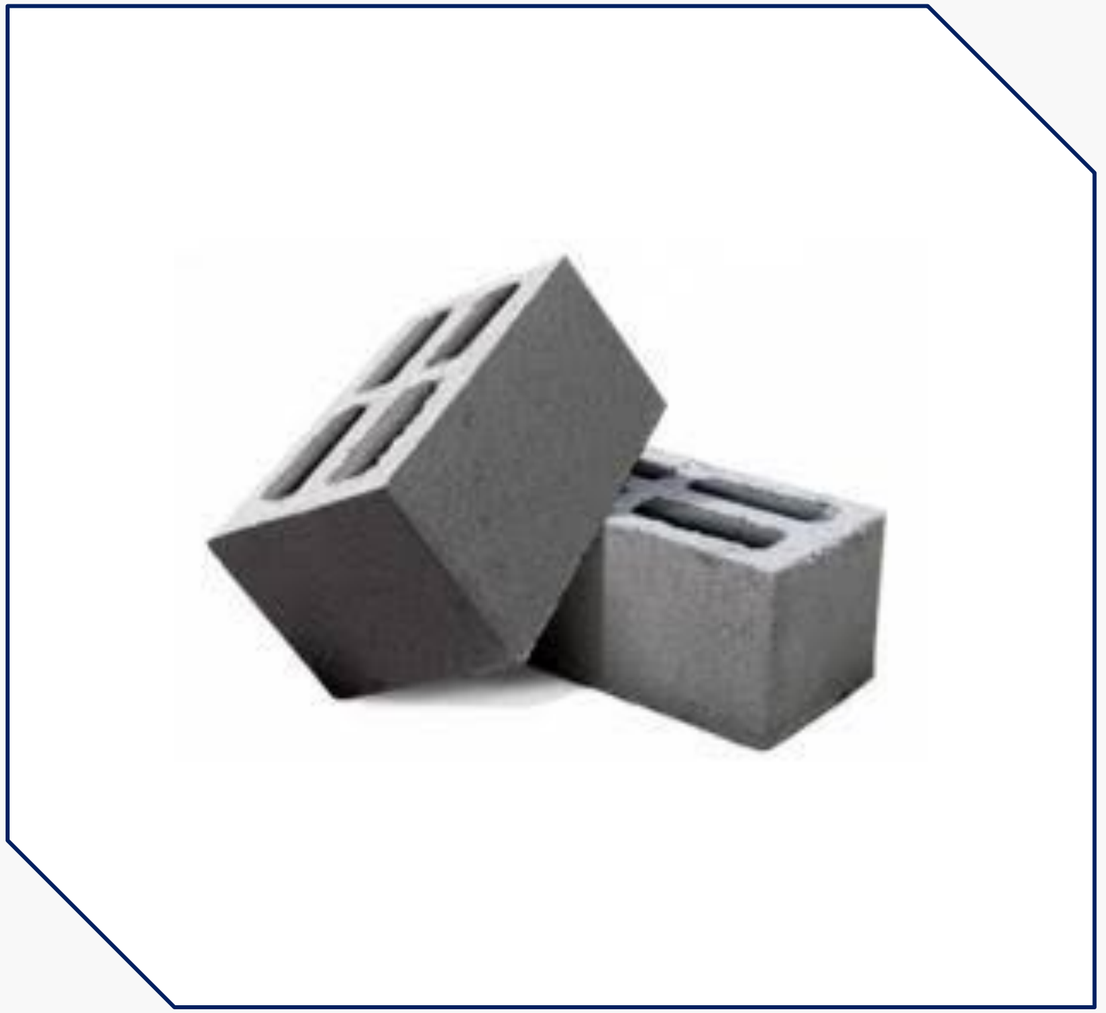
Практики:

- Разработка;
- Тестирование;
- **Сопровождение:**

  - инфраструктура;

  - логирование;

  - мониторинг.



4.3 ЭТАП 3. Мутация блоков и методологии

Практики:

- Разработка;
- Тестирование;
- Сопровождение;
- **Безопасности:**

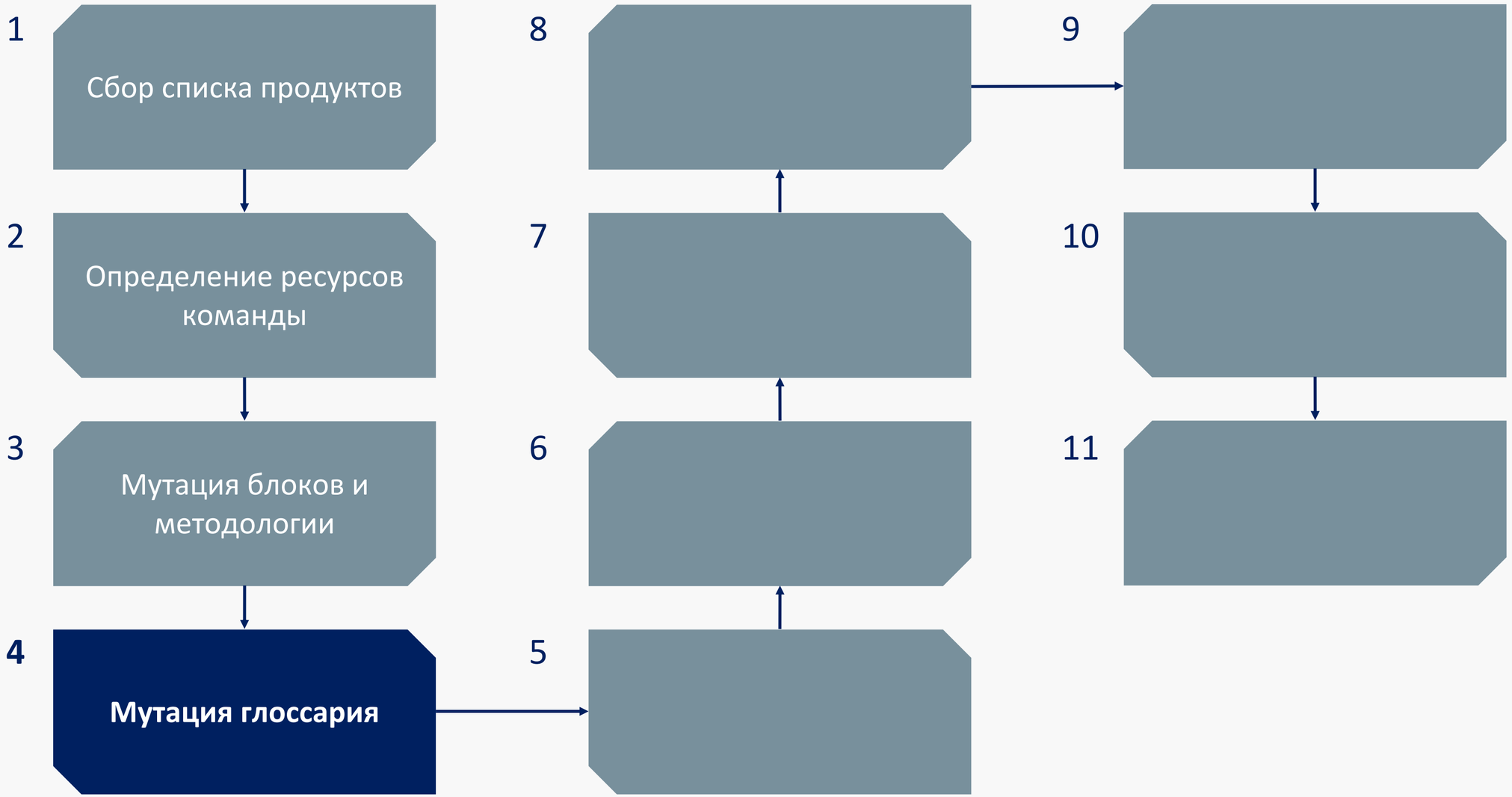
  - анализ на уязвимости;

  - безопасная разработка (SAST, SCA);

  - DAST;

  - безопасность инфраструктуры.

4.4 ЭТАПЫ ВНЕДРЕНИЯ. ЭТАП 4



4.4 ЭТАП 4. СОСТАВЛЕНИЕ ГЛОССАРИЯ ПО РАЗДЕЛАМ

Глоссарий:

- Терминология для команд (общий язык);
- Описание процессов (обязано быть);
- Описание инструментов (должно быть);
- Описание методологии (что должно быть и чего не должно).

https://www.rfc-editor.org/rfc/rfc2119

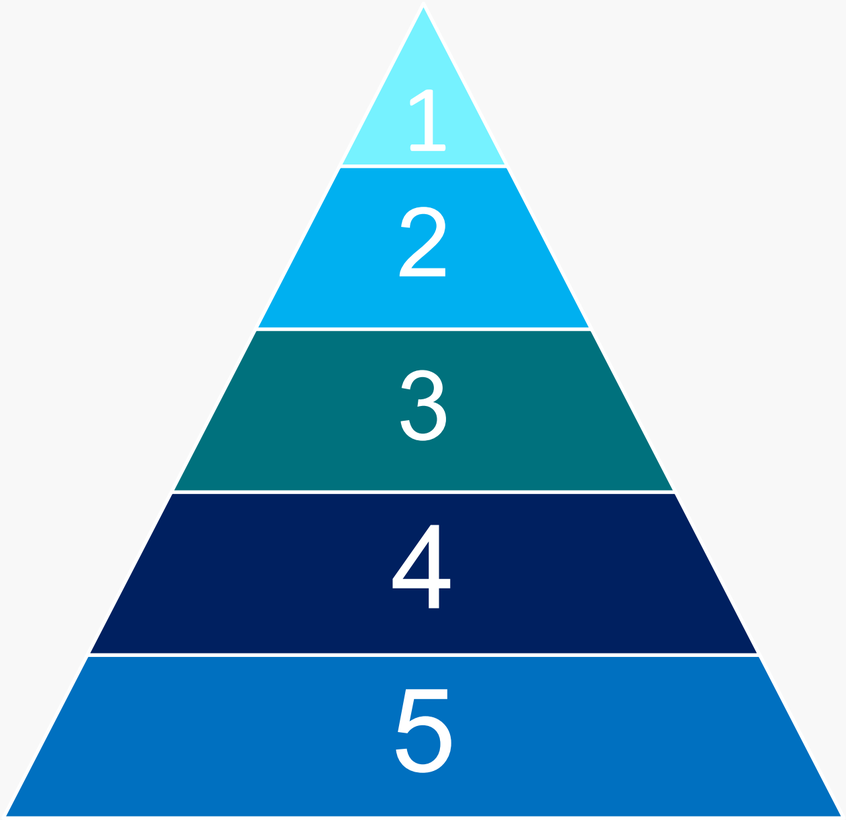
4.4 ЭТАП 4. СОСТАВЛЕНИЕ ГЛОССАРИЯ ПО РАЗДЕЛАМ

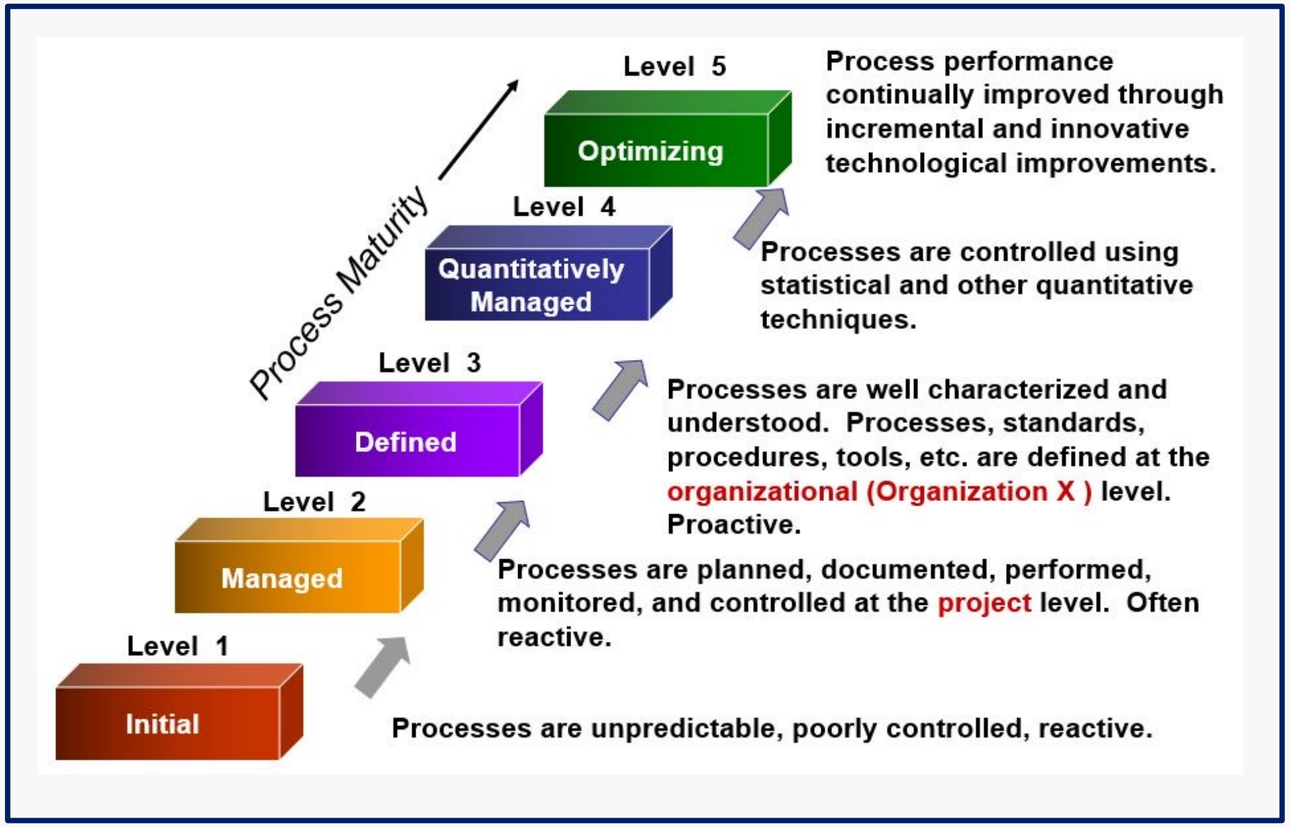
Сопровождение	Инфраструктура - базовый слой, сервисный слой, слой приложений	Используются статические среды, создаваемые и обслуживаемые по запросу	Автоматическое развертывание для бинарников, конфигураций и настроек	Автоматическое единообразное развертывание для всех сред (одни и те же артефакты двигаются по средам, меняется только конфигурационные параметры, инфраструктура). Сохраняется принцип идемпотентности.	Применяется практика "Инфраструктура как код" (IaC) (конфигурация инфраструктуры хранится в системе контроля версий, применяется автоматически с помощью инструмента авторазвертывания) Операции обновление версии платформы, обновление версий системных пакетов, драйверов, обновление SSL-сертификатов, обновления настроек безопасности должны проходить по конвейеру как и изменения в коде продукта, сначала проверяться на не интеграционных стендах, после чего тестироваться с интеграциями и только потом применяться к продуктивной среде.	Применяется практика "Инфраструктура, как сервис" используются среды, создаваемые динамически, расширяемые по потребностям, настраиваемые с помощью практики инфраструктура как код.
		Развертывание всех компонентов производится в ручном режиме	Ручные смок тесты после развёртывания	Автопроверка установок через смок-тесты на всех средах - смок тесты встроены в работу пайплайн. Автоматически отслеживаются параметры работы автотестов: (Метрика №6)	Для инфраструктурного кода используется CI конвейер, проверяющий корректность вносимых в инфраструктурный код изменений. Инфраструктурный код покрыт текстами и встроен в pipeline.	Полностью автоматическое развертывание, включая БД. Автоматически отслеживаются параметры работы развертывания: (Метрика №8)

4.5 ЭТАПЫ ВНЕДРЕНИЯ. ЭТАП 5

4.5 ЭТАП 5. СОГЛАСОВАНИЕ КОЛИЧЕСТВА УРОВНЕЙ

- Уровни зрелости на базе CMMI





Уровни зрелости управления CMMI

4.5 ЭТАП 5. СОГЛАСОВАНИЕ КОЛИЧЕСТВА УРОВНЕЙ

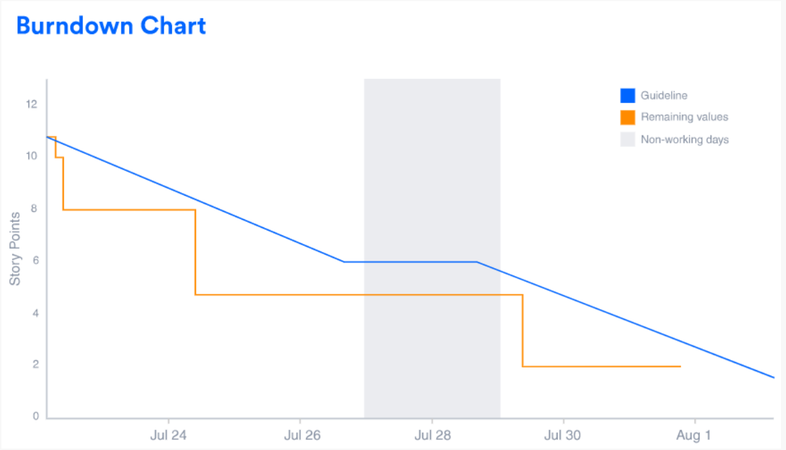
- Нулевой
- Базовый
- Средний
- Продвинутый
- Максимальный

0 - нулевой	Первый - базовый	Второй - средний	Третий - продвинутый	Четвёртый - максимальный
Код хранится локально (не используется репозиторий)	Код хранится в системе контроля версий gitlab	Приемка кода в ручном режиме (при merge/rebase PR)	Приемка кода в авто режиме (стандарты кодирования)	Документация на код собирается вместе со сборкой

4.6 ЭТАПЫ ВНЕДРЕНИЯ. ЭТАП 6

4.6 ЭТАП 6. ПЕРЕСМОТР ПЕРВИЧНЫХ МЕТРИК

	High	Medium	Low
Частота развертываний Как часто происходит развертывание новой версии приложения на продуктовое окружение (плановых изменений исключая хотфиксы и реакцию на инциденты)?	По требованию (несколько раз в день)	От раза в неделю до раза в месяц	От раза в месяц до раза в 6 месяцев
Срок поставки Сколько в среднем времени проходит между коммитом изменения (написанием функциональности в виде кода) и развертыванием изменения на продуктовом окружении?	Меньше дня	От дня до недели	От недели до месяца
Время восстановления Сколько в среднем времени занимает восстановление приложения на продуктовом окружении после инцидента, деградации сервиса или обнаружения ошибки, влияющей на пользователей приложения?	Меньше часа	Меньше дня	От дня до недели
Неуспешные изменения Какой процент развертываний на продуктовом окружении приводит к деградации приложения или инцидентам и требует устранения последствий (откат изменений, разработку хотфикса или патча)?	0-15%*	0-15%*	16-30%



[Метрики дора](#)
[agile/project-management/metrics](#)
[Состояние devops в 2023](#)

4.6 ЭТАП 6. ПЕРЕСМОТР ПЕРВИЧНЫХ МЕТРИК

- Зеленый – реализовано
- Жёлтый – в процессе
- Красный – не реализовано

0 - нулевой	Первый - базовый	Второй - средний	Третий - продвинутый	Четвёртый - максимальный
Код хранится локально (не используется репозиторий)	Код хранится в системе контроля версий gitlab	Приемка кода в ручном режиме (при merge/rebase PR)	Приемка кода в авто режиме (стандарты кодирования)	Документация на код собирается вместе со сборкой
Отсутствуют правила кодирования	Есть стандарты кодирования Стайлгады: • Git Commit Guidelines • Rebase flow • Frontend Style Guide Линтер • front: ◦ eslint, ◦ pretier, • back ◦ валидатор resharper	Код документируется в ручном режиме Документирования Api	Код документируется в авто режиме BIM-19635 - Генерация автодокументации при написании кода/ автосоздание Release notes В РАБОТЕ	Вынос кода в прод происходит без простоя системы (наличие CD)
Отсутствует практика приемки кода	Все изменения идут через отколотые (feature/hotfix*name task) ветки с указанием номера задачи в коммитах.	Код мерджится только после успешного прогона build пайплайна с последующей автоматической выкладкой на dev или test контур (+ - Автосборка есть Авторазвёртывания нет)	Код автоматически раскатывается после merge PR на тестовые контура	Новый функционал пишется с учетом практики Feature Flags (Feature Toggles)

4.6 ЭТАП 6. ПЕРЕСМОТР ПЕРВИЧНЫХ МЕТРИК

Уровень зрелости		0 - нулевой	Первый - базовый	Второй - средний	Третий - продвинутый	Четвёртый - максимальный
Разработка	Кодирование (1,5)	Код хранится локально (не используется репозиторий)	Код хранится в системе контроля версий gitlab	Приемка кода в ручном режиме (при merge/rebase PR)	Приемка кода в авто режиме (стандарты кодирования)	Документация на код собирается вместе со сборкой
		Отсутствуют правила кодирования	Есть стандарты кодирования Стайлгады: • Git Commit Guidelines • Rebase flow • Frontend Style Guide Линтер • front: ◦ eslint, ◦ pretier, • back ◦ валидатор resharper	Код документируется в ручном режиме Документирования Api	Код документируется в авто режиме BIM-19635 - Генерация автодокументации при написании кода/ автосоздание Release notes В РАБОТЕ	Вынос кода в прод происходит без простоя системы (наличие CD)

#	Метрика
1	% покрытия модульными тестами
2	время на сборку при непрерывной интеграции

4.6 ЭТАП 6. ПЕРЕСМОТР ПЕРВИЧНЫХ МЕТРИК

Название практики	Название метрики
Модульное тестирование	% покрытия модульными тестами
CI/CD	время на сборку при непрерывной интеграции
	% успешных сборок непрерывной интеграции
	время на поставку
	% успешных поставок
Методология тестирования	время на прохождение смок-тестов
	время на прохождение регресс-тестов
	% автоматизации тестовых сценариев
	% ложных срабатываний у автотестов
	% пропущенных дефектов в интеграционные среды
	% успешно пройденных тестов
Инфраструктура	время на подготовку нового окружения

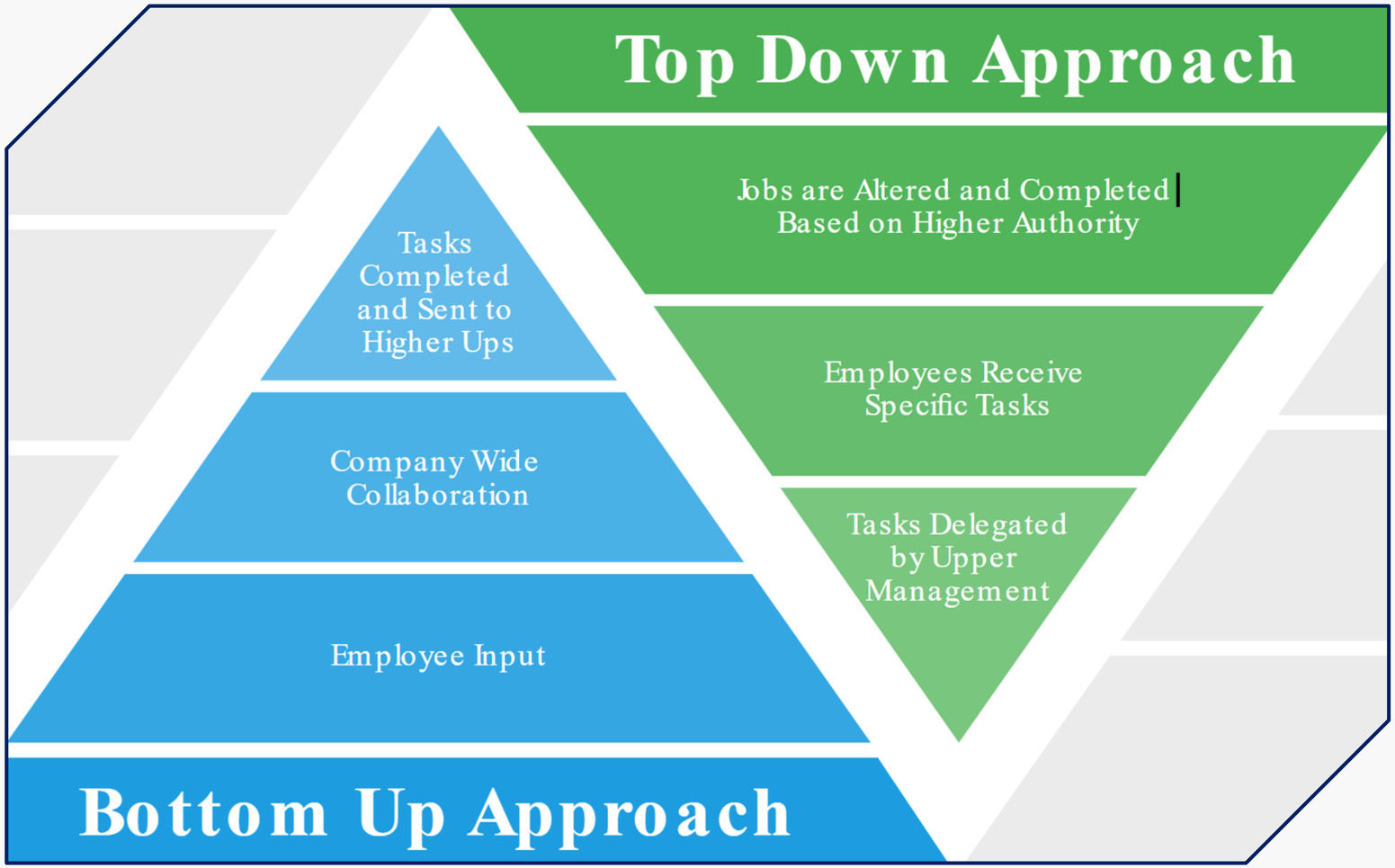
4.7 ЭТАПЫ ВНЕДРЕНИЯ. ЭТАП 7

4.7 ЭТАП 7. АУДИТ ПО СОГЛАСОВАННЫМ МЕТРИКАМ

4.7 ЭТАП 7. АУДИТ ПО СОГЛАСОВАННЫМ МЕТРИКАМ

Аудит продуктов:

- Зачем нам это надо?
- Что это такое?
- У нас нет ресурсов!
- Согласуйте ресурсы!
- И т.д.



4.7 ЭТАП 7. АУДИТ ПО СОГЛАСОВАННЫМ МЕТРИКАМ

Этап	Статус	Продукт	Framework	Потребность в реализации	Задача	Ответственный
1	Аудит 08.2023	Bimeister	.Net	Что-то	DevOps-XXX	TL DevOps TL QA TL DEV
2	2024					
	Целевой					

4.7 ЭТАП 7. АУДИТ ПО СОГЛАСОВАННЫМ МЕТРИКАМ

Статус	Разработка					
	Кодирование	Модульное тестирование	Статический анализ	Сборка	Анализ кода на уязвимости по ИБ	CI/CD
08.2023	1,5	1	0	1,5	1	1,5
2024	2	1,5	1	2	2	2
Целевой	3	3	3	3	3	3

Шкала уровней от 0 до 4

4.7 ЭТАП 7. АУДИТ ПО СОГЛАСОВАННЫМ МЕТРИКАМ

Роли	Служба DevOps	Тестирование	Продукт
Инженеры DevOps	Y FTE	X FTE	1,5 FTE
Тестировщики			1 FTE
Разработчики (back + front)			6 FTE

4.7 ЭТАП 7. АУДИТ ПО СОГЛАСОВАННЫМ МЕТРИКАМ

Кто может помочь:

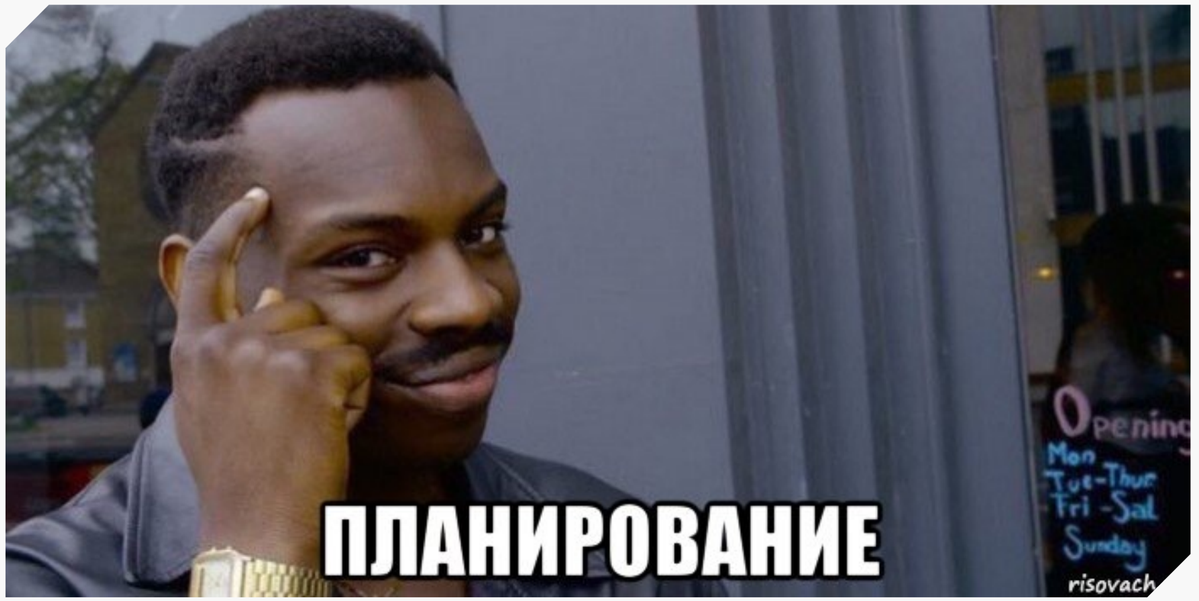
- Разработчики;
- Тестировщики;
- Архитектура;
- Лиды направлений.

4.8 ЭТАПЫ ВНЕДРЕНИЯ. ЭТАП 8

4.8 ЭТАП 8. ПЛАНИРОВАНИЕ РАБОТ

Планирование работ по улучшению зрелости продукта:

- Тип работ;
- Скоуп работ;
- Объём ресурсов команд;
- Согласование ресурсов;
- Согласование сроков.

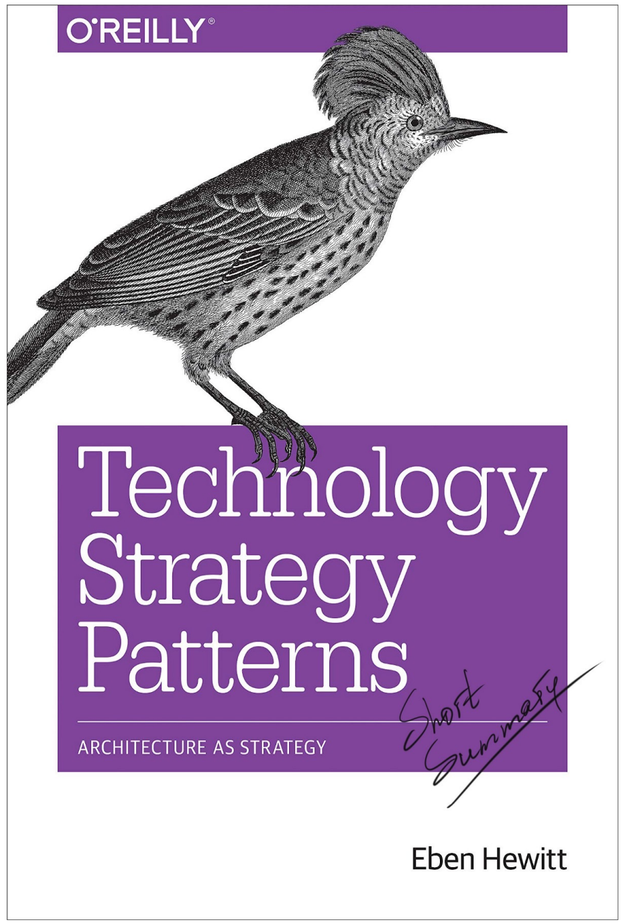


4.8 ЭТАП 8. КЕЙС ЗАЩИТЫ РАБОТ ПЕРЕД БИЗНЕСОМ

4.8 ЭТАП 8. КЕЙС ЗАЩИТЫ РАБОТ ПЕРЕД БИЗНЕСОМ

Докажем необходимость бизнесу:

- Говорим про ограничения и влияние;
- Готовим slide decks;
- Описываем постановку;
- Описываем технику/автоматизацию/процессы на понятном бизнесу языке;
- Описываем преимущества;
- Таймлайн внедрения;
- Согласование ресурсов.



4.8 ЭТАП 8. ДОКАЖЕМ БИЗНЕСУ НЕОБХОДИМОСТЬ

Проблематика ограничения в продукте X:

- недостаточная прозрачность процессов разработки;
- невозможность rollback или только ручной rollback;
- риски простоя тестировщиков из-за неработающих контуров;
- увеличенное время на troubleshooting проблемы CI/CD;
- невозможность реализации задач по АТ.

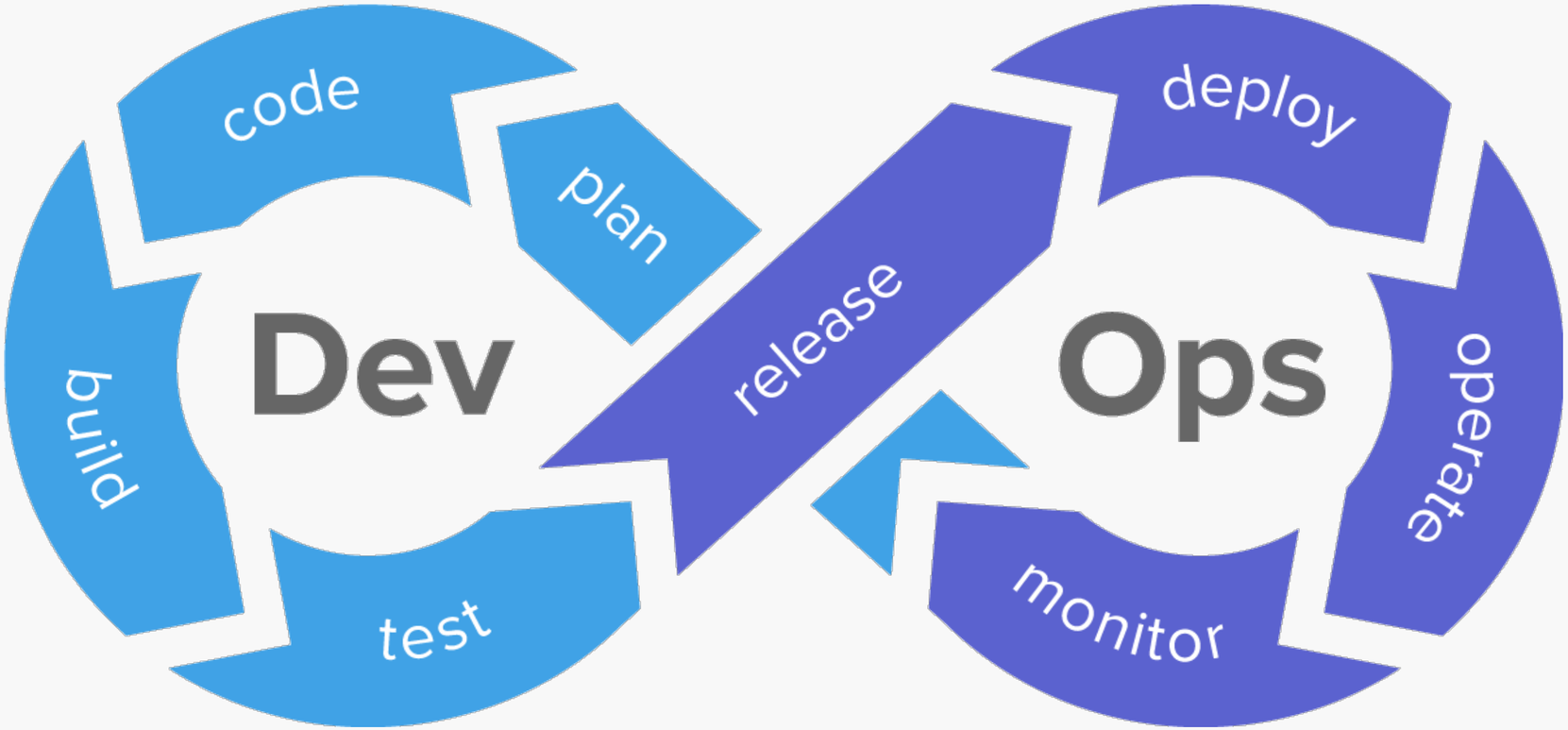
4.8 ЭТАП 8. ДОКАЖЕМ БИЗНЕСУ НЕОБХОДИМОСТЬ

Постановка задачи:

- Уменьшение времени разработки текущего количества фич в спринт в продукте X — решаемая задача;
- За счёт доп автоматизации = больше радости участникам цикла разработки;
- Возможность переиспользовать высвобожденные ресурсы на большее количество фич в спринт и инкремент продукта.

4.8 ЭТАП 8. ДОКАЖЕМ БИЗНЕСУ НЕОБХОДИМОСТЬ

- **Описание решения**



4.8 ЭТАП 8. ДОКАЖЕМ БИЗНЕСУ НЕОБХОДИМОСТЬ

CI:

- Повышение качества кода – SonarQube;
- Gitlab — QG rules merge PR;
- Покрытие кода модульными тестами;
- Автосборка и запуск асинхронных тестов в GitlabCI;
- Возможность версионирования продукта в Sonatype Nexus + Harbor.

4.8 ЭТАП 8. ДОКАЖЕМ БИЗНЕСУ НЕОБХОДИМОСТЬ

CD

- Ansible/Helm и Sonatype Nexus + Harbor с доп настройками и автоматизацией могут уменьшить T2M: экономия ресурсов по управлению окружениями и решению проблем на них;
- Контроль AT - TestIT.

4.8 ЭТАП 8. ДОКАЖЕМ БИЗНЕСУ НЕОБХОДИМОСТЬ

Мониторинг и логирование

EFK (elasticsearch + fluentd + kibana) – logs;

Prometheus + Grafana – perfmon, APP, metrics;

Jaeger + Open Telemetry – log-tracing.

Что даёт:

- обнаружение дефектов продукта по косвенным признакам;
- наблюдение за поведением персональных контуров;
- раннее обнаружение проблем с инфраструктурой;
- проактивное реагирование на инциденты.

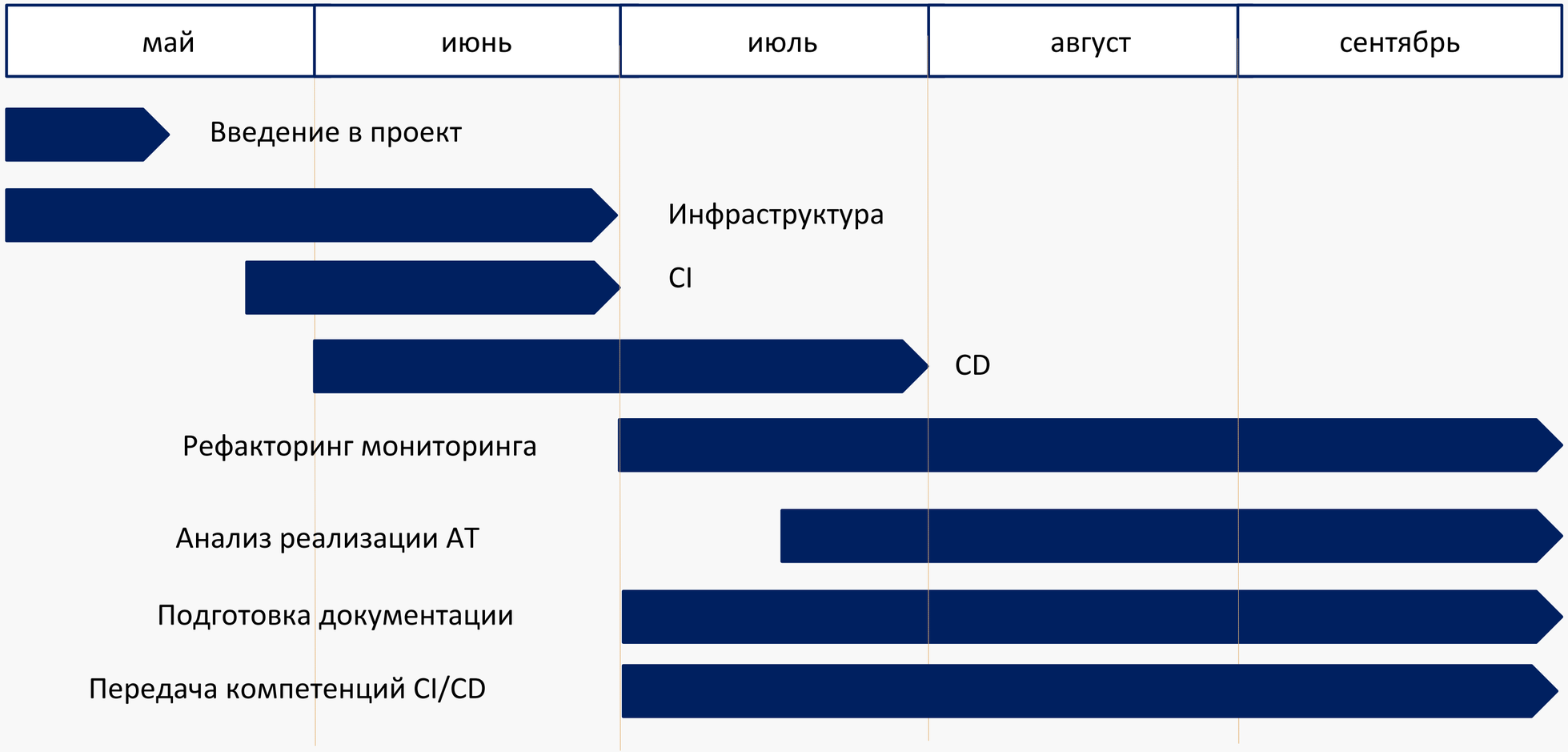
4.8 ЭТАП 8. ДОКАЖЕМ БИЗНЕСУ НЕОБХОДИМОСТЬ

Преимущества за счет автоматизации:

- Уменьшить расходы на релизный цикл, сократив человеческий фактор;
- Повысить чистоту и качество кода, снизив количество дефектов;
- Уменьшение T2M;
- Уменьшение времени сборки;
- Уменьшение времени на траблшутинг команды тестирования;
- Ускорение устранений сбоев.

4.8 ЭТАП 8. ДОКАЖЕМ БИЗНЕСУ НЕОБХОДИМОСТЬ

Таймлайн внедрения



4.8 ЭТАП 8. ДОКАЖЕМ БИЗНЕСУ НЕОБХОДИМОСТЬ

Аллокация ресурсов

Команда	Ресурс (FTE)	Задачи
• DEV • Test	N N	Консультации по особенностям разработки, прием экспертизы по CI/CD Проработка процедур АТ, прием экспертизы по CI/CD
DevOps	N	Подбор, настройка инструментов, разработка сценариев, обучение команд, документирование
ИТОГО	4N	

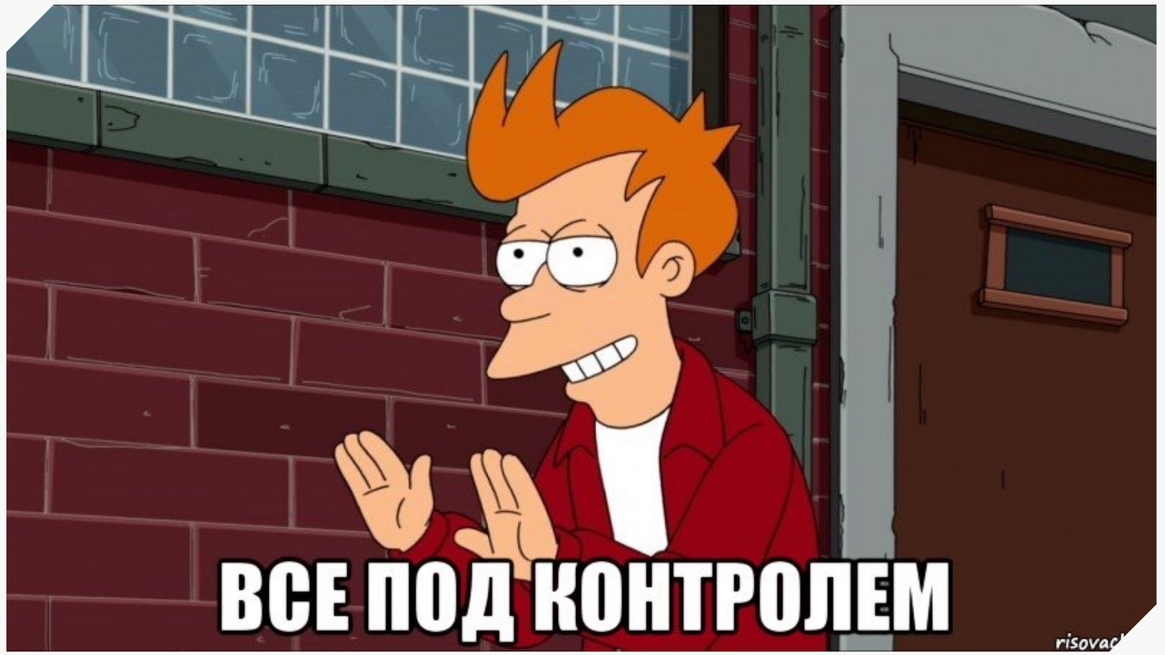
4.9 ЭТАПЫ ВНЕДРЕНИЯ. ЭТАП 9

4.10 ЭТАПЫ ВНЕДРЕНИЯ. ЭТАП 10

4.10 ЭТАП 10. КОНТРОЛЬ РАБОТ

Контроль:

- Статусы с product owner;
- Уточнение по срокам работ;
- Ведение работ;
- Контроль исполнителей;
- Выделение куратора работ от службы DevOps.



4.11 ЭТАПЫ ВНЕДРЕНИЯ. ЭТАП 11

Bimeister

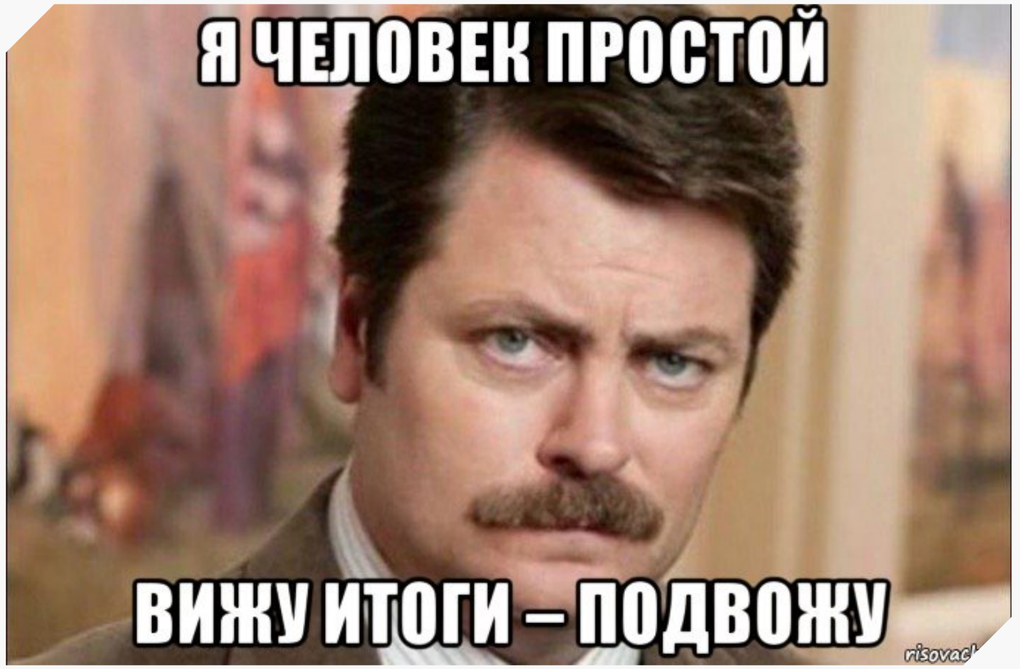
DEVOPS for love

РЕЗУЛЬТАТ ВНЕДРЕНИЯ

5. ПОДВЕДЁМ ИТОГИ

Реализовано:

- внедрена матрица зрелости продуктов;
- сформированы точки контроля и временные срезы;
- реализован прозрачный контроль за развитием продукта;
- понятен и реализован процесс снижения T2M.



5. ПОДВЕДЁМ ИТОГИ

Профит:

- Есть единый прозрачный процесс контроля развития технической части продукта и взаимодействия всех участников цикла разработки;
- Детальные метрики зрелости для бизнеса с пониманием, что на что влияет;
- За счёт доп автоматизации:

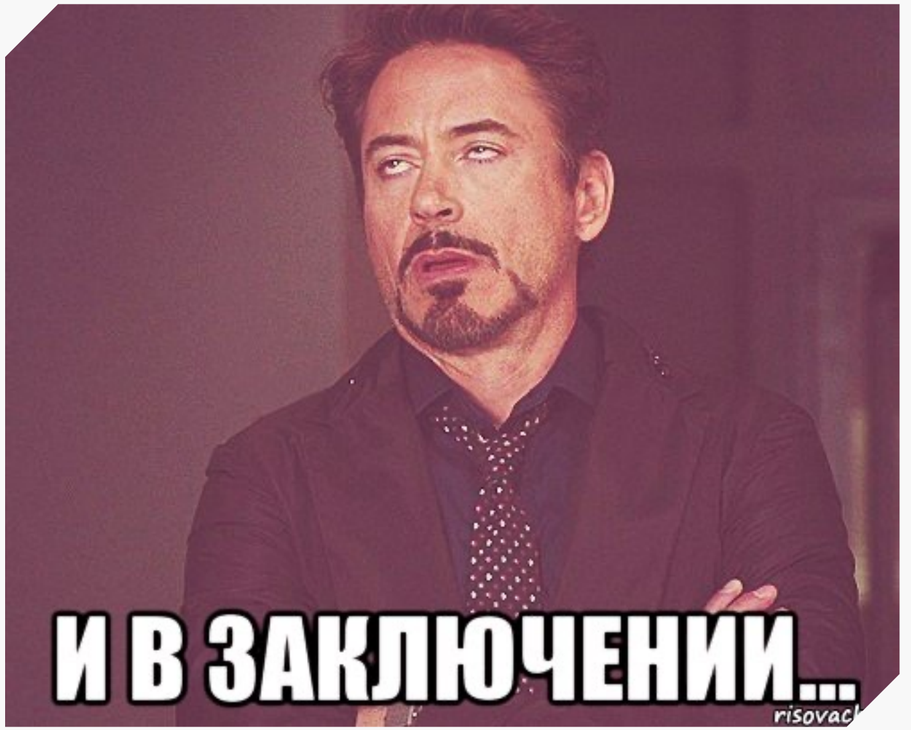
  - больше фич в sprint с текущими ресурсами;

  - повышение качества тестирования спринтов;

  - повышение чистоты кода.

ЗАКЛЮЧЕНИЕ

- при внедрении процесса найдите пилот и того, кому его продать, на ком обкатать;
- полноценное внедрение заняло 1,5 месяца (до 9 этапа);
- первая полная картина работ от матрицы – квартал;
- когда один продукт или до 5, то это быстро.



КОНТАКТЫ

https://youtu.be/nn_8oJHbEug

DEVOPS for love

https://t.me/devopsforlove

https://youtu.be/5836KL9jRkM

СПАСИБО ЗА ВНИМАНИЕ!