Грабли и плюшки обновления Spring Boot микросервисов с Java 8 на 11

Владимир Плизгá
ЦФТ

Привет!

- Владимир Плизгá
  https://toparvion.pro/
- ЦФТ (Центр Финансовых Технологий)
  Один из крупнейших разработчиков ПО в России
- Backend-разработчик (Java)
  8+ лет в деле
- TechLead в команде Интернет-банка

ЦФТ ЦЕНТР ФИНАНСОВЫХ ТЕХНОЛОГИЙ

Интернет-банк для предоплаченных карт

- 20+ федеральных партнеров

- В топ-10 рейтинга Markswebb Mobile Banking Rank 2018
  http://markswebb.ru/e-finance/mobile-banking-rank-2018/

Проект состоит из
1 монолита и
≈36 микросервисов
на Spring Boot

Ой, что будет…

1. Погружение
2. Особенности перехода
   - Сборка проекта
   - Обновление Spring
   - Deployment
3. Новшества платформы
   - Single-File Programs
   - Class Data Sharing
   - JShell
4. Всплытие

Чего точно не будет:

Обзора всех
новых фич Java
с 9 по 11 версии

Инструкции
по распилу
на модули

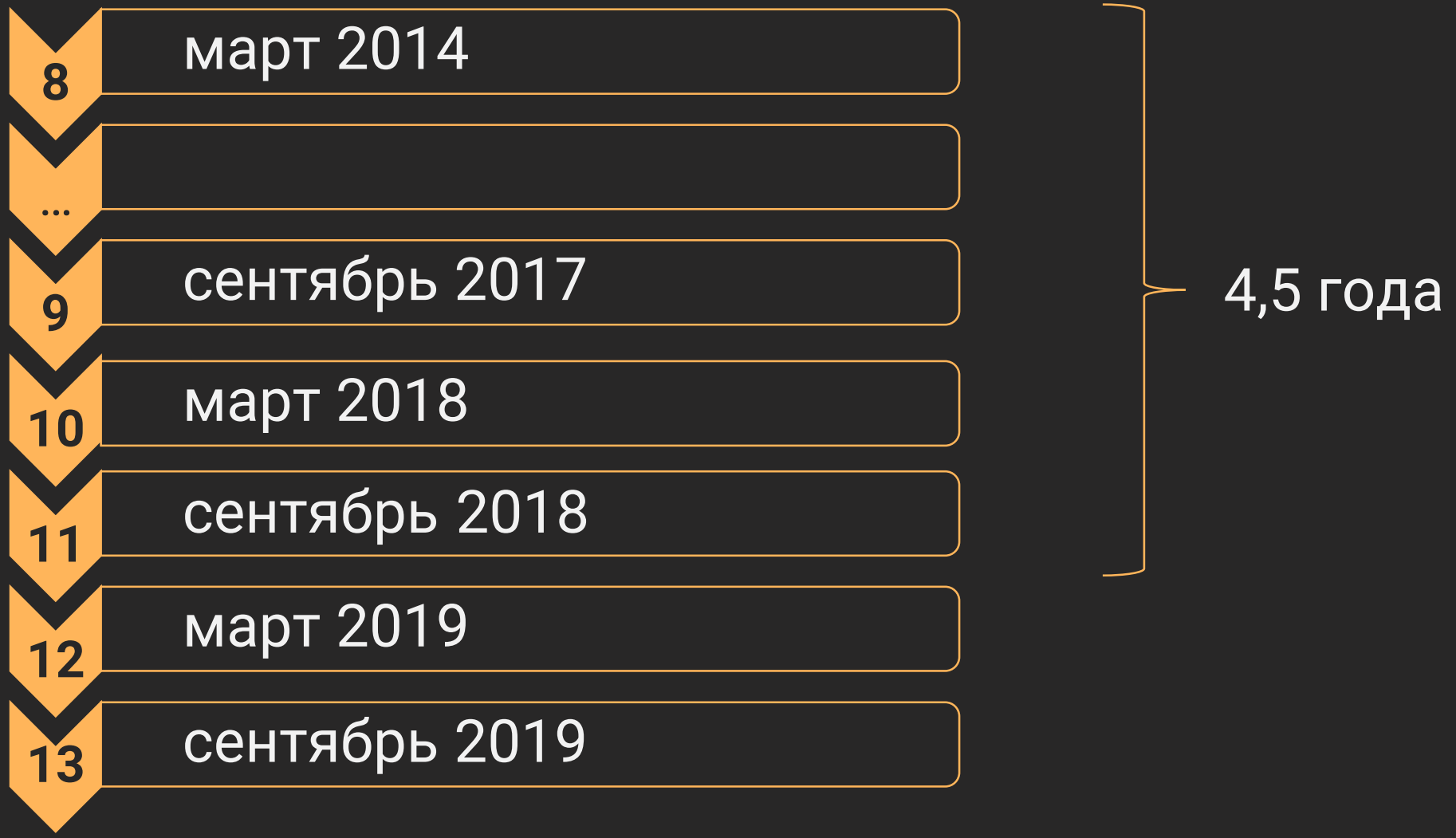
Агитации
переходить
немедленно

1. Погружение
2. Особенности перехода
   - Сборка проекта
   - Обновление Spring
   - Deployment
3. Новшества платформы
   - Single-File Programs
   - Class Data Sharing
   - JShell
4. Всплытие

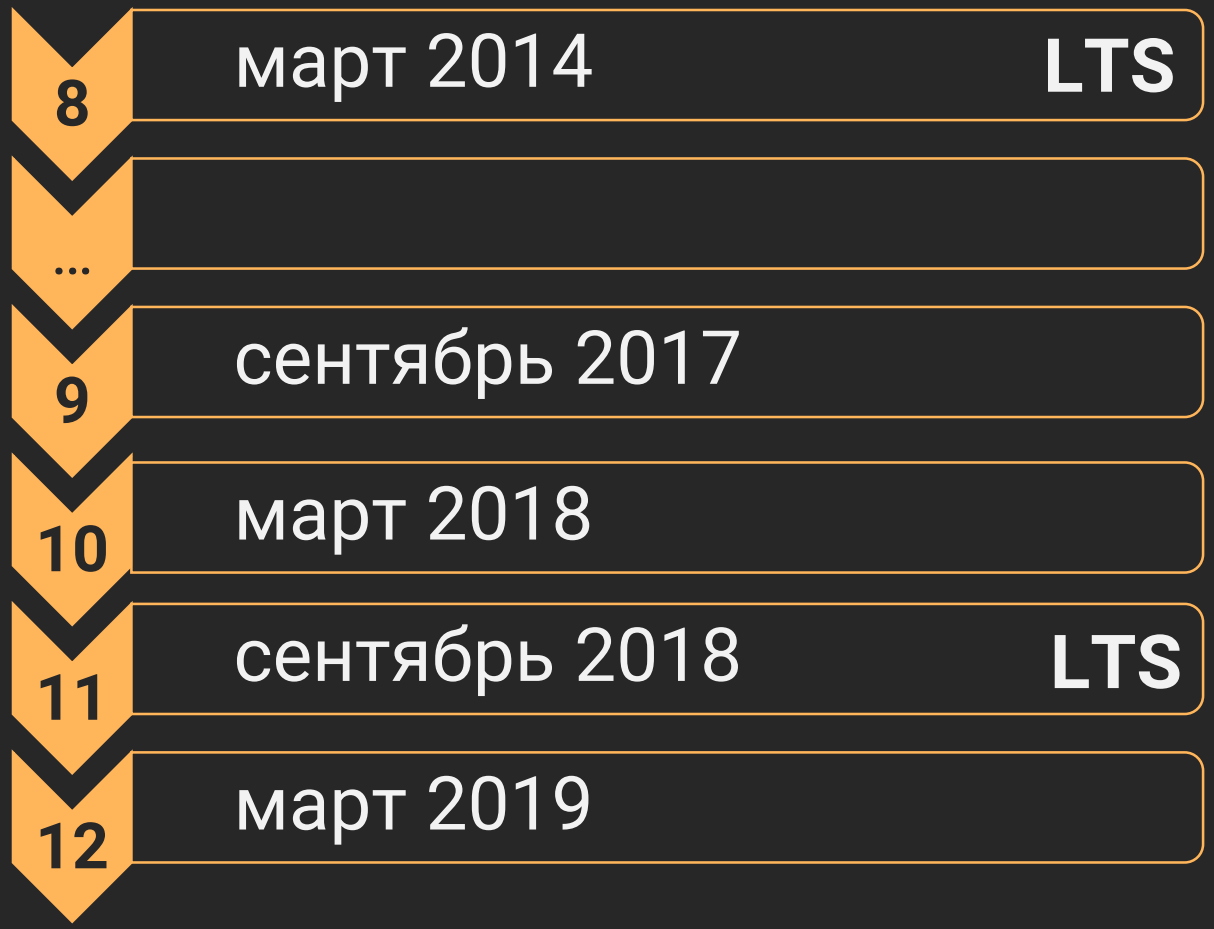
Погружение

Почему и зачем?

Цикл релизов Java

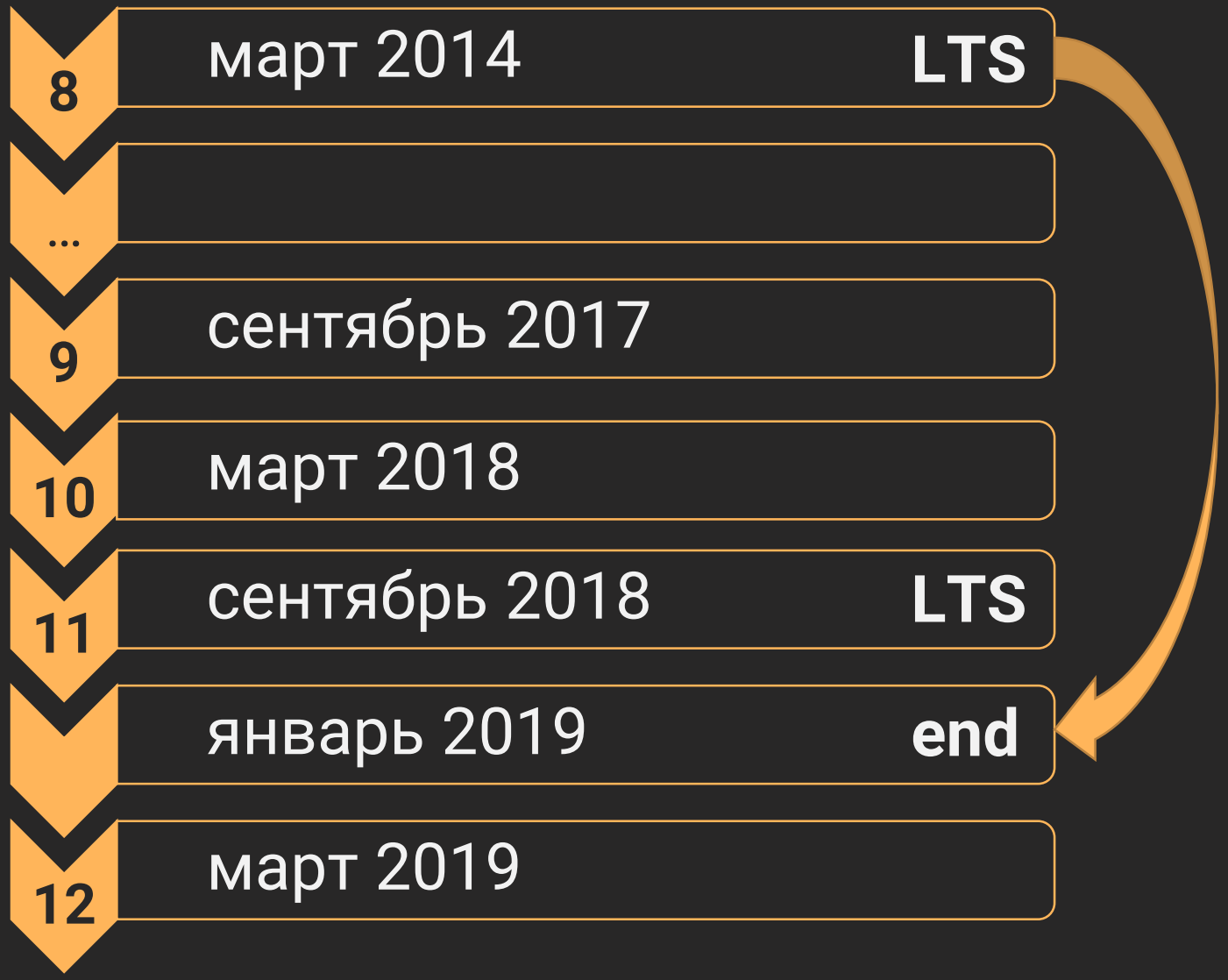


Среди релизов есть особенные



* **LTS** – Long-Term Support (Oracle)

В январе’19 версия 8 резко постарела



Public Updates от Oracle прекращены 17.01.19

Условия поддержки Java 8 у других вендоров могут отличаться

Ориентация на LTS-релизы – распространенная практика

Пример с Amazon Corretto

https://aws.amazon.com/en/corretto

Пример со Spring (и его Boot)

> Spring Framework 5.1 requires JDK 8 …
> and specifically supports JDK 11
> (as the next long-term support release)

https://github.com/spring-projects/spring-framework/wiki/Upgrading-to-Spring-Framework-5.x

И зачем это всё?

1. Погружение
2. Особенности перехода
   - Сборка проекта
   - Обновление Spring
   - Deployment
3. Новшества платформы
   - Single-File Programs
   - Class Data Sharing
   - JShell
4. Всплытие

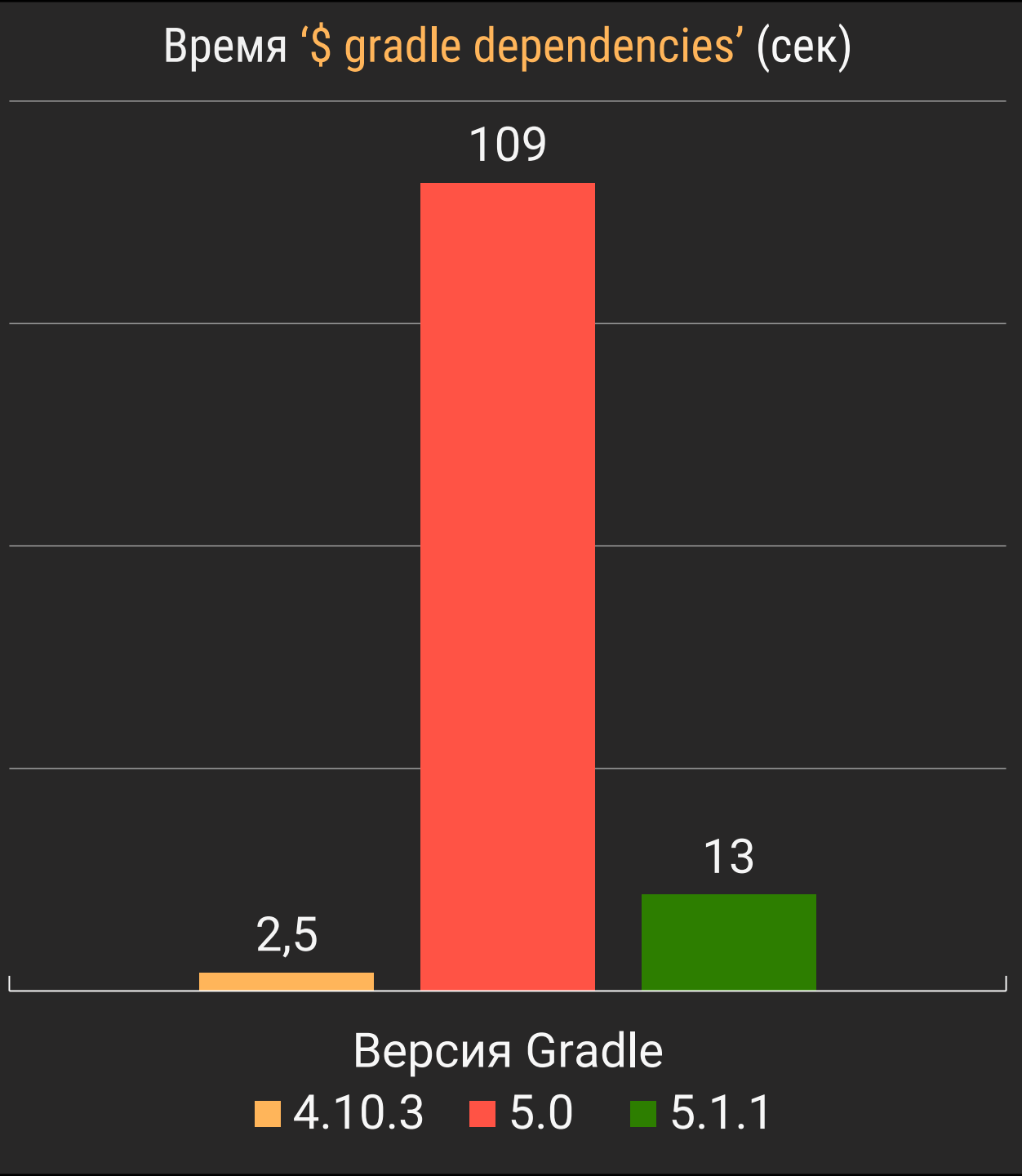
Сборка проекта на Gradle

Грабли на самом раннем этапе

Для сборки на Java 11 нужен Gradle 5+

https://github.com/gradle/gradle/issues/5120

Некоторые сборки
на Gradle 5.0
замедлились в 10+ раз



> 164 issues have been fixed in Gradle 5.0

https://docs.gradle.org/5.0/release-notes.html#fixed-issues

http://images6.fanpop.com/image/photos/40000000/The-X-Files-The-Truth-Is-Out-There-cosmiccastaway-40025205-500-283.gif

Не все новые версии Gradle одинаково полезны

Поддержка BOM (как было)

```
plugins {
  id 'org.springframework.boot' version '2.1.3.RELEASE'
}
apply plugin: 'io.spring.dependency-management'
dependencies {
  implementation 'org.springframework.boot'
                 + ':spring-boot-starter'
                 #+ ':2.1.3.RELEASE'
}
```

build.gradle

Поддержка BOM (как стало)

```
dependencies {
  implementation platform(
            'org.springframework.boot'
          + ':spring-boot-dependencies'
          + ':2.1.3.RELEASE')
  implementation 'org.springframework.boot'
                + ':spring-boot-starter'
                #+ ':2.1.3.RELEASE'
```

build.gradle

Разделение зависимостей

Compile & Runtime Scopes

Во времена Gradle 4 жил-был импорт…

```java
// ...

import org.apache.commons.collections.CollectionUtils;

// ...
```

SomeClass.java

… а потом случился Gradle 5

```
$ gradle compileJava
...
SomeClass.java:24: error: package
org.apache.commons.collections does not exist
import org.apache.commons.collections.CollectionUtils;
                                     ^
```

```
$ gradle compileJava
```

Зависимость оказалась транзитивной

```
$ gradle dependencyInsight --dependency commons-collections \
                           --configuration compile
> Task dependencyInsight

commons-collections
\--- ribbon-httpclient
     \--- spring-cloud-starter-netflix-ribbon
          \--- spring-cloud-starter-netflix-eureka-client
               \--- project :shared
                    \--- compile
```

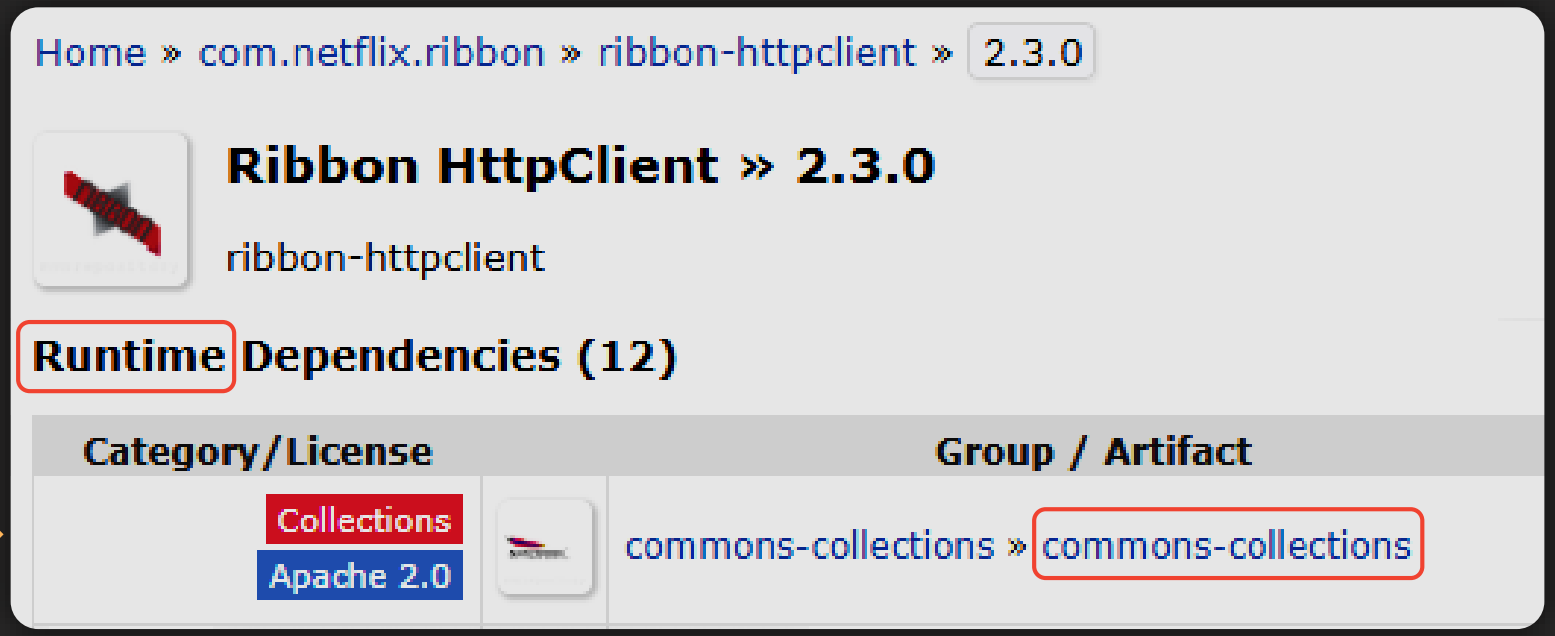
$ gradle dependencyInsight

Новшество версии Gradle 5.0

> … the compilation classpath **only** includes compile-scoped dependencies

https://docs.gradle.org/5.0/userguide/upgrading_version_4.html

Источник runtime зависимости



https://mvnrepository.com/artifact/com.netflix.ribbon/ribbon-httpclient/2.3.0

```
commons-collections
\--- ribbon-httpclient
     \--- spring-cloud-starter-netflix-ribbon
          \--- spring-cloud-starter-netflix-eureka-client
               \--- project :shared
                    \--- compile
```

И как быть?

- Менять транзитивные зависимости на явные
- Анализировать зависимости (Gradle):
  - `dependencies --configuration compile`
  - `dependencyInsight --configuration compile --dependency myDep`
  - Build Scan https://scans.gradle.com/

Минутка справедливости: Maven

Для работы с Java 11 нужен Maven 3.5.0, а также:

- compiler plugin: 3.8.0
- surefire & failsafe: 2.22.0

Полезные материалы:

- https://blog.codefx.org/java/java-11-migration-guide/
- https://winterbe.com/posts/2018/08/29/migrate-maven-projects-to-java-11-jigsaw/

1. Погружение
2. Особенности перехода
   - Сборка проекта
   - Обновление Spring
   - Deployment
3. Новшества платформы
   - Single-File Programs
   - Class Data Sharing
   - JShell
4. Всплытие

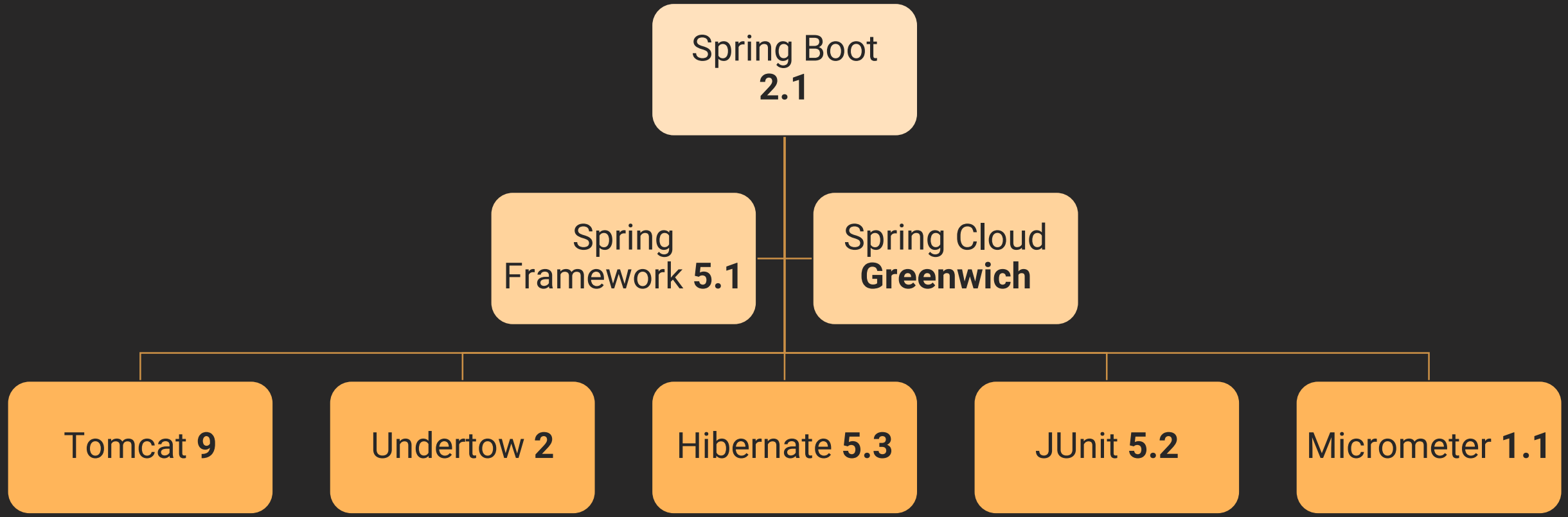
Причём тут Spring?

> Java 11 is supported as of Spring Boot 2.1.0.M2*

https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-with-Java-9-and-above

** Но можно пробовать и на меньших версиях*

Версии транзитивных зависимостей



Запрет переопределения бинов

Новшество версии 2.1

Bean overriding
has been disabled by default
to prevent a bean being
accidentally overridden

https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-2.1-Release-Notes

Такие изменения бывают не только случайными

Пример 1: библиотечный код

```
@Configuration
public class FeignClientsConfiguration {

  @Bean
  @ConditionalOnMissingBean(FeignLoggerFactory.class)
  public FeignLoggerFactory feignLoggerFactory() {
    return new DefaultFeignLoggerFactory(logger);
  }
}
```

org.springframework.cloud.openfeign.FeignClientsConfiguration

Пример 1: прикладной код

```
@Configuration
public class DefaultFeignClientConfiguration {

  @Bean
  public FeignLoggerFactory feignLoggerFactory() {
    return new MyFeignLoggerFactory();
  }
```

com.example.DefaultFeignClientConfiguration

Пример 1: последствия

```
BeanDefinitionOverrideException: …
Cannot register bean definition […]
  for bean 'feignLoggerFactory':
There is already […] bound.
```

application log

Решение (вариант 1): глобально

```
spring.main.allow-bean-definition-overriding=true
```

application.properties

+ Быстро, просто, понятно
+ Обратно совместимо
+ Масштабно

– Не безопасно
– Не гибко

Решение (вариант 2): локально

```java
@SpringBootTest
class MyTest {
  // (test methods)

  @Configuration
  static class TestConfiguration {
    @Bean
    @Primary
    RetryListener customRetryListener() {
      return new TestRetryListener();
    }
  }
}
```

MyTest.java

Пояснения к решению 2

- Префикс `custom` (или иной) – для различия имен
- Аннотация `@Primary` – для перекрытия исходного бина

\+ Безопасно
\+ Контролируемо
\+ Гибко

– Не масштабируемо
– [Трудоемко]

Оптимизация производительности

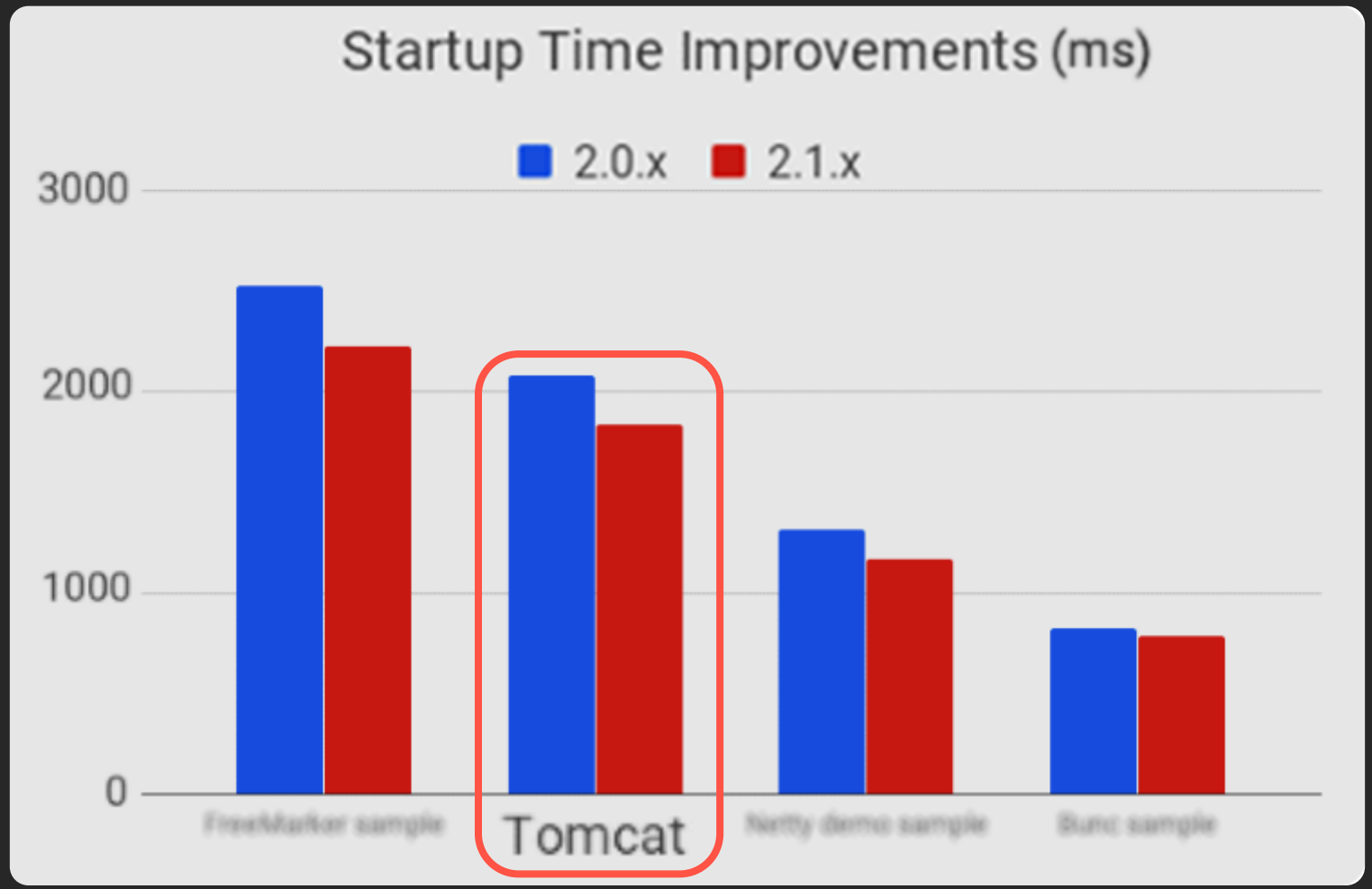
Есть хорошая новость

> … Spring Boot 2.1 and Spring 5.1 have some quite nice optimizations for startup time and heap usage

https://spring.io/blog/2018/12/12/how-fast-is-spring

По данным испытаний от разработчиков

Время запуска
на Tomcat
сократилось
на ≈14%



https://github.com/dsyer/spring-boot-allocations

По нашему опыту

Как еще можно ускорить запуск?

Источники вдохновения от Dave Syer (@david_syer):

- https://spring.io/blog/2018/12/12/how-fast-is-spring
- https://www.youtube.com/watch?v=97UTDmonq7w
- https://github.com/dsyer/spring-boot-allocations

Кардинальный метод ускорения

https://jpoint.ru/talks/1emn9byzaklzozk6ec56dp

Опробованные способы (1/2)

- Fix the location of the Spring Boot config file(s)

  Удобно (кое-где), но не очень эффективно

- Unpack the fat jar and run with an explicit classpath

  Перспективно

- Run the JVM with –noverify

  Сомнительно, см. https://www.youtube.com/watch?v=-OocG7tFIOQ

Опробованные способы (2/2)

- Use Class Data Sharing (CDS)

  Не просто, но полезно (см. далее)

- «Let's make SpringBoot app start faster»

  https://dev.to/bufferings/lets-make-springboot-app-start-faster-k9m

- Use Spring Boot 2.1 and Spring 5.1

  Бесплатно, без регистрации, без СМС*

*)

> … several introspection algorithms
> have been streamlined …
> potentially causing side effects

https://github.com/spring-projects/spring-framework/wiki/Upgrading-to-Spring-Framework-5.x

Lombok

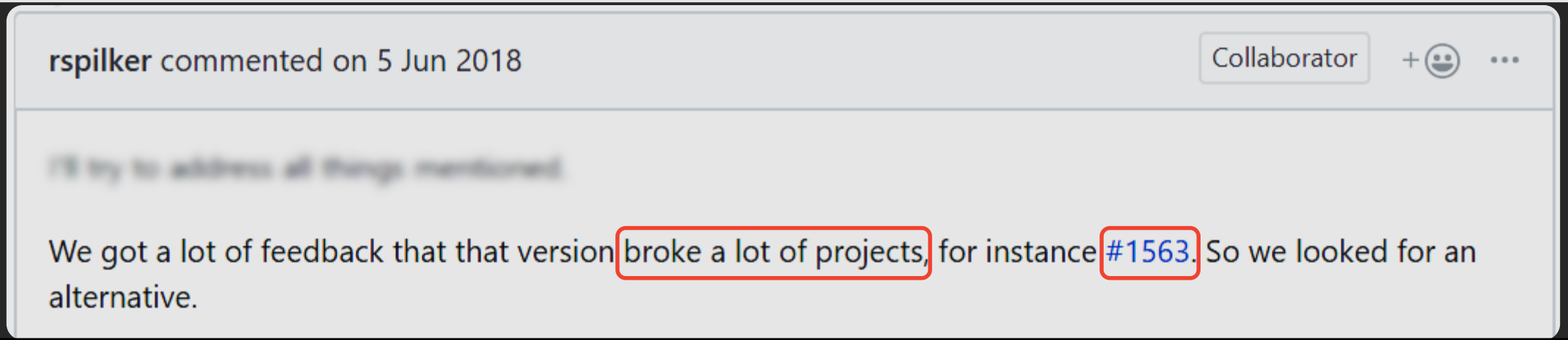
Причем здесь Lombok?

> Spring Boot 2.1 has upgraded to Lombok 1.18.x
> [which] will no longer generate
> a private, no-args constructor by default

https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-2.1-Release-Notes

Причина изменения

https://github.com/rzwitserloot/lombok/issues/1708



Пример из реальной жизни

```java
@Getter
@ToString
@RequiredArgsConstructor
class SendInfoCommand {
  private final String first;
  private final String second;
}
```

SendInfoCommand.java

```java
@RabbitListener(queues = QUEUE_NAME)
public void receive(@Payload SendInfoCommand sendInfoCommand) {
  // listener implementation
}
```

CommandListener.java

И это работает!
(по крайней мере,
до 1.16.20)

Запуск на Lombok 1.16.20

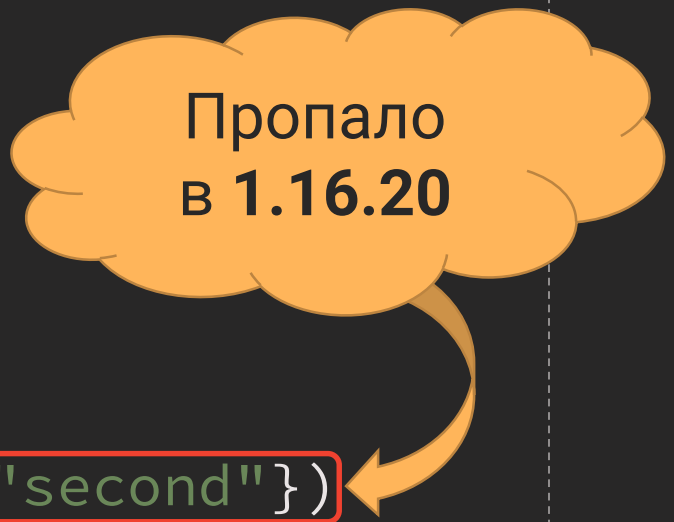
```
Caused by:
com.fasterxml.jackson.databind.exc.InvalidDefinitionException:
Cannot construct instance of `SendInfoCommand`
  (no Creators, like default construct, exist):
...
```

application log

DeLombok

```
@Getter
@ToString
@RequiredArgsConstructor
class SendInfoCommand {
  private final String first;
  private final String second;

  @java.beans.ConstructorProperties({"first", "second"})
  public SendInfoCommand(String first, String second) {
    this.first = first;
    this.second = second;
  }
}
```



SendInfoCommand.java

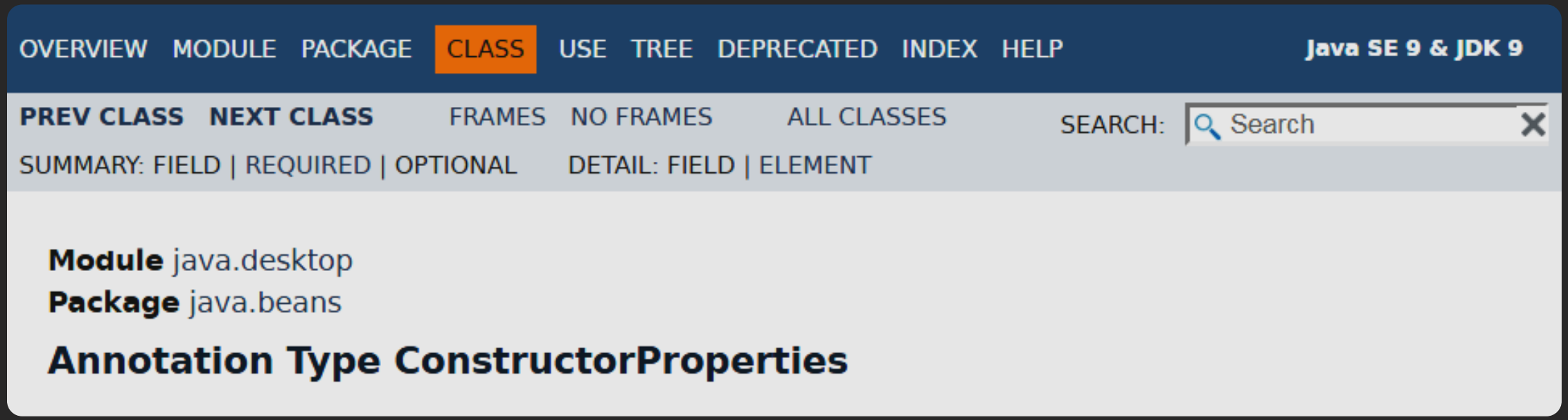
Lombok Changelog (v1.16.20)

BREAKING CHANGE: by default lombok no longer … generates @ConstructorProperties…

Oracle … broke this annotation with the release of JDK9

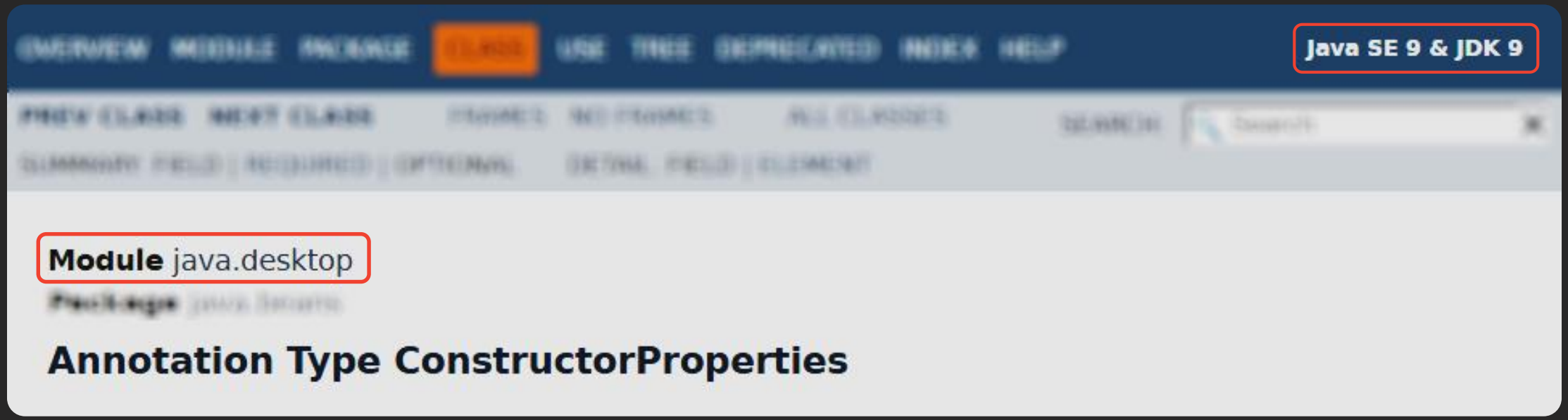
https://projectlombok.org/changelog

Причем тут JDK9?



https://docs.oracle.com/javase/9/docs/api/java/beans/ConstructorProperties.html

Причем тут JDK9?



https://docs.oracle.com/javase/9/docs/api/java/beans/ConstructorProperties.html

Зачем добавили extraPrivate=true?

> The reason we added this feature was that since java9 @ConstructorProperties moved to the module java.desktop, which is not added by default.

https://github.com/rzwitserloot/lombok/issues/1708

Варианты решения

1. Вернуть генерацию аннотации вручную:

```
lombok.anyConstructor.addConstructorProperties=true
```

lombok.config

   - Требует зависимости от `java.desktop` на Jigsaw

2. Использовать аннотацию `@Setter`
   - Делает класс изменяемым (mutable)
   - Может не помочь на 1.18, потому что...

Lombok Changelog (v1.18.0)

BREAKING CHANGE: configuration key
lombok.noArgsConstructor.extraPrivate
is now false by default

https://projectlombok.org/changelog

Варианты решения II

1. Вернуть генерацию конструктора вручную:

```
lombok.noArgsConstructor.extraPrivate=true
```

lombok.config

2. Использовать обходные пути:
   - ➢ https://github.com/rzwitserloot/lombok/issues/1708

Краткая хроника событий

Не все то золото, что Lombok

Что можно сделать?

- Завести в проекте файл `lombok.config`
- Договориться с командой о порядке применения lombok

Пример файла `lombok.config`

```
# Stops further searching for lombok.config
config.stopBubbling = true

# Stable features
lombok.val.flagUsage = error
# ...

# Experimental features
lombok.extensionMethod.flagUsage = error
# ...

# Our discussed settings
lombok.equalsAndHashCode.doNotUseGetters = true
lombok.anyConstructor.addConstructorProperties = true
```

lombok.config

Что можно сделать?

- Завести в проекте файл `lombok.config`
- Договориться с командой о порядке применения lombok
- Не полагаться на версию Lombok из Spring Boot BOM
- Не забывать о delombok
-  ;)

1. Погружение
2. Особенности перехода
   - Сборка проекта
   - Обновление Spring
   - Deployment
3. Новшества платформы
   - Single-File Programs
   - Class Data Sharing
   - JShell
4. Всплытие

Базовый образ JDK

Для запуска в Docker

В чем вопрос?

- При Java 8 был популярен образ

```
openjdk:8-jre-alpine*
```

- Для Java 11 такого образа нет

✓ Проверенный
✓ Обновляемый
✓ Легкий (80 МБ)

* **Alpine** – компактный дистрибутив Linux

В Java 11 больше нет JRE.
Но можно собрать из модулей.

И как это сделать?

https://2018.jokerconf.com/2018/talks/4sanwhahpe8kgagmceqsyk

Официальный образ
`openjdk:11-jre-slim`
весит ≈270 МБ

Некоторые способы остаться на Alpine

- Использовать AdoptOpenJDK
  https://github.com/AdoptOpenJDK/openjdk-docker — ≈240 MB
- Перейти на Azul Zulu OpenJDK
  или на Bellsoft Liberica OpenJDK — ≈200 MB
- Собрать самому
  https://stackoverflow.com/a/53669152/3507435 — ≈75 MB
- Дождаться реализации проекта Portola
  https://openjdk.java.net/projects/portola/ — ? MB

Образы AdoptOpenJDK

- Пример тега

```
adoptopenjdk/openjdk11:jdk-11.0.2.9-alpine-slim
```

- Включают опцию `-XX:+UseContainerSupport`
- Поставляются с HotSpot и OpenJ9

> The Alpine Linux and the slim images are not yet TCK certified

https://github.com/AdoptOpenJDK/openjdk-docker

Подробнее о других вариантах

https://jpoint.ru/talks/35hks9g6etavlgijj8nfvh

1. Погружение
2. Особенности перехода
   - Сборка проекта
   - Обновление Spring
   - Deployment
3. Новшества платформы
   - Single-File Programs
   - Class Data Sharing
   - JShell
4. Всплытие

1. Погружение
2. Особенности перехода
   - Сборка проекта
   - Обновление Spring
   - Deployment
3. Новшества платформы
   - Single-File Programs
   - Class Data Sharing
   - JShell
4. Всплытие

Суть [App]CDS в одном слайде

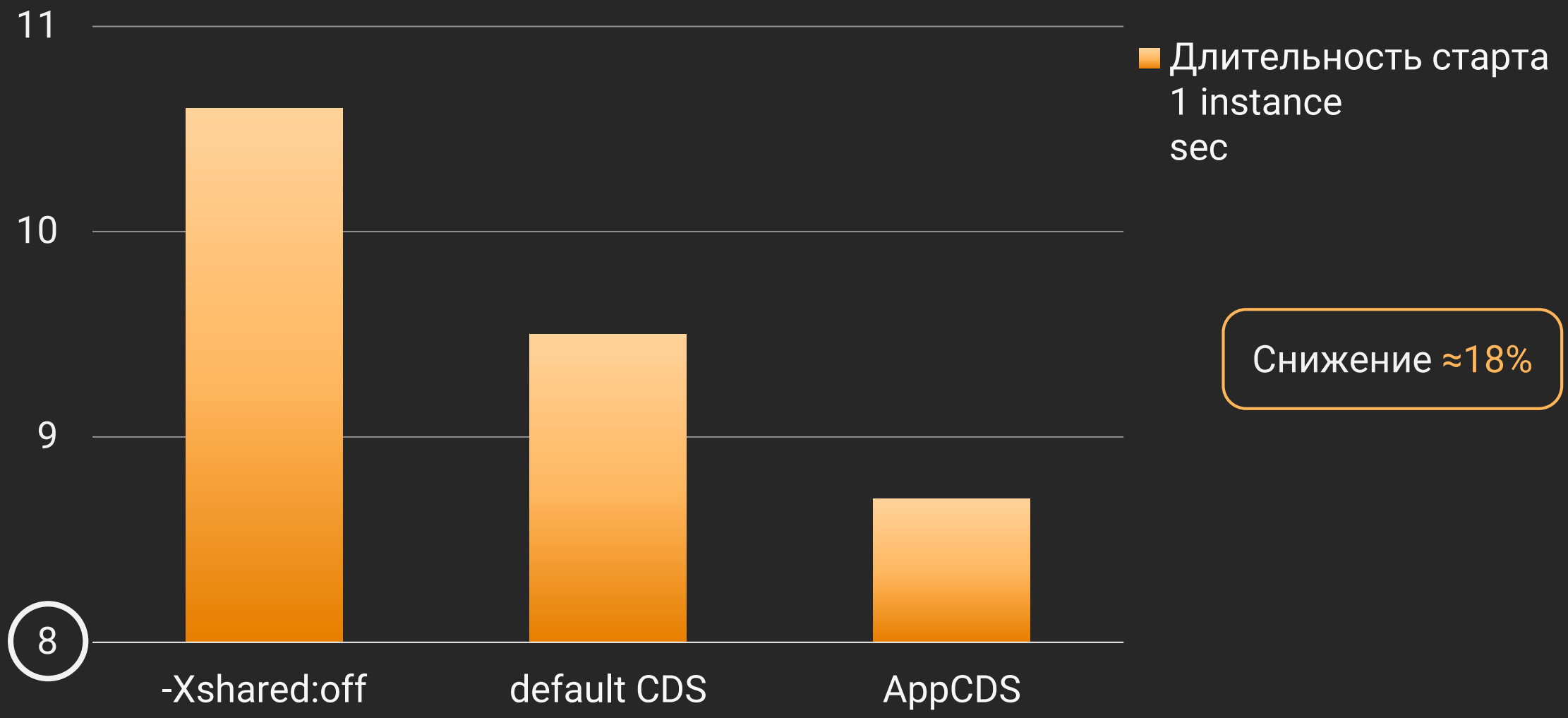
- Разные экземпляры JVM загружают одни и те же классы JDK
- На это тратится время CPU и пространство RAM
- Идея 1: загрузить эти классы в архив и расшарить его между экземплярами JVM

CDS

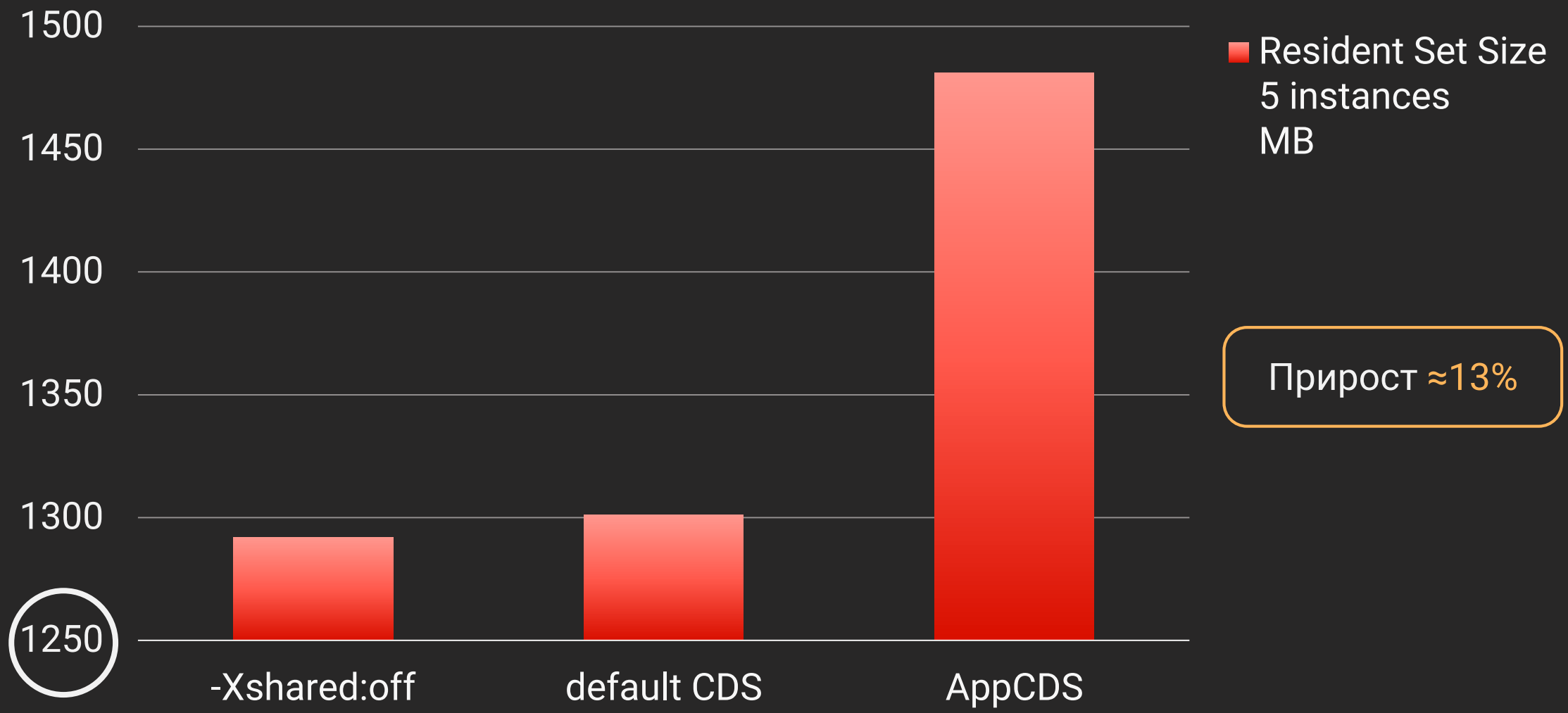
- Идея 2: сделать то же самое для классов приложения и сторонних библиотек

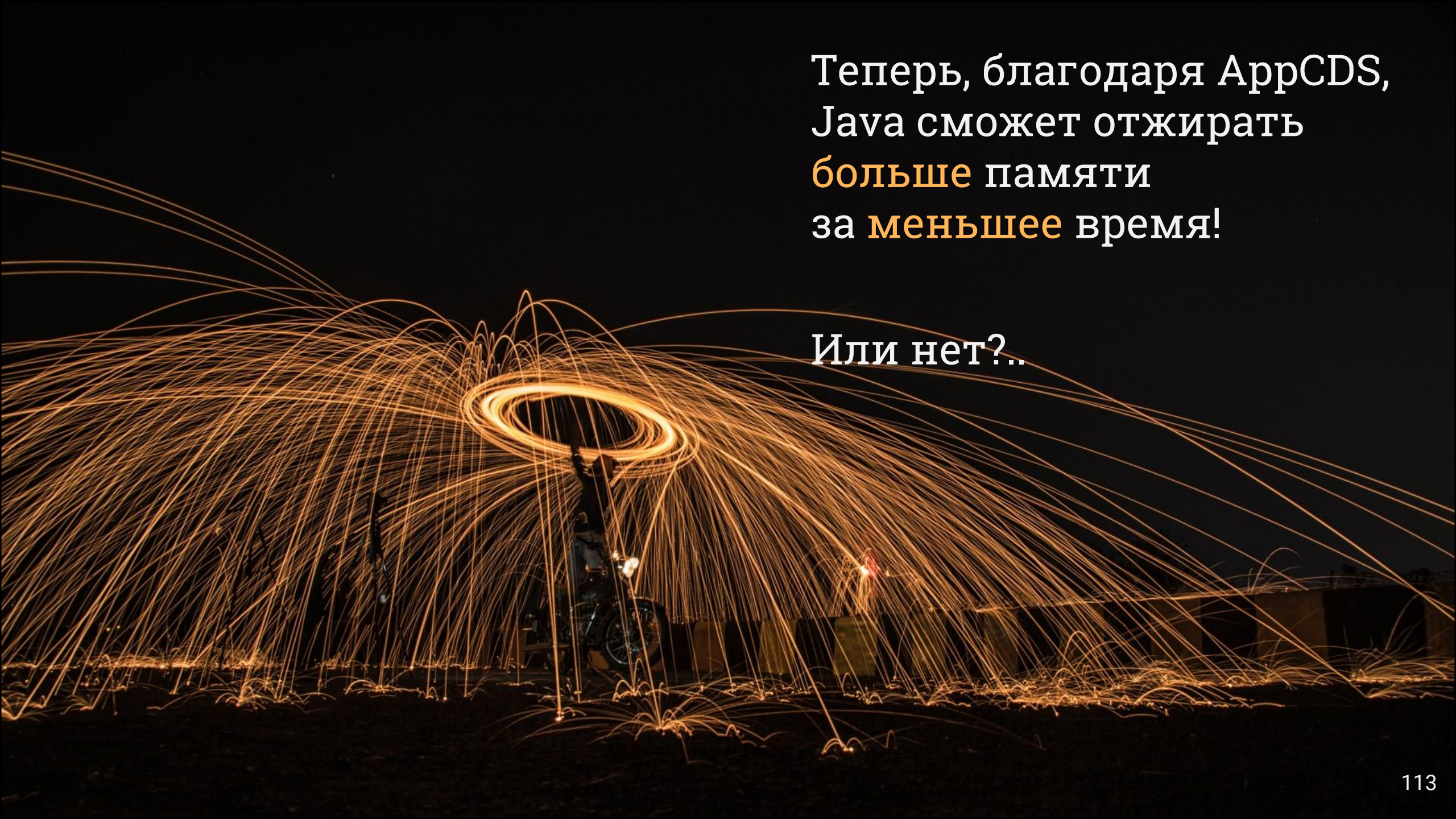
AppCDS

Время запуска микросервиса



Потребление памяти (RSS)





Теперь, благодаря AppCDS,
Java сможет отжирать
больше памяти
за меньшее время!

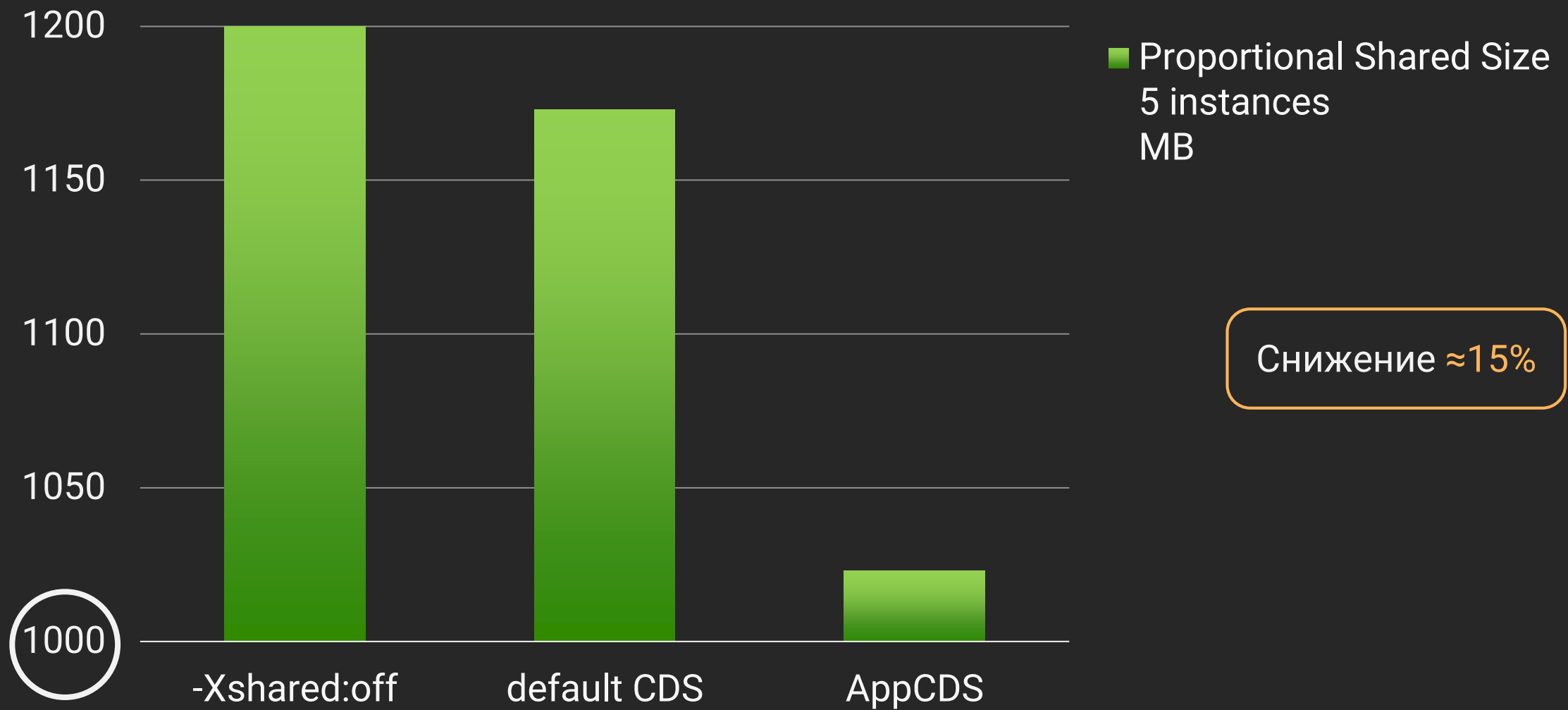
Или нет?..

Поправка на реальность

> … the memory footprint of a process …
> might appear to increase,
> because more pages are mapped
> to the process’s address space

https://docs.oracle.com/en/java/javase/11/vm/class-data-sharing.html

Потребление памяти (PSS)



Профит в том, что
та же память достанется
другим JVM процессам

[App]CDS: текущий расклад дел

– Требует много ручных действий

+ Улучшает и время старта, и потребление памяти
+ Легко применим для основных классов JDK
+ Активно развивается и обрастает практиками

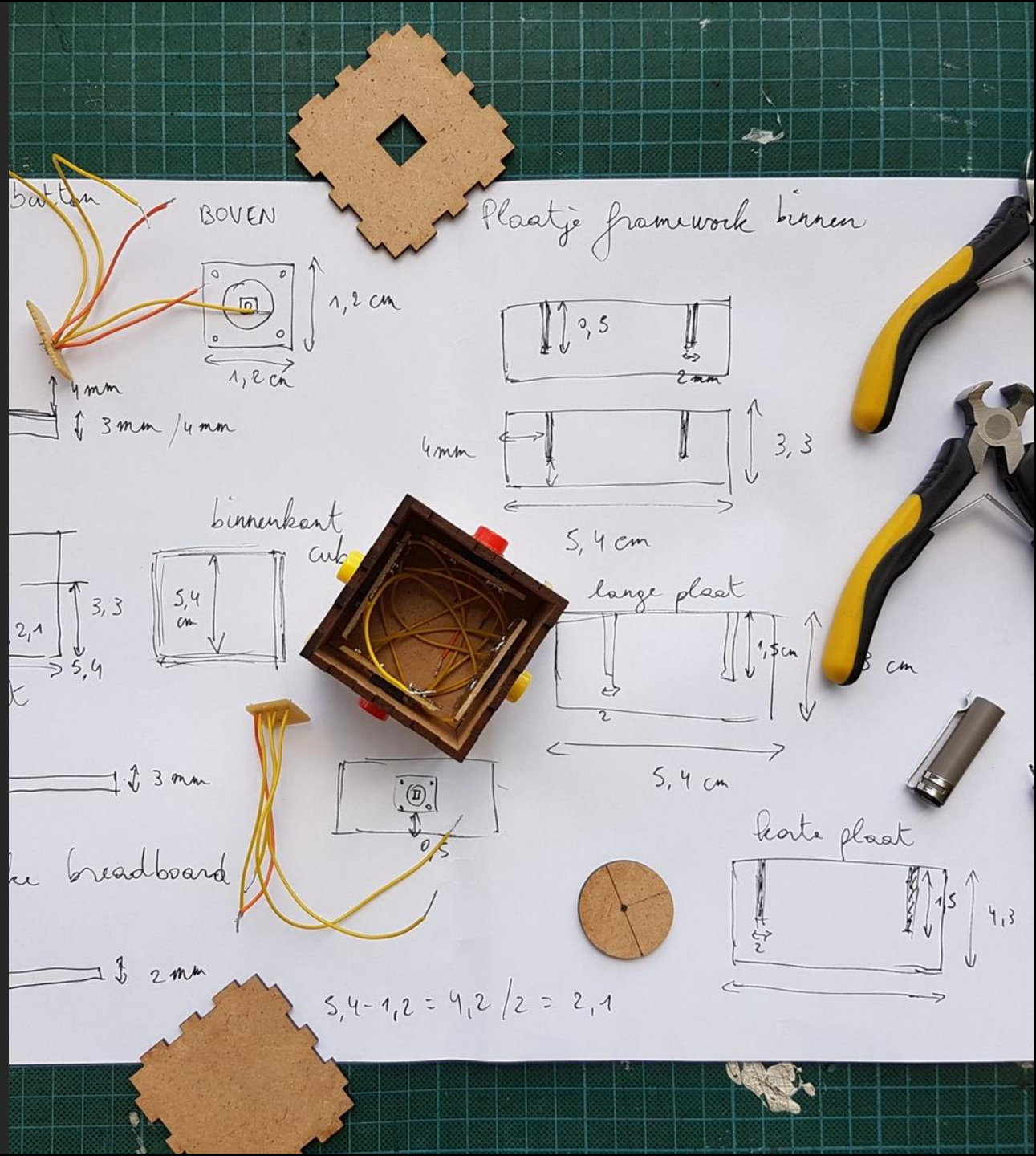
А если хочется подробнее

- AppCDS для приложений на Spring Boot: первый контакт*
  https://toparvion.pro/post/2019/10/appcds-with-spring-boot/

* *включает Dynamic CDS Archives из Java 13*

1. Погружение
2. Особенности перехода
   - Сборка проекта
   - Обновление Spring
   - Deployment
3. Новшества платформы
   - Single-File Programs
   - Class Data Sharing
   - JShell
4. Всплытие

Мы часто пишем
наброски кода
для проверки гипотез
и создания прототипов



Прототипы можно писать в:

- JUnit тестах
- IDEA Scratches
- ~~методах main() в прикладных классах~~
  (не надо так делать)
- *(свой вариант)*
- JShell *(не путать с “JS hell”)*

  JEP-222, Java 9
  Командная оболочка для Java


Write once,
~~run~~ test
everywhere!

Пример применения

Дано:

- Библиотека Spring Cloud Commons
- Метод `InetUtils#findFirstNonLoopbackAddress`
- Возвращает «внешний» сетевой адрес машины

Найти:

- Что вернет метод при вызове изнутри Docker-контейнера?

```java
public InetAddress findFirstNonLoopbackAddress() {
  InetAddress result = null;
  try {
    int lowest = Integer.MAX_VALUE;
    for (Enumeration<NetworkInterface> nics =
            NetworkInterface.getNetworkInterfaces();
         nics.hasMoreElements(); ) {
      NetworkInterface ifc = nics.nextElement();
      if (ifc.isUp()) {
        log.trace("Testing interface: " + ifc.getDisplayName());
        if (ifc.getIndex() < lowest || result == null) {
          lowest = ifc.getIndex();
        } else if (result != null) {
          continue;
        }
        if (!ignoreInterface(ifc.getDisplayName())) {
          for (Enumeration<InetAddress> addrs = ifc
              .getInetAddresses(); addrs.hasMoreElements(); ) {
            InetAddress address = addrs.nextElement();
            if (address instanceof Inet4Address
                && !address.isLoopbackAddress()
                && isPreferredAddress(address)) {
              log.trace("Found non-loopback interface: "
                  + ifc.getDisplayName());
              result = address;
            }
          }
        }
      }
```

InetUtils.java

Предсказуемо, не правда ли?

Поправка на реальность: Spring Boot

- JShell нужны классы приложения и библиотек
- Классы запакованы в über JAR/WAR
- JShell не умеет работать с такими архивами

Вариант решения

1. Распаковать über JAR/WAR
2. Составить значение опции `--class-path` для JShell
3. Запустить JShell с этой опцией
4. ~~Упахаться, повторяя это каждый раз~~
   Автоматизировать шаги 1-3

jshellw – обертка над JShell для запуска
с classpath'ом из Spring Boot Jar/War

https://github.com/toparvion/springboot-jshell-adapter

Запуск JShell через обертку

```
Created temp directory /tmp/…
Extracted 191 files from the archive to /tmp/…
Starting JShell with '/usr/bin/jshell --class-path …
|  Welcome to JShell -- Version 11.0.1
|  For an introduction type: /help intro

jshell>
```

```
./jshellw app.jar
```

jshellw – обертка над JShell для запуска
с classpath’ом из Spring Boot Jar/War

https://github.com/toparvion/springboot-jshell-adapter

Запуск прототипа в JShell

```
jshell> import ...InetUtils
jshell> import ...InetUtilsProperties
jshell> var utils = new InetUtils(new InetUtilsProperties())
jshell> var myAddress = utils.findFirstNonLoopbackAddress()
myAddress ==> /172.29.6.175
|  created variable myAddress : InetAddress
```

```
./jshellw app.jar
```

jshellw – обертка над JShell для запуска
с classpath'ом из Spring Boot Jar/War

https://github.com/toparvion/springboot-jshell-adapter

Другие применения JShell в контейнерах

Отладка операций с файлами

Работа со свойствами окружения

Тестирование специфичных библиотек (API)

1. Погружение
2. Особенности перехода
   - Сборка проекта
   - Обновление Spring
   - Deployment
3. Новшества платформы
   - Single-File Programs
   - Class Data Sharing
   - JShell
4. Всплытие

Обновлять Java –
как делать ремонт:
не обязательно,
сложно, но полезно

Версии инструментов с поддержкой Java 11

- Сборка:
  - Maven 3.5
  - Gradle 5.1
- Spring:
  - Boot 2.1
  - Framework 5.1
  - Cloud Greenwich
  - Lombok 1.18

Java 8 (пока) никуда не уходит ☺

Когда ждать следующей мажорной версии?

> There are no “major releases” per se any more …
> Instead there is a steady stream
> of “feature releases.”

https://blogs.oracle.com/java-platform-group/update-and-faq-on-the-java-se-release-cadence

Режимы обновления на любой вкус

- «На каждый чих»
  По мере появления фиксов
- При выходе feature releases
  2 раза в год (март, сентябрь)
- При выходе LTS releases
  1 раз в 3 года (2014, 2018, 2021)

Диетологи советуют
питаться часто,
но по чуть-чуть

Слайды

Владимир Плизгá
ЦФТ
@toparvion
Toparvion
toparvion.pro

Q & A