Express 42

MAGNIT OMNI

Разделяй и властвуй

BSA-модель для IaC и практика её применения в «Магнит OMNI»

Express 42

Спикеры

Андрей Шилов

Старший инженер
по практикам DevOps, «Флант»

andrey.shilov@flant.com

Владимир Дроздецкий

DevOps Team Lead,
«Магнит OMNI»

mrgreyves@gmail.com

О чём будем говорить

01 **Проблема:** хаос в управлении IaC

02 **Решение:** BSA-модель

03 **Реализация:** подходы и инструменты для внедрения

04 **Практика:** история внедрения в «Магнит OMNI»

05 **Советы:** что важно учесть при старте

О компании «Флант»

Express 42

17 +
лет опыта
в Open Source

С 2017
года используем
Kubernetes в production

№ 1
контрибьютор в проекты
CNCF из России

500 +
сотрудников

>260
компаний-пользователей

В топе
вендоров ИТ-решений для банков *
и промышленности **

Реестр
российского ПО

Лицензии и сертификат
ФСТЭК России

АРПП «Отечественный
софт»

* Рейтинг «Крупнейшие ИТ-вендоры в банках», TAdviser, 2024 год
** Рейтинг «Крупнейшие ИТ-вендоры в промышленности», TAdviser, 2024 год

ФЛАНТ

Синергия опыта вендора, интегратора, сервисной и консалтинговой компании

Deckhouse

Deckhouse — продуктовое подразделение, разработчик продуктов для построения надёжной enterprise-инфраструктуры

DaaS

DaaS — комплексное DevOps-сопровождение инфраструктуры в режиме 24/7 силами выделенной DevOps-команды

Express 42

«Экспресс 42» — DevOps-консалтинг. От анализа узких мест в ИТ-процессах до создания роадмапа изменения ИТ для реализации цифровой трансформации

Когда IaC не спасает, а создаёт новую боль

Неструктурированный IaC автоматизирует беспорядок, а не решает проблему

Опыт наших клиентов

IaC есть, а порядка нет

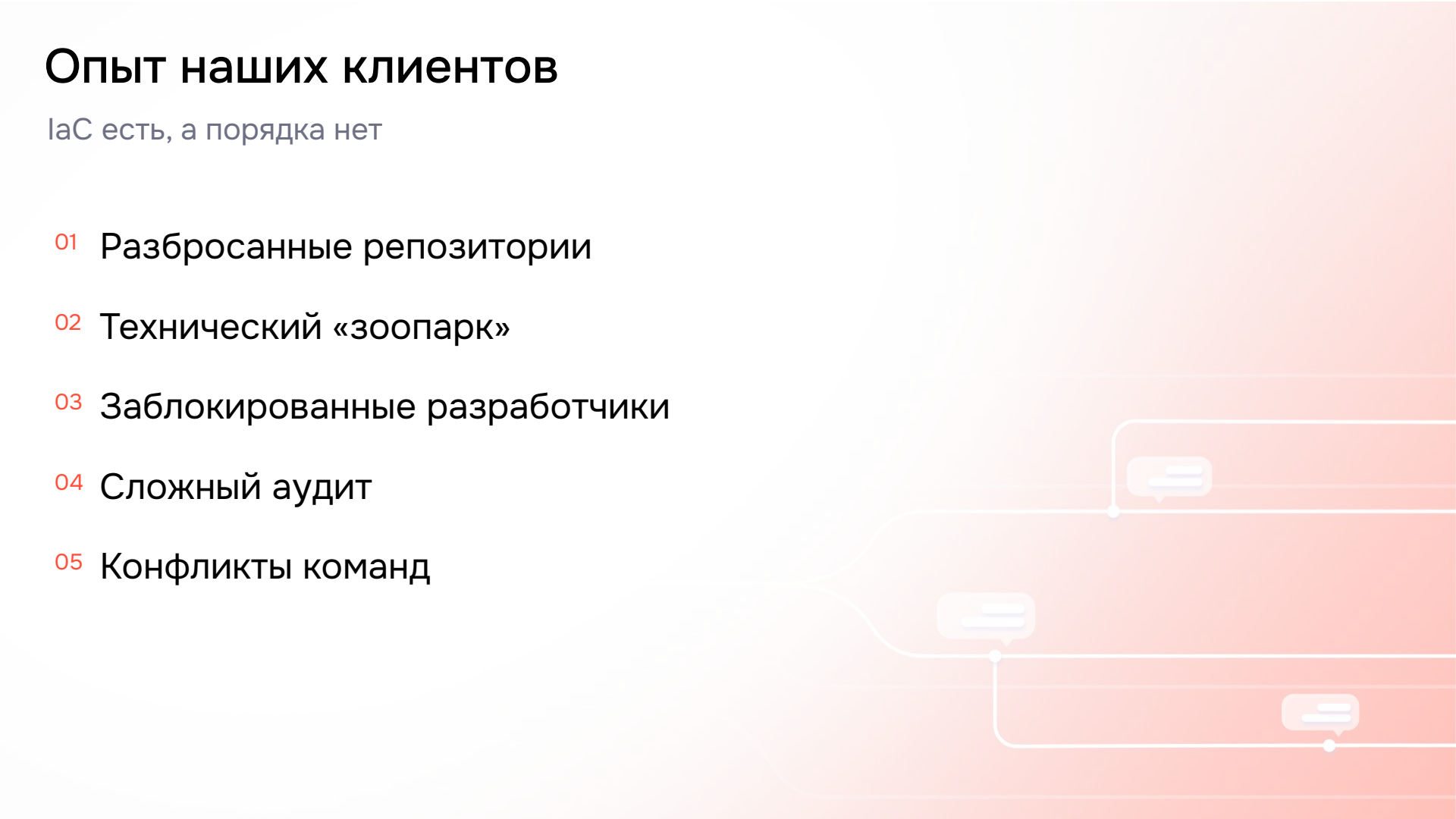
01 Разбросанные репозитории

02 Технический «зоопарк»

03 Заблокированные разработчики

04 Сложный аудит

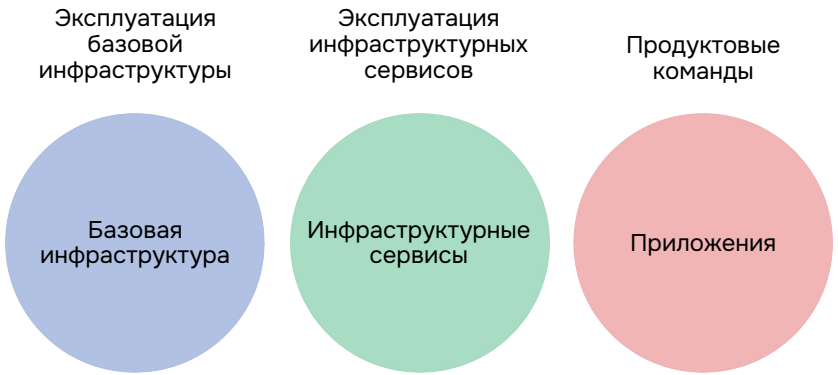
05 Конфликты команд



Почему так происходит? Закон Конвея

Чтобы изменить систему, нужно изменить схему коммуникаций

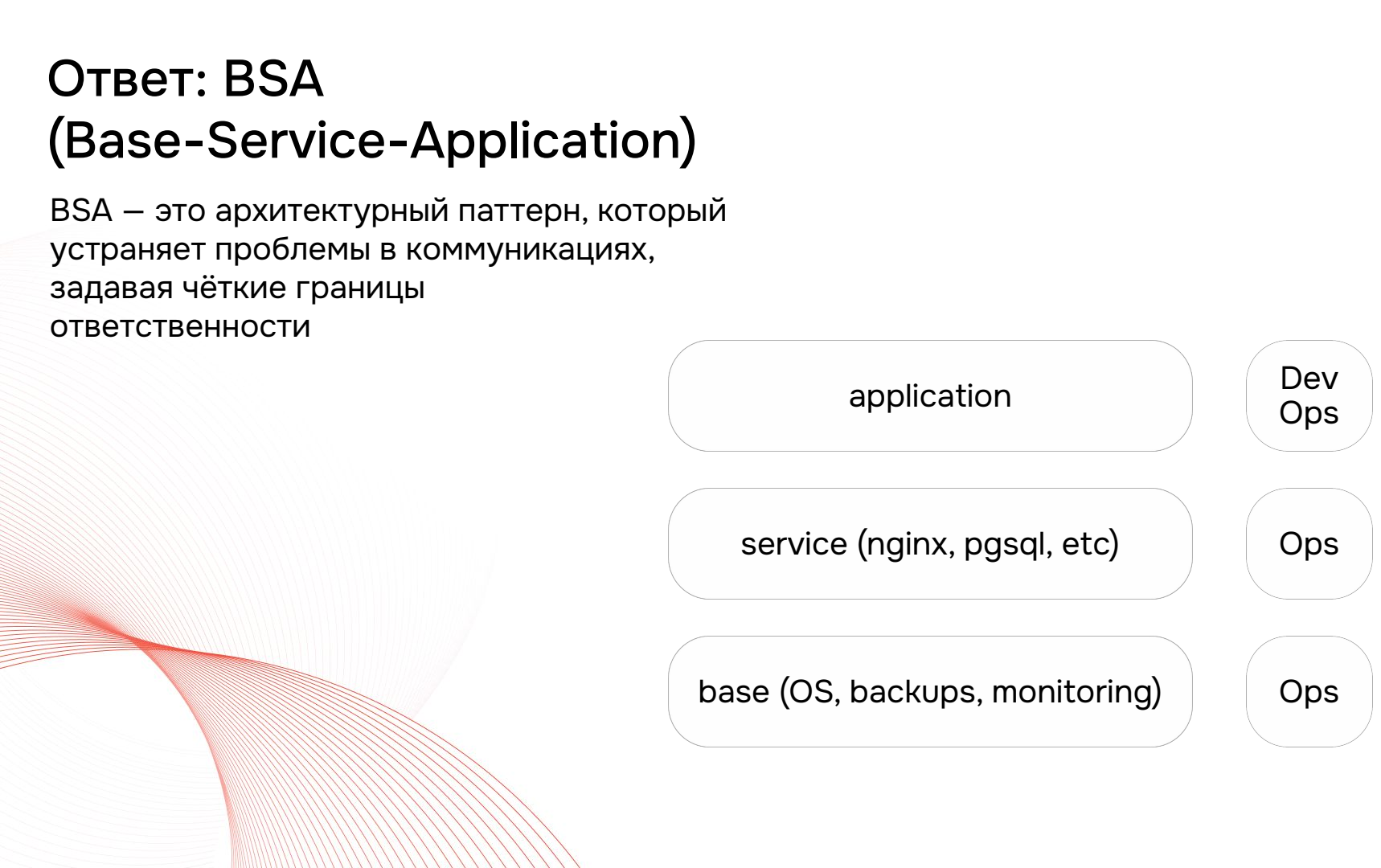
До



После

Ответ: BSA (Base-Service-Application)

BSA — это архитектурный паттерн, который устраняет проблемы в коммуникациях, задавая чёткие границы ответственности



Историческая справка
Модель BSA

2006

AWS запускает EC2, появляются первые предпосылки для управления инфраструктурой как кодом

2011

Команда Skype (задолго до популяризации IaC) сталкивается с вызовами масштабирования и начинает писать код для управления инфраструктурой. На практике рождаются подходы, которые позже назовут BSA

2016

Выходит книга Кифа Морриса «Infrastructure as Code», термин становится общеупотребительным

Модель предложена Александром Титовым в далёком 2011 году

Суть модели BSA: что в каждом слое?

Base (Базовый)

Ответственность:

Инфраструктурная команда

Примеры артефактов:

- облачные аккаунты
- сети (VPC)
- политики безопасности (IAM)
- базовые образы ВМ

Service (Сервисный)

Ответственность:

Платформенная команда (DevOps/SRE)

Примеры артефактов:

- кластеры (Kubernetes), управляемые БД (PostgreSQL)
- очереди (Kafka)
- стек мониторинга (Prometheus/Grafana)

Application (Приложение)

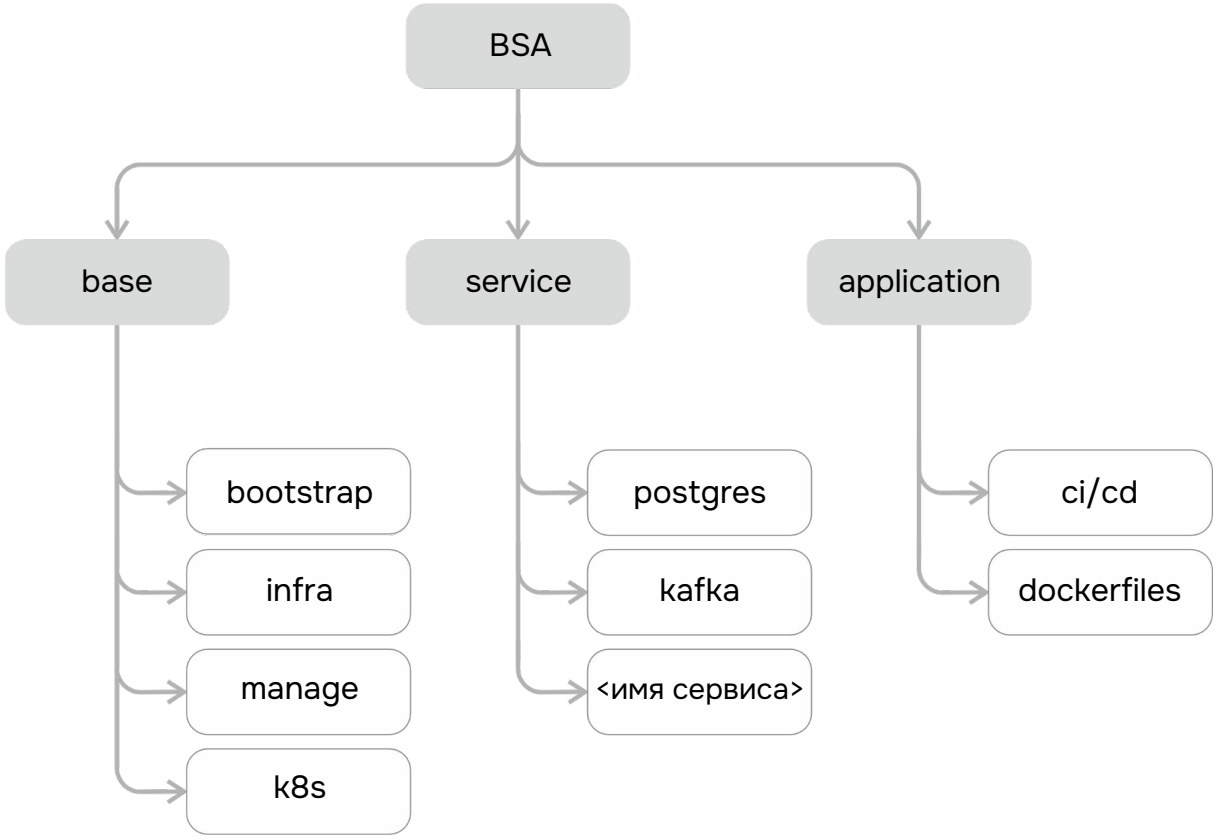
Ответственность:

Команда разработки

Примеры артефактов:

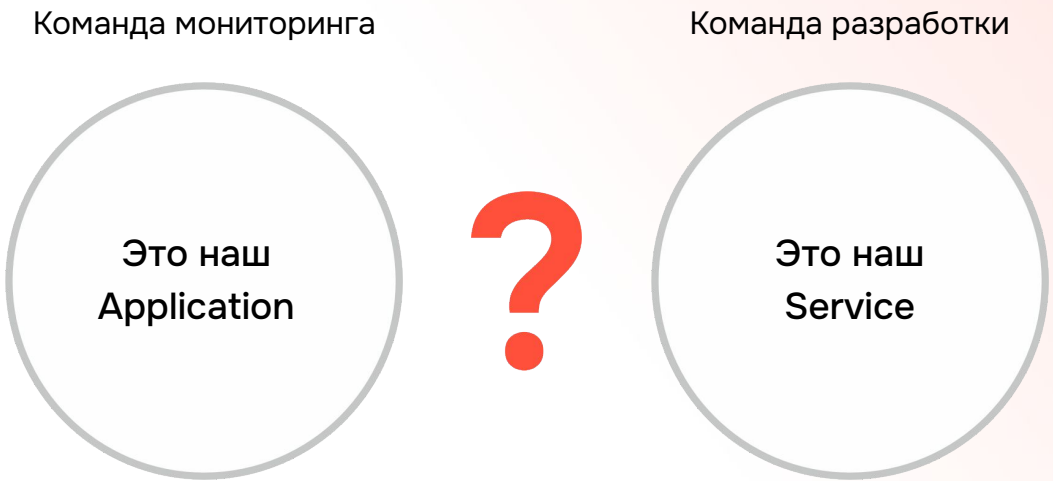
- пайплайны CI/CD (Jenkinsfile, .gitlab-ci.yml)
- конфигурации приложения (ConfigMap, Helm values)
- манифесты Kubernetes

Пример реализации



Конфликт слоёв

Без единого взгляда —размытые границы,
споры и недопонимание

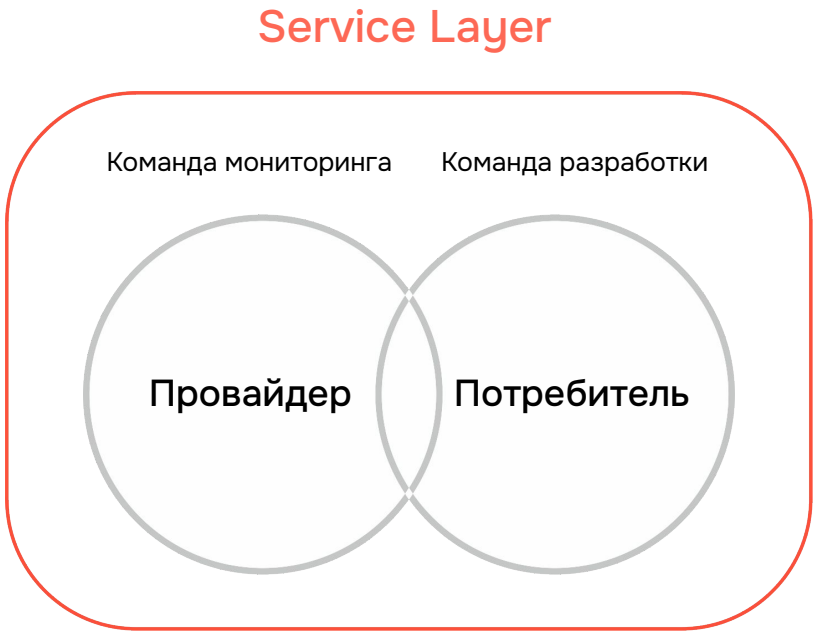


Главное условие — договориться

01 Команды должны согласовать, кто за какой слой отвечает

02 Код должен отражать эти границы (структура репозиториев, модулей)

03 Процессы должны закрепить ответственность (код-ревью, приёмка изменений)

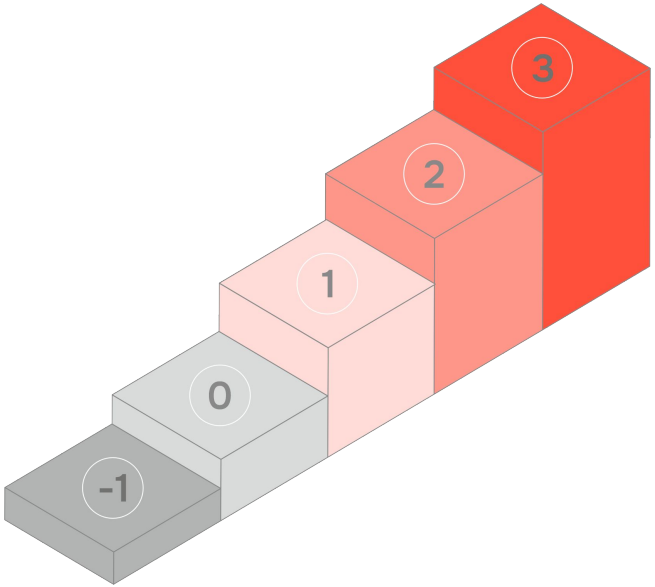


Подробнее о модели зрелости IaC

Express 42

Куда мы растём?

Модель зрелости IaC



-1 Регрессивный

- Часть инфраструктуры развёрнута и управляется кодом
- Инфраструктурное развёртывание сопровождается большим количеством действий, выполняемых вручную
- Инфраструктурный код написан без использования инструментов и подходов, которые уже устоялись в отрасли разработки программного обеспечения
- Инфраструктурный код, процессы и процедуры документированы плохо и недоступны для всех заинтересованных сторон

0 Повторяемый

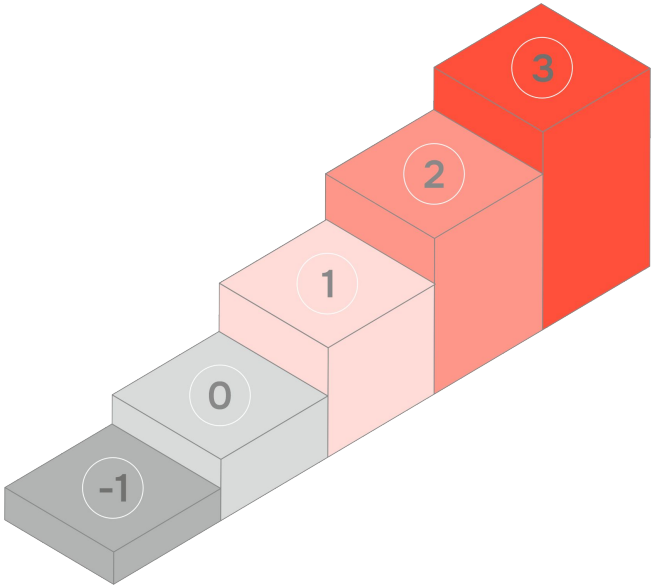
- Весь инфраструктурный код и конфигурации расположены в системе контроля версий
- Тестирование, развёртывание инфраструктуры и управление ею выполняются в рамках автоматизированного пайплайна
- Весь код, процессы и процедуры задокументированы и доступны
- «Неизменяемость» (immutability) инфраструктуры

1 Улучшенный

- Полностью автоматизированные развёртывание и управление инфраструктурой
- Минимальное использование неподдерживаемых или самописных инструментов для работы с инфраструктурой
- Применяются модульные тесты, код постоянно тестируется при каждой регистрации в системе контроля версий
- Используются шаблоны конфигурационных файлов
- Автомасштабирование на основе параметров работы нагрузки, задаваемых пользователями

Куда мы растём?

Модель зрелости IaC



2 Количественно управляемый

- Реализована возможность выполнения автоматизированного отката (rollback) изменений
- Использование сервисов для хранения внешней конфигурации с понятными механизмами внесения изменений
- Весь код, процессы и процедуры хорошо документированы в системе управления знаниями
- Инфраструктурный код использует декларативную, а не императивную модель программирования

3 Оптимизированный

- Самовосстанавливающаяся, самонастраиваемая, самооптимизирующаяся инфраструктура
- Производительность тестируется и отслеживается на основании бизнес-KPI
- Уровень загрузки инфраструктуры оптимально высокий, инфраструктура используется максимально эффективно
- Создаваемый инфраструктурный код инвариантен относительно поставщика(-ов) облачных услуг

BSA и уровни зрелости: как они связаны

-1	Регрессивный	Модель BSA не применяется
0	Повторяемый	Структура репозиториев (base/, service/, app/), первичное разделение ответственности, стандарты именования
1	Улучшенный	Модульность слоёв, переиспользуемые компоненты, чёткие интерфейсы между командами, база для самообслуживания (capabilities)
2	Количественно управляемый	Возможность собирать метрики по каждому слою отдельно (например, время создания ресурса в Base, частота деплоя в Application)
3	Оптимизированный	Автономность слоёв, унифицированные интерфейсы, сквозная наблюдаемость, инвариантность к провайдерам

Подходы к реализации и инструменты слоя Base

Инструменты

Provisioning (Base, Service):

- Terraform
- Terragrunt
- OpenTofu
- Pulumi
- Crossplane

Успех BSA зависит от архитектурного разделения кода, а не от выбора конкретного инструмента

Подходы к реализации и инструменты слоя Service

Инструменты

Конфигурация (Service):

- Terraform
- Ansible
- OpenTofu
- Chef
- Puppet

Успех BSA зависит от архитектурного разделения кода, а не от выбора конкретного инструмента

Подходы к реализации и инструменты слоя App

Инструменты

Шаблоны (Application):

- Helm
- Kustomize
- ArgoCD
- FluxCD
- Gitlab CI

Успех BSA зависит от архитектурного разделения кода, а не от выбора конкретного инструмента

Почему это работает?

Ключевые принципы

01 Явное разделение ответственности

02 Стандартизация через код

03 Самообслуживание разработчиков

04 Независимость от вендора

05 Фокус на поставке ценности

Итоговый результат: что вы получаете?

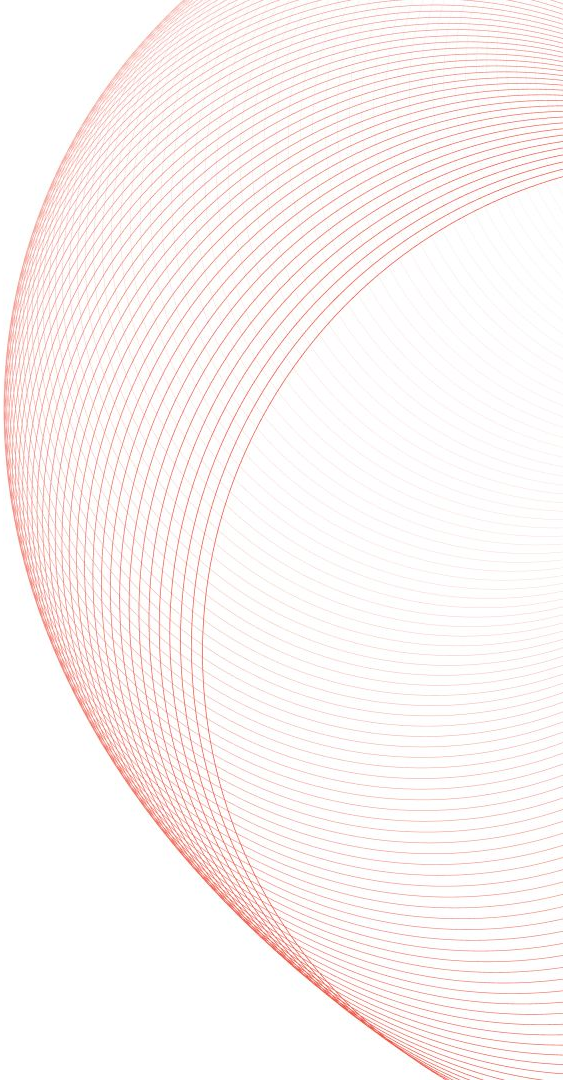
Было

- **Процессы:**
  Ручные, уникальные, долгие
- **Ответственность:**
  Размытая, вызывает конфликты
- **Качество:**
  Непредсказуемое, зависит от человека
- **Скорость:**
  Низкая, разработчики ждут

Стало

- **Процессы:**
  Автоматизированные, стандартные, прогнозируемые
- **Ответственность:**
  Чёткая, по слоям
- **Качество:**
  Контролируемое через код и ревью
- **Скорость:**
  Высокая, самообслуживание

А как это выглядит в реальном проекте?



Internal Development Platform для зрелой разработки: опыт с IDP от Deckhouse

Анонс вебинара

Приглашаем на вебинар, где разберём, почему IDP — это больше, чем Kubernetes. Обсудим, с какими ограничениями сталкиваются команды при попытке построить платформу самостоятельно и как Deckhouse Development Platform помогает выстроить сервисный подход к разработке, не повторяя ошибок, с которыми команды сталкиваются при самостоятельном построении IDP.

Спикеры:

Алексей Зимонин — старший технический лид по практикам DevOps, «Экспресс 42» (АО «Флант»)

Никита Вельгин — технический менеджер продукта Deckhouse Development Platform

26 марта, 12:00

Зарегистрироваться на вебинар

Заключение

Слои - Base, Service, Application

Чёткая архитектура вместо беспорядка

Договорённости — единый POV, роли провайдер/потребитель

Общий язык для команд

Результат — предсказуемость, скорость, самообслуживание

Подтверждено опытом «Магнит OMNI»

Express 42

Приходите за экспертизой!

input@express42.com

express42.com

Telegram-канал
«Экспресс 42»