Alfa Digital

Жизнь Java-приложений в облачных контейнерах.

Или рождённый на серверах летать ~~не~~ может.

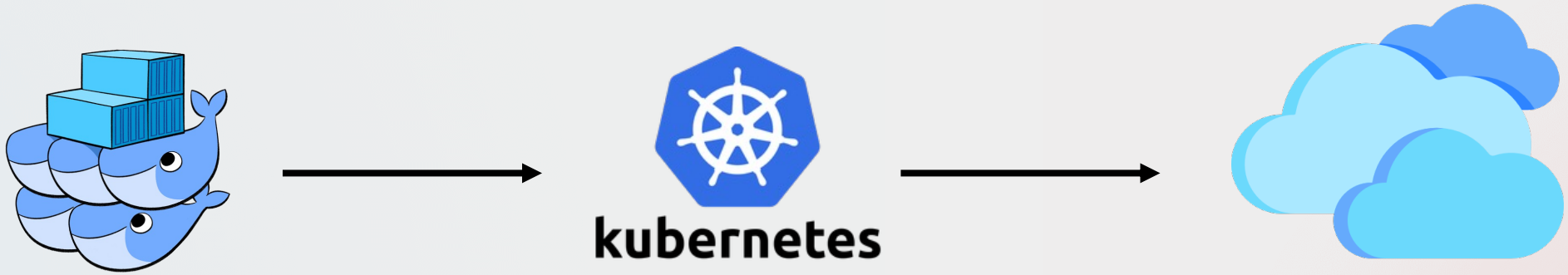
О чём сегодня поговорим

1. Как мы перестроили работу java приложения для Kubernetes

2. Пришли к cloud native подходу

3. С какими вызовами столкнулись когда начали переезжать в облако

Немного истории



Kubernetes

Возможность гибко управлять контейнерами

Возможность масштабироваться под нагрузку

Возможность быстро запускать упавшие контейнеры

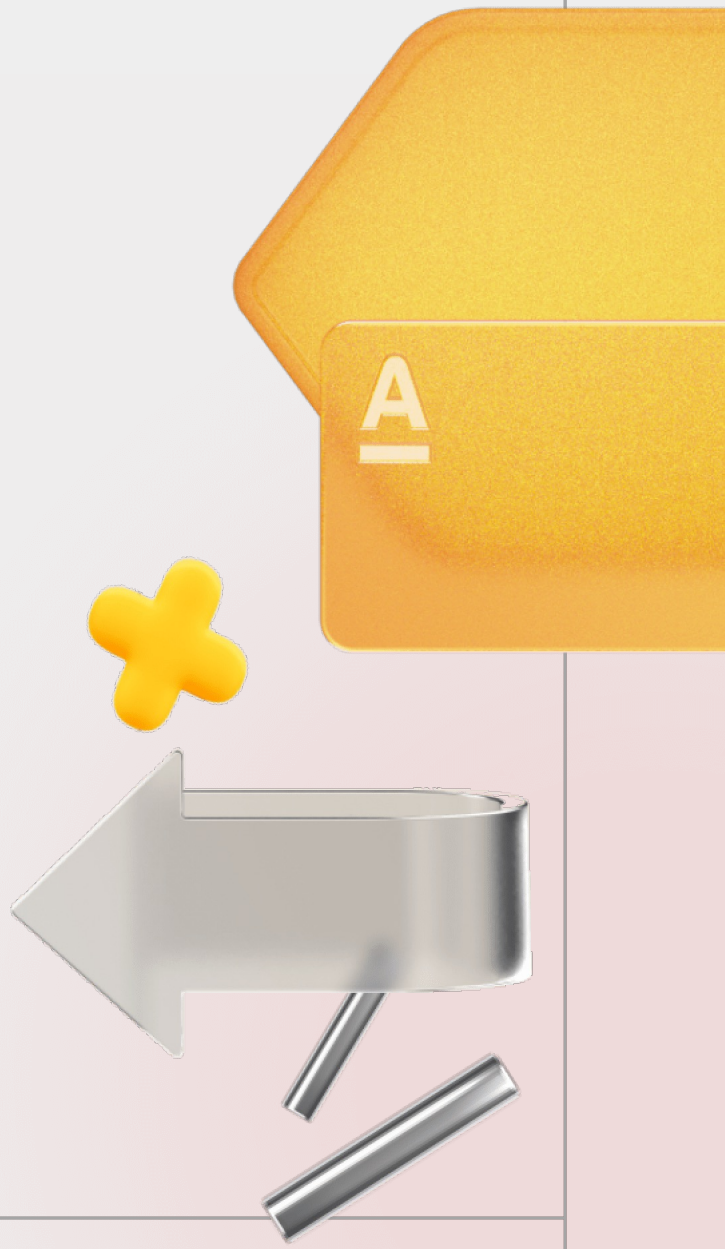
Инструменты работы со средой

О чём сегодня поговорим

1. Скорость работы приложений, их деплоя и поднятия

2. Использование инструментов Kubernetes для работы java приложений

3. Уезжаем в облако, и вызовы, которые оно нам приготовило

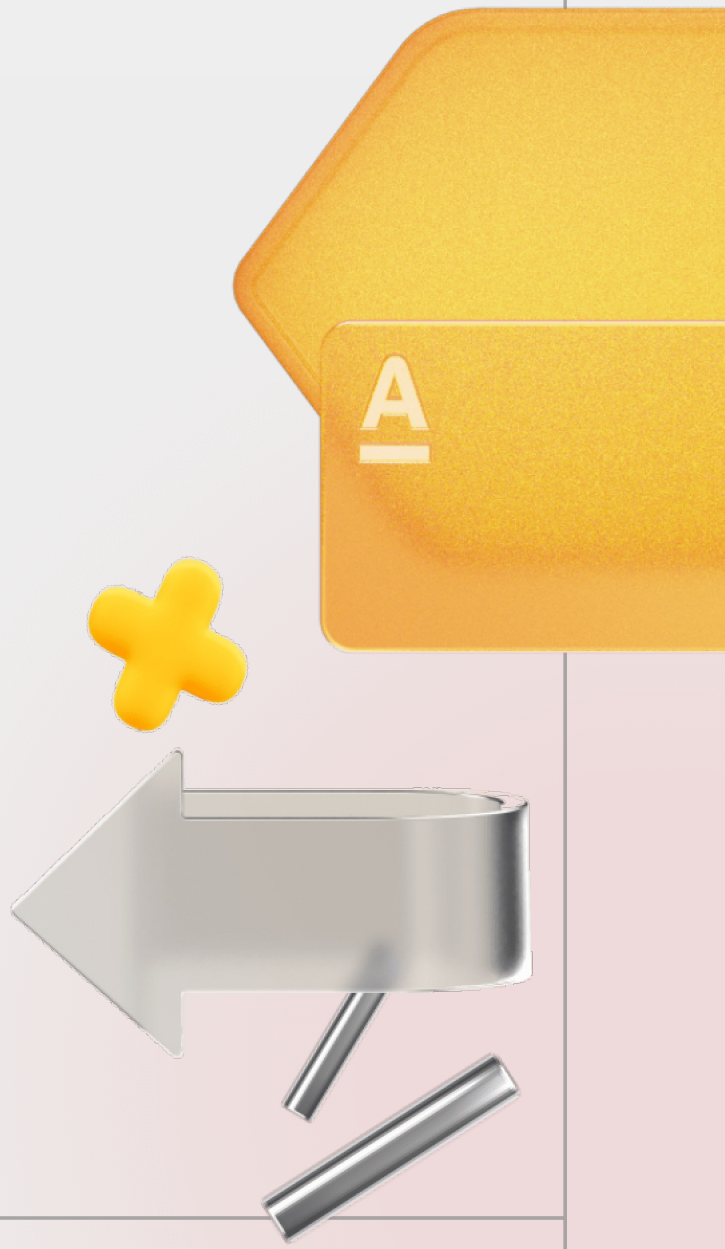


Мы тут

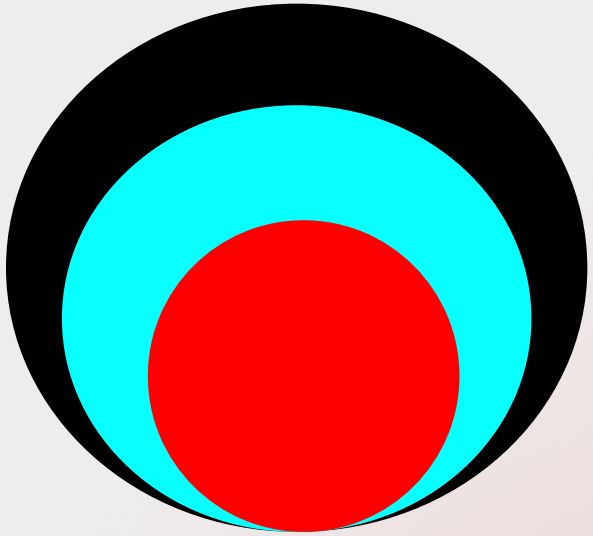
1. Скорость работы приложений, их деплоя и поднятия

2. Использование инструментов Kubernetes для работы java приложений

3. Уезжаем в облако, и вызовы, которые оно нам приготовило

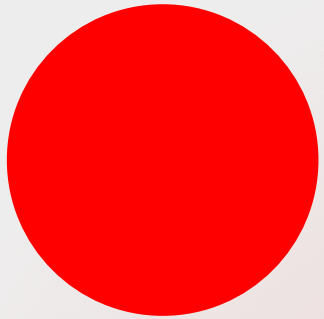


Многослойный build



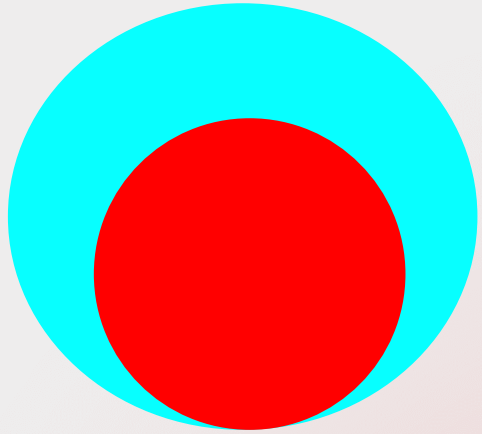
Многослойный build

1. Alpine image



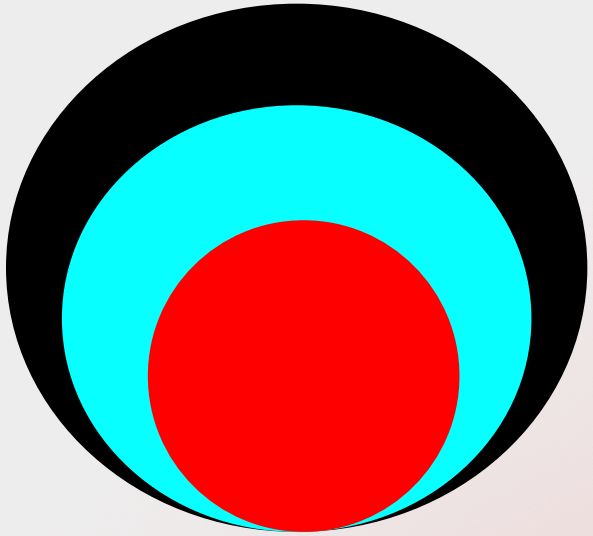
Многослойный build

1. Alpine image
2. Добавляем зависимости и внутренние библиотеки



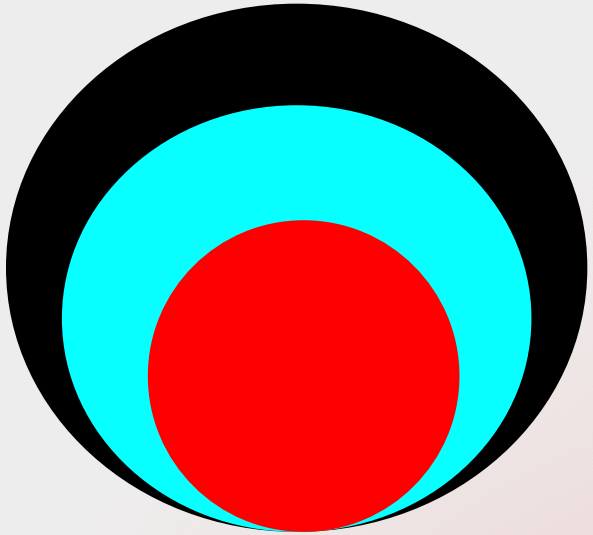
Многослойный build

1. Alpine image
2. Добавляем зависимости и внутренние библиотеки
3. Добавляем код нашего сервиса



Многослойный build

1. Alpine image
2. Добавляем зависимости и внутренние библиотеки
3. Добавляем код нашего сервиса



```
ARG DOCKER_INTERNAL_REPOSITORY
FROM ${DOCKER_INTERNAL_REPOSITORY}/alfabank/infra/openjdk:15

COPY dist/unwrap/BOOT-INF/lib /app/lib
COPY dist/unwrap/META-INF /app/META-INF
COPY dist/unwrap/BOOT-INF/classes /app/classes
```

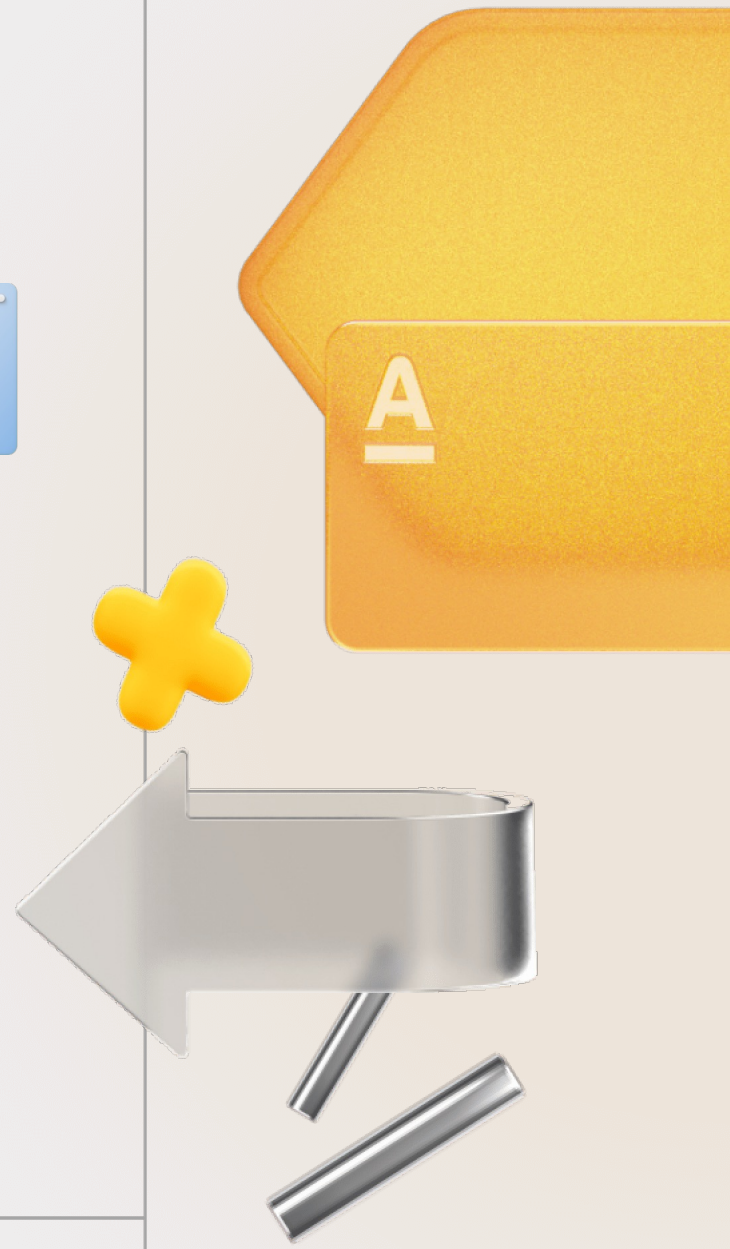
Старт приложения

Старт приложения

Старт приложения
java -jar hello_world.jar

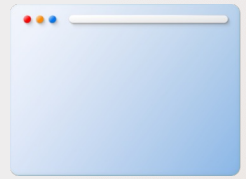
Старт приложения

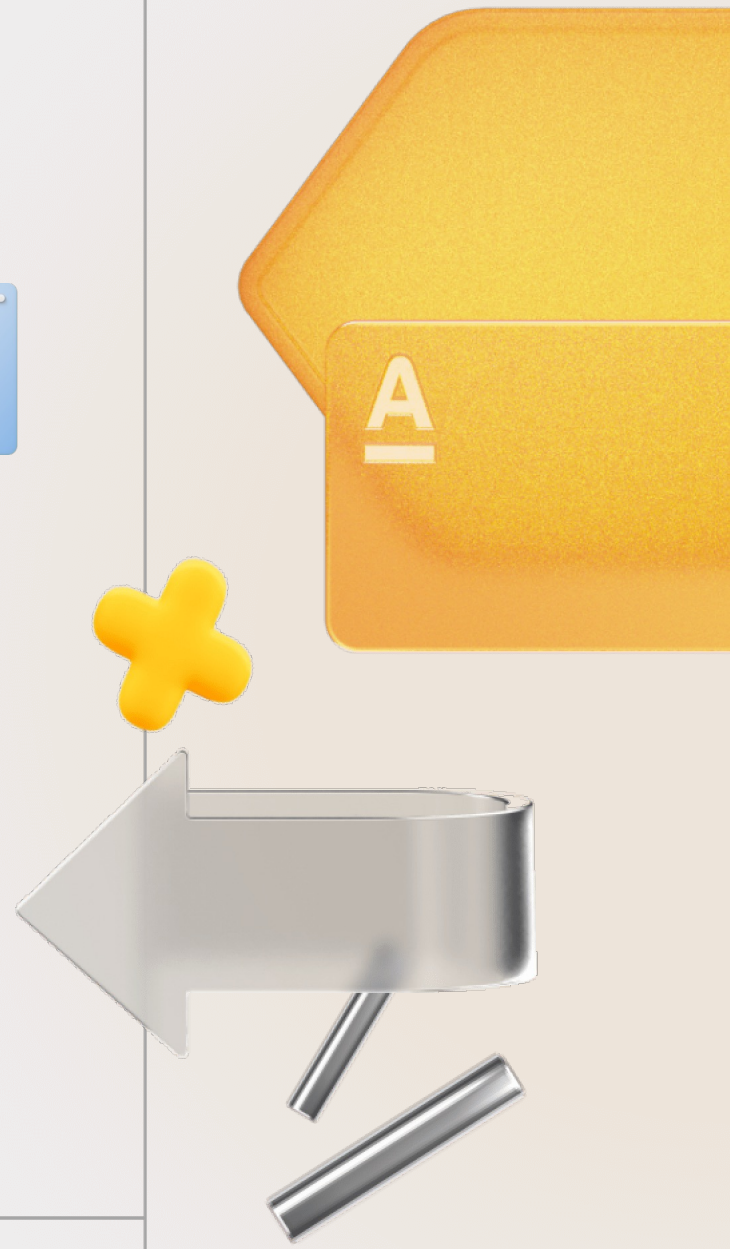
Старт приложения
java -jar hello_world.jar



Старт приложения

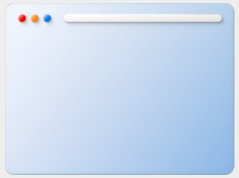
Старт приложения
java -jar hello_world.jar





Старт приложения

Старт приложения
java -jar hello_world.jar



```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-antrun-plugin</artifactId>
    <version>${maven.antrun.version}</version>
    <executions>
        <execution>
            <phase>package</phase>
            <configuration>
                <target>
                    <copy file="target/${project.artifactId}-${project.version}.jar"
                          tofile="./dist/app.jar"/>
                    <copy todir="./dist/unwrap" failonerror="false">
                        <fileset dir="target/dependency"/>
                    </copy>
                    <copy todir="./dist/surefire-reports" failonerror="false">
                        <fileset dir="target/surefire-reports"/>
                    </copy>
                </target>
            </configuration>
            <goals>
                <goal>run</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

Время старта приложения

Время старта
~ 45 сек.

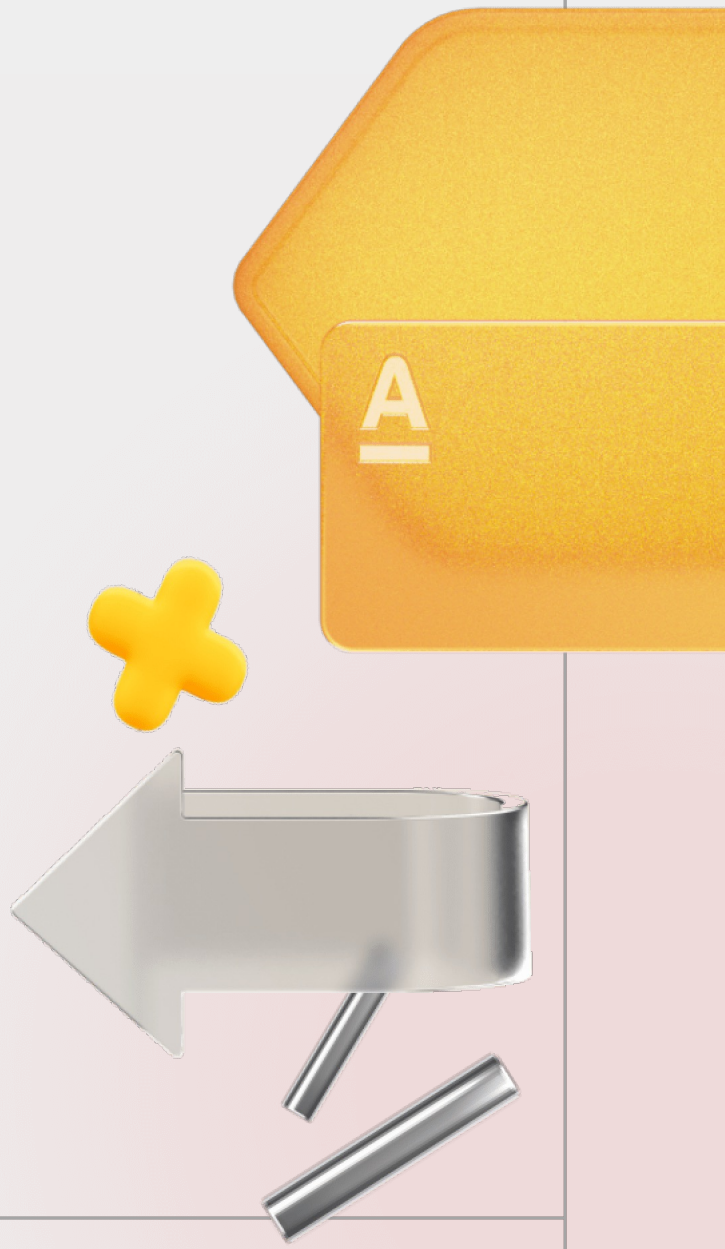
Время старта приложения

Время старта
~ 45 сек.

Время старта
~ 10 сек.

Java AppCDS

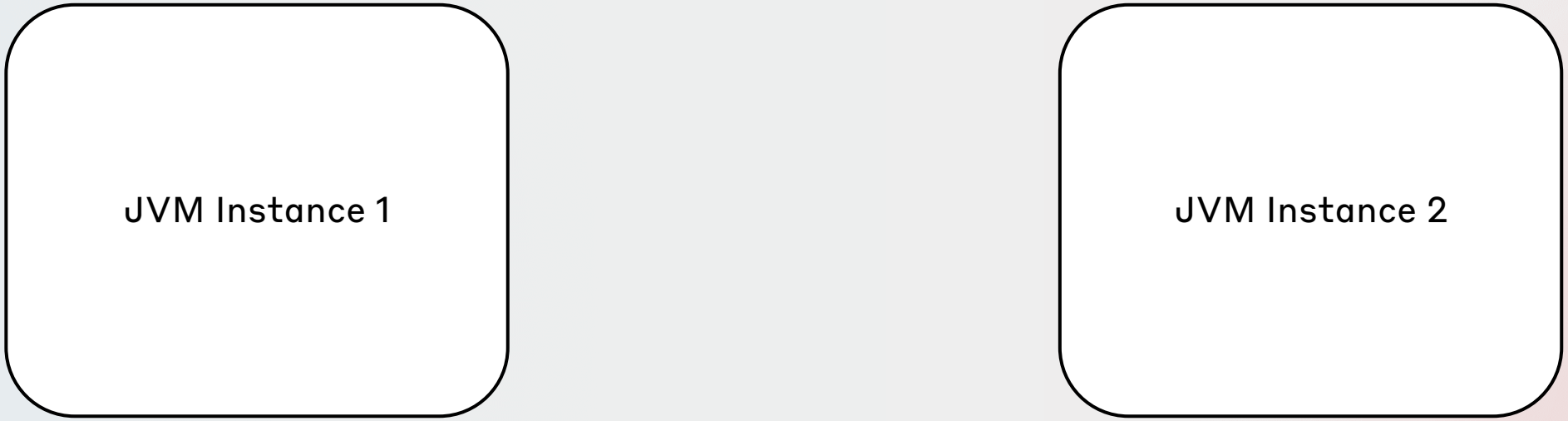
Application Class Data Sharing (AppCDS)



Спойлер: очень сложно, а профита мало

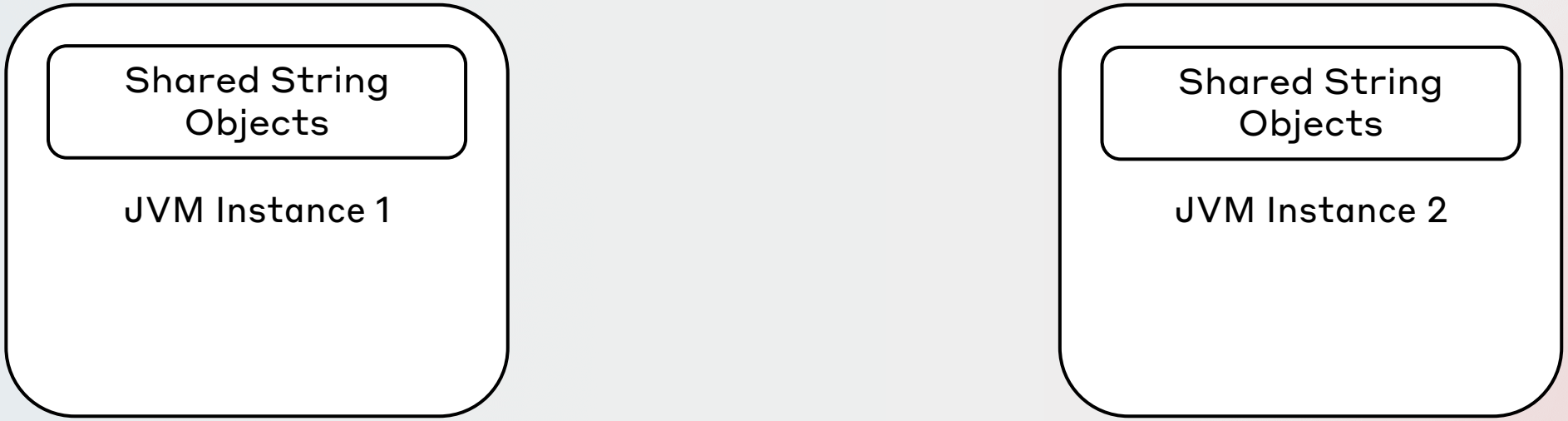
Java AppCDS

Application Class Data Sharing (AppCDS)



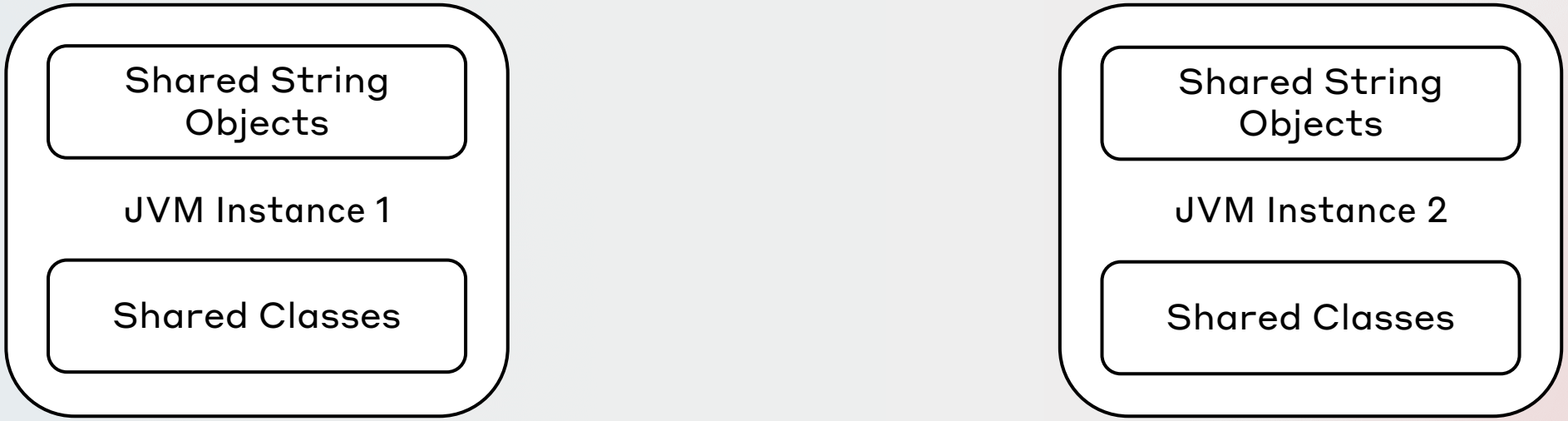
Java AppCDS

Application Class Data Sharing (AppCDS)



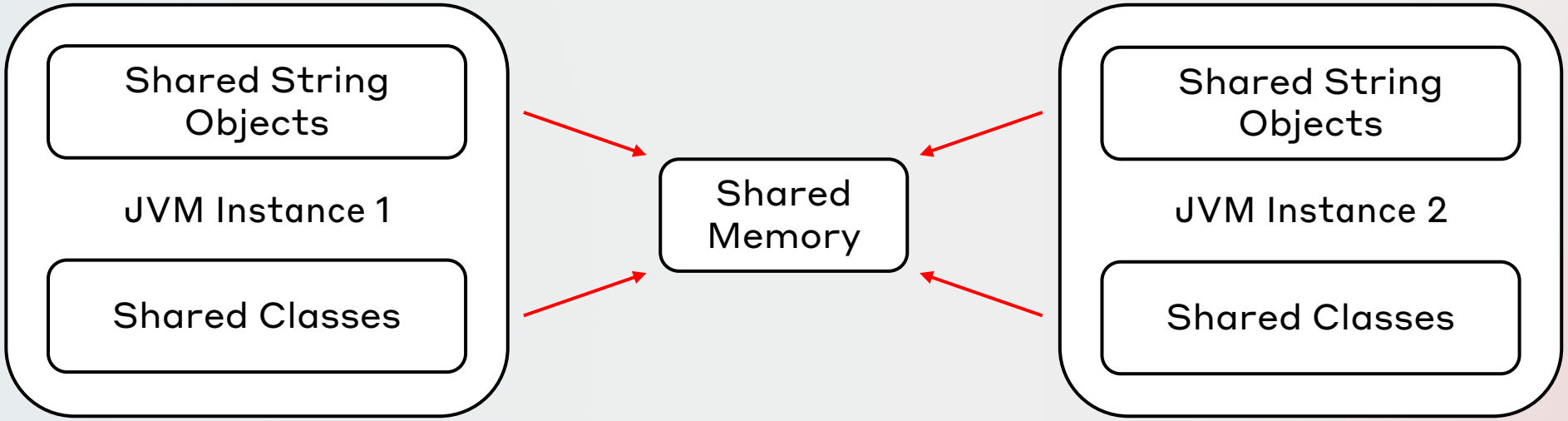
Java AppCDS

Application Class Data Sharing (AppCDS)



Java AppCDS

Application Class Data Sharing (AppCDS)



Java AppCDS

Java 11
Ускорение ~ 3 сек, а то и меньше

Java AppCDS

Java 11
Ускорение ~ 3 сек, а то и меньше

**Вывод: не всё помогает,
что написано в интернете**

Java AppCDS

Java 11
Ускорение ~ 3 сек, а то и меньше

Вывод: не всё помогает, что написано в интернете

Java 13 JEP-350 Dynamic CDS Archives

Мы тут

1. Скорость работы приложений, их деплоя и поднятия

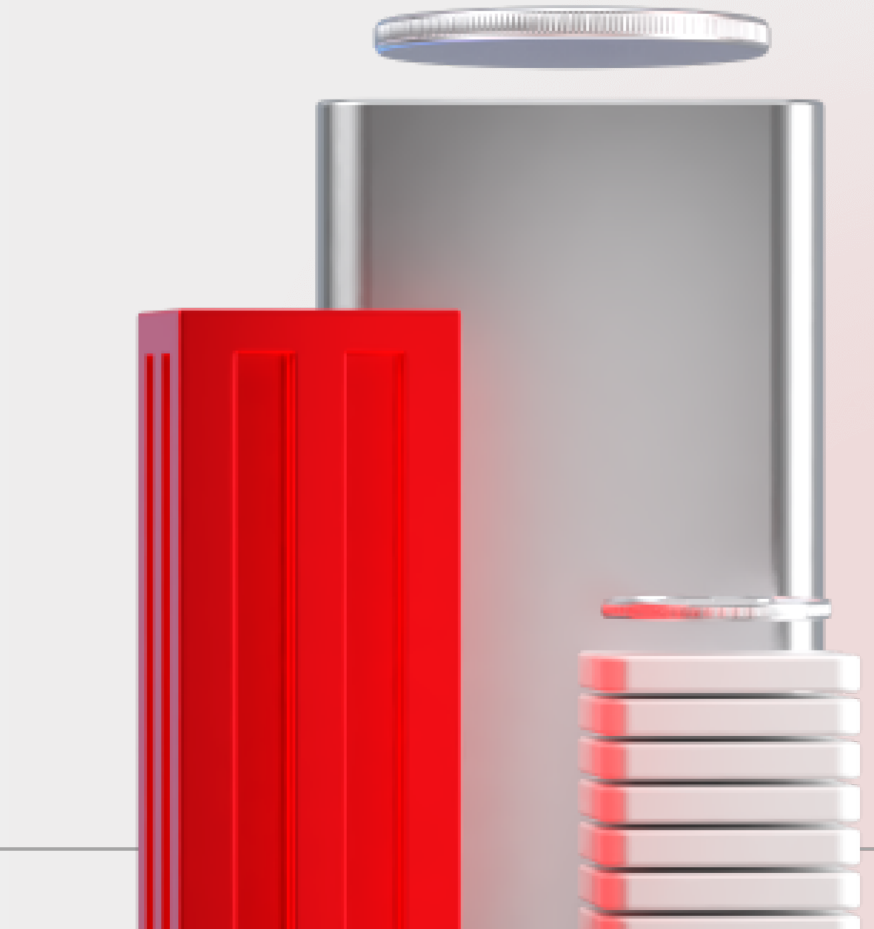
2. Использование инструментов Kubernetes для работы java приложений

3. Уезжаем в облако, и вызовы, которые оно нам приготовило

Как инициализируется приложение

1. Запустить Config Server (кеш)
2. Запустит Service Registry
3. Запустить само приложение
4. Запросить конфиги для инициализации приложения у Config Server
5. Зарегистрировать приложение в Service Registry
6. Запросить адреса сервисов и служб у Service Registry

Конфиги

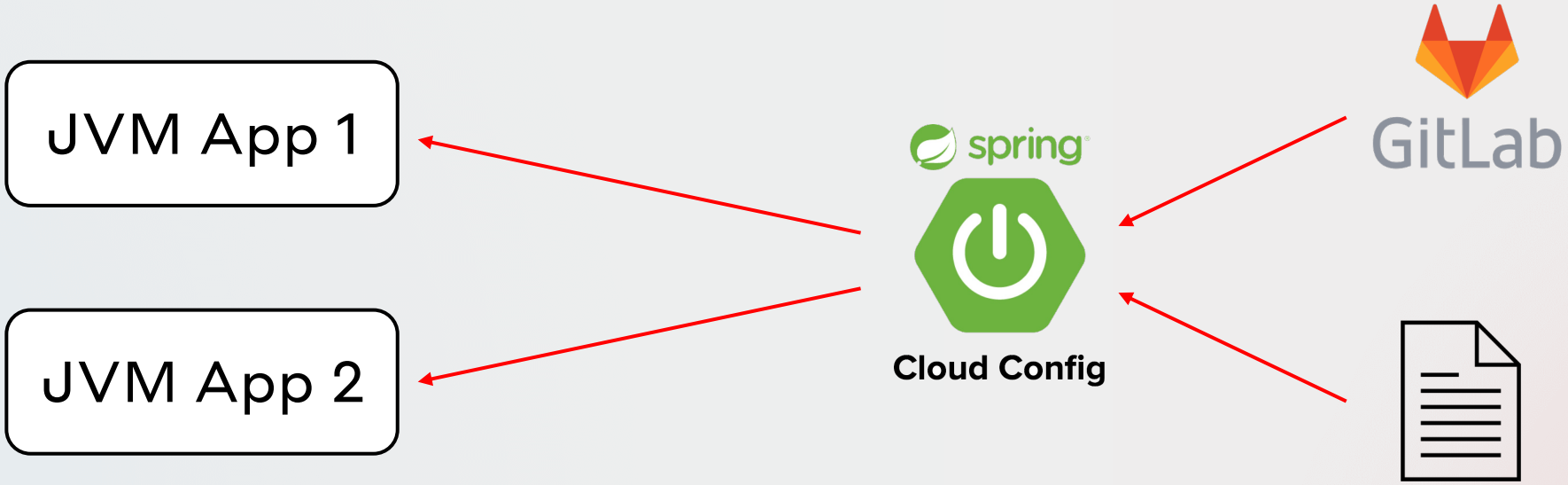


Конфиги

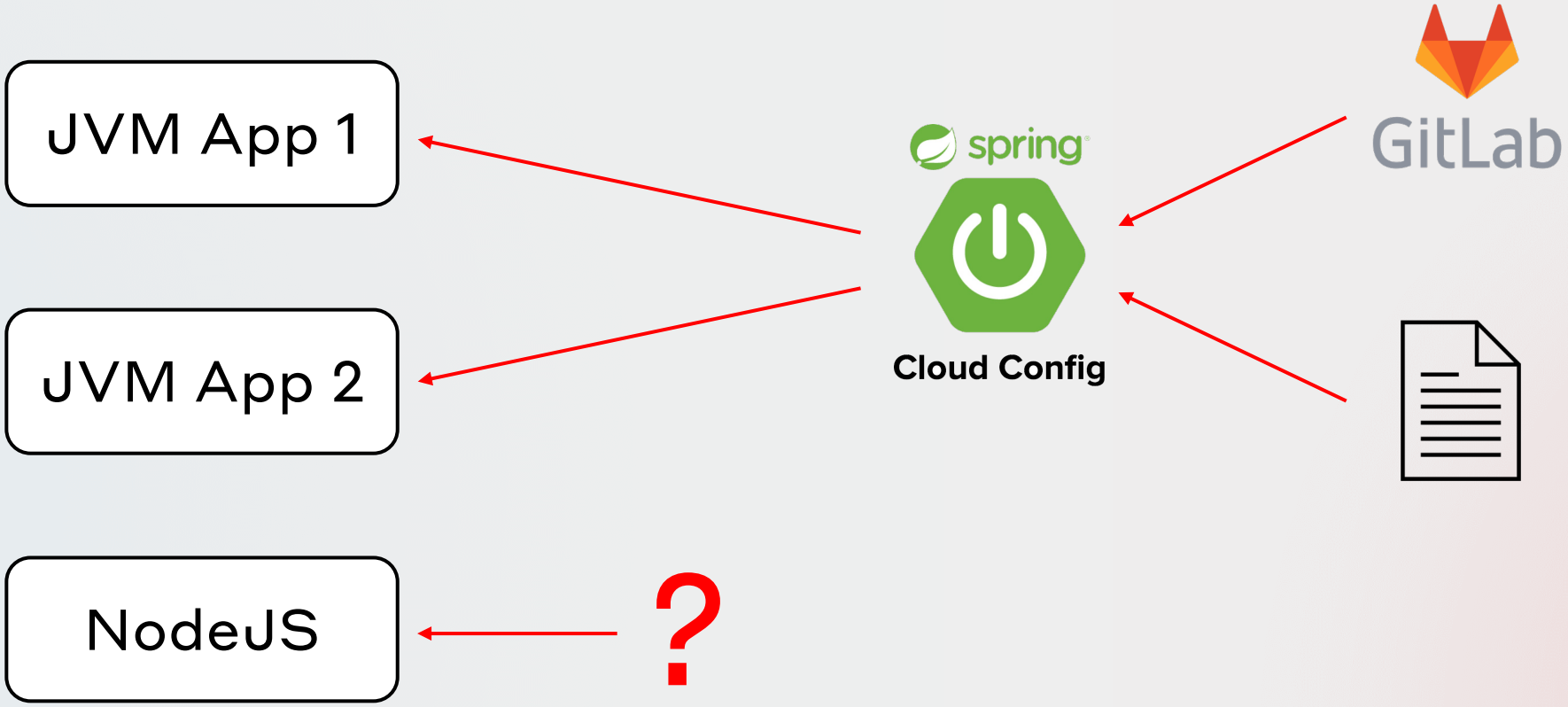
JVM App 1

JVM App 2

Spring Cloud Config Server



Spring Cloud Config Server



The Twelve-Factor App

III. CONFIG

Окружение хранит конфигурацию и передаёт приложению

The Twelve-Factor App

III. CONFIG

Окружение хранит конфигурацию и передаёт приложению

1. Идентификаторы подключения к ресурсам типа базы данных, кэш-памяти и другим сторонним службам

The Twelve-Factor App

III. CONFIG

Окружение хранит конфигурацию и передаёт приложению

1. Идентификаторы подключения к ресурсам типа базы данных, кэш-памяти и другим сторонним службам

2. Регистрационные данные для подключения к внешним сервисам

The Twelve-Factor App

III. CONFIG

Окружение хранит конфигурацию и передаёт приложению

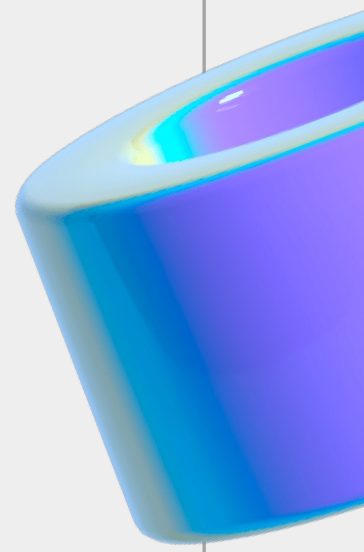
1. Идентификаторы подключения к ресурсам типа базы данных, кэш-памяти и другим сторонним службам

2. Регистрационные данные для подключения к внешним сервисам

3. Значения зависимые от среды развёртывания такие, как имя хоста

Kubernetes ConfigMaps

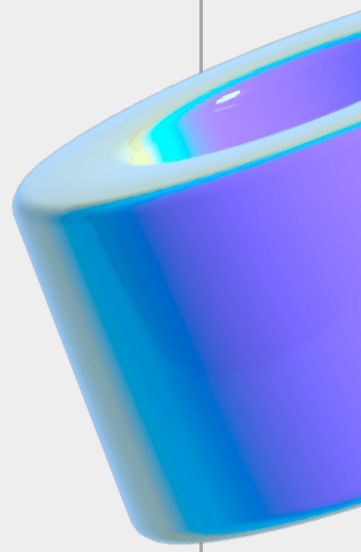
Место для хранения env переменных



Kubernetes ConfigMaps

Место для хранения env переменных

В каждом контейнере можно получить
к ним доступ



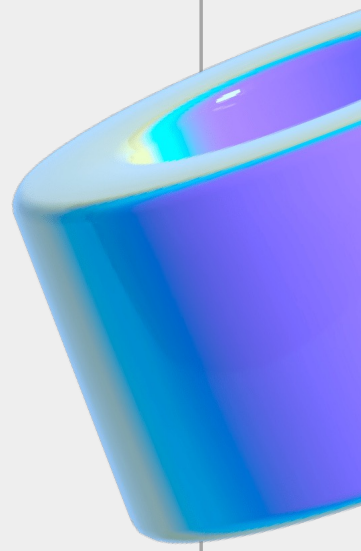
Kubernetes ConfigMaps

Место для хранения env переменных

В каждом контейнере можно получить
к ним доступ

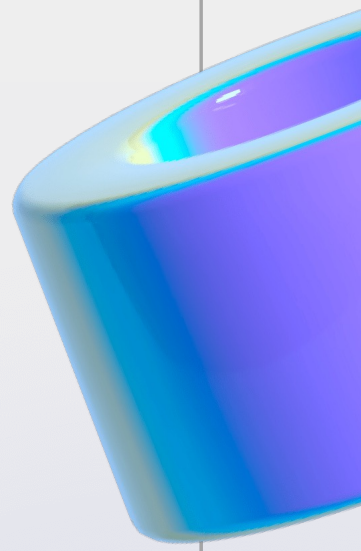
Два подхода:

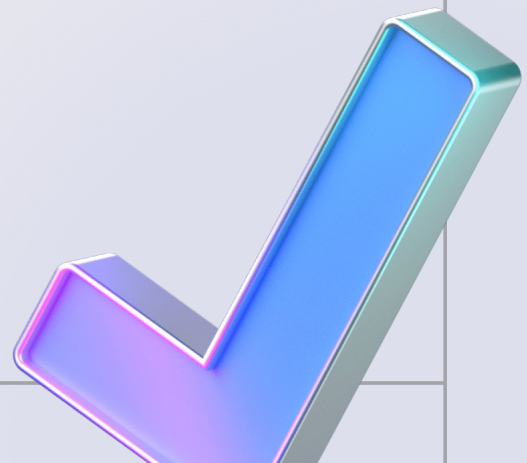
- Хранить много ConfigMaps под каждый сервис
- Хранить одну ConfigMaps под все сервисы



Kubernetes ConfigMaps

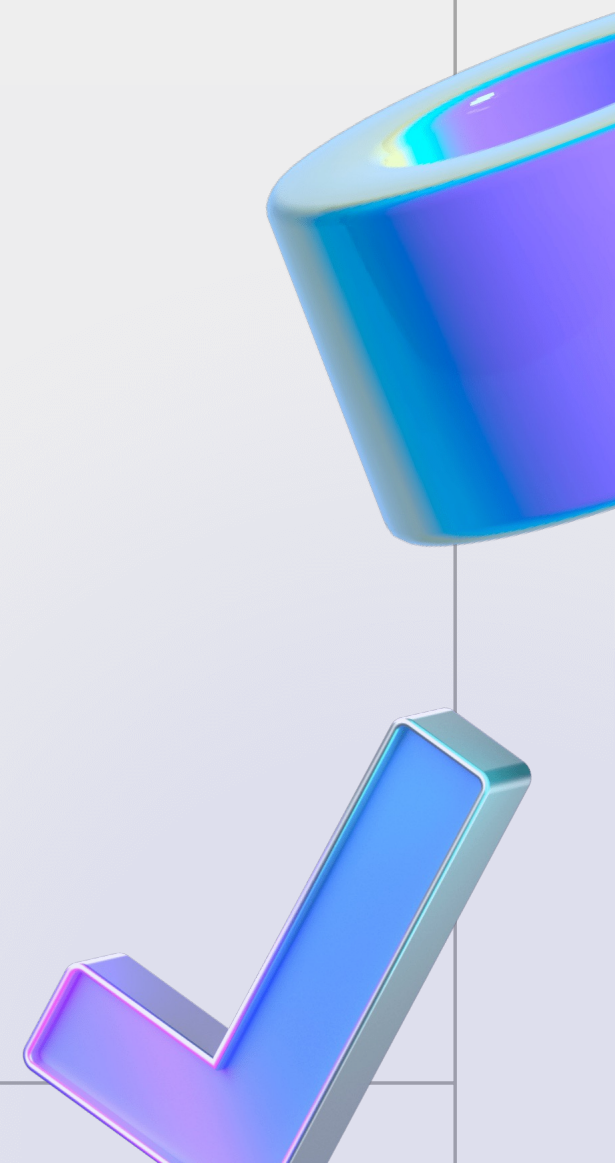
```yaml
nameOverride: ""
fullnameOverride: "common-config-preprod"

imageCredentials:
  - registry: alfab[redacted]
    username: tech-abru-ci
    password: AKCp5btg[redacted]
    # email:

configMaps:
  common-node:
    NODE_HOST: "0.0.0.0"
  common-keycloak:
    KEYCLOAK_REALM_URL: "https://keycl[redacted]
    KEYCLOAK_CLIENT_SSL_ALLOW_ANY_HOSTNAME: "false"
    KEYCLOAK_CLIENT_SSL_DISABLE_TRUST_MANAGER: "true"
    KEYCLOAK_BASE_PATH: "https://key[redacted]
    KEYCLOAK_URL: "http://ke[redacted]
    KEYCLOAK_REALM: "local_users"
    KEYCLOAK_INSECURE: "true"
  common-logging:
    LOGGING_LEVEL_ROOT: "INFO"
  common-postgres:
    PGHOST: "c-c9q4j[redacted]
    PGPORT: "6432"
    PGPORT12: "6432"
    PGADDITIONALPARAMS: "?prepareThreshold=0"
    PGMAXCONNECTIONS: "5"
    SPRING_PG_URL: "jdbc:postgresql://c-c9q4[redacted]

    YC_MANAGED_DB: "yes"
```





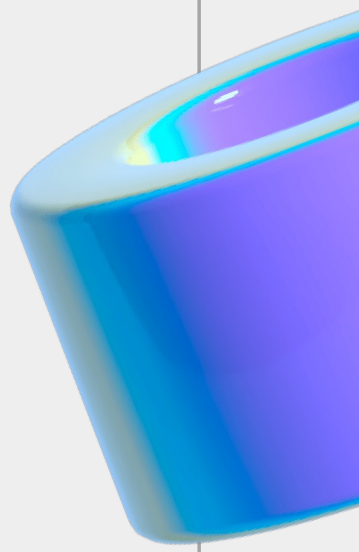
Kubernetes ConfigMaps

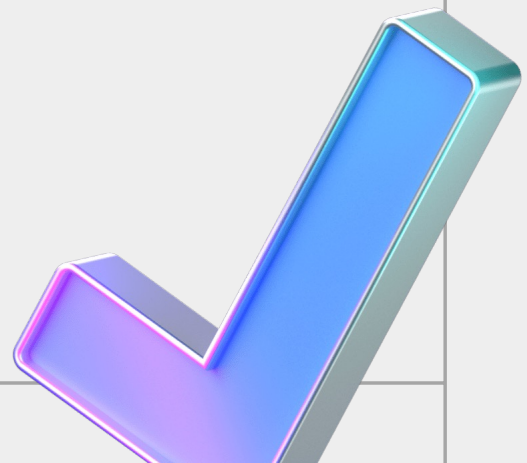
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: configmap-demo-pod
spec:
  containers:
    - name: demo
      image: alpine
      command: ["sleep", "3600"]
      env:
        # Define the environment variable
        - name: TEST_PARAMS # Notice that the case is different here
                            # from the key name in the ConfigMap.
```



Kubernetes Secrets

Разграничение прав RBAC

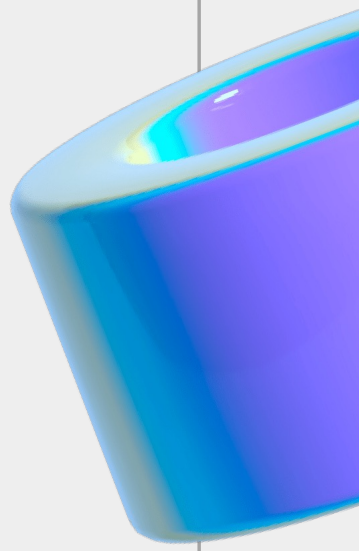


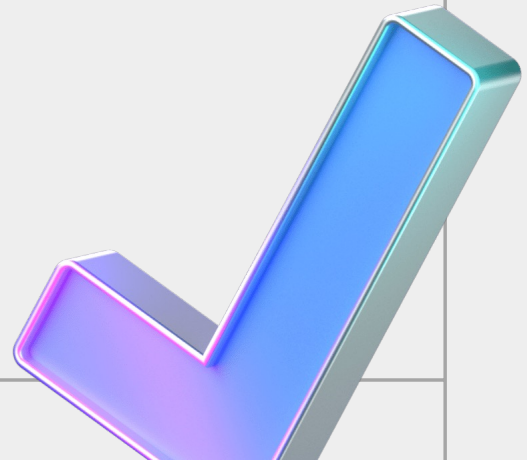


Kubernetes Secrets

Разграничение прав RBAC

Доступ к ним можно получить через env



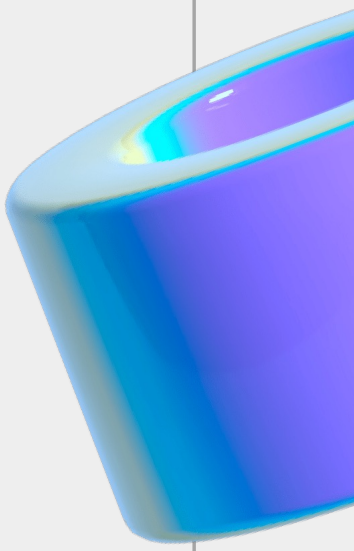


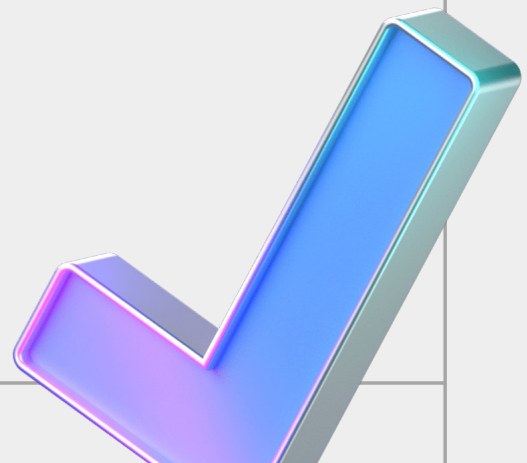
Kubernetes Secrets

Разграничение прав RBAC

Доступ к ним можно получить через env

Хранятся в базе Kubernetes в зашифрованном виде

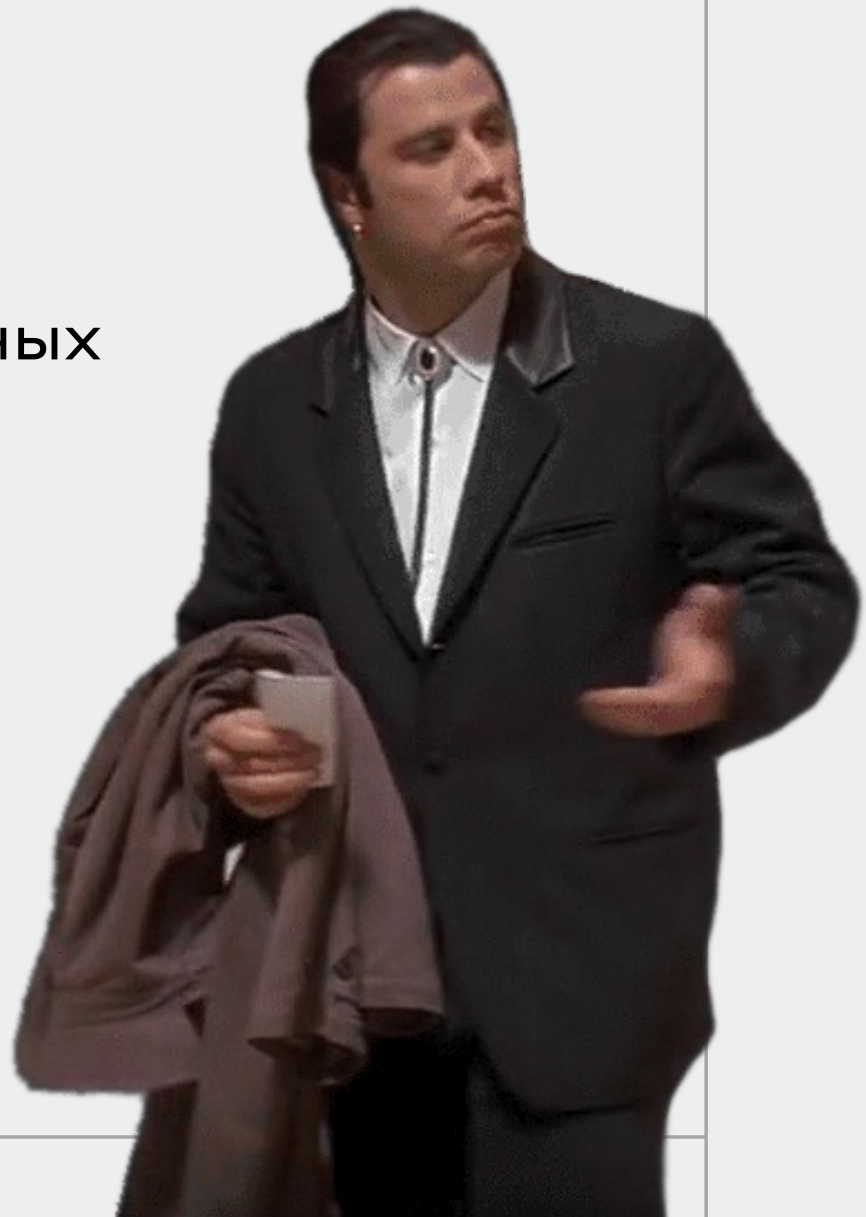




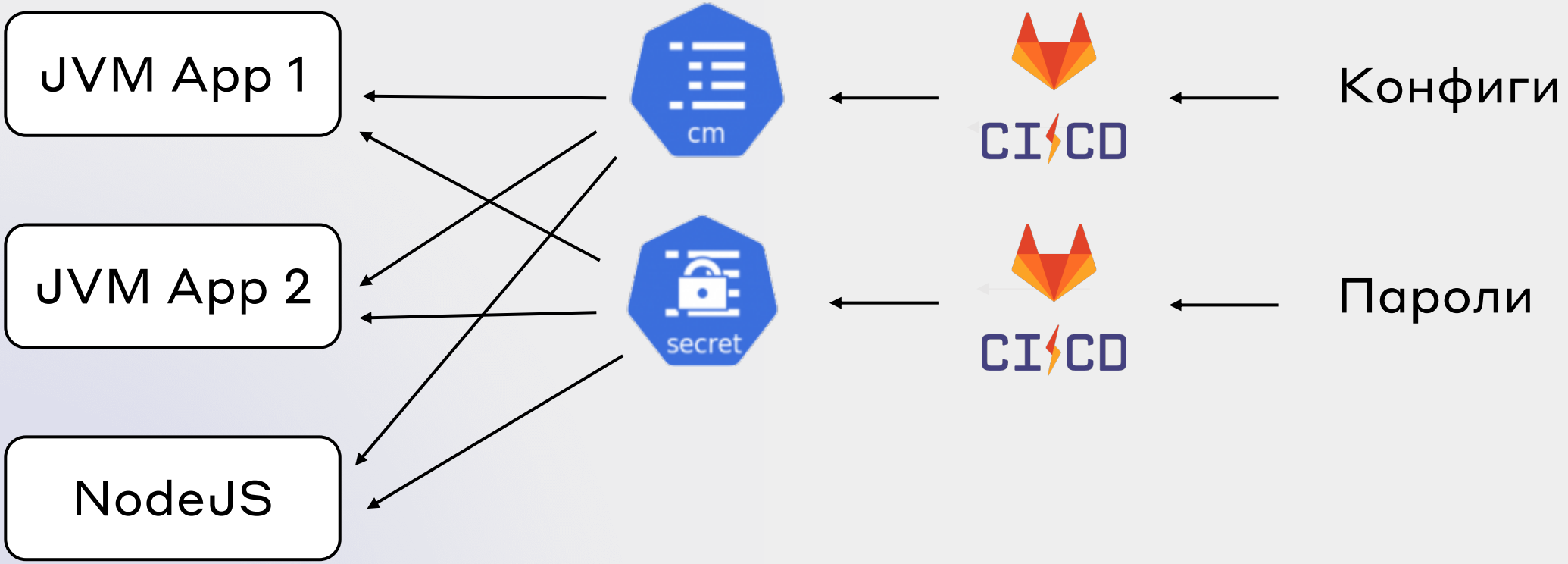
Kubernetes ConfigMaps & Secrets

Конфиги стали находится в ENV переменных

А где автоматизация?

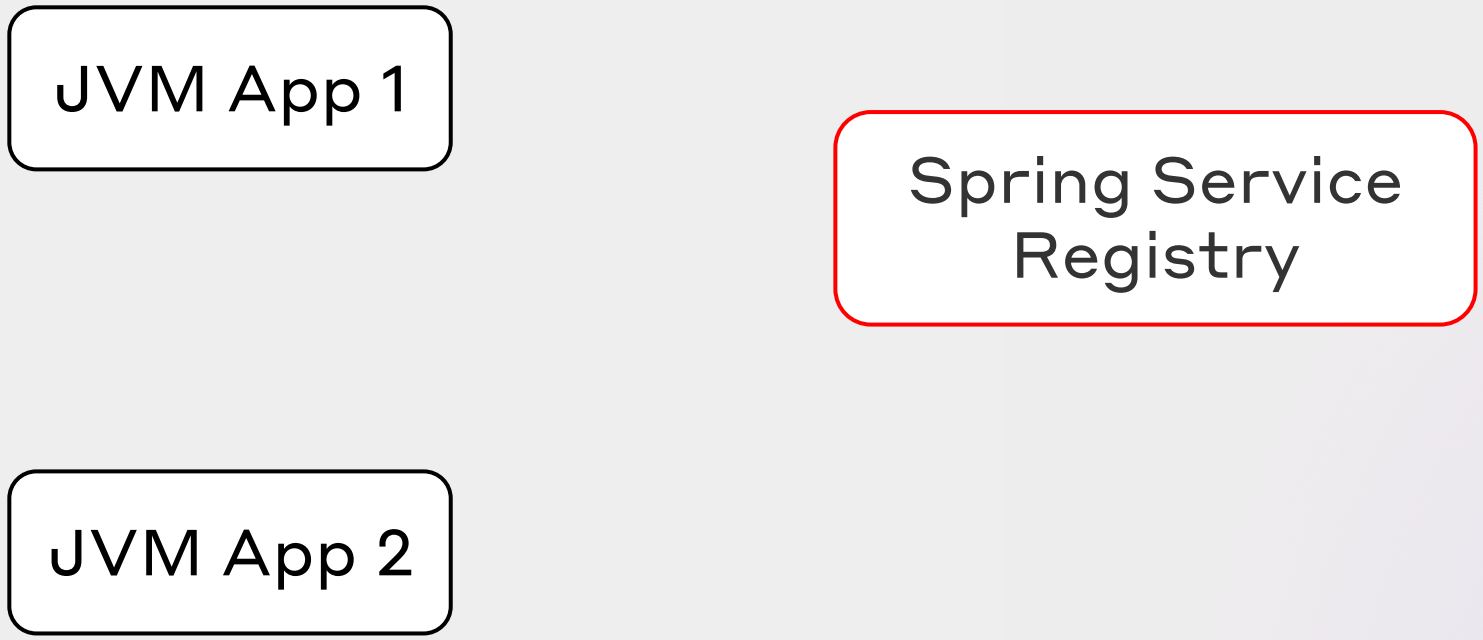


Kubernetes ConfigMaps & Secrets

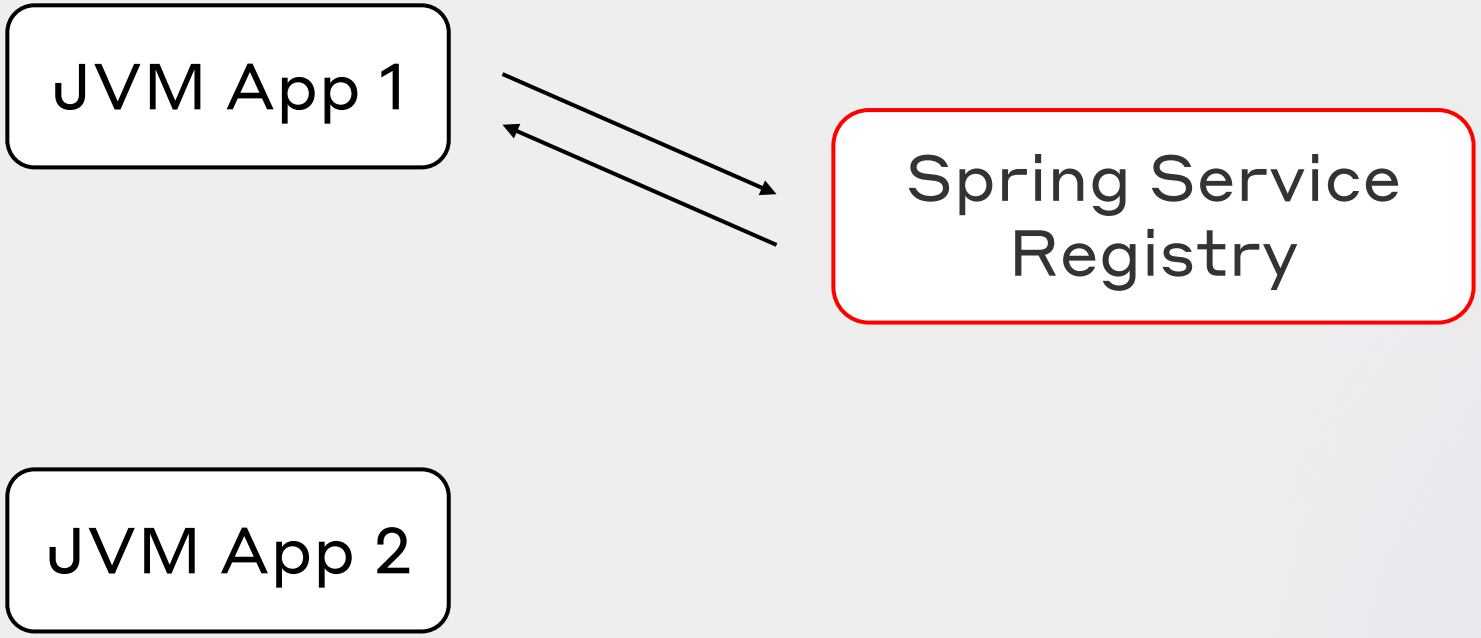


Балансировка запросов

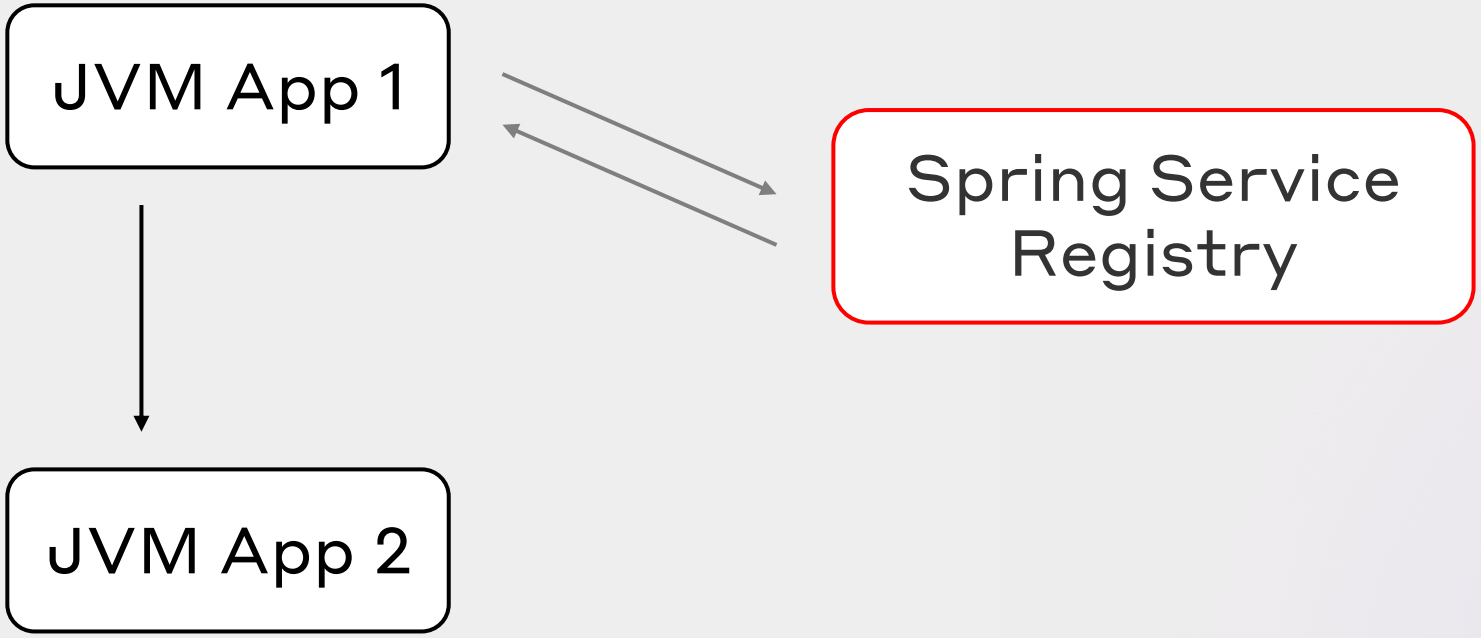
Spring Service Discovery



Spring Service Discovery



Spring Service Discovery



Kubernetes Load Balancing Networking

В Kubernetes есть встроенная возможность балансирования запросов

Kubernetes Load Balancing Networking

В Kubernetes есть встроенная возможность балансирования запросов

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

Kubernetes Load Balancing Networking

В Kubernetes есть встроенная возможность балансирования запросов

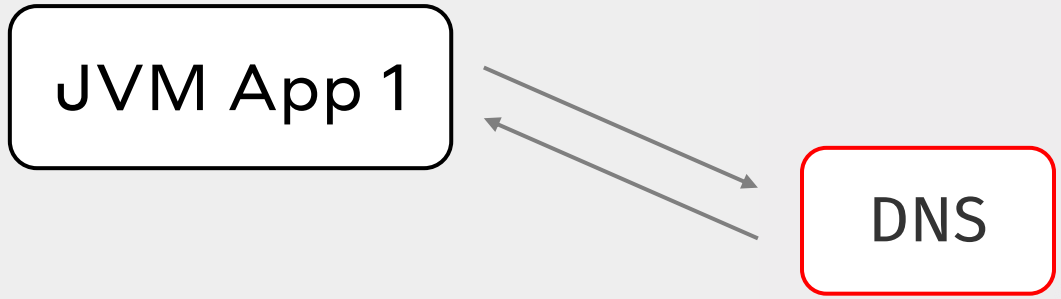
```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

Kubernetes Load Balancing Networking

В Kubernetes есть встроенная возможность балансирования запросов

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

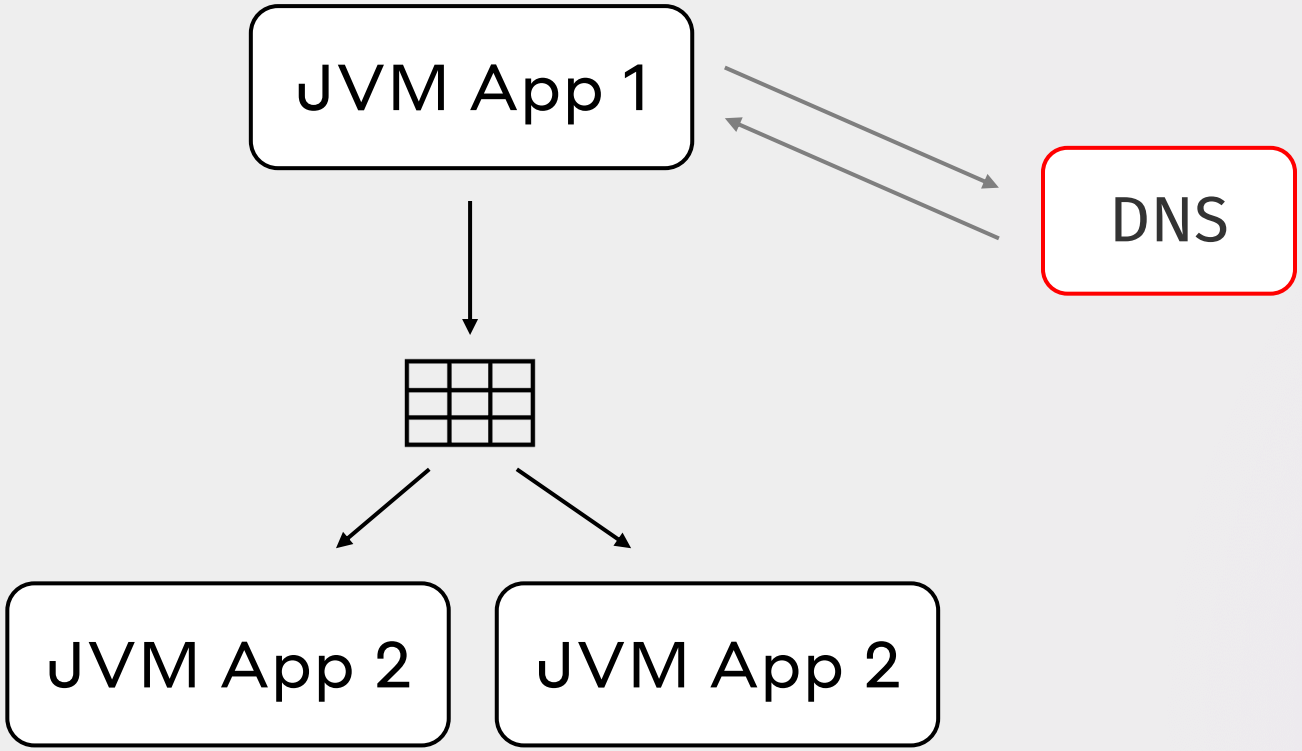
Kubernetes Load Balancing Networking



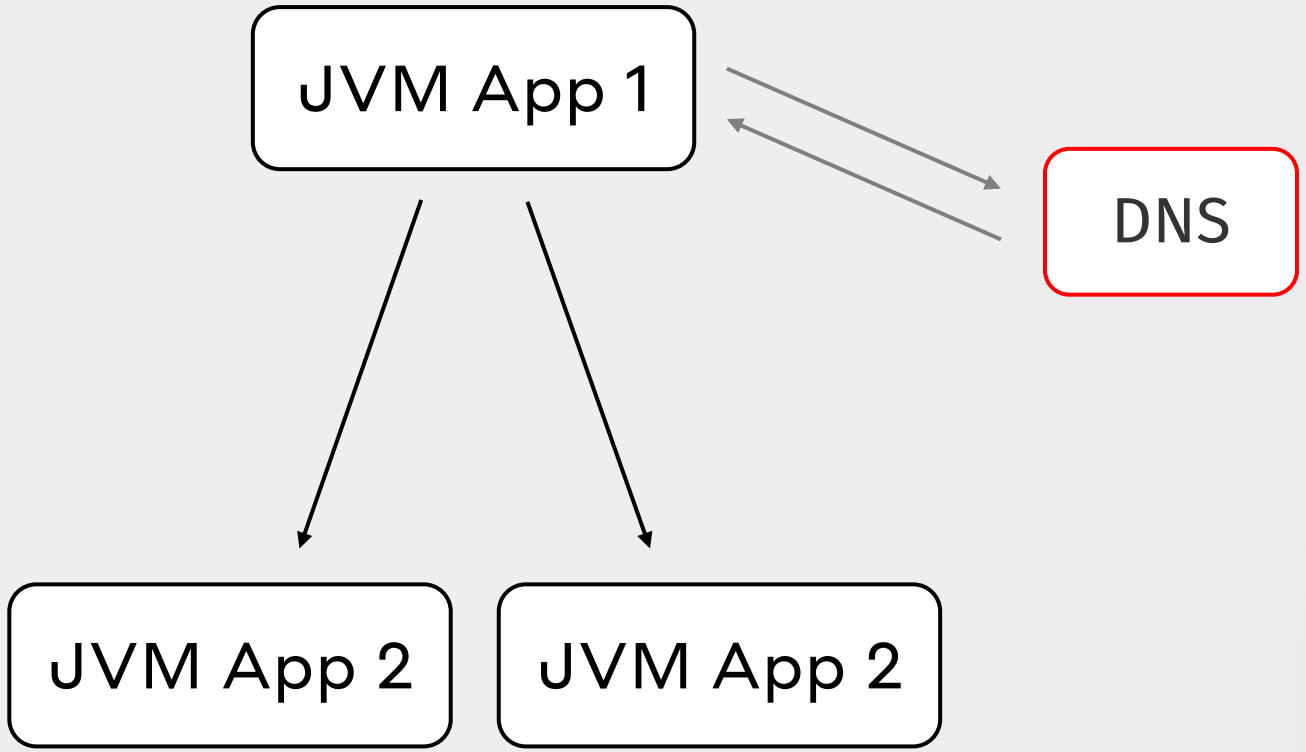
JVM App 2

JVM App 2

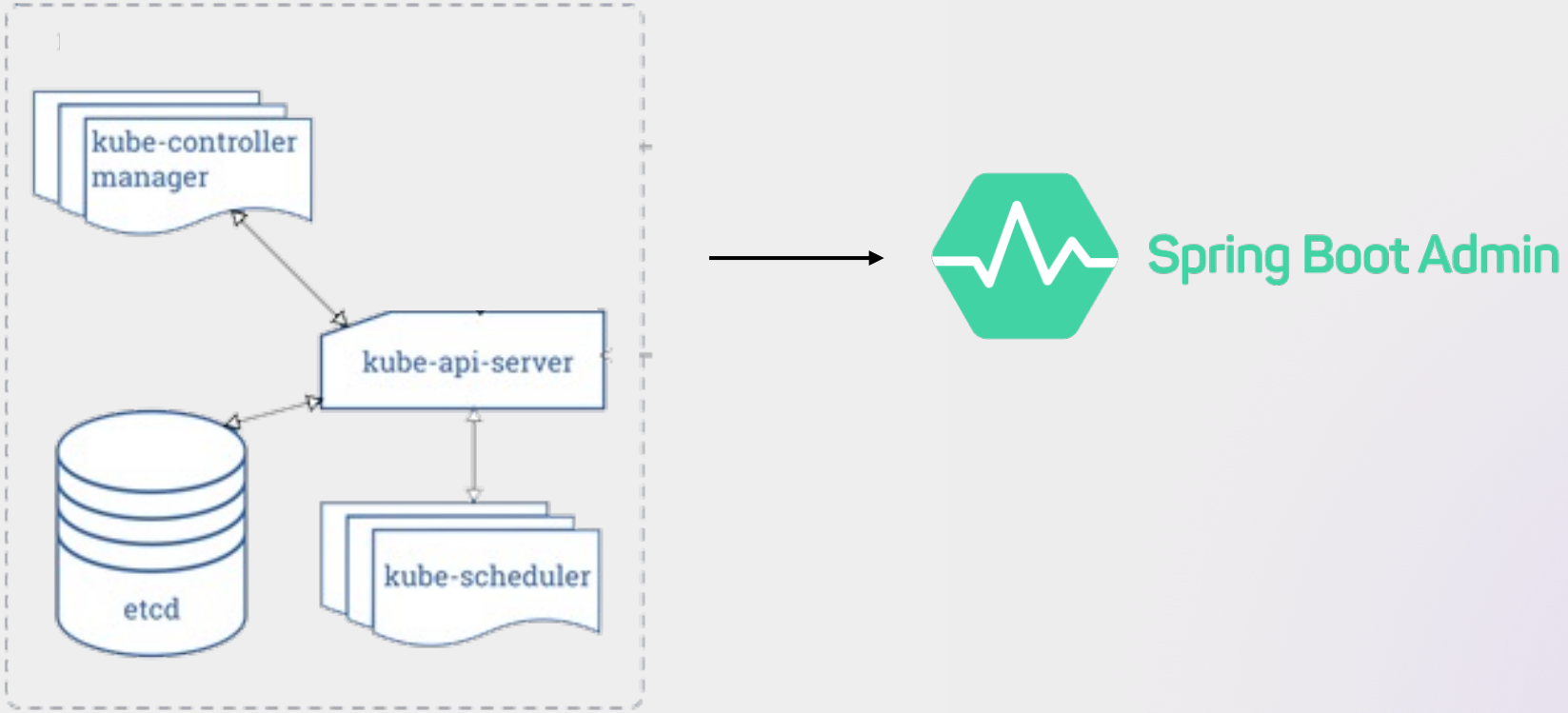
Kubernetes Load Balancing Networking



Kubernetes Load Balancing Networking



Kubernetes и Spring Boot Admin



Cloud Native

Сократили время старта приложений

Уменьшили количества библиотек
в сервисах

Уменьшили число точек отказа

Сформировали единые инструменты работы с инфраструктурой

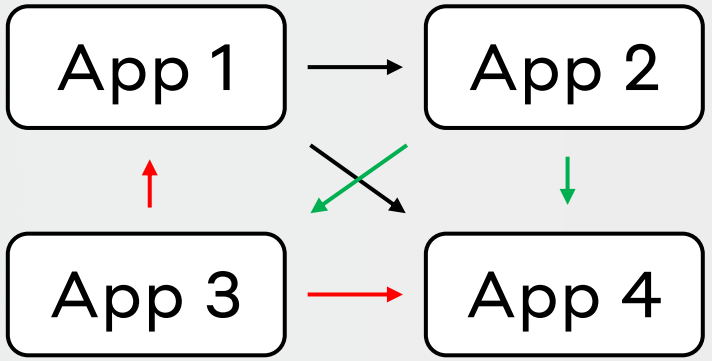
Мы тут

1. Скорость работы приложений, их деплоя и поднятия

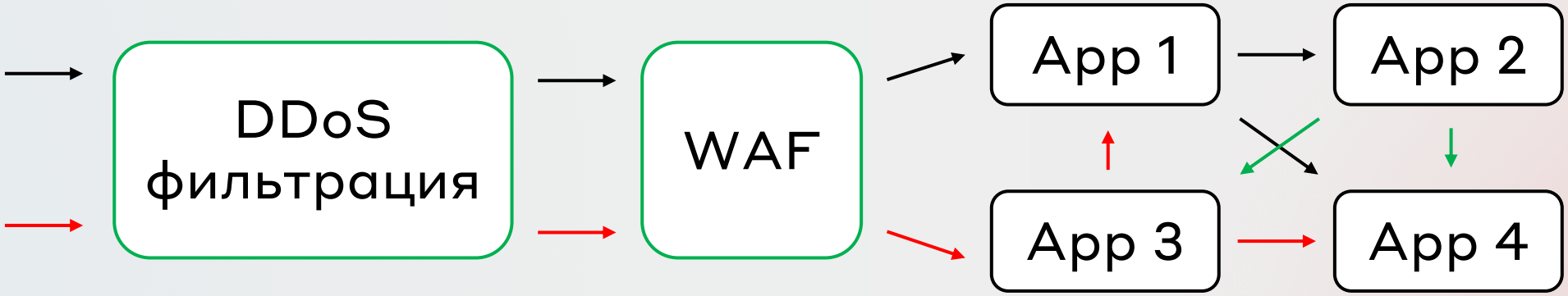
2. Использование инструментов Kubernetes для работы java приложений

3. Уезжаем в облако, и вызовы, которые оно нам приготовило

Безопасность во внутренней инфраструктуре

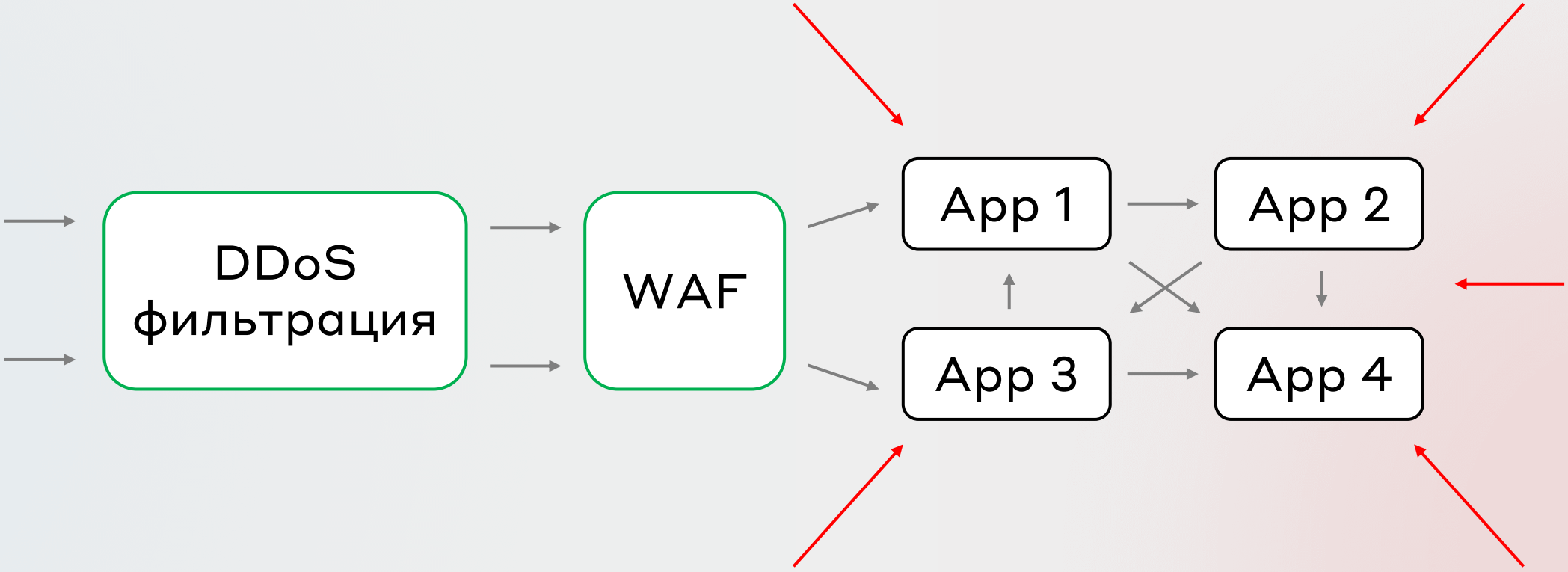


Безопасность во внутренней инфраструктуре



Безопасность в облачной инфраструктуре

Безопасность в облачной инфраструктуре



Безопасность в облачной инфраструктуре

Каждый из сервисов рассматриваться как вектор атаки

Безопасность в облачной инфраструктуре

Каждый из сервисов рассматриваться как вектор атаки

Нельзя рассчитывать на устойчивый периметр

Безопасность в облачной инфраструктуре

Каждый из сервисов рассматриваться как вектор атаки

Нельзя рассчитывать на устойчивый периметр

Аутентификация и авторизация (минимум привилегий)

Безопасность в облачной инфраструктуре

Каждый из сервисов рассматриваться как вектор атаки

Нельзя рассчитывать на устойчивый периметр

Аутентификация и авторизация (минимум привилегий)

Необходимо шифрование взаимодействий

Безопасность в облачной инфраструктуре

Каждый из сервисов рассматриваться как вектор атаки

Нельзя рассчитывать на устойчивый периметр

Аутентификация и авторизация (минимум привилегий)

Необходимо шифрование взаимодействий

Мониторинг и сбор телеметрии

Безопасность в облачной инфраструктуре

Каждый из сервисов рассматриваться как вектор атаки

Нельзя рассчитывать на устойчивый периметр

Аутентификация и авторизация (минимум привилегий)

Необходимо шифрование взаимодействий

Мониторинг и сбор телеметрии

Микросегментация

Безопасность в облачной инфраструктуре

Каждый из сервисов рассматриваться как вектор атаки

Нельзя рассчитывать на устойчивый периметр

Аутентификация и авторизация (минимум привилегий)

Необходимо шифрование взаимодействий

Мониторинг и сбор телеметрии

Микросегментация

ZERO TRUST

Безопасность в облачной инфраструктуре

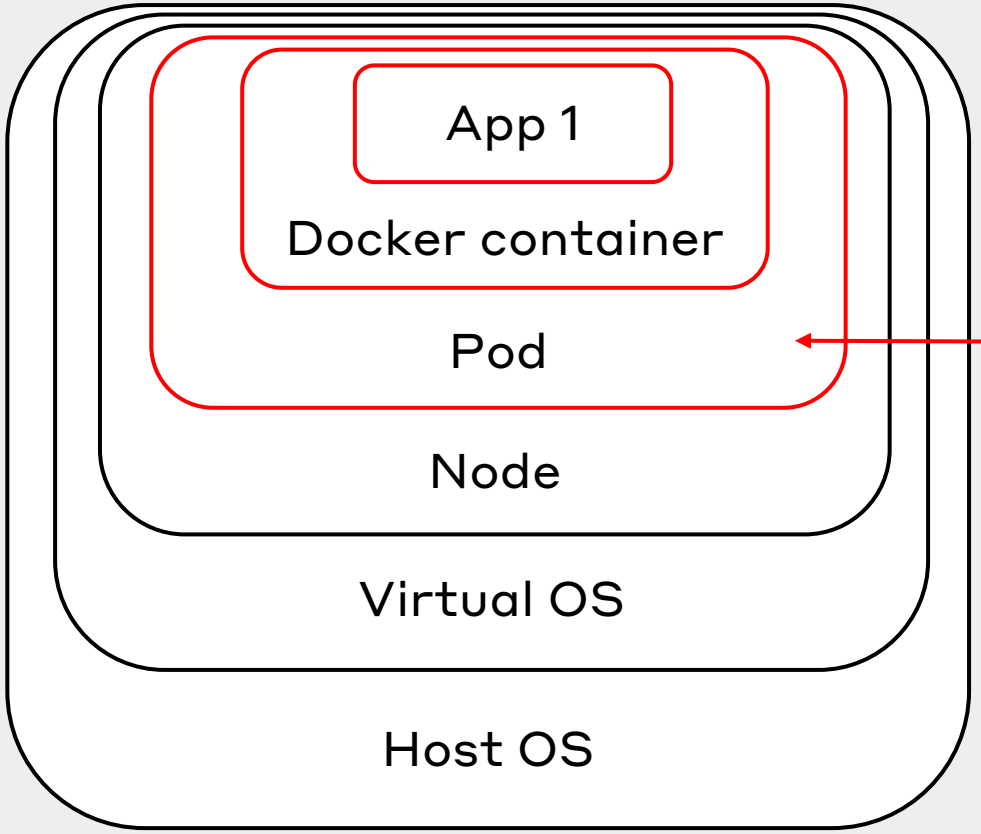
Запрещено запускать поды под учётной записью root – UID 0.

Для всех сервисов должен быть установлен параметр runAsUser

Должен быть выставлен параметр «allowPrivelegeEscflation – false»

Запрещён запуск привилегированного пода (priveleged: true)

Безопасность в облачной инфраструктуре



Тут может быть больше одного контейнера

Безопасность в облачной инфраструктуре

Должен быть выставлен параметр “readonlyRootSystem – true”

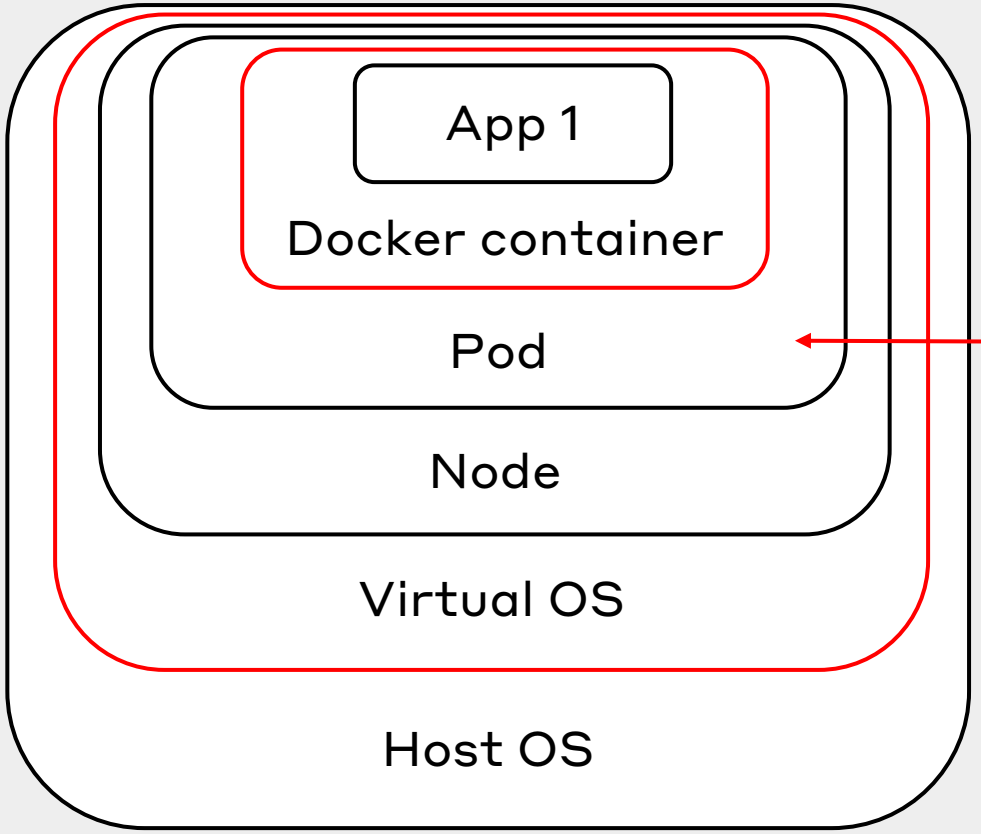
Параметры hostPID и hostIPC должны быть выставлены в значении “false”

Параметр hostNetwork должен быть выставлен в значении “false”

Запрещено использование небезопасных системных вызовов (sysctl):

- kernel.shm*,
- kernel.msg*,
- kernel.sem,
- fs.mqueue.*,”

Безопасность в облачной инфраструктуре



Тут может быть больше одного контейнера

Безопасность в облачной инфраструктуре

Все смонтированные volume должны быть только на чтение (readonly)

Capabilities выдаются по принципу наименьших привилегий drop ‘ALL’, после чего необходимые capabilities для работы

Безопасность в облачной инфраструктуре

Запрещено использовать

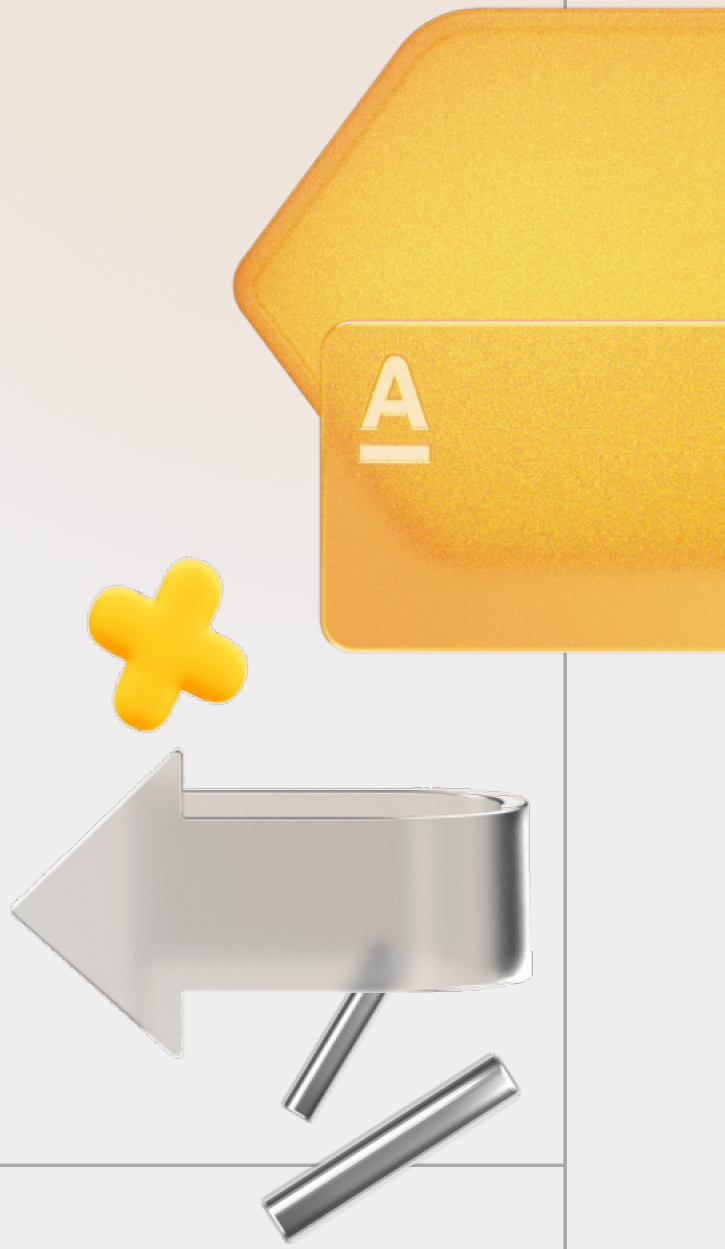
CAP_FSETID	CAP_CHOWN	CAP_SETFCAP
CAP_SETUID	CAP_NET_RAW	CAP_NET_BIND_SERVICE
CAP_SETGRID	CAP_NET_ADMIN	CAP_DAC_OVERRIDE
CAP_SYS_CHROOT	CAP_SYS_ADMIN	CAP_DAC_READ_SEARCH
CAP_SYS_PTRACE	CAP_FORMER	

Итоги

Использование нативных инструментов

- Меньше точек отказа
- Меньше кода
- Выше скорость

Время старта для приложений
важно, и за этим нужно следить



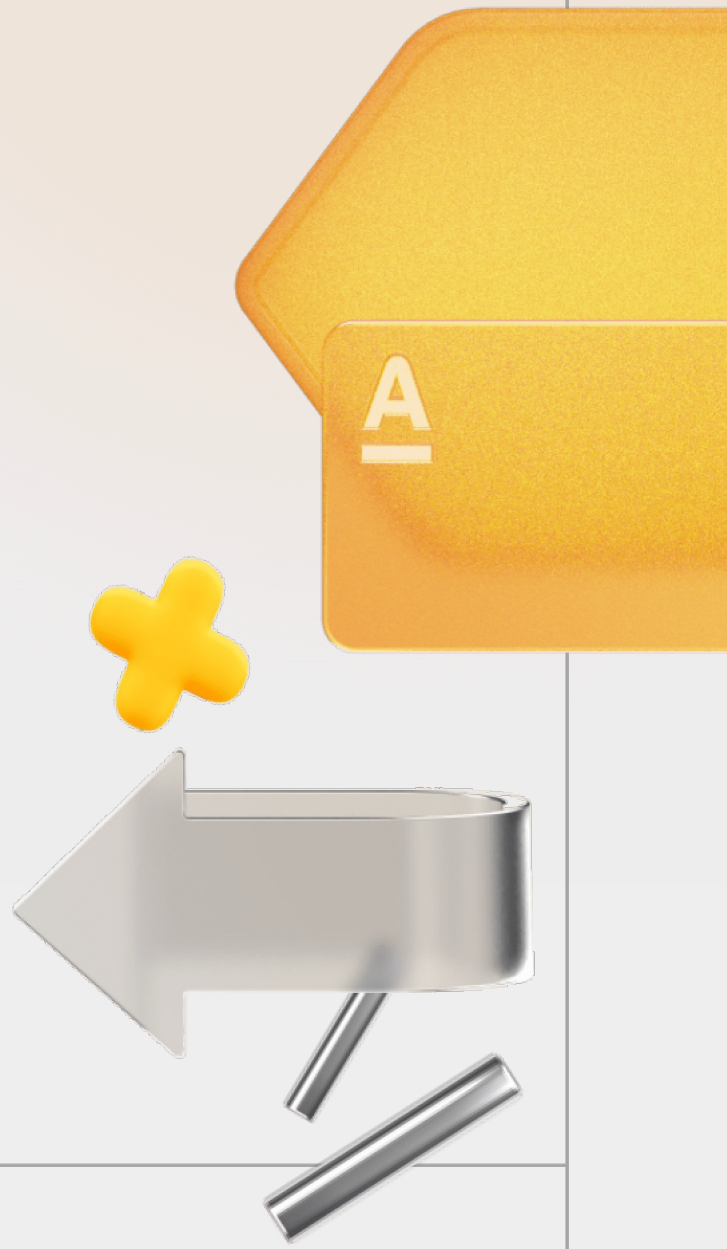
Итоги

Использование нативных инструментов

- Меньше точек отказа
- Меньше кода
- Выше скорость

Время старта для приложений
важно, и за этим нужно следить

Облако — отличный способ проверить ваш
проект на best practice



Вопросы

Чернухин Максим

https://www.facebook.com/chernukhin.maksim

@MaksCher