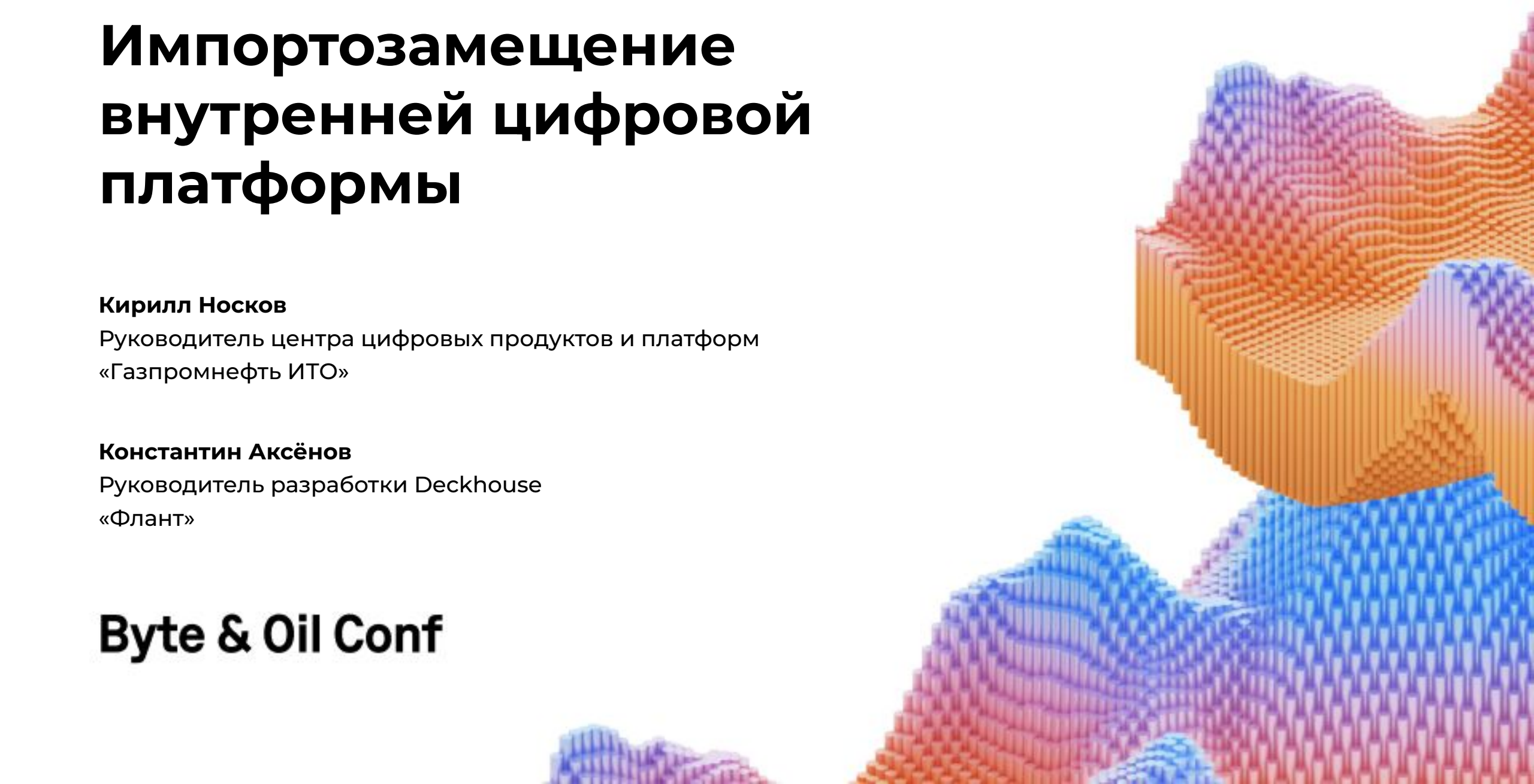
Импортозамещение внутренней цифровой платформы

Кирилл Носков
Руководитель центра цифровых продуктов и платформ
«Газпромнефть ИТО»

Константин Аксёнов
Руководитель разработки Deckhouse
«Флант»

Byte & Oil Conf



Кирилл Носков

Руководитель центра цифровых продуктов и платформ «Газпромнефть ИТО»

noskov.kg@gazprom-neft.ru

Чем занимаюсь

Владелец продукта, руковожу разработкой и изменениями
Практикую K8s с 2017 года

Опыт

- С 2008 года занимаюсь инфраструктурной разработкой
- С 2019 года работаю в компании «Газпромнефть ИТО»
- С 2020 года владею продуктом «Эластичная инфраструктура»

С чем работаю больше всего

kubernetes

FLANT Deckhouse Kubernetes Platform

N1

Константин Аксёнов

Руководитель разработки
Deckhouse Kubernetes Platform

konstantin.aksenov@flant.com
github.com/konstantin-axenov

Чем занимаюсь

Больше 5 лет засыпаю и просыпаюсь с мыслями о Kubernetes

Опыт

- С 2011 года занимаюсь разработкой
- С 2017 года работаю в компании «Флант»
- С 2020 года руководитель разработки Deckhouse K8s Platform

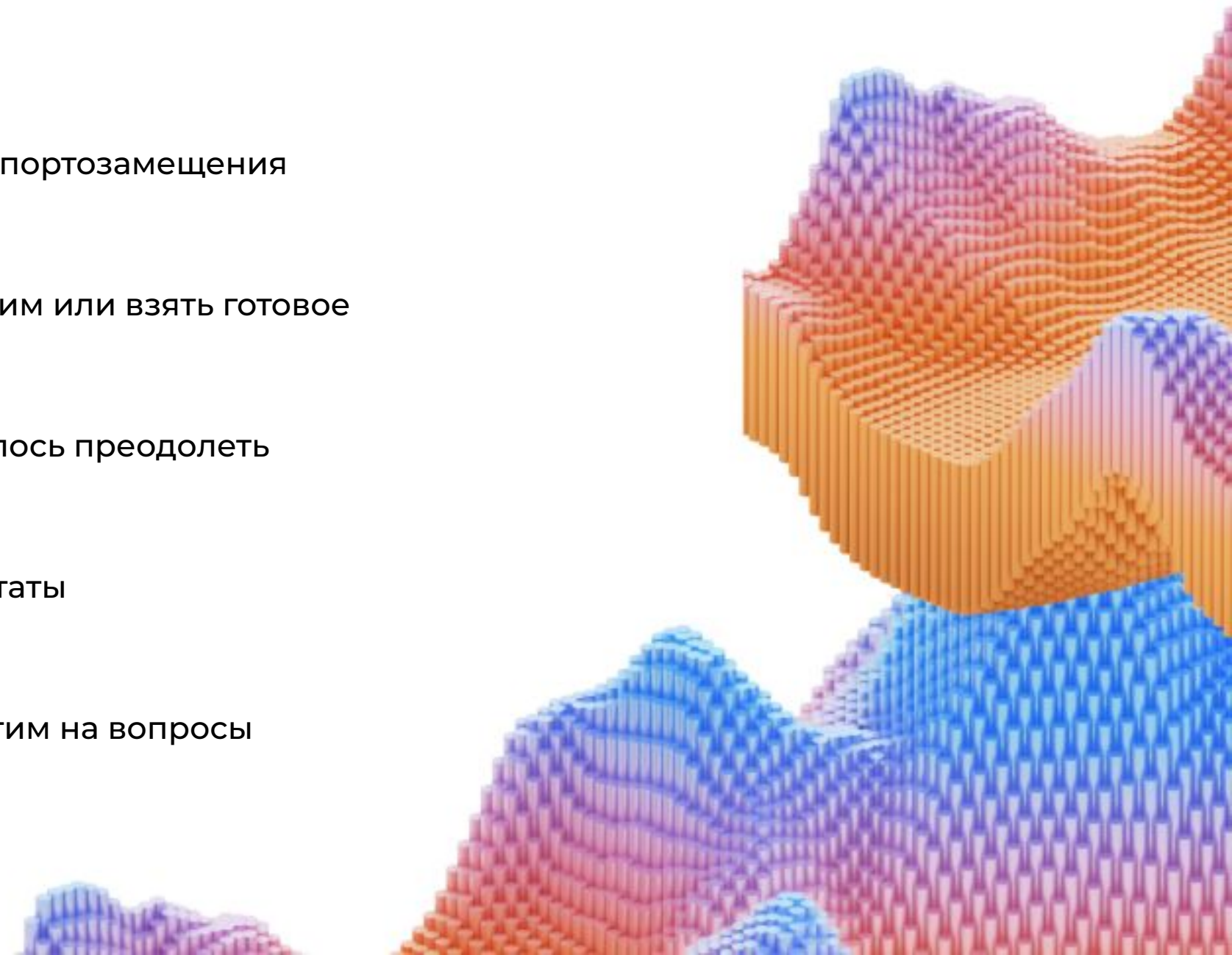
С чем работаю больше всего

kubernetes

FLANT Deckhouse Kubernetes Platform

План

1. Продукт N1 и задачи импортозамещения
2. Выбор пути: делать самим или взять готовое
3. Какие проблемы пришлось преодолеть
4. Изменения и их результаты
5. Подведём итоги и ответим на вопросы



Описание продукта

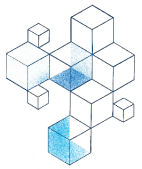
Что такое N1

- Набор PaaS- и SaaS-сервисов и инструментов, системно решающих задачи проектных команд по реализации цифровых продуктов внутри компании.
- Что получают разработчики: инфраструктуру, инструменты для организации разработки DevSecOps.
- Работа с данными, интеграция с другими системами компании.
- Цель: вовремя давать нужные бизнесу продукты.

Какие задачи решают сервисы N1?

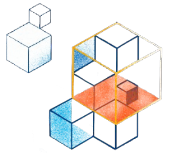
Реализация продуктов

Интеграция



Инфраструктура

Бизнес-инструменты



Данные

Машинное обучение

Что N1 дал разработке?

В 6 РАЗ

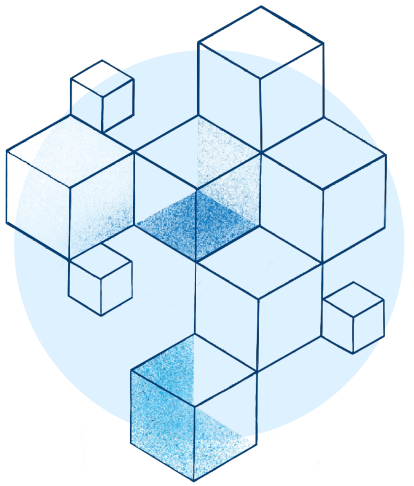
сокращен срок поставки кода в периметр компании

5 МИНУТ

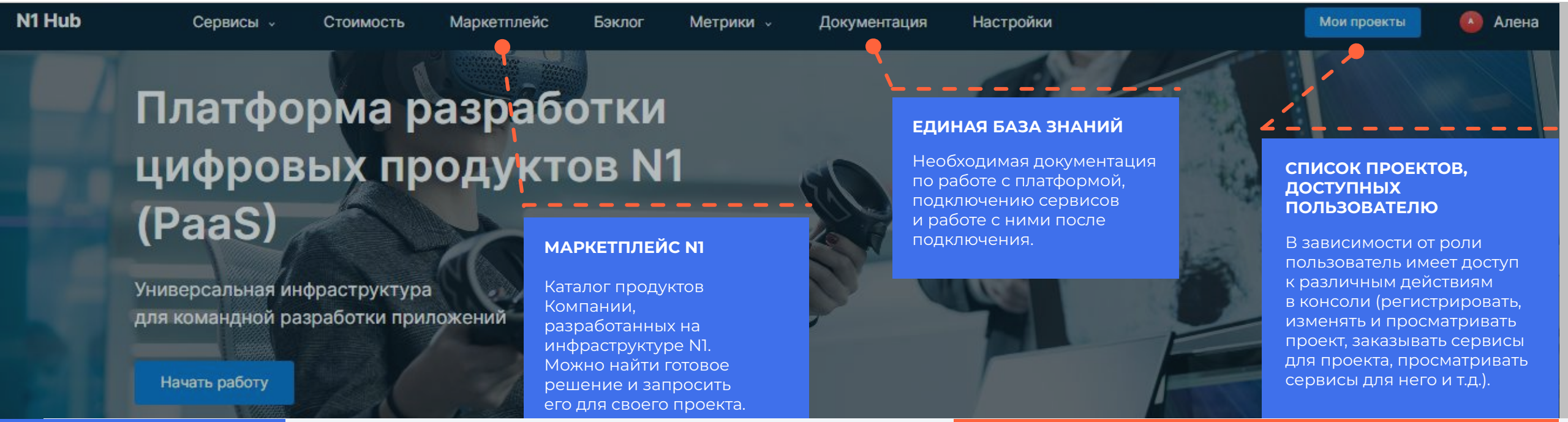
средний срок разворачивания на N1

Ключевые особенности и преимущества цифровой платформы:

1. Сервисы и инструменты
2. Сквозное управление доступами
3. Портал & команда customer success



Единая точка доступа. Портал N1 hub*



МАРКЕТПЛЕЙС N1

Каталог продуктов Компании, разработанных на инфраструктуре N1. Можно найти готовое решение и запросить его для своего проекта.

ЕДИНАЯ БАЗА ЗНАНИЙ

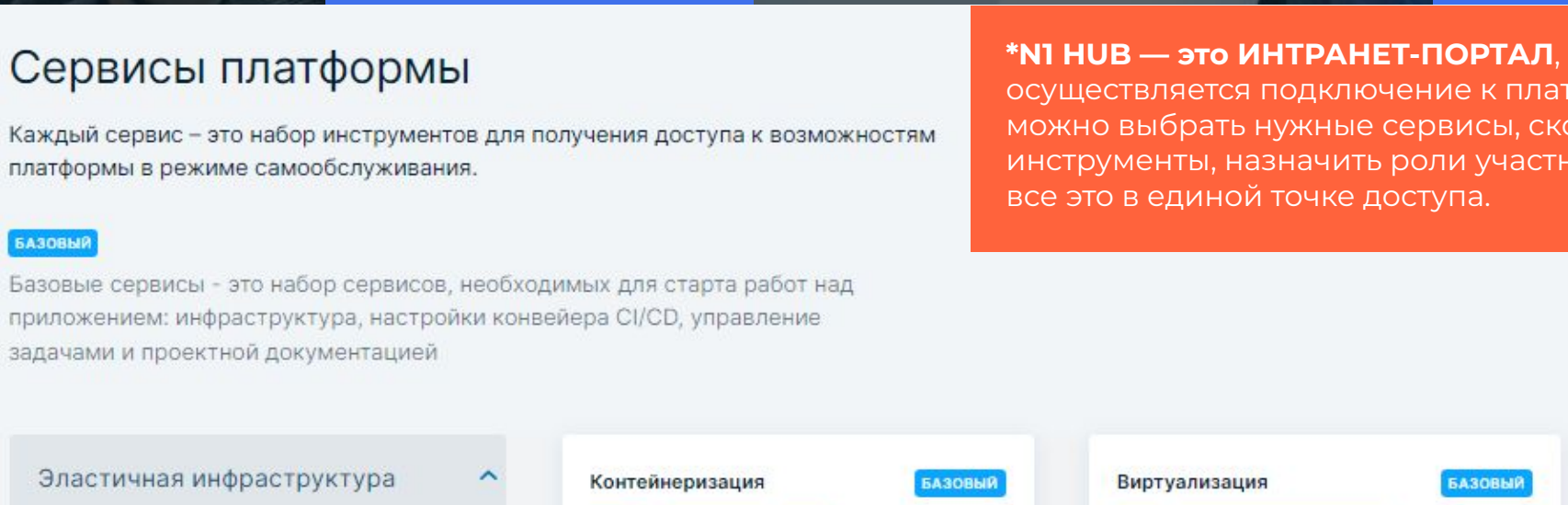
Необходимая документация по работе с платформой, подключению сервисов и работе с ними после подключения.

СПИСОК ПРОЕКТОВ, ДОСТУПНЫХ ПОЛЬЗОВАТЕЛЮ

В зависимости от роли пользователь имеет доступ к различным действиям в консоли (регистрировать, изменять и просматривать проект, заказывать сервисы для проекта, просматривать сервисы для него и т.д.).

КАТАЛОГ СЕРВИСОВ

Сервисы разделены на группы/подгруппы. У каждого сервиса есть описание, документация по его использованию и кнопка «Подключить» либо инструкция по его заказу.



***N1 HUB — это ИНТРАНЕТ-ПОРТАЛ**, через который осуществляется подключение к платформе N1. Здесь можно выбрать нужные сервисы, сконфигурировать инструменты, назначить роли участникам команды — все это в единой точке доступа.

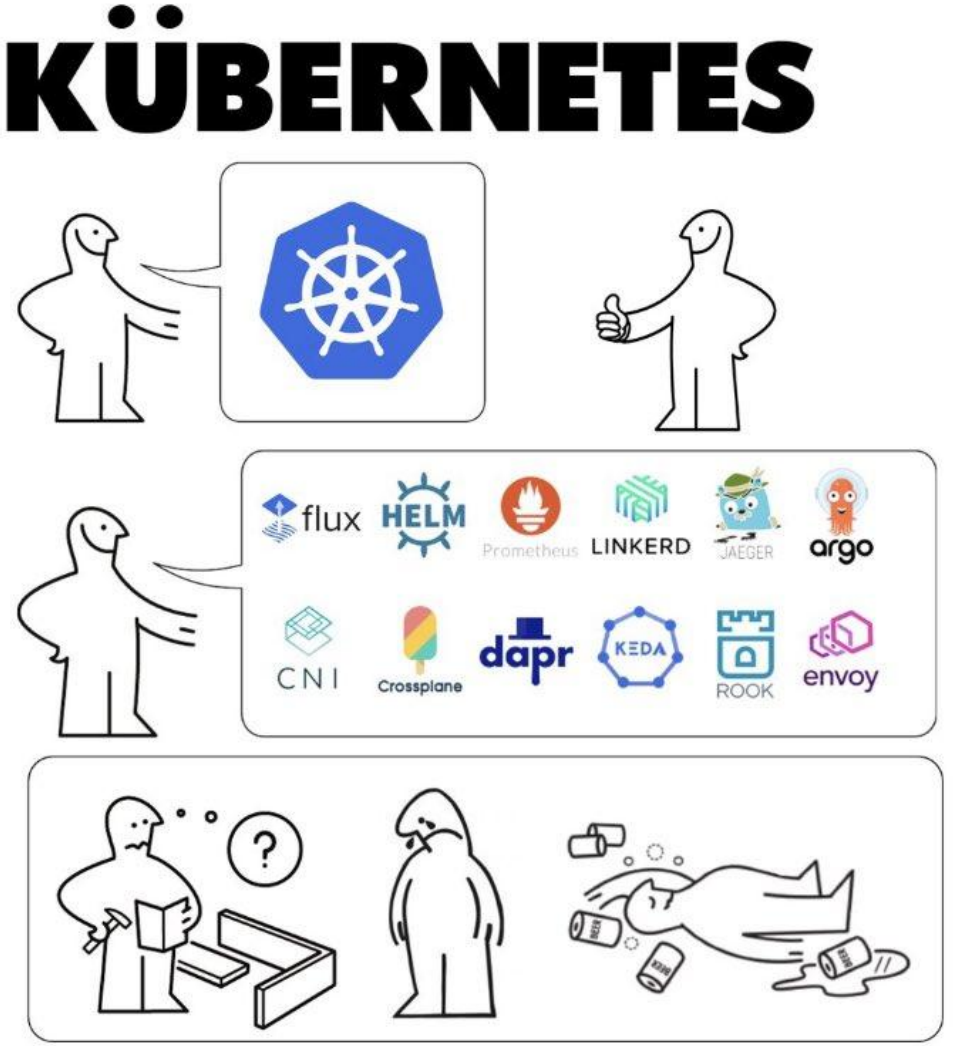
Kubernetes-платформа для замещения самостоятельно

Основные плюсы:

- Очень интересная работа
- Полный контроль

Основные минусы:

- Долго
- Нет гарантии результата



Стоимость самостоятельной разработки

- Для запуска и поддержки 2 кластеров (например, Dev/Test и Prod) требуются 2 инженера.
- Высокий bus-фактор — люди болеют, увольняются и придется начинать сначала, а время запуска — около 3 месяцев.
- Для поддержки и обновлений нужны full-time-специалисты со средней зарплатой с налогами 400-500 тыс. руб.
- Для 3-6 и более кластеров требуется больше людей
- Если кластеров 10 и более, то речь уже идёт про платформу, автоматические обновления, более серьёзную автоматизацию, и стоимость такого решения может уходить в бесконечность.

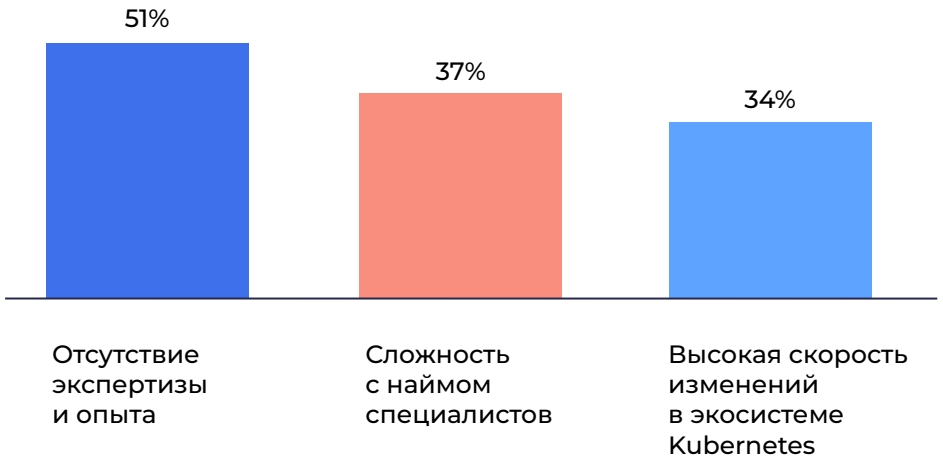
	1-2 кластера	3-6 кластеров	6 и более кластеров
Кол-во инженеров	1-2	2-4	>10
Время запуска	1-3 месяца	5-6 месяцев	От года
ЗП в год, руб.	5-14,4 млн	10-28,8 млн	От 50 млн

Проблемы с «ванильным» Kubernetes

Основными проблемами являются отсутствие экспертизы и необходимого опыта.

Исследование VMware The State of Kubernetes 2022

Вызовы при использовании Kubernetes



Сам по себе Kubernetes — это конструктор

Какую ОС выбрать для узлов кластера? Как обновлять ОС?

Какую архитектуру кластера выбрать? etcd на отдельных узлах?

Какой CNI выбрать? Cilium или Calico? containerd или Docker?

Какой Ingress controller выбрать? NGINX ingress или Envoy?

Что использовать для хранения метрик? Prometheus или VictoriaMetrics?

Мне нужно управление сертификатами? Multicloud или hybrid cloud?

Мне нужен Service Mesh? Istio или что-то другое?

Что такое HPA? Что такое VPA? А что с безопасностью?

Где взять людей, которые всё это умеют? 🙂

Появились для решения «проблемы» Kubernetes

Managed Kubernetes от облачных провайдеров

Kubernetes-платформы

Managed Kubernetes от облачных провайдеров

Основные плюсы:

- Быстрый старт
- Оплата за фактическое потребление

Основные минусы:

- Использование публичных облаков небезопасно
- Отсутствие кастомизаций

Kubernetes-платформы или Kubernetes-дистрибутивы

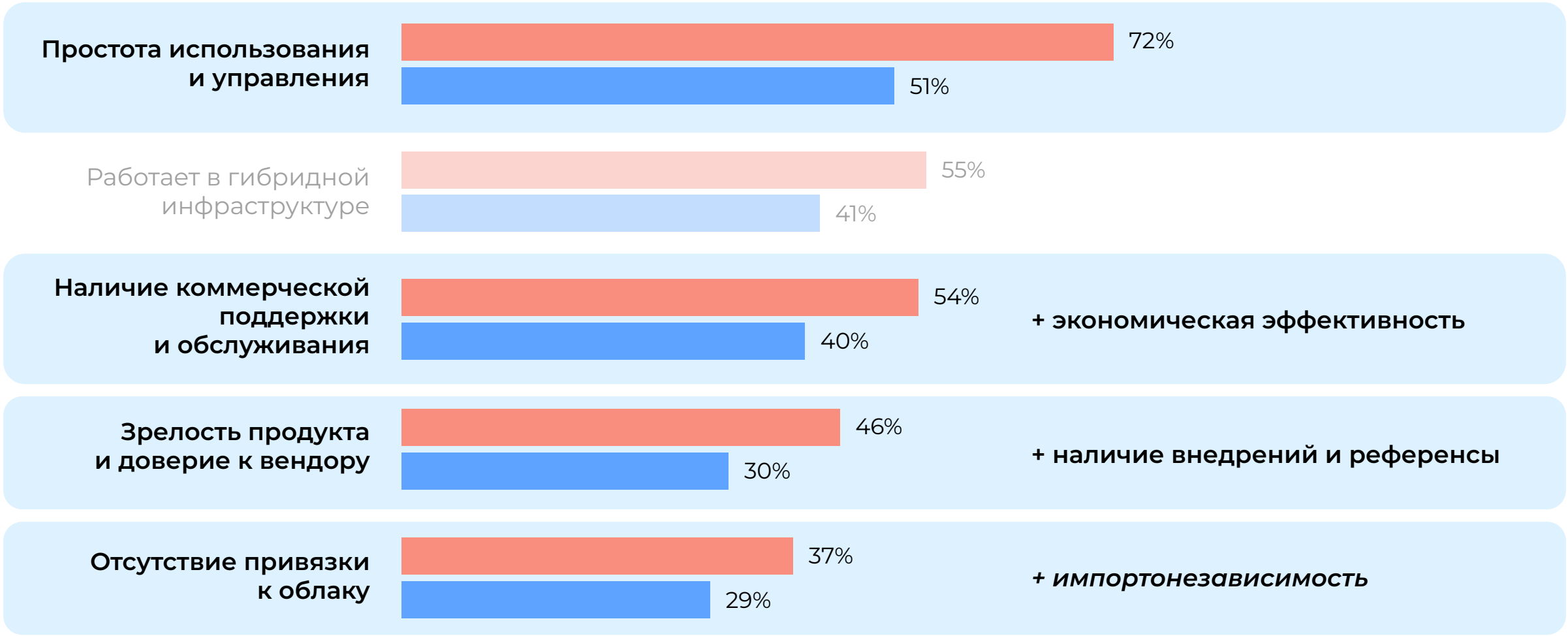
Решают задачи:

- Работа в локальной инфраструктуре
- Возможность настроить под себя
- Поддержка отечественных операционных систем

Критерии выбора Kubernetes-платформы

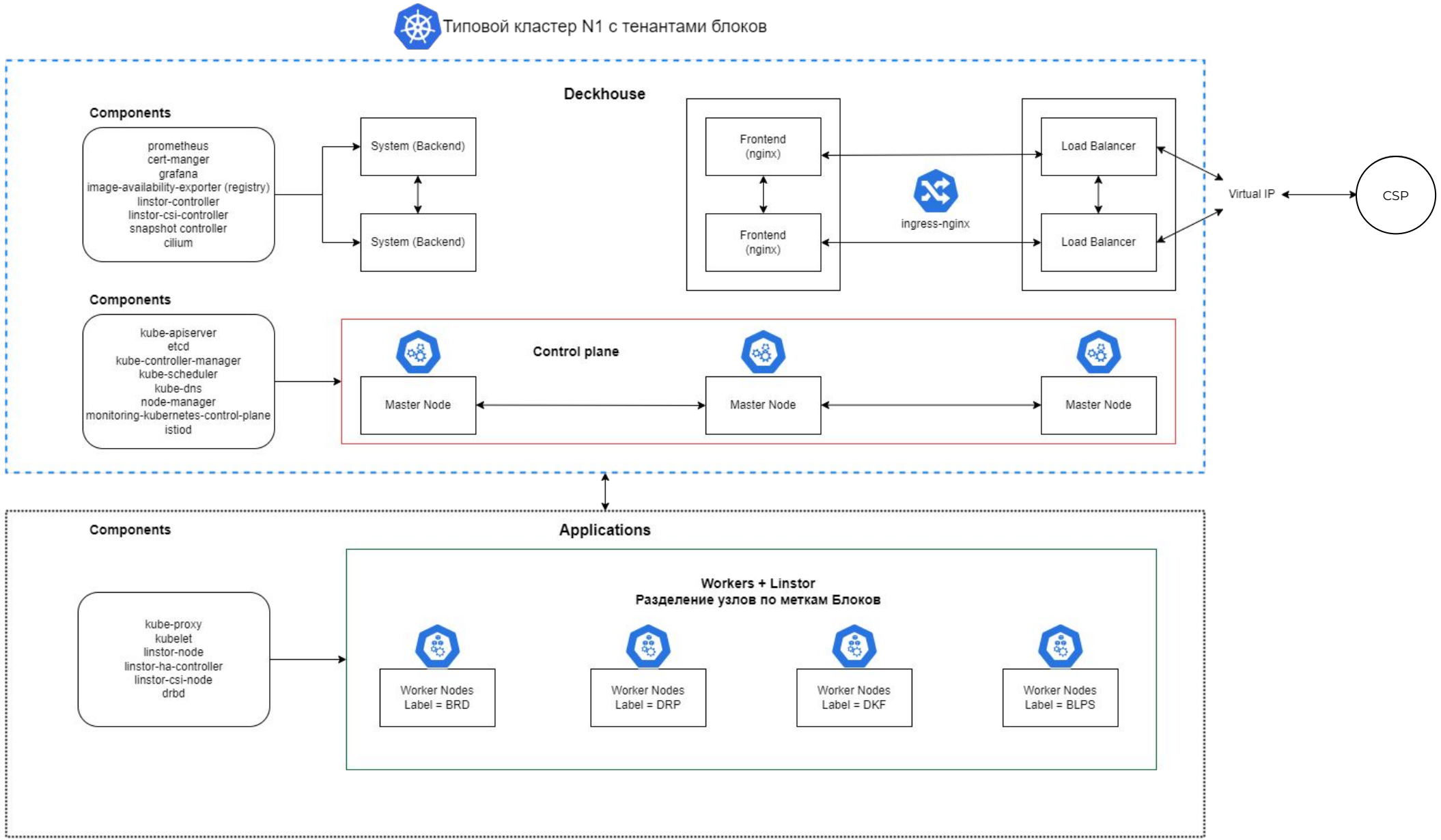
Исследование VMware The State of Kubernetes 2023

Критерии выбора Kubernetes-платформы

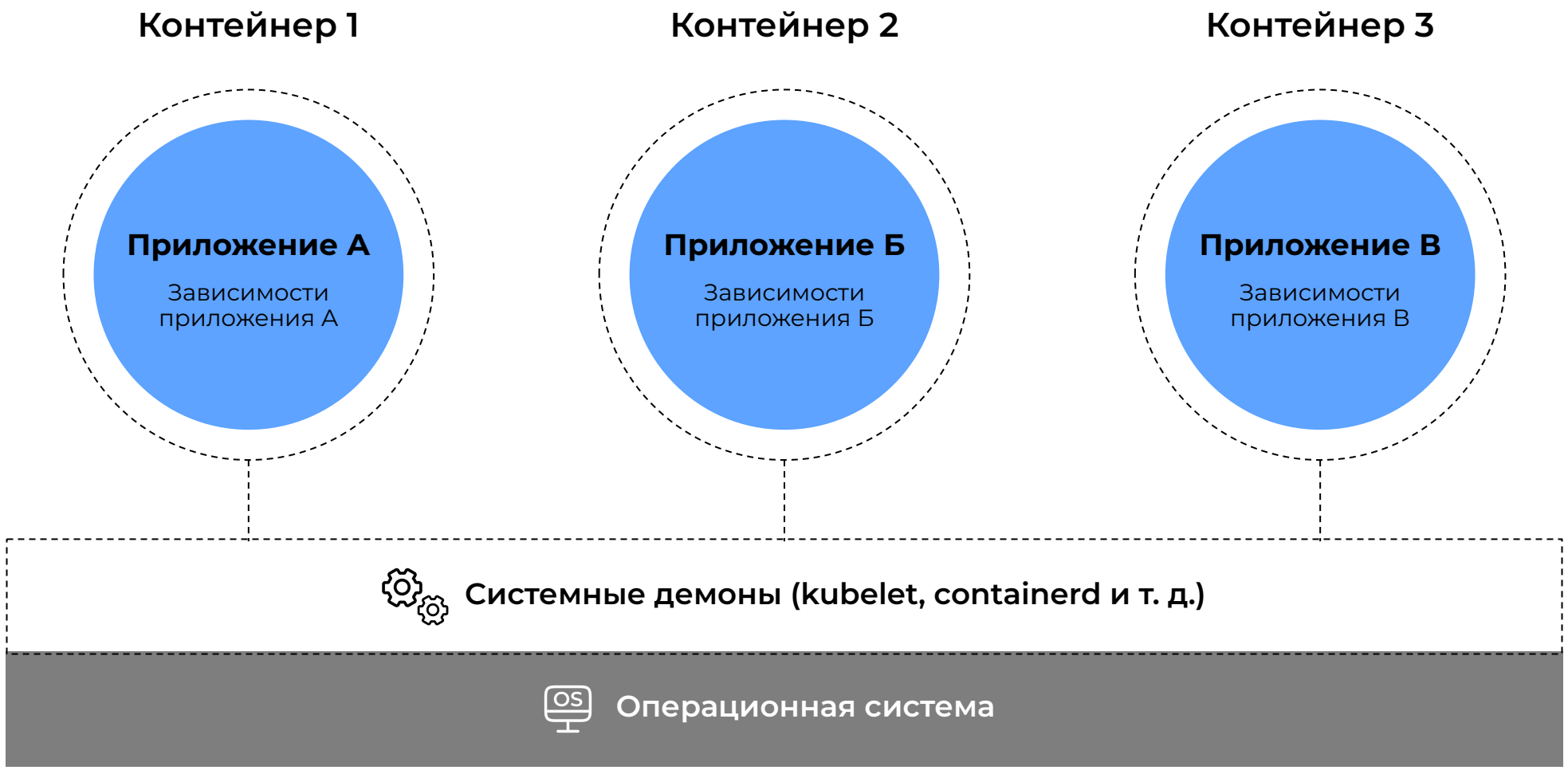


Исследование VMware The State of Kubernetes 2023

Схема кластера под управлением Deckhouse



Концептуальная схема узла кластера Kubernetes



На чём строим?

- Выбор между РЕД ОС, Astra Linux, ALT Linux (ставка Минцифры)
- До 2022 CentOS => выбрали РЕД ОС
- Глаза боятся, но руки делают. Всё решаемо
- Полноценное сообщество

Результаты по замене операционной системы

- Для контейнеризации перешли полностью на РЕД ОС
- Сейчас в процессе изучения перехода по базовому слою
- В Kubernetes благодаря контейнерам переезд получился быстрее
- Производим миграцию продуктивных сервисов

Нюансы, с которыми пришлось разбираться

В общем:

- Переход на Cilium
- Поддержка образов отечественными средствами защиты
- Проработка сценариев для Istio
- Переход на LINSTOR

Нюансы, с которыми пришлось разбираться

Для Deckhouse:

- Kaspersky KESL
- Secureboot и DRBD на РЕД ОС
- Falco и eBPF на РЕД ОС
- Отсутствие конвейера по сборке образов

Дорожная карта перехода/внедрения

- Изменения в архитектуре
- Изменения в приложениях
- Изменения в процессах и людях

Изменения в архитектуре

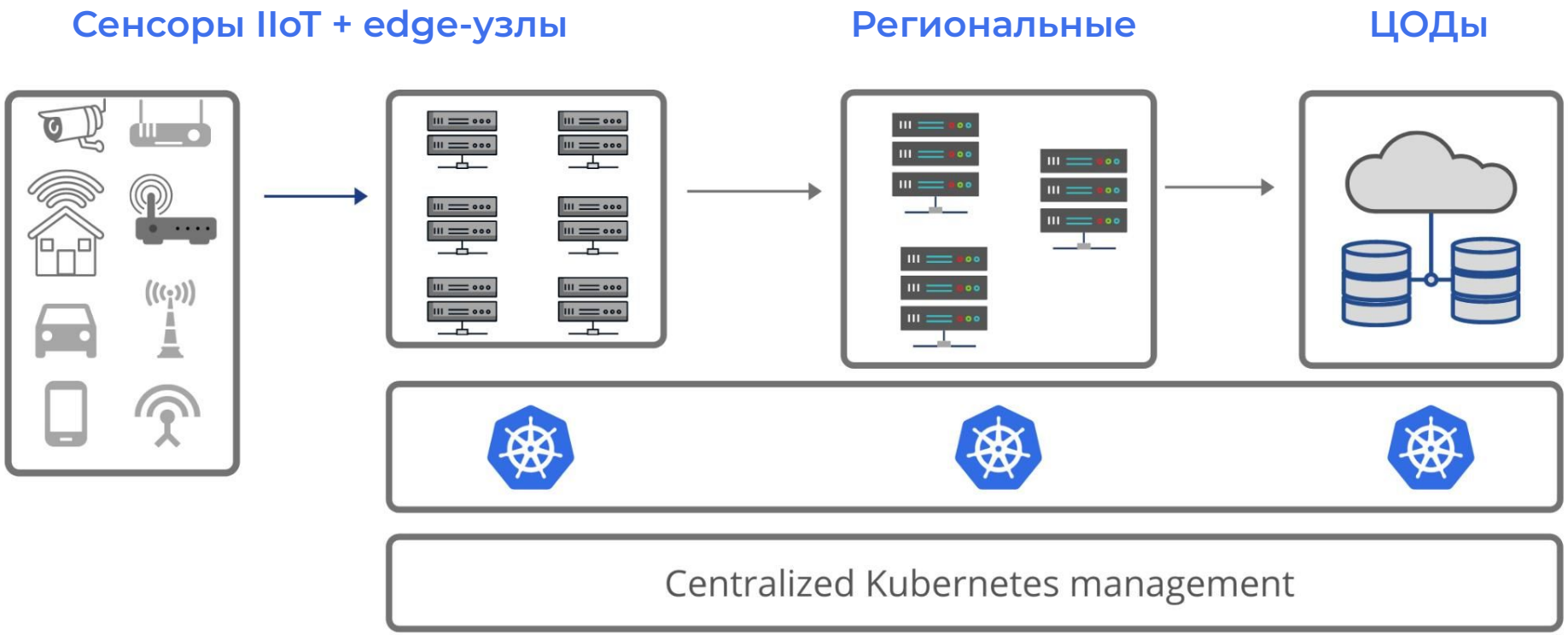
- Автоскейлинг
- Безопасность
- Сеть
- Интерфейсы управления
- Логирование
- Разработка
- Отказоустойчивость
- Интеграция
- Мониторинг
- Администрирование
- Балансировка
- Хранение
- Виртуализация
- **Kubernetes**

Инфраструктура

- Private Cloud
- Bare metal
- IaaS

Общая концепция архитектуры

Отказ от больших «коммуналок» и региональные кластеры



Изменения в приложениях

Требуется изменение в манифестах приложений в случае использования специфичных для платформы примитивов

OpenShift	Kubernetes
Template	Отказываемся в пользу Helm chart
DeploymentConfig	Меняем на Deployment (не забывая об особенностях, связанных с ImageStream и BuildConfig)
Route	Меняем на Ingress
Deployment/Statefulset/Daemonset	Не требуют изменений (только замена параметризации)
Service/ConfigMap и т. д.	Не требуют изменений (только замена параметризации)

Изменения в процессах: shared library развиваем как inner-source

Общие библиотеки поддерживают единый подход к передаче артефактов сборки в репозитории Nexus для релиз-инженеров и архитекторов.

Функция	Назначение
login	Авторизация в реестр
build	Создание образа
push	Загрузка образа в реестр
logout	Выход из учётной записи (сброс кэша)
full	Полный набор команд для загрузки одного образа

Общая библиотека в разы снижает нагрузку для написания helm-чартов на разработку.

Плюсы от перехода

Модули — это часть платформы

- Модуль политик контроля доступа включен по умолчанию
- Реализованы политики Privileged, Baseline и Restricted из Pod Security Standards
- Чтобы начать пользоваться, нужно просто назначить label на namespace

```
$ kubectl label ns my-namespace security.deckhouse.io/pod-policy=restricted
```

Плюсы от перехода

Простота конфигурации, NoOps-платформа

- Все модули преднастроены и у платформы свой удобный API

```yaml
apiVersion: deckhouse.io/v1alpha1
kind: OperationPolicy
metadata:
  name: common
spec:
  policies:
    allowedRepos:
      - myregistry.example.com
    requiredResources:
      limits:
        - memory
      requests:
        - cpu
        - memory
  match:
    namespaceSelector:
      labelSelector:
        matchLabels:
          operation-policy.deckhouse.io/enabled: "true"
```

Плюсы от перехода

По-настоящему расширенная поддержка

- Быстрое привлечение команды поддержки не для решения аварии,
  а для прохождения внеплановых испытаний в нерабочее время.

- Возможность оперативно получить необходимый функционал через feature request.
  Например, реализована поддержка мутаций в модуле политик контроля доступа.

  github.com/deckhouse/deckhouse/issues/3731

Доработки Deckhouse при миграции: мультитенантность

- Модуль multitenancy-manager позволяет настраивать изолированные окружения в кластере Kubernetes. Повторяем пользовательский опыт из OpenShift

- По подготовленному шаблону с помощью Custom Resource Project в кластере Kubernetes можно получить одинаковые, изолированные друг от друга окружения

```
apiVersion: deckhouse.io/v1alpha1
kind: ProjectType
metadata:
  name: test-project-type
spec:
  resourcesTemplate: |
    ---
    apiVersion: v1
    kind: ResourceQuota
    metadata:
      name: all-pods
    spec:
      hard:
        {{ with .params.requests.cpu }}requests.cpu: {{ .
}}{{ end }}
        {{ with .params.requests.memory }}requests.memory:
{{ . }}{{ end }}
```

```
apiVersion: deckhouse.io/v1alpha1
kind: Project
metadata:
  name: test-project
spec:
  description: Test project
  projectTypeName: test-project-type
  template:
    requests:
      cpu: 5
      memory: 5Gi
      storage: 1Gi
    limits:
      cpu: 5
      memory: 5Gi
```

Доработки Deckhouse при миграции: внутренний compliance

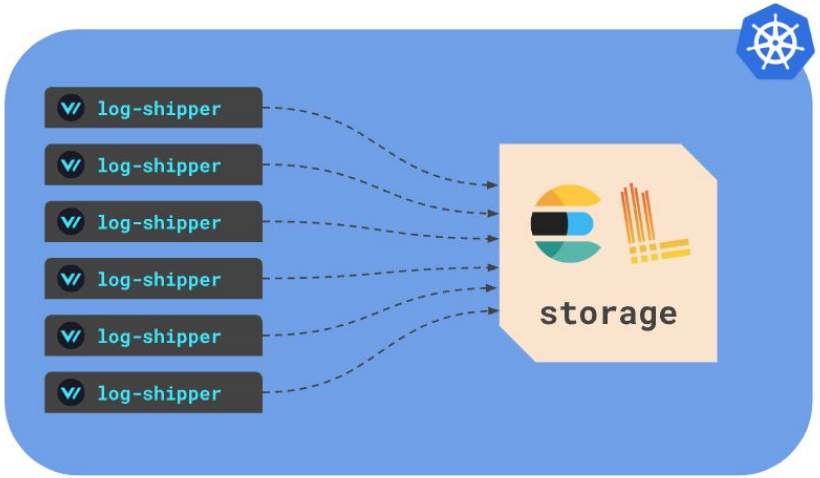
- Расширенные политики безопасности

- Соответствие внутренним каталогам требований и benchmark’ам по ИБ

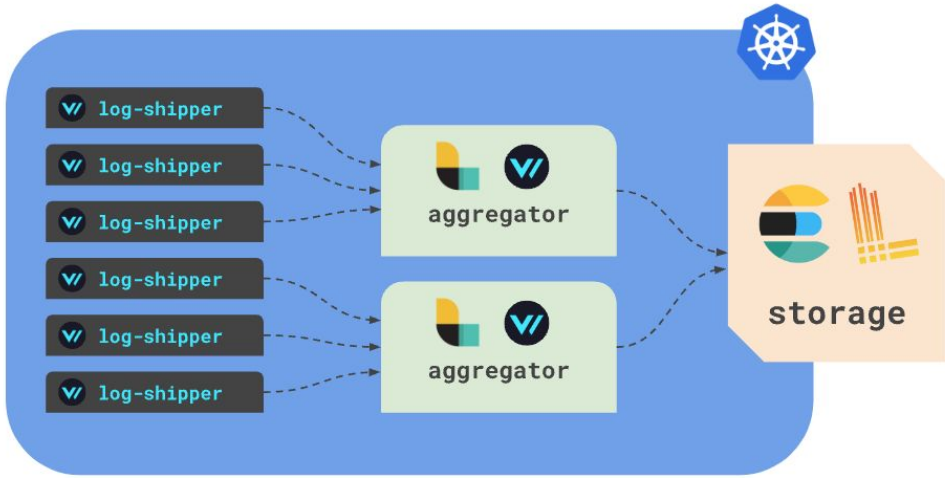
```yaml
apiVersion: deckhouse.io/v1alpha1
kind: SecurityPolicy
metadata:
  name: mypolicy
spec:
  enforcementAction: Deny
  policies:
    runAsUser:
      ranges:
      - max: 200
        min: 100
      rule: MustRunAs
  match:
    namespaceSelector:
      labelSelector:
        matchLabels:
          enforce: mypolicy
```

Доработки Deckhouse при миграции: отправка логов в Apache Kafka

Распределённая

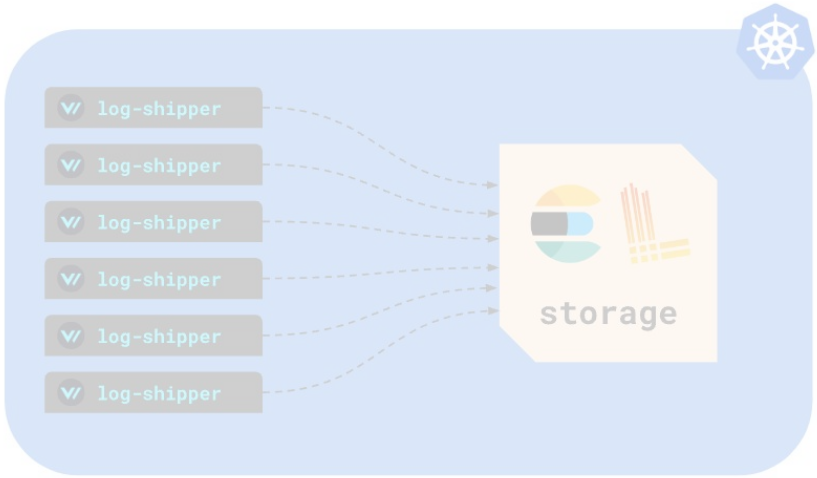


Централизованная

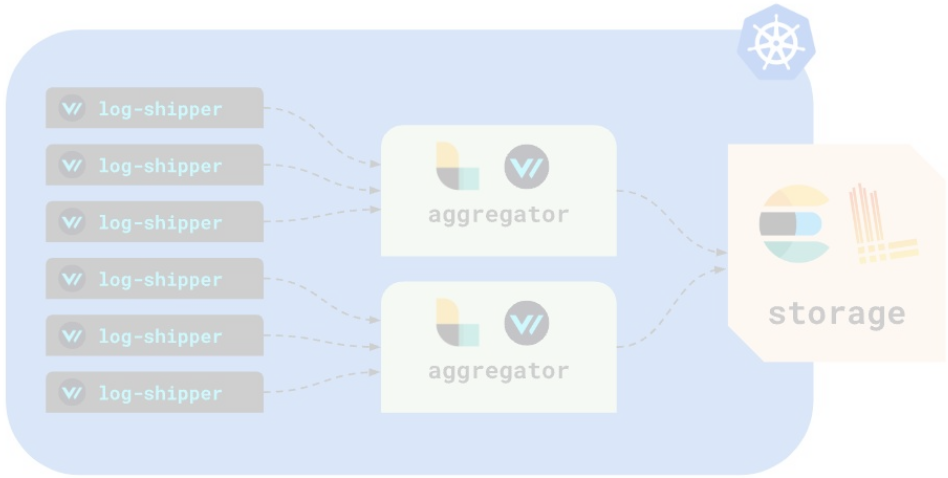


Доработки Deckhouse при миграции: отправка логов в Apache Kafka

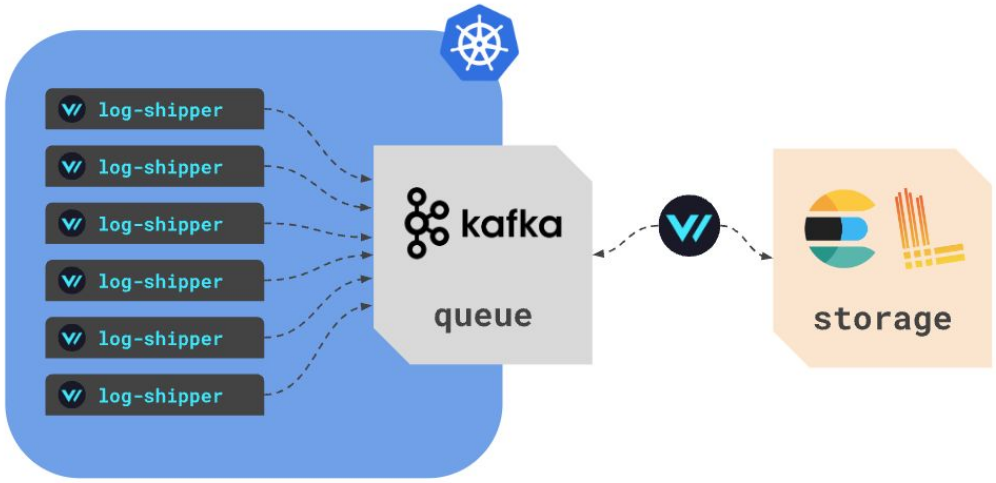
Распределённая



Централизованная



Потоковая



Открытый RoadMap

Публичная разработка
на Github, открытые планы
по всем направлениям
развития продукта: UI, ФСТЭК,
Deckhouse Commander и т. д.

Возможность влиять на RoadMap

Адаптированный под наши задачи RICE

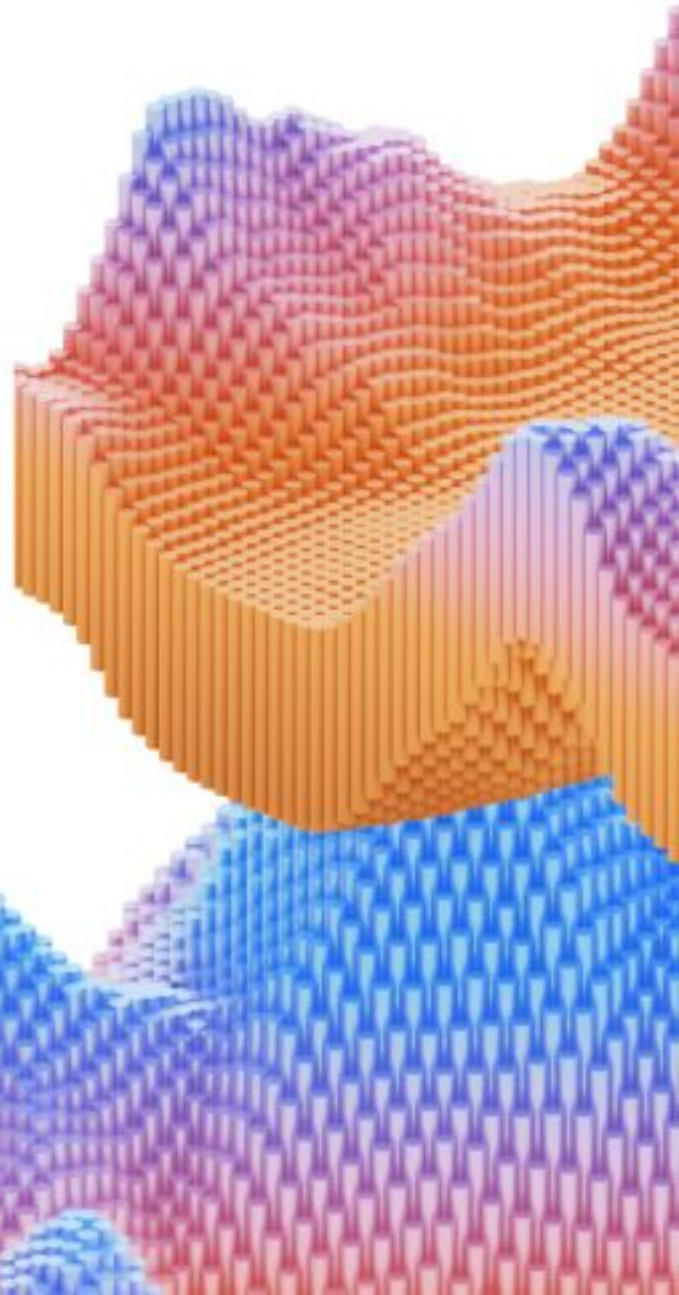
$$\frac{\textbf{R}\text{each} \times \textbf{I}\text{mpact} \times \textbf{C}\text{onfidence}}{\textbf{E}\text{ffort}} = \textbf{RICE}\ \text{Score}$$

Примеры использования RICE

- Отправка логов через Apache Kafka реализована в первую очередь, потому что Effort был минимальный
- После подтверждения важности ещё от нескольких клиентов (увеличение Reach и Confidence) взяли в работу и выпустили в тестовом режиме поддержку проектов в Deckhouse

Итоги

1. Импортонезависимость и удовлетворение требований
2. Снятие юридических рисков
3. Существенное повышение экспертизы собственной команды
4. Закрытие потребностей отраслевого сообщества через развитие вендорских решений



Оставьте ваш отзыв

Кирилл Носков
Руководитель центра цифровых продуктов и платформ
«Газпромнефть ИТО»

Константин Аксёнов
Руководитель разработки Deckhouse
«Флант»