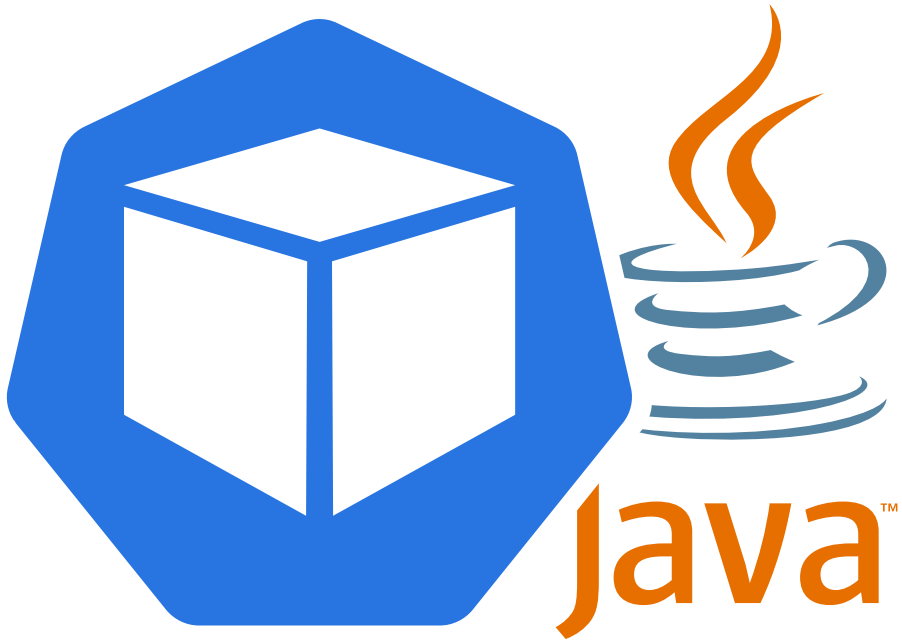
Профилирование JVM в Kubernetes : три больших шага

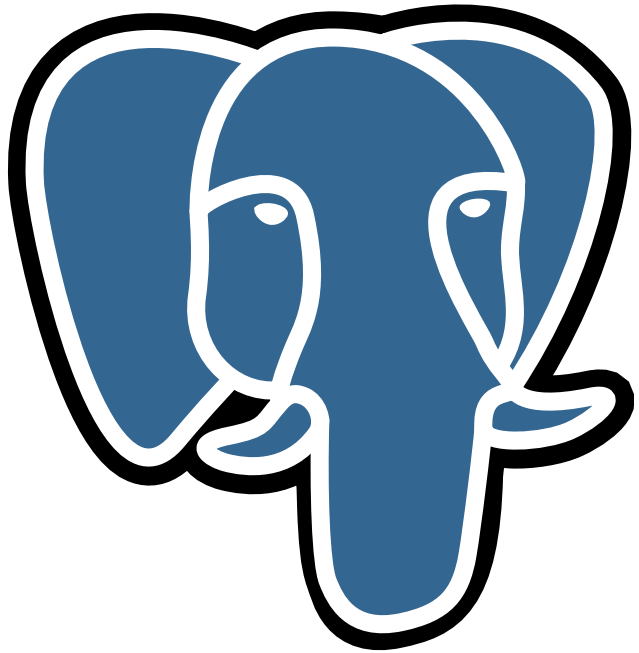
Смирнов Вячеслав, 2021


Исследую и создаю результаты нагрузки в ВТБ, ДБО: vtbbo.ru
И развиваю чат @qa_load

100 JVM работающих друг с другом и базой

На тестовом стенде





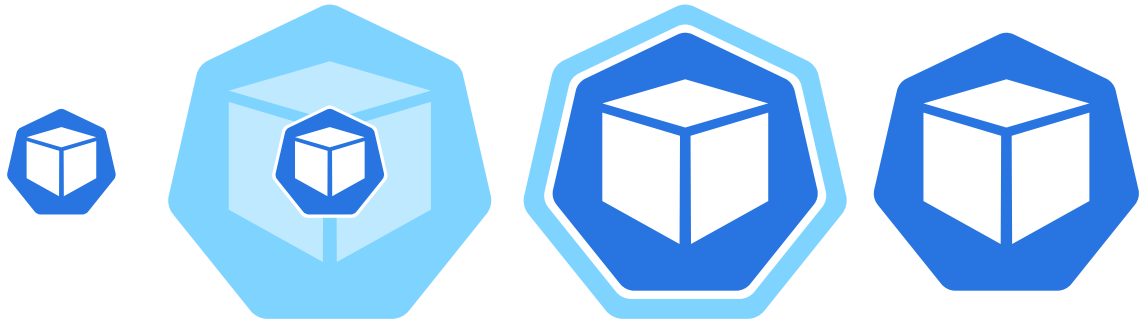
Особенности профилирования JVM в Kubernetes

Особенности Kubernetes

Профайлер

Выделение ресурсов для нужд профилирования

Особенности Kubernetes

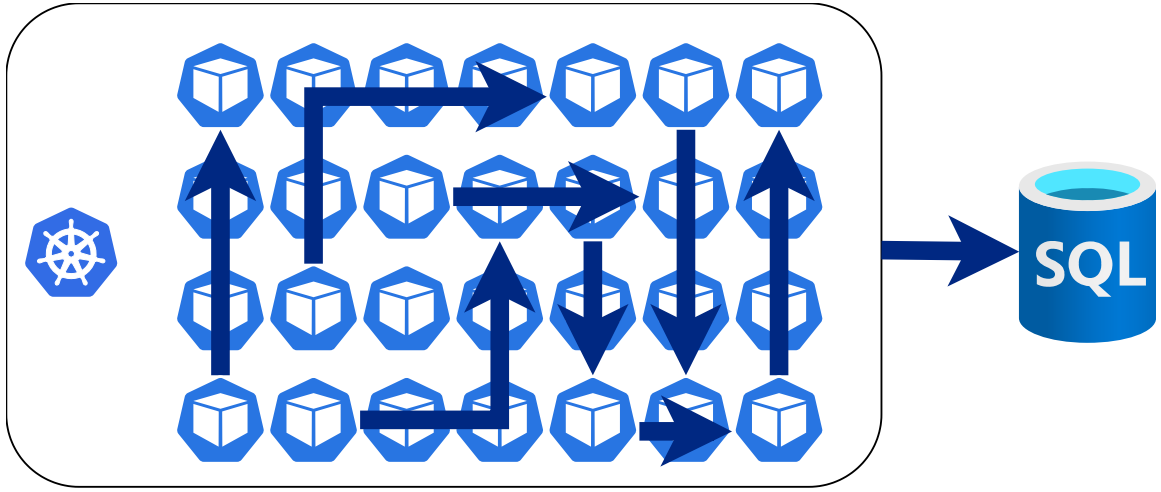


Как выполнять анализ: от потоков к коду

Анализ

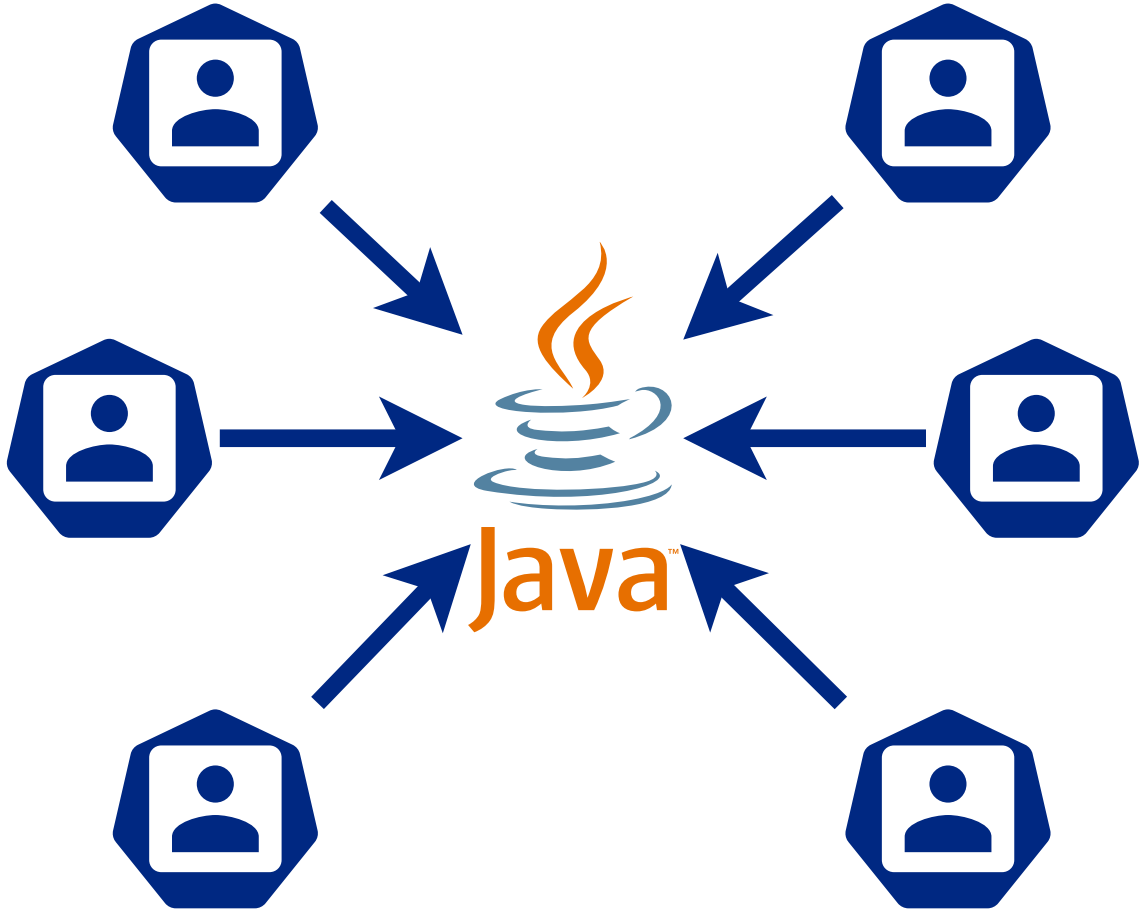
Анализ взаимодействия микросервисов

Анализ



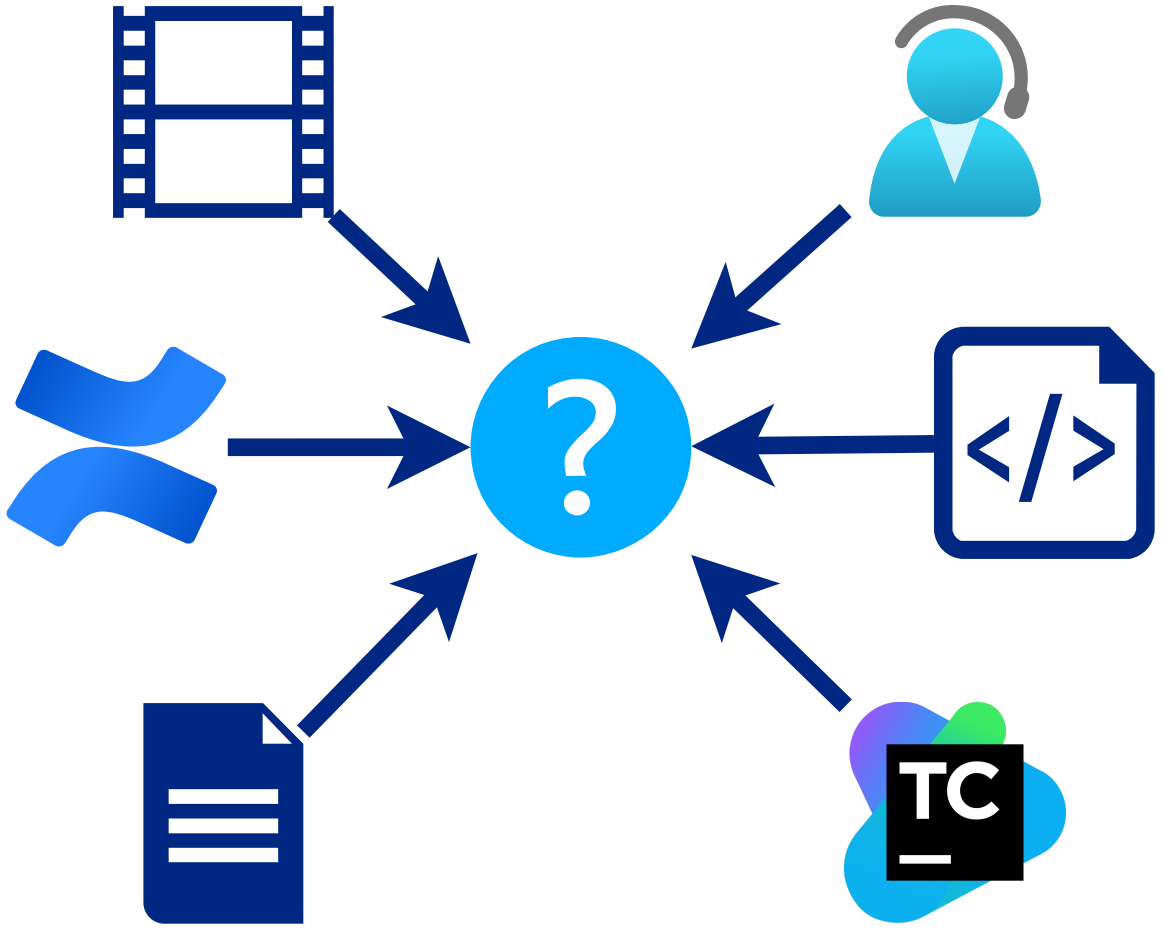
Стандартизация процесса в большой команде

Масштабирование



Обмен знаниями, передача опыта, автоматизация

Масштабирование



Особенности профилирования JVM в Kubernetes

Особенности Kubernetes

Профайлер

Бизнес-отчет по тестированию производительности

Особенности Kubernetes

Нагрузочное

Стабильности

Масштабируемости

Бизнес-отчет по тестированию производительности

Особенности Kubernetes

Сколько реплик?

Какие настройки?

Ресурсы

Какие настройки?

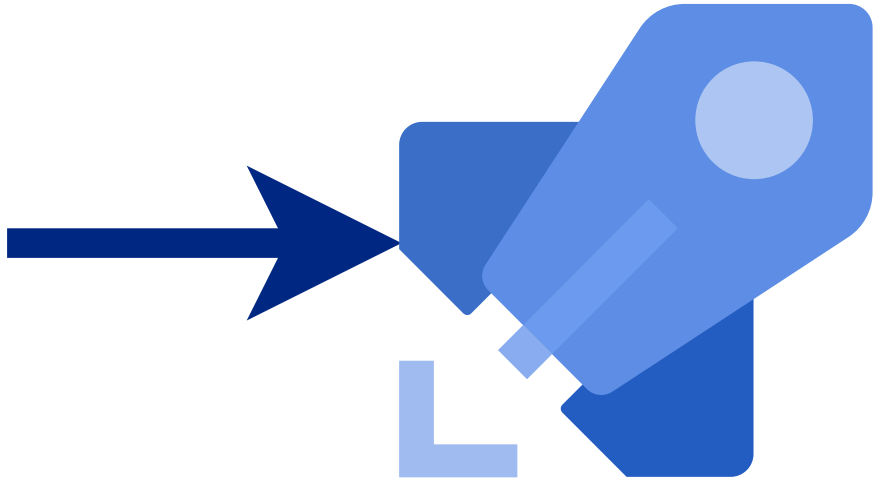
Сколько реплик?

Ресурсы

Какие настройки?

Сколько реплик?

Ресурсы



Ускорение

Фиксируем реплики

Фиксируем настройки

Мониторинг

Ресурсы

Ускорение

Немного реплик

Какие настройки?

Ресурсы

Профайлер

Ускорение

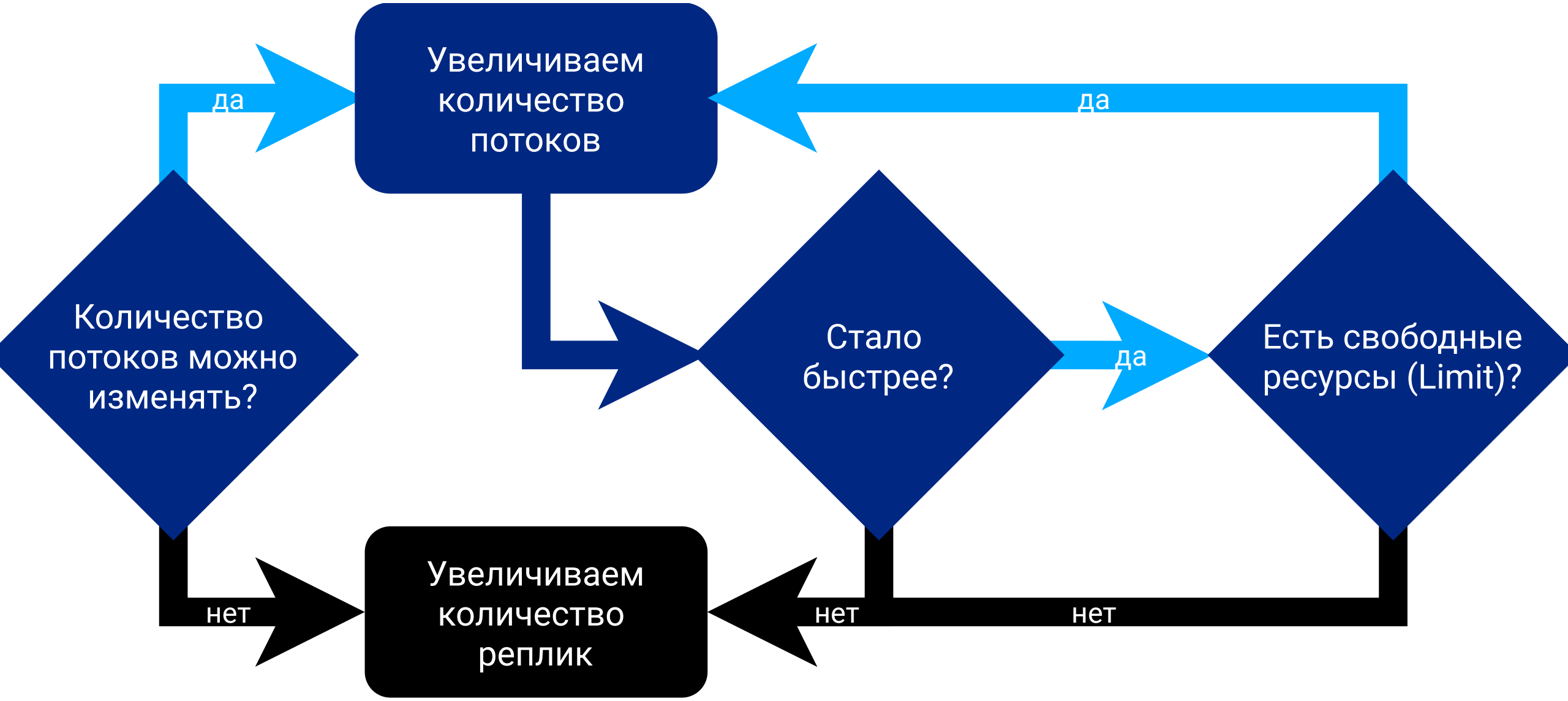
Увеличиваем количество потоков или реплик?

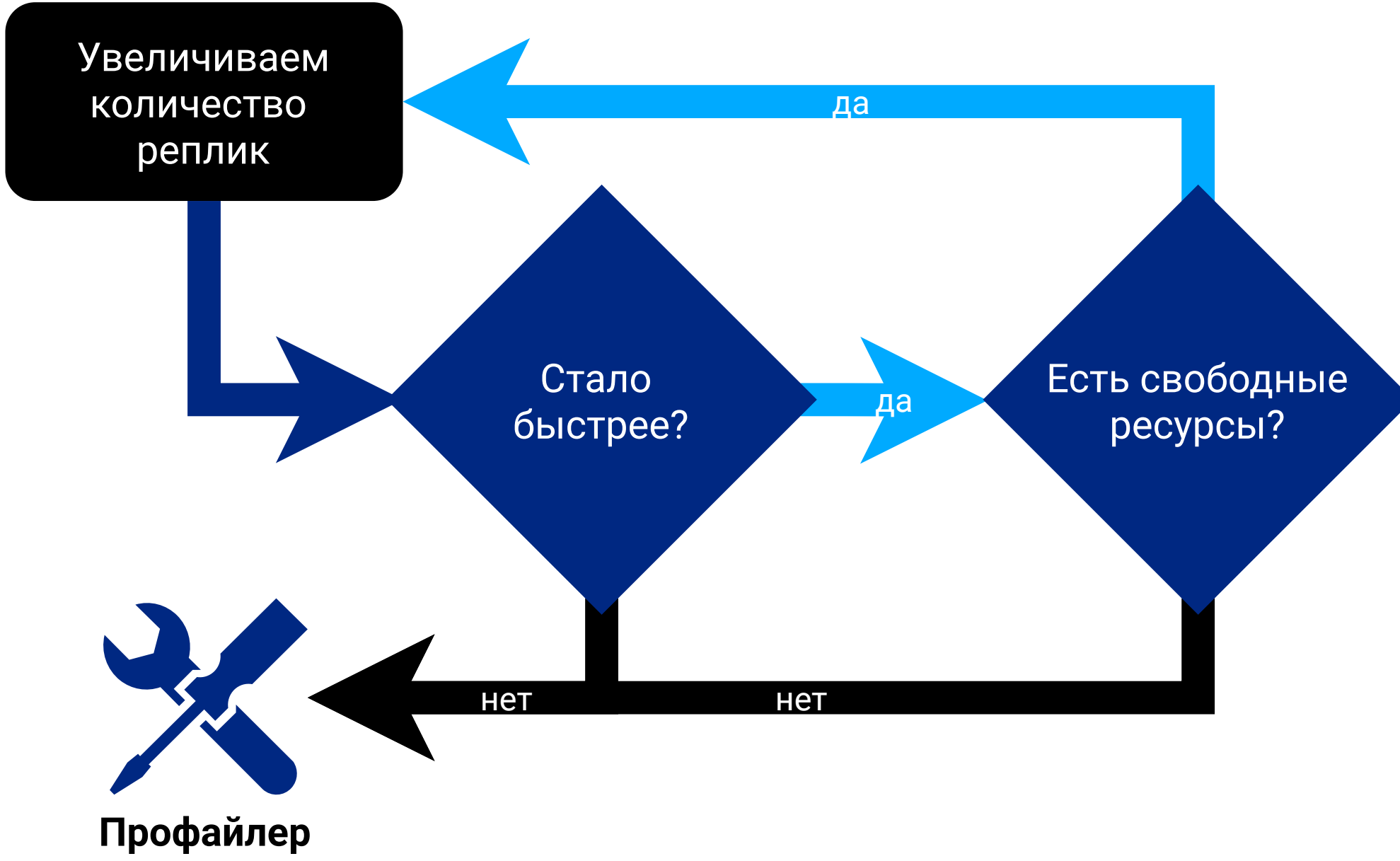
Особенности Kubernetes

Сколько реплик?

Какие настройки?

Ресурсы


Увеличиваем количество потоков
да
Количество потоков можно изменять?
Стало быстрее?
да
Есть свободные ресурсы (Limit)?
да
нет
Увеличиваем количество реплик
нет
нет


Увеличиваем количество реплик
да
Стало быстрее?
да
Есть свободные ресурсы?
нет
нет

Профайлер

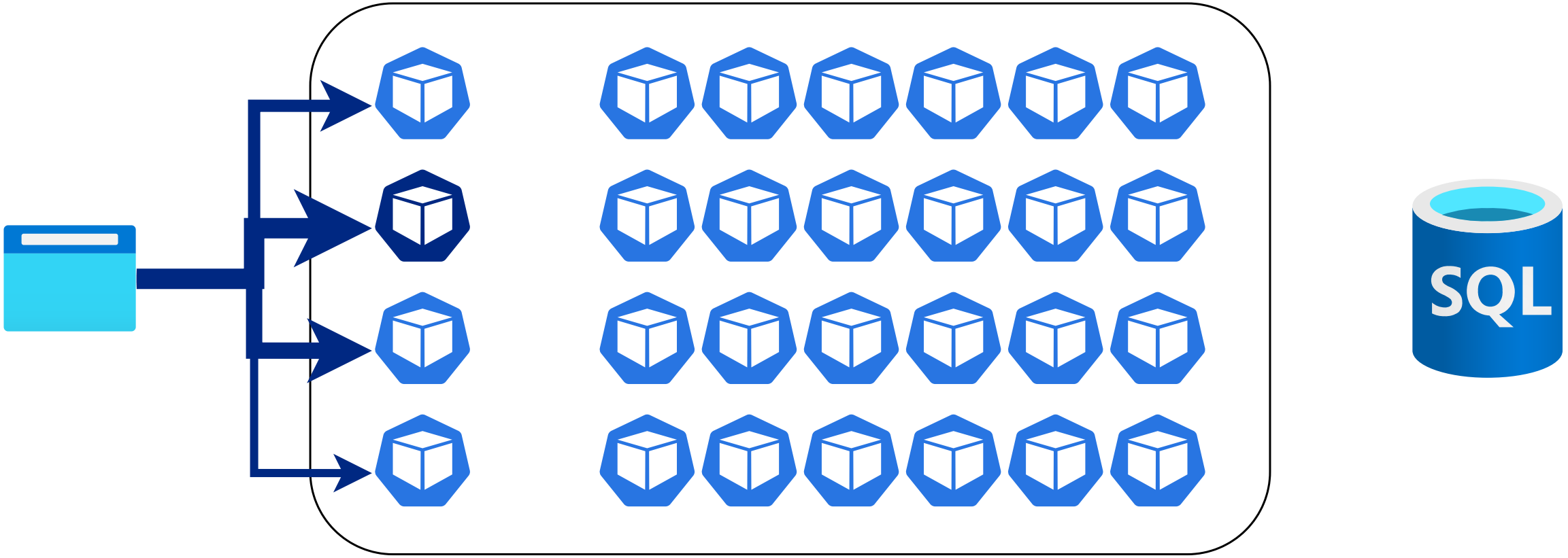
Когда JVM профайлер не нужен и чем его заменить

Особенности Kubernetes

Профайлер

Профилирование в цикле тестирования

1. Регрессионный тест производительности (JMeter/Gatling)
2. Статистика по логам, детали по ERROR (Kibana/Grafana, grep)
3. Бизнесс-метрики производительности (Яндекс.Метрика, ...)
4. Статистика по запросам (Zipkin, Jaeger)
5. Мониторинг системных метрик (CPU, Memory, IO)
6. Прикладной мониторинг (JVM MBean, SQL stat)
7. **Профилирование JVM**
8. perf, strace, lsof, ...


SQL

Профайлер

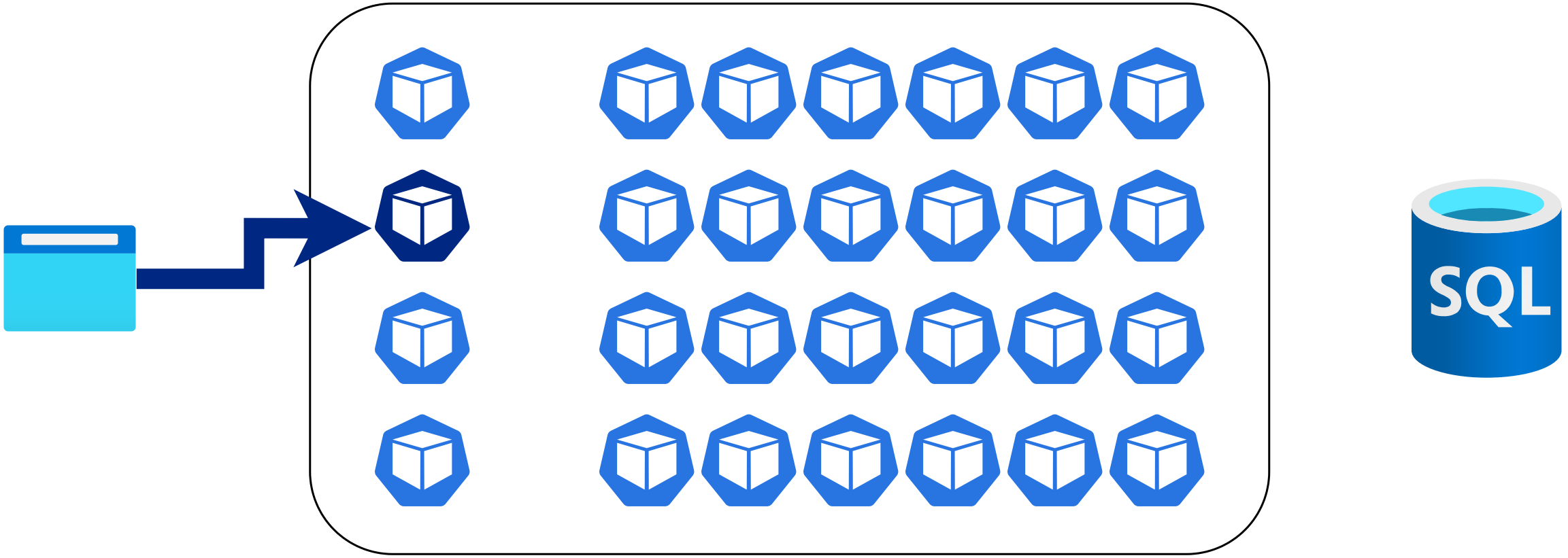
ZIPKIN

JAEGER

Отчет по нагрузке

Бизнес-метрики

Статистика по логам


SQL

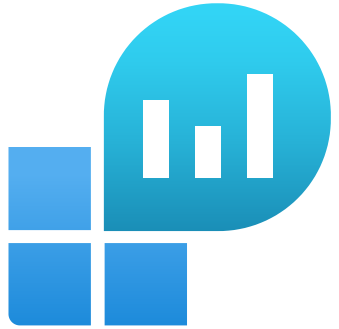
Профайлер

ZIPKIN

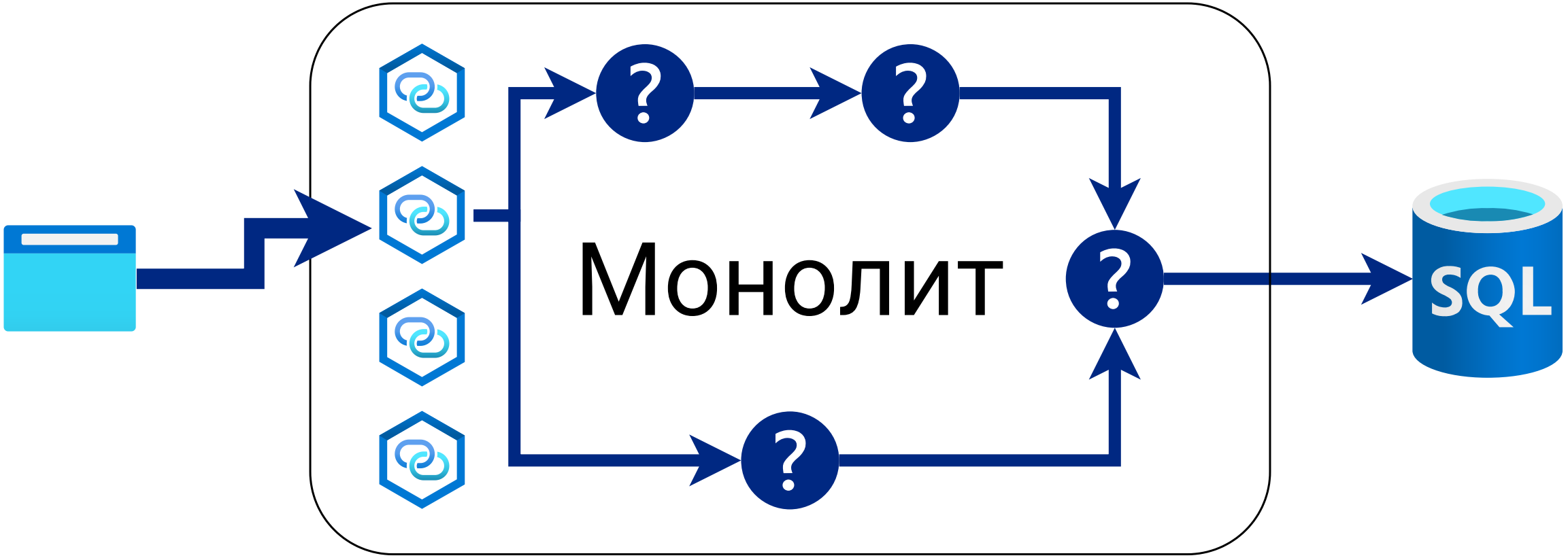
JAEGER

Отчет по нагрузке

Бизнес-метрики



Статистика по логам

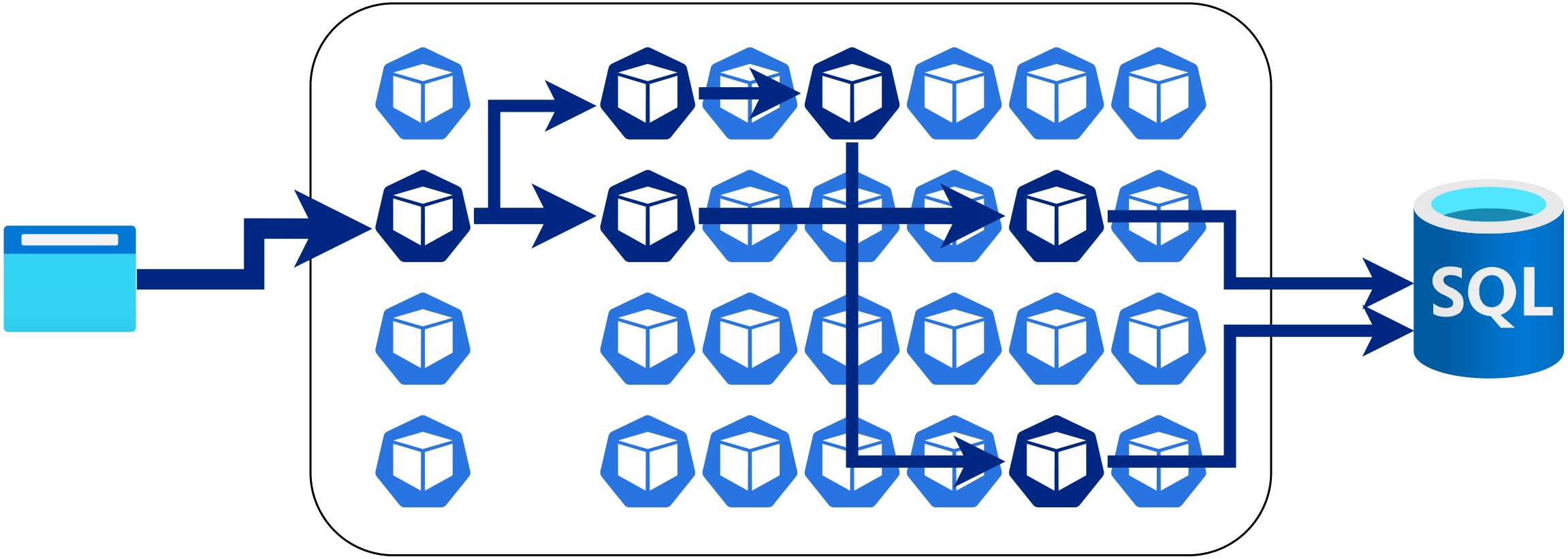

Монолит
SQL

Профайлер

Отчет по нагрузке

Бизнес-метрики

Статистика по логам


SQL

Профайлер

Отчет по нагрузке

Бизнес-метрики

Статистика по логам

Влияние количества потоков сервиса на профилирование

Особенности профайлеров

Профайлер

Если нужно посчитать процент активности

Инструмент	Примечание
JFR (Java Fligth Recorder)	
SJK (Swiss Java Knife)	
AsyncProfiler	
JVisualVM	в режиме семплирования
JProfiler	в режиме семплирования
YourKit Java Profiler	в режиме семплирования

Stack Trace

Live Threads 21:10:47

Search the table

Thread Name	Thread State	Blocked Count	Total CPU Usage	Deadlocked	Allocated Memory
Thread Group 1-1	RUNNABLE	1	12,4 %	false	293 GiB
RMI TCP Connection(2)-127.0.0.1	RUNNABLE	415	0,325 %	false	119 MiB
RMI TCP Connection(8)-127.0.0.1	TIMED_WAITING	80	0,13 %	false	40,8 MiB
StandardJMeterEngine	TIMED_WAITING	1	0 %	false	50,4 KiB
NanoOffset	TIMED_WAITING	0	0 %	false	0 B

Stack Traces for Selected Threads

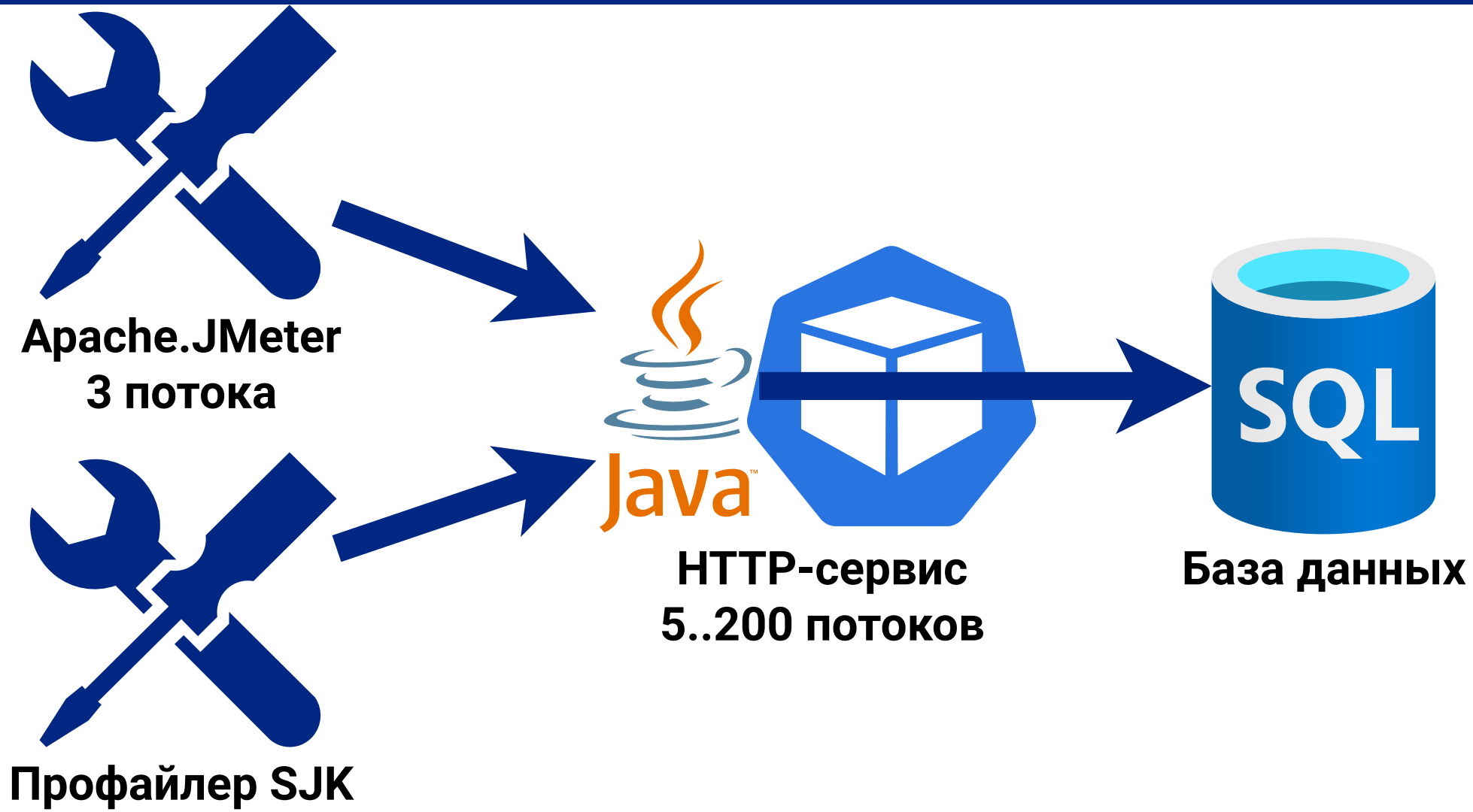
Stack traces for selected threads 21:10:47

- Thread Group 1-1 [63] (RUNNABLE)
  - java.util.concurrent.ConcurrentHashMap.putVal line: not available
  - java.util.concurrent.ConcurrentHashMap.put line: not available
  - org.apache.http.protocol.BasicHttpContext.setAttribute line: 75
  - org.apache.http.protocol.HttpCoreContext.setAttribute line: 105
  - org.apache.jmeter.protocol.http.sampler.HTTPHC4Impl.setupClient line: 1050
  - org.apache.jmeter.protocol.http.sampler.HTTPHC4Impl.sample line: 613
  - org.apache.jmeter.protocol.http.sampler.HTTPSamplerProxy.sample line: 66
  - org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.sample line: 1281
  - org.apache.jmeter.protocol.http.sampler.HTTPSamplerBase.sample line: 1270
  - org.apache.jmeter.threads.JMeterThread.doSampling line: 630
  - org.apache.jmeter.threads.JMeterThread.executeSamplePackage line: 558

Метод	Семплы
HashMap.putVal	4
HashMap.put	5
HttpContext.setAttribute:75	7
CoreContext.setAttribute:105	7
Impl.setupClient:1050	8
Impl.sample:613	8
Proxy.sample:613	8
...	**9**

Метод	Семплы
HashMap.putVal	49%
HashMap.put	57%
HttpContext.setAttribute:75	77%
CoreContext.setAttribute:105	79%
Impl.setupClient:1050	80%
Impl.sample:613	81%
Proxy.sample:613	81%
...	**90%**

Соберем тестовый стенд



Результаты замеров длительности

Средняя длительность для	мсек
HTTP-запрос	100
SQL-запрос	50
Java-метод	10

Профилирование надо проводить под нагрузкой

1000 Java-методов * 3 потока * 10 мс / 11 мс = 2 700 семплов

Средняя длительность для	мсек	частота (в сек)
HTTP-запрос	100	
SQL-запрос	50	
Java-метод	**10**	
SJK (**3 активных потока** + 120 спящих)	**11**	90

Профилирование надо проводить под нагрузкой

1000 HTTP запросов * 3 потока * 100 мс / 11 мс = 27 000 семплов

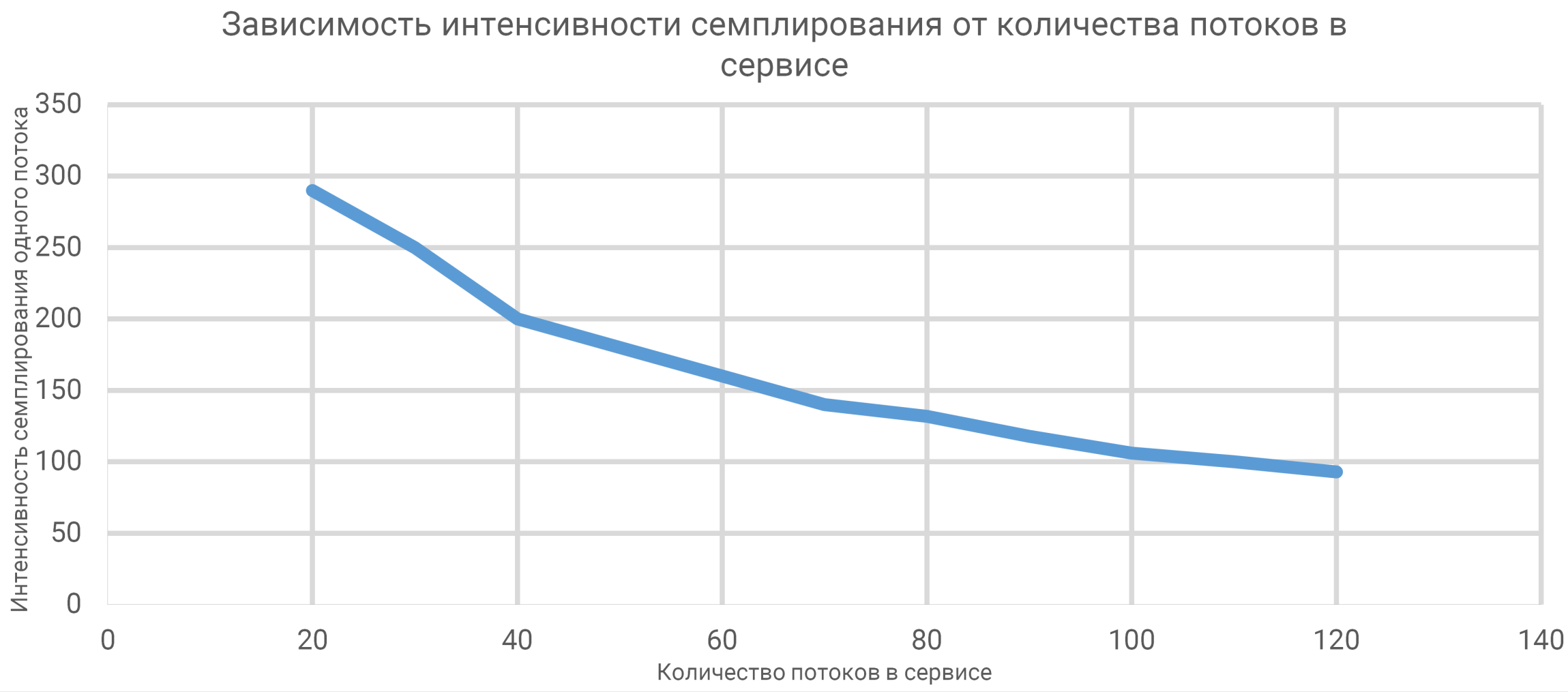
Средняя длительность для	мсек	частота (в сек)
HTTP-запрос	**100**	
SQL-запрос	50	
Java-метод	10	
SJK (**3 активных потока** + 120 спящих)	**11**	90

Меньше спящих потоков - выше точность

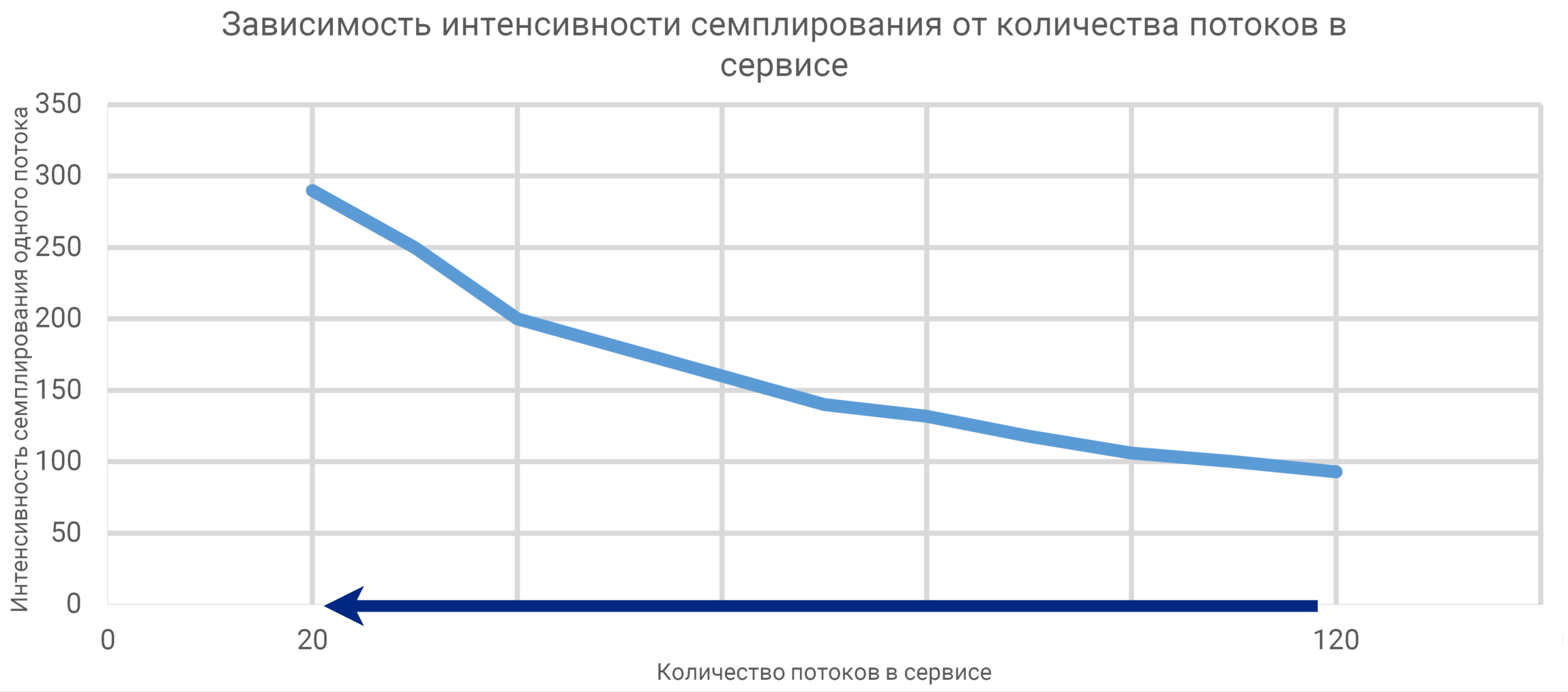
1000 HTTP запросов * 3 потока * 100 мс / 3.5 мс = 85 000 семплов

Средняя длительность для	мсек	частота (в сек)
HTTP-запрос	**100**	
SQL-запрос	50	
Java-метод	10	
SJK (3 активных потока + 120 спящих)	11	90
SJK (**3 активных потока** + 20 спящих)	**3.5**	290

Меньше спящих потоков - выше точность

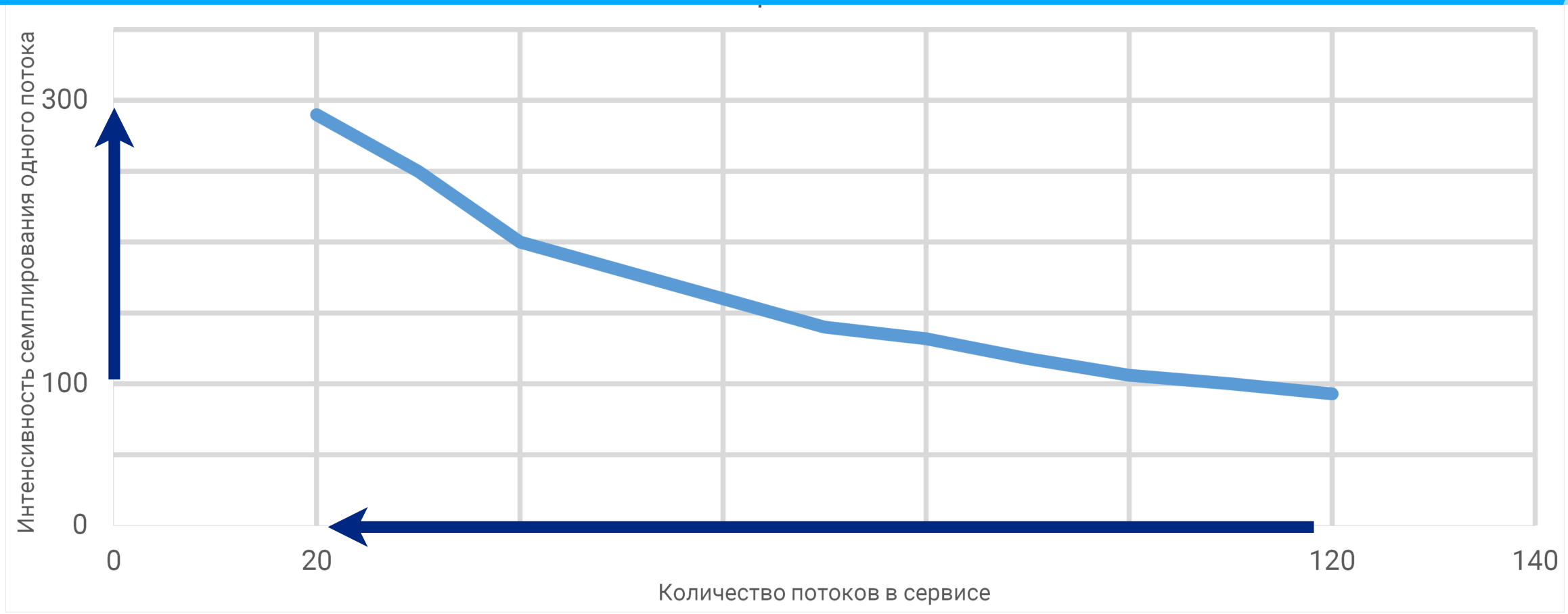


Снижаем server.tomcat.max-threads с 100 до 5



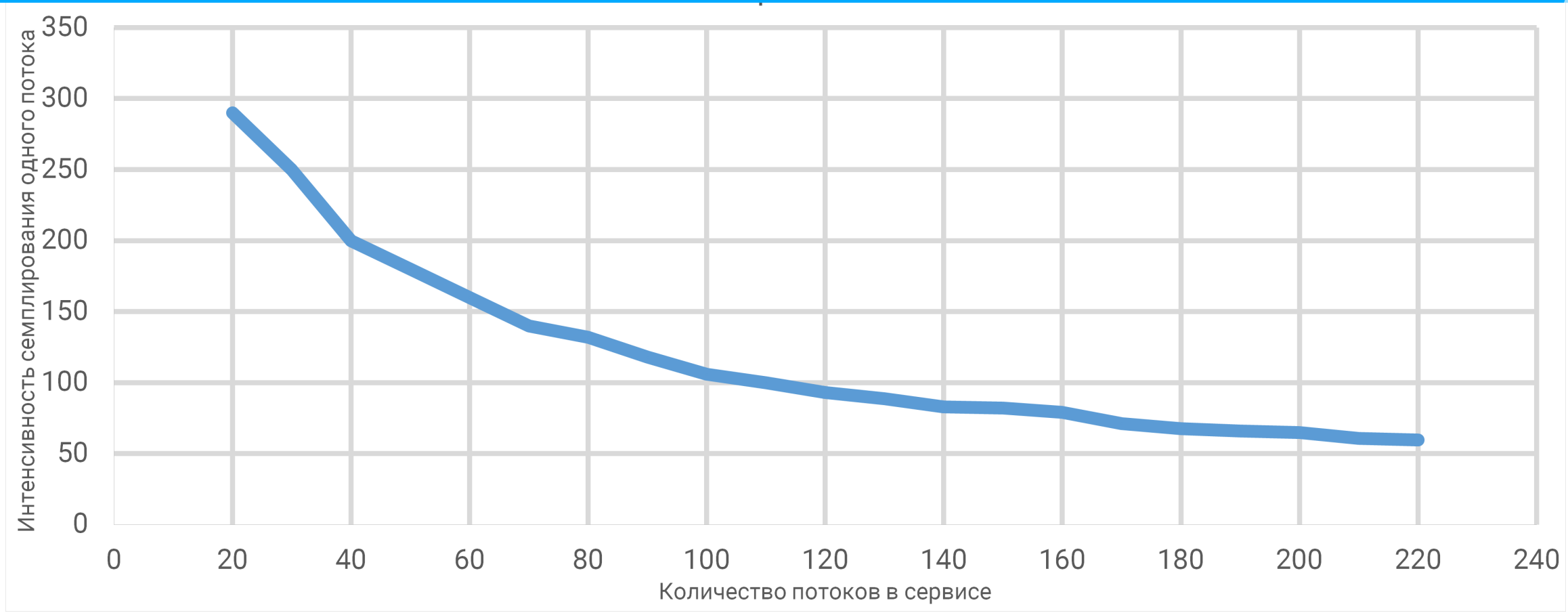
Снижаем server.tomcat.max-threads с 100 до 5

Повышается точность профилирования в 3 раза



Снижаем server.tomcat.max-threads с 200 до 5

Повышается точность профилирования в 5 раз



Частота семплирования в SJK (по факту)

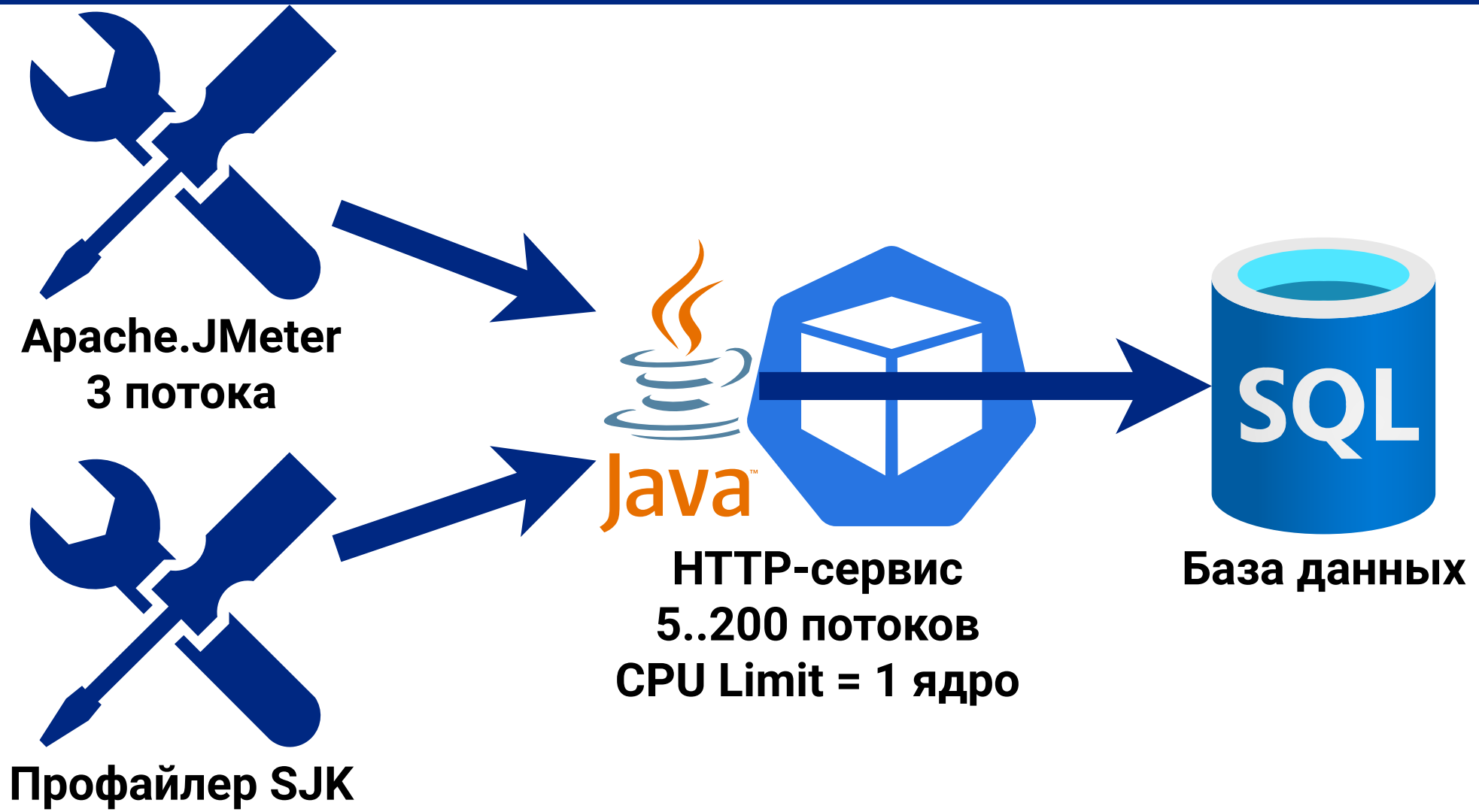
```
# Запуск профилирования с максимальной интенсивностью
$ java -jar ./sjk-0.17.jar stcap -s localhost:9010 \
  -o "result.sjk"

# Отображение фактической интенсивности
$ java -jar ./sjk-0.17.jar ssa -f "result.sjk" \
  --thread-info --numeric -co -si FREQ
Freq.
59.5
59.5
59.5
59.5
59.5
59.5
```

Влияние CPU Limit на профилирование

Особенности Kubernetes

Профайлер

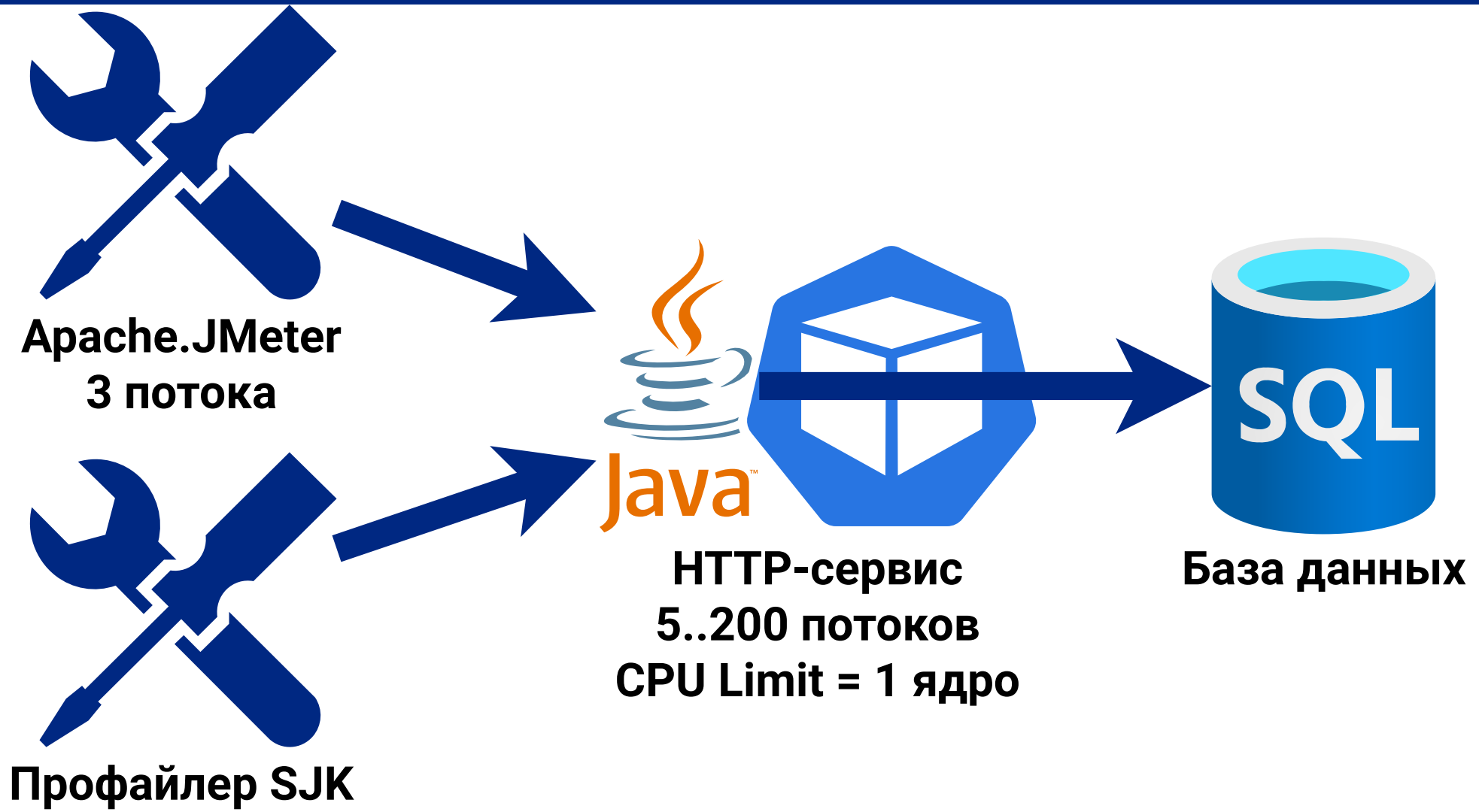
Соберем тестовый стенд с малым CPU Limit



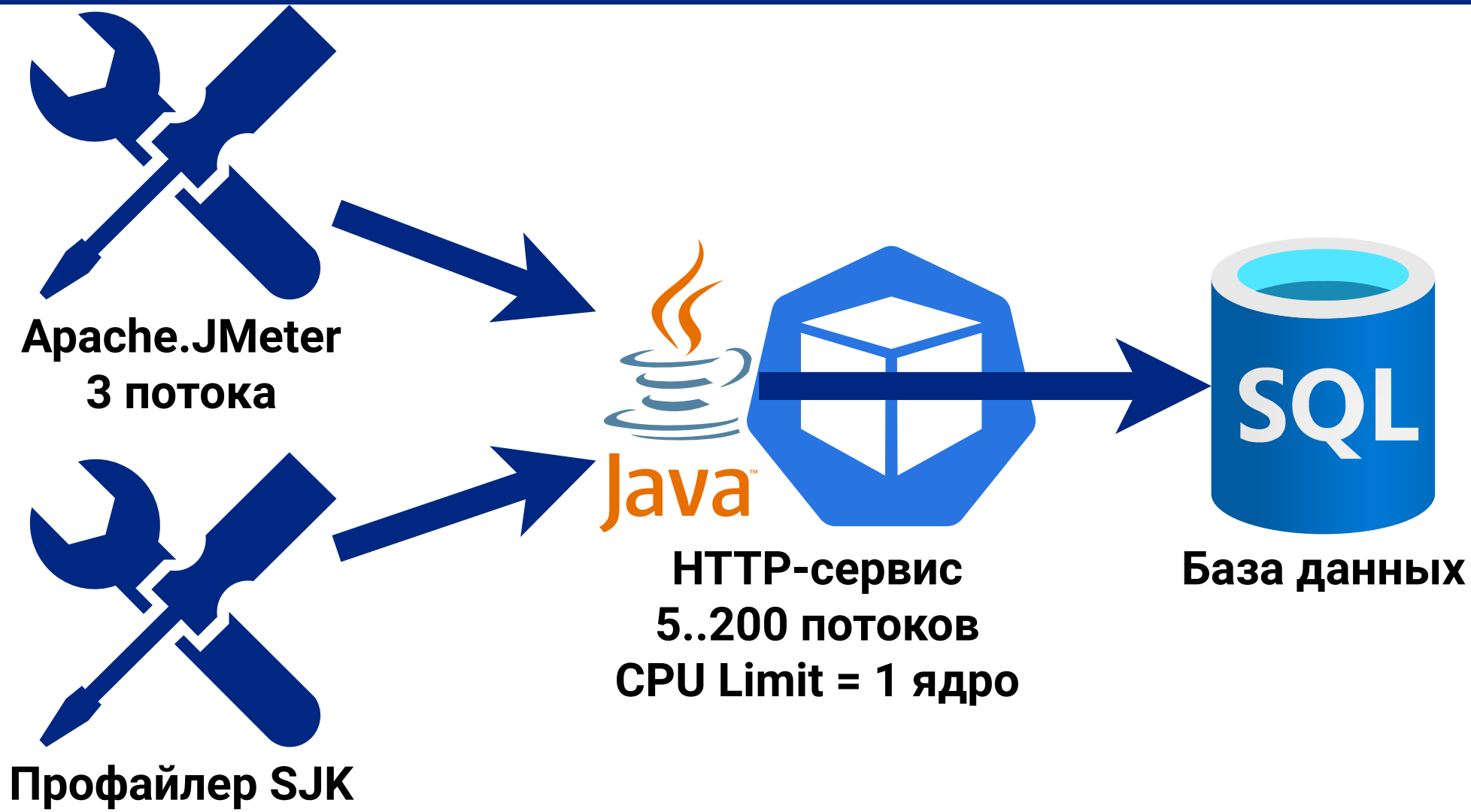
Точность профилирования снизилась в 10-13 раз

Средняя длительность для	мсек	частота (в сек)
SJK (когда было достаточно CPU)	11	90
SJK (нехватка CPU, малый CPU Limit)	**143**	7

Сам SJK потребляет до 0,4 ядра при 200 потоках



Задавая лимит CPU помни о накладных расходах



Комфортная интенсивность: раз в 100 мсек

Для большей точности можно собирать метрики дольше

```
# Профилирование с заданной интенсивностью
$ java -jar ./sjk-0.17.jar stcap \
  -s localhost:9010 \
  -o "result.sjk" \
  --sampler-interval 100ms \
  --timeout 10m
```

Влияние Memory Limit на профилирование

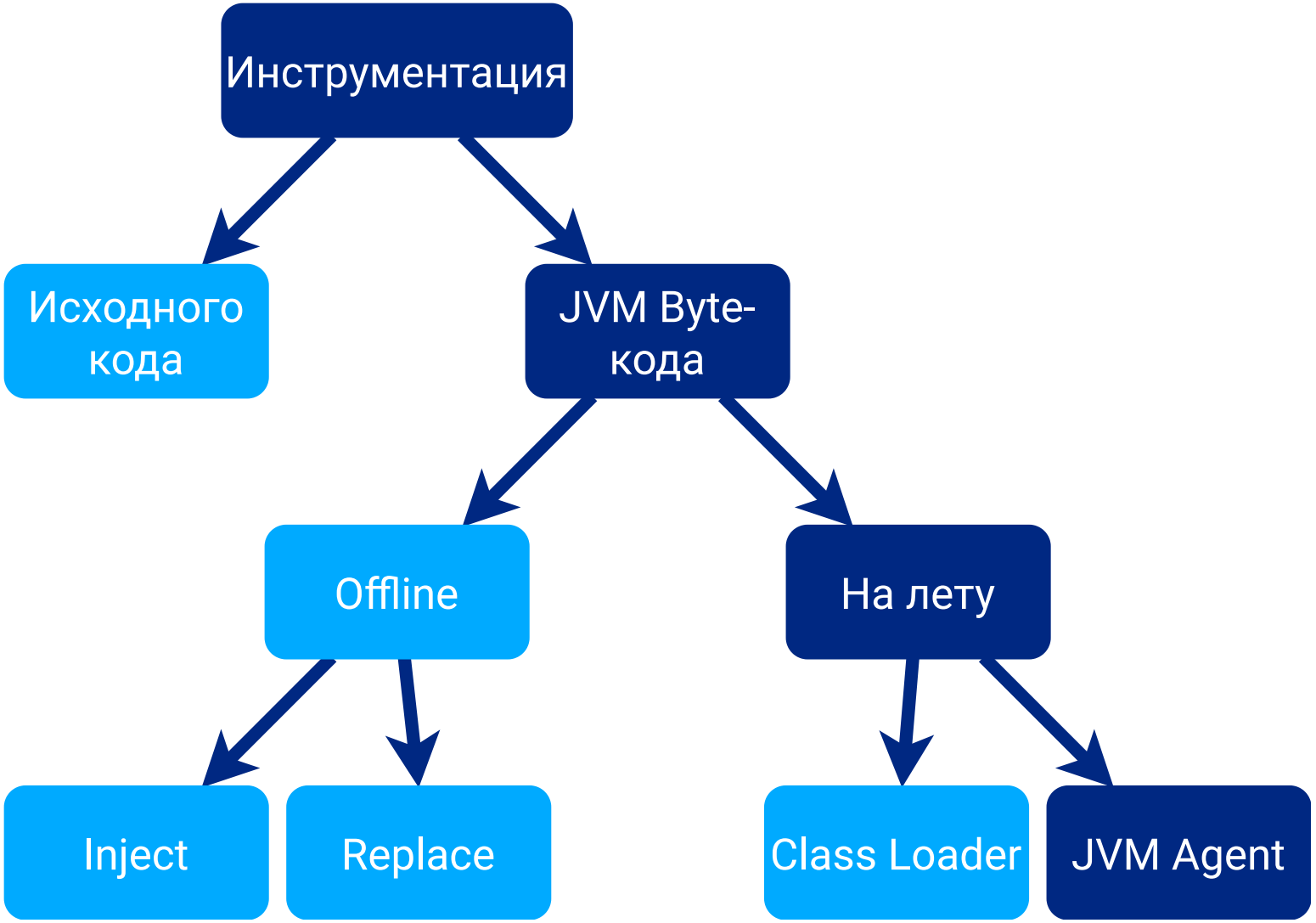
Особенности Kubernetes

Профайлер

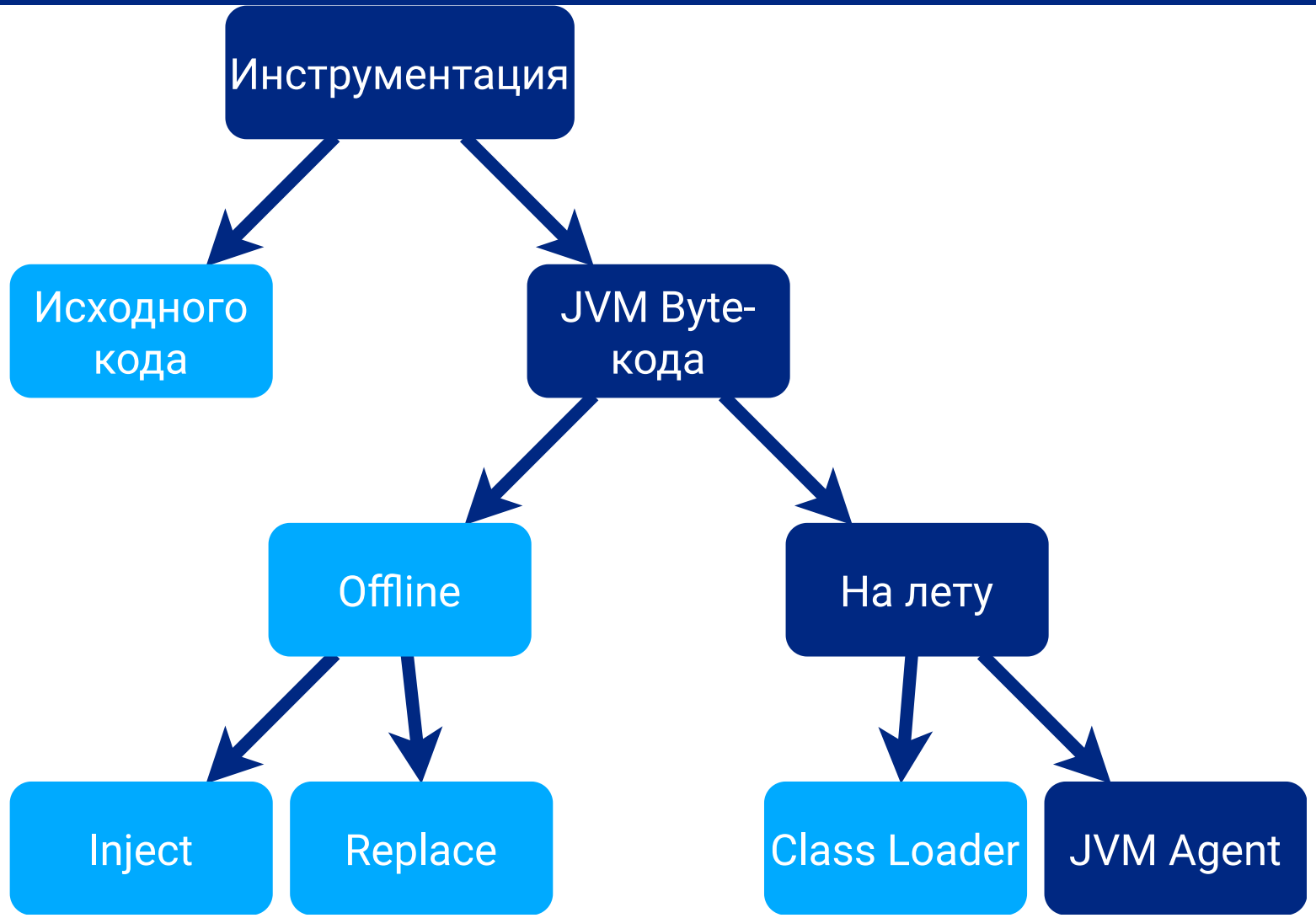
Процент активности, длительность и количество

Их позволяет оценить инструментирующее профилирование

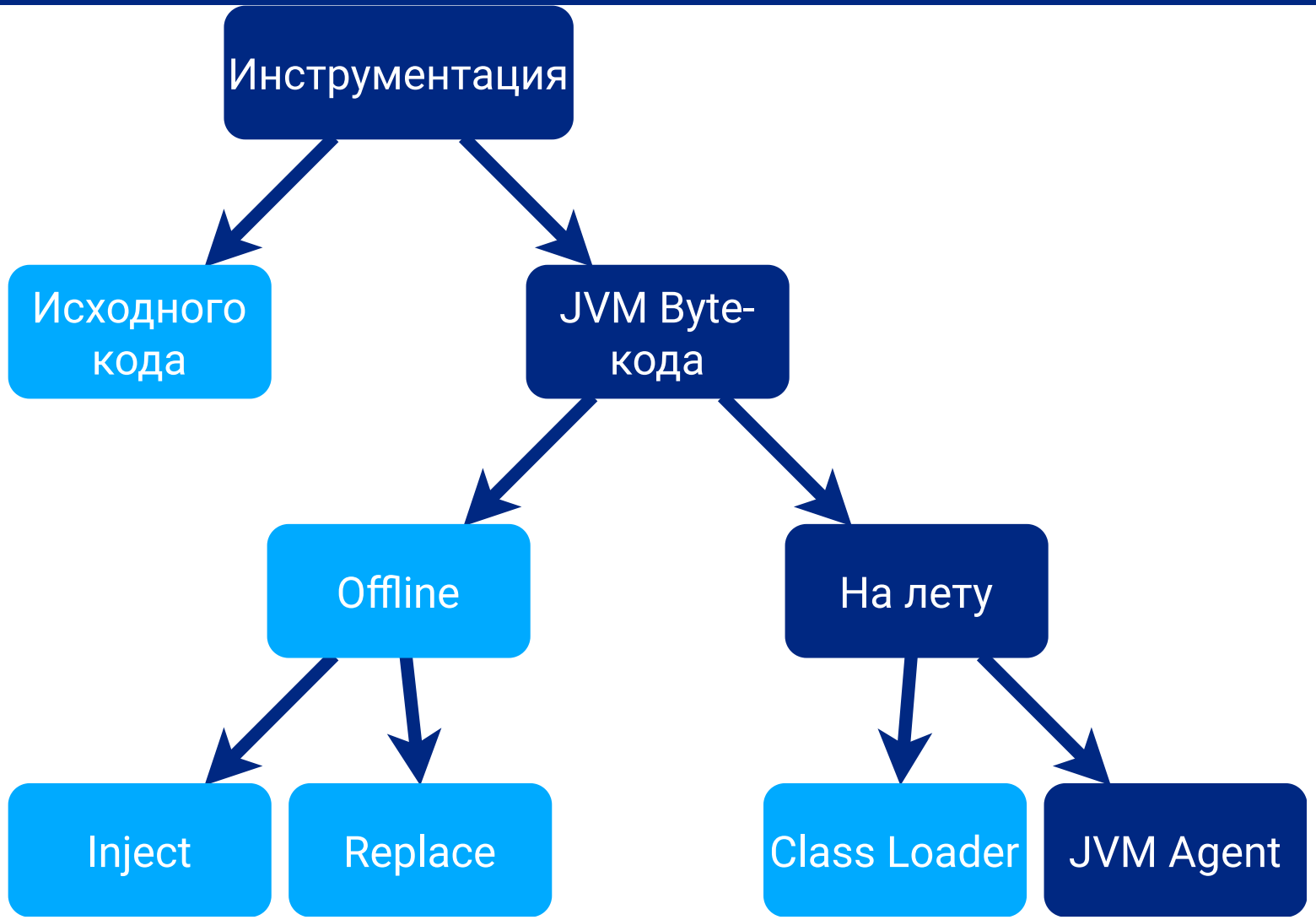
Инструмент	Примечание
JVisualVM	в режиме Startup Profiler
JProfiler	в режиме инструментации
YourKit Java Profiler	в режиме инструментации


Инструментация
Исходного кода
JVM Byte-кода
Offline
На лету
Inject
Replace
Class Loader
JVM Agent

Инструментация на лету расходует HEAP



Инструментация новых объектов расходует CPU



При добавлении JvmAgent для инструментации

Стоит увеличить HEAP Xmx, CPU и Memory Limit

Было

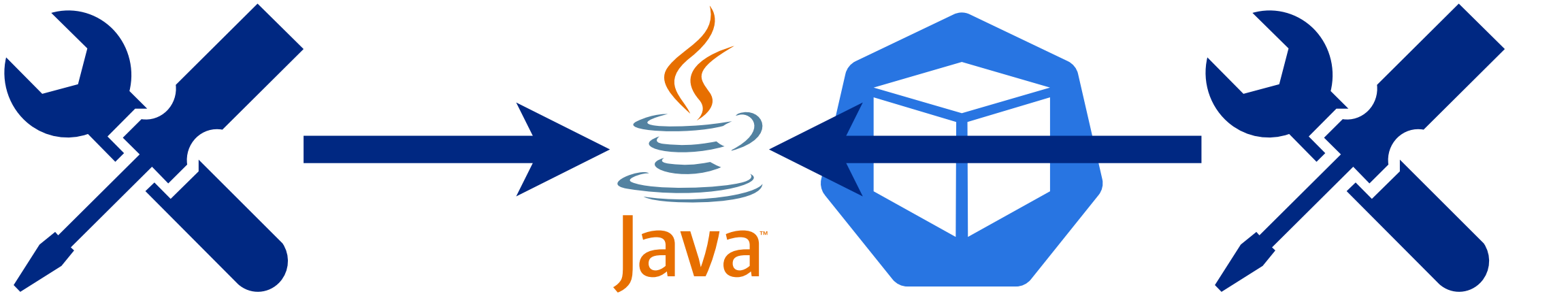
```
limits:
    memory: 5Gi
    cpu: 3
requests:
    memory: 1Gi
    cpu: 0.5
# JAVA_MAX_MEM_RATIO=50
# Xmx = 2.5Gi
```

Стало +1 на всё

```
limits:
    memory: 7Gi
    cpu: 4
requests:
    memory: 2Gi
    cpu: 1.5
# JAVA_MAX_MEM_RATIO=50
# limits.memory += 2Gi
```

При добавлении JvmAgent для инструментации

Не стоит подавать большую нагрузку, достаточно ручных запросов



CURL, WGET, браузер

HTTP-сервис Xmx +1Gi

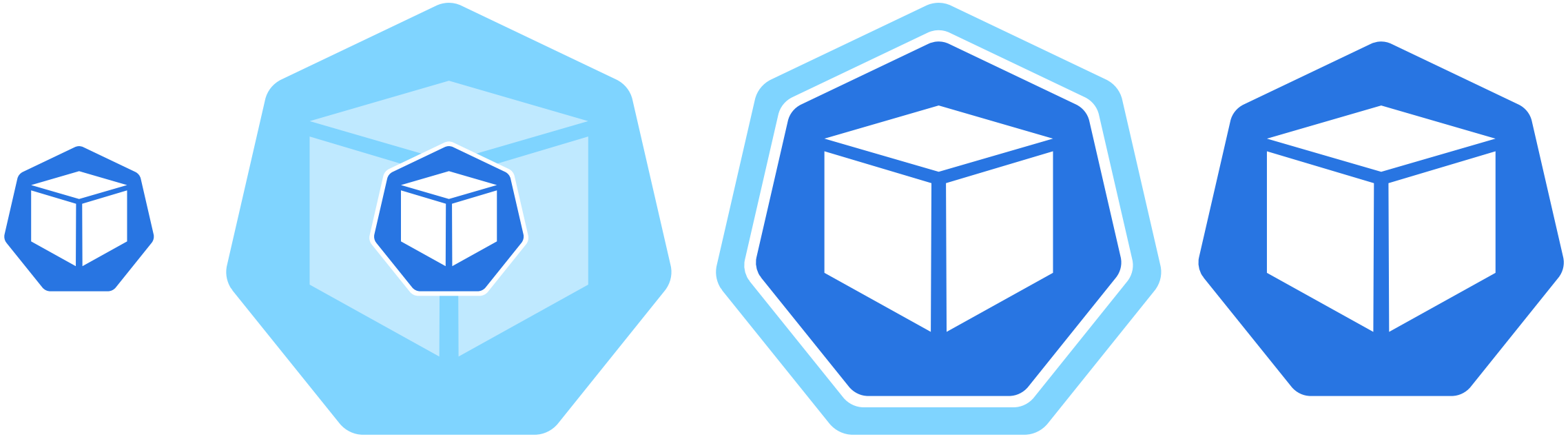
CPU +1 Memory +1Gi

Профайлер с инструментацией

Добавление ресурсов при профилировании

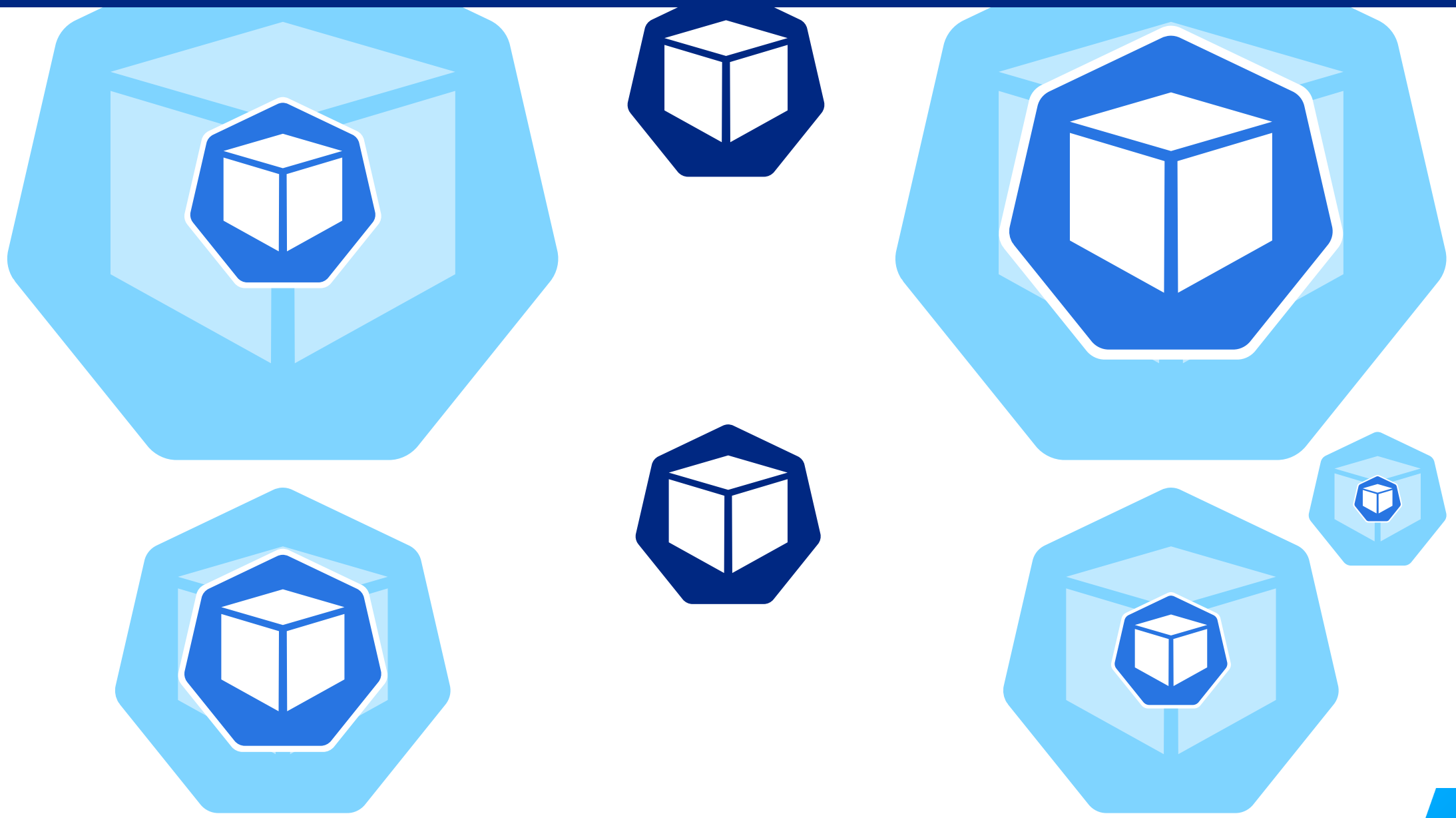
Особенности Kubernetes

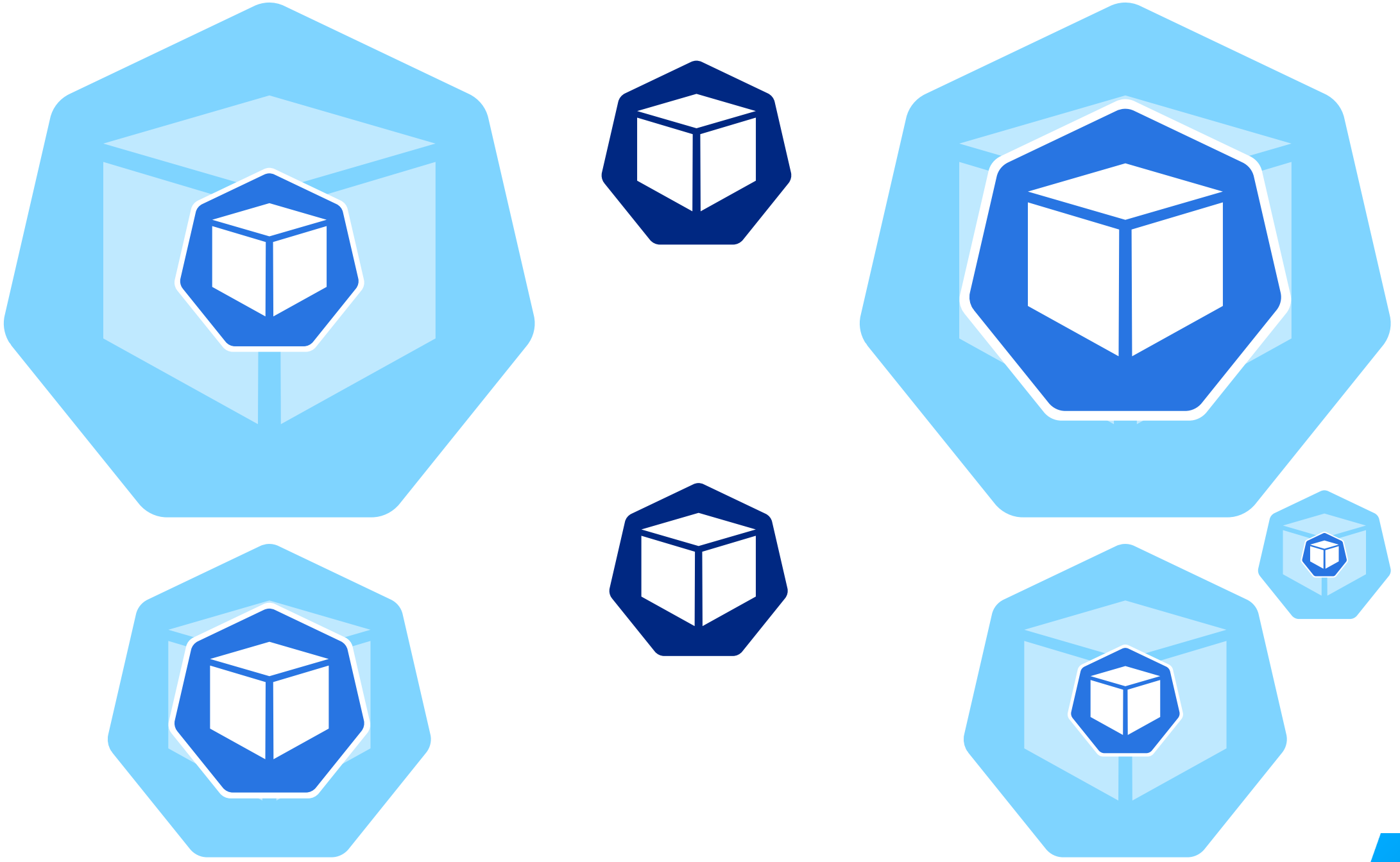
Профайлер

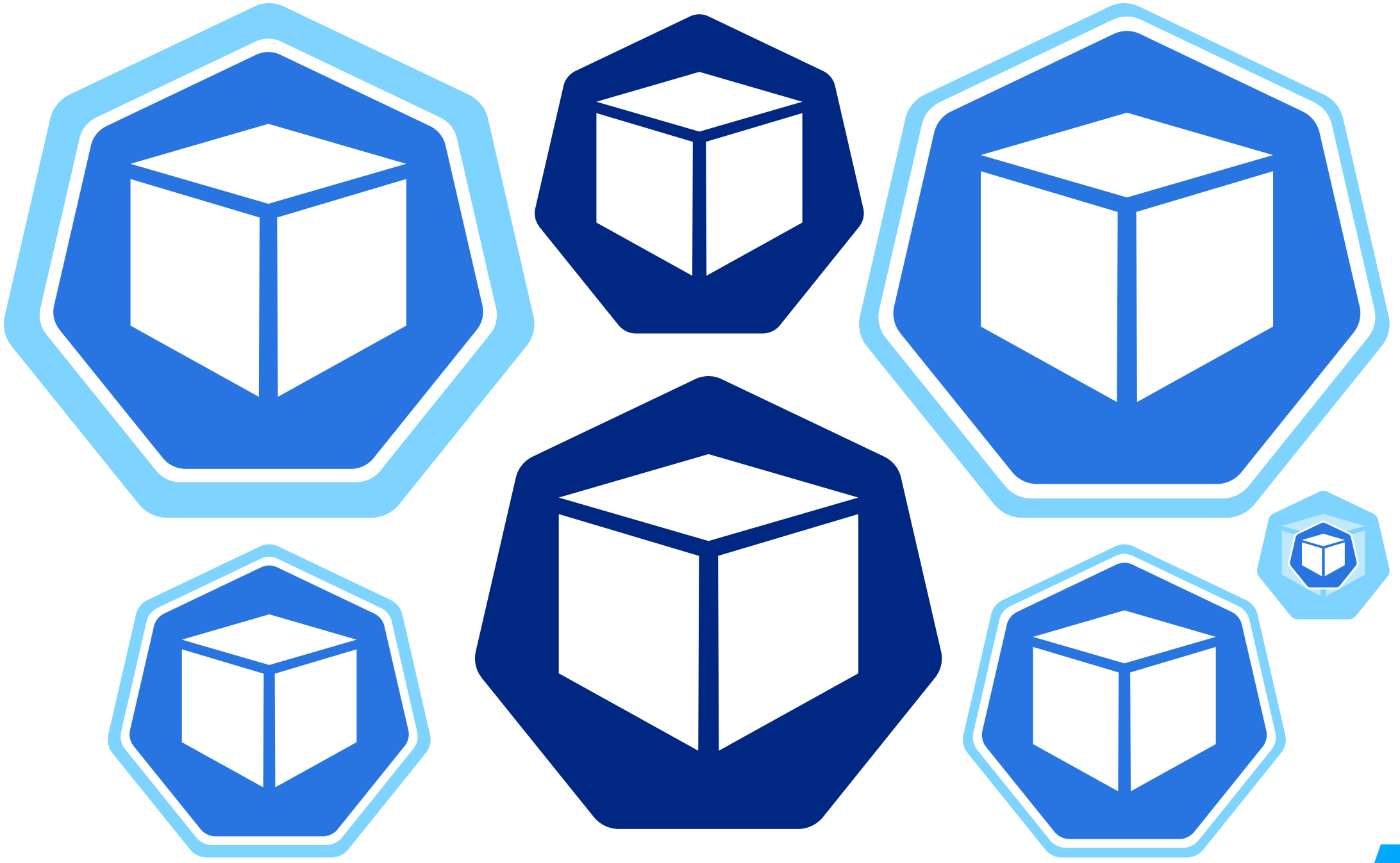


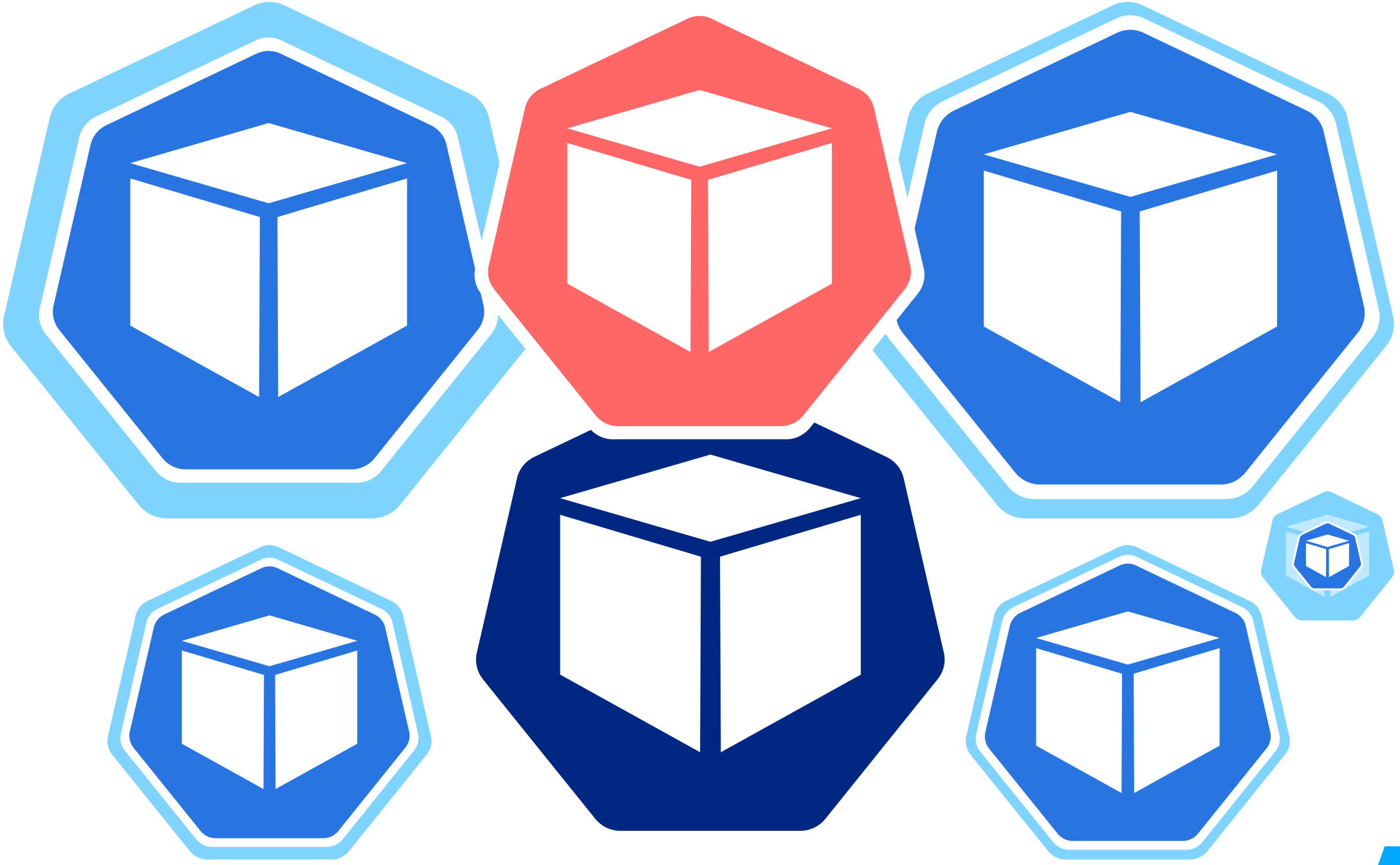
Может понадобиться +1 CPU, +1 GiB Memory

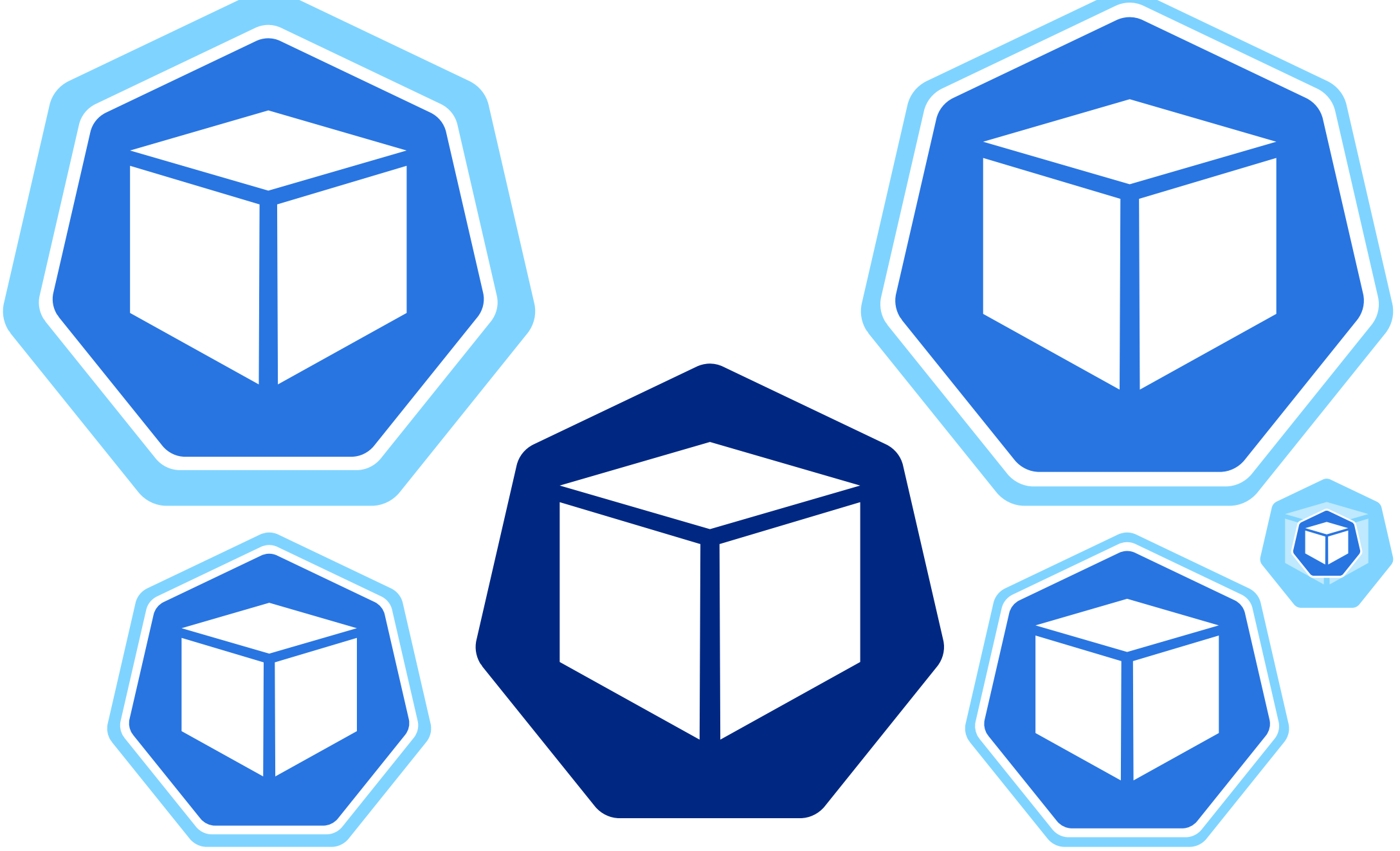
Limit не задан у профилируемых сервисов



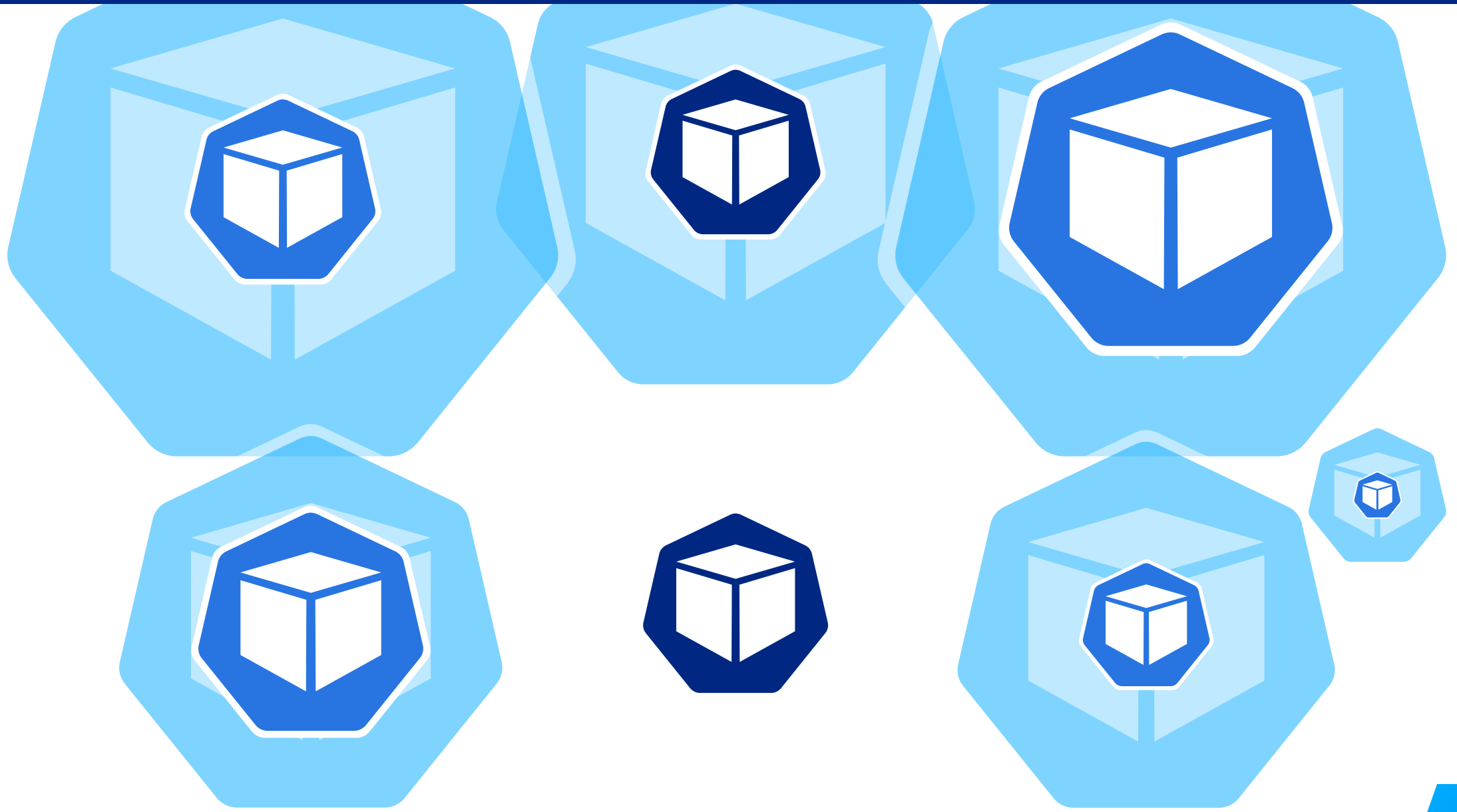


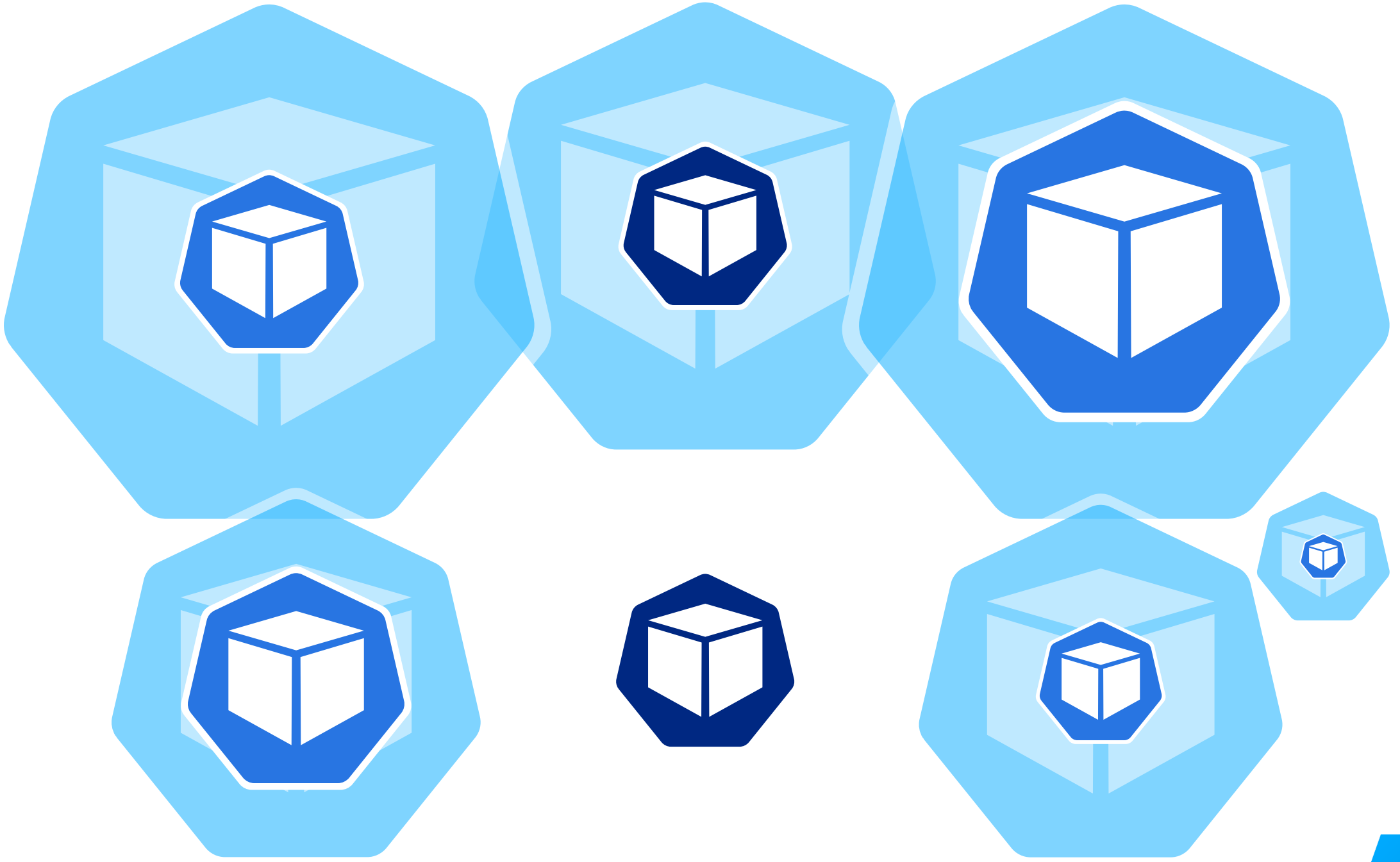


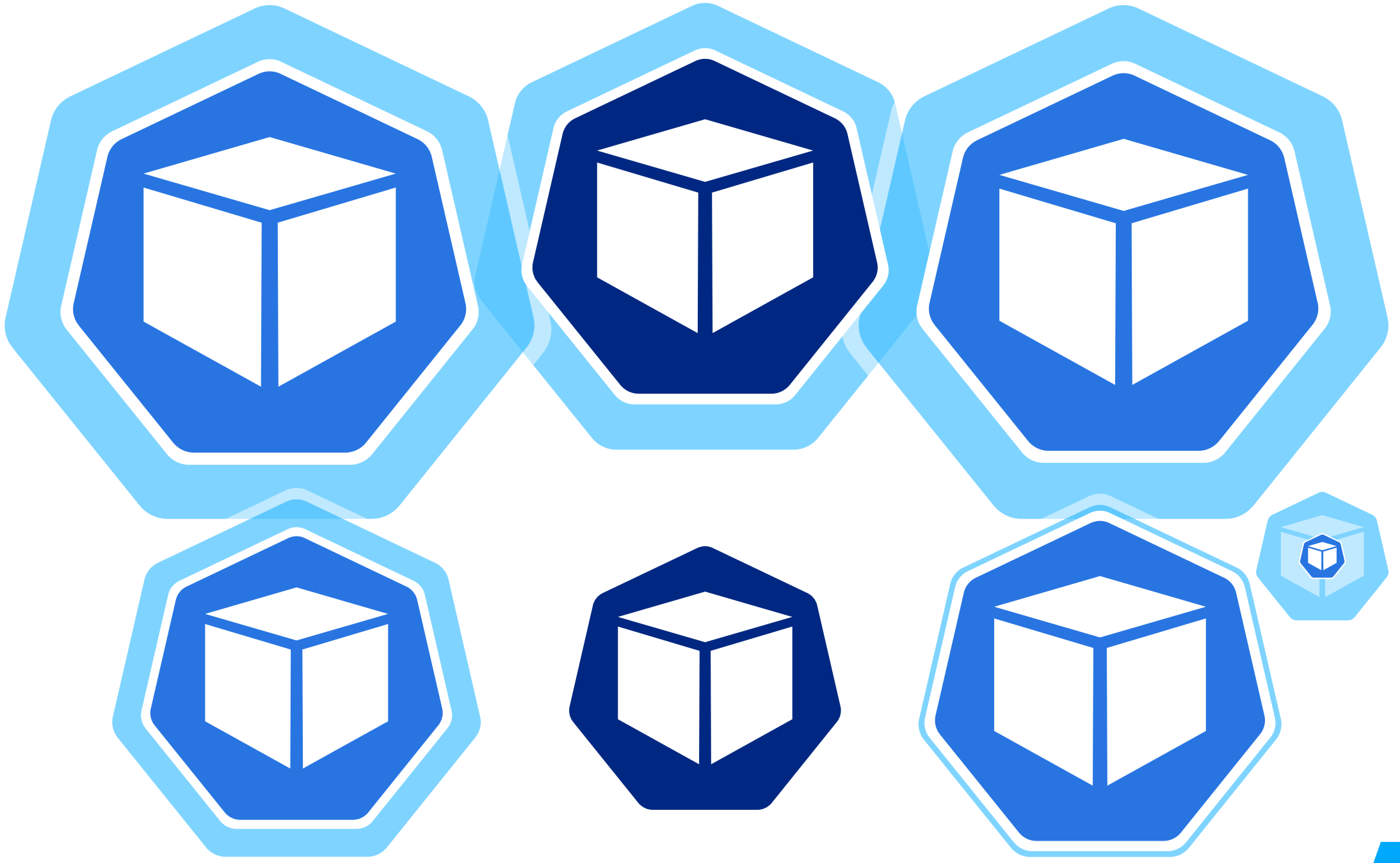


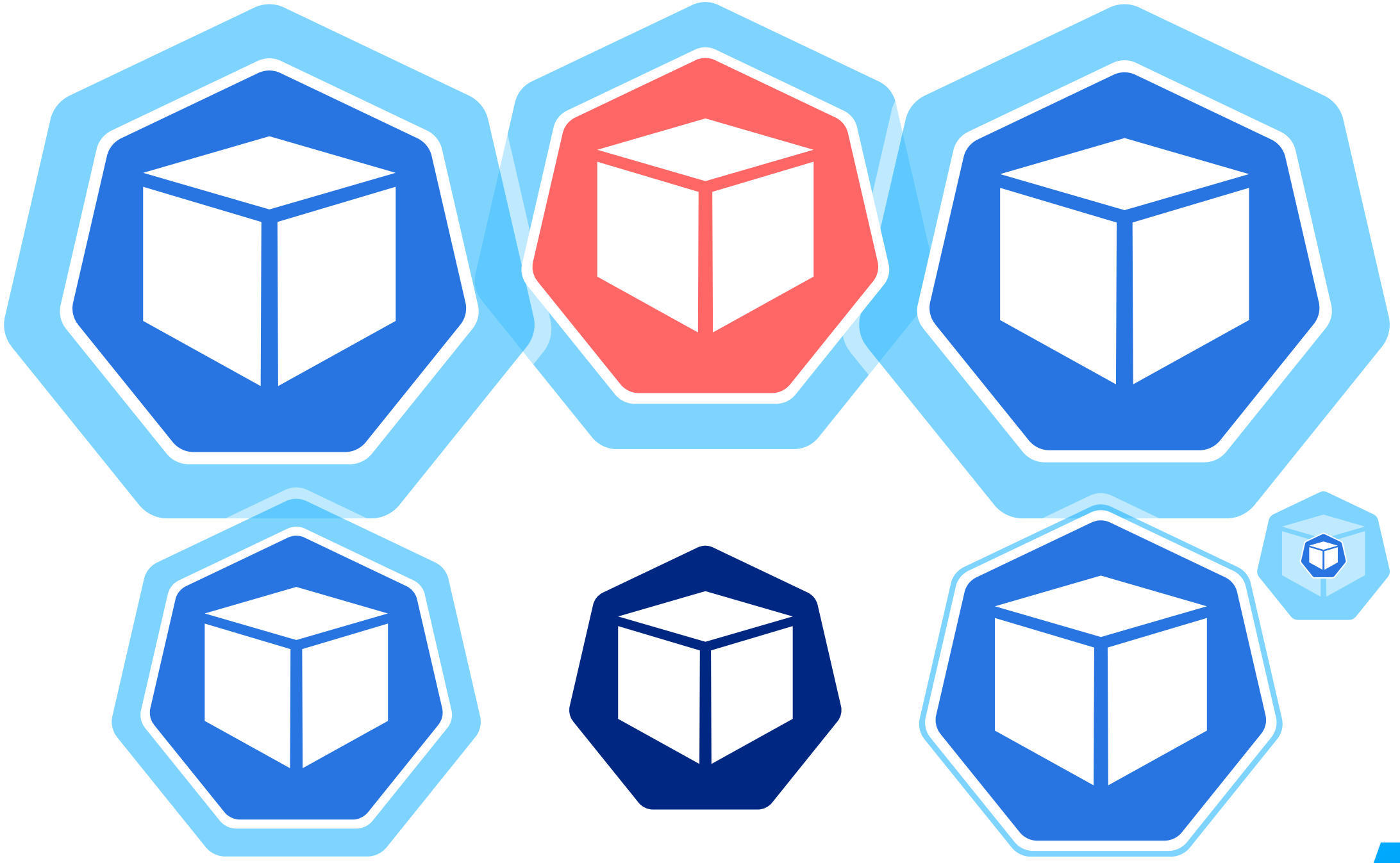


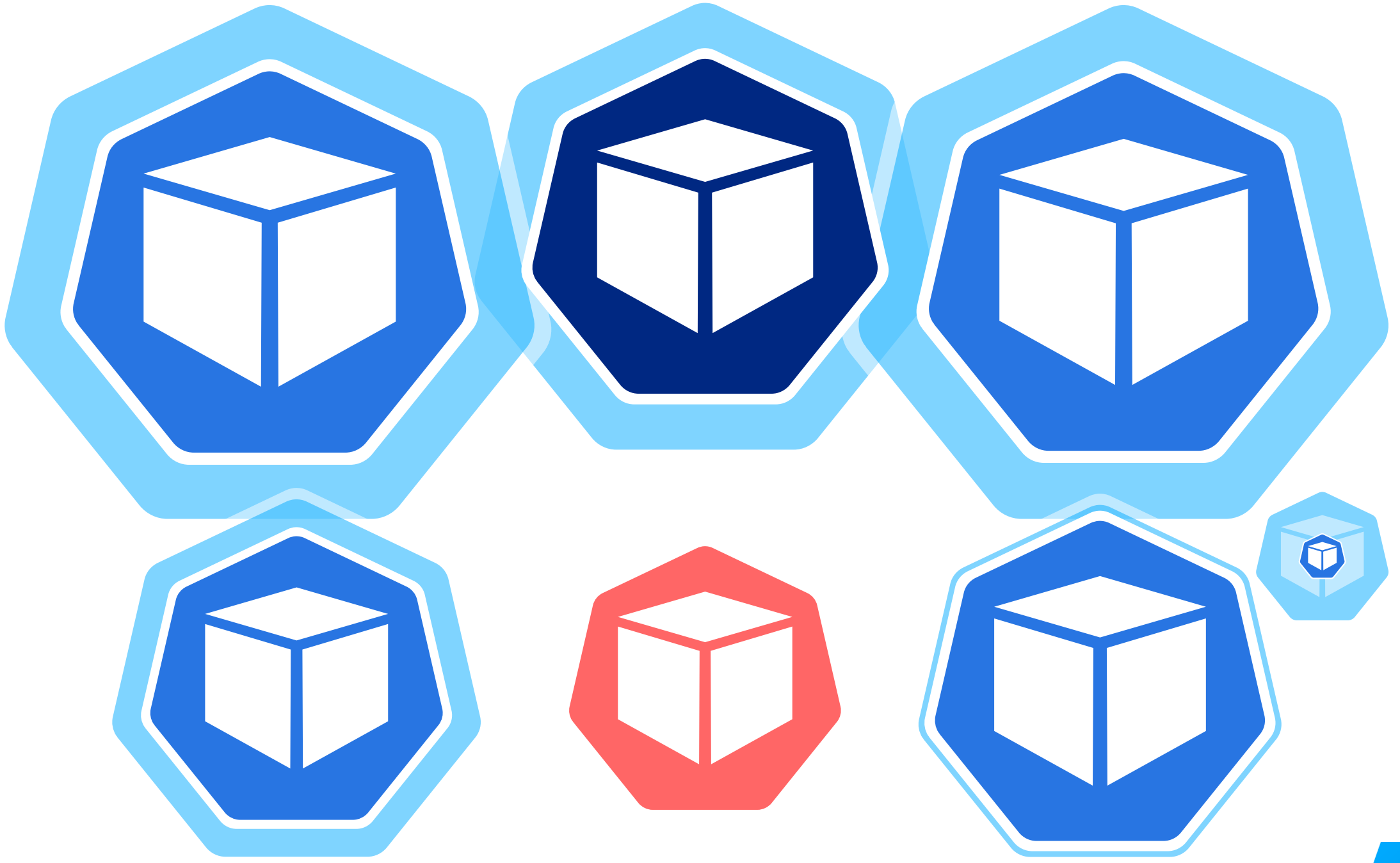
Limit задан не у всех профилируемых сервисов

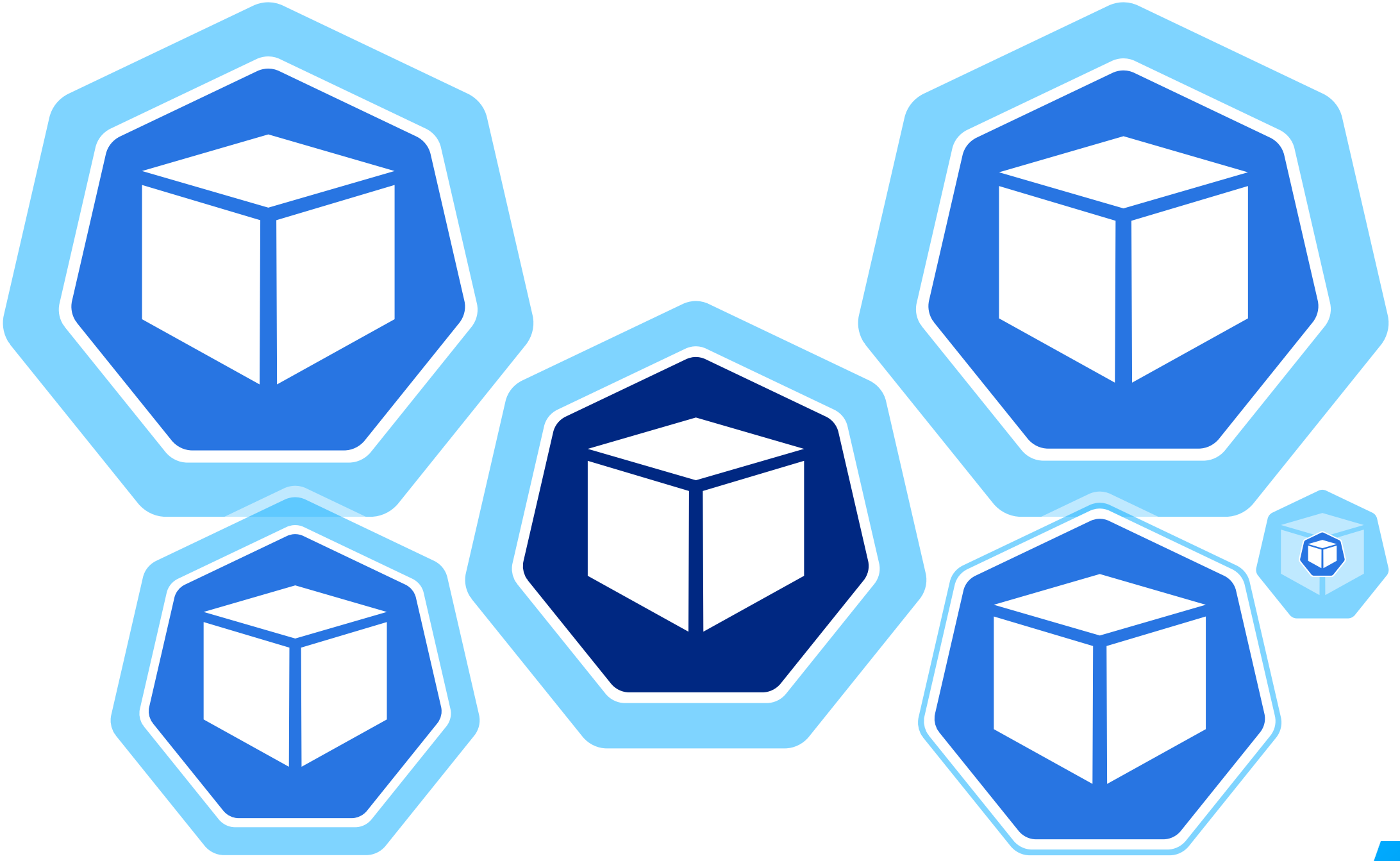




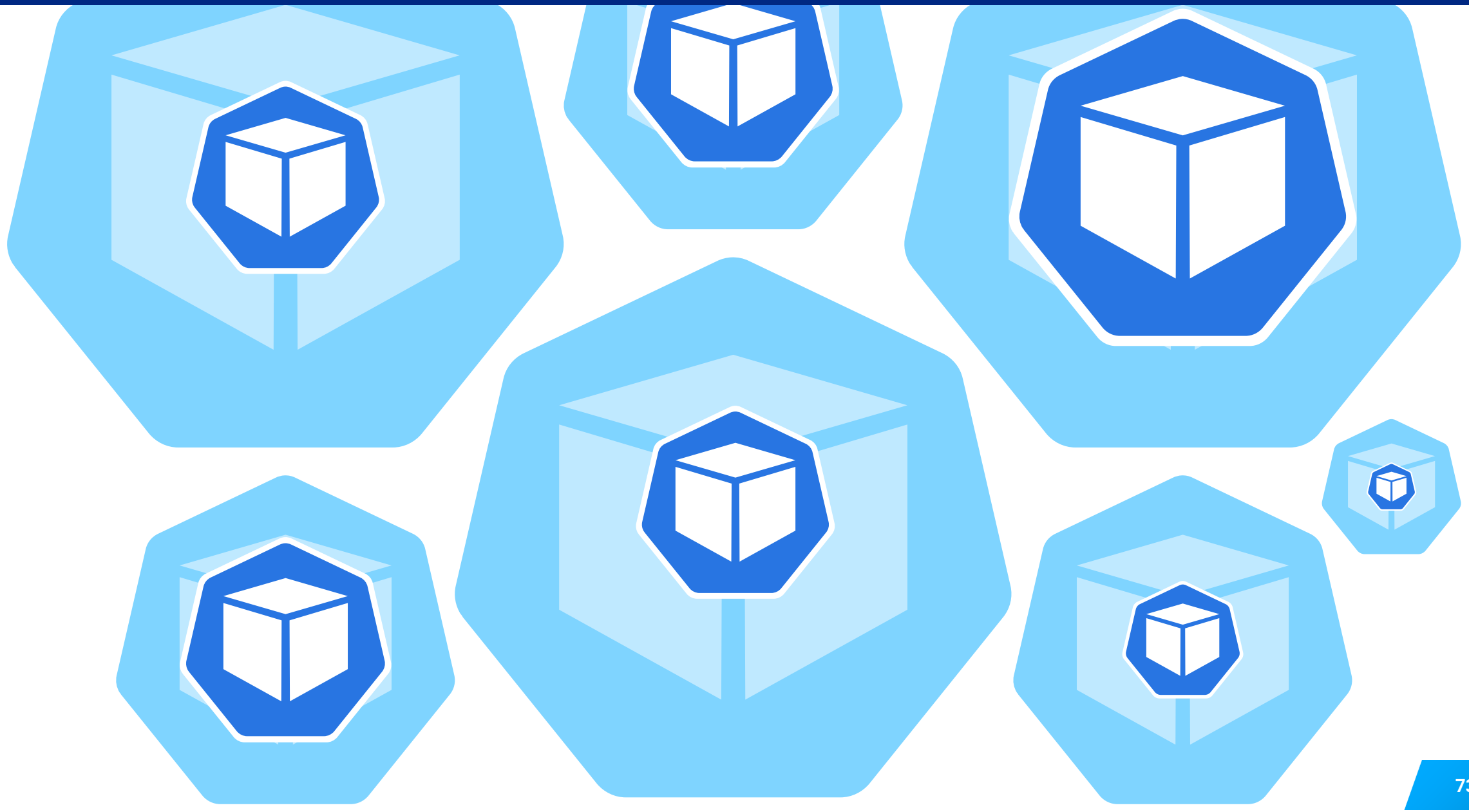


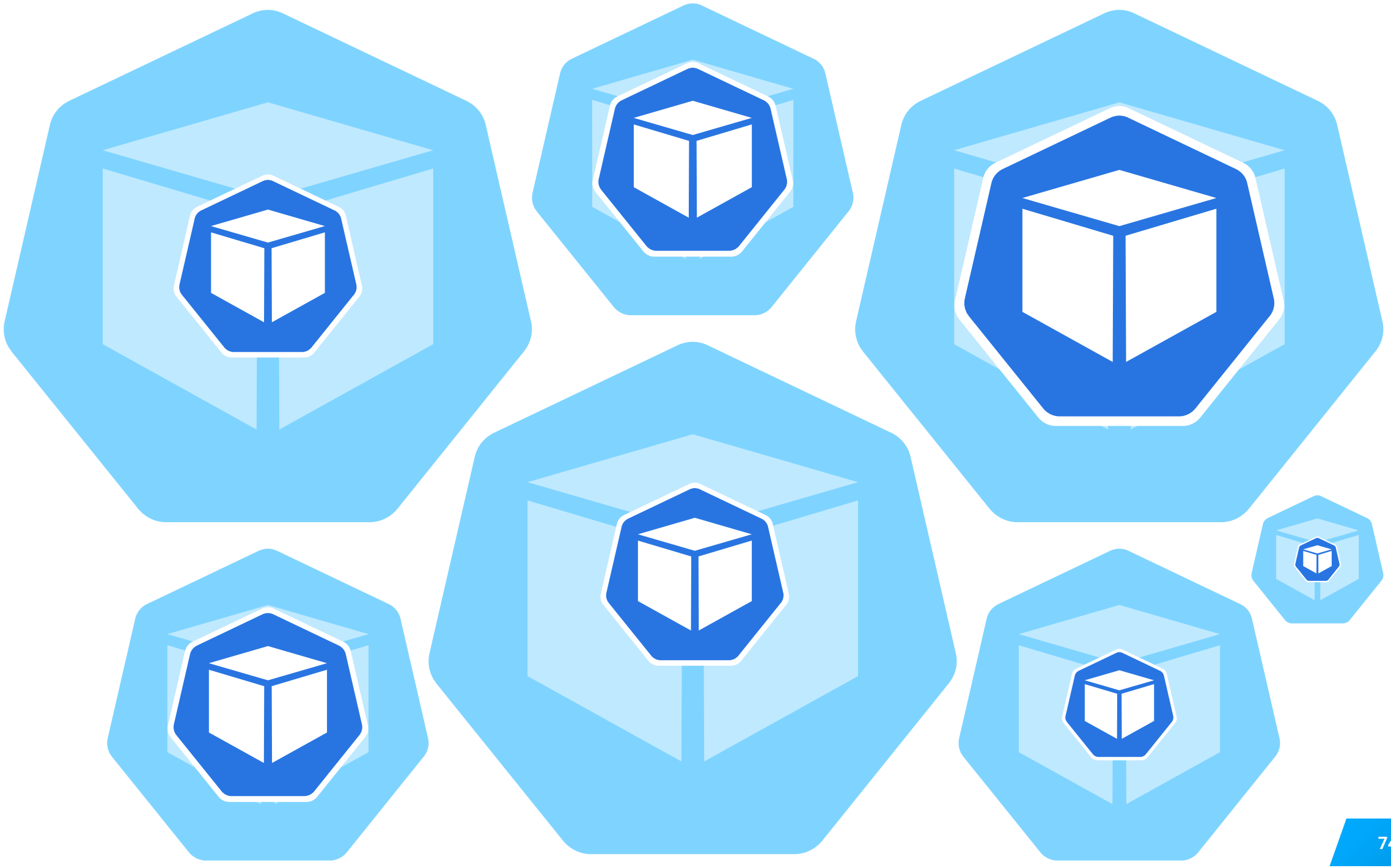


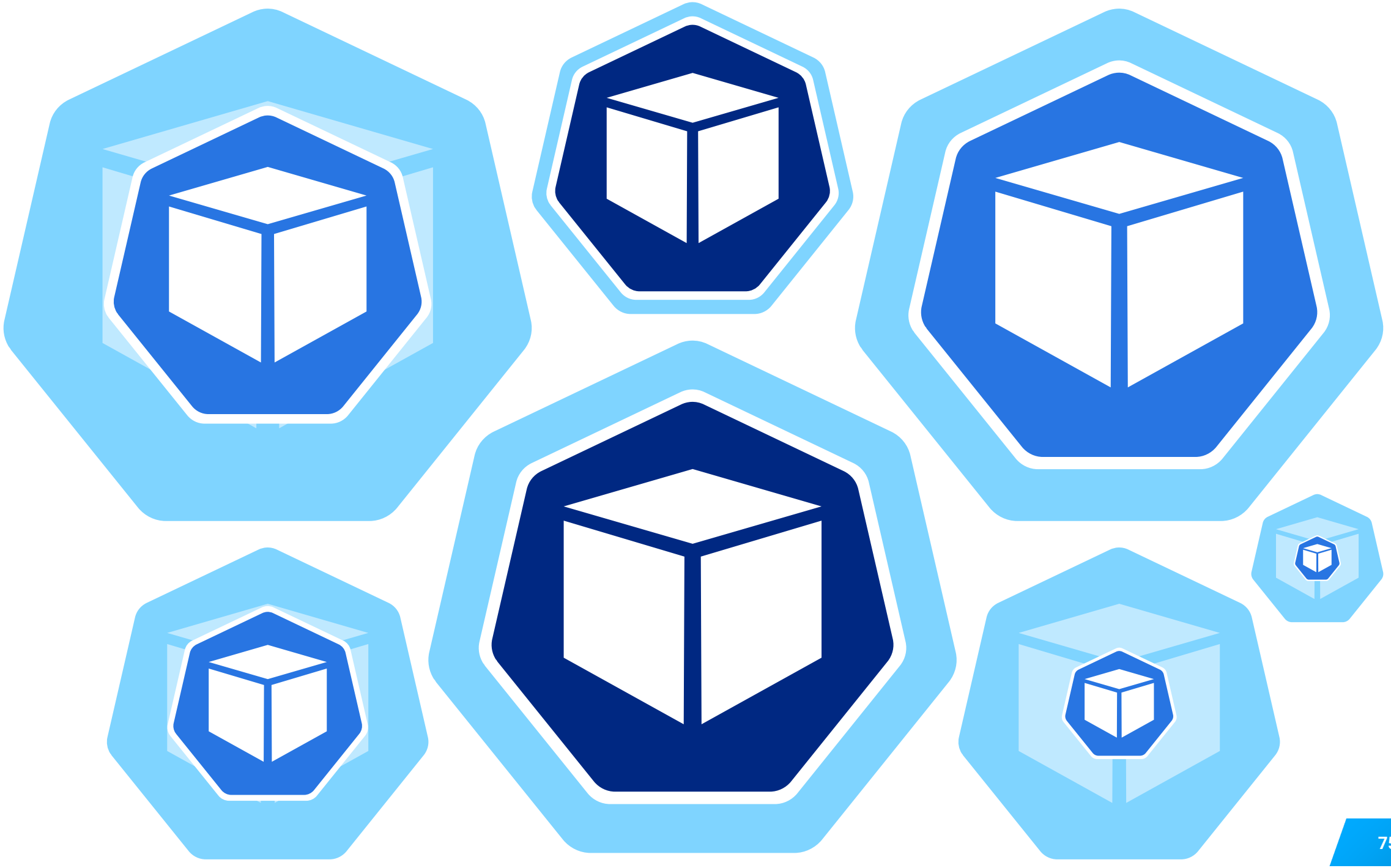


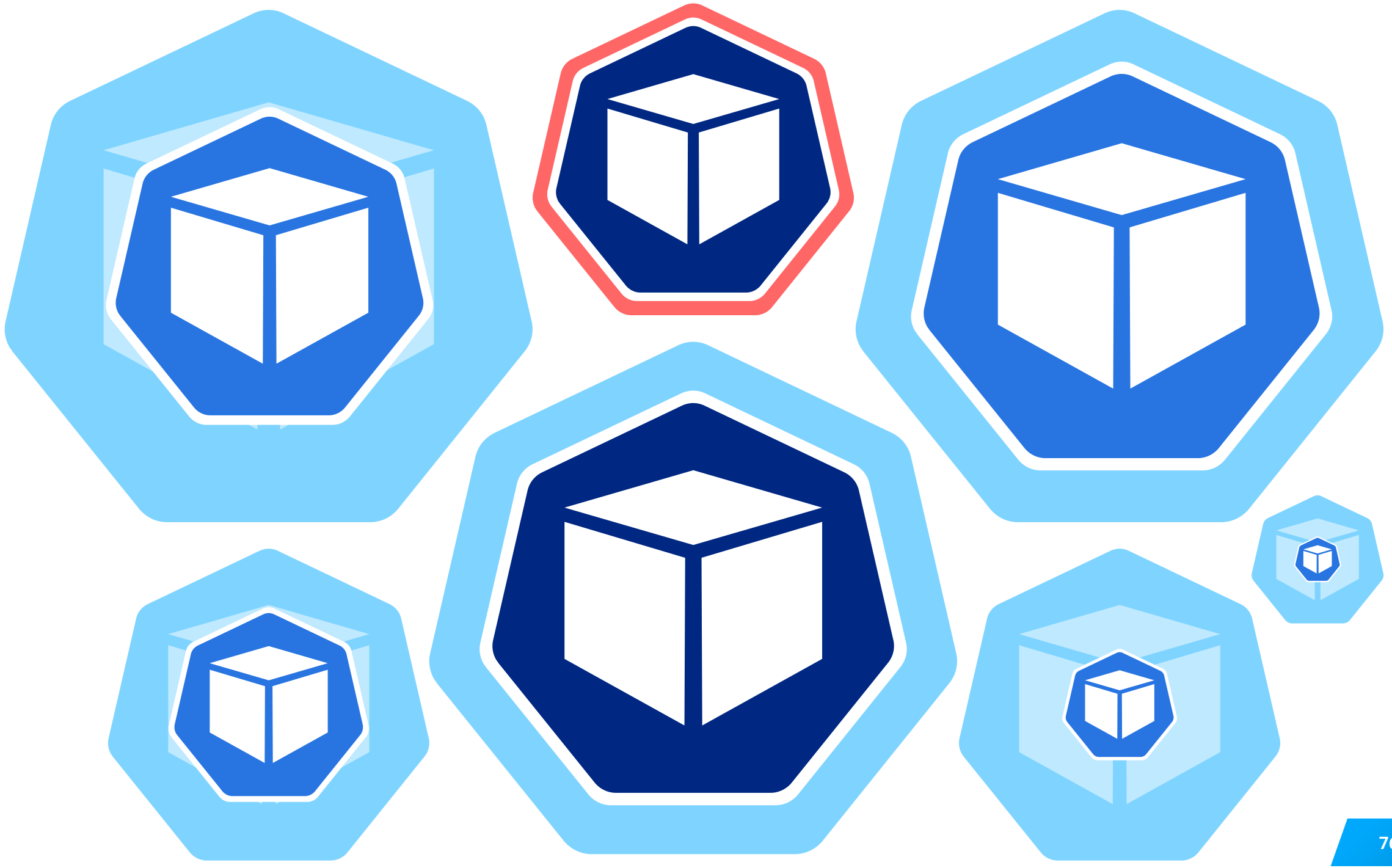


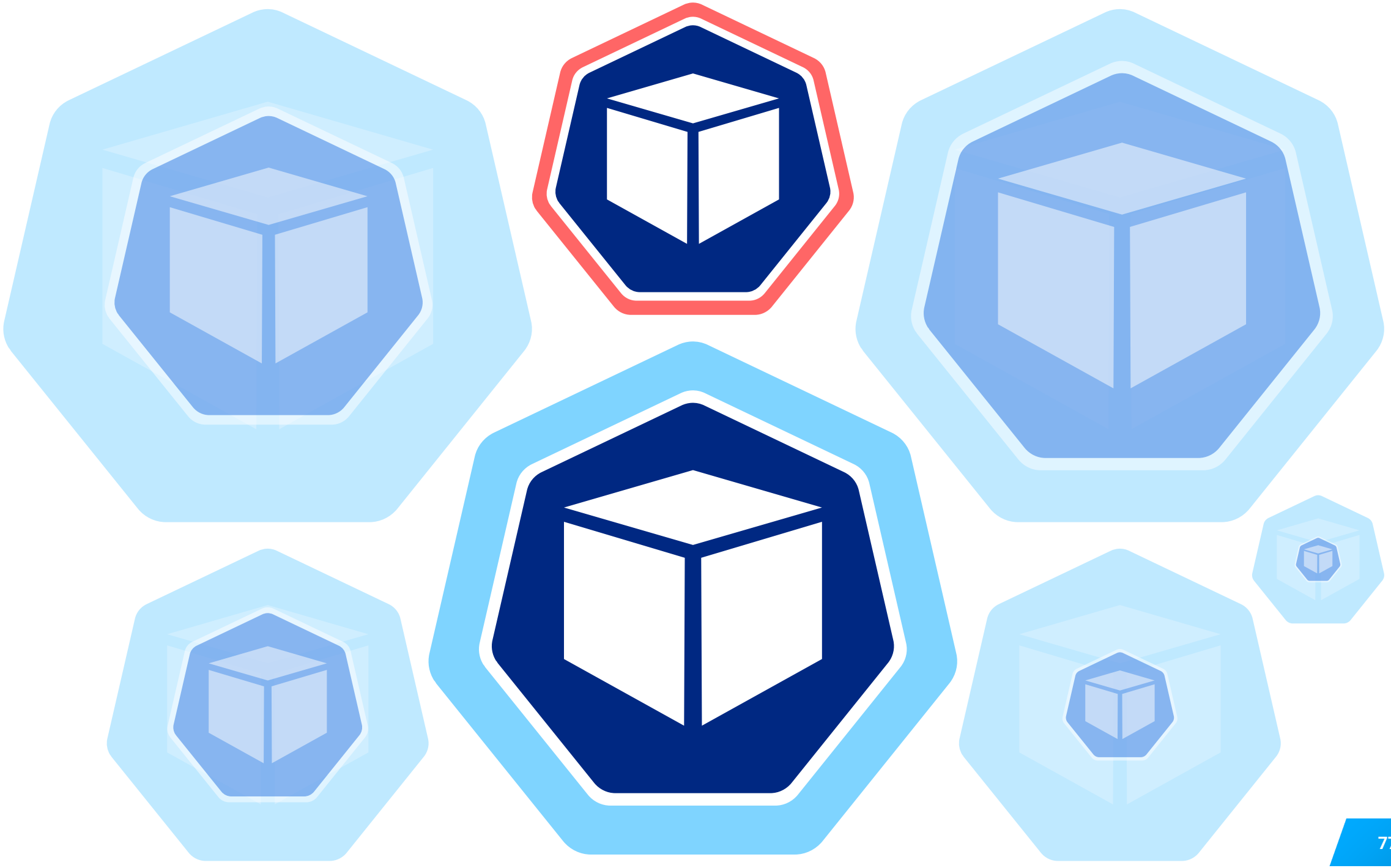
Limit задан у всех профилируемых сервисов









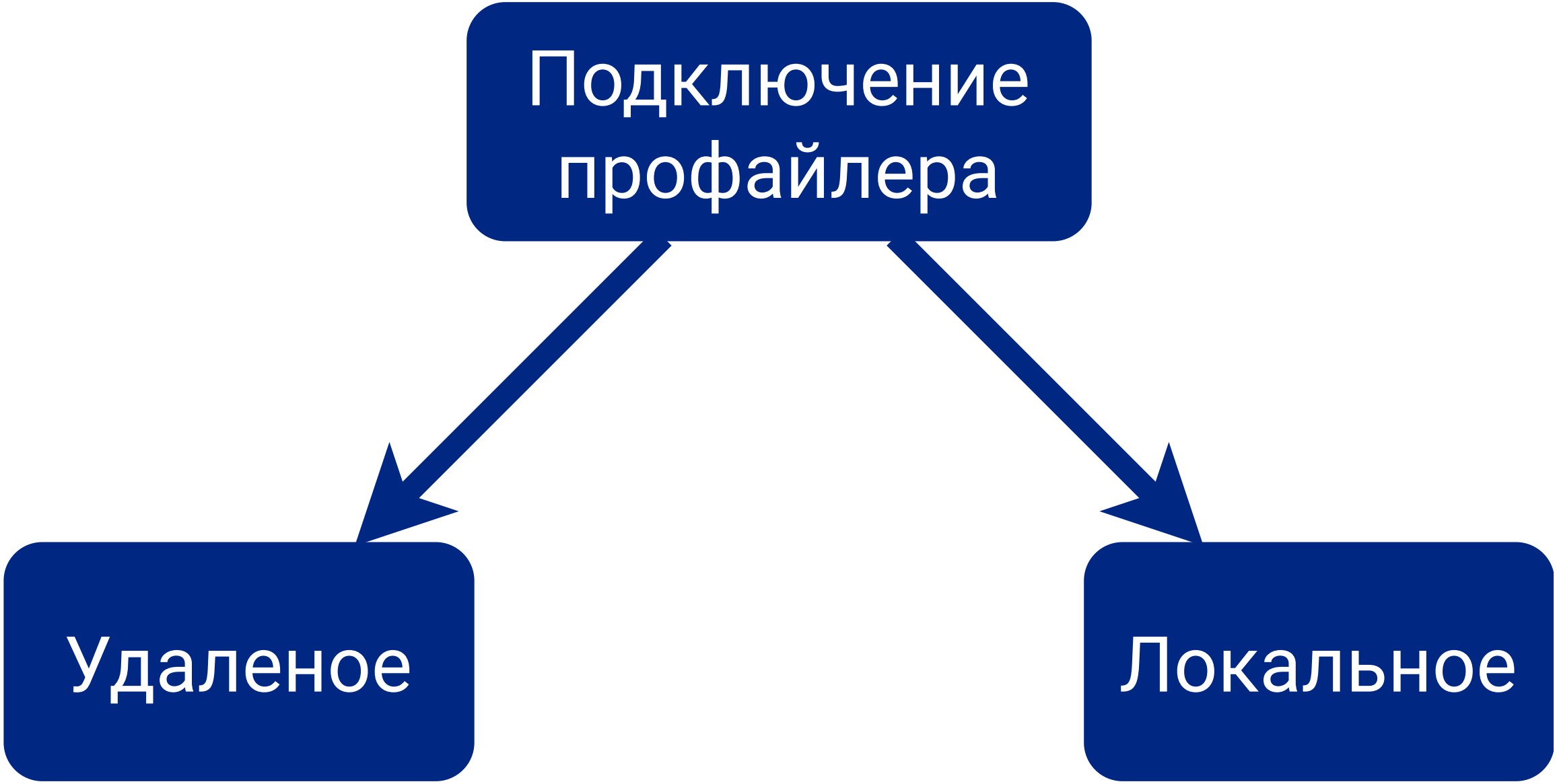


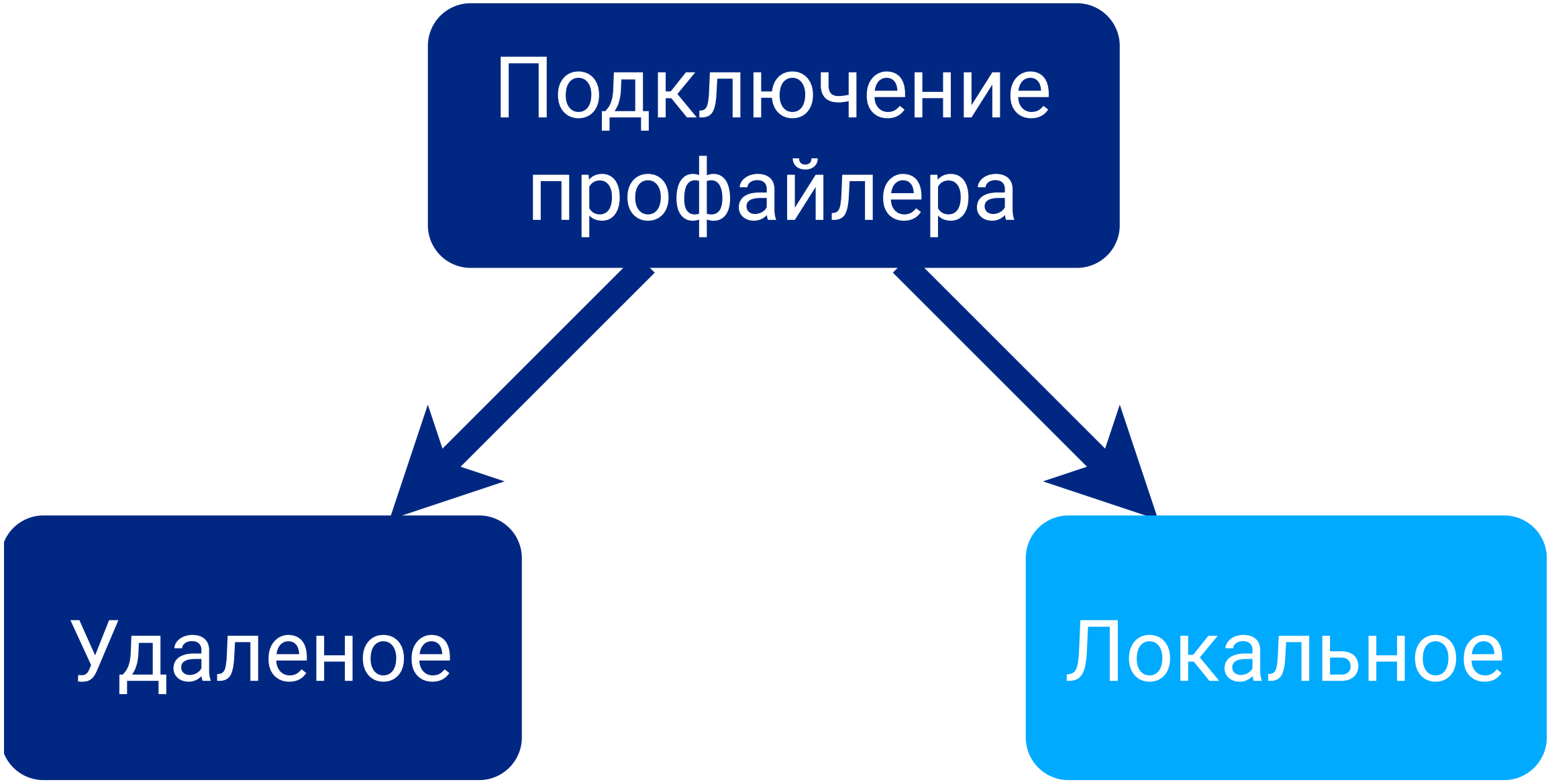
Может понадобиться +1 CPU, +1 GiB Memory

Подключение профайлера к JVM в Kubernetes

Особенности профайлеров

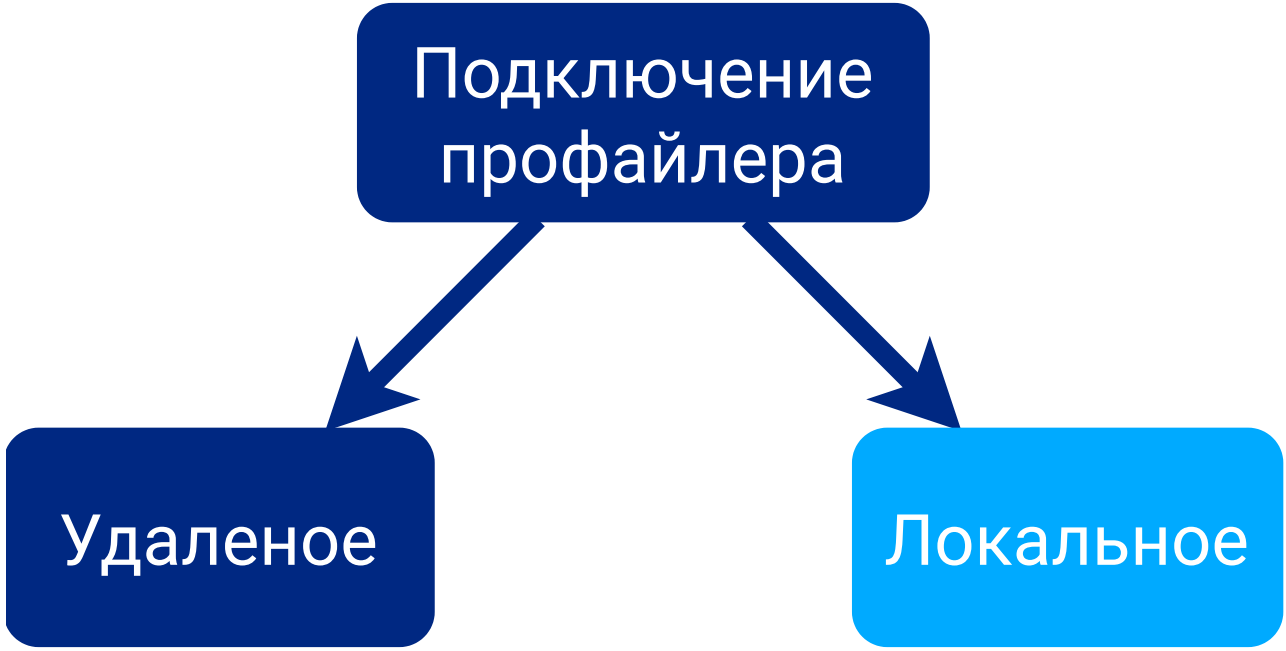
Профайлер

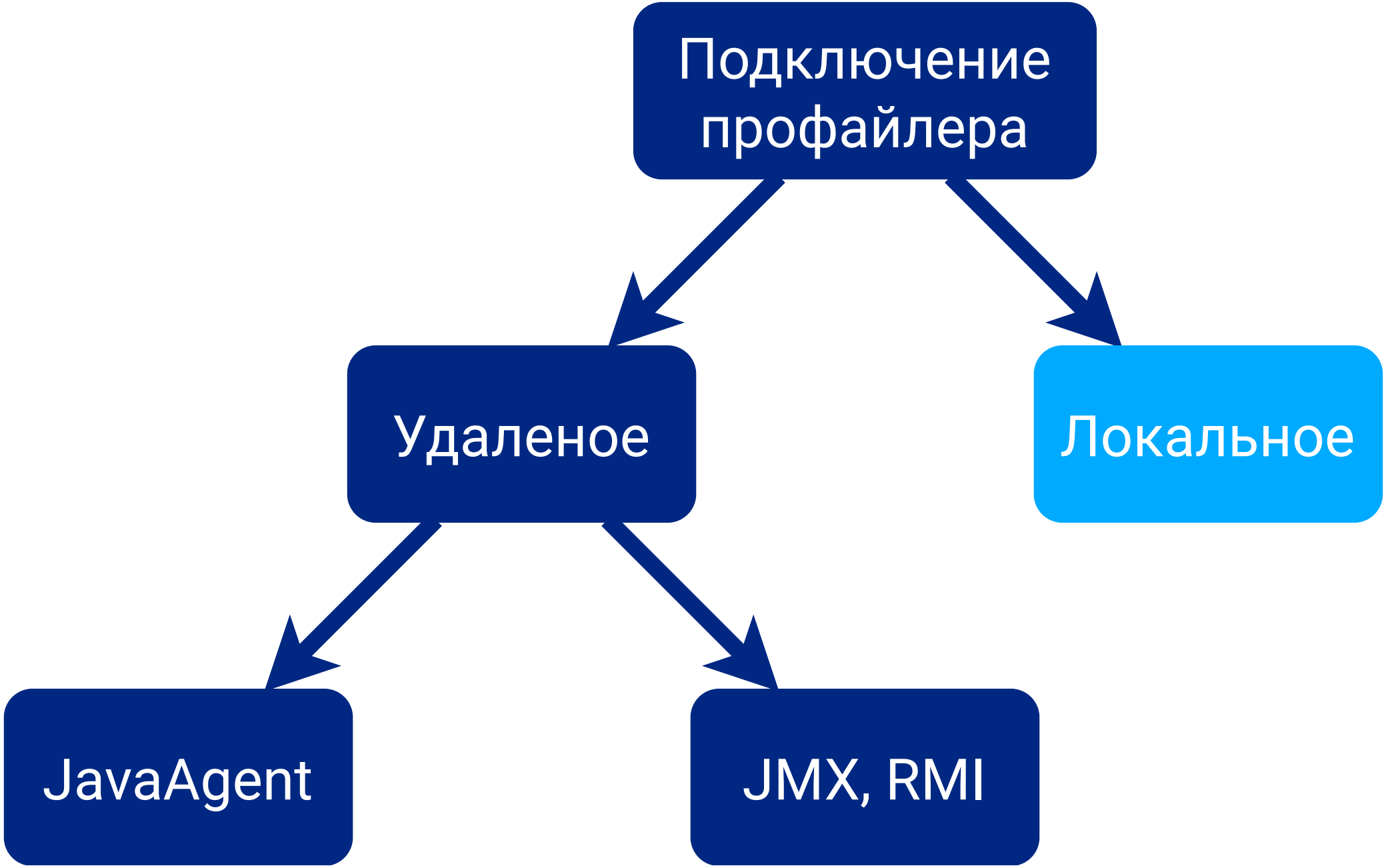

Подключение профайлера
Удаленое
Локальное

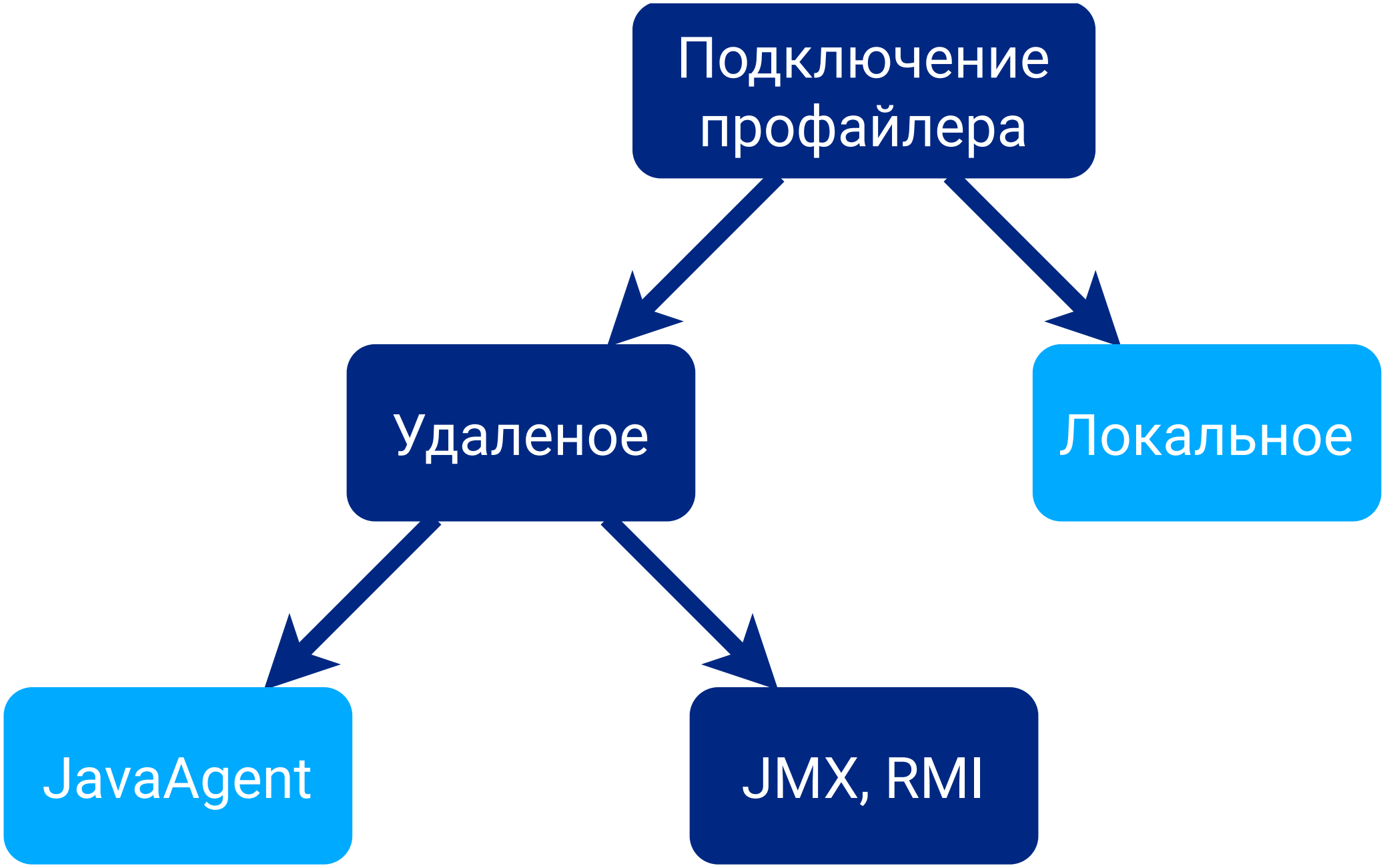

Подключение профайлера
Удаленое
Локальное

Удаленное подключение

- Доступ до Pod-ы на время:
  - **PortForward**
- Доступ до Service (1 Pod):
  - **NodePort**
  - LoadBalancer
- Доступ до Service (2+ Pod):
  - Не получится




Подключение профайлера
Удаленое
Локальное
JavaAgent
JMX, RMI


Подключение профайлера
Удаленое
Локальное
JavaAgent
JMX, RMI

Опции JMX, RMI для удаленного подключения

Опции JVM задаются в Deployment

```
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=9010
-Dcom.sun.management.jmxremote.rmi.port=9010
-Dcom.sun.management.jmxremote.local.only=true
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
-Djava.rmi.server.hostname=127.0.0.1
```

Проброс локального порта в Pod

```
kubectl port-forward "<имя поды>" 9010:9010
```

Две Pod одного Service так не подключить

Опции JVM задаются в Deployment — общие для Pod-ов

```
-Dcom.sun.management.jmxremote.port=9010
-Dcom.sun.management.jmxremote.rmi.port=9010
```

Проброс локального порта в Pod дважды не сделать

```
$ kubectl port-forward "<имя поды 1>" 9010:9010
Forwarding from 127.0.0.1:9010 → 9010
Forwarding from [::1]:9010 → 9010

$ kubectl port-forward "<имя поды 2>" 9010:9010
unable to create listener: Error listen tcp4 127.0.0.1:9010: bind:
Only one usage of each socket address (protocol/network address/port)
is normally permitted.
```

Профилирование другого Service — другой порт

Опции JVM на другой порт 9011

```
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=9011
-Dcom.sun.management.jmxremote.rmi.port=9011
-Dcom.sun.management.jmxremote.local.only=true
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
-Djava.rmi.server.hostname=127.0.0.1
```

Проброс другого порта для другого сервиса

```
kubectl port-forward "<имя поды>" 9011
```

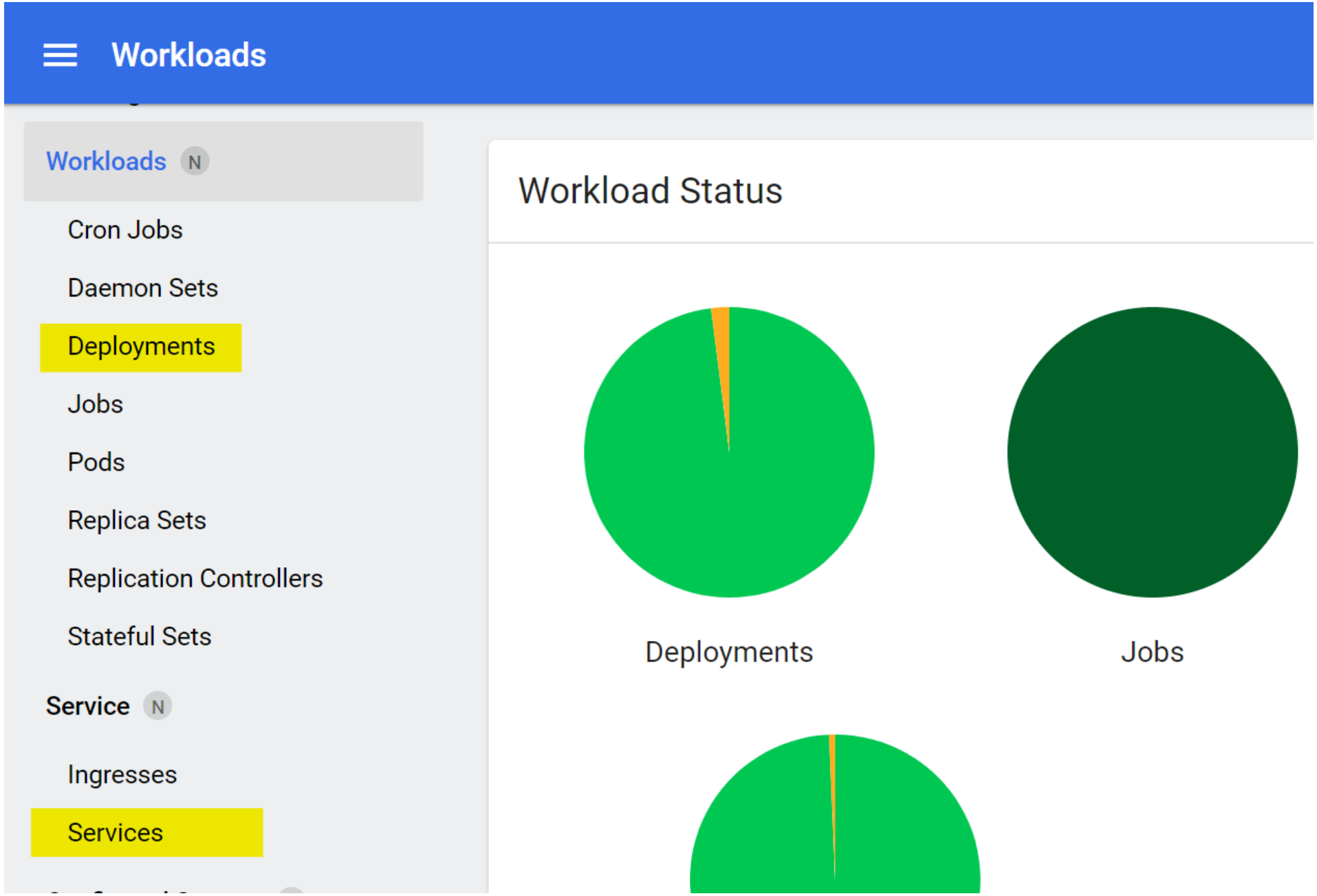
Опции JMX, RMI для NodePort

Открыть NodePort в Service

```
ports:
  - name: JMX
    protocol: TCP
    port: 31111
    targetPort: 31111
    nodePort: 31111
```

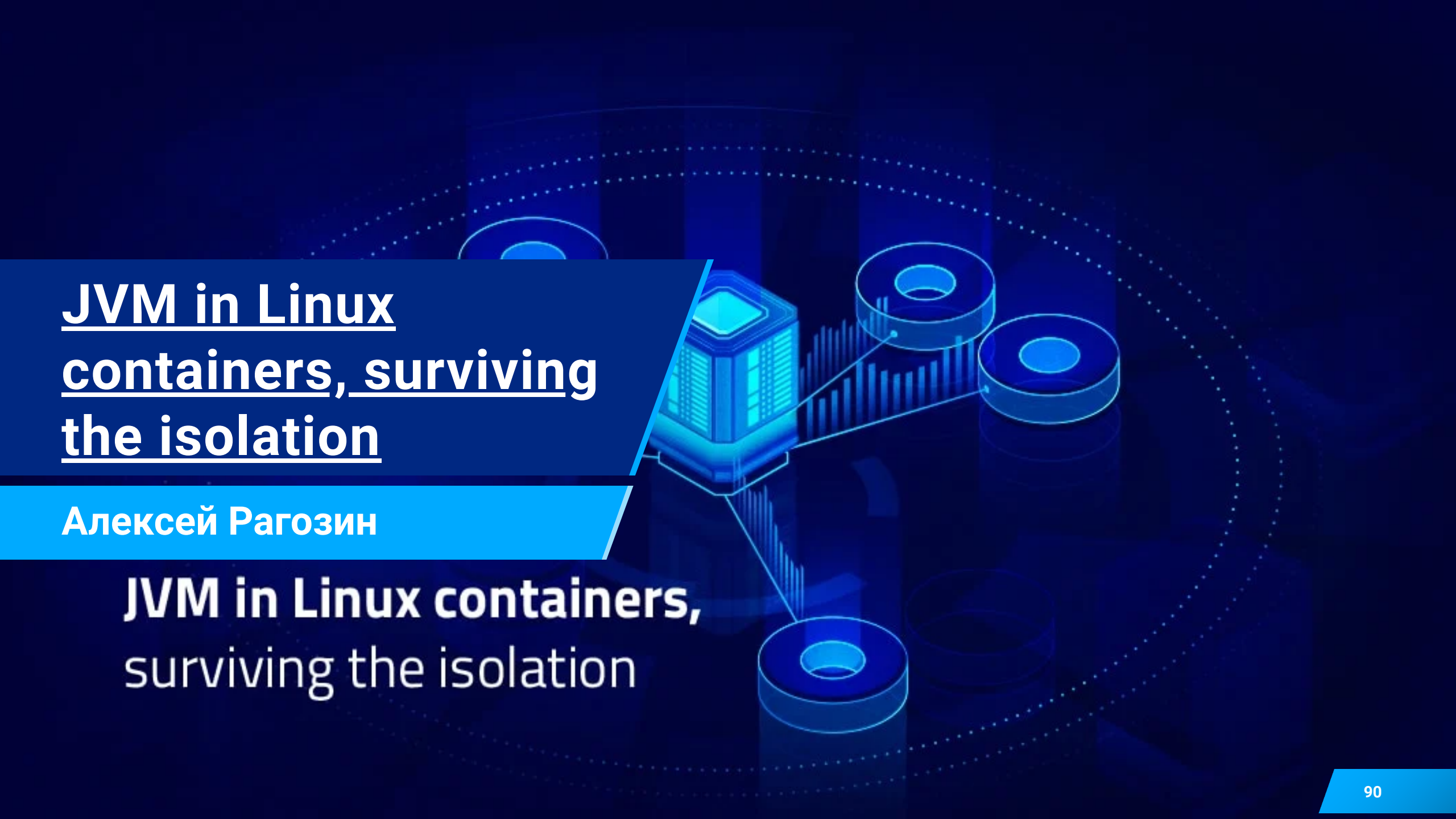
Если порт 31111 свободен и открылся, то задать его в Deployment

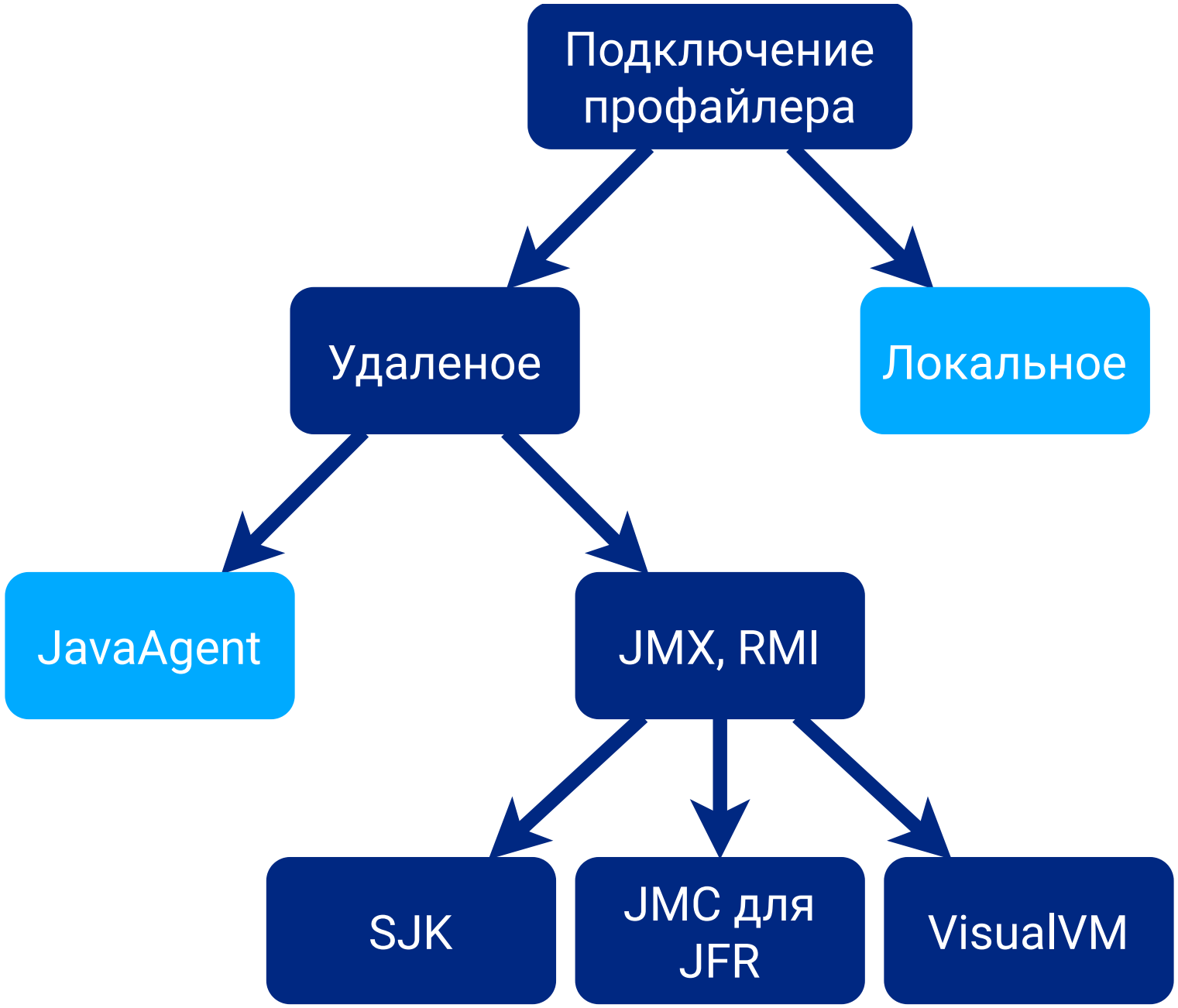
```
-Dcom.sun.management.jmxremote.port=31111
-Dcom.sun.management.jmxremote.rmi.port=31111
-Djava.rmi.server.hostname=<адрес NodeHost>
```

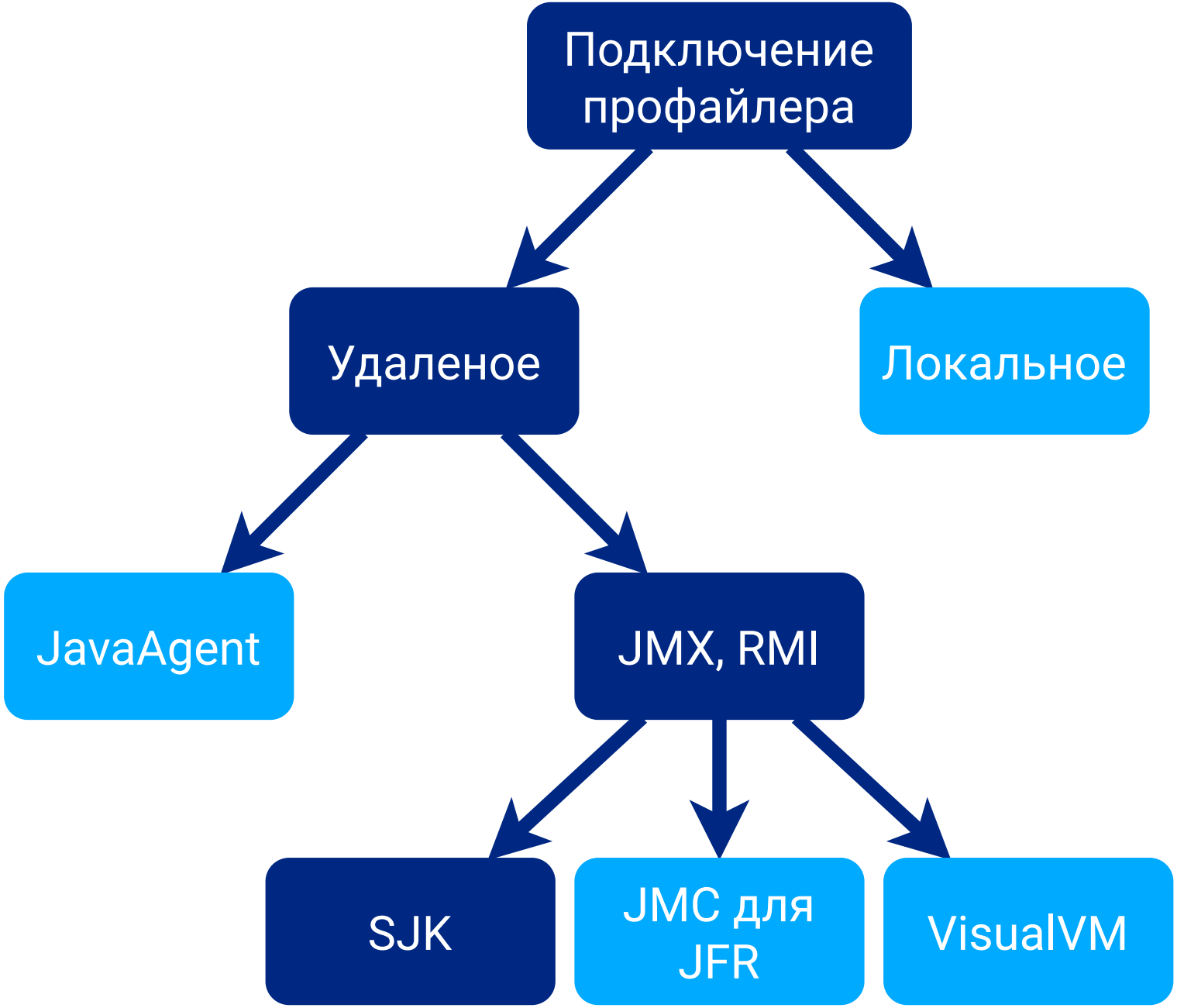

Workloads
Workloads N
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Service N
Ingresses
Services
Workload Status
Deployments
Jobs

JVM in Linux containers, surviving the isolation

Алексей Рагозин




Подключение профайлера
Удаленое
Локальное
JavaAgent
JMX, RMI
SJK
JMC для JFR
VisualVM


Подключение профайлера
Удаленое
Локальное
JavaAgent
JMX, RMI
SJK
JMC для JFR
VisualVM

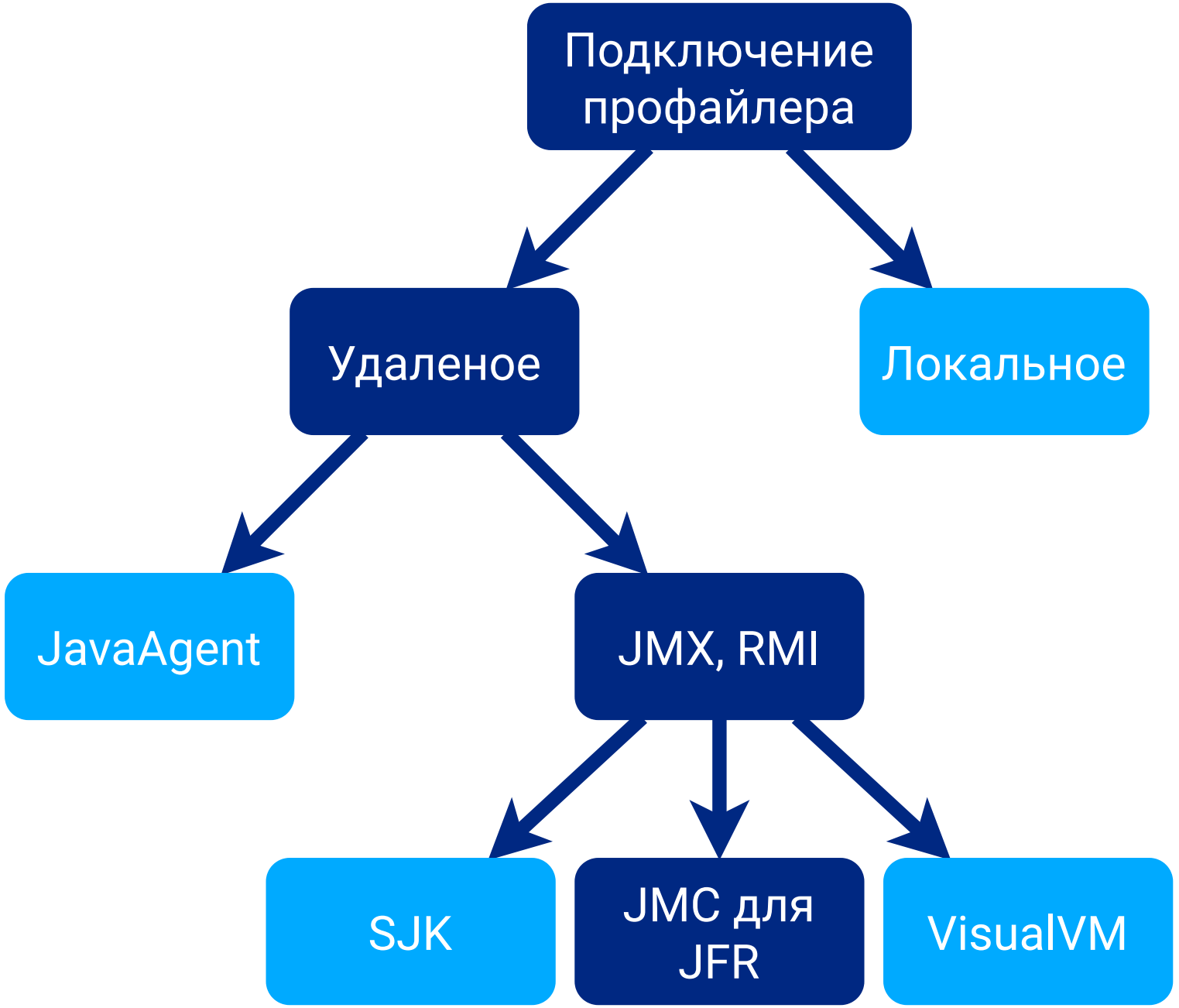
SJK удаленное (локальное) профилирование

С ограниченной интенсивностью

```
java -jar ./sjk-0.17.jar stcap \
  -s localhost:9010 \
  -o "result.sjk" \
  --sampler-interval 100ms \
  --timeout 5m
```

С максимальной интенсивностью

```
java -jar ./sjk-0.17.jar stcap \
  -s localhost:9010 \
  -o "result.sjk" \
  --timeout 5m
```


Подключение профайлера
Удаленое
Локальное
JavaAgent
JMX, RMI
SJK
JMC для JFR
VisualVM

JMC для JFR удаленное профилирование

Для OpenJDK 8u272 и старше, OpenJDK 11, OpenJDK 12, ...

- Настройки JVM, кроме опций JMX/RMI, не требуются

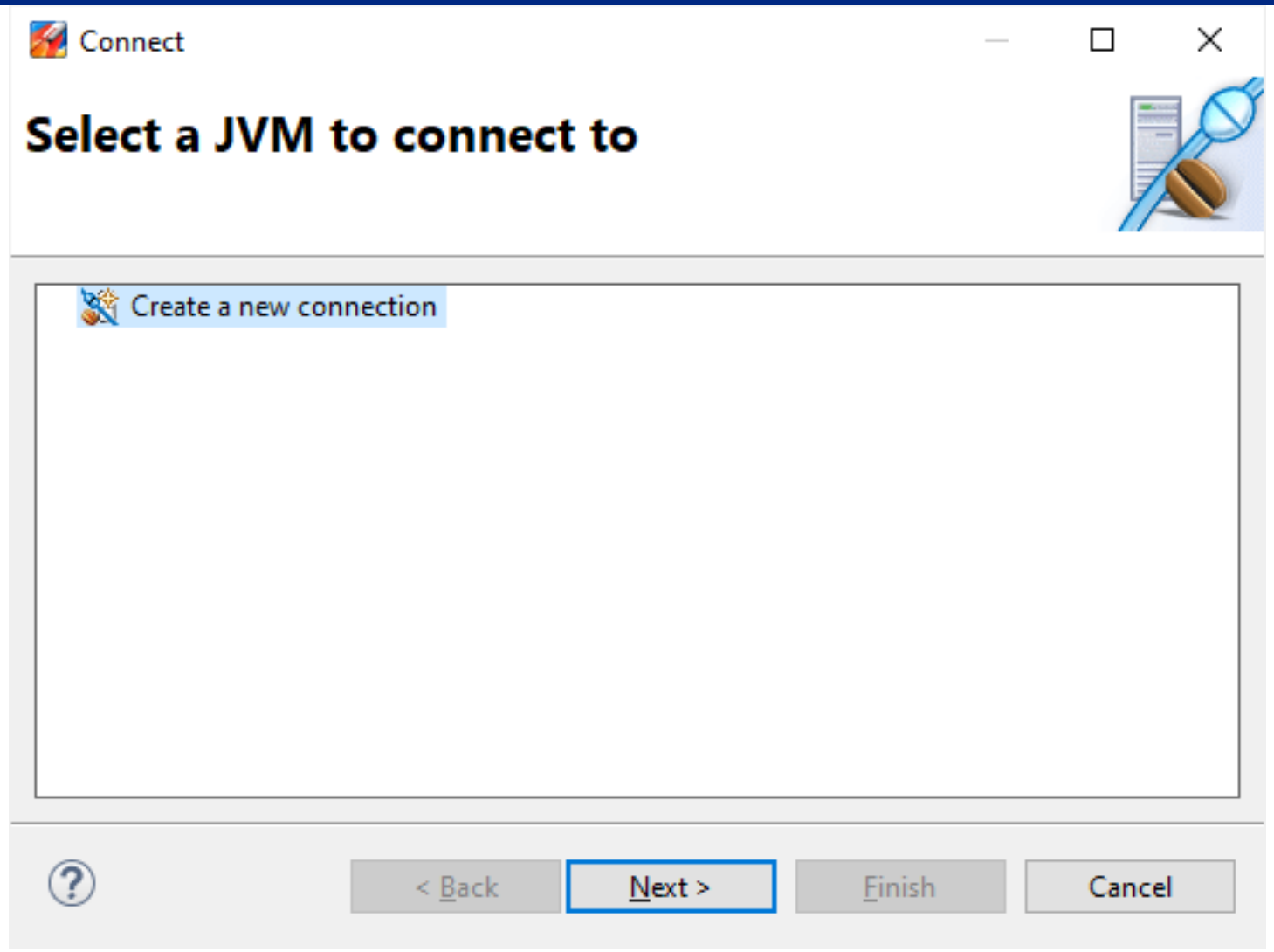
Для OpenJDK 8u271 и младше

- Монтировать OpenJDK 8u272 и старше в контейнер

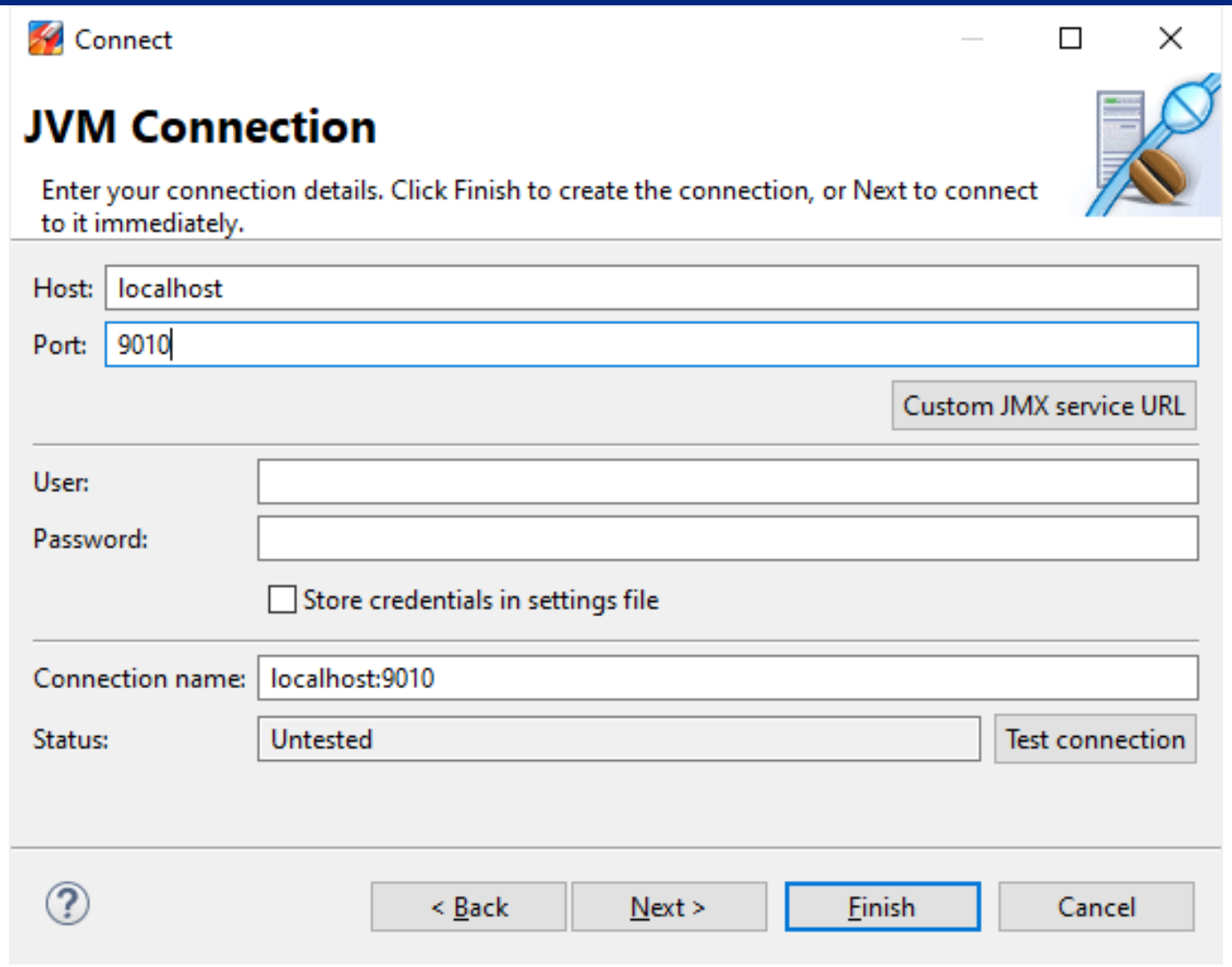
Для Oracle JDK 8

- -XX:+UnlockCommercialFeatures -XX:+FlightRecorder

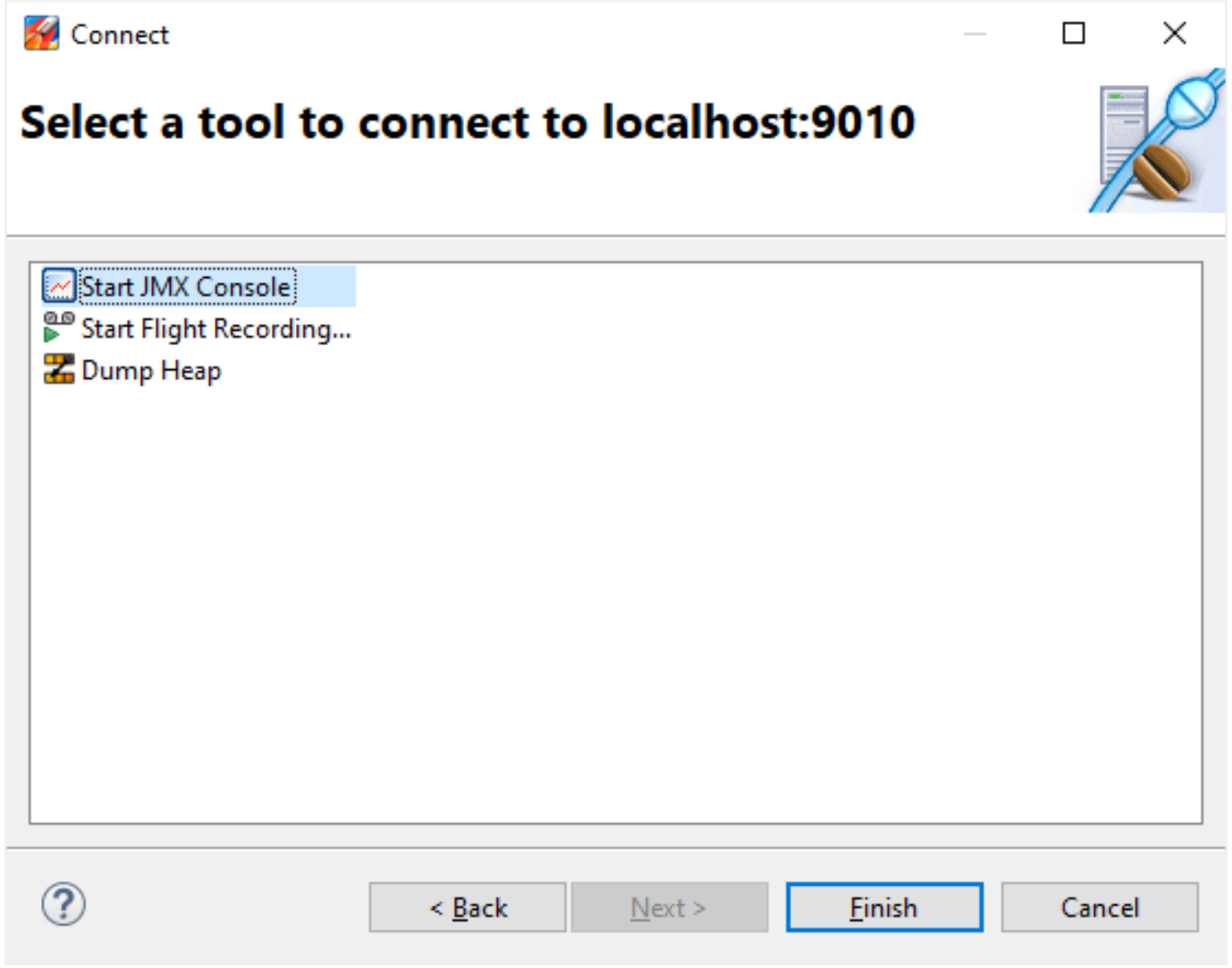
JMC: в меню File / Connect ...



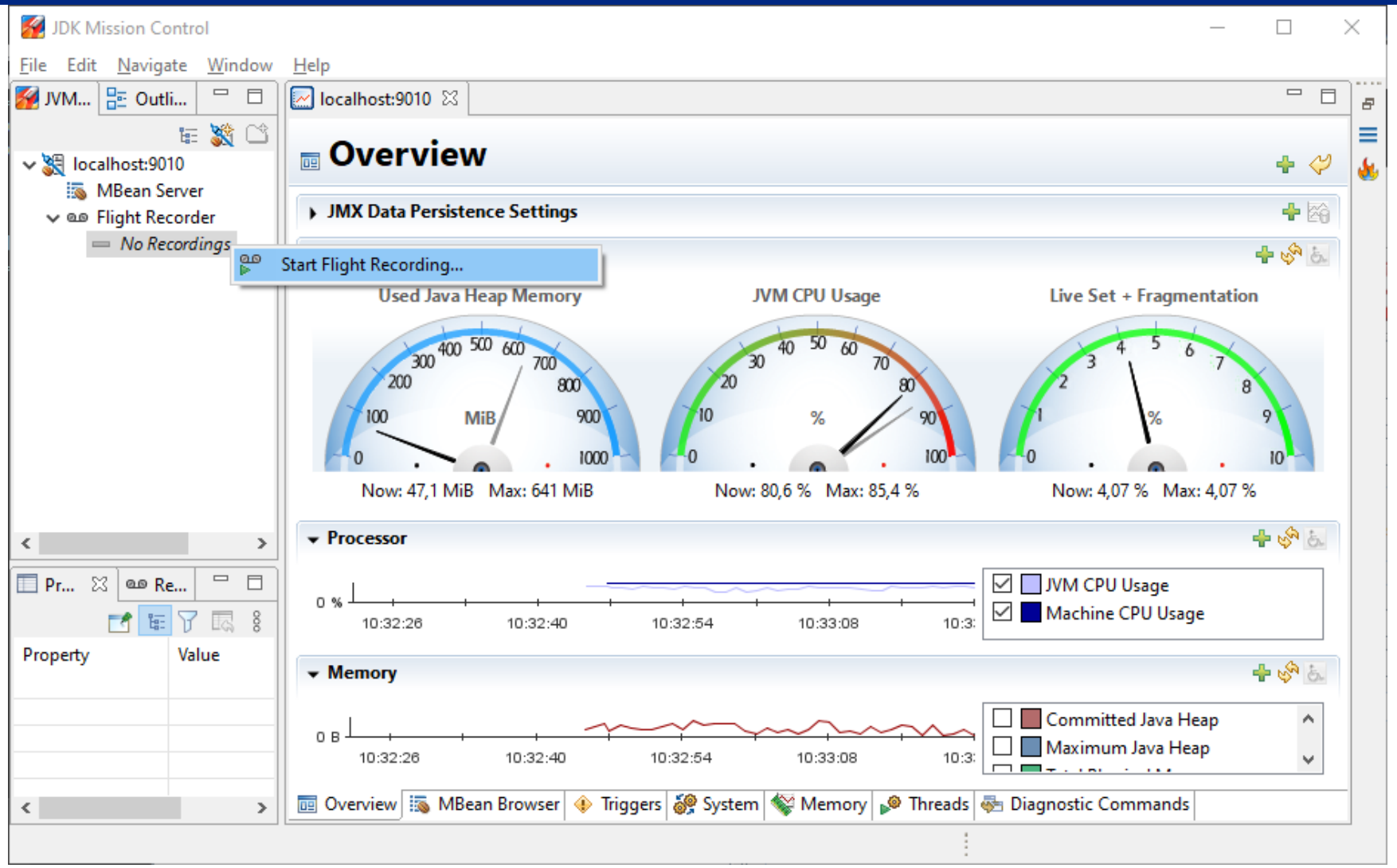
JMC: указать JMX/RMI порт

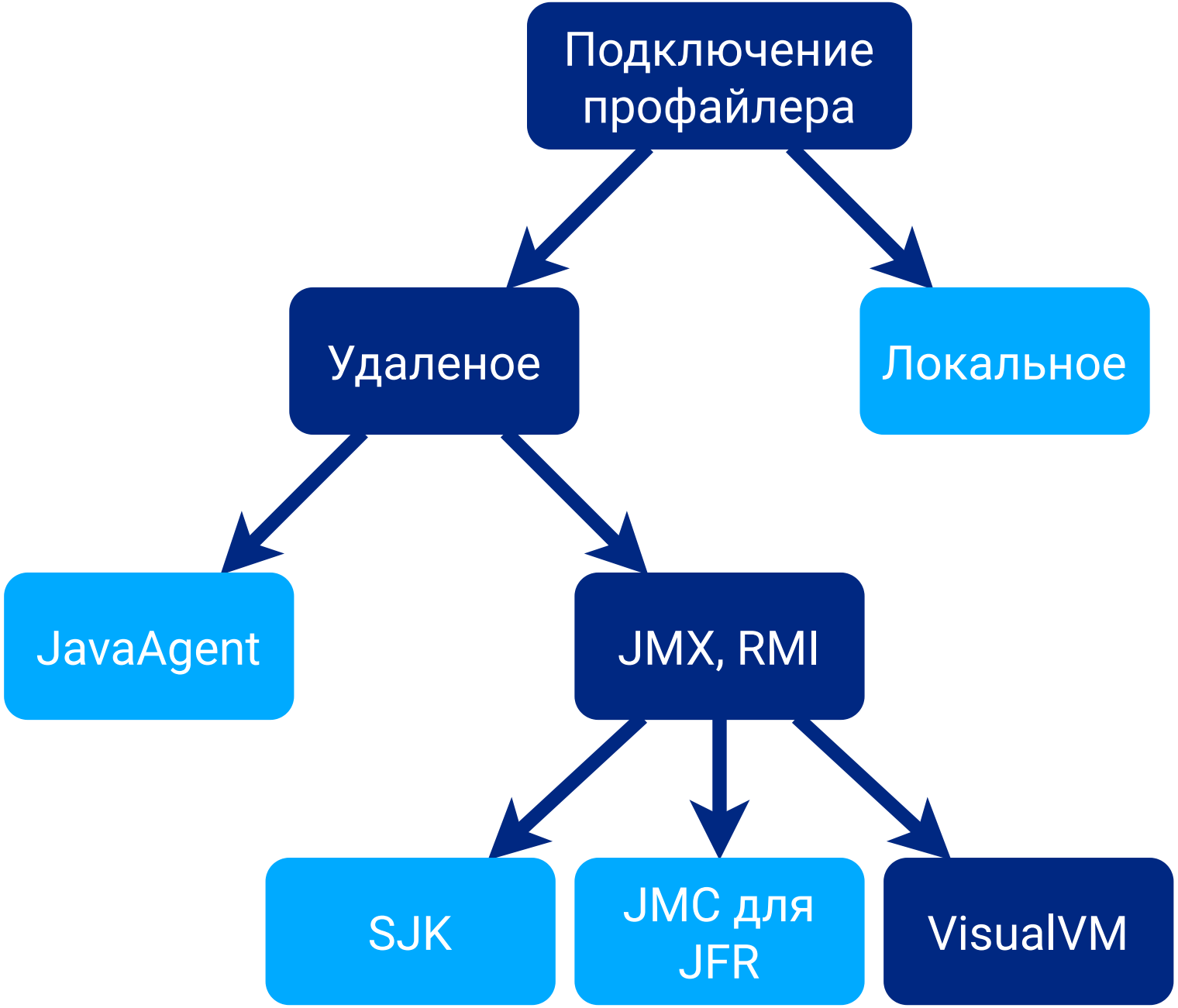


JMC: запустить JMX Console, для проверки

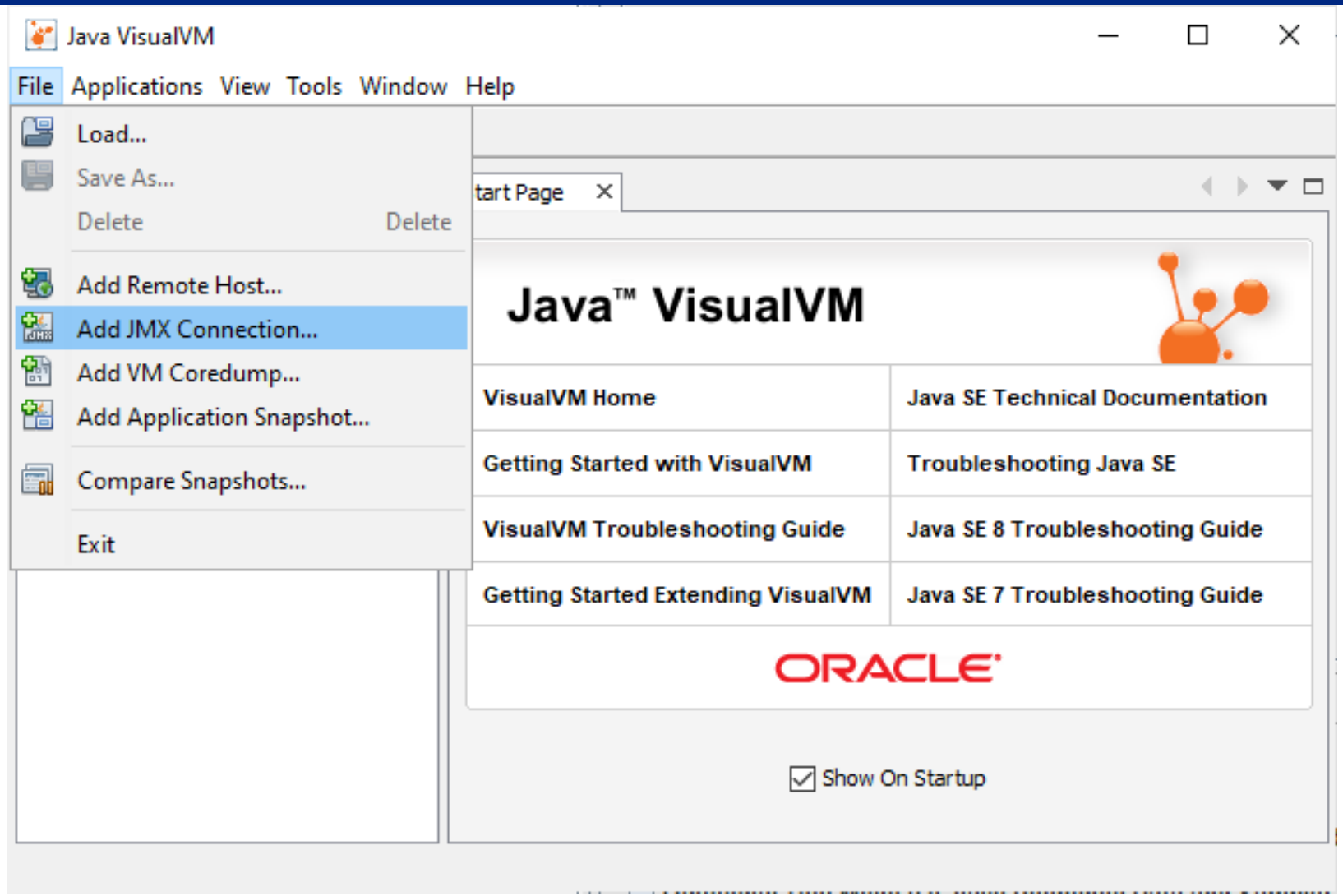


JMC: запустить Flight Recording

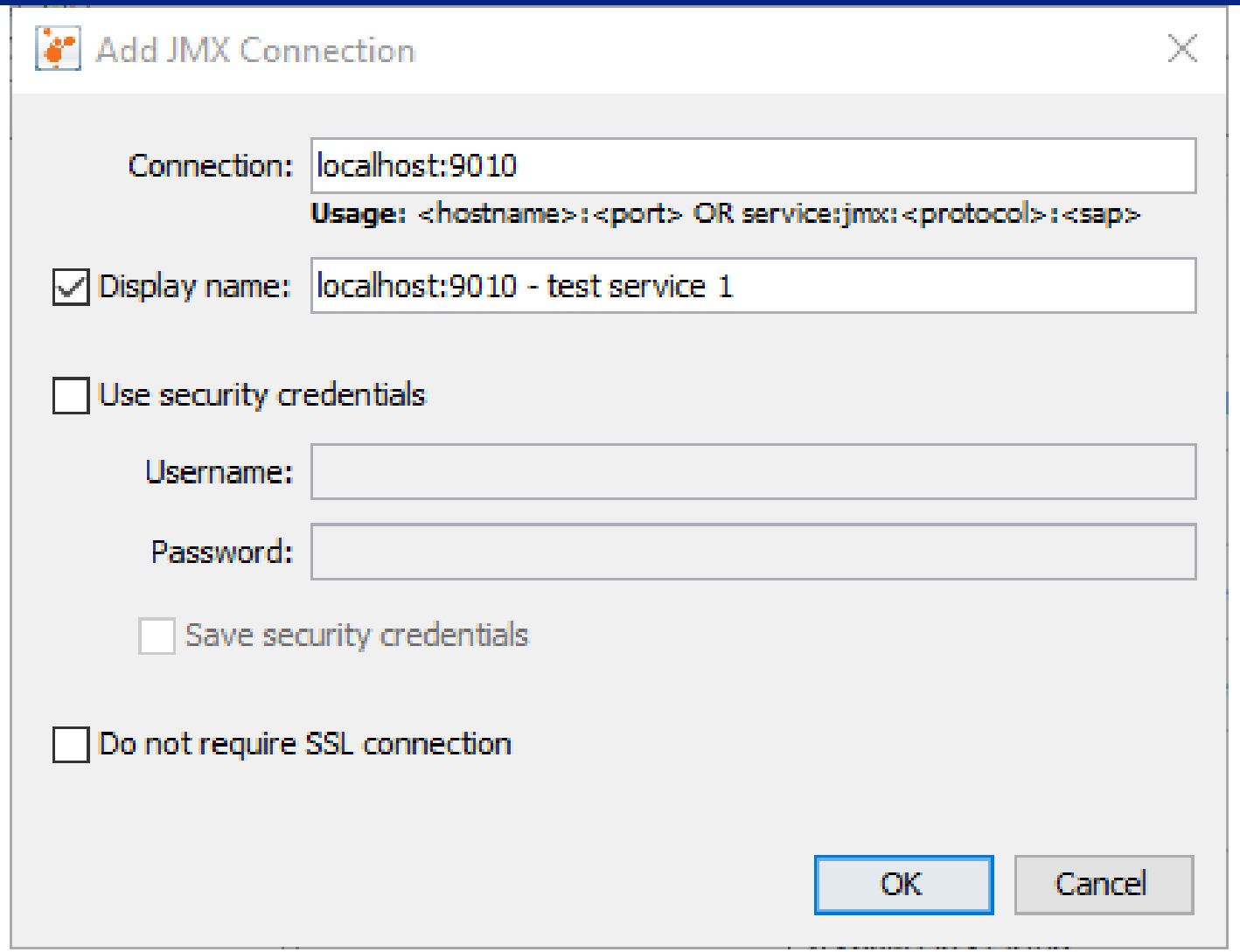



Подключение профайлера
Удаленое
Локальное
JavaAgent
JMX, RMI
SJK
JMC для JFR
VisualVM

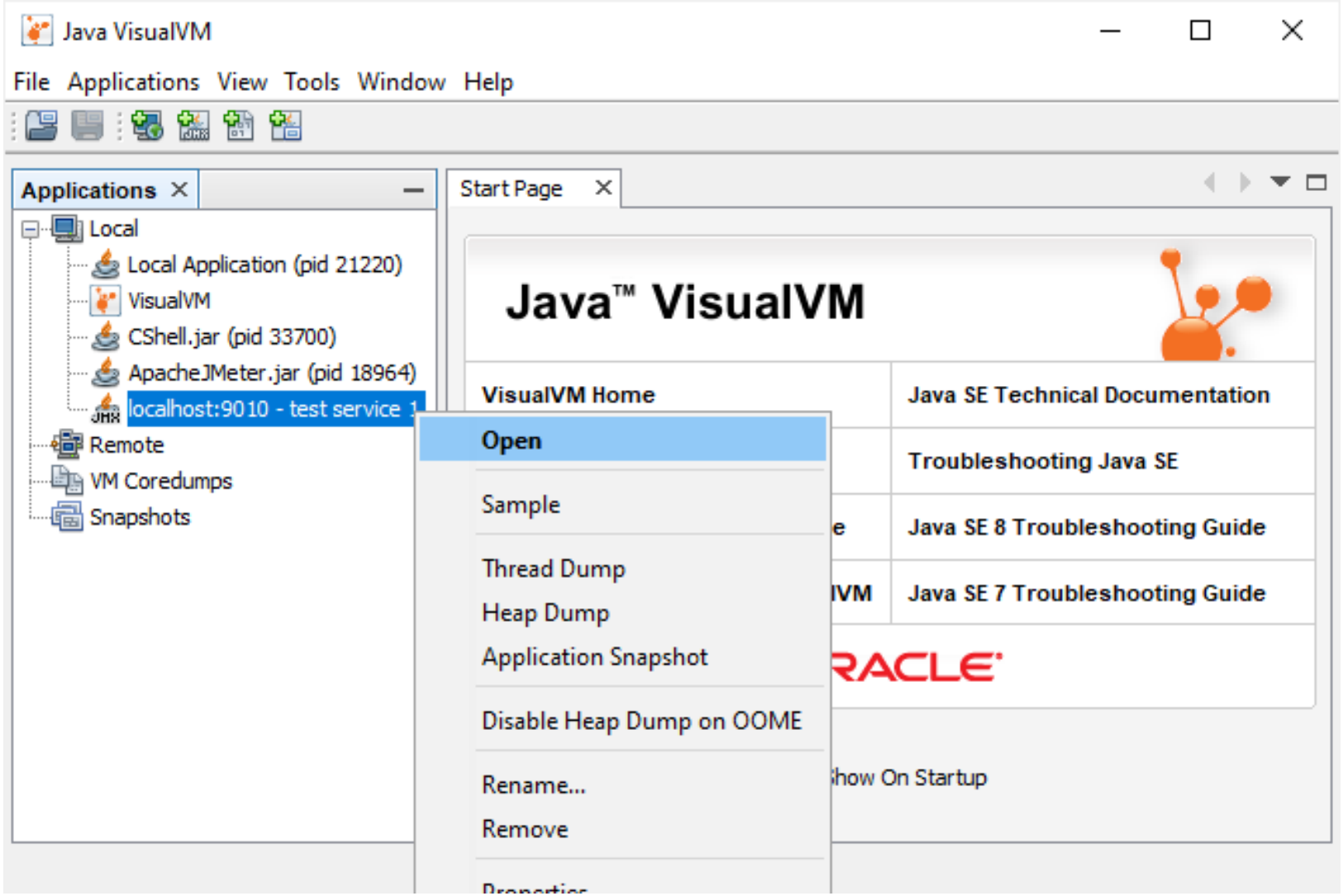
VisualVM: File / Add JMX Connection ...



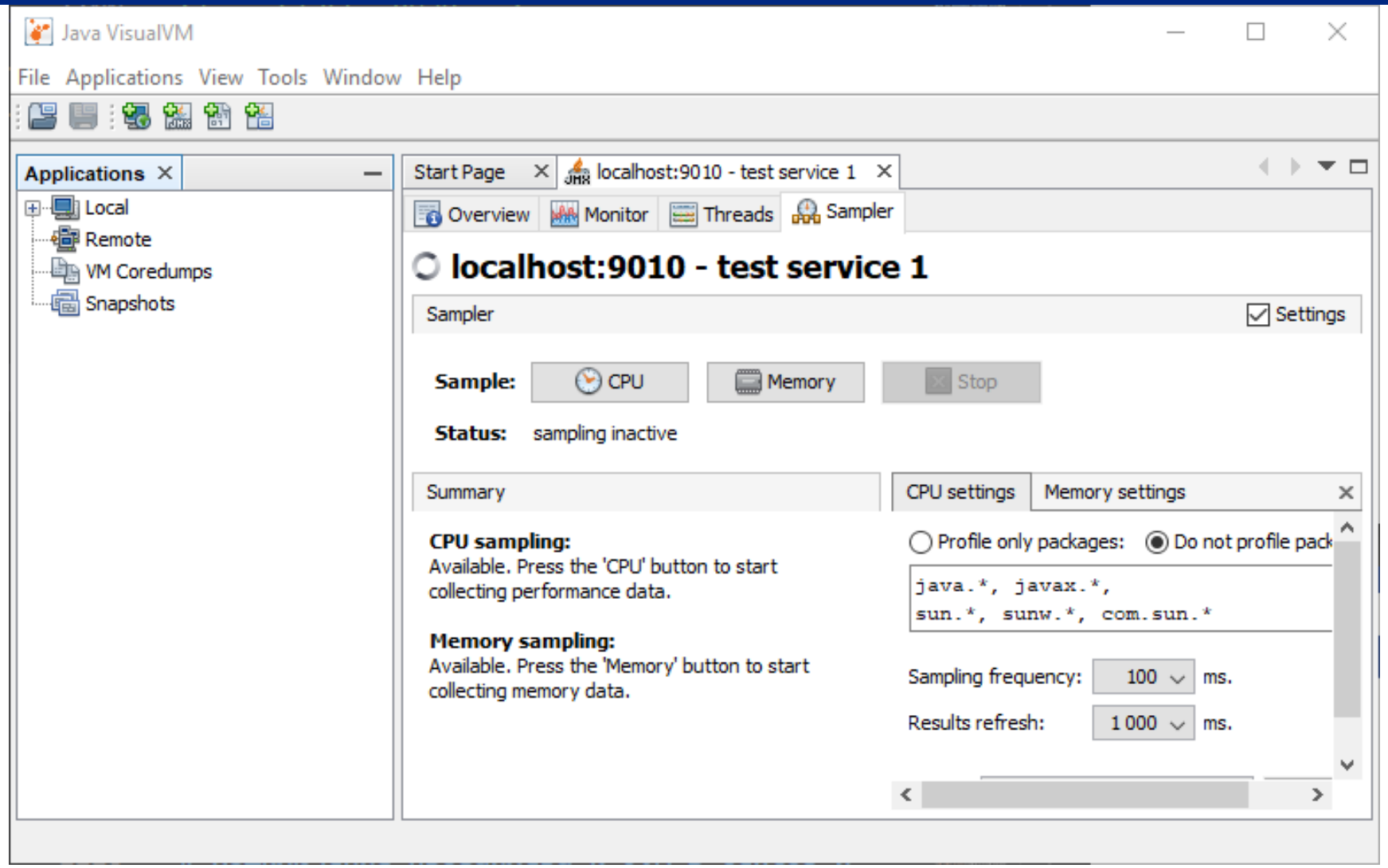
VisualVM: указать JMX-порт и имя подключения

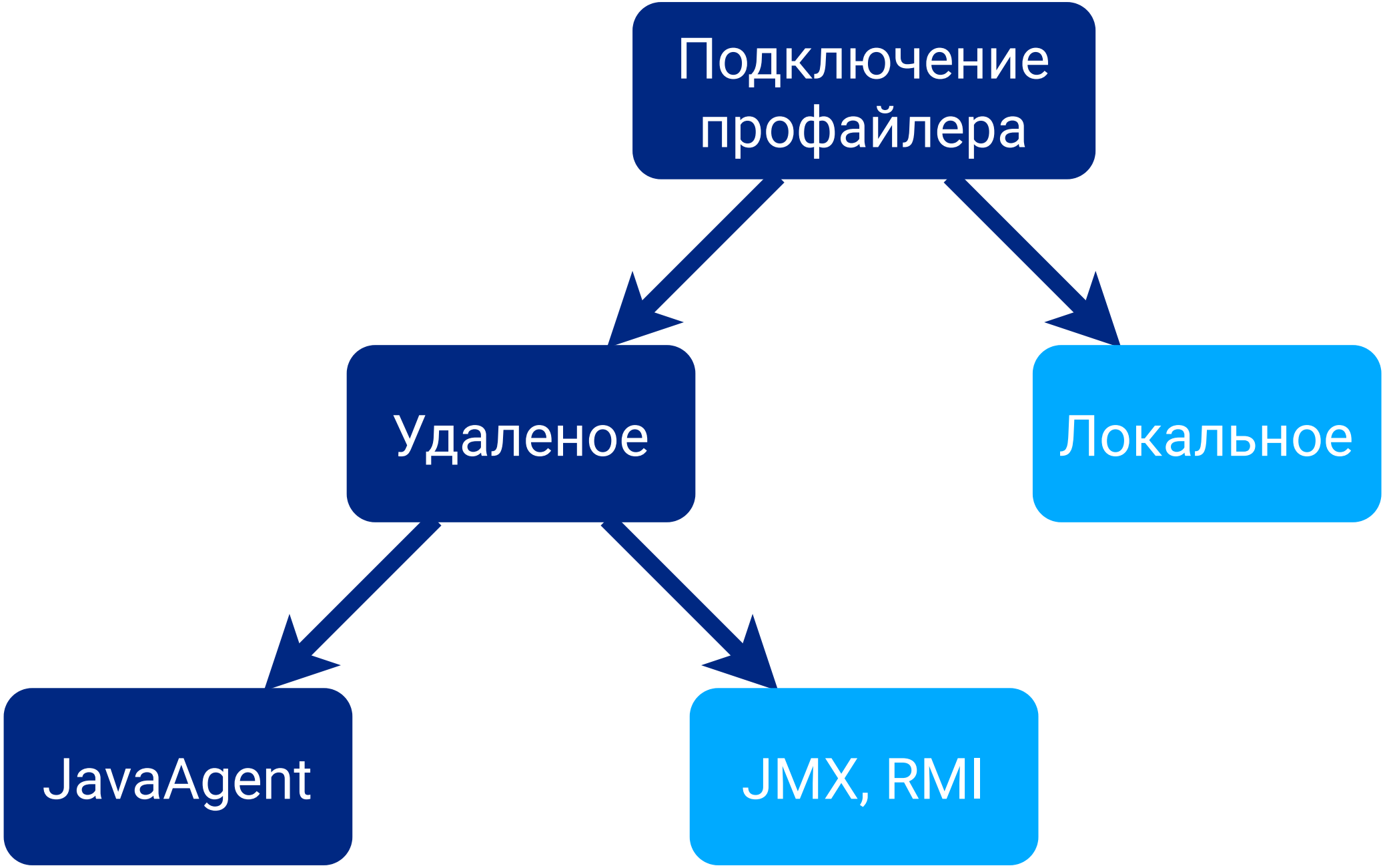


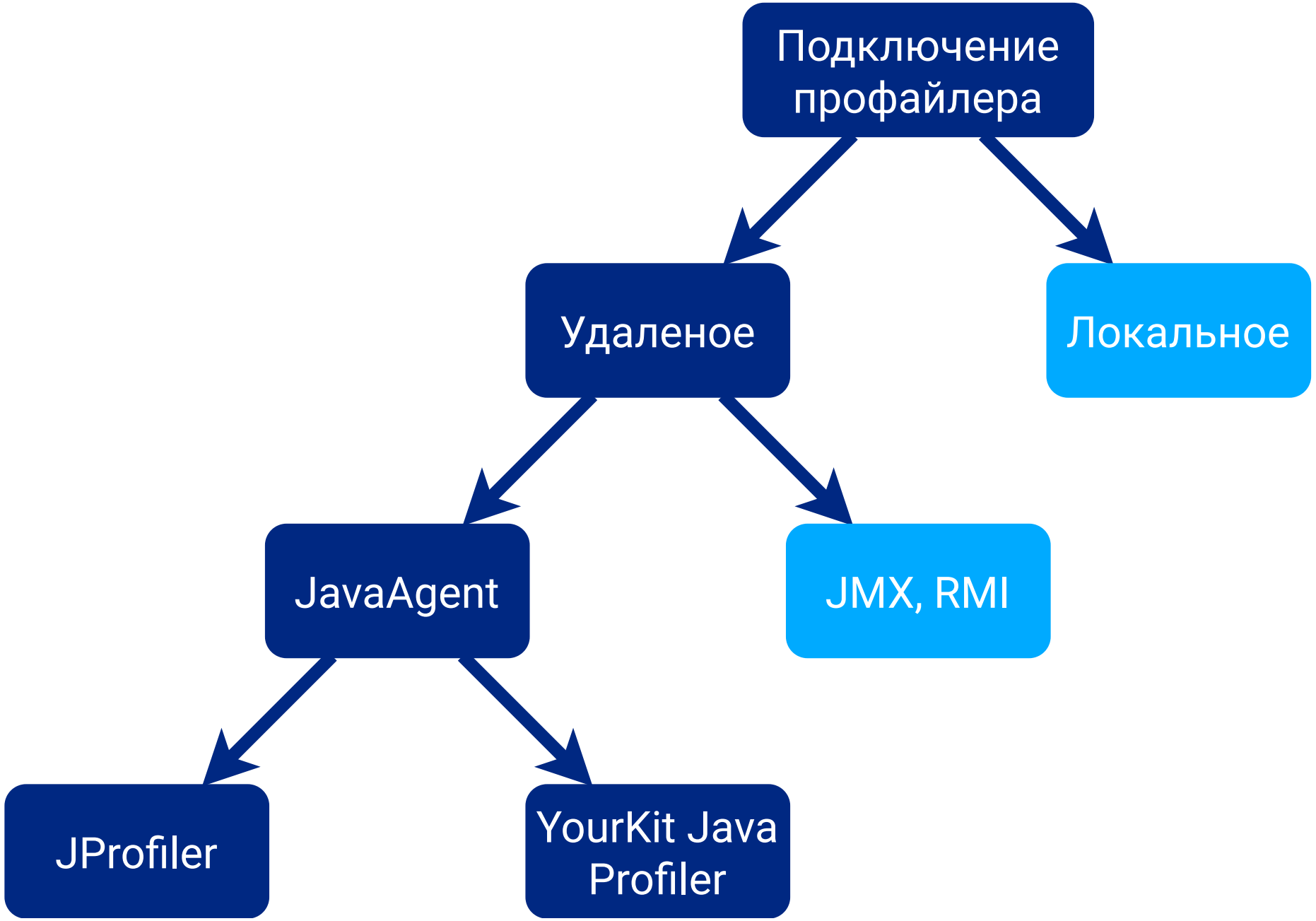
VisualVM: в Applications/Local открыть



VisualVM: на вкладке Sampler нажать CPU



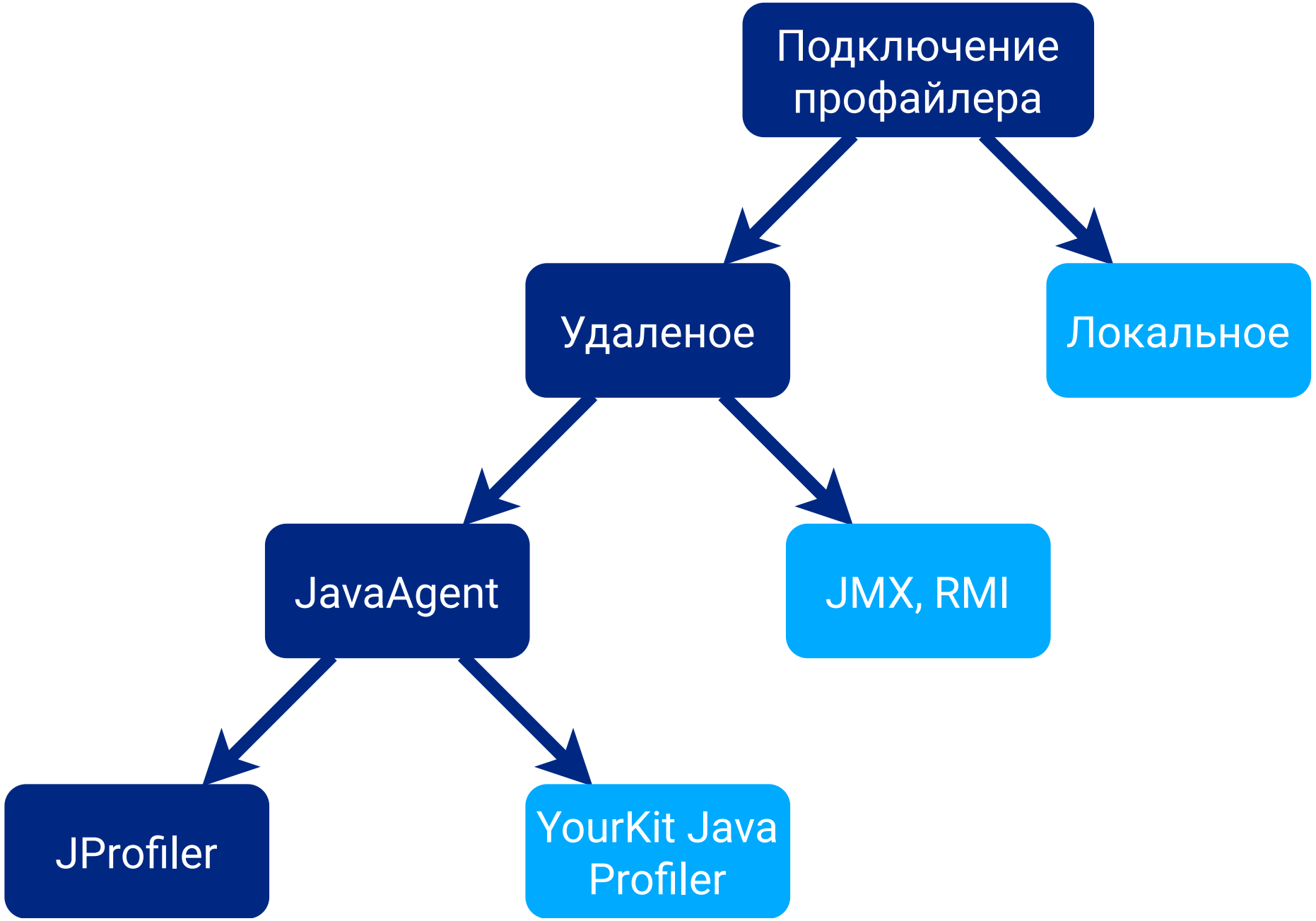

Подключение профайлера
Удаленое
Локальное
JavaAgent
JMX, RMI


Подключение профайлера
Удаленое
Локальное
JavaAgent
JMX, RMI
JProfiler
YourKit Java Profiler

Удаленное подключение к JavaAgent

1. Создать Persistent Storage или каталог на NodeHost (**hostpath**)
2. **Deployments**: смонтировать каталог в поду
3. **Kubectl** (cp): загрузить в каталог профайлер для Linux
4. **Deployments**: указать путь к javaagent и свободный порт
5. **Service**: открыть NodePort или сделать Port Forward (**Kubectl**)
6. Запустить профайлер локально (**JProfiler**, **YourKit**)
7. Создать удаленное подключение и профилировать

- Всего один сетевой порт
- Можно подключиться к 2+ подам одного сервиса


Подключение профайлера
Удаленое
Локальное
JavaAgent
JMX, RMI
JProfiler
YourKit Java Profiler

ej-technologies.com/download/jprofiler/files

64-BIT WINDOWS

Setup Executable with JRE (108 MB)

ZIP Archive (173 MB)

Supported architectures: x64/AMD64

Supported versions: 10/8/7, Server 2019/2016/2012

ej-technologies.com/download/jprofiler/files

LINUX

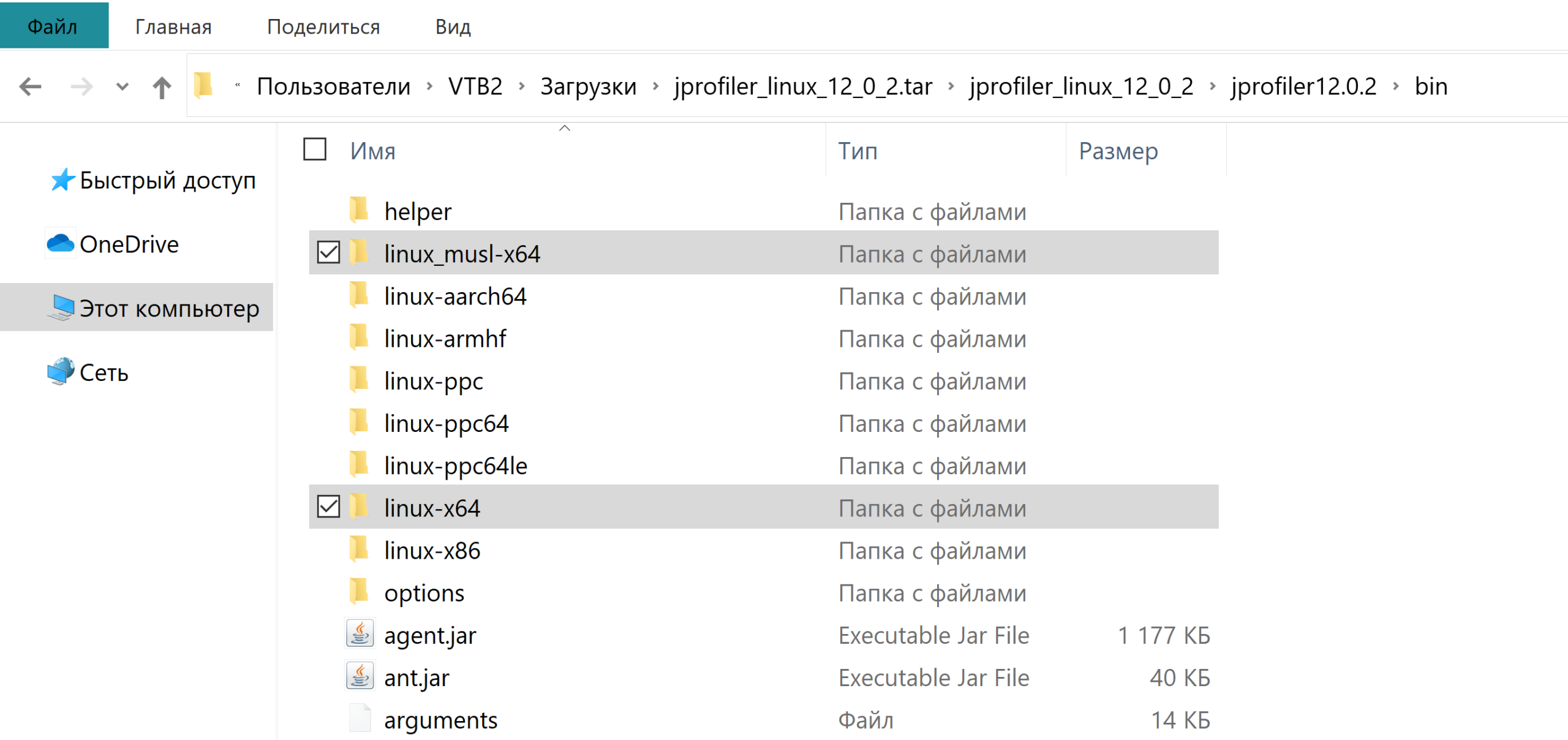
Setup Executable (64 MB)

RPM (107 MB)

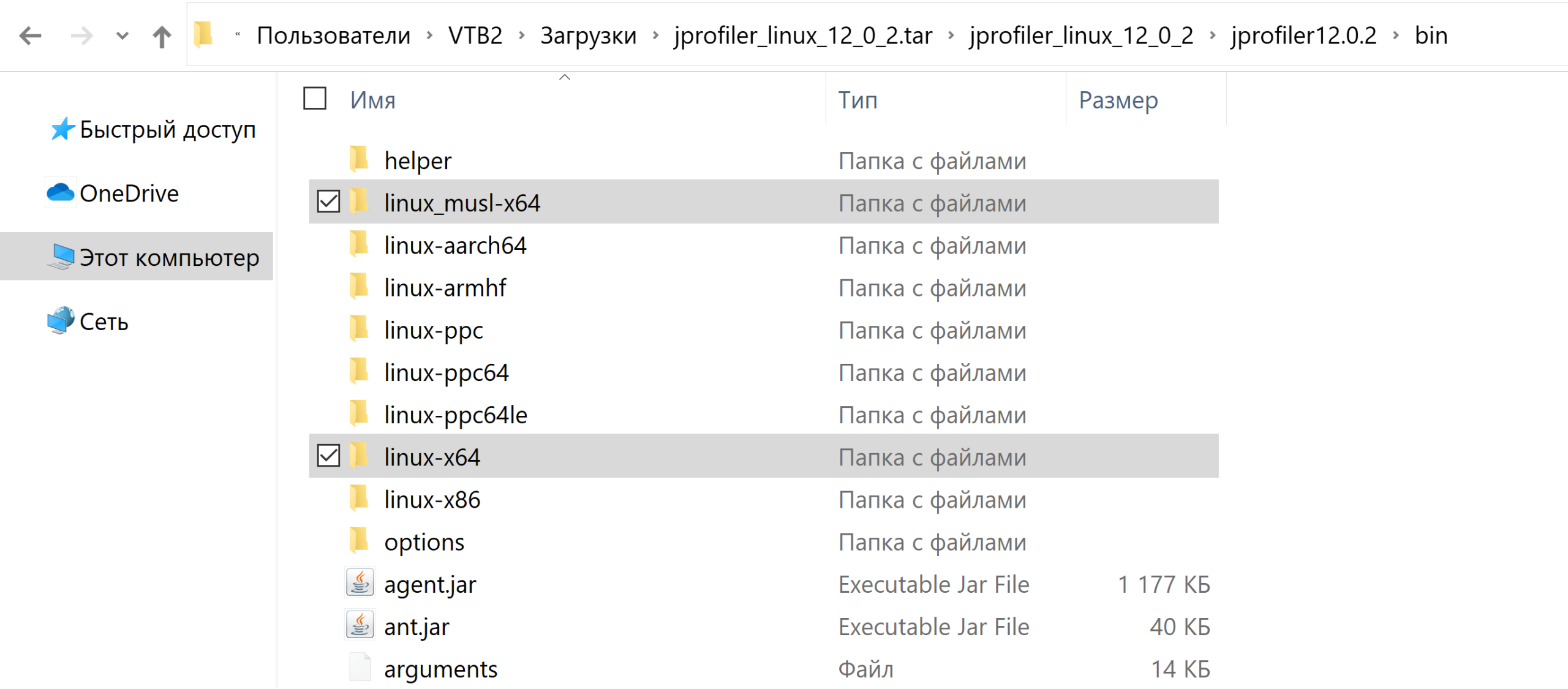
DEB (107 MB)

TAR.GZ Archive (107 MB)

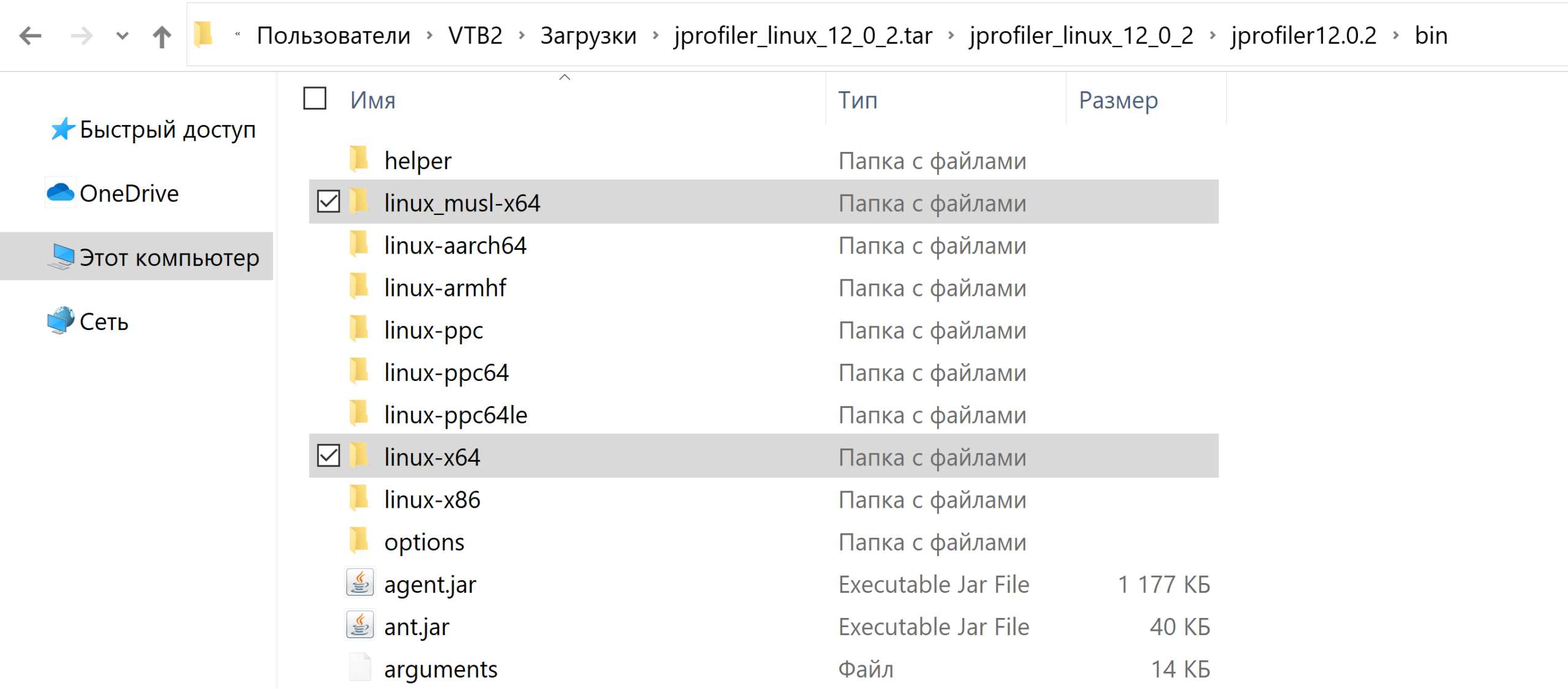
Supported architectures: x86, x64/AMD64, PPC, PPC64, PPC64LE, ARMv7, ARMv8


Файл
Главная
Поделиться
Вид
Пользователи › VTB2 › Загрузки › jprofiler_linux_12_0_2.tar › jprofiler_linux_12_0_2 › jprofiler12.0.2 › bin
Быстрый доступ
OneDrive
Этот компьютер
Сеть
Имя
Тип
Размер
helper
Папка с файлами
linux_musl-x64
Папка с файлами
linux-aarch64
Папка с файлами
linux-armhf
Папка с файлами
linux-ppc
Папка с файлами
linux-ppc64
Папка с файлами
linux-ppc64le
Папка с файлами
linux-x64
Папка с файлами
linux-x86
Папка с файлами
options
Папка с файлами
agent.jar
Executable Jar File
1 177 КБ
ant.jar
Executable Jar File
40 КБ
arguments
Файл
14 КБ

Для Alpine Linux: linux_musl-x64



Для CentOS, RHEL и других: linux-x64



В Pods / Exec посмотреть на версию ОС

Для Alpine получится так

```
$ cat /etc/os-release
NAME="Alpine Linux"
ID=alpine
...
```

Для других ОС получится другое имя

```
$ cat /etc/os-release
NAME="CentOS Linux"
VERSION="7 (Core)"
...
```

В Deployment смонтировать и подключить агент

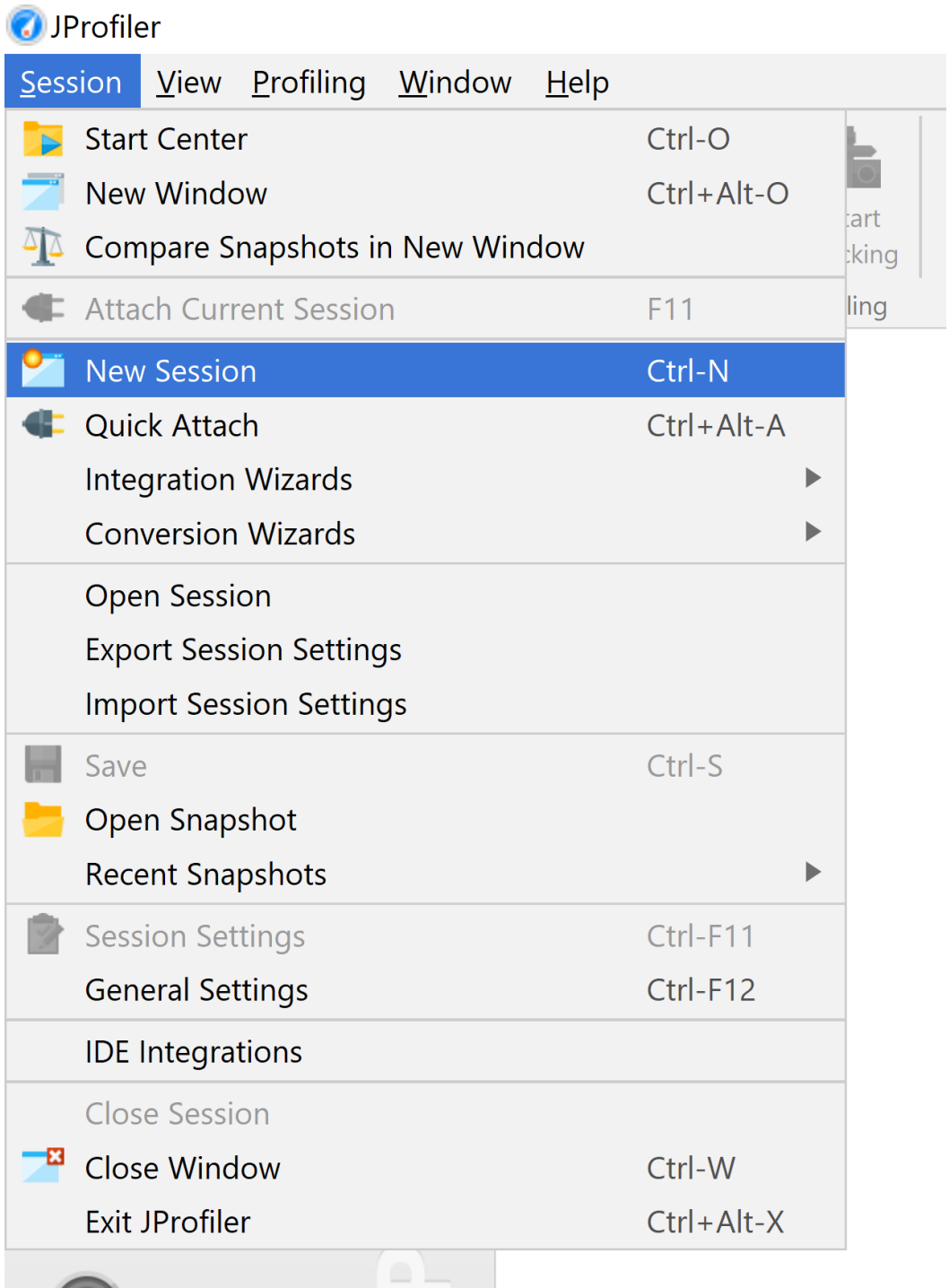
```
kind: Deployment
spec:
  template:
    spec:
      volumes:
        - name: jprofiler
          hostPath:
            path: /opt/data/jprofiler12.0.2/bin/linux_musl-x64 # на диске
      containers:
        - name: test-webserver
          env:
            - name: JAVA_OPTIONS                                    # свободный порт
              value: -agentpath:/tmp/jprofiler/libjprofilerti.so=port=8849,nowait
          volumeMounts:
            - name: jprofiler
              mountPath: /tmp/jprofiler                             # в поде
```

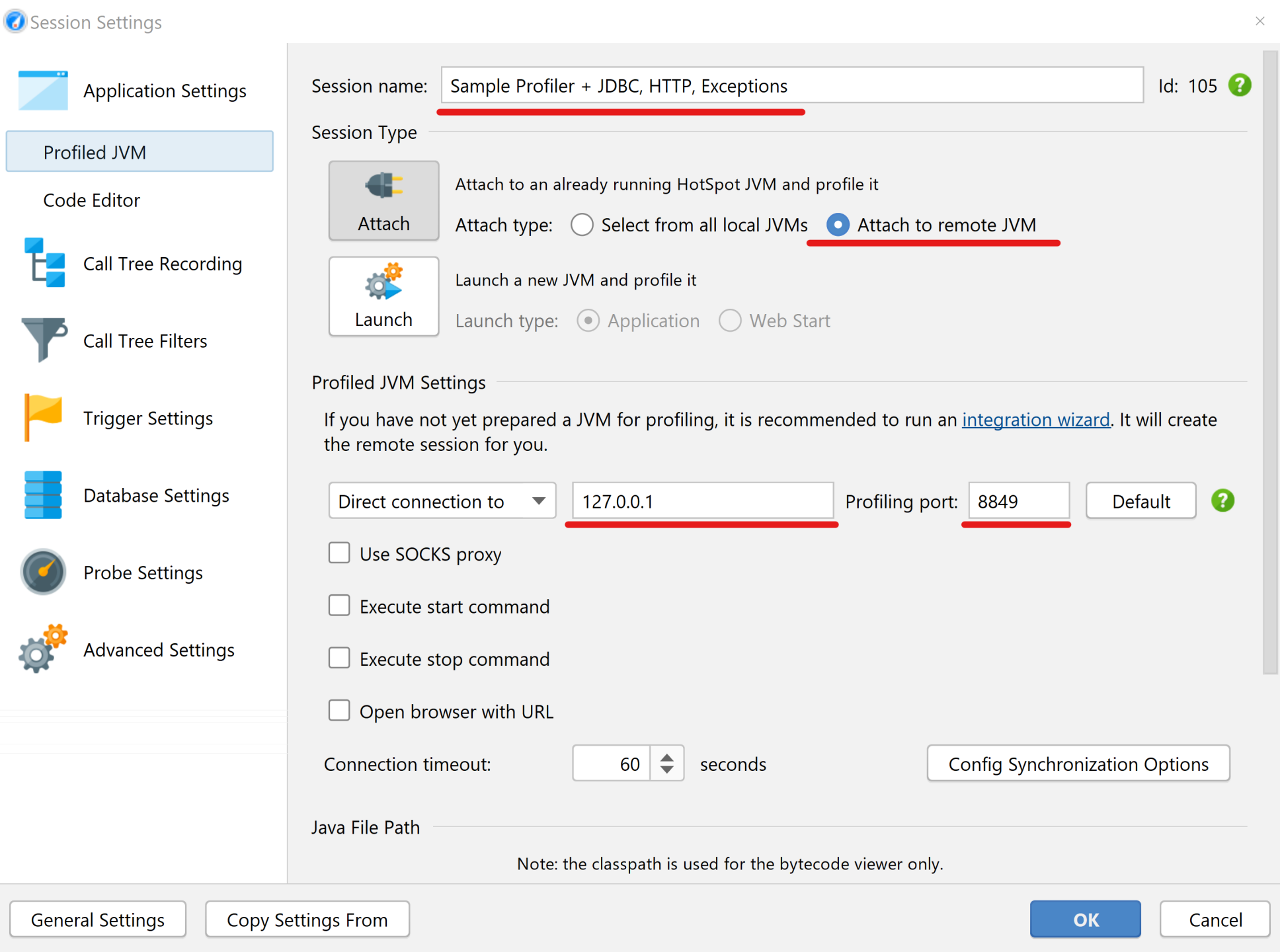
Для CentOS: linux-x64 вместо linux_musl-x64

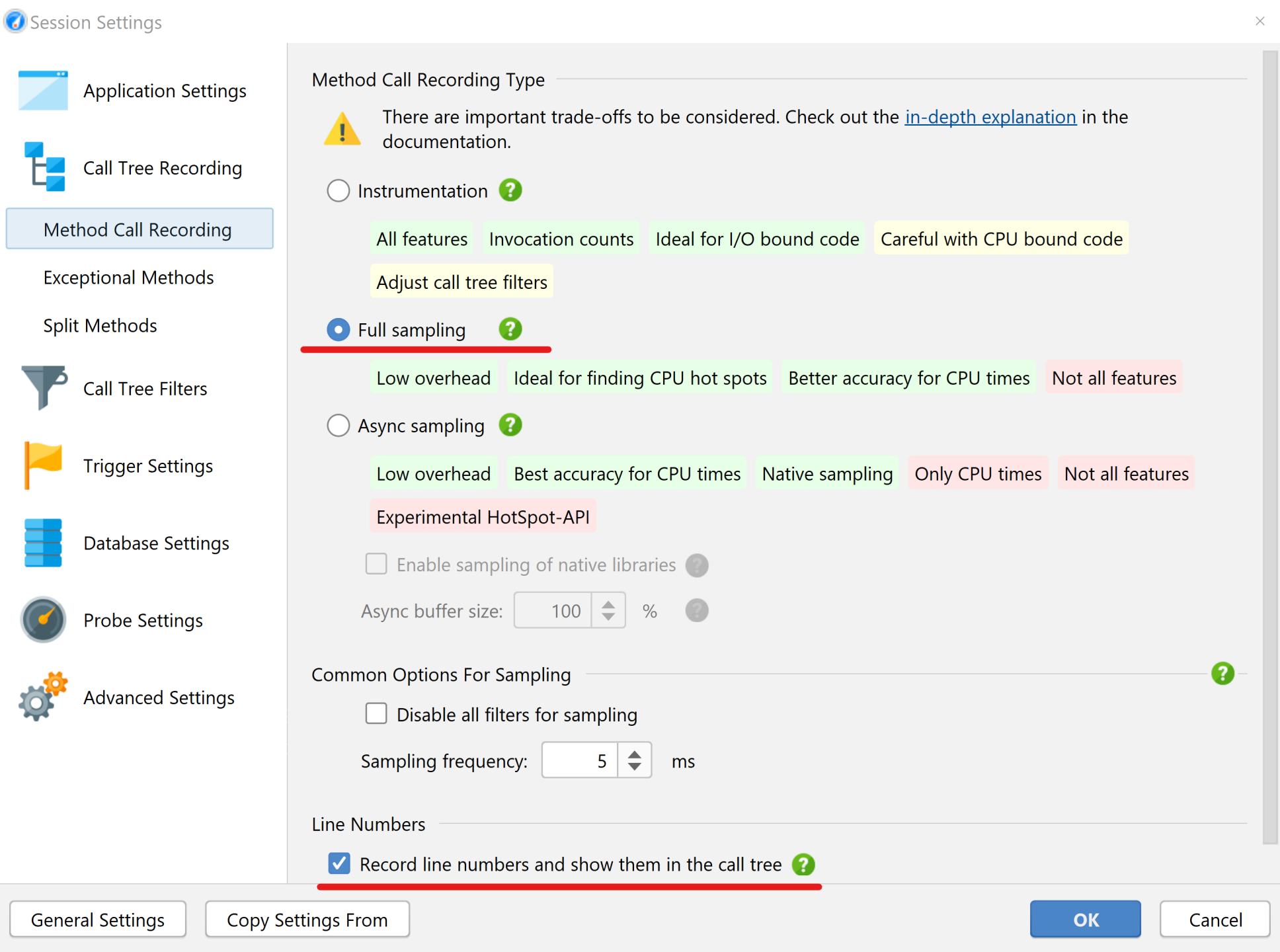
```
kind: Deployment
spec:
  template:
    spec:
      volumes:
        - name: jprofiler
          hostPath:
            path: /opt/data/jprofiler12.0.2/bin/linux-x64 # просто linux-x64
      containers:
        - name: test-webserver
          env:
            - name: JAVA_OPTIONS
              value: -agentpath:/tmp/jprofiler/libjprofilerti.so=port=8849,nowait
          volumeMounts:
            - name: jprofiler
              mountPath: /tmp/jprofiler
```

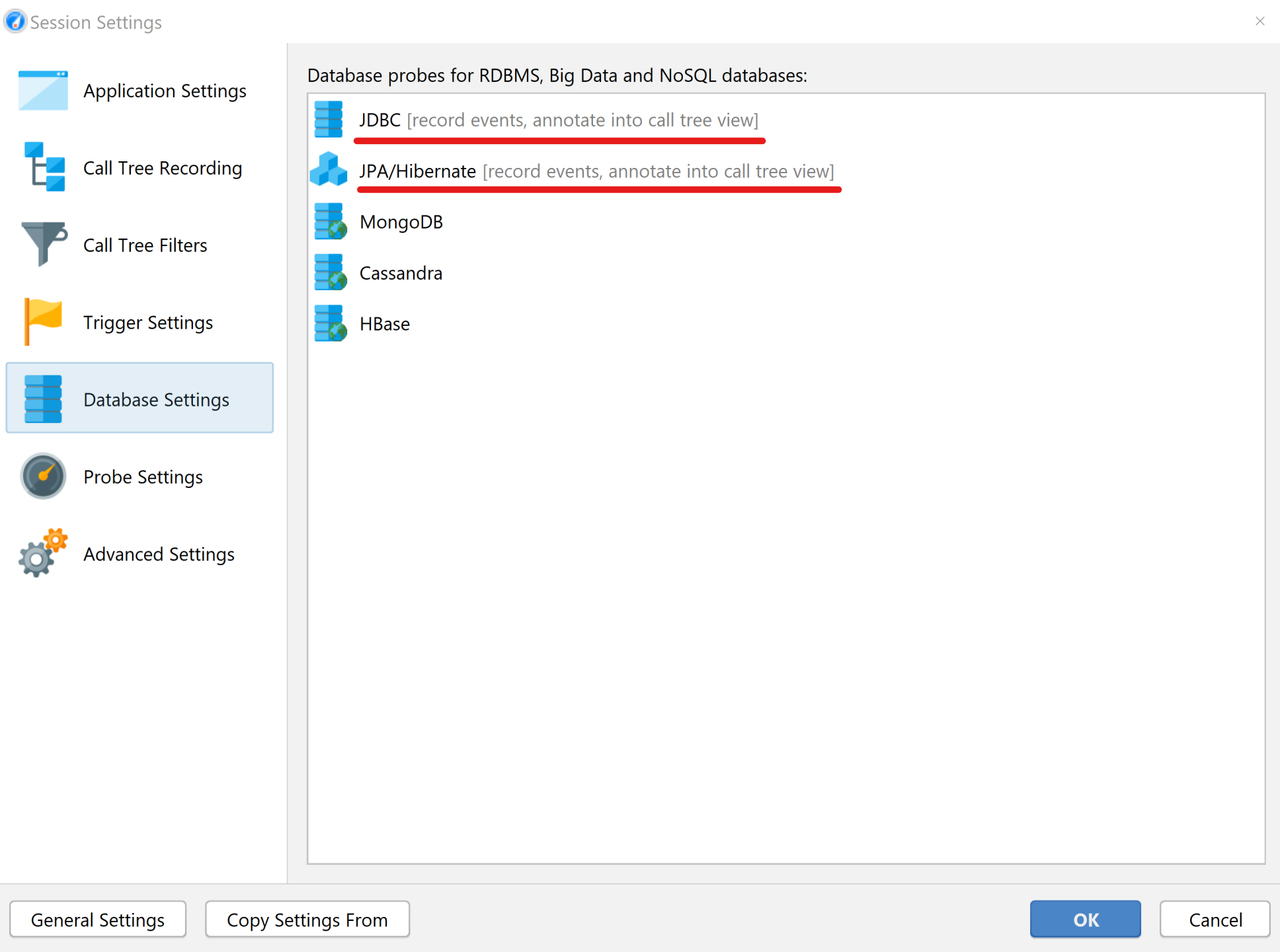
Для другого сервиса можно оставить все также

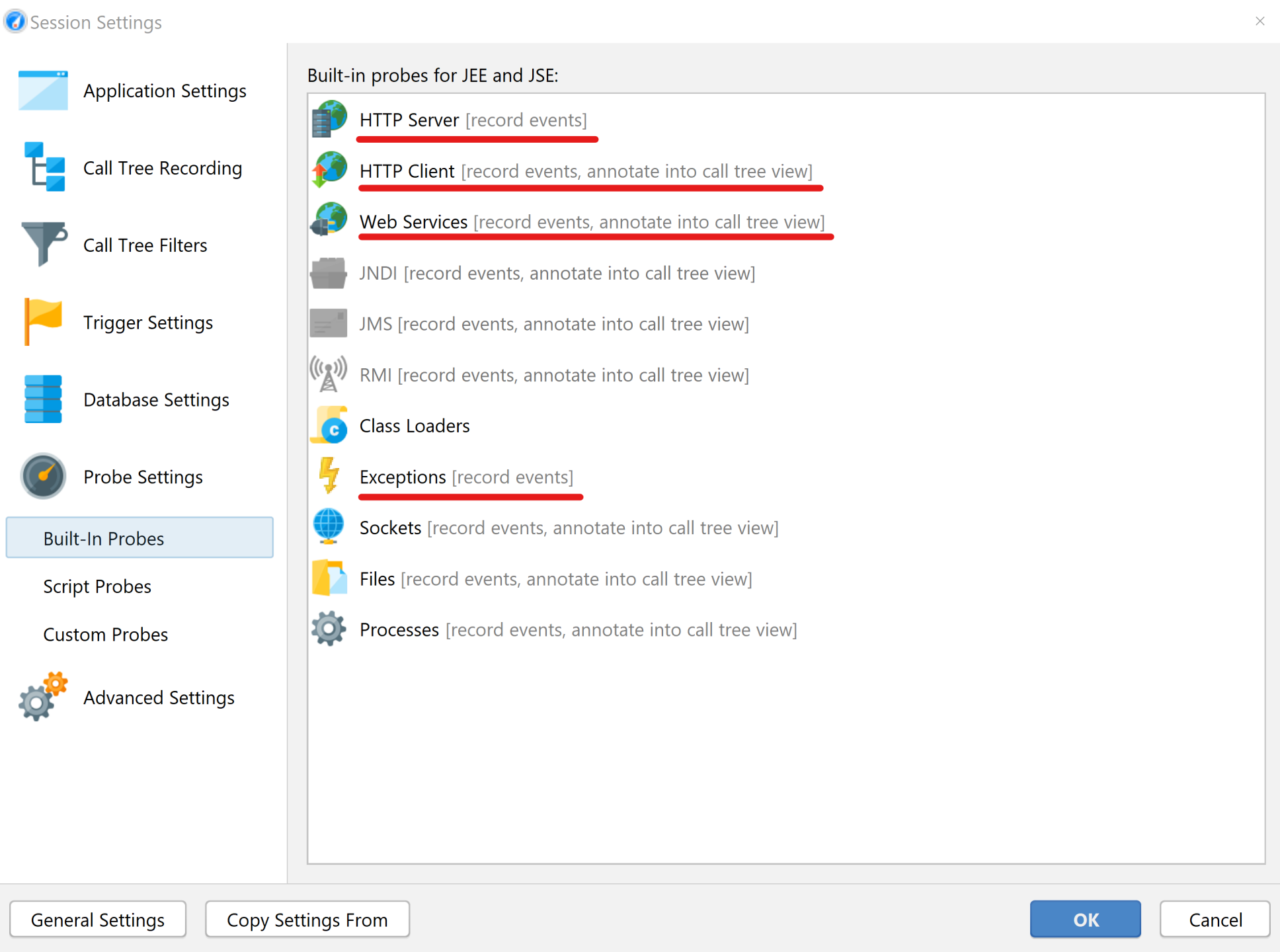
```
kind: Deployment
spec:
  template:
    spec:
      volumes:
        - name: jprofiler
          hostPath:
            path: /opt/data/jprofiler12.0.2/bin/linux-x64 # или linux_musl-x64
      containers:
        - name: test-webserver-other
          env:
            - name: JAVA_OPTIONS
              value: -agentpath:/tmp/jprofiler/libjprofilerti.so=port=8849,nowait
          volumeMounts:
            - name: jprofiler
              mountPath: /tmp/jprofiler
```

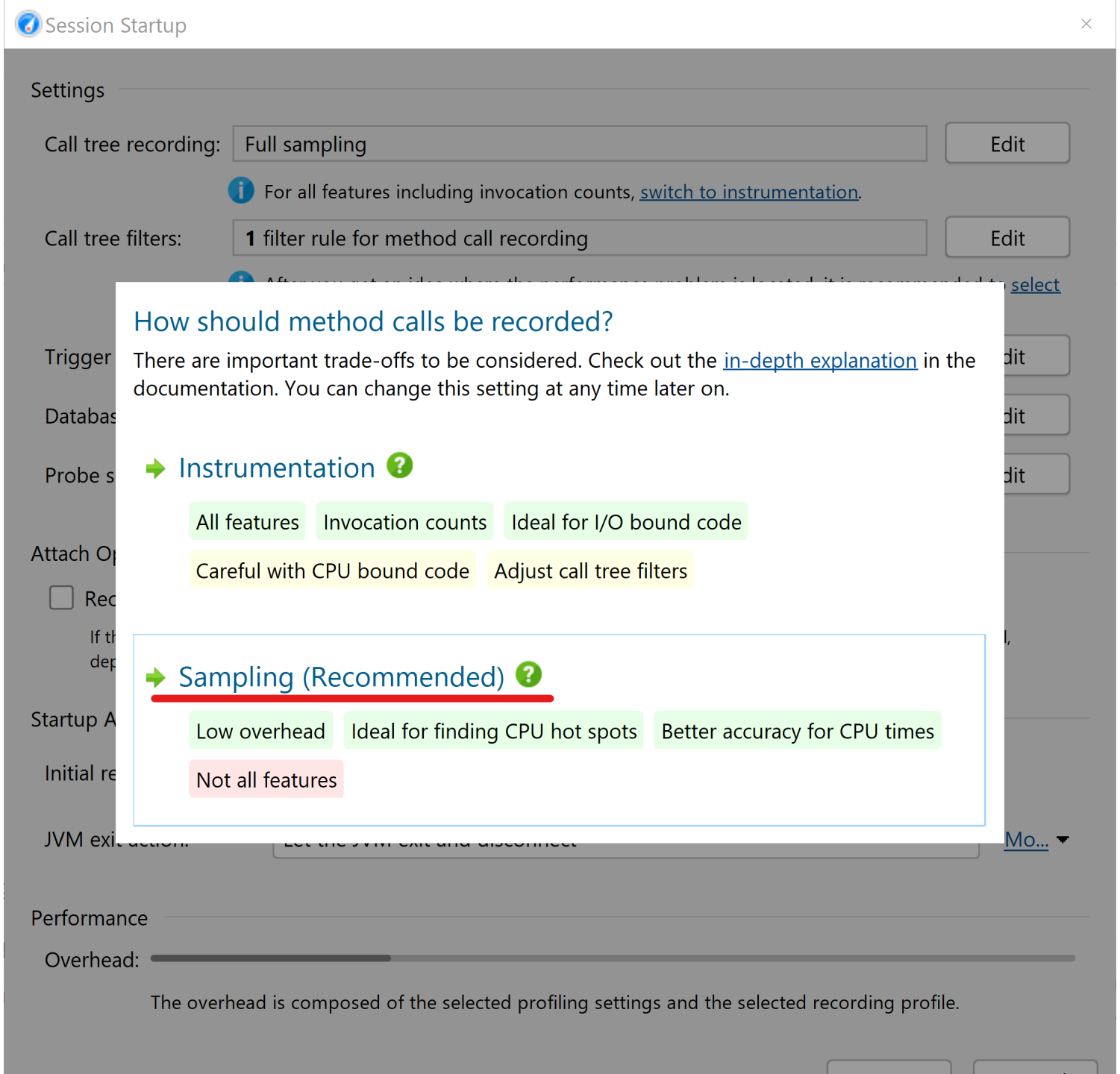

JProfiler
Session
View
Profiling
Window
Help
Start Center Ctrl-O
New Window Ctrl+Alt-O
Compare Snapshots in New Window
Attach Current Session F11
New Session Ctrl-N
Quick Attach Ctrl+Alt-A
Integration Wizards
Conversion Wizards
Open Session
Export Session Settings
Import Session Settings
Save Ctrl-S
Open Snapshot
Recent Snapshots
Session Settings Ctrl-F11
General Settings Ctrl-F12
IDE Integrations
Close Session
Close Window Ctrl-W
Exit JProfiler Ctrl+Alt-X

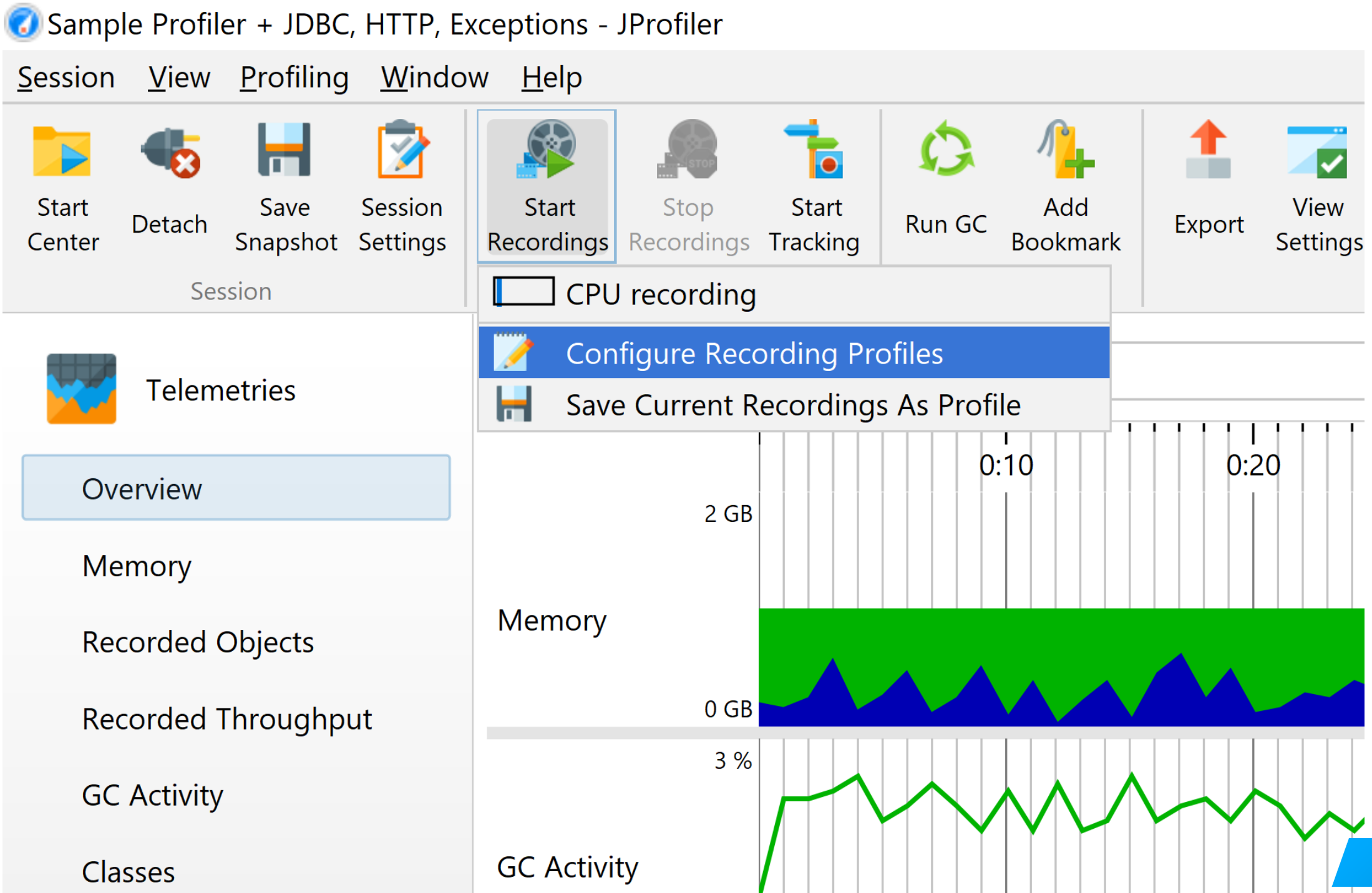

Session Settings
Application Settings
Profiled JVM
Code Editor
Call Tree Recording
Call Tree Filters
Trigger Settings
Database Settings
Probe Settings
Advanced Settings
Session name: Sample Profiler + JDBC, HTTP, Exceptions
Id: 105
Session Type
Attach
Attach to an already running HotSpot JVM and profile it
Attach type: Select from all local JVMs Attach to remote JVM
Launch
Launch a new JVM and profile it
Launch type: Application Web Start
Profiled JVM Settings
If you have not yet prepared a JVM for profiling, it is recommended to run an integration wizard. It will create the remote session for you.
Direct connection to
127.0.0.1
Profiling port: 8849
Default
Use SOCKS proxy
Execute start command
Execute stop command
Open browser with URL
Connection timeout: 60 seconds
Config Synchronization Options
Java File Path
Note: the classpath is used for the bytecode viewer only.
General Settings
Copy Settings From
OK
Cancel

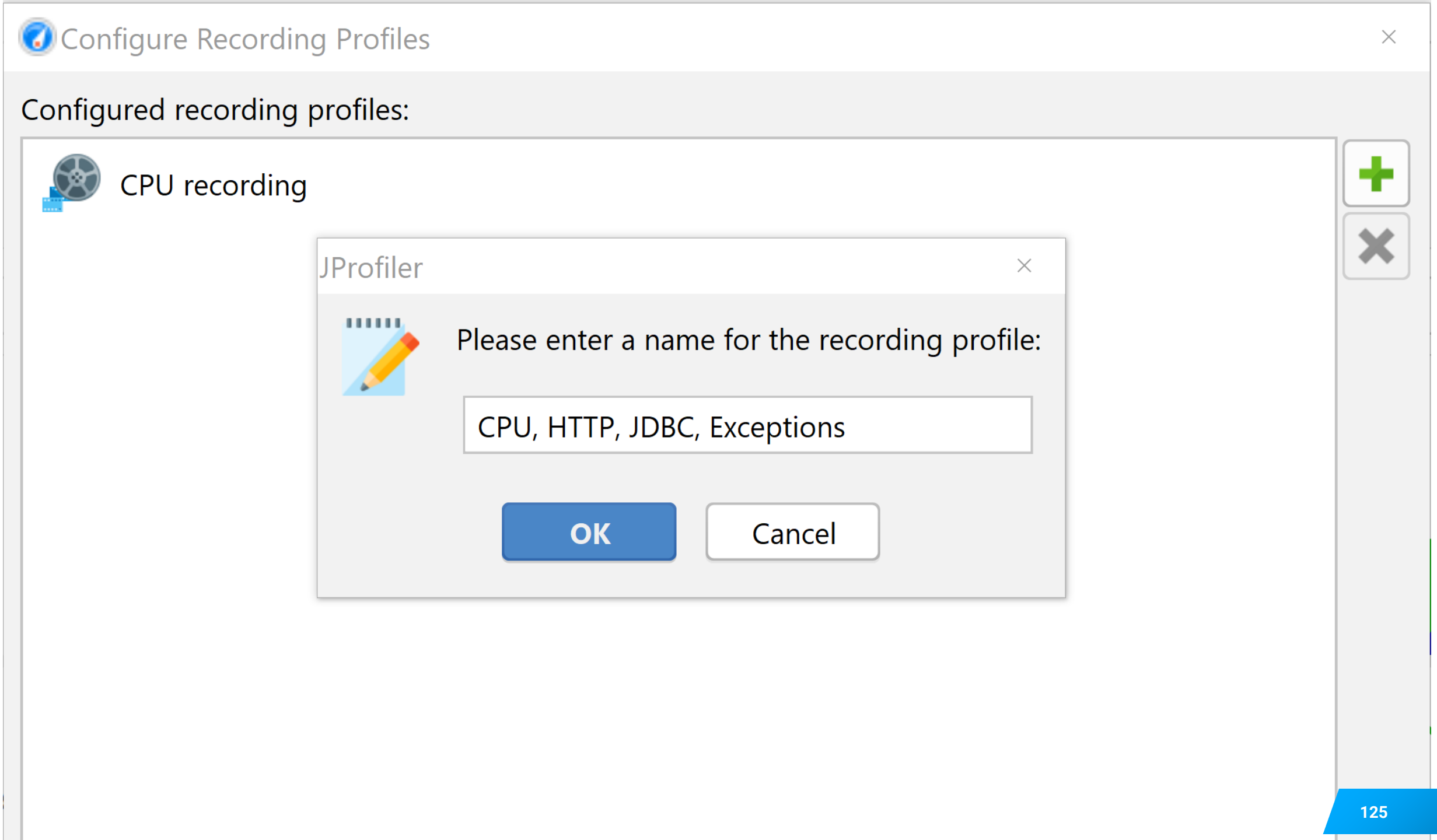

Session Settings
Application Settings
Call Tree Recording
Method Call Recording
Exceptional Methods
Split Methods
Call Tree Filters
Trigger Settings
Database Settings
Probe Settings
Advanced Settings
Method Call Recording Type
There are important trade-offs to be considered. Check out the in-depth explanation in the documentation.
Instrumentation
All features
Invocation counts
Ideal for I/O bound code
Careful with CPU bound code
Adjust call tree filters
Full sampling
Low overhead
Ideal for finding CPU hot spots
Better accuracy for CPU times
Not all features
Async sampling
Low overhead
Best accuracy for CPU times
Native sampling
Only CPU times
Not all features
Experimental HotSpot-API
Enable sampling of native libraries
Async buffer size:
100
%
Common Options For Sampling
Disable all filters for sampling
Sampling frequency:
5
ms
Line Numbers
Record line numbers and show them in the call tree
General Settings
Copy Settings From
OK
Cancel

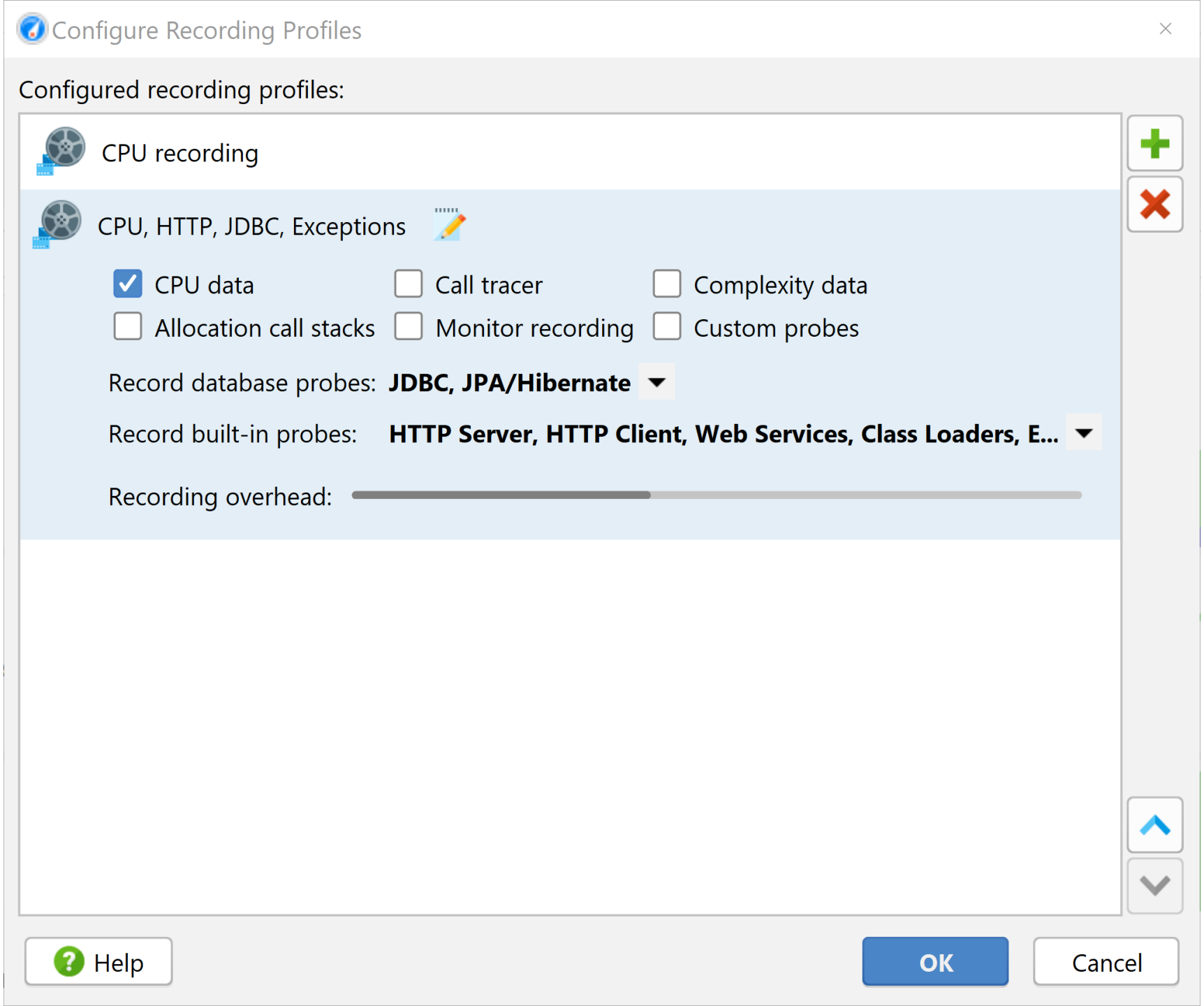

Session Settings
Application Settings
Call Tree Recording
Call Tree Filters
Trigger Settings
Database Settings
Probe Settings
Advanced Settings
Database probes for RDBMS, Big Data and NoSQL databases:
JDBC [record events, annotate into call tree view]
JPA/Hibernate [record events, annotate into call tree view]
MongoDB
Cassandra
HBase
General Settings
Copy Settings From
OK
Cancel

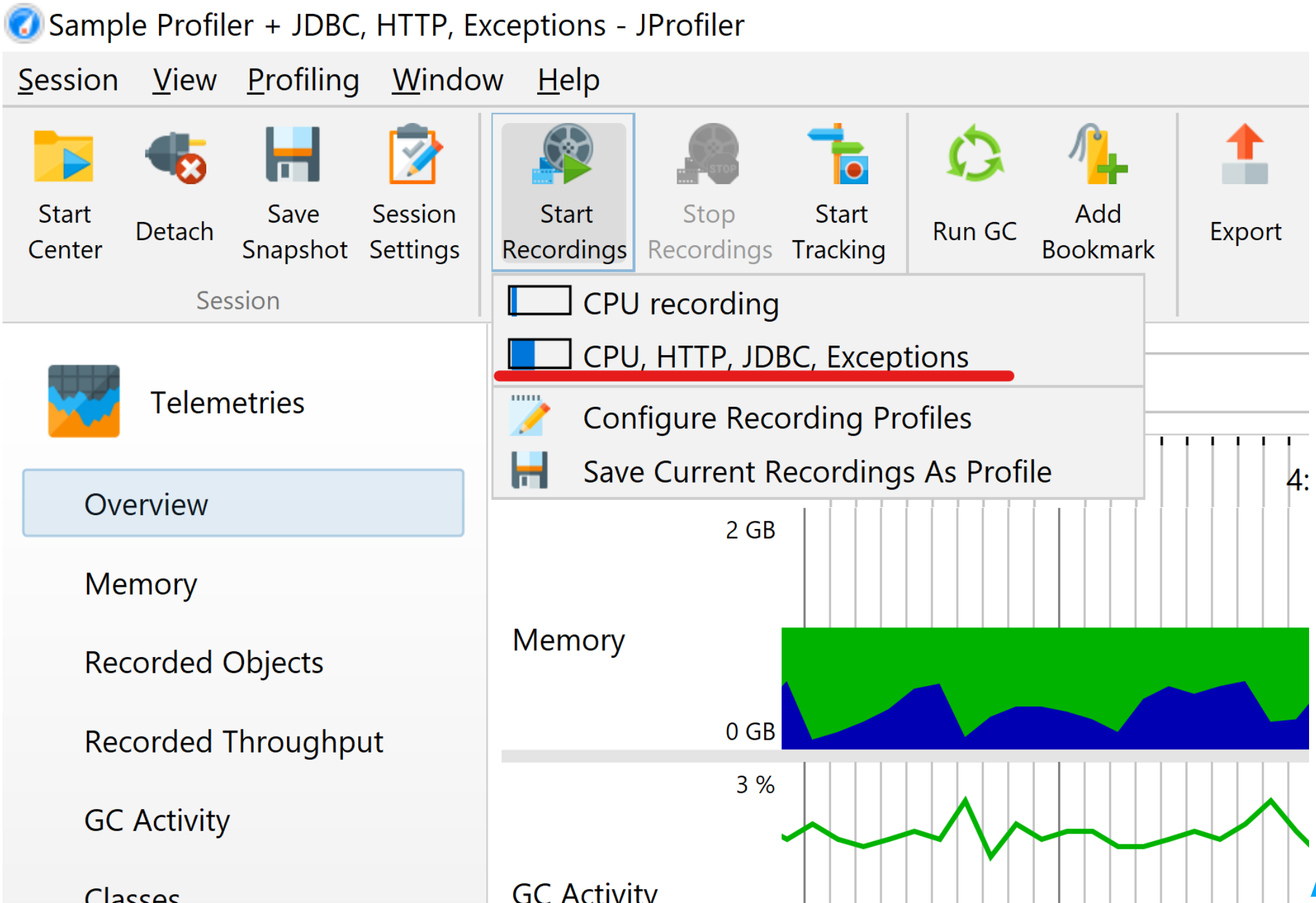

Session Settings
Application Settings
Call Tree Recording
Call Tree Filters
Trigger Settings
Database Settings
Probe Settings
Built-In Probes
Script Probes
Custom Probes
Advanced Settings
Built-in probes for JEE and JSE:
HTTP Server [record events]
HTTP Client [record events, annotate into call tree view]
Web Services [record events, annotate into call tree view]
JNDI [record events, annotate into call tree view]
JMS [record events, annotate into call tree view]
RMI [record events, annotate into call tree view]
Class Loaders
Exceptions [record events]
Sockets [record events, annotate into call tree view]
Files [record events, annotate into call tree view]
Processes [record events, annotate into call tree view]
General Settings
Copy Settings From
OK
Cancel

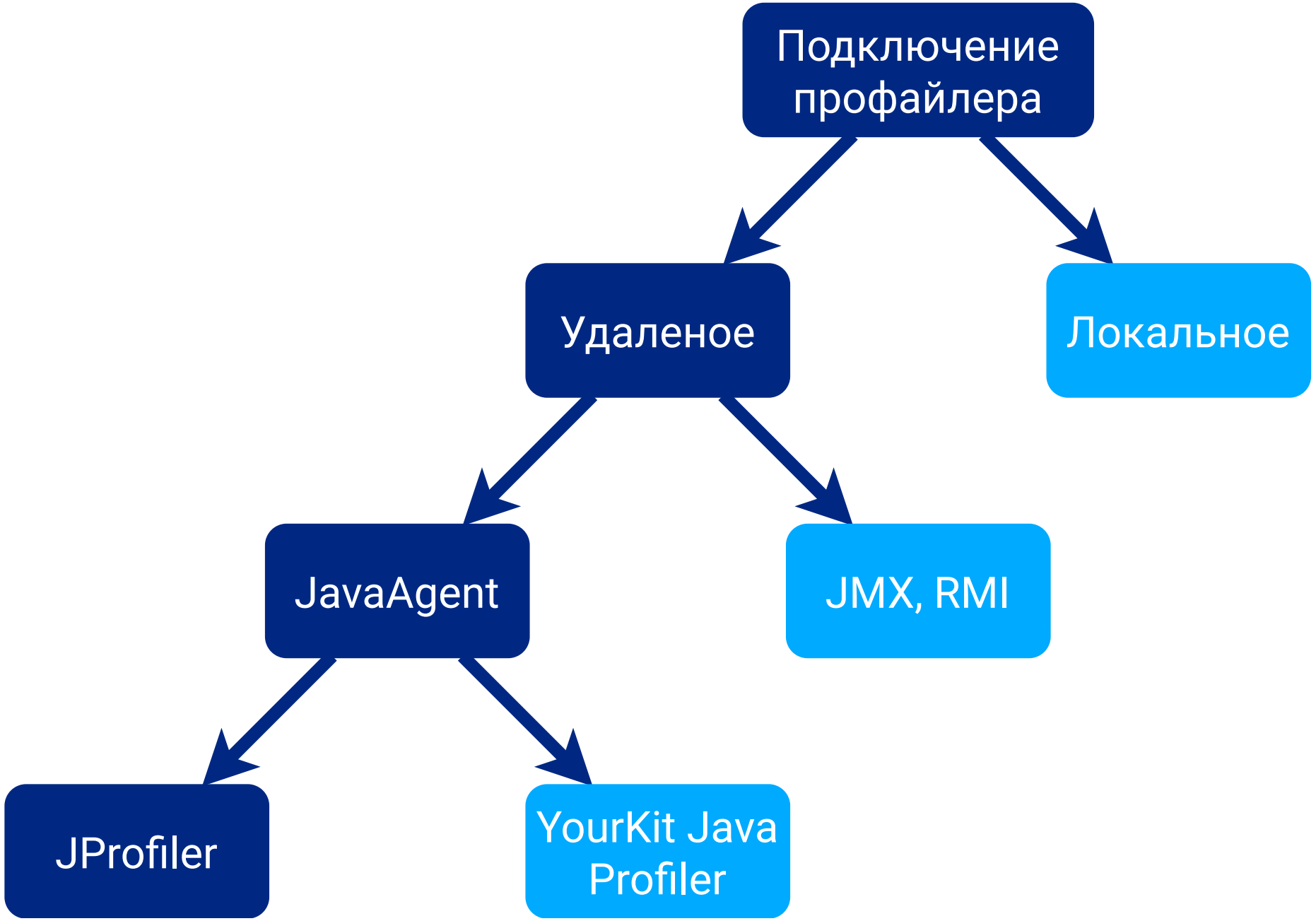

Session Startup
Settings
Call tree recording:
Full sampling
Edit
For all features including invocation counts, switch to instrumentation.
Call tree filters:
1 filter rule for method call recording
Edit
How should method calls be recorded?
There are important trade-offs to be considered. Check out the in-depth explanation in the documentation. You can change this setting at any time later on.
Instrumentation
All features
Invocation counts
Ideal for I/O bound code
Careful with CPU bound code
Adjust call tree filters
Sampling (Recommended)
Low overhead
Ideal for finding CPU hot spots
Better accuracy for CPU times
Not all features
Performance
Overhead:
The overhead is composed of the selected profiling settings and the selected recording profile.


Sample Profiler + JDBC, HTTP, Exceptions - JProfiler
Session
View
Profiling
Window
Help
Start Center
Detach
Save Snapshot
Session Settings
Session
Start Recordings
Stop Recordings
Start Tracking
Run GC
Add Bookmark
Export
View Settings
CPU recording
Configure Recording Profiles
Save Current Recordings As Profile
Telemetries
Overview
Memory
Recorded Objects
Recorded Throughput
GC Activity
Classes
0:10
0:20
2 GB
Memory
0 GB
3 %
GC Activity


Configure Recording Profiles
Configured recording profiles:
CPU recording
JProfiler
Please enter a name for the recording profile:
CPU, HTTP, JDBC, Exceptions
OK
Cancel

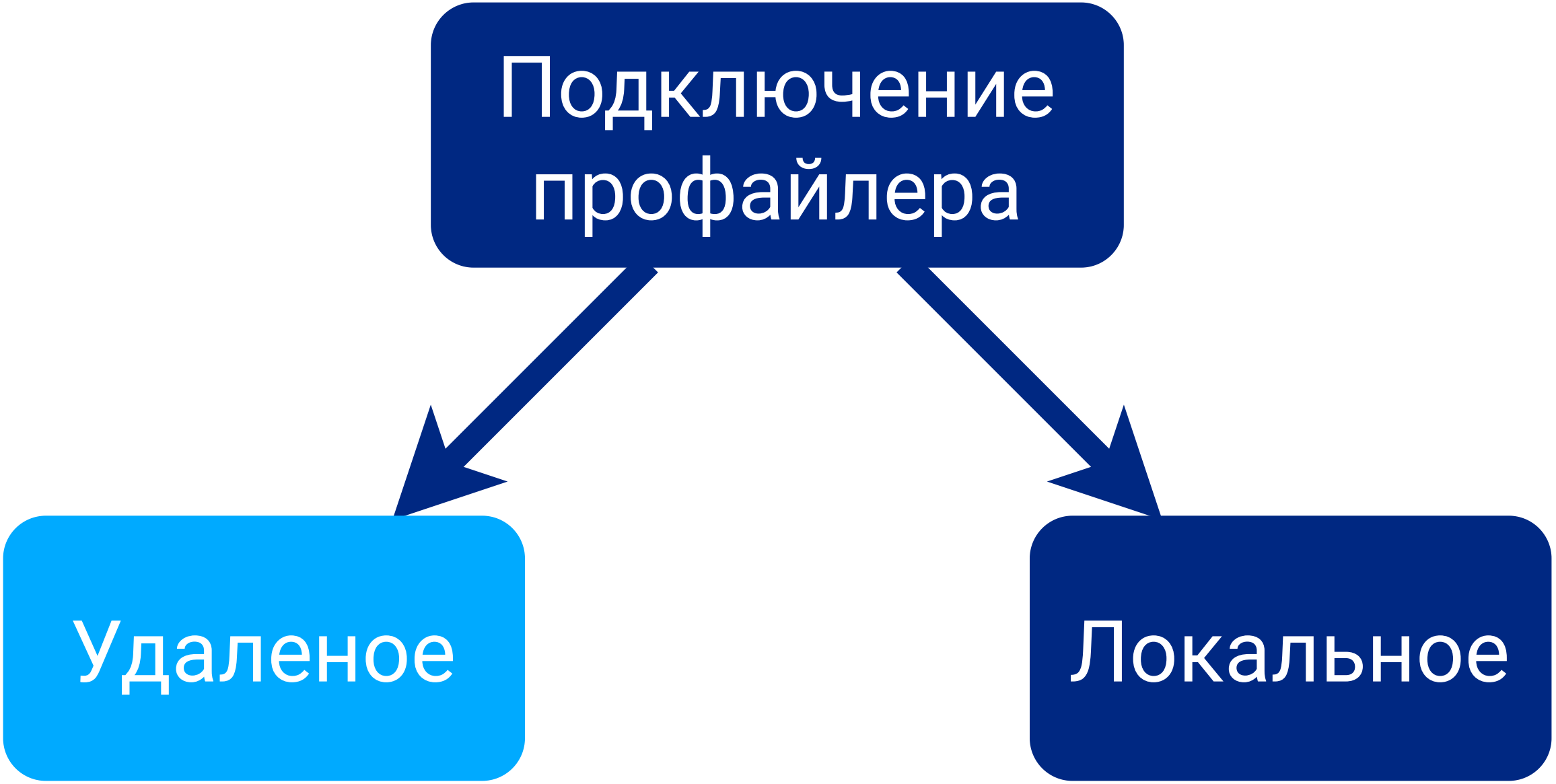

Configure Recording Profiles
Configured recording profiles:
CPU recording
CPU, HTTP, JDBC, Exceptions
CPU data
Call tracer
Complexity data
Allocation call stacks
Monitor recording
Custom probes
Record database probes: JDBC, JPA/Hibernate
Record built-in probes: HTTP Server, HTTP Client, Web Services, Class Loaders, E...
Recording overhead:
Help
OK
Cancel

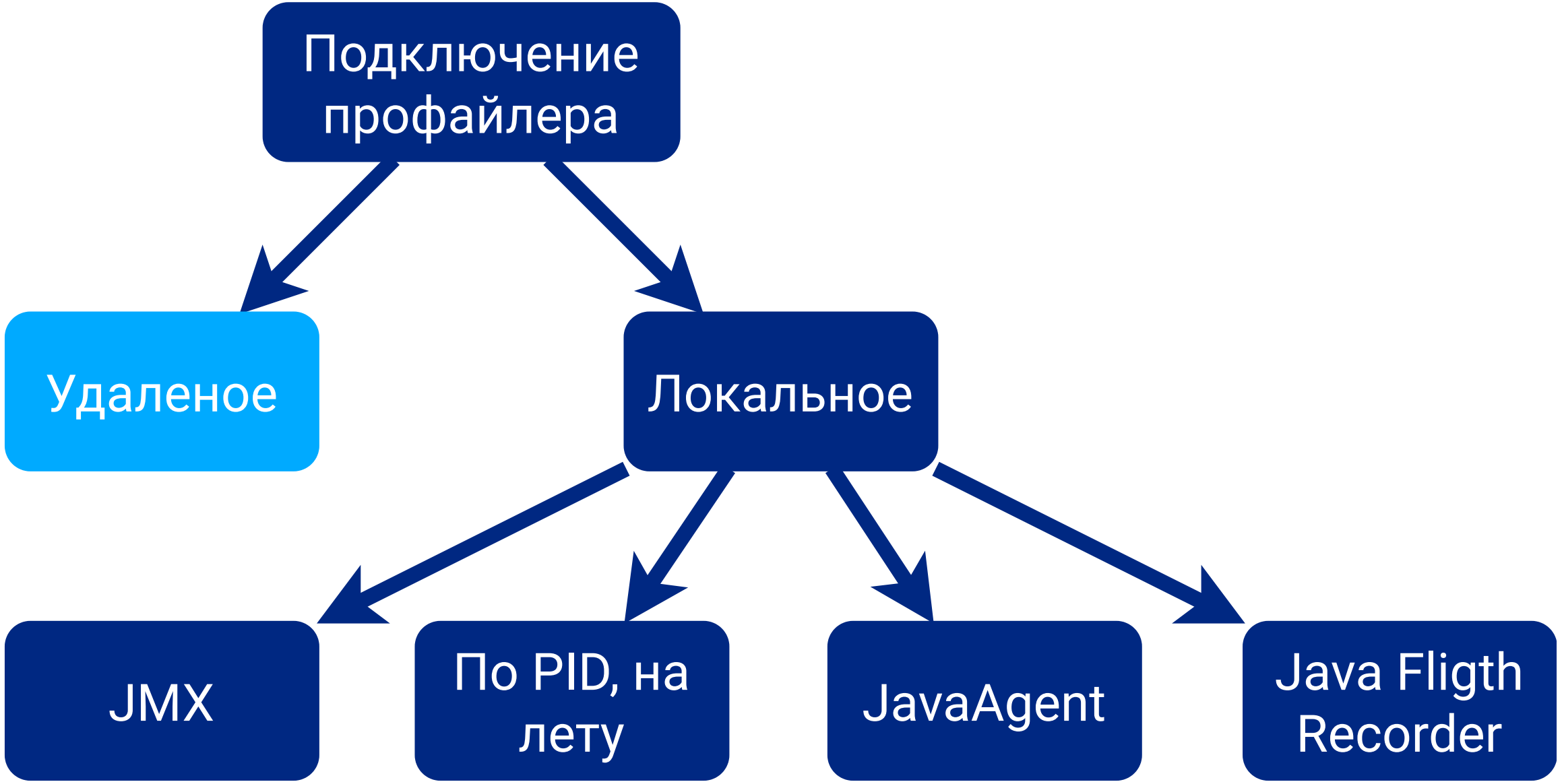

Sample Profiler + JDBC, HTTP, Exceptions - JProfiler
Session
View
Profiling
Window
Help
Start Center
Detach
Save Snapshot
Session Settings
Session
Start Recordings
Stop Recordings
Start Tracking
Run GC
Add Bookmark
Export
CPU recording
CPU, HTTP, JDBC, Exceptions
Configure Recording Profiles
Save Current Recordings As Profile
Telemetries
Overview
Memory
Recorded Objects
Recorded Throughput
GC Activity
Classes
4:
2 GB
Memory
0 GB
3 %
GC Activity

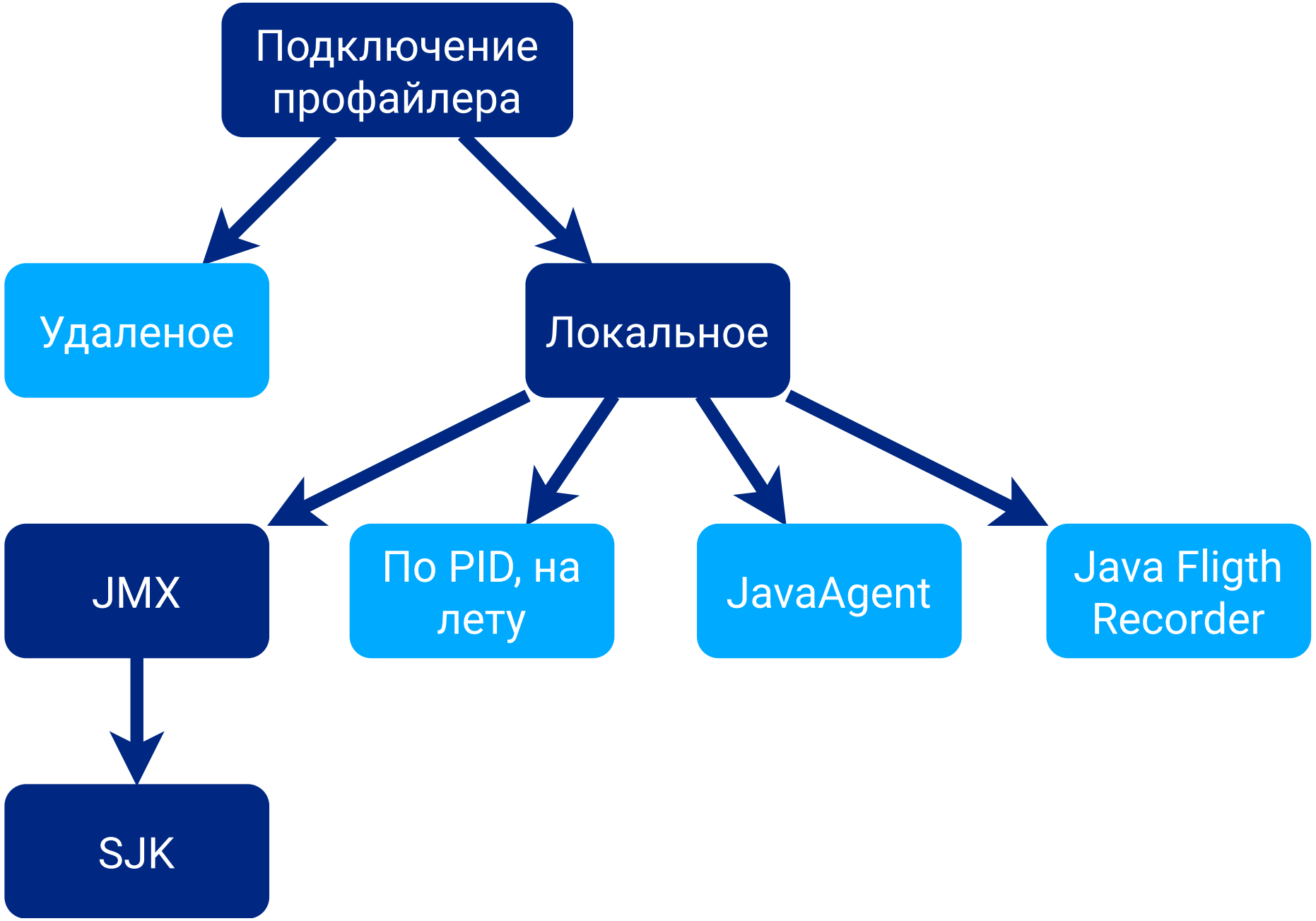

Подключение профайлера
Удаленое
Локальное
JavaAgent
JMX, RMI
JProfiler
YourKit Java Profiler

Изображение Free-Photos с сайта Pixabay

Изображение LEEROY Agency с сайта Pixabay


Подключение профайлера
Удаленое
Локальное


Подключение профайлера
Удаленое
Локальное
JMX
По PID, на лету
JavaAgent
Java Fligth Recorder


Подключение профайлера
Удаленое
Локальное
JMX
По PID, на лету
JavaAgent
Java Fligth Recorder
SJK

Swiss Java Knife (SJK) (репозиторий проекта), простой

Алексей Рагозин

Упрощенная настройка JMX для SJK

Опции JVM

```
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=9010
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
#-Dcom.sun.management.jmxremote.rmi.port=9010
#-Dcom.sun.management.jmxremote.local.only=true
#-Djava.rmi.server.hostname=127.0.0.1
```

Упрощенная настройка JMX для SJK

```
kind: Deployment
spec:
  template:
    spec:
      containers:
        - name: test-webserver
          env:
            - name: JAVA_OPTIONS
              value: >-
                -Dcom.sun.management.jmxremote
                -Dcom.sun.management.jmxremote.port=9010
                -Dcom.sun.management.jmxremote.authenticate=false
                -Dcom.sun.management.jmxremote.ssl=false
```

Запуск SJK осуществляется из POD-ы

Найти имя поды <podname>

```
kubectl get pods | grep test-webserver
pod="<podname>"
```

Проброс портов не нужен, нужно копирование файла

```
kubectl cp "sjk-0.17.jar" $pod:/tmp/sjk.jar
```

Запуск профайлера

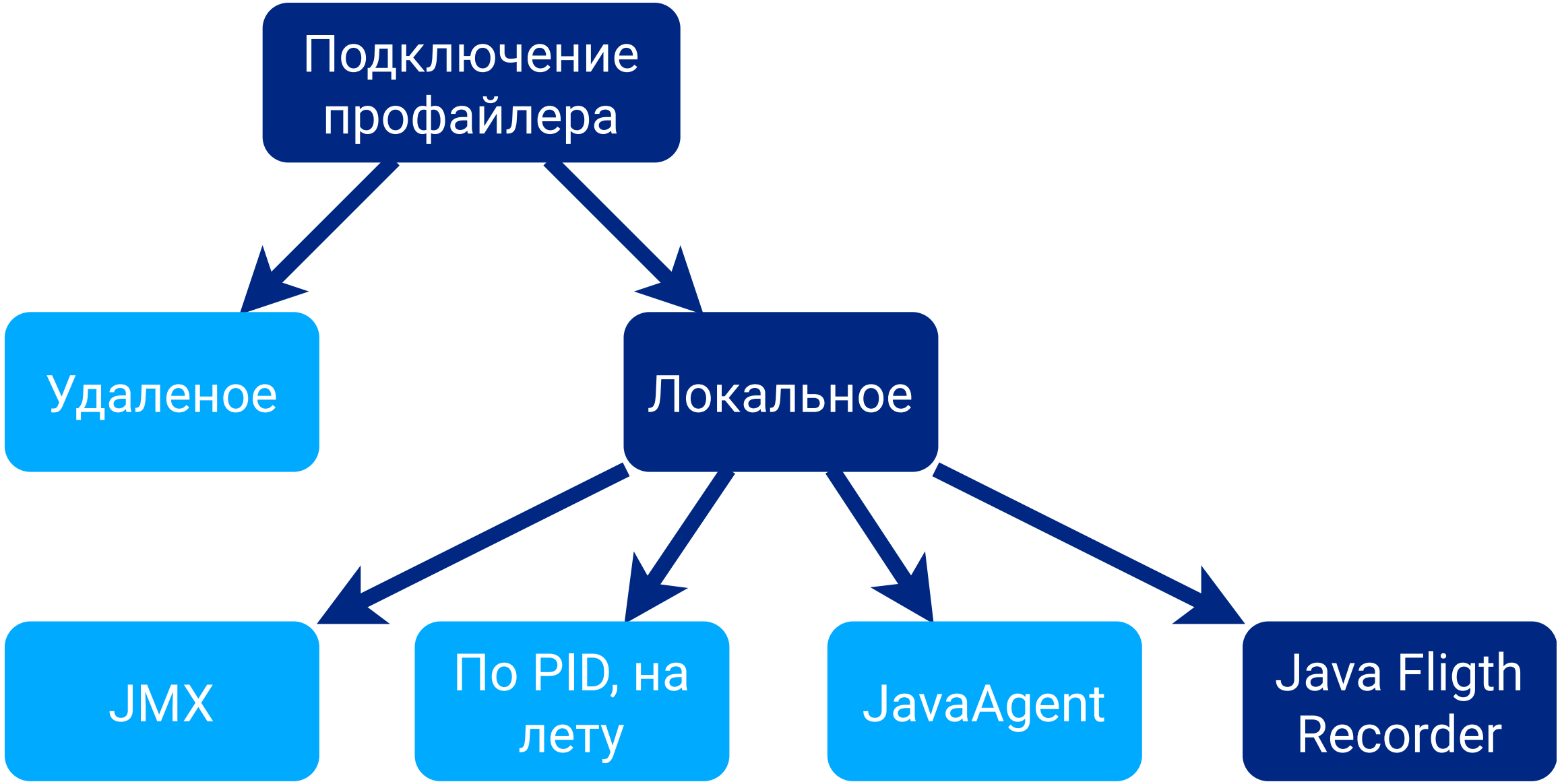
```
kubectl exec $pod -- java -jar /tmp/sjk.jar stcap
--socket localhost:9010 --timeout 5m --sampler-interval 100ms
--output /tmp/result.sdt &
```

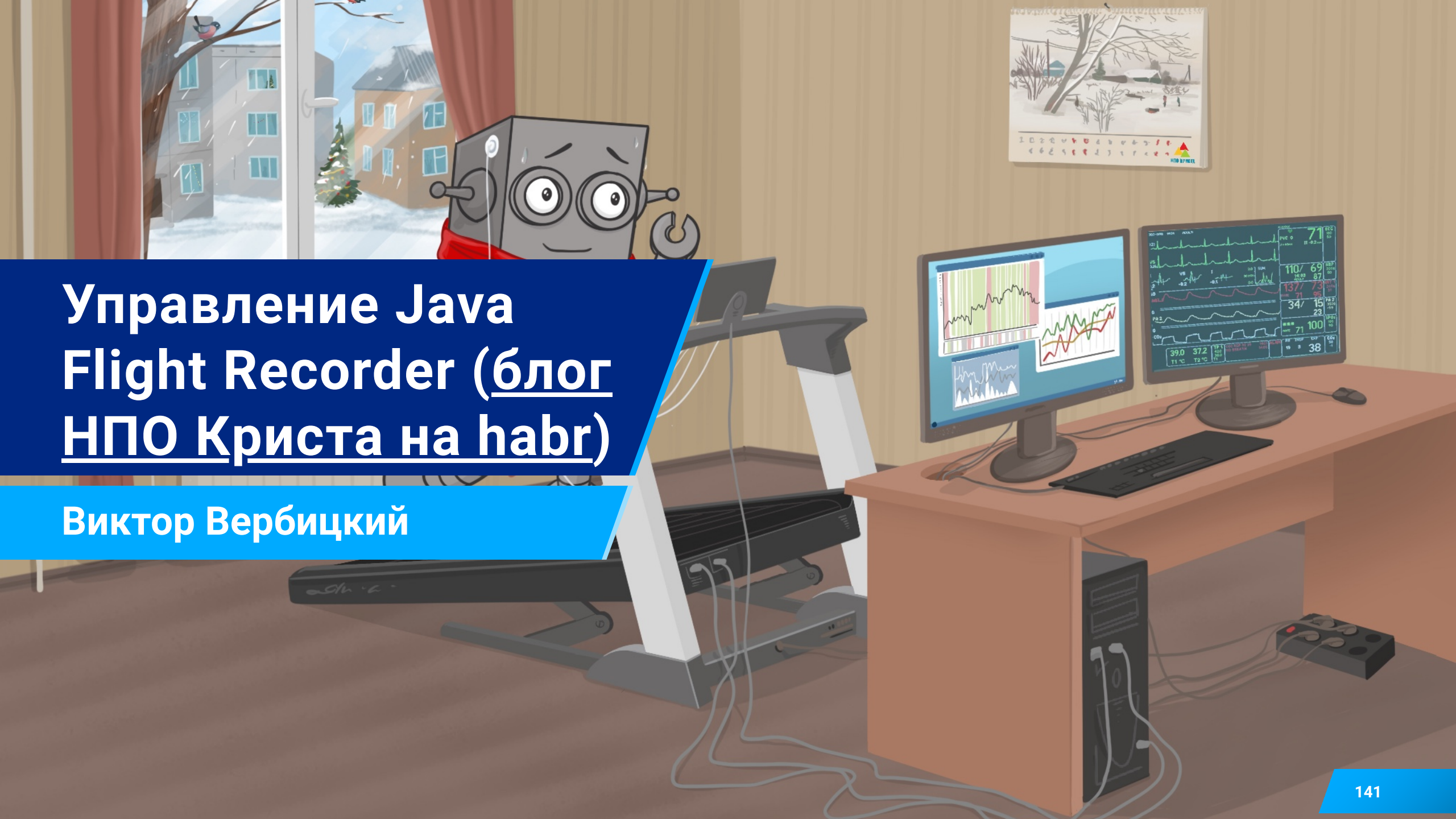
Запуск SJK из git bash для Windows

```
#!/bin/sh
duration=5m
pod="<podname>"
kubectl exec $pod -- sh -x -c \
"java -jar   /tmp/sjk.jar
    -X stcap
    --socket localhost:9010
    --timeout $duration
    --sampler-interval 100ms
    --output /tmp/result.sdt
             >/tmp/result.out.txt
            2>/tmp/result.error.txt ;
echo Complete" &
```

Скачивание результатов через kubectl exec

```
#!/bin/sh
pod="<podname>"
currdir=$(pwd)
currdate=$(date '+%Y-%m-%d_%H-%M-%S')
# Каталог с результатами
result="$currdir/$currdate/$pod/"
mkdir -p "$result"
cd "$result"
# Упаковать файлы /tmp/result.* POD-ы и распаковать локально
kubectl exec $pod -- sh -c 'cd /tmp ; tar cf - result.*' \
    | tar xf - -C "$result"
explorer .
cd "$currdir"
```


Подключение профайлера
Удаленое
Локальное
JMX
По PID, на лету
JavaAgent
Java Fligth Recorder



Управление Java Flight Recorder (блог НПО Криста на habr)

Виктор Вербицкий

Основные опции

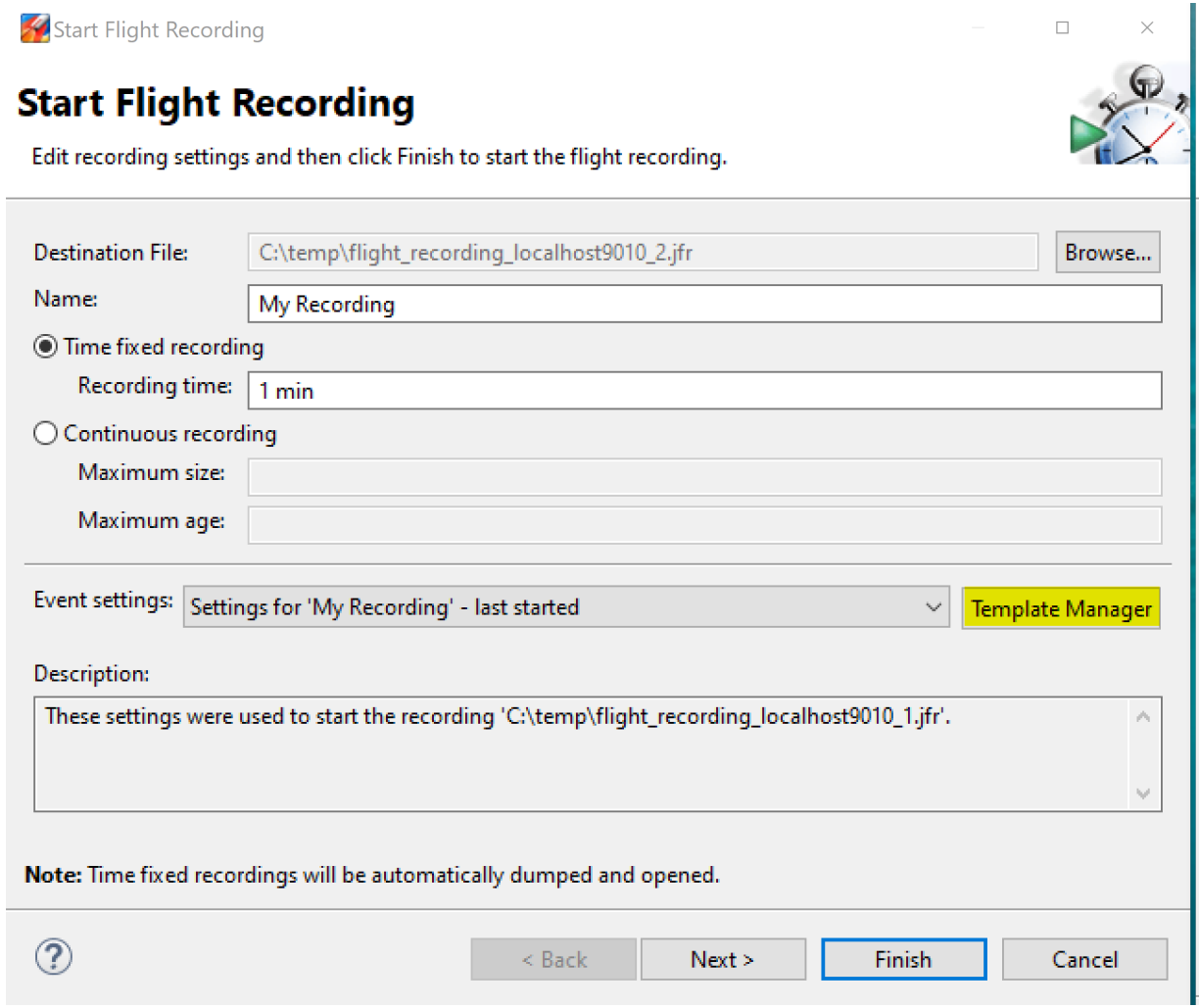
```
-
XX:StartFlightRecording=disk=true,maxsize=1g,maxage=24h,fi
lename=/tmp/recording.jfr \ -
XX:FlightRecorderOptions=repository=/tmp/
diagnostics/,maxchunksize=1m,stackdepth=1024
```

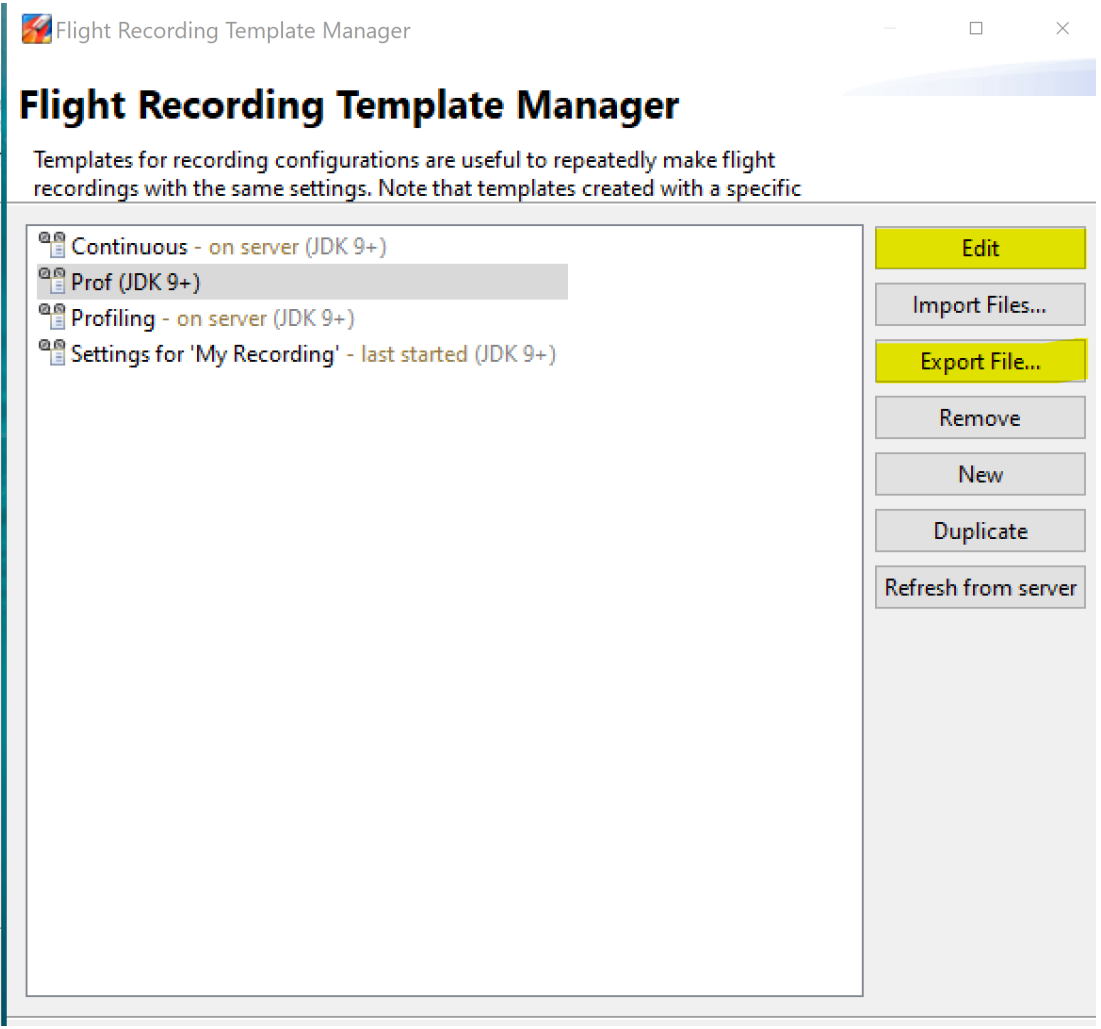
- `disk=true` запись на диск, а не в память
- `maxsize=1g,maxage=24h` чтобы диск не переполнился
- `filename=/tmp/recording.jfr` параметры JFR
- `repository=/tmp/diagnostics/` каталог результатов
- `maxchunksize=1m` размер одного файла будет 2-3 МБайт
- `stackdepth=1024` глубина стека увеличена

Параметры JFR: filename=Путь-к-файлу

- Параметры по умолчанию в файле jre\lib\jfr\default.jfc
- Файл в формате XML
- Можно редактировать в JMC

Параметры JFR можно редактировать в JMC





Упрощенная настройка Java Flight Recorder

```
kind: Deployment
spec:
  template:
    spec:
      containers:
        - name: test-webserver
          env:
            - name: JAVA_OPTIONS
              value: >-
                -XX:StartFlightRecording=disk=true,maxsize=1g
                -XX:FlightRecorderOptions=repository=/tmp/results,maxchunksize=1m,stackdepth=1024
```

Тонкая настройка Java Flight Recorder

С файлом настроек /tmp/jfr/prof.jfc

```
kind: Deployment
spec:
  template:
    spec:
      volumes:
        - name: jfr
          hostPath:
            path: /opt/data/jfr
      containers:
        - name: test-webserver
          env:
            - name: JAVA_OPTIONS
              value: >-
                -XX:StartFlightRecording=disk=true,maxsize=1g,filename=/tmp/jfr/prof.jfc
                -XX:FlightRecorderOptions=repository=/tmp/results,maxchunksize=1m,stackdepth=1024
          volumeMounts:
            - name: jfr
              mountPath: /tmp/jfr
```

Тонкая настройка Java Flight Recorder

Файл настроек prof.jfc сохраняется во внешний каталог

```
kind: Deployment
spec:
  template:
    spec:
      volumes:
        - name: jfr
          hostPath:
            path: /opt/data/jfr      # Каталог /opt/data/jfr с файлом prof.jfc
      containers:
        - name: test-webserver
          env:
            - name: JAVA_OPTIONS
              value: >-
                -XX:StartFlightRecording=disk=true,maxsize=1g,filename=/tmp/jfr/prof.jfc
                -XX:FlightRecorderOptions=repository=/tmp/results,maxchunksize=1m,stackdepth=1024
          volumeMounts:
            - name: jfr
              mountPath: /tmp/jfr
```

Тонкая настройка Java Flight Recorder

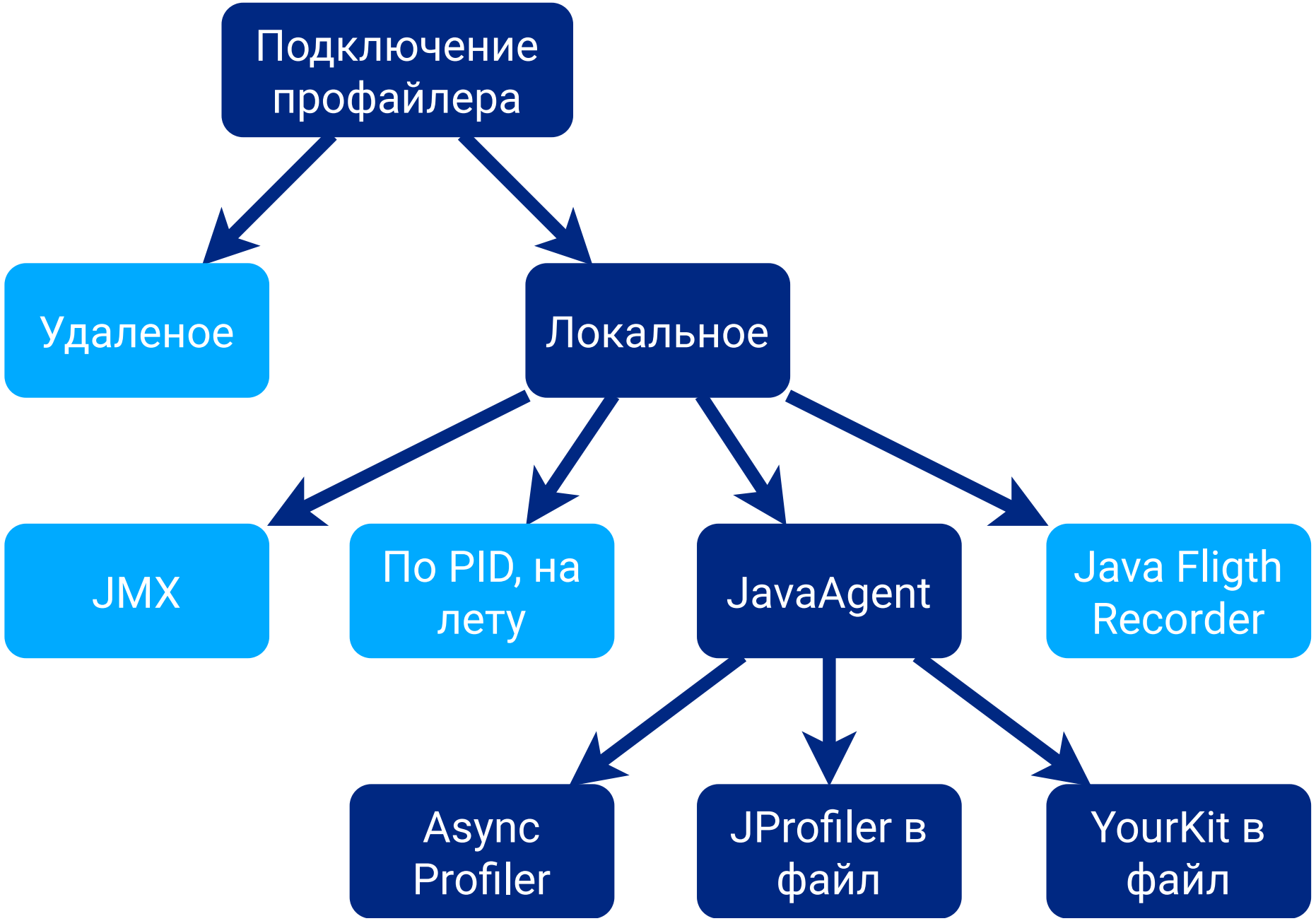
Внешний каталог с файлом монтируется в Pod

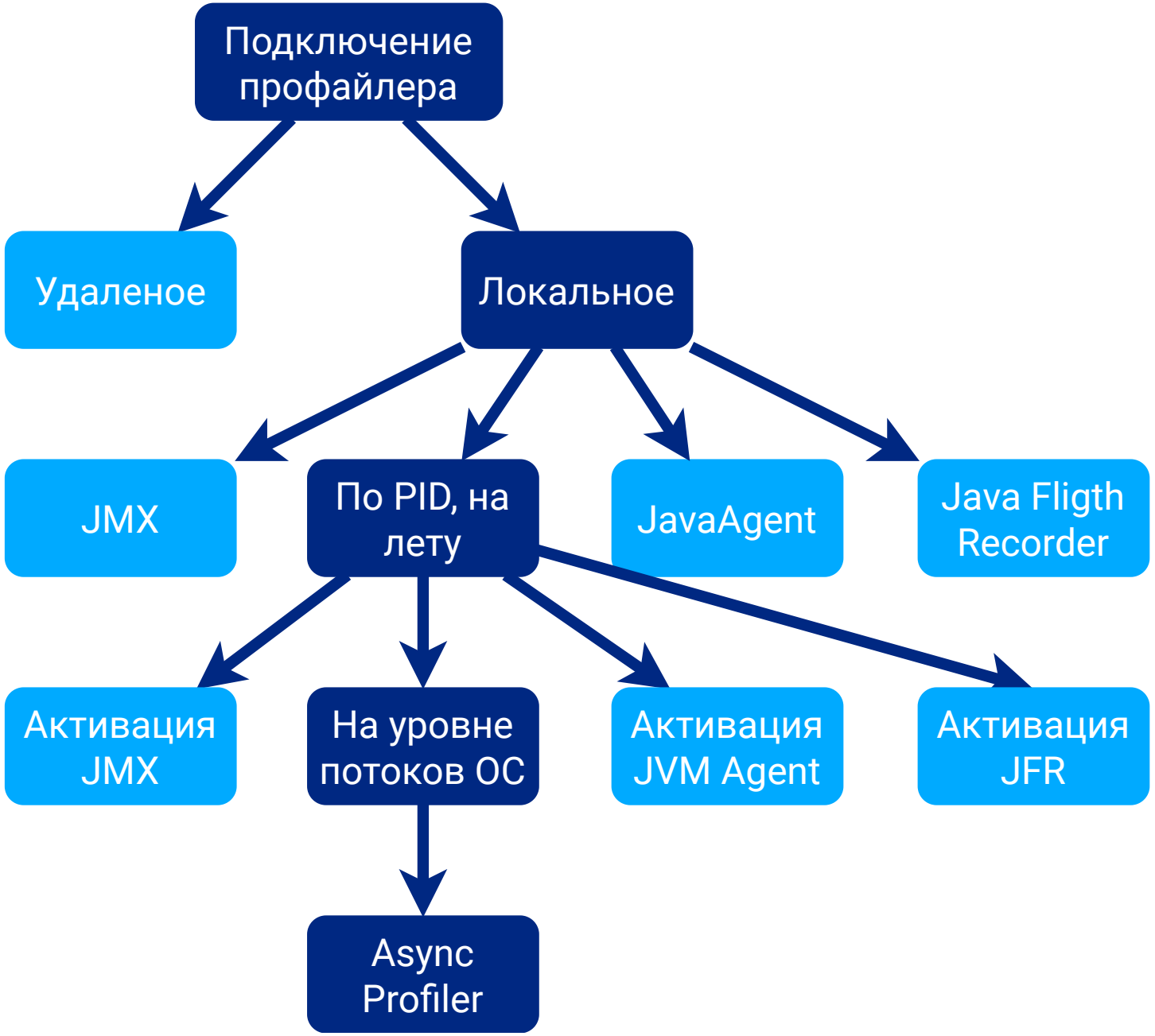
```
kind: Deployment
spec:
  template:
    spec:
      volumes:
        - name: jfr
          hostPath:
            path: /opt/data/jfr
      containers:
        - name: test-webserver
          env:
            - name: JAVA_OPTIONS
              value: >-
                -XX:StartFlightRecording=disk=true,maxsize=1g,filename=/tmp/jfr/prof.jfc
                -XX:FlightRecorderOptions=repository=/tmp/results,maxchunksize=1m,stackdepth=1024
          volumeMounts:
            - name: jfr
              mountPath: /tmp/jfr    # Монтируем /opt/data/jfr в /tmp/jfr
```

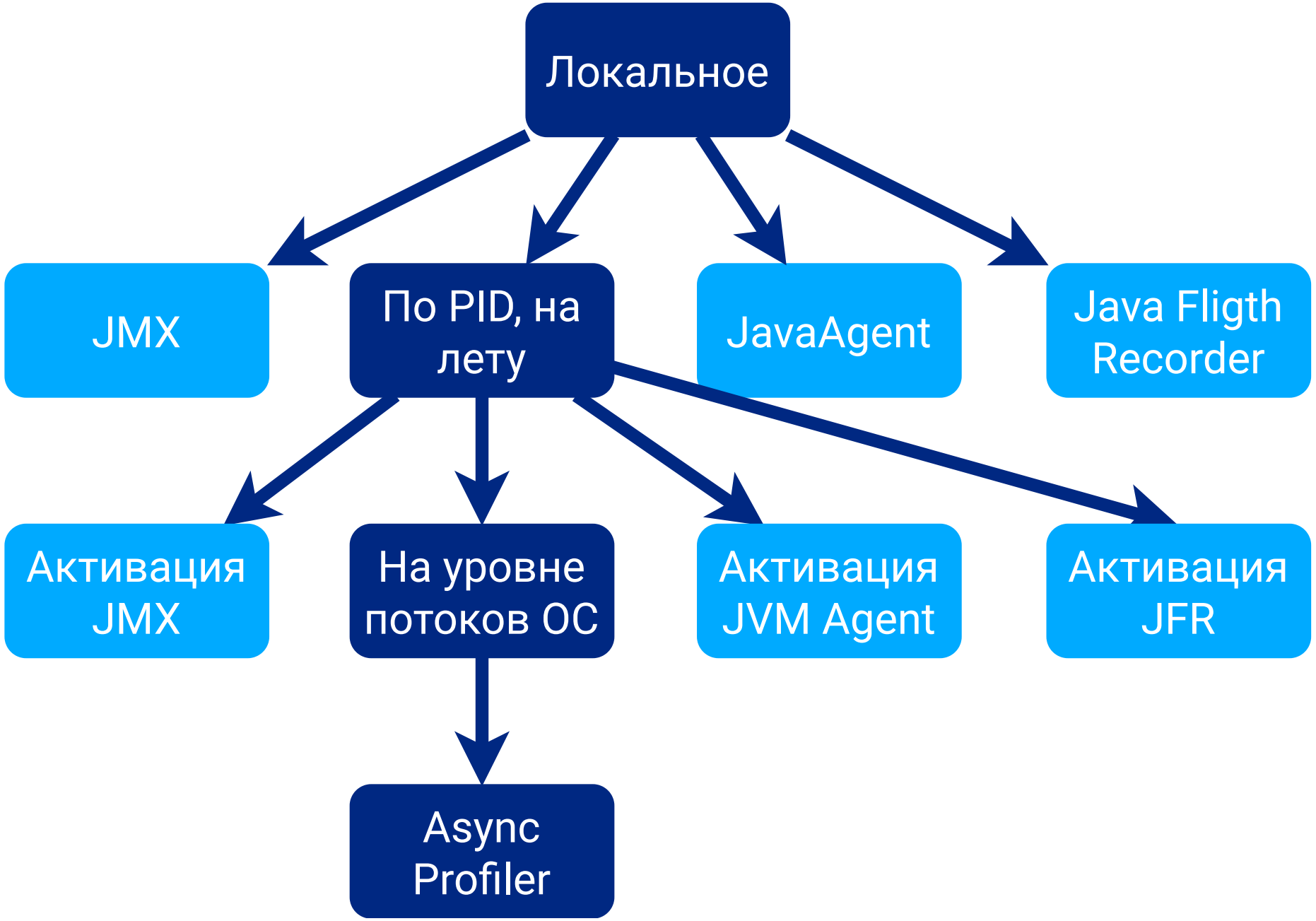
Тонкая настройка Java Flight Recorder

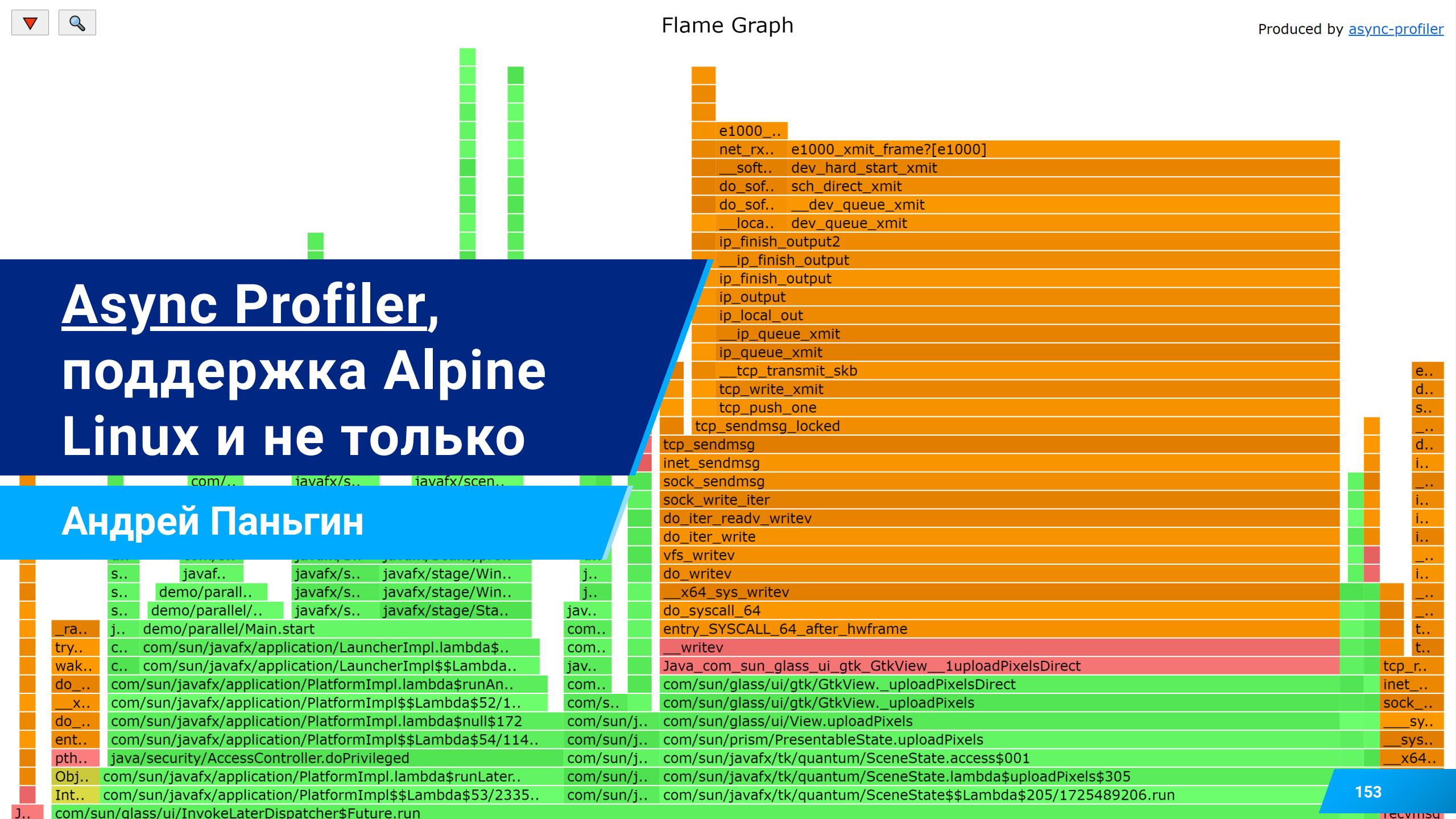
Файл настроек prof.jfc передается в JAVA_OPTIONS

```
kind: Deployment
spec:
  template:
    spec:
      volumes:
        - name: jfr
          hostPath:
            path: /opt/data/jfr
      containers:
        - name: test-webserver
          env:
            - name: JAVA_OPTIONS
              value: >-                # Передаем путь /tmp/jfr/prof.jfc в параметр filename
                -XX:StartFlightRecording=disk=true,maxsize=1g,filename=/tmp/jfr/prof.jfc
                -XX:FlightRecorderOptions=repository=/tmp/results,maxchunksize=1m,stackdepth=1024
          volumeMounts:
            - name: jfr
              mountPath: /tmp/jfr
```


Подключение профайлера
Удаленое
Локальное
JMX
По PID, на лету
JavaAgent
Java Fligth Recorder
Async Profiler
JProfiler в файл
YourKit в файл

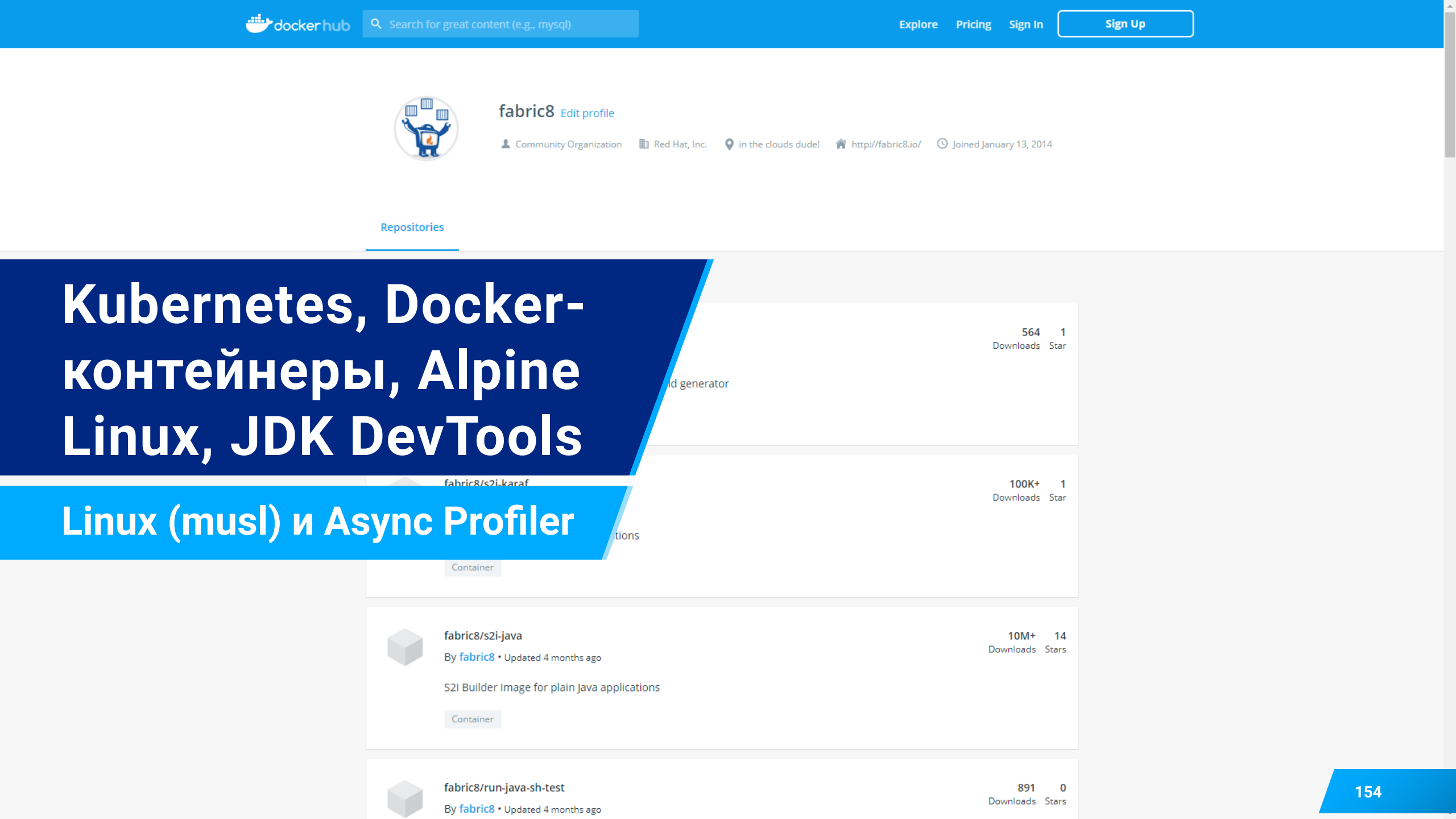

Подключение профайлера
Удаленое
Локальное
JMX
По PID, на лету
JavaAgent
Java Fligth Recorder
Активация JMX
На уровне потоков ОС
Активация JVM Agent
Активация JFR
Async Profiler


Локальное
JMX
По PID, на лету
JavaAgent
Java Fligth Recorder
Активация JMX
На уровне потоков ОС
Активация JVM Agent
Активация JFR
Async Profiler



Async Profiler, поддержка Alpine Linux и не только

Андрей Паньгин



Kubernetes, Docker-контейнеры, Alpine Linux, JDK DevTools

Linux (musl) и Async Profiler

Факторы выбора контейнеров

fabric8/java-centos-openjdk8-jre

By fabric8 • Updated 5 months ago

Fabric8 Java Base Image (CentOS, OpenJDK 8, JRE)

Container

100K+ Downloads 4 Stars

fabric8/java-jboss-openjdk7-jdk

By fabric8 • Updated 5 months ago

Fabric8 Java Base Image (JBoss, OpenJDK 7)

Container

2.6K Downloads 1 Star

fabric8/java-alpine-openjdk11-jre

By fabric8 • Updated 5 months ago

Fabric8 Java Base Image (Alpine, OpenJDK 11, JRE)

Container

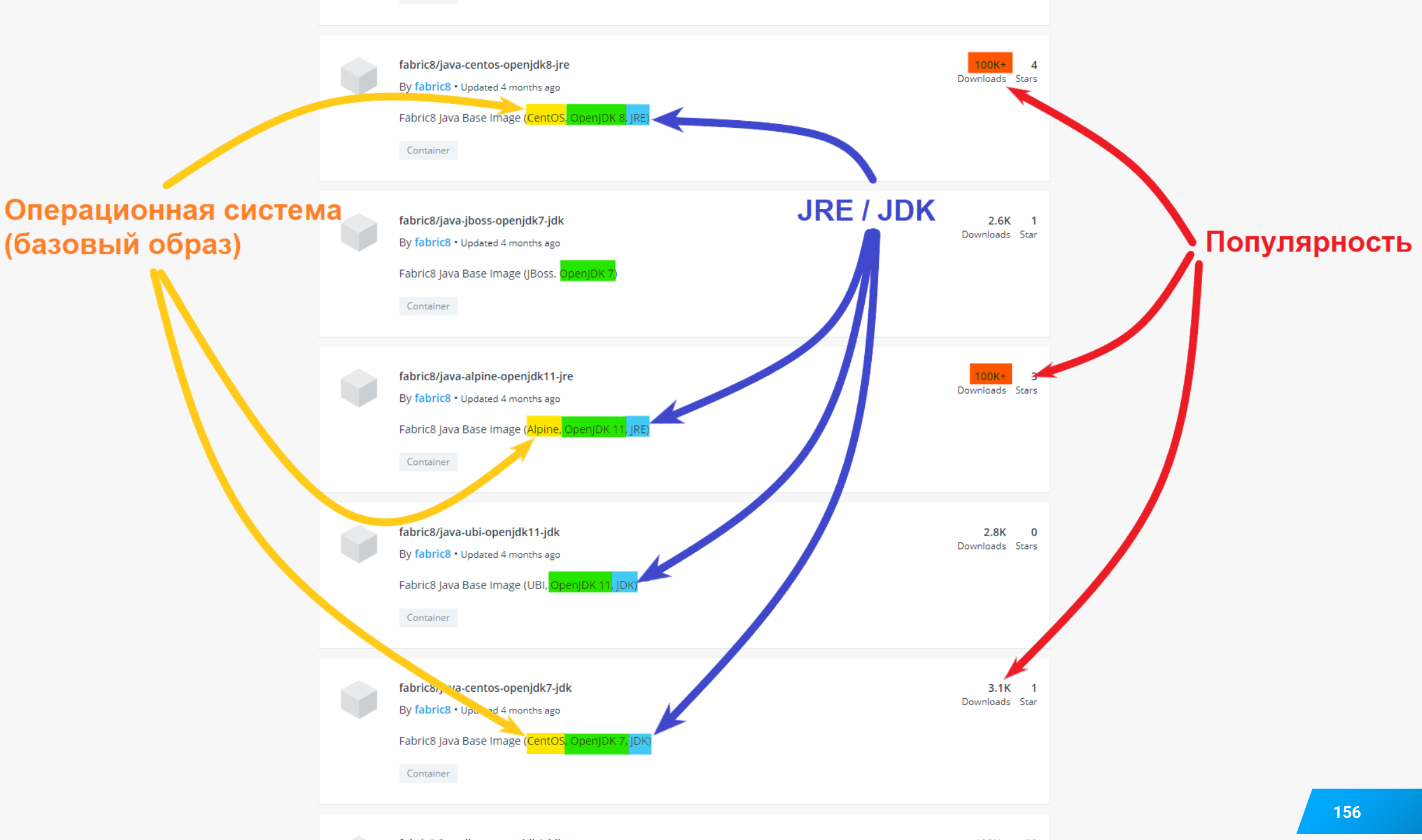
100K+ Downloads 3 Stars

fabric8/java-ubi-openjdk11-jdk

By fabric8 • Updated 5 months ago

Fabric8 Java Base Image (UBI, OpenJDK 11, JDK)

3.2K Downloads 1 Star


Операционная система
(базовый образ)
JRE / JDK
Популярность
fabric8/java-centos-openjdk8-jre
By fabric8 • Updated 4 months ago
Fabric8 Java Base Image (CentOS, OpenJDK 8, JRE)
Container
100K+ Downloads
4 Stars
fabric8/java-jboss-openjdk7-jdk
By fabric8 • Updated 4 months ago
Fabric8 Java Base Image (JBoss, OpenJDK 7)
Container
2.6K Downloads
1 Star
fabric8/java-alpine-openjdk11-jre
By fabric8 • Updated 4 months ago
Fabric8 Java Base Image (Alpine, OpenJDK 11, JRE)
Container
100K+ Downloads
3 Stars
fabric8/java-ubi-openjdk11-jdk
By fabric8 • Updated 4 months ago
Fabric8 Java Base Image (UBI, OpenJDK 11, JDK)
Container
2.8K Downloads
0 Stars
fabric8/java-centos-openjdk7-jdk
By fabric8 • Updated 4 months ago
Fabric8 Java Base Image (CentOS, OpenJDK 7, JDK)
Container
3.1K Downloads
1 Star

Популярные образы с OpenJDK

Name	OS	Ver	Dev?	Hit
s2i-java	CentOS	8/11	JRE	10M
java-centos-openjdk8-jre	CentOS	8	JRE	100k
java-centos-openjdk8-jdk	CentOS	8	JDK	100k
java-alpine-openjdk8-jre	Alpine	8	JRE	100k
java-alpine-openjdk8-jdk	Alpine	8	JDK	100k
java-alpine-openjdk11-jre	Alpine	11	JRE	100k

Популярные образы с OpenJDK

	Первое место	Второе место
Операционная систеа	CentOS	Alpine
Версия Java в OpenJDK	8	11
Средства разработки	JRE (нет dev tools)	JDK (есть dev tools)
Маркировка для профайлеров	linux-x64	linux-**musl**-x64

README.md

Download

Current release (2.0):

- Linux x64 (glibc): async-profiler-2.0-linux-x64.tar.gz
- Linux x86 (glibc): async-profiler-2.0-linux-x86.tar.gz
- Linux x64 (musl): async-profiler-2.0-linux-musl-x64.tar.gz
- Linux ARM: async-profiler-2.0-linux-arm.tar.gz
- Linux AArch64: async-profiler-2.0-linux-aarch64.tar.gz
- macOS x64: async-profiler-2.0-macos-x64.tar.gz
- Converters between profile formats: converter.jar
  (JFR to Flame Graph, JFR to FlameScope, collapsed stacks to Flame Graph)

Previous releases

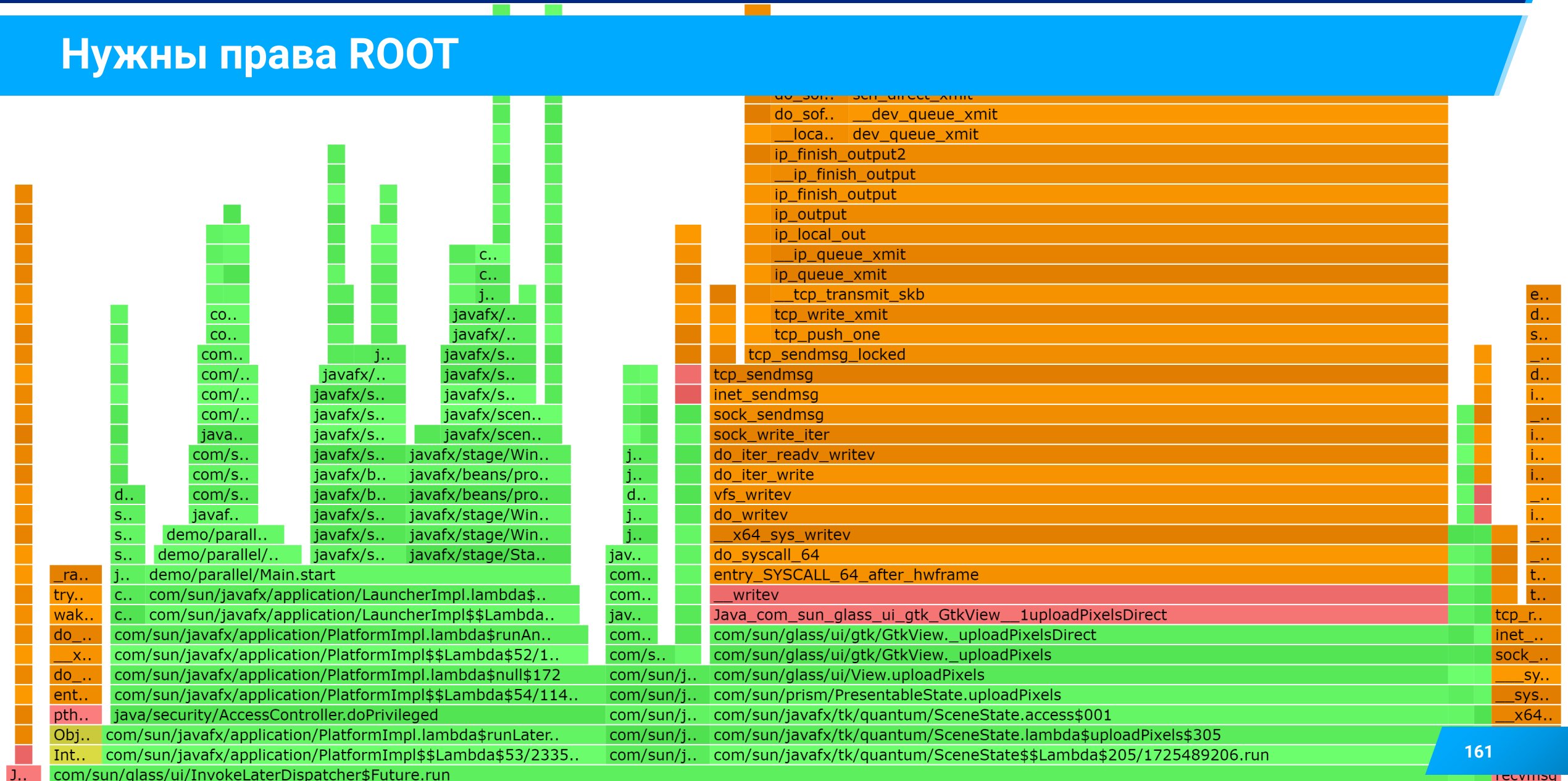
Популярные образы с OpenJDK

	Второе место
Операционная систеа	Alpine
Версия Java в OpenJDK	11
Средства разработки	**JDK (есть dev tools)**
Маркировка для профайлеров	linux-**musl**-x64

- DevTools есть, но их как бы нет, не работают утилиты **jcmd**, ...
- К счастью в **Async Profiler** есть утилита **jattach**
- **jattach** — аналог jcmd и она работает в Alpine внутри Docker

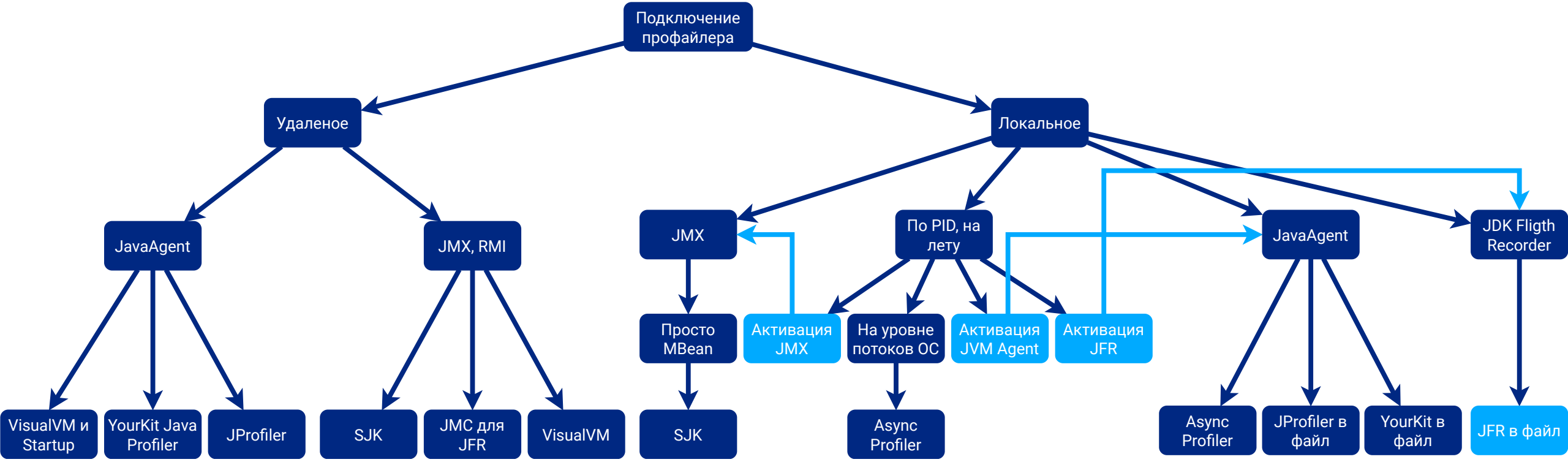
Async Profiler, поддержка Alpine Linux

Нужны права ROOT



Подключение профайлера к JVM в Kubernetes

Особенности профайлеров

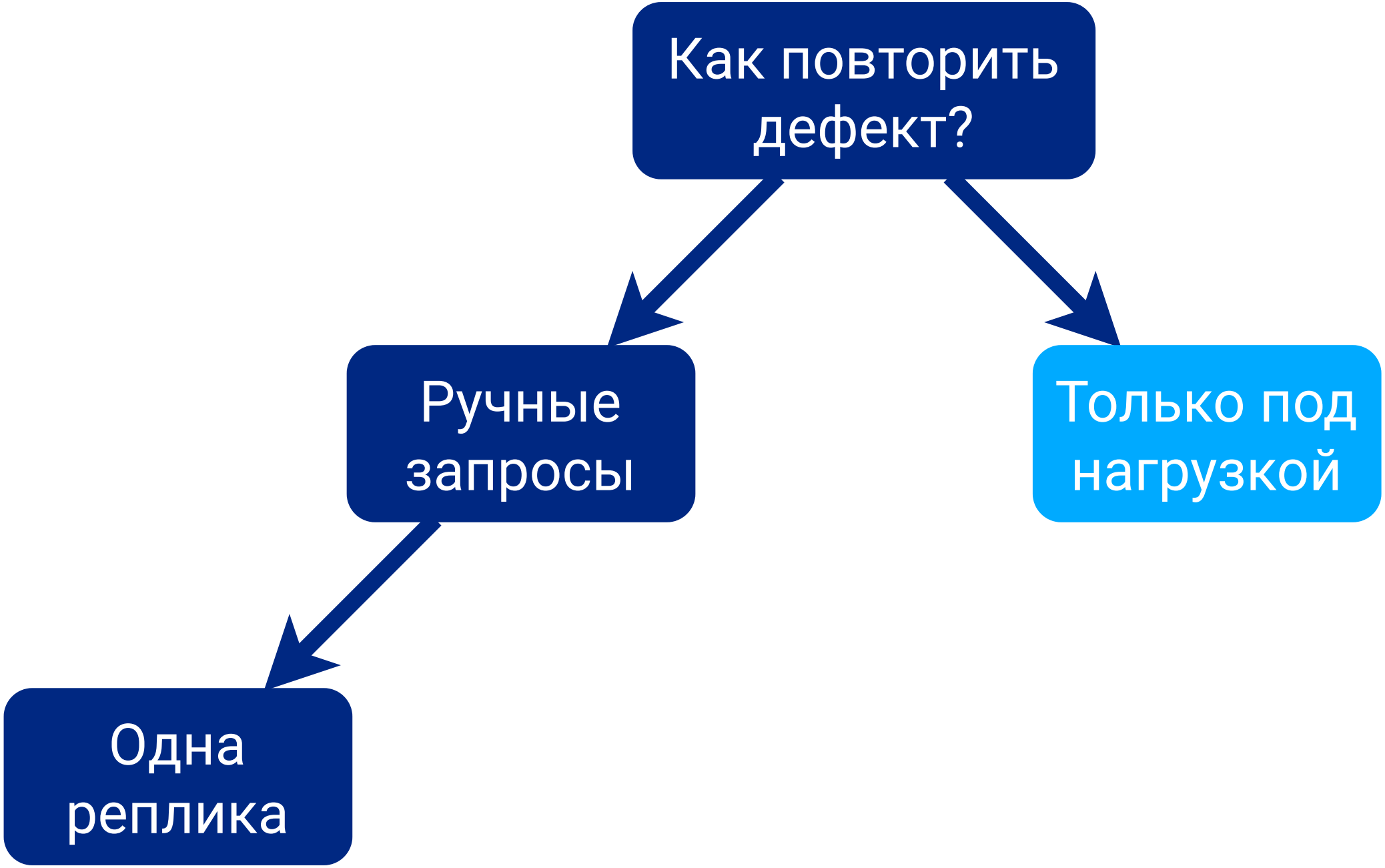


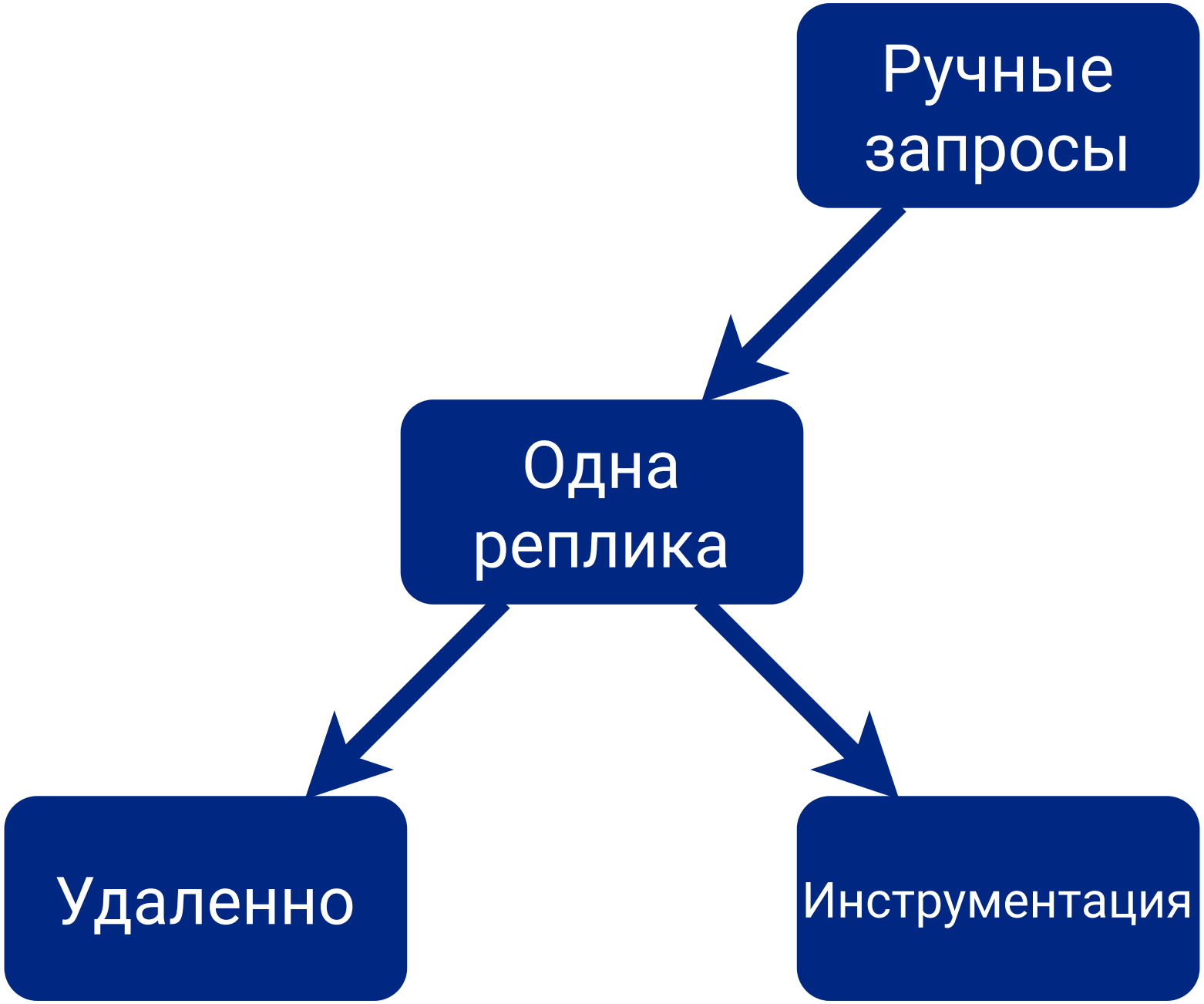
Выбор количества реплик сервиса и инструмента

Особенности Kubernetes

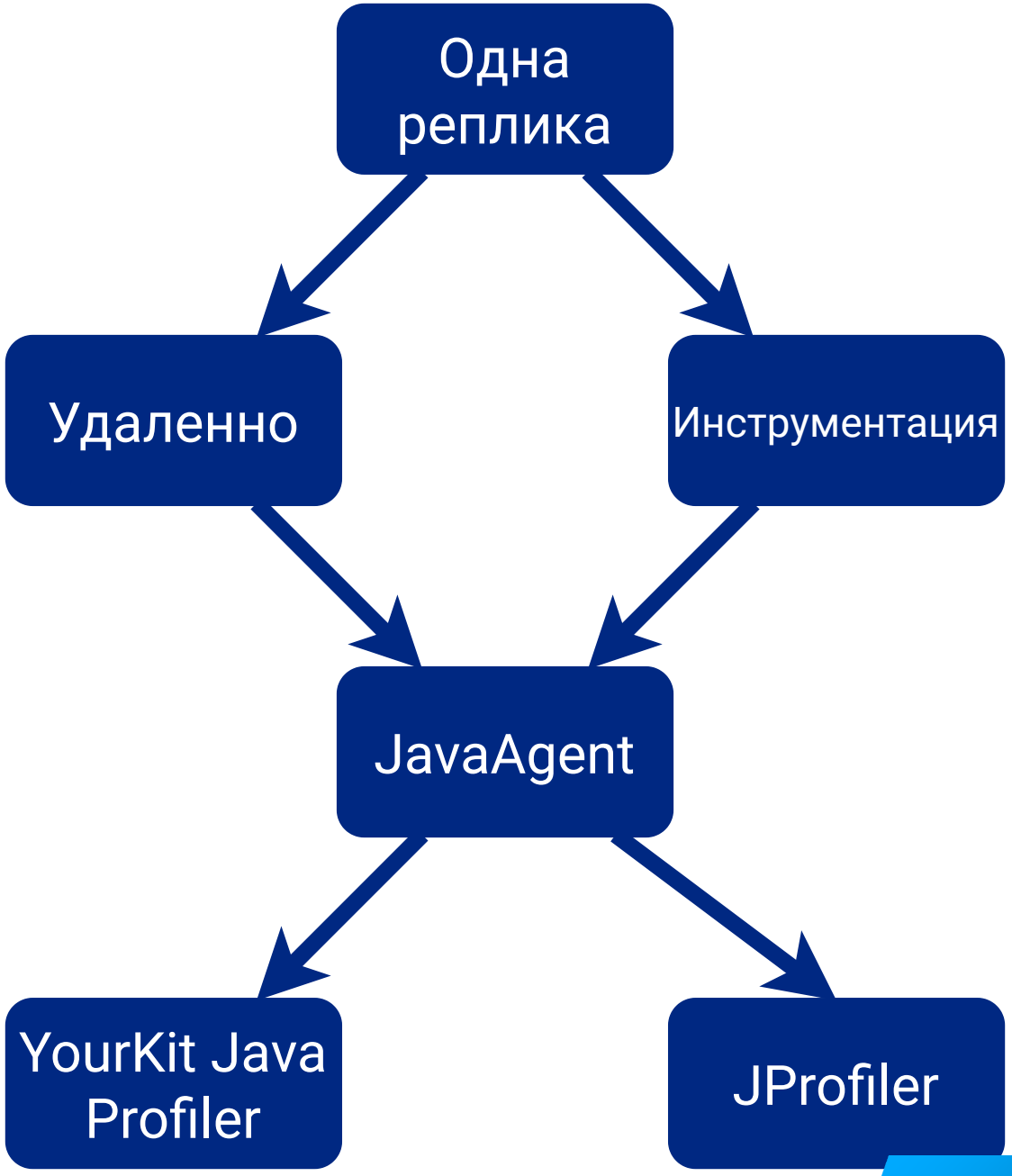
Профайлер

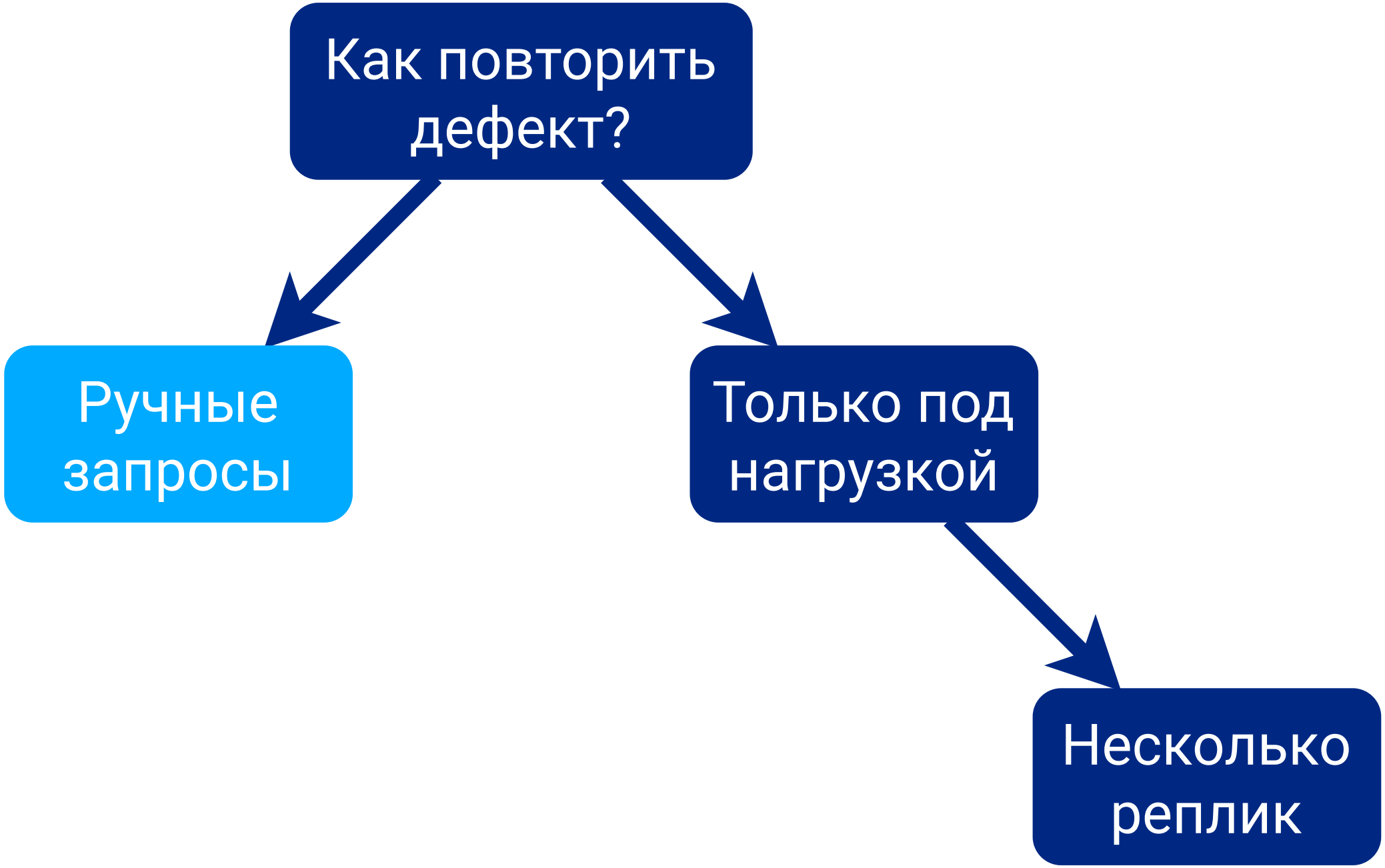

Как повторить дефект?
Ручные запросы
Только под нагрузкой

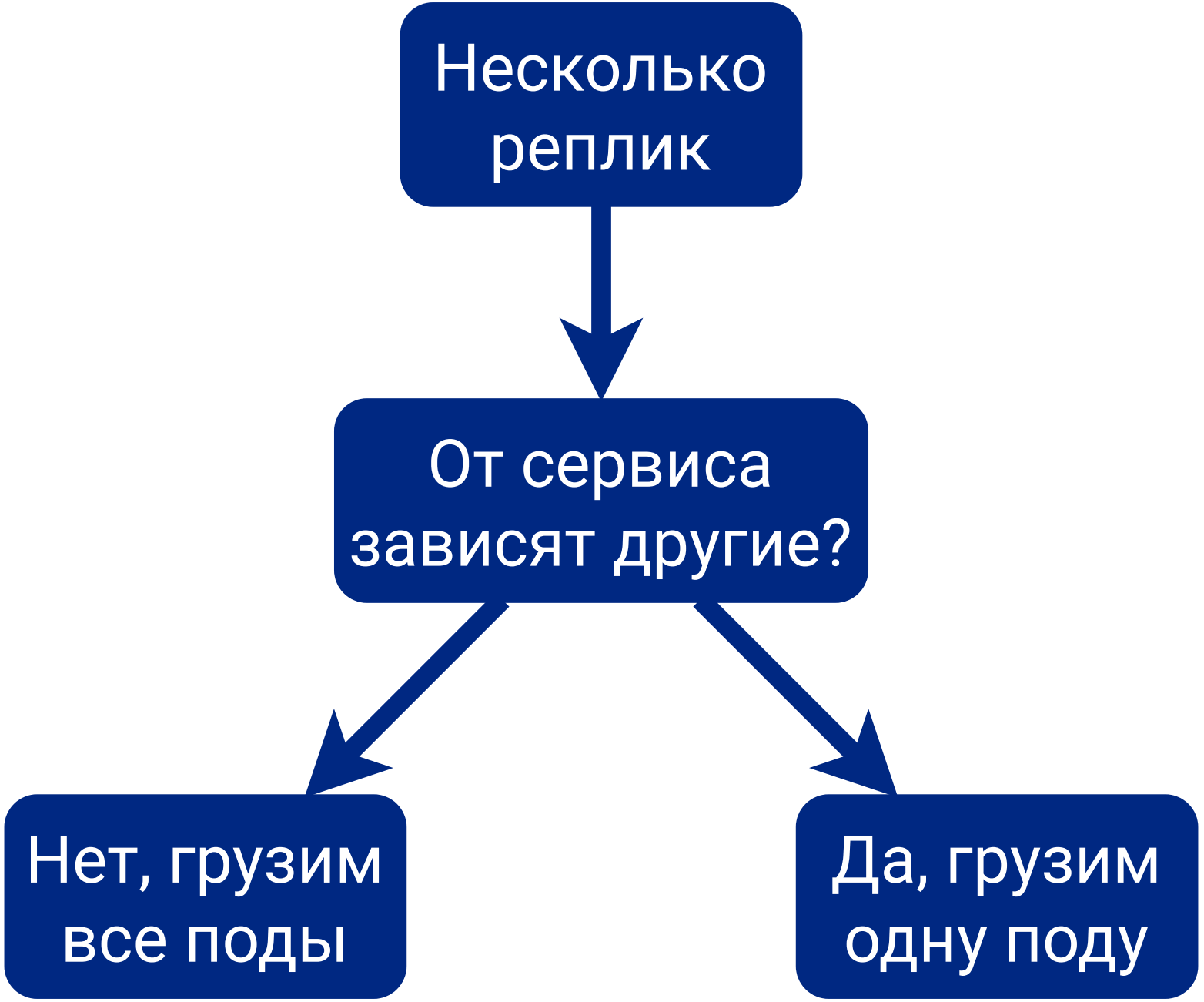

Как повторить дефект?
Ручные запросы
Только под нагрузкой
Одна реплика

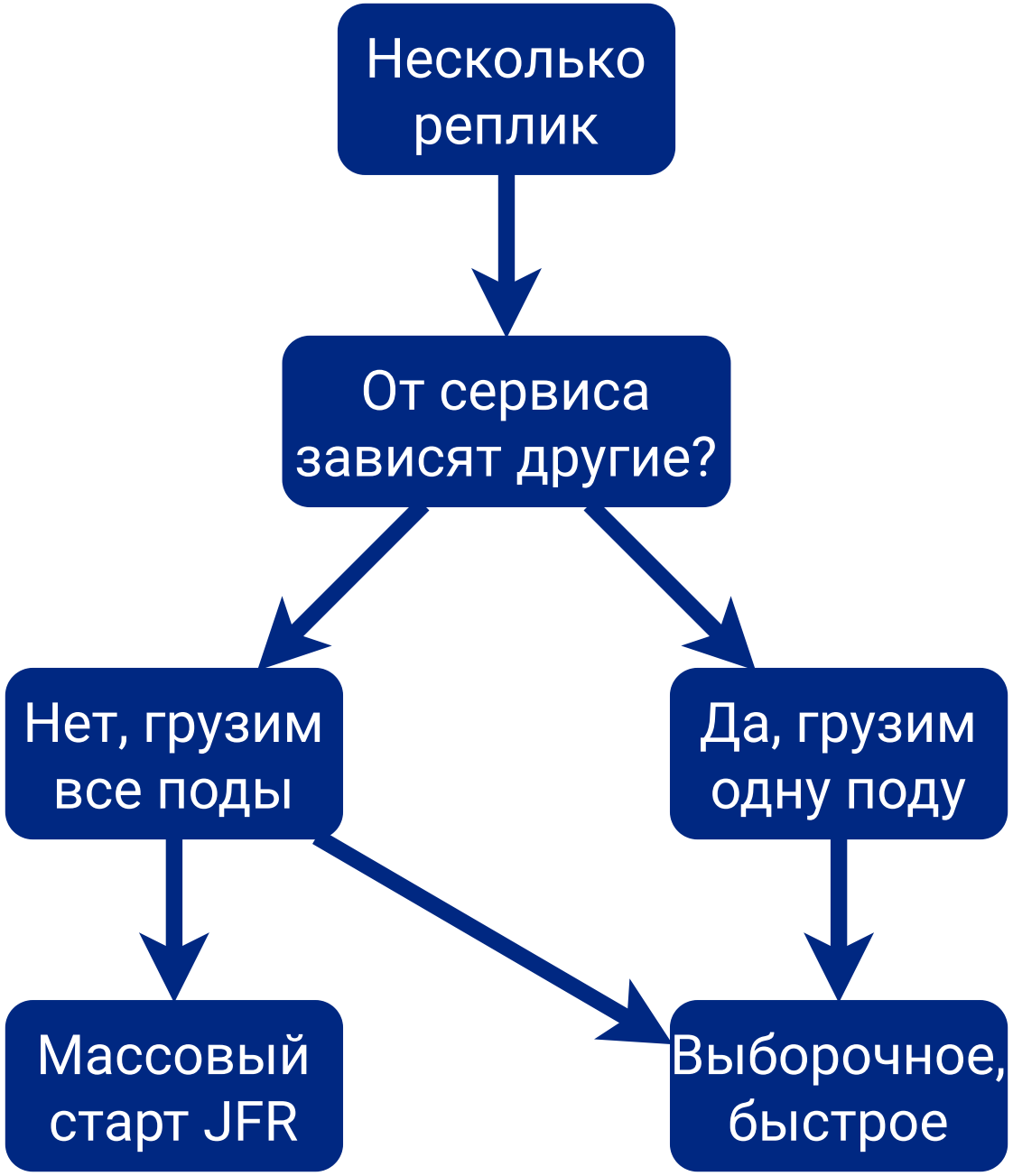

Ручные запросы
Одна реплика
Удаленно
Инструментация

Выберу инструмент для детального профилирования

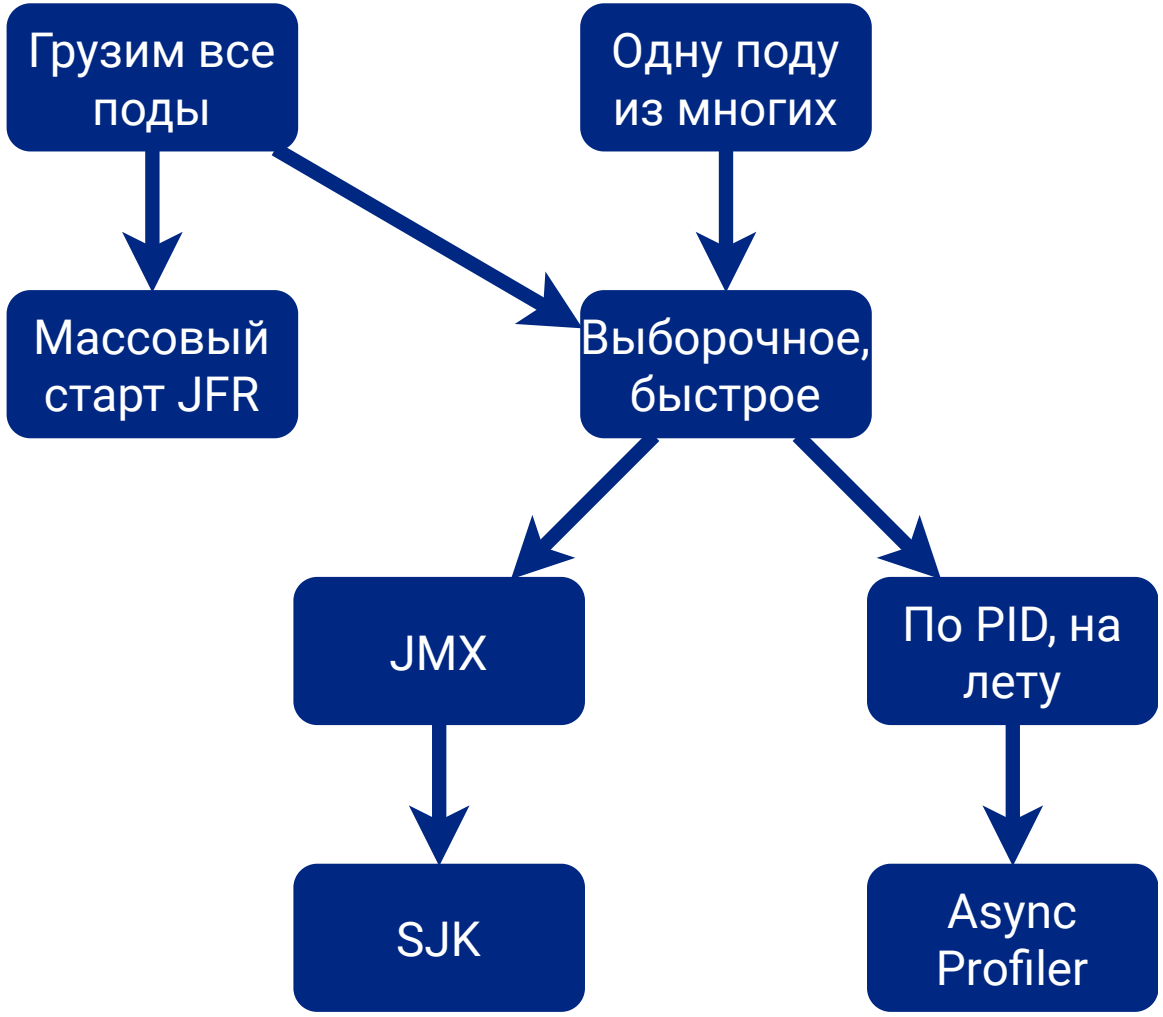



Как повторить дефект?
Ручные запросы
Только под нагрузкой
Несколько реплик


Несколько реплик
От сервиса зависят другие?
Нет, грузим все поды
Да, грузим одну поду

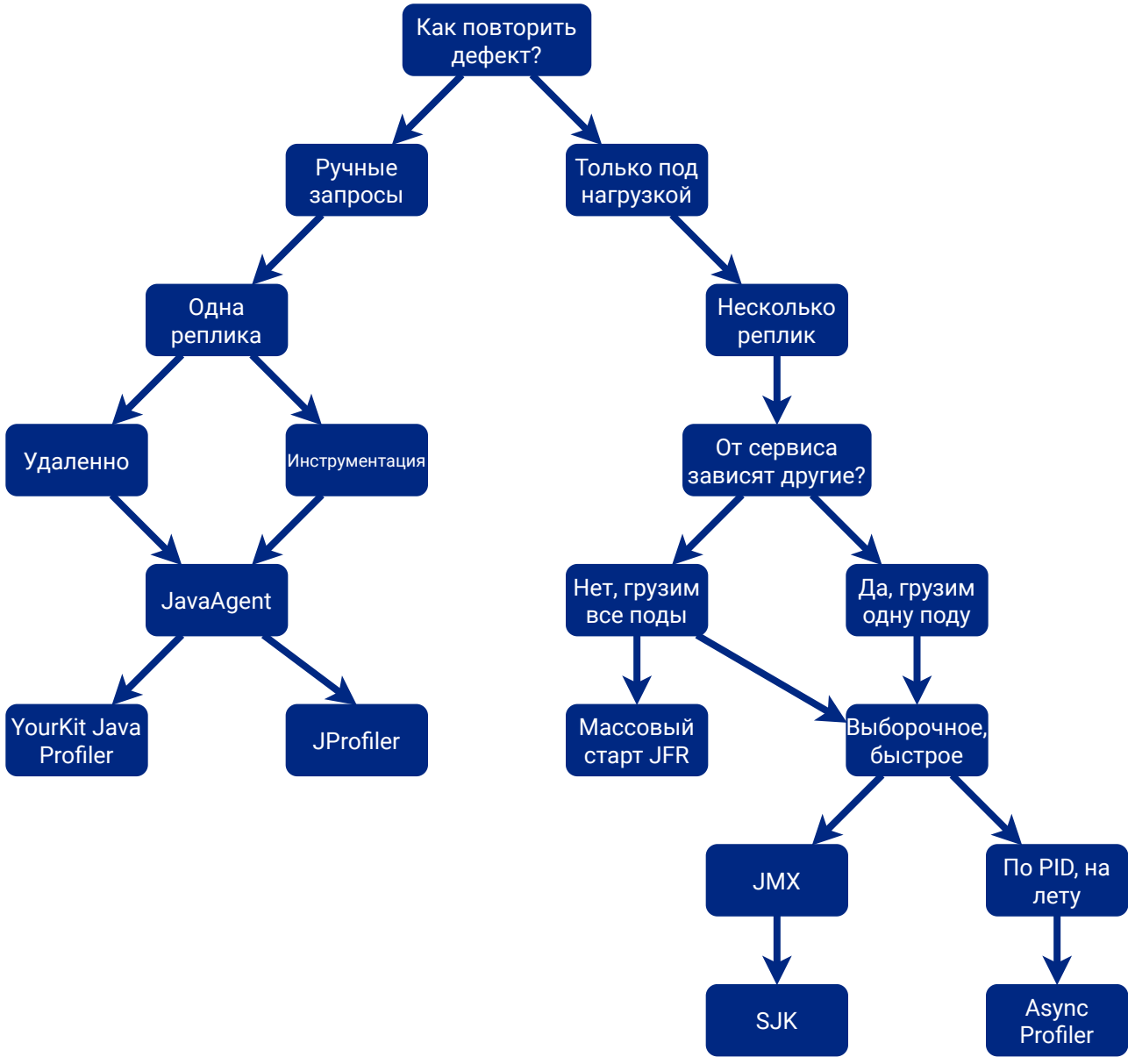

Несколько реплик
От сервиса зависят другие?
Нет, грузим все поды
Да, грузим одну поду
Массовый старт JFR
Выборочное, быстрое

Выберу инструмент с наименьшим замедлением



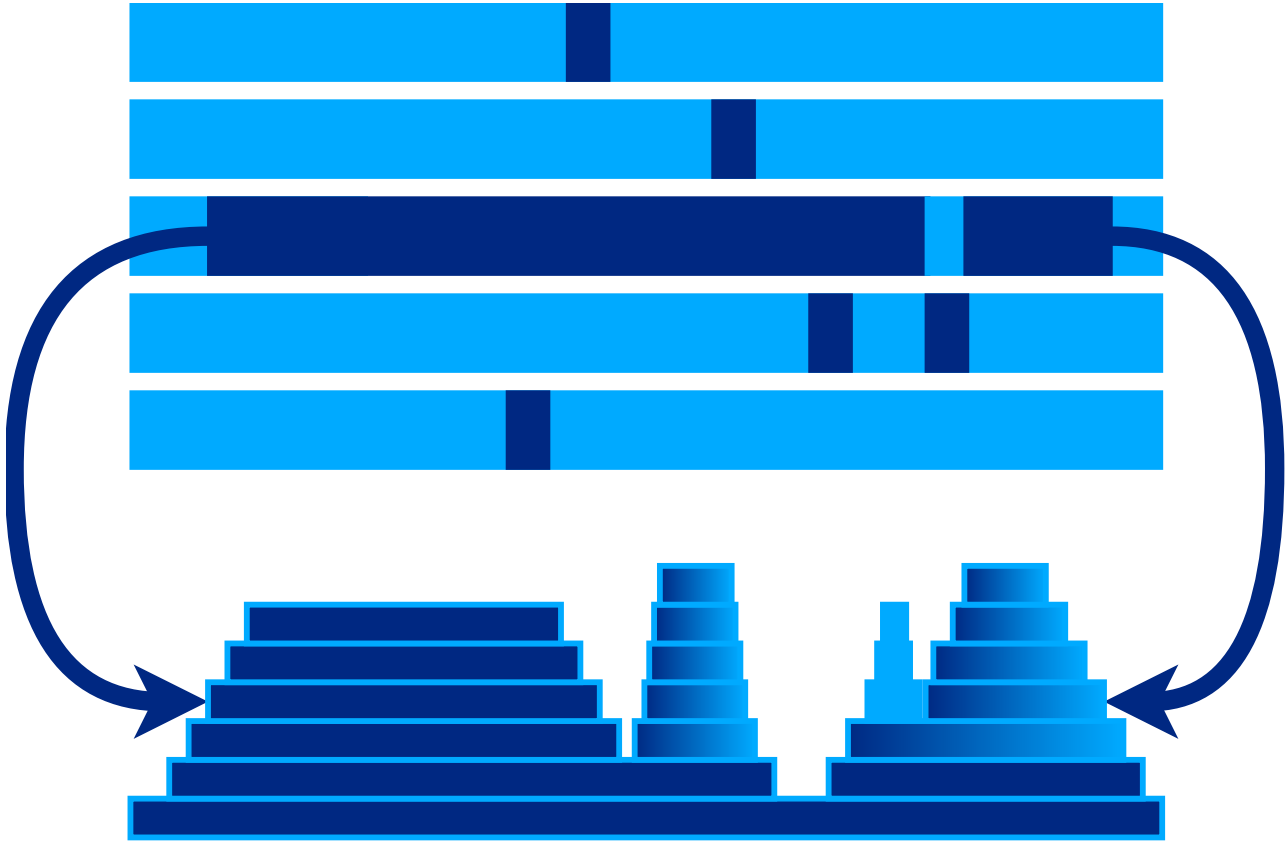
Выберу путь наименьшего сопротивления

От задачи к инструменту



Как анализировать результаты профилирования

Анализ



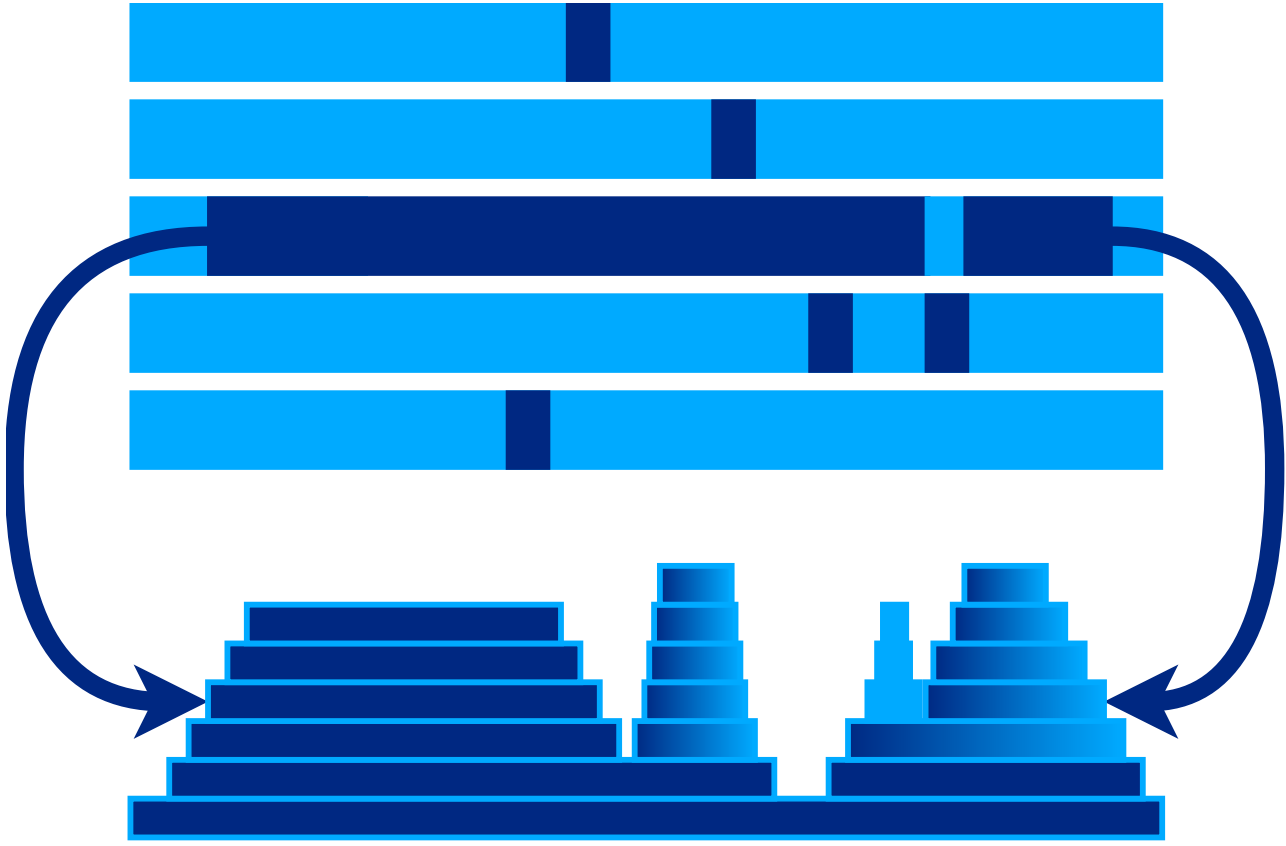
Как анализировать результаты профилирования

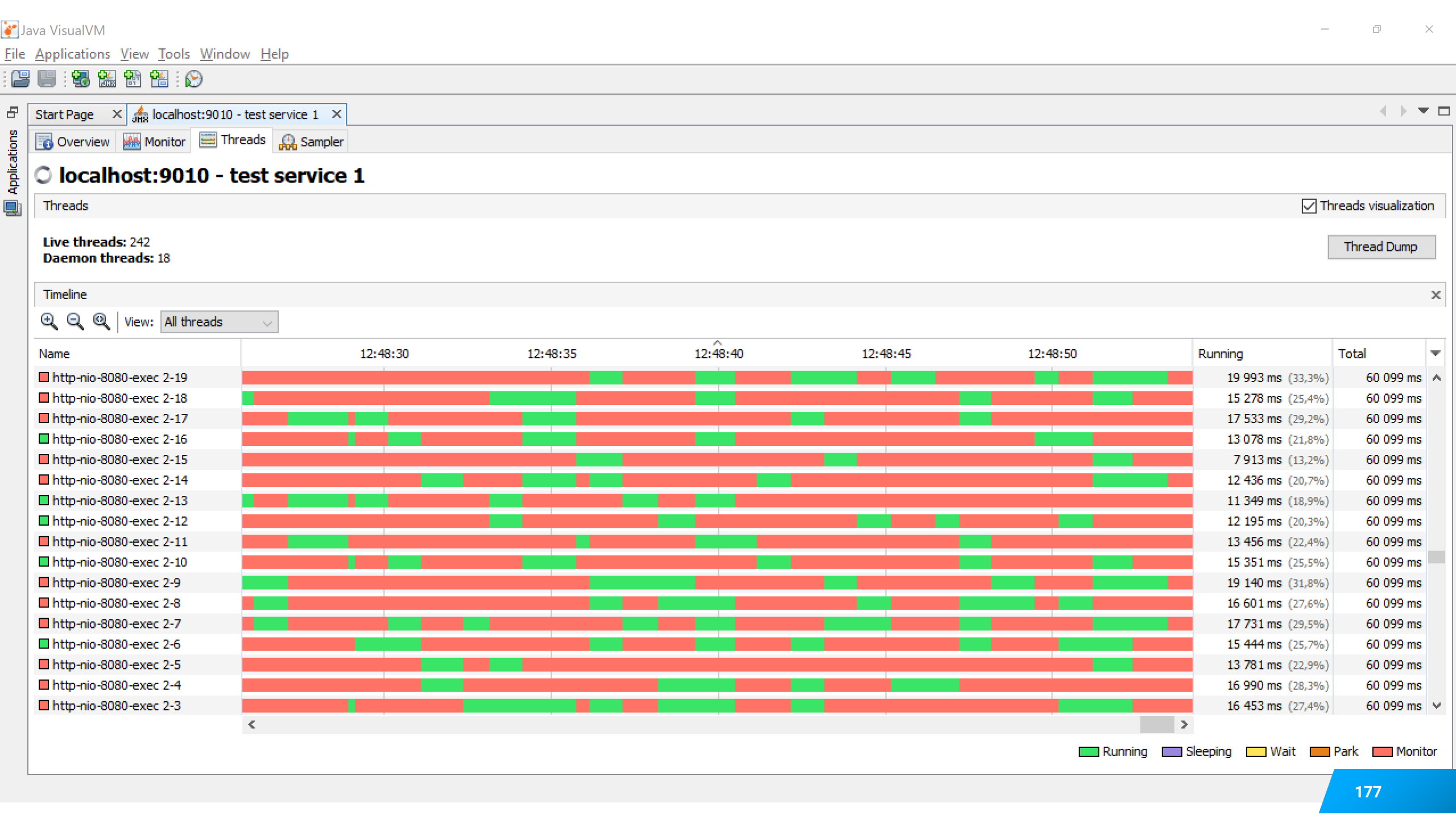
Анализ результатов семплирования

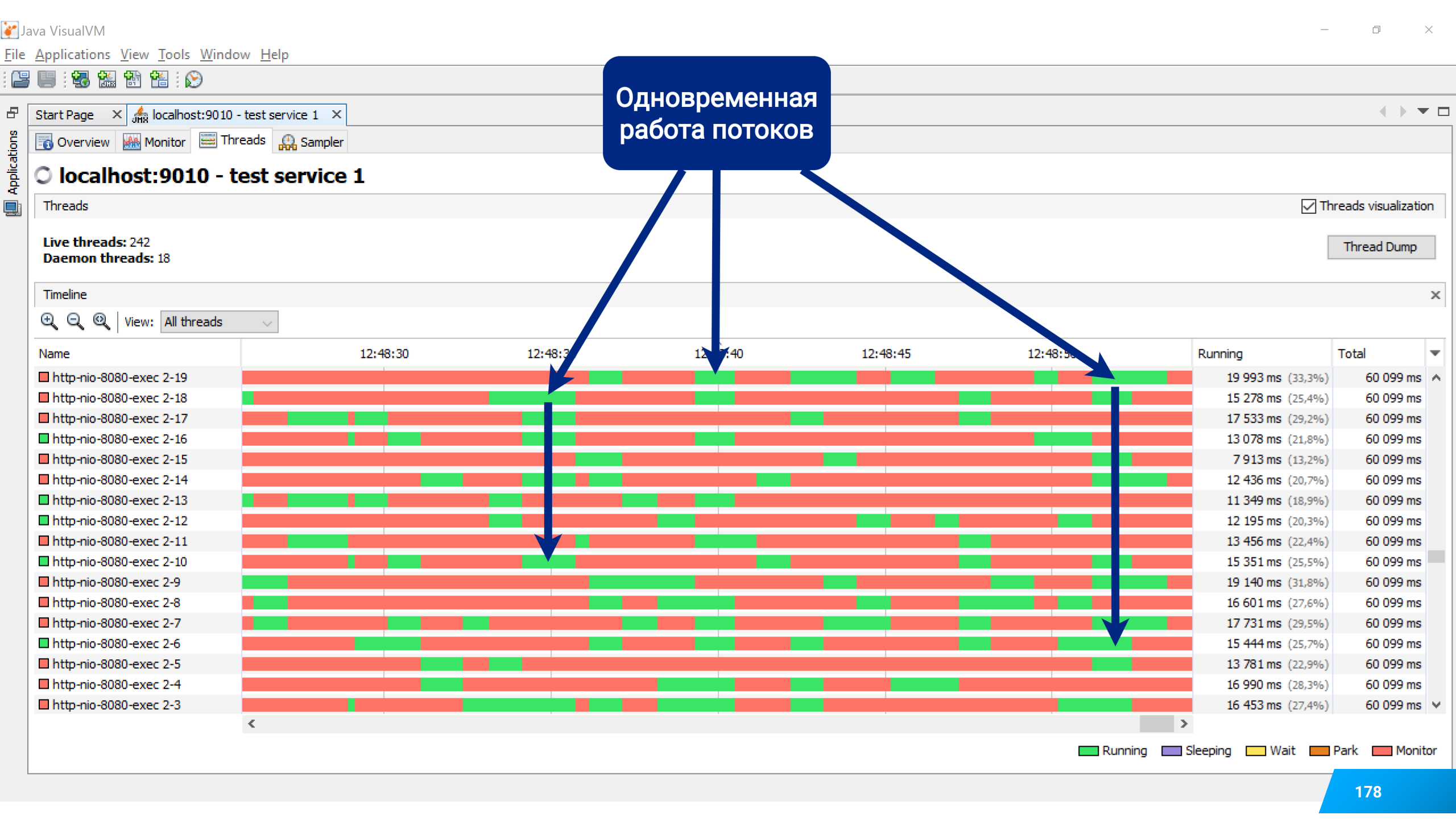
1. Визуально оценить работу потоков
2. Собрать статистику по работе потоков
3. Выбрать проблемные потоки, исключить несущественные
4. Собрать статистику и отчет только по выбранным потокам
5. Выбрать проблемные методы, выделить их в статистике
6. Выделить ожидание внешних сервисов и систем
7. Наложить статистику по программный код сервиса

Как визуально оценить работу потоков

Анализ

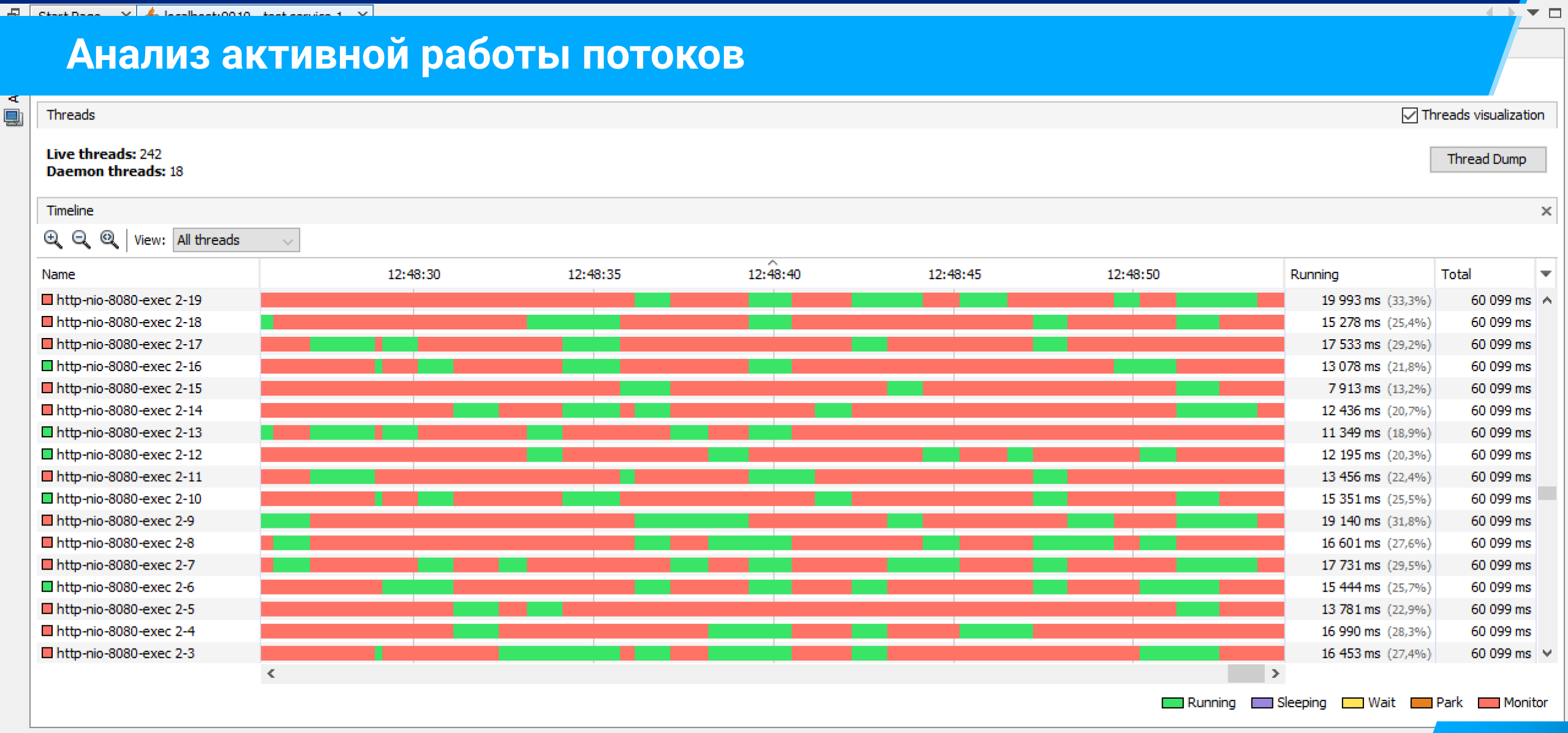



Java VisualVM
File Applications View Tools Window Help
Start Page
localhost:9010 - test service 1
Overview Monitor Threads Sampler
Applications
localhost:9010 - test service 1
Threads
Threads visualization
Live threads: 242
Daemon threads: 18
Thread Dump
Timeline
View: All threads
Name
12:48:30
12:48:35
12:48:40
12:48:45
12:48:50
Running
Total
http-nio-8080-exec 2-19 19 993 ms (33,3%) 60 099 ms
http-nio-8080-exec 2-18 15 278 ms (25,4%) 60 099 ms
http-nio-8080-exec 2-17 17 533 ms (29,2%) 60 099 ms
http-nio-8080-exec 2-16 13 078 ms (21,8%) 60 099 ms
http-nio-8080-exec 2-15 7 913 ms (13,2%) 60 099 ms
http-nio-8080-exec 2-14 12 436 ms (20,7%) 60 099 ms
http-nio-8080-exec 2-13 11 349 ms (18,9%) 60 099 ms
http-nio-8080-exec 2-12 12 195 ms (20,3%) 60 099 ms
http-nio-8080-exec 2-11 13 456 ms (22,4%) 60 099 ms
http-nio-8080-exec 2-10 15 351 ms (25,5%) 60 099 ms
http-nio-8080-exec 2-9 19 140 ms (31,8%) 60 099 ms
http-nio-8080-exec 2-8 16 601 ms (27,6%) 60 099 ms
http-nio-8080-exec 2-7 17 731 ms (29,5%) 60 099 ms
http-nio-8080-exec 2-6 15 444 ms (25,7%) 60 099 ms
http-nio-8080-exec 2-5 13 781 ms (22,9%) 60 099 ms
http-nio-8080-exec 2-4 16 990 ms (28,3%) 60 099 ms
http-nio-8080-exec 2-3 16 453 ms (27,4%) 60 099 ms
Running Sleeping Wait Park Monitor


Java VisualVM
File Applications View Tools Window Help
Start Page
localhost:9010 - test service 1
Overview Monitor Threads Sampler
localhost:9010 - test service 1
Threads
Threads visualization
Live threads: 242
Daemon threads: 18
Thread Dump
Timeline
View: All threads
Name
12:48:30
12:48:45
Running
Total
http-nio-8080-exec 2-19 19 993 ms (33,3%) 60 099 ms
http-nio-8080-exec 2-18 15 278 ms (25,4%) 60 099 ms
http-nio-8080-exec 2-17 17 533 ms (29,2%) 60 099 ms
http-nio-8080-exec 2-16 13 078 ms (21,8%) 60 099 ms
http-nio-8080-exec 2-15 7 913 ms (13,2%) 60 099 ms
http-nio-8080-exec 2-14 12 436 ms (20,7%) 60 099 ms
http-nio-8080-exec 2-13 11 349 ms (18,9%) 60 099 ms
http-nio-8080-exec 2-12 12 195 ms (20,3%) 60 099 ms
http-nio-8080-exec 2-11 13 456 ms (22,4%) 60 099 ms
http-nio-8080-exec 2-10 15 351 ms (25,5%) 60 099 ms
http-nio-8080-exec 2-9 19 140 ms (31,8%) 60 099 ms
http-nio-8080-exec 2-8 16 601 ms (27,6%) 60 099 ms
http-nio-8080-exec 2-7 17 731 ms (29,5%) 60 099 ms
http-nio-8080-exec 2-6 15 444 ms (25,7%) 60 099 ms
http-nio-8080-exec 2-5 13 781 ms (22,9%) 60 099 ms
http-nio-8080-exec 2-4 16 990 ms (28,3%) 60 099 ms
http-nio-8080-exec 2-3 16 453 ms (27,4%) 60 099 ms
Running Sleeping Wait Park Monitor
Одновременная работа потоков

Потоков много и они работают одновременно

Анализ активной работы потоков

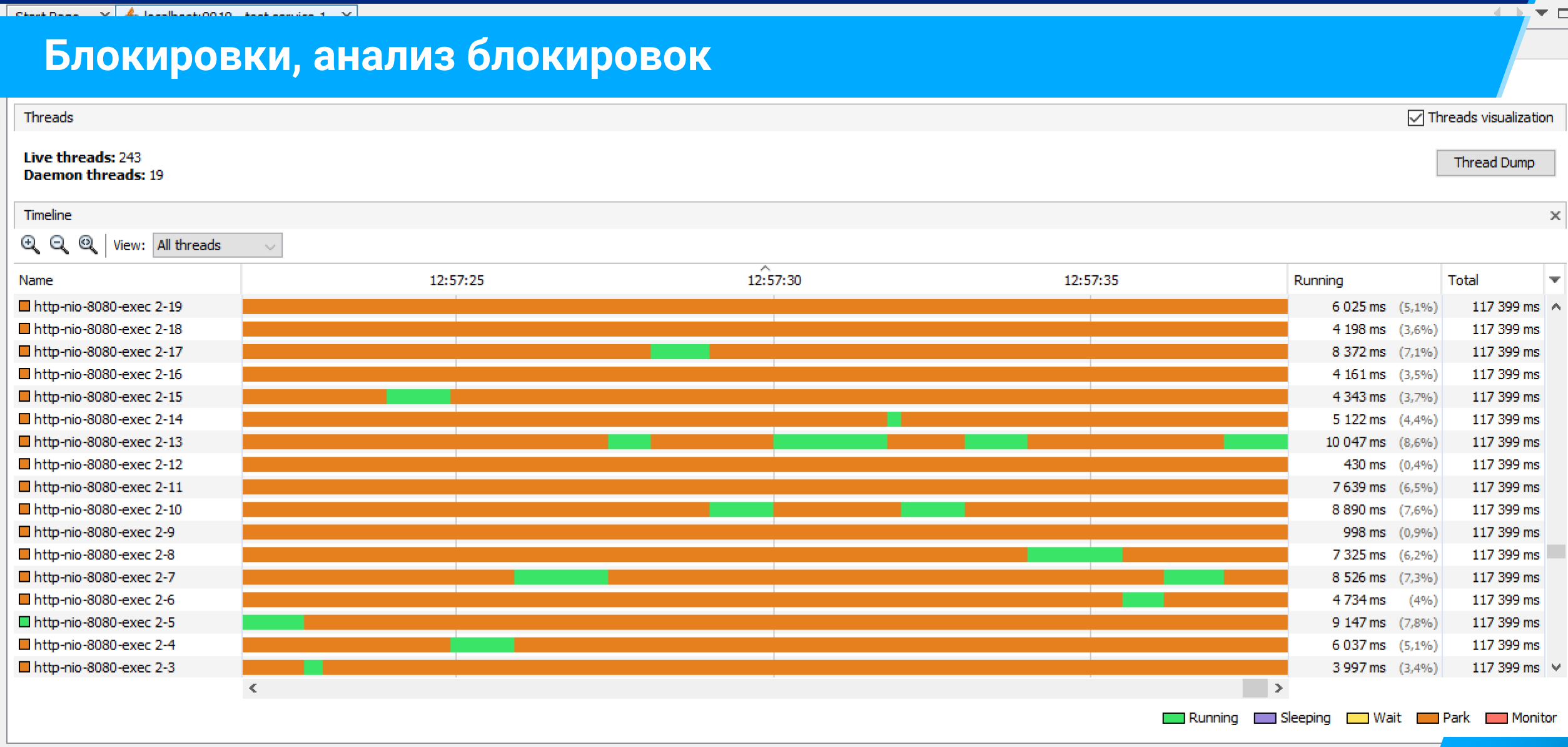



Java VisualVM
File Applications View Tools Window Help
Start Page
localhost:9010 - test service 1
Overview Monitor Threads Sampler
localhost:9010 - test service 1
Threads
Threads visualization
Live threads: 243
Daemon threads: 19
Thread Dump
Timeline
View: All threads
Name
12:57:25
12:57:30
12:57:35
Running
Total
http-nio-8080-exec 2-19 6 025 ms (5,1%) 117 399 ms
http-nio-8080-exec 2-18 4 198 ms (3,6%) 117 399 ms
http-nio-8080-exec 2-17 8 372 ms (7,1%) 117 399 ms
http-nio-8080-exec 2-16 4 161 ms (3,5%) 117 399 ms
http-nio-8080-exec 2-15 4 343 ms (3,7%) 117 399 ms
http-nio-8080-exec 2-14 5 122 ms (4,4%) 117 399 ms
http-nio-8080-exec 2-13 10 047 ms (8,6%) 117 399 ms
http-nio-8080-exec 2-12 430 ms (0,4%) 117 399 ms
http-nio-8080-exec 2-11 7 639 ms (6,5%) 117 399 ms
http-nio-8080-exec 2-10 8 890 ms (7,6%) 117 399 ms
http-nio-8080-exec 2-9 998 ms (0,9%) 117 399 ms
http-nio-8080-exec 2-8 7 325 ms (6,2%) 117 399 ms
http-nio-8080-exec 2-7 8 526 ms (7,3%) 117 399 ms
http-nio-8080-exec 2-6 4 734 ms (4%) 117 399 ms
http-nio-8080-exec 2-5 9 147 ms (7,8%) 117 399 ms
http-nio-8080-exec 2-4 6 037 ms (5,1%) 117 399 ms
http-nio-8080-exec 2-3 3 997 ms (3,4%) 117 399 ms
Running Sleeping Wait Park Monitor

Потоков много, но нет параллельности работы

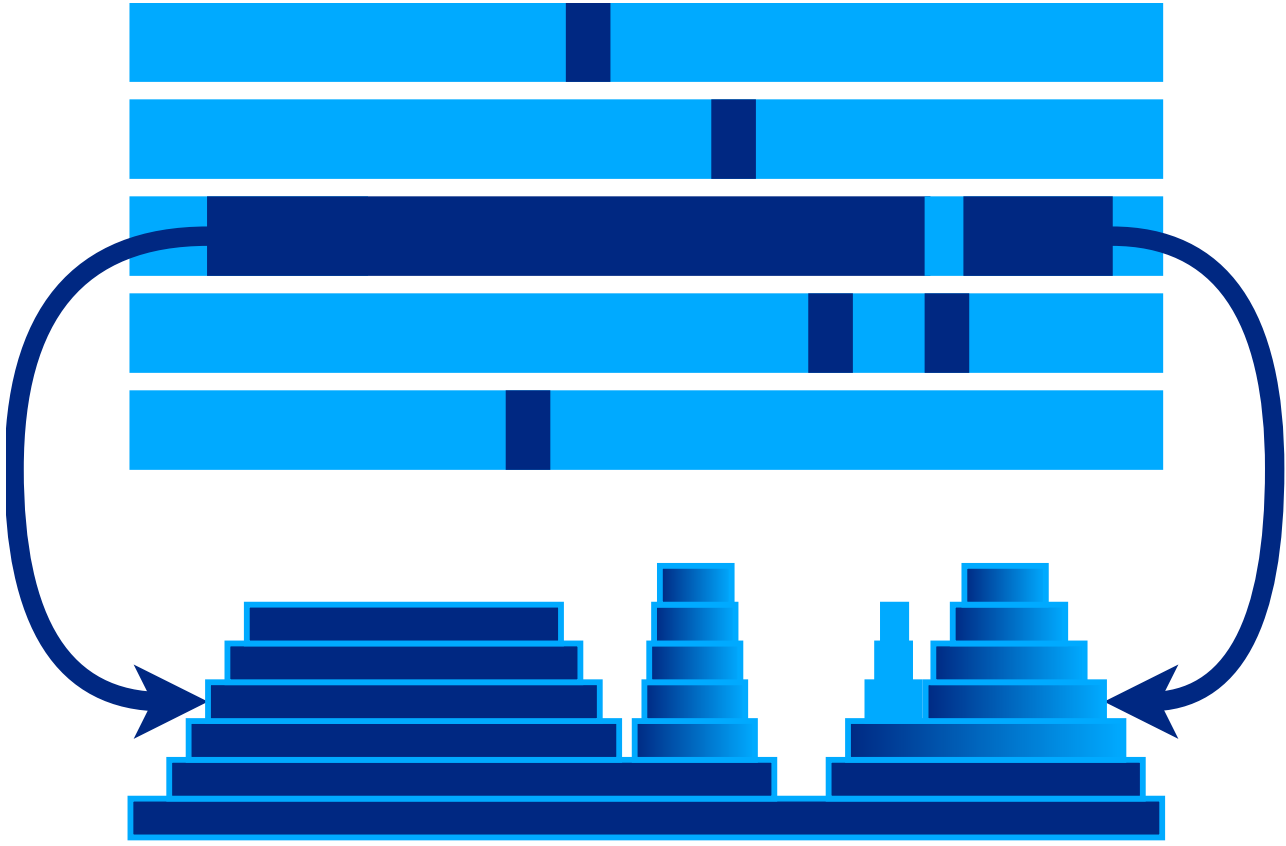
Потоков много, но нет параллельности работы

Блокировки, анализ блокировок



Ключевые потоки JVM для Spring Boot сервиса

Анализ



Что можно исключить из детального анализа

Достаточно статистику посмотреть

- RMI и JMX - это само профилирование и мониторинг
    - RMI Scheduler
    - RMI TCP Accept
    - RMI TCP Connection
    - JMX server connection timeout
- kafka-coordinator-heartbeat-thread
- Reference Handler
- Finalizer
- GC Daemon

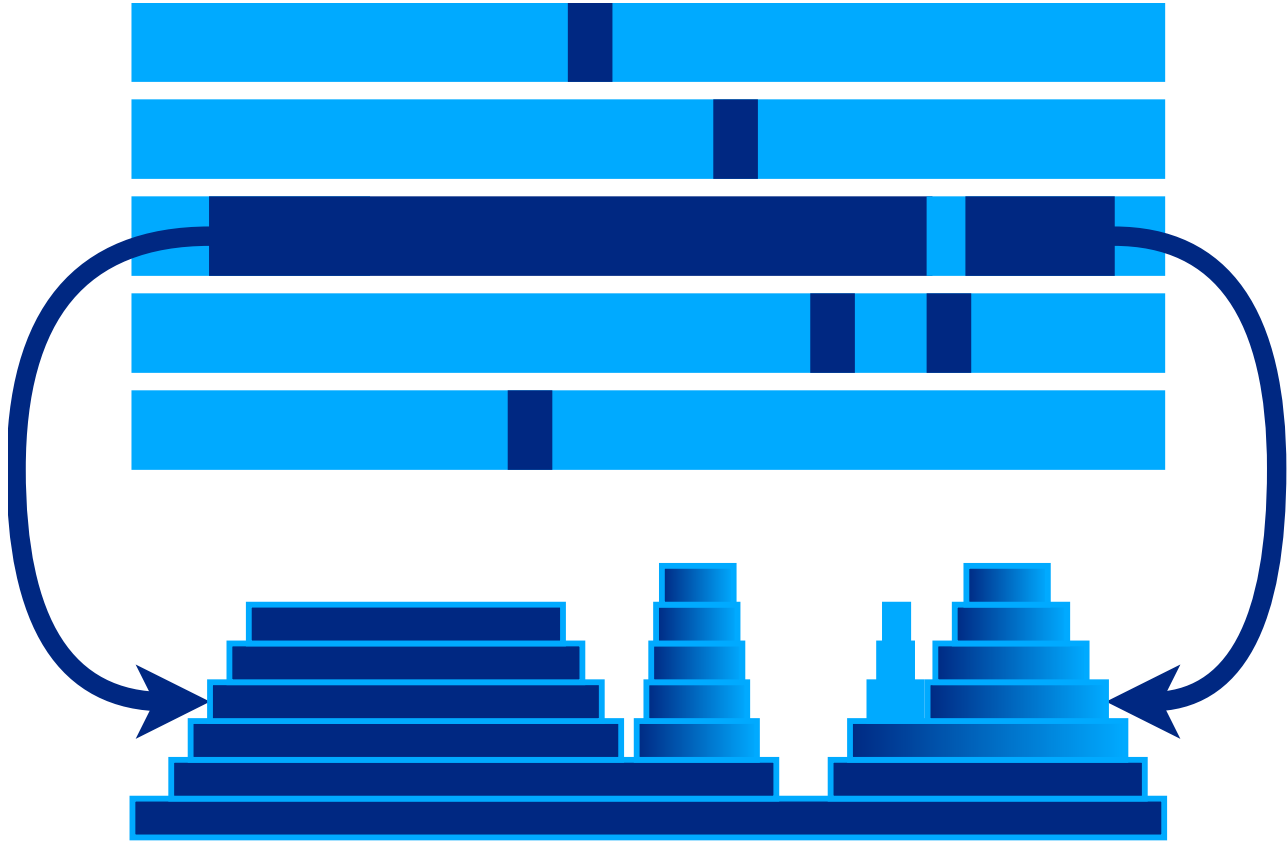
Ключевые потоки

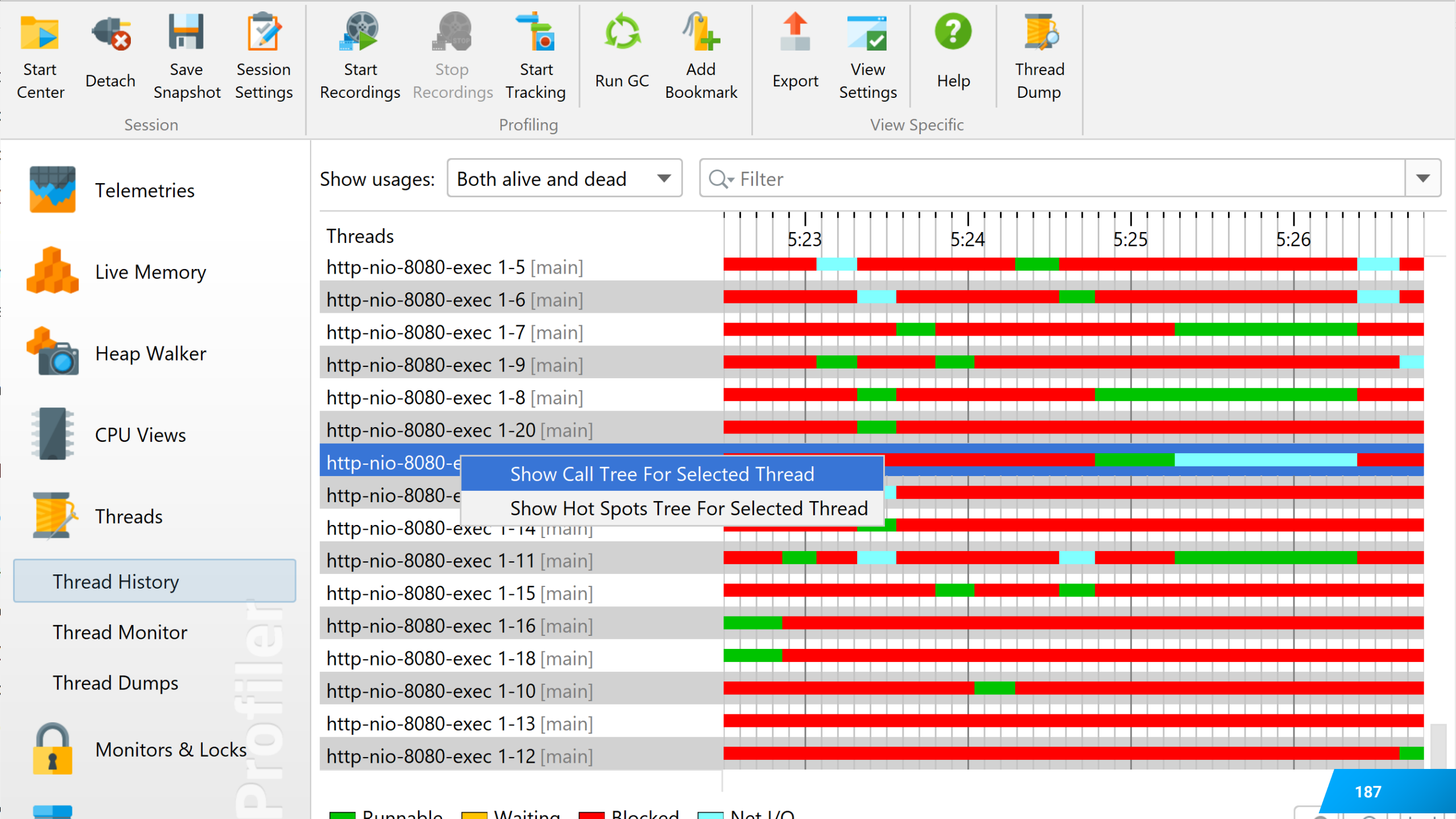
Посмотреть на статистику и заглянуть внутрь

- http-nio-8080-exec-номер
  - http-nio-8080-exec-1
  - http-nio-8080-exec-2
- Thread-pool-номер
- Thread-номер
- OkHttp
- WebSocket
- SockJS

Статистика выполнения кода в одном потоке

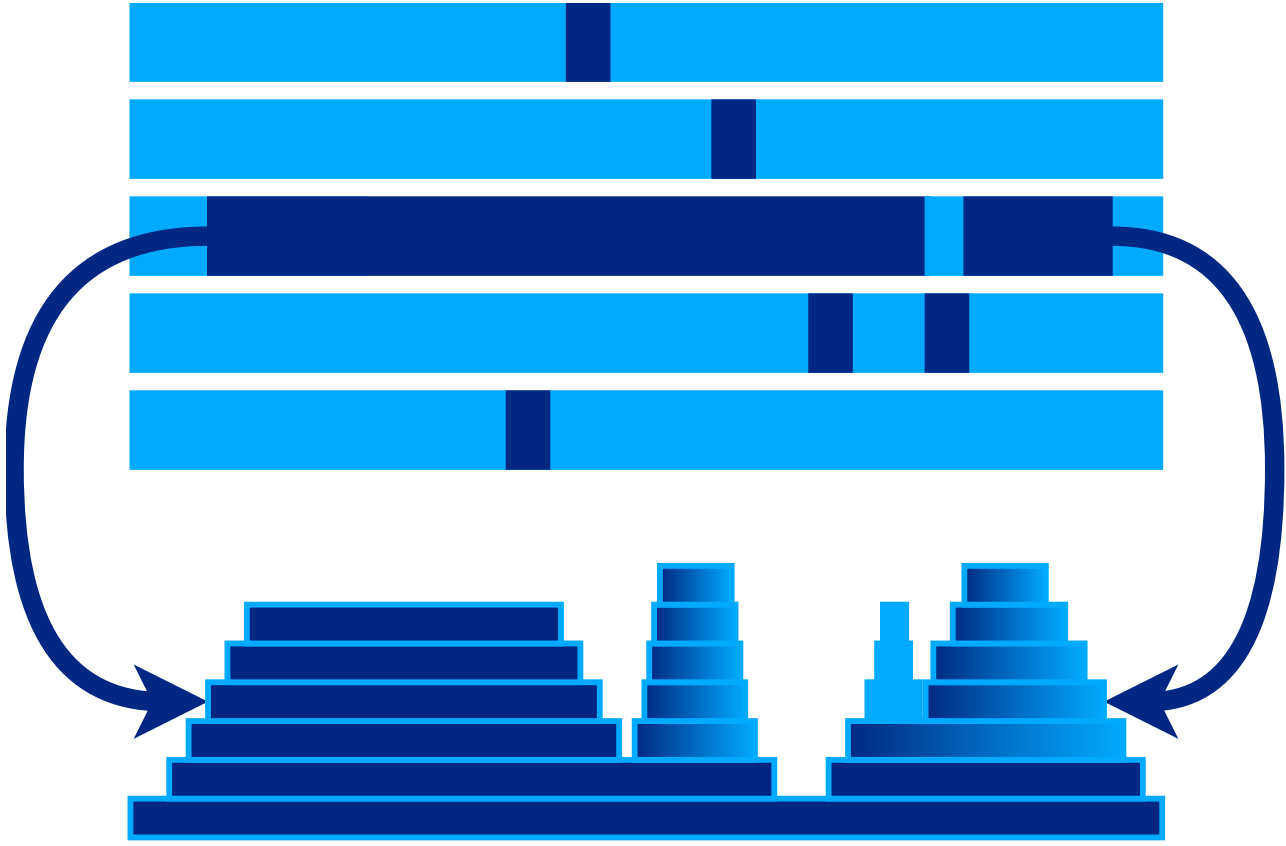
Анализ




Start Center
Detach
Save Snapshot
Session Settings
Session
Start Recordings
Stop Recordings
Start Tracking
Run GC
Add Bookmark
Profiling
Export
View Settings
Help
Thread Dump
View Specific
Telemetries
Live Memory
Heap Walker
CPU Views
Threads
Thread History
Thread Monitor
Thread Dumps
Monitors & Locks
Show usages:
Both alive and dead
Filter
Threads
5:23
5:24
5:25
5:26
http-nio-8080-exec 1-5 [main]
http-nio-8080-exec 1-6 [main]
http-nio-8080-exec 1-7 [main]
http-nio-8080-exec 1-9 [main]
http-nio-8080-exec 1-8 [main]
http-nio-8080-exec 1-20 [main]
Show Call Tree For Selected Thread
Show Hot Spots Tree For Selected Thread
http-nio-8080-exec 1-11 [main]
http-nio-8080-exec 1-15 [main]
http-nio-8080-exec 1-16 [main]
http-nio-8080-exec 1-18 [main]
http-nio-8080-exec 1-10 [main]
http-nio-8080-exec 1-13 [main]
http-nio-8080-exec 1-12 [main]
Profiler
187

Статистика выполнения кода в нескольких потоках

Анализ



Задаем настройки и потоки для анализа

```
#!/bin/bash -x

reportFolder=$(pwd)
java="java"
sjk="$java -jar ../sjk-0.17.jar"

declare -A threads
threads=(\
  ["_all"]='.+' \
  ["http-nio-8080-exec"]='http-nio-8080-exec.+' \
  ["pool-3-thread"]='pool-3-thread-.+' \
  ["OkHttp https"]='OkHttp https.+' \
  ["Thread-2"]='Thread-2' \
```

Выделяем статистику по потокам в файл

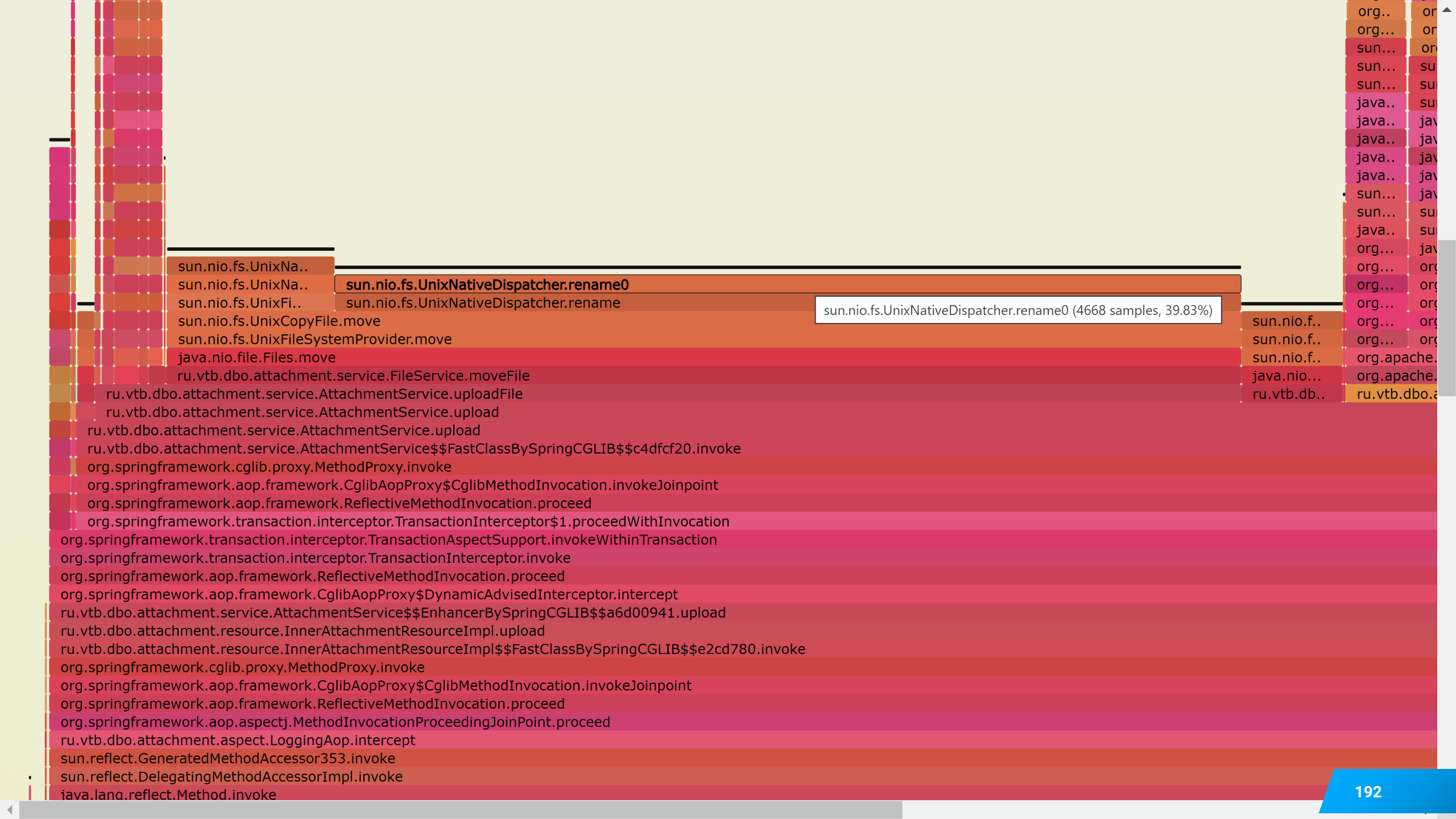
И строим гистограмму по выделенной статистике

```
for thread in "${!threads[@]}"
do
  mkdir "${reportFolder}/${thread}"

  $sjk \
  stcpy \
  --input "${reportFolder}/sjk.sdt"  \
  --thread-name "${threads[$thread]}" \
  --trace-filter "!sun.misc.Unsafe.park" \
  --output "${reportFolder}/${thread}.sdt"

  $sjk \
  ssa \
  --file "${reportFolder}/${thread}.sdt" \
  --histo \
  > "${reportFolder}/${thread}/${thread}.histo.txt"
done
```

Выделяем статистику по потокам в файл

Или flame-диаграмму по выделенной статистике

```
for thread in "${!threads[@]}"
do
  mkdir "${reportFolder}/${thread}"

  $sjk \
  stcpy \
  --input "${reportFolder}/sjk.sdt"  \
  --thread-name "${threads[$thread]}" \
  --trace-filter "!sun.misc.Unsafe.park" \
  --output "${reportFolder}/${thread}.sdt"

  $sjk \
  ssa \
  --file  "$reportFolder/${thread}.sdt" \
  --flame \
  --width 2000 \
  > "${reportFolder}/${thread}/${thread}.flame.svg"
```


sun.nio.fs.UnixNa..
sun.nio.fs.UnixNa..
sun.nio.fs.UnixNativeDispatcher.rename0
sun.nio.fs.UnixFi..
sun.nio.fs.UnixNativeDispatcher.rename
sun.nio.fs.UnixNativeDispatcher.rename0 (4668 samples, 39.83%)
sun.nio.fs.UnixCopyFile.move
sun.nio.fs.UnixFileSystemProvider.move
java.nio.file.Files.move
ru.vtb.dbo.attachment.service.FileService.moveFile
ru.vtb.dbo.attachment.service.AttachmentService.uploadFile
ru.vtb.dbo.attachment.service.AttachmentService.upload
ru.vtb.dbo.attachment.service.AttachmentService.upload
ru.vtb.dbo.attachment.service.AttachmentService$$FastClassBySpringCGLIB$$c4dfcf20.invoke
org.springframework.cglib.proxy.MethodProxy.invoke
org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invokeJoinpoint
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.transaction.interceptor.TransactionInterceptor$1.proceedWithInvocation
org.springframework.transaction.interceptor.TransactionAspectSupport.invokeWithinTransaction
org.springframework.transaction.interceptor.TransactionInterceptor.invoke
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.aop.framework.CglibAopProxy$DynamicAdvisedInterceptor.intercept
ru.vtb.dbo.attachment.service.AttachmentService$$EnhancerBySpringCGLIB$$a6d00941.upload
ru.vtb.dbo.attachment.resource.InnerAttachmentResourceImpl.upload
ru.vtb.dbo.attachment.resource.InnerAttachmentResourceImpl$$FastClassBySpringCGLIB$$e2cd780.invoke
org.springframework.cglib.proxy.MethodProxy.invoke
org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invokeJoinpoint
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.aop.aspectj.MethodInvocationProceedingJoinPoint.proceed
ru.vtb.dbo.attachment.aspect.LoggingAop.intercept
sun.reflect.GeneratedMethodAccessor353.invoke
sun.reflect.DelegatingMethodAccessorImpl.invoke
java.lang.reflect.Method.invoke
sun.nio.f..
sun.nio.f..
sun.nio.f..
java.nio...
ru.vtb.db..
org..
org...
sun...
sun...
sun...
java..
java..
java..
java..
java..
sun...
sun...
java..
org...
org...
org...
org...
org...
org...
org.apache.
org.apache.
ru.vtb.dbo.a

File Edit Selection Find View Goto Tools Project Preferences Help

http-nio-8080-exec.histo.txt

Trc	(%)	Frm N	Term	(%)	Frame
4668	39%	4668	4668	39%	sun.nio.fs.UnixNativeDispatcher.rename0(Native Method)
2732	23%	2732	2732	23%	java.lang.Throwable.fillInStackTrace(Native Method)
957	8%	957	957	8%	java.net.SocketInputStream.socketRead0(Native Method)
865	7%	865	862	7%	sun.nio.fs.UnixNativeDispatcher.lstat0(Native Method)
613	5%	613	613	5%	sun.nio.fs.UnixNativeDispatcher.access0(Native Method)
137	1%	137	137	1%	sun.misc.URLClassPath.getLoader(URLClassPath.java:504)
132	1%	132	132	1%	org.apache.tika.io.LookaheadInputStream.<init>(LookaheadInputStream.java:68)
112	0%	112	112	0%	sun.nio.fs.UnixNativeDispatcher.open0(Native Method)
100	0%	100	100	0%	java.util.zip.Inflater.inflateBytes(Native Method)
85	0%	85	77	0%	java.lang.UNIXProcess.forkAndExec(Native Method)
72	0%	72	72	0%	sun.nio.ch.FileDispatcherImpl.read0(Native Method)
59	0%	59	59	0%	sun.nio.ch.FileDispatcherImpl.close0(Native Method)
57	0%	57	57	0%	java.util.Arrays.copyOfRange(Arrays.java:3664)
55	0%	55	55	0%	java.net.SocketOutputStream.socketWrite0(Native Method)
3352	28%	3917	47	0%	java.security.AccessController.doPrivileged(Native Method)
40	0%	40	40	0%	java.lang.ClassLoader.findLoadedClass0(Native Method)
34	0%	34	34	0%	java.lang.Object.hashCode(Native Method)
34	0%	34	34	0%	java.lang.String.indexOf(String.java:1769)
31	0%	31	31	0%	org.apache.catalina.webresources.CachedResource.validateResources(CachedResource.java:138)
31	0%	31	31	0%	java.lang.AbstractStringBuilder.<init>(AbstractStringBuilder.java:68)
91	0%	91	29	0%	sun.misc.URLClassPath$Loader.findResource(URLClassPath.java:701)
88	0%	88	29	0%	java.net.URL.toExternalForm(URL.java:929)
29	0%	29	29	0%	sun.misc.URLClassPath.getNextLoader(URLClassPath.java:479)
27	0%	27	27	0%	sun.nio.ch.NativeThread.current(Native Method)
27	0%	27	27	0%	java.io.RandomAccessFile.open0(Native Method)
2747	23%	2747	21	0%	org.springframework.boot.loader.jar.JarURLConnection.getInputStream(JarURLConnection.java:170)
21	0%	21	21	0%	java.io.FileOutputStream.writeBytes(Native Method)
21	0%	21	21	0%	java.lang.System.identityHashCode(Native Method)
20	0%	20	19	0%	org.springframework.util.AntPathMatcher.doMatch(AntPathMatcher.java:195)
18	0%	18	18	0%	sun.nio.ch.FileDispatcherImpl.write0(Native Method)
16	0%	16	16	0%	org.springframework.boot.loader.jar.Handler.getFileFromContext(Handler.java:194)
39	0%	39	14	0%	java.net.URL.<init>(URL.java:622)
12	0%	12	12	0%	java.util.Arrays.binarySearch0(Arrays.java:1921)
12	0%	12	12	0%	sun.nio.ch.EPollArrayWrapper.interrupt(Native Method)
11	0%	11	11	0%	java.net.Inet6AddressImpl.getHostByAddr(Native Method)
11	0%	11	11	0%	java.util.Arrays.copyOf(Arrays.java:3332)
11	0%	11	11	0%	java.util.zip.ZipFile.read(Native Method)
11	0%	11	11	0%	java.lang.Class.isAssignableFrom(Native Method)
11	0%	11	11	0%	java.io.UnixFileSystem.getBooleanAttributes0(Native Method)
48	0%	48	10	0%	java.util.HashMap.hash(HashMap.java:339)
10	0%	10	10	0%	sun.reflect.Reflection.getCallerClass(Native Method)

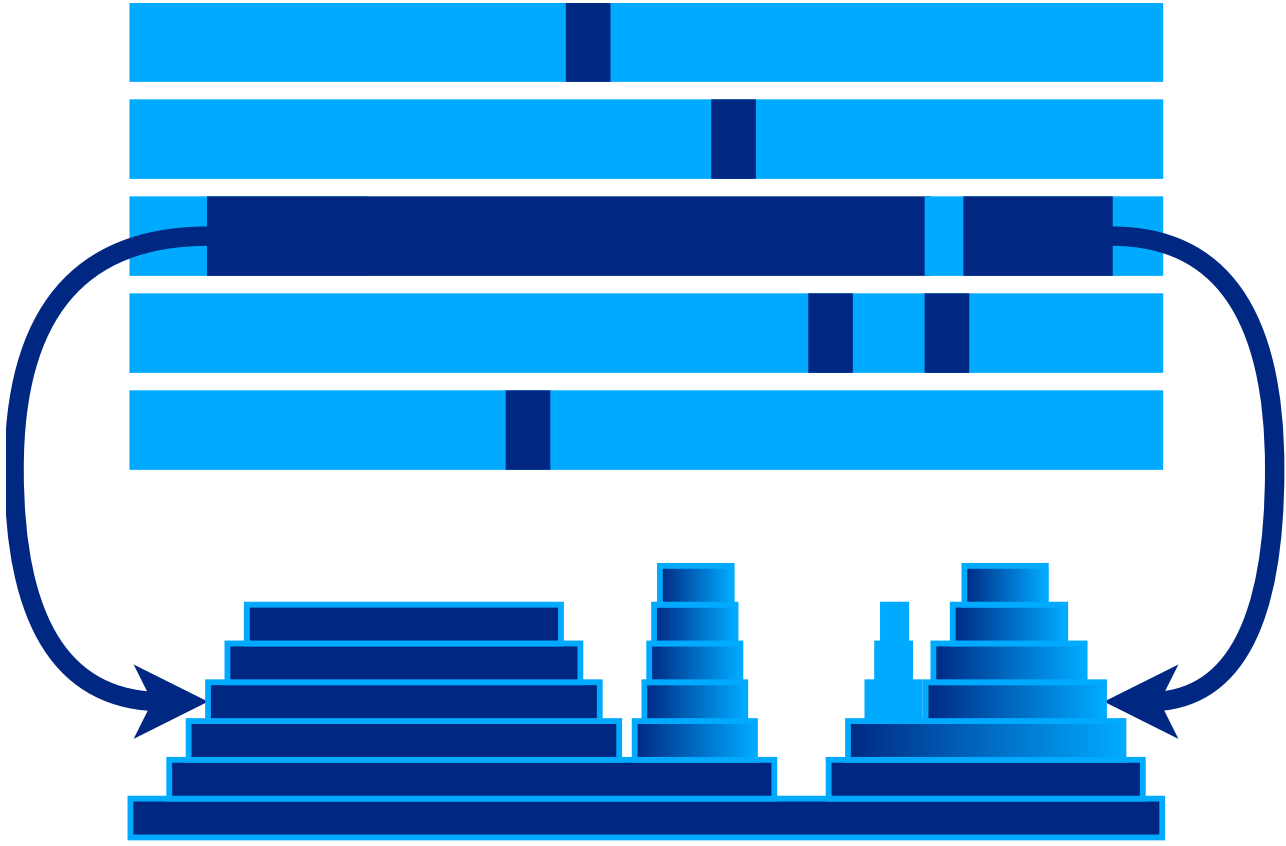
Line 1, Column 1

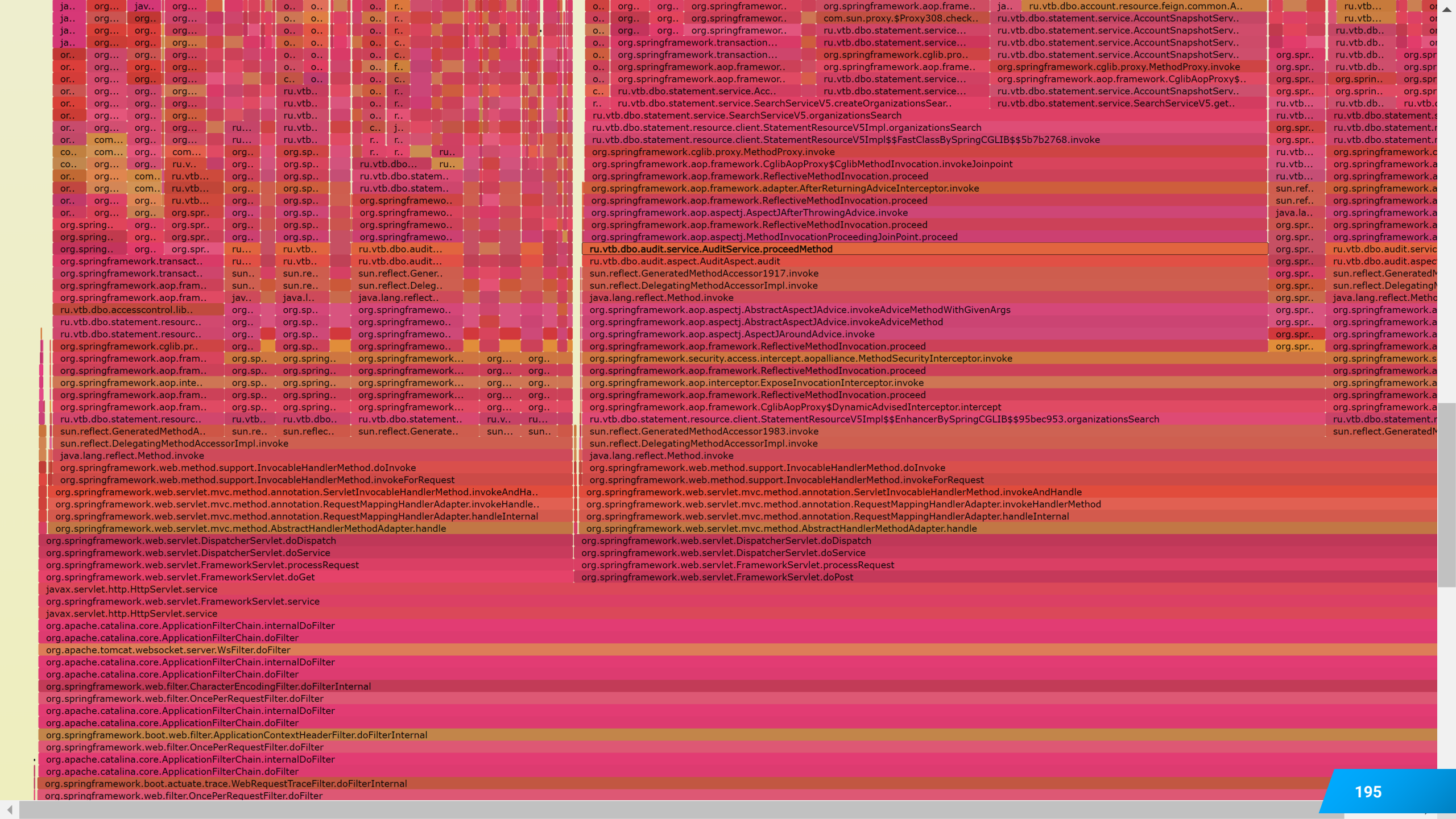
Spaces: 2

193

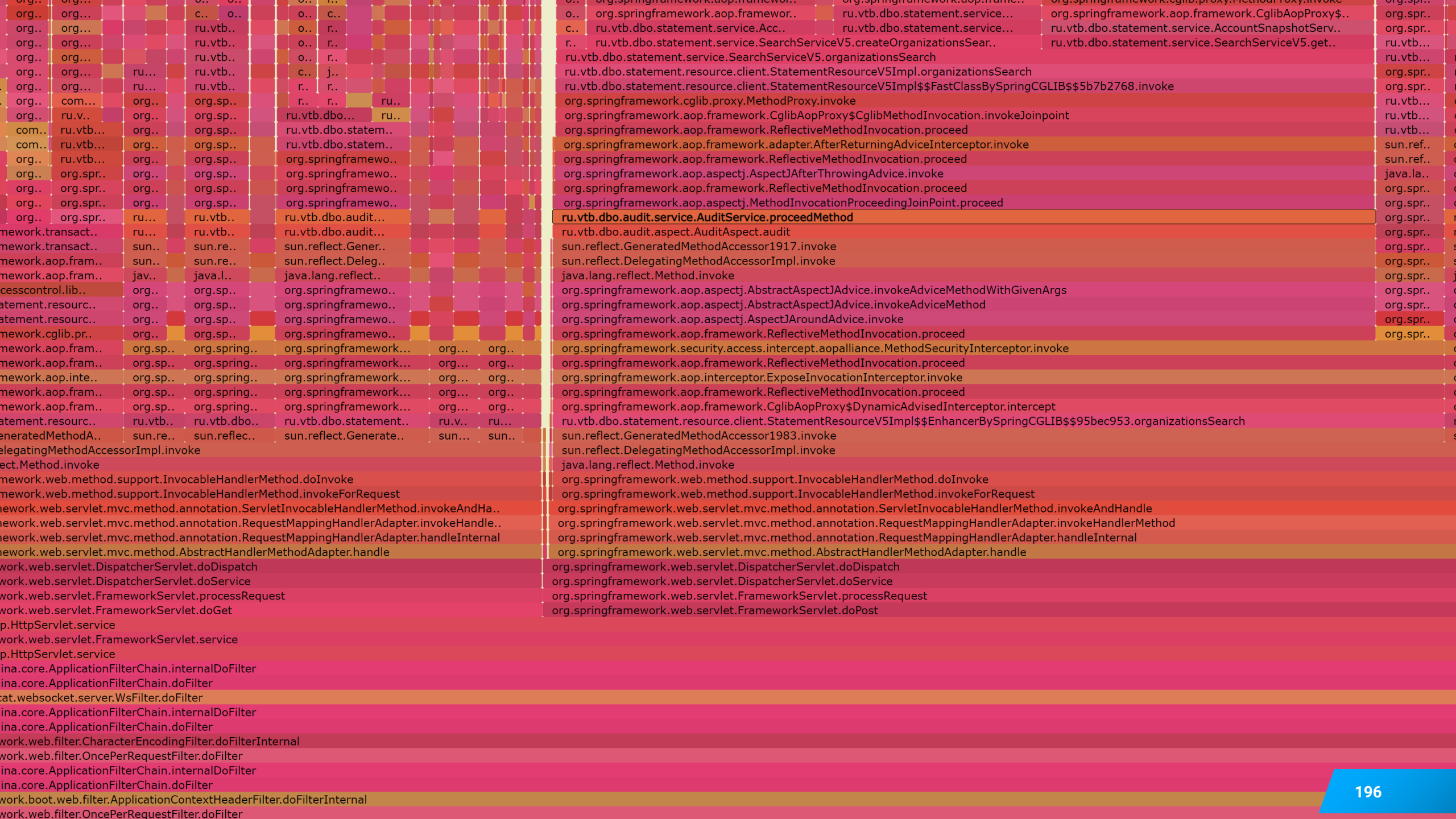
Статистика выполнения отдельного метода

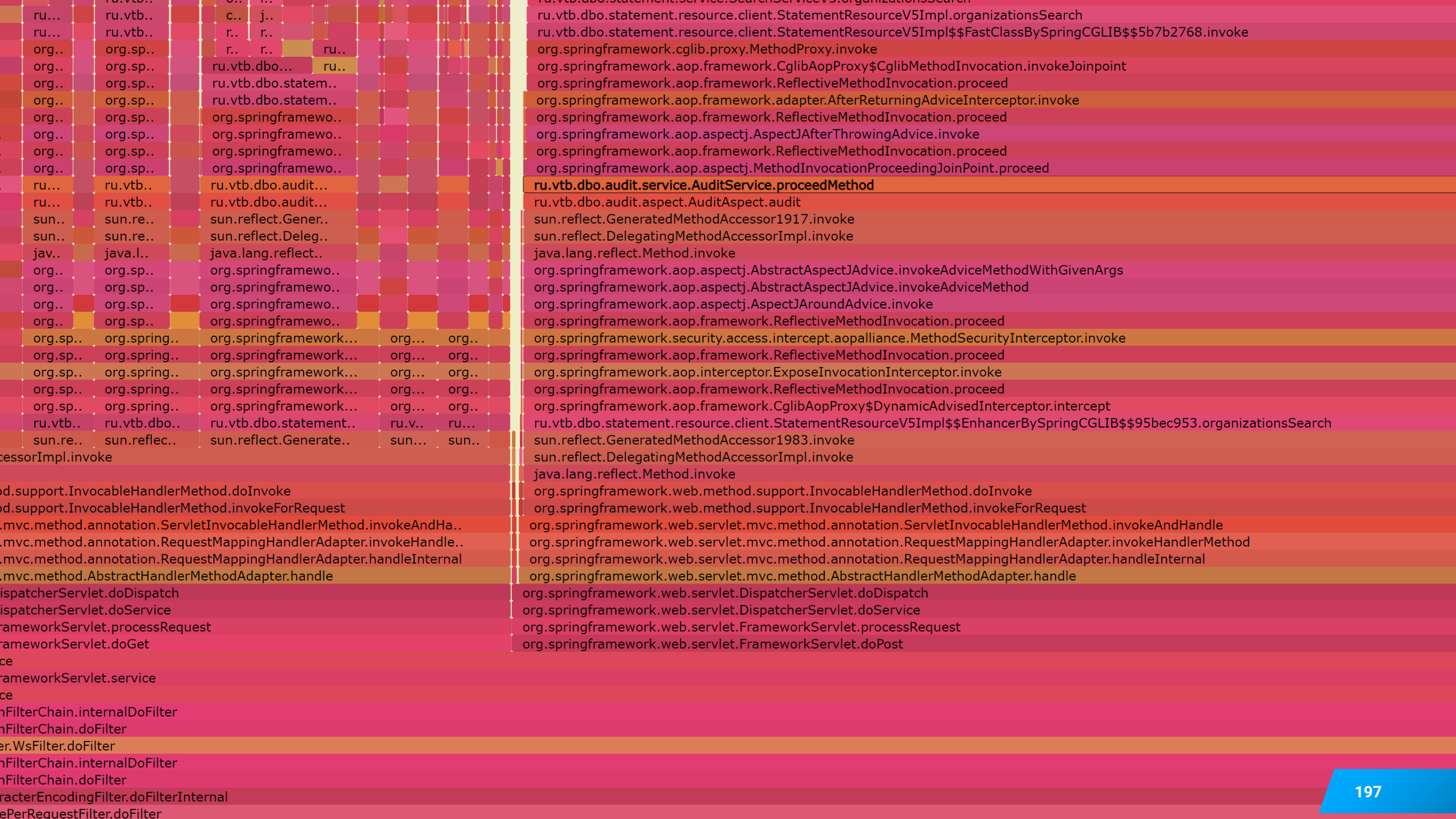
Анализ

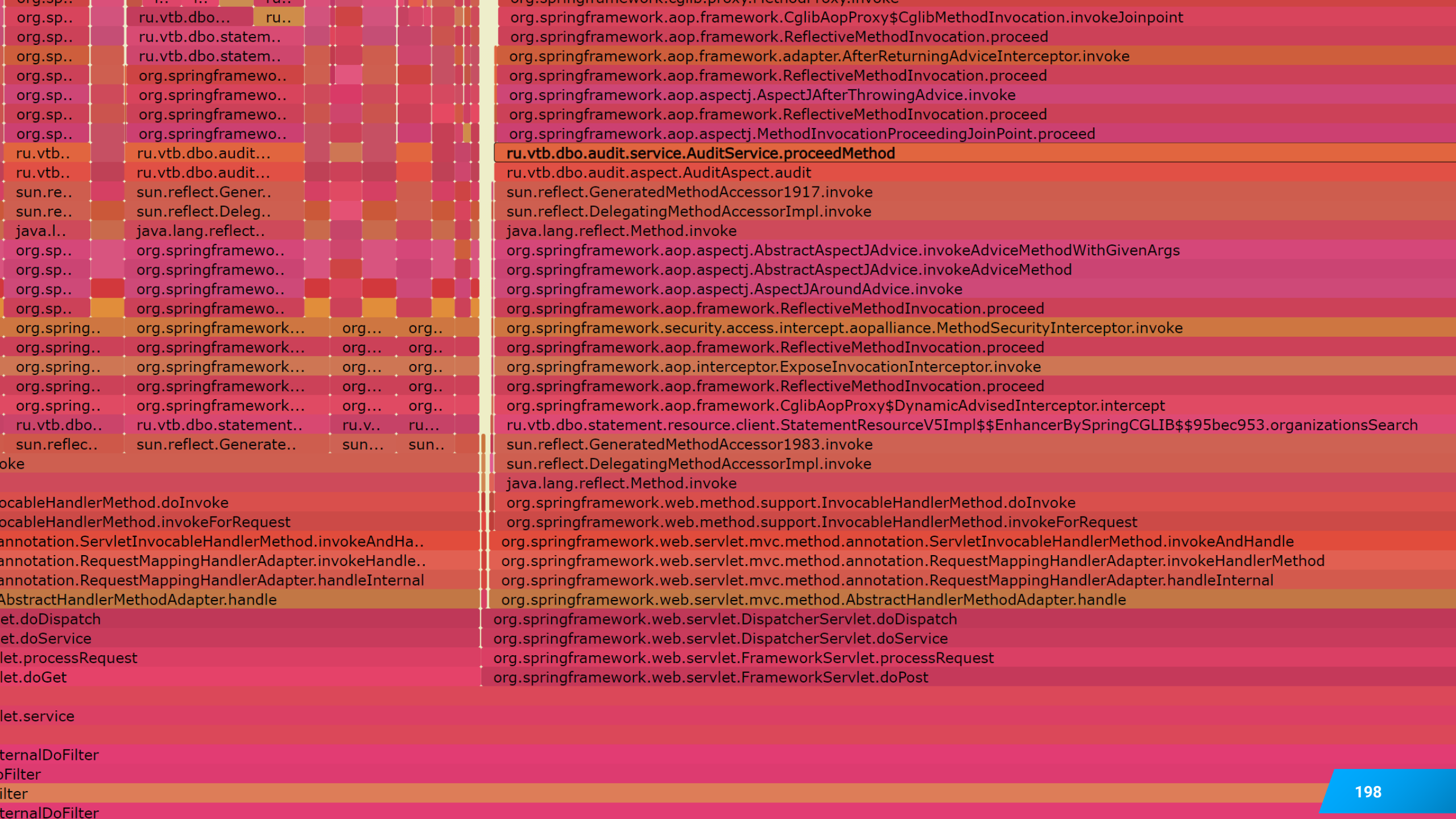



ru.vtb.dbo.audit.service.AuditService.proceedMethod
org.springframework.security.access.intercept.aopalliance.MethodSecurityInterceptor.invoke
org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke
org.springframework.aop.framework.CglibAopProxy$DynamicAdvisedInterceptor.intercept
ru.vtb.dbo.statement.resource.client.StatementResourceV5Impl$$EnhancerBySpringCGLIB$$95bec953.organizationsSearch
sun.reflect.GeneratedMethodAccessor1983.invoke
sun.reflect.DelegatingMethodAccessorImpl.invoke
java.lang.reflect.Method.invoke
org.springframework.web.method.support.InvocableHandlerMethod.doInvoke
org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest
org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal
org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle
org.springframework.web.servlet.DispatcherServlet.doDispatch
org.springframework.web.servlet.DispatcherServlet.doService
org.springframework.web.servlet.FrameworkServlet.processRequest
org.springframework.web.servlet.FrameworkServlet.doPost
javax.servlet.http.HttpServlet.service
org.springframework.web.servlet.FrameworkServlet.service
javax.servlet.http.HttpServlet.service
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter
org.apache.catalina.core.ApplicationFilterChain.doFilter
org.apache.tomcat.websocket.server.WsFilter.doFilter
org.springframework.web.filter.CharacterEncodingFilter.doFilterInternal
org.springframework.web.filter.OncePerRequestFilter.doFilter
org.springframework.boot.web.filter.ApplicationContextHeaderFilter.doFilterInternal
org.springframework.boot.actuate.trace.WebRequestTraceFilter.doFilterInternal

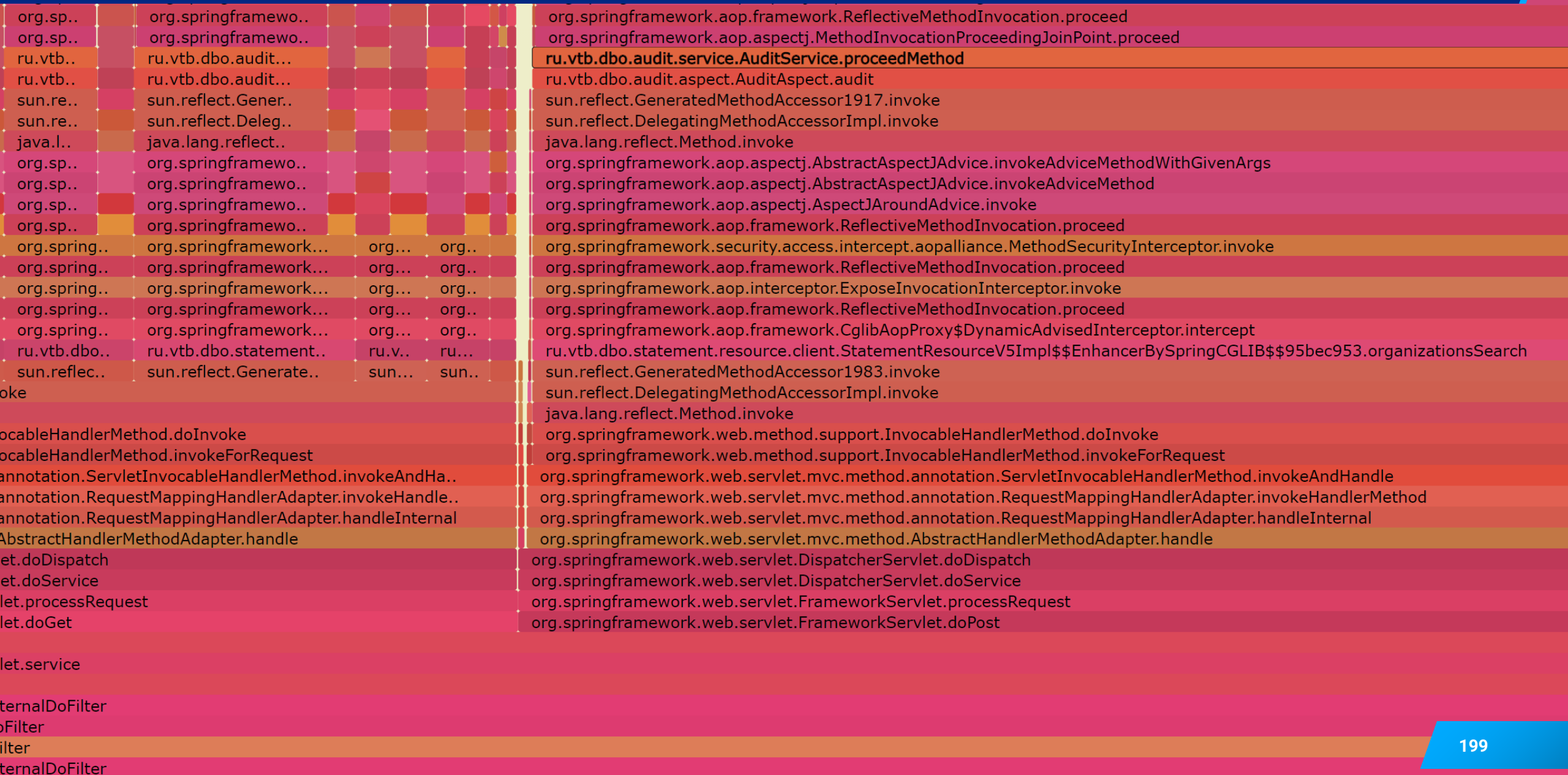
195


ru.vtb.dbo.audit.service.AuditService.proceedMethod
196


ru.vtb.dbo.statement.resource.client.StatementResourceV5Impl.organizationsSearch
ru.vtb.dbo.statement.resource.client.StatementResourceV5Impl$$FastClassBySpringCGLIB$$5b7b2768.invoke
org.springframework.cglib.proxy.MethodProxy.invoke
org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocationInvokeJoinpoint
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.aop.framework.adapter.AfterReturningAdviceInterceptor.invoke
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.aop.aspectj.AspectJAfterThrowingAdvice.invoke
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.aop.aspectj.MethodInvocationProceedingJoinPoint.proceed
ru.vtb.dbo.audit.service.AuditService.proceedMethod
ru.vtb.dbo.audit.aspect.AuditAspect.audit
sun.reflect.GeneratedMethodAccessor1917.invoke
sun.reflect.DelegatingMethodAccessorImpl.invoke
java.lang.reflect.Method.invoke
org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethodWithGivenArgs
org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethod
org.springframework.aop.aspectj.AspectJAroundAdvice.invoke
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.security.access.intercept.aopalliance.MethodSecurityInterceptor.invoke
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.aop.framework.CglibAopProxy$DynamicAdvisedInterceptor.intercept
ru.vtb.dbo.statement.resource.client.StatementResourceV5Impl$$EnhancerBySpringCGLIB$$95bec953.organizationsSearch
sun.reflect.GeneratedMethodAccessor1983.invoke
sun.reflect.DelegatingMethodAccessorImpl.invoke
java.lang.reflect.Method.invoke
org.springframework.web.method.support.InvocableHandlerMethod.doInvoke
org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest
org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal
org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle
org.springframework.web.servlet.DispatcherServlet.doDispatch
org.springframework.web.servlet.DispatcherServlet.doService
org.springframework.web.servlet.FrameworkServlet.processRequest
org.springframework.web.servlet.FrameworkServlet.doPost

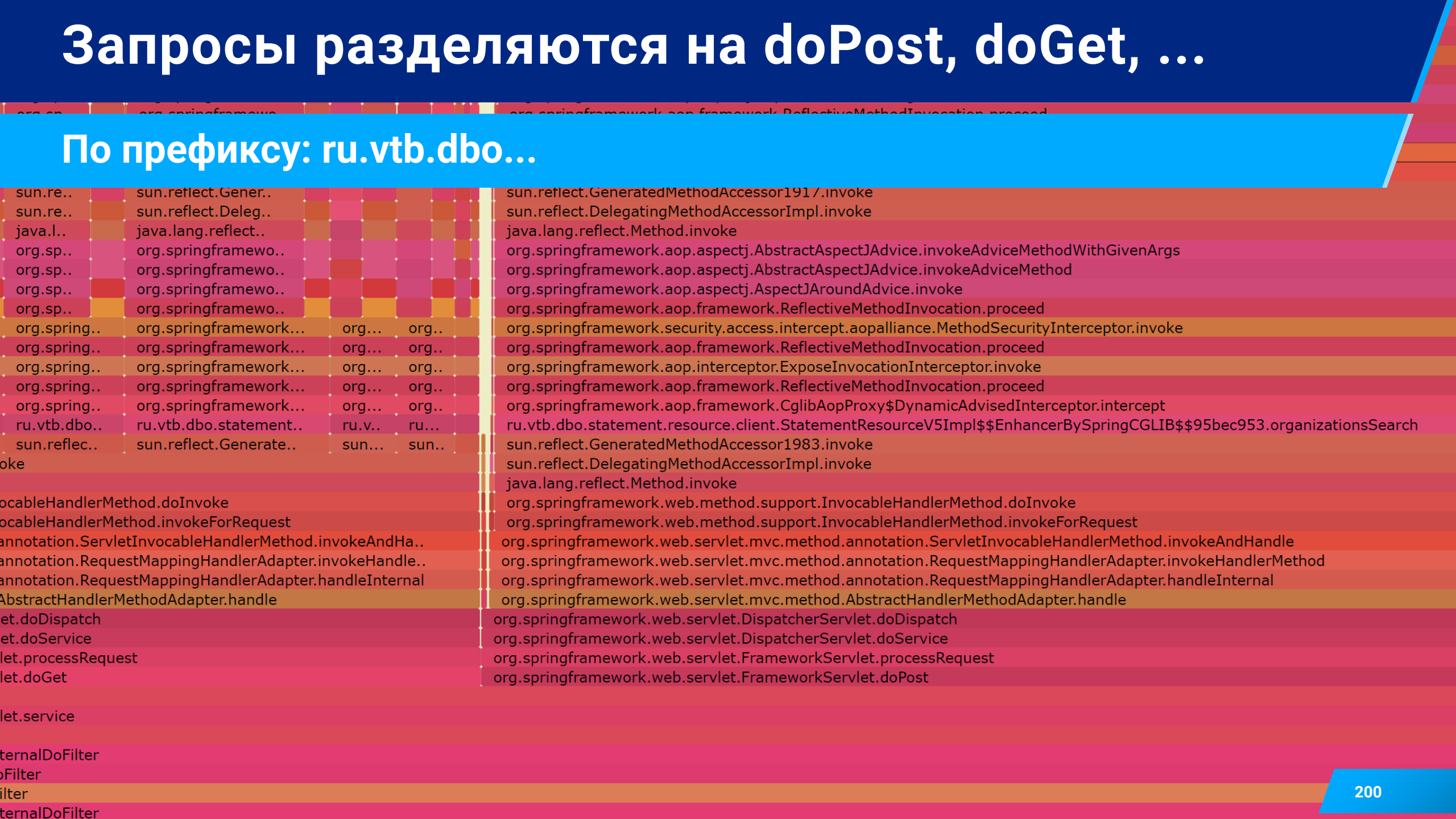

org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invokeJoinpoint
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.aop.framework.adapter.AfterReturningAdviceInterceptor.invoke
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.aop.aspectj.AspectJAfterThrowingAdvice.invoke
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.aop.aspectj.MethodInvocationProceedingJoinPoint.proceed
ru.vtb.dbo.audit.service.AuditService.proceedMethod
ru.vtb.dbo.audit.aspect.AuditAspect.audit
sun.reflect.GeneratedMethodAccessor1917.invoke
sun.reflect.DelegatingMethodAccessorImpl.invoke
java.lang.reflect.Method.invoke
org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethodWithGivenArgs
org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethod
org.springframework.aop.aspectj.AspectJAroundAdvice.invoke
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.security.access.intercept.aopalliance.MethodSecurityInterceptor.invoke
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
org.springframework.aop.framework.CglibAopProxy$DynamicAdvisedInterceptor.intercept
ru.vtb.dbo.statement.resource.client.StatementResourceV5Impl$$EnhancerBySpringCGLIB$$95bec953.organizationsSearch
sun.reflect.GeneratedMethodAccessor1983.invoke
sun.reflect.DelegatingMethodAccessorImpl.invoke
java.lang.reflect.Method.invoke
org.springframework.web.method.support.InvocableHandlerMethod.doInvoke
org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest
org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal
org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle
org.springframework.web.servlet.DispatcherServlet.doDispatch
org.springframework.web.servlet.DispatcherServlet.doService
org.springframework.web.servlet.FrameworkServlet.processRequest
org.springframework.web.servlet.FrameworkServlet.doPost

Запросы разделяются на doPost, doGet, ...



Запросы разделяются на doPost, doGet, ...

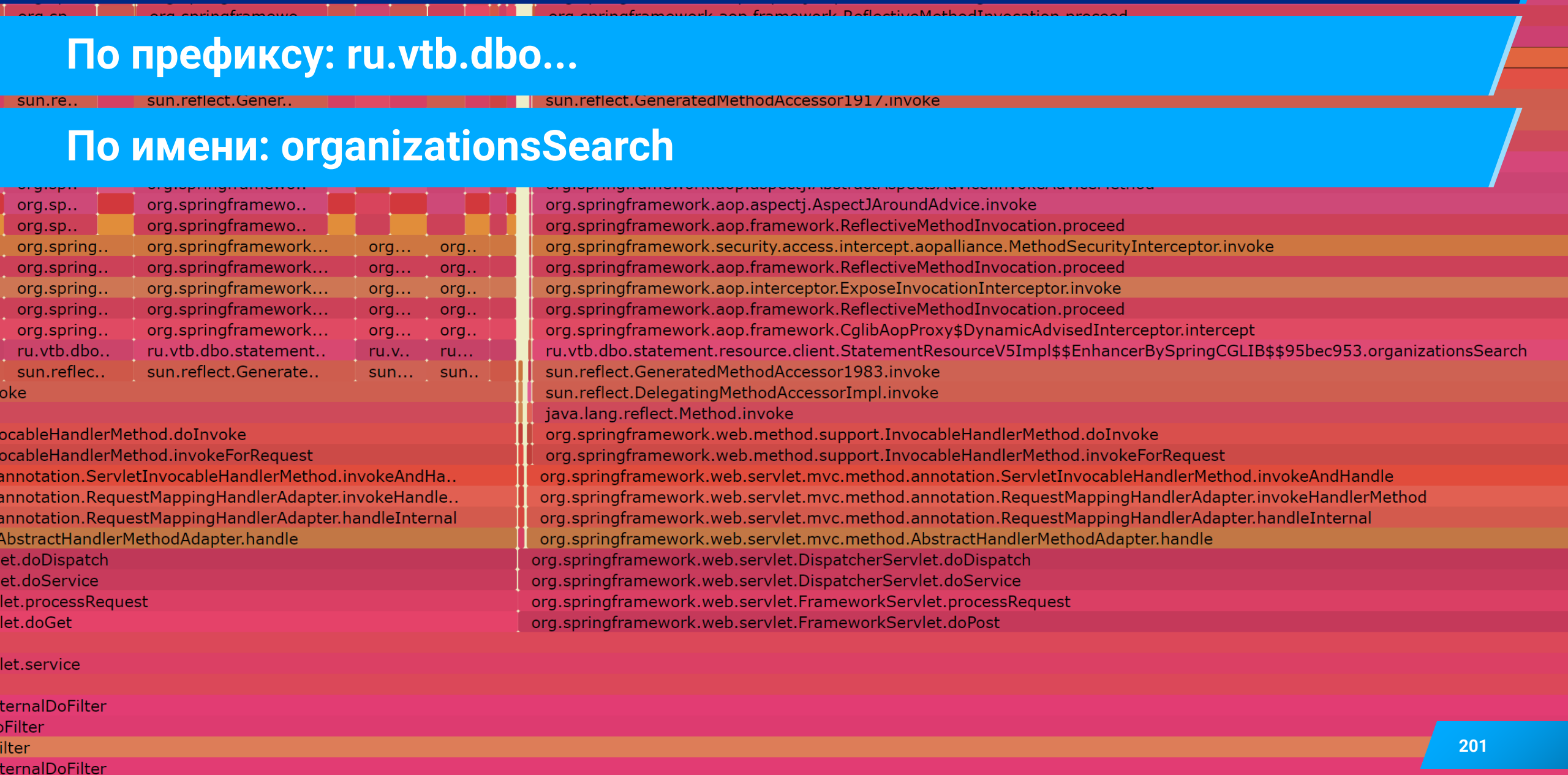
По префиксу: ru.vtb.dbo...



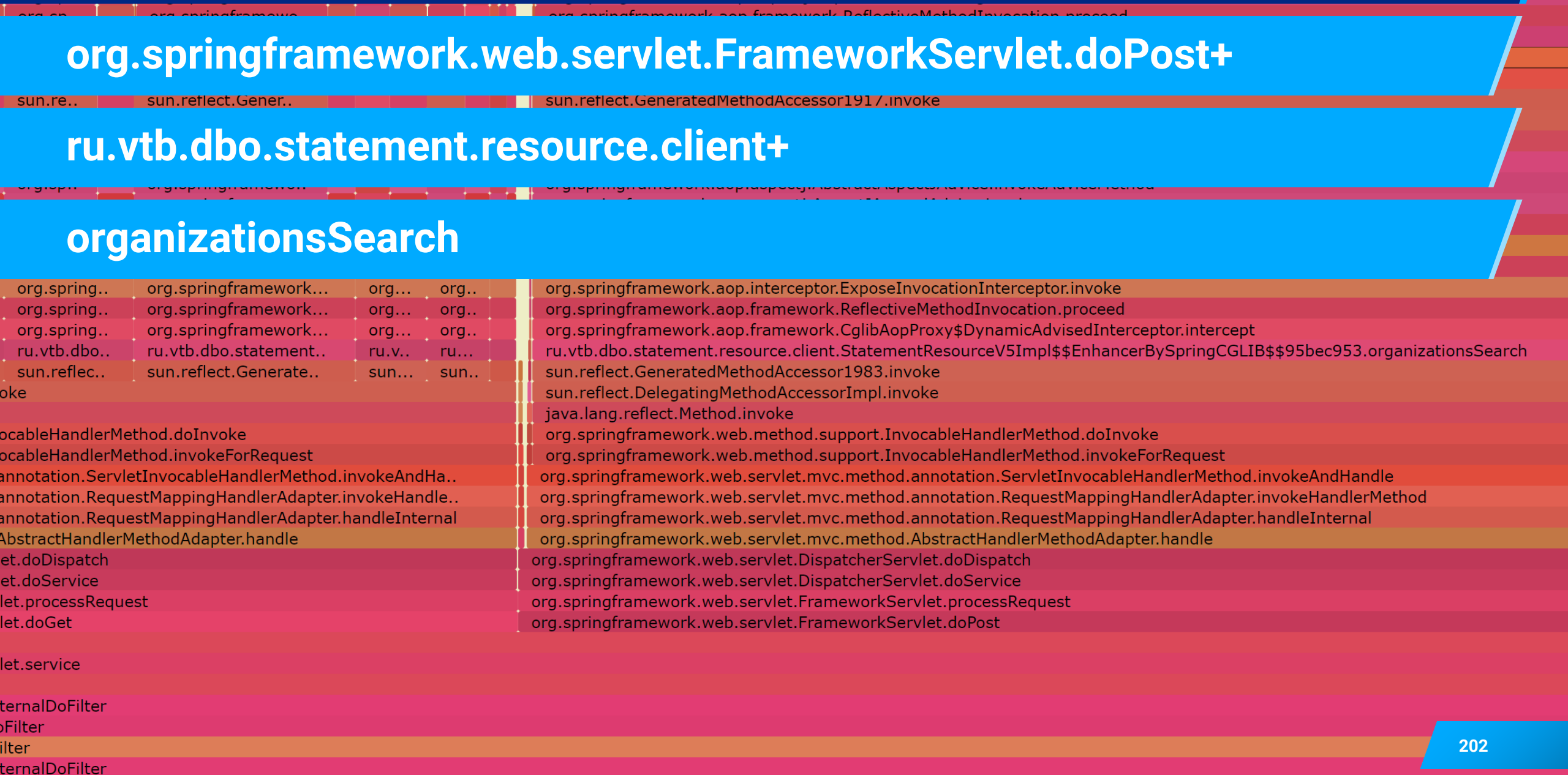
Запросы разделяются на doPost, doGet, ...

По префиксу: ru.vtb.dbo...

По имени: organizationsSearch



В SJK фильтры методов можно объединять по +



rescan.stmt.sh

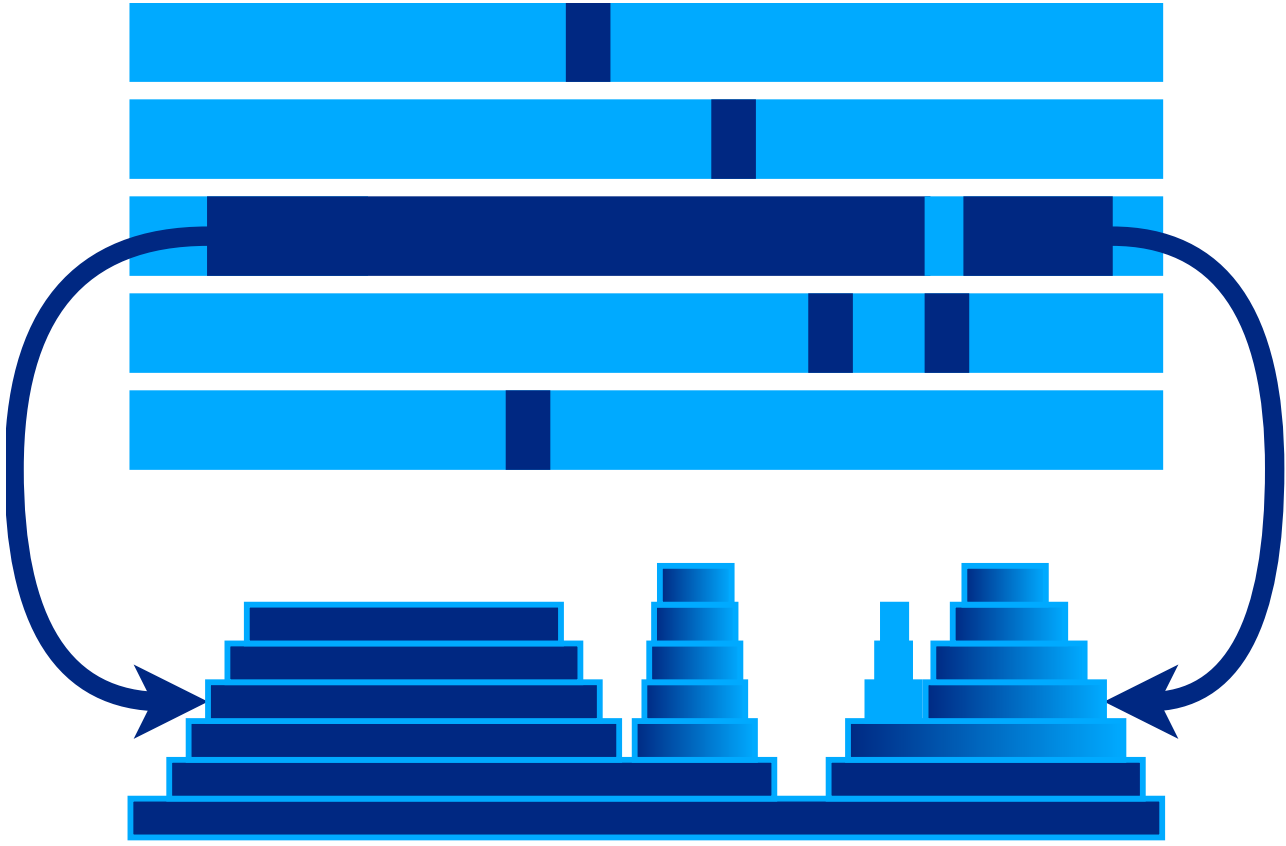
```
16  ["pool-1-thread"]='pool-1-thread.+' \
17  ["pool-10-thread"]='pool-10-thread.+' \
18  ["HikariPool-{N} connection adder"]='HikariPool-.+ connection adder' \
19  ["kafka-producer-network-thread"]='kafka-producer-network-thread.+' \
20  ["http-nio-{N}-ClientPoller-{N}"]='http-nio-.+-ClientPoller.+' \
21  ["statement-STMT_OVERDUE_PAYMENT"]='statement-STMT_OVERDUE_PAYMENT.+' \
22  )
23
24  methods=(\
25  ["statement v5 ui statements organizations search (POST)"]='ru.vtb.dbo.statement.resource.clien
26  ["statement v5 ui statements organizations {organizationId} accounts {accountId} dates search (
27  ["statement v5 ui statements actions (GET)"]='ru.vtb.dbo.statement.resource.client.StatementRes
28  ["statement v5 ui statements {statementId} operations {operationId} document (GET)"]='ru.vtb.db
29  ["statement v5 ui statements {statementId} operations (GET)"]='ru.vtb.dbo.statement.resource.cl
30  ["statement v5 ui statements {statementId} (GET)"]='ru.vtb.dbo.statement.resource.client.Statem
31  )
32
33  for thread in "${!threads[@]}"
34  do
35      echo "${thread} - ${threads[$thread]}"
36  done
37
38  echo "================================"
39
40  for method in "${!methods[@]}"
```

Выполнять фильтрацию по методам

```
thread="http-nio-8080-exec"
for method in "${!methods[@]}"
do
    $sjk \
    stcpy \
    --input "${reportFolder}/${thread}.sdt"  \
    --trace-filter "${methods[$method]}" \
    --output "${reportFolder}/${thread}.${method}.sdt"
    mkdir "${reportFolder}/${thread}/${method}"

    $sjk \
    ssa \
    --file "${reportFolder}/${thread}.${method}.sdt" \
    --histo \
```

Ожидание ответа от SQL-сервера

Анализ



С дополнительной фильтрацией

От Postgre SQL Server, например

```
$sjk \
ssa \
--file "${reportFolder}/${thread}.${method}.sdt" \
--trace-filter "org.postgresql.jdbc" \
--histo \
> "${reportFolder}/${thread}/${method}/${method}.jdbc.only.histo.txt"
```

С дополнительной фильтрацией

Без Postgre SQL Server, например

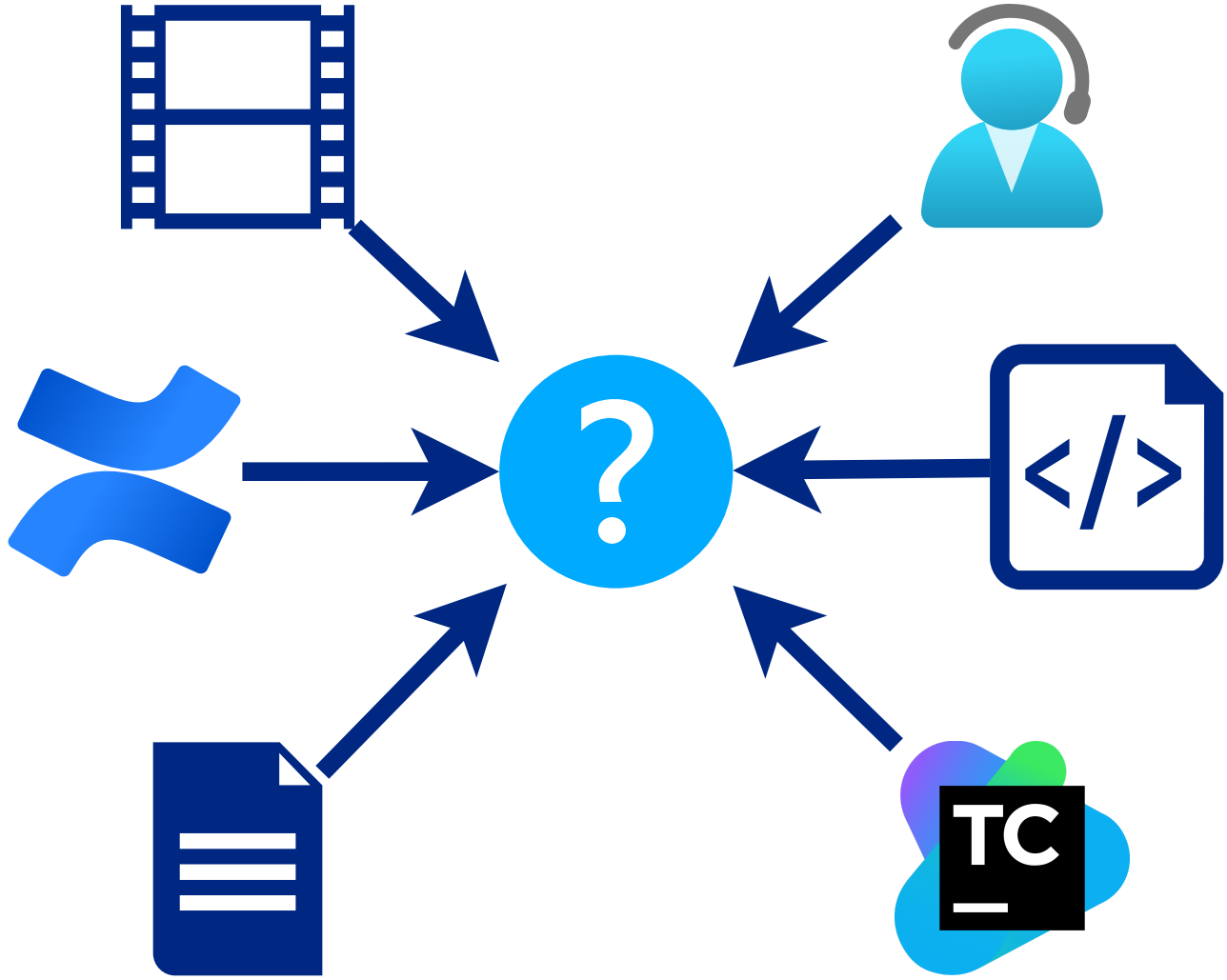
```
$sjk \
ssa \
--file "${reportFolder}/${thread}.${method}.sdt" \
--trace-filter "!org.postgresql.jdbc" \
--histo \
> "${reportFolder}/${thread}/${method}/${method}.wo.jdbc.histo.txt"
```

Автоматизация формирования отчета

```
17  ["pool-10-thread"]="pool-10-thread.+" \
18  ["HikariPool-{N} connection adder"]='HikariPool-.+ connection adder' \
19  ["kafka-producer-network-thread"]='kafka-producer-network-thread.+' \
20  ["http-nio-{N}-ClientPoller-{N}"]='http-nio-.+-ClientPoller.+' \
21  ["statement-STMT_OVERDUE_PAYMENT"]='statement-STMT_OVERDUE_PAYMENT.+' \
22  )
23
24  methods=(\
25  ["statement v5 ui statements organizations search (POST)"]='ru.vtb.dbo.statement.resource.clien
26  ["statement v5 ui statements organizations {organizationId} accounts {accountId} dates search (
27  ["statement v5 ui statements actions (GET)"]='ru.vtb.dbo.statement.resource.client.StatementRes
28  ["statement v5 ui statements {statementId} operations {operationId} document (GET)"]='ru.vtb.db
29  ["statement v5 ui statements {statementId} operations (GET)"]='ru.vtb.dbo.statement.resource.cl
30  ["statement v5 ui statements {statementId} (GET)"]='ru.vtb.dbo.statement.resource.client.Statem
31  )
32
33  for thread in "${!threads[@]}"
34  do
35      echo "${thread} - ${threads[$thread]}"
36  done
37
38  echo "================================"
39
40  for method in "${!methods[@]}"
```

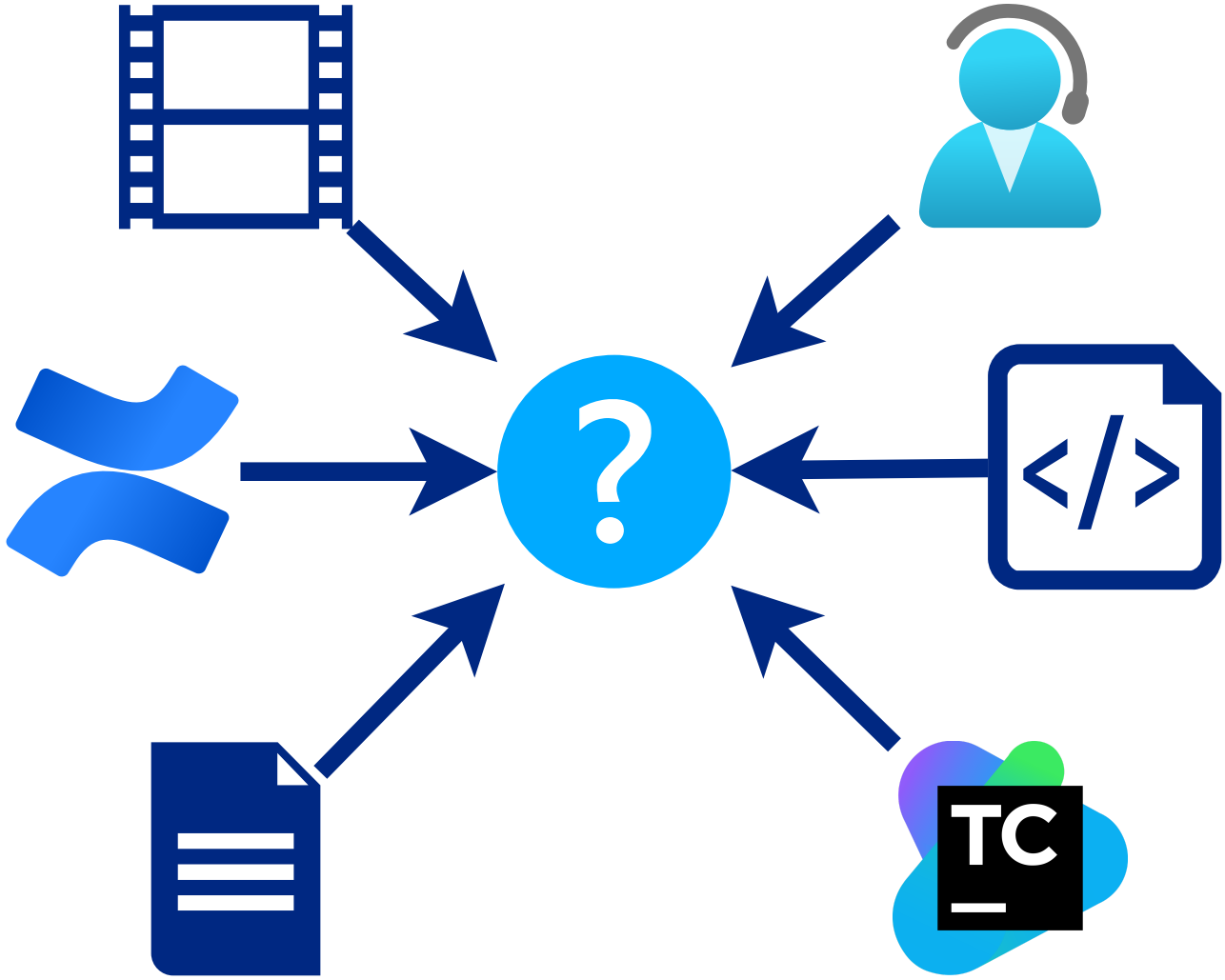
Масштабирование профилирования на всю команду

Масштабирование



Пишем подробный отчет по профилированию

Масштабирование



Export to CSV

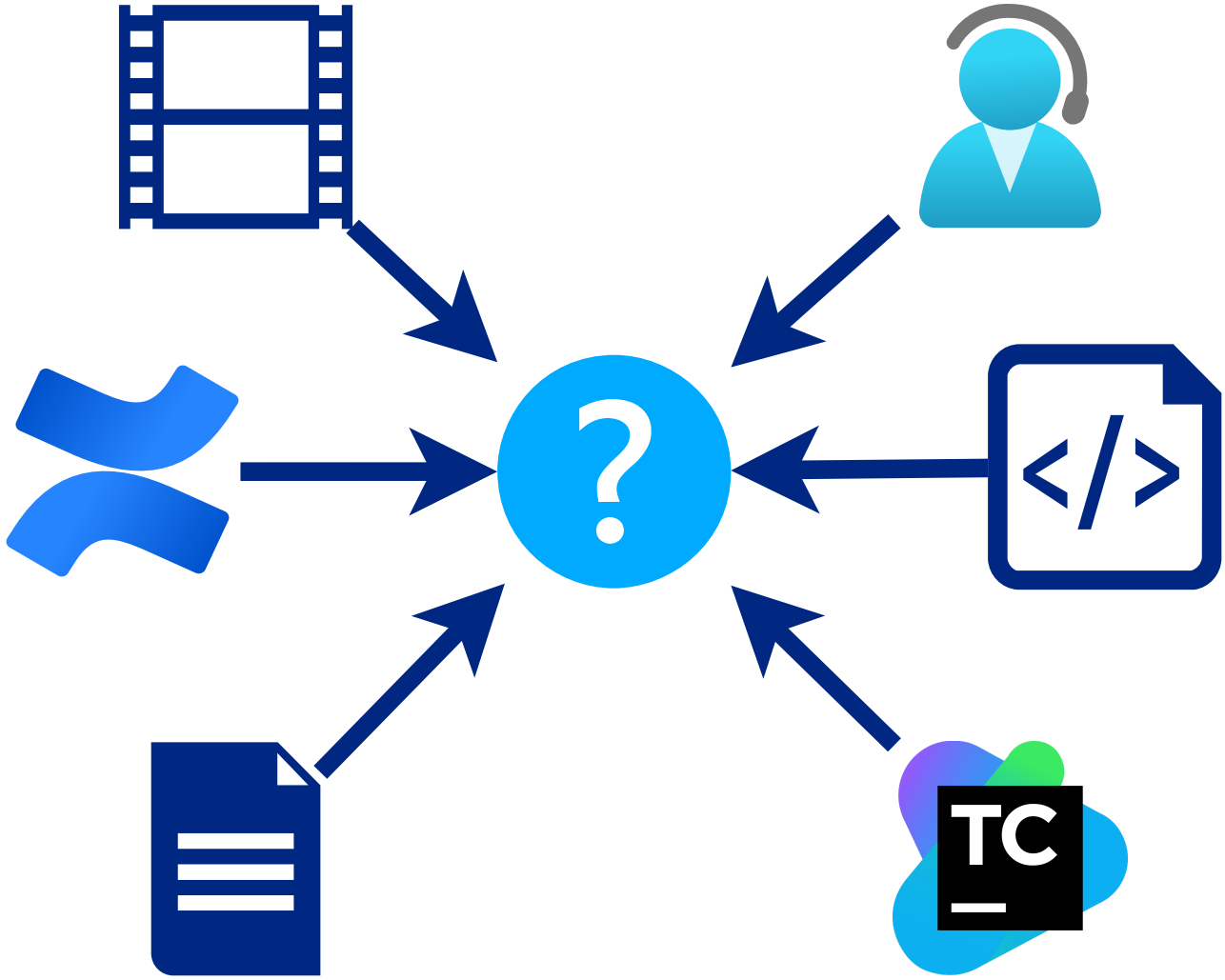
Дата	2020-03-18
Цель теста	Профилирование migration-core, migration-dbo317, rupayment
Профиль нагрузки	Предмиграция трех организаций, 1700 документов у каждой.
Система	migration-core 1.9.0.9, migration-dbo317 1.9.0.10, rupayment 1.22.1.2
Стенд	Load-1
Задача	VTBDBOMIGR-7571 - [migration-core] В методе BatchSav создания документов) отказ от получения и сохранения Pr документов на 25% ST VTBDBOPPKO-28625 - [rupayment] В клас /migration/rupayments/batch) если не делат вместо RuPaymentDto, то создание РПП уск CREATED VTBDBOPPKO-28640 - [rupayment] В классе RuPay /migration/rupayments/batch не создавать запись ис предмиграции CREATED
Тип теста	Профилирование

Содержание

- Содержание
- Задача
- Результаты
- Параметры
  - Количество экземпляров сервисов
  - Выделенные ресурсы
  - Настройки ConfigMap
- migration-core
  - Потоки
  - Поток SAVE_JOBS_POOL_thread-11
    - 29,5% - BatchSaveService.createContext - fromXML - подготовка запроса
    - 45,2% - AbstractSimpleBatchIntegrator.save - Ожидание ответа от rupayment
    - 25,3% - BatchSaveService.done - toXML - обработка ответа на запрос
- migration-dbo317
- 45,2% - rupayment
  - Потоки
  - 11% - DeltaHistoryBuilderService.addHistoryEntryBeforeCommit - Создание истории при создании докуме
  - 9,7% -

Пишем инструкцию по профилированию в ручном режиме

Масштабирование



ДЕРЕВО СТРАНИЦ

- Функциональное тестирование
- Автоматизированное тестирование
- Нагрузочное тестирование
  - Автоматизация в НТ
  - Архитектура и Ресурсы стендов НТ и Пром
  - База знаний
    - 0. Подготовка рабочего места
    - 1. Методика, профиль нагрузки, требования
    - 2. Настройка стенда
    - 3. Изучение работы с системой
    - 4. Разработка тестов и заглушек
    - 5. Генерация тестовых данных
    - 6. Запуск тестов
    - 7. Анализ метрик и логов
    - 8. Оформление отчёта и инструкций

Инструменты для пространства

операционная система используется сервисом надо или посмотреть на исходный Docker-файл сервиса или просто открыть консоль **Exec** в поде сервиса.

Если команда

cat /etc/os-release

```
1  cat /etc/os-release
```

покажет **Alpine** Linux, то в пути подключения надо будет использовать **linux_musl-x64**

если покажет **CentOS** Linux, то в пути подключения надо будет использовать просто **linux-x64**

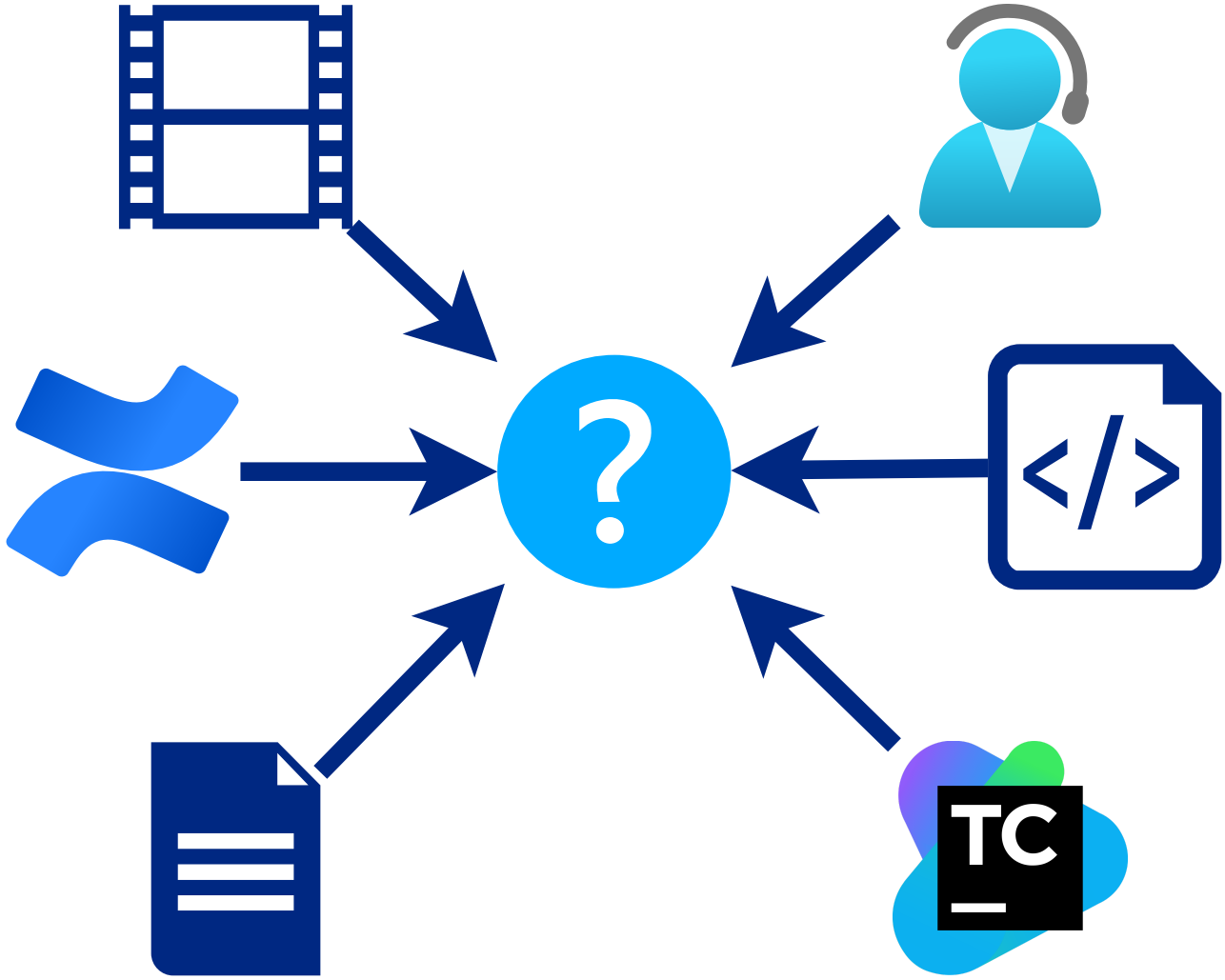
cat /etc/os-release для Alpine Linux › Развернуть исходный код

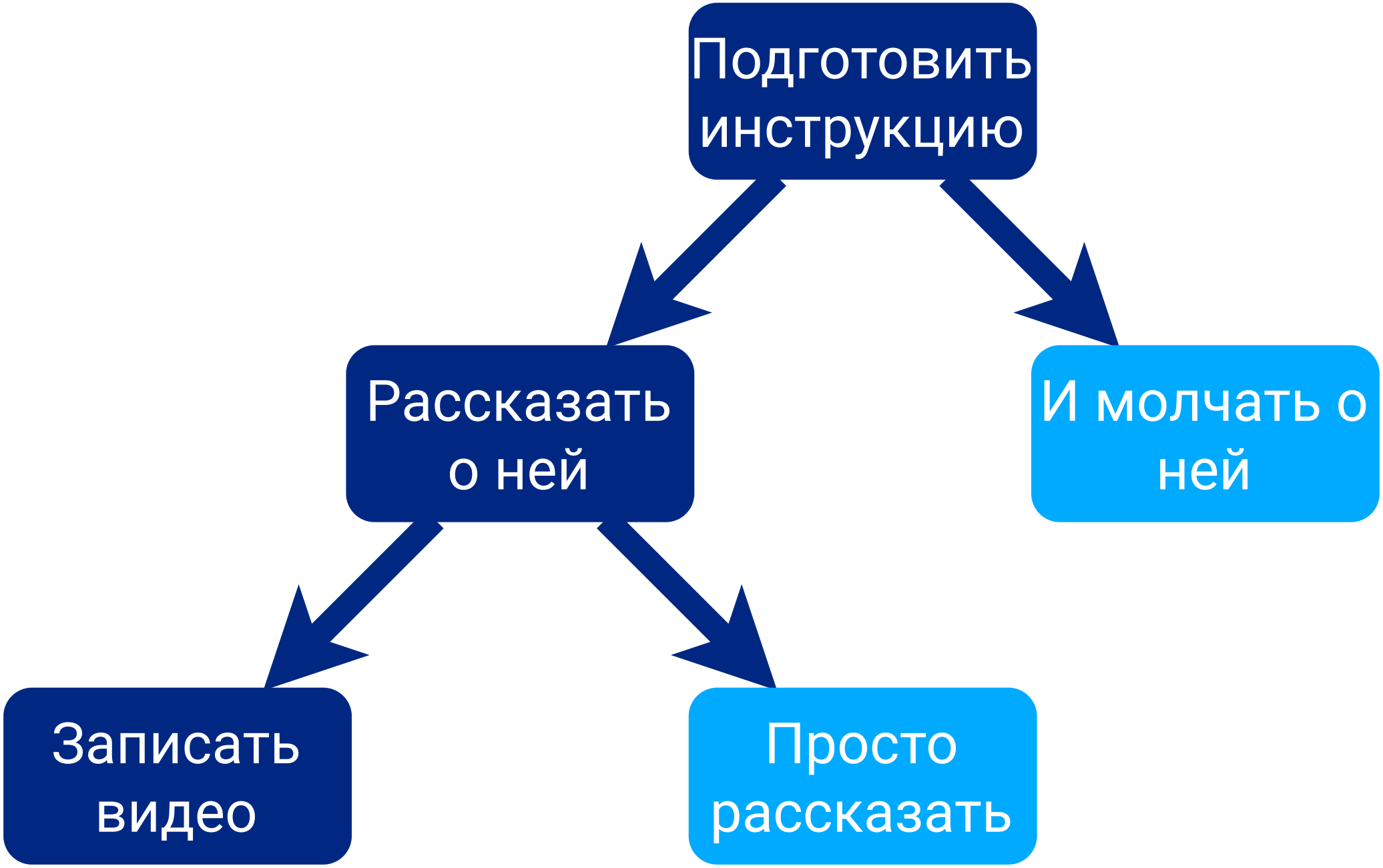
cat /etc/os-release для CentOS Linux › Развернуть исходный код

Настройка сервиса

Записываем видео с демонстрацией профилирования

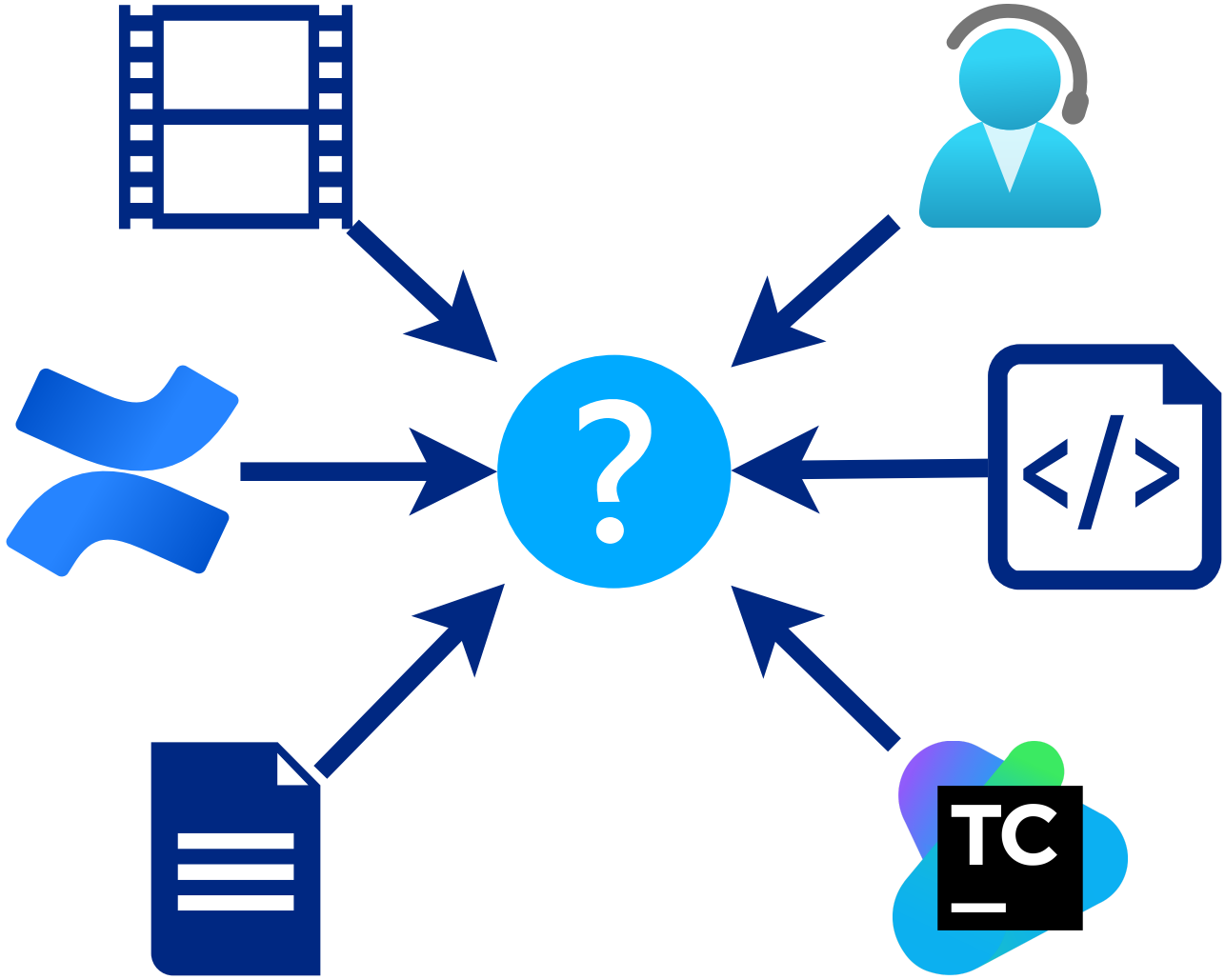
Масштабирование

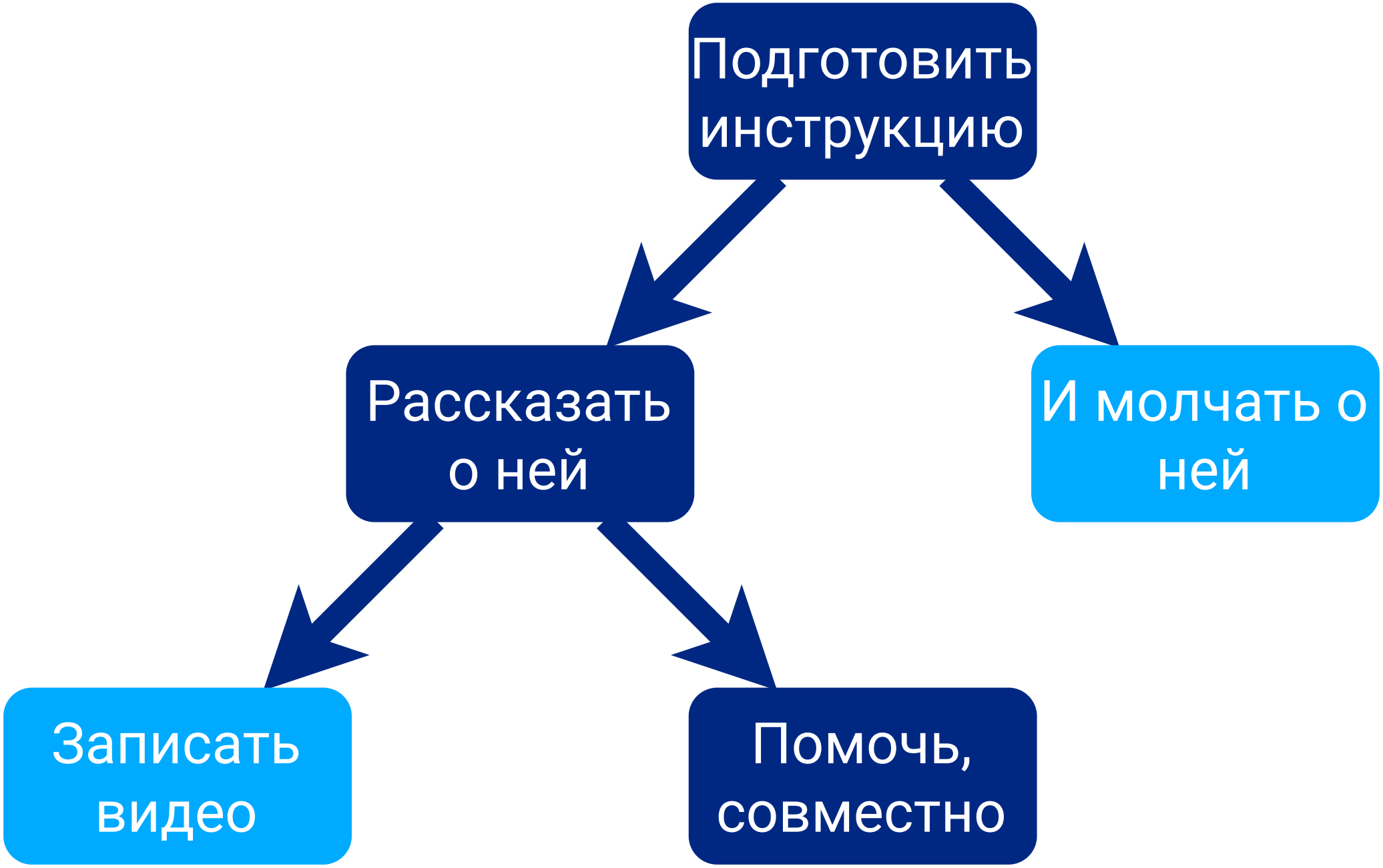



Подготовить инструкцию
Рассказать о ней
И молчать о ней
Записать видео
Просто рассказать

Парное профилирование и анализ результатов

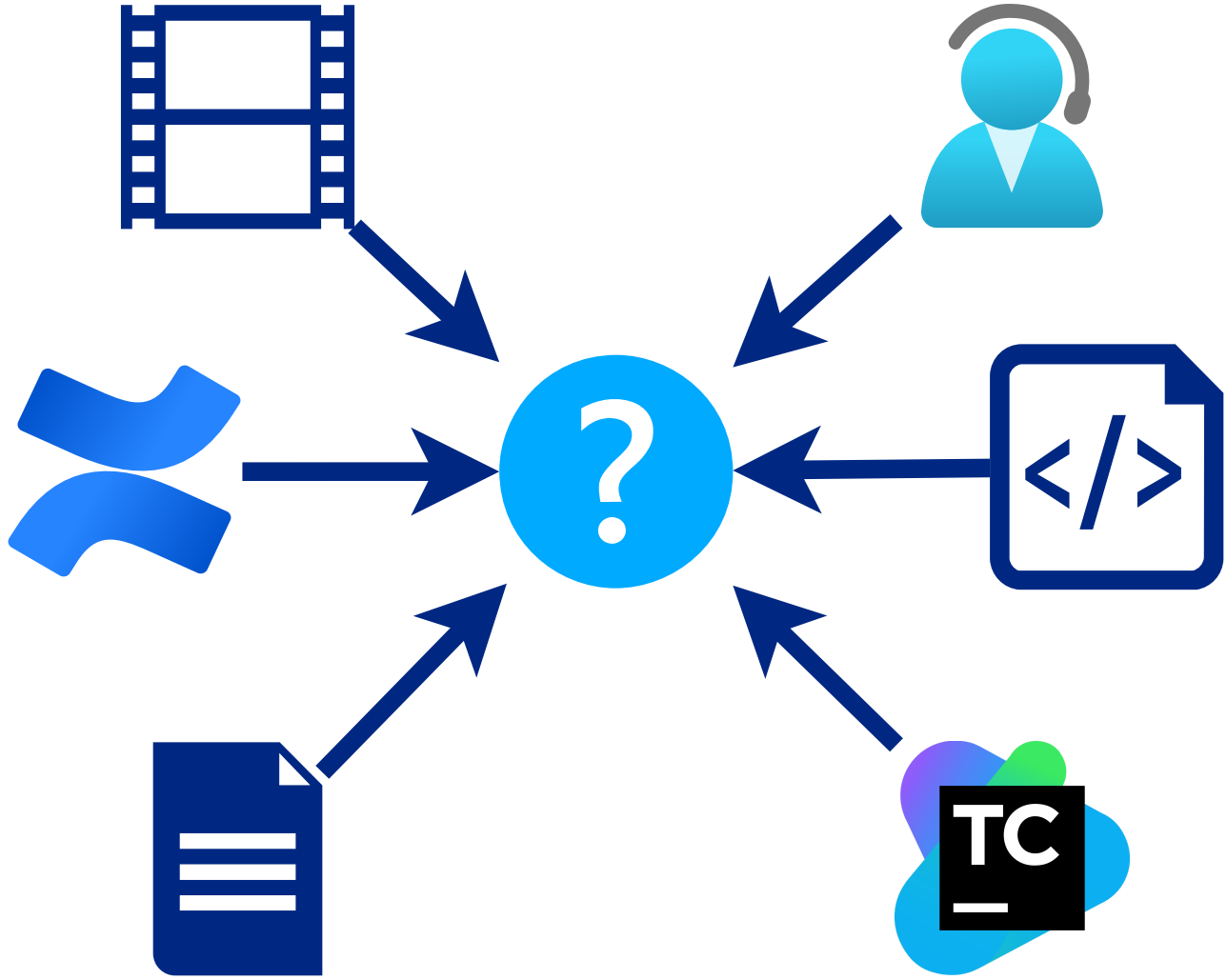
Масштабирование




Подготовить инструкцию
Рассказать о ней
И молчать о ней
Записать видео
Помочь, совместно

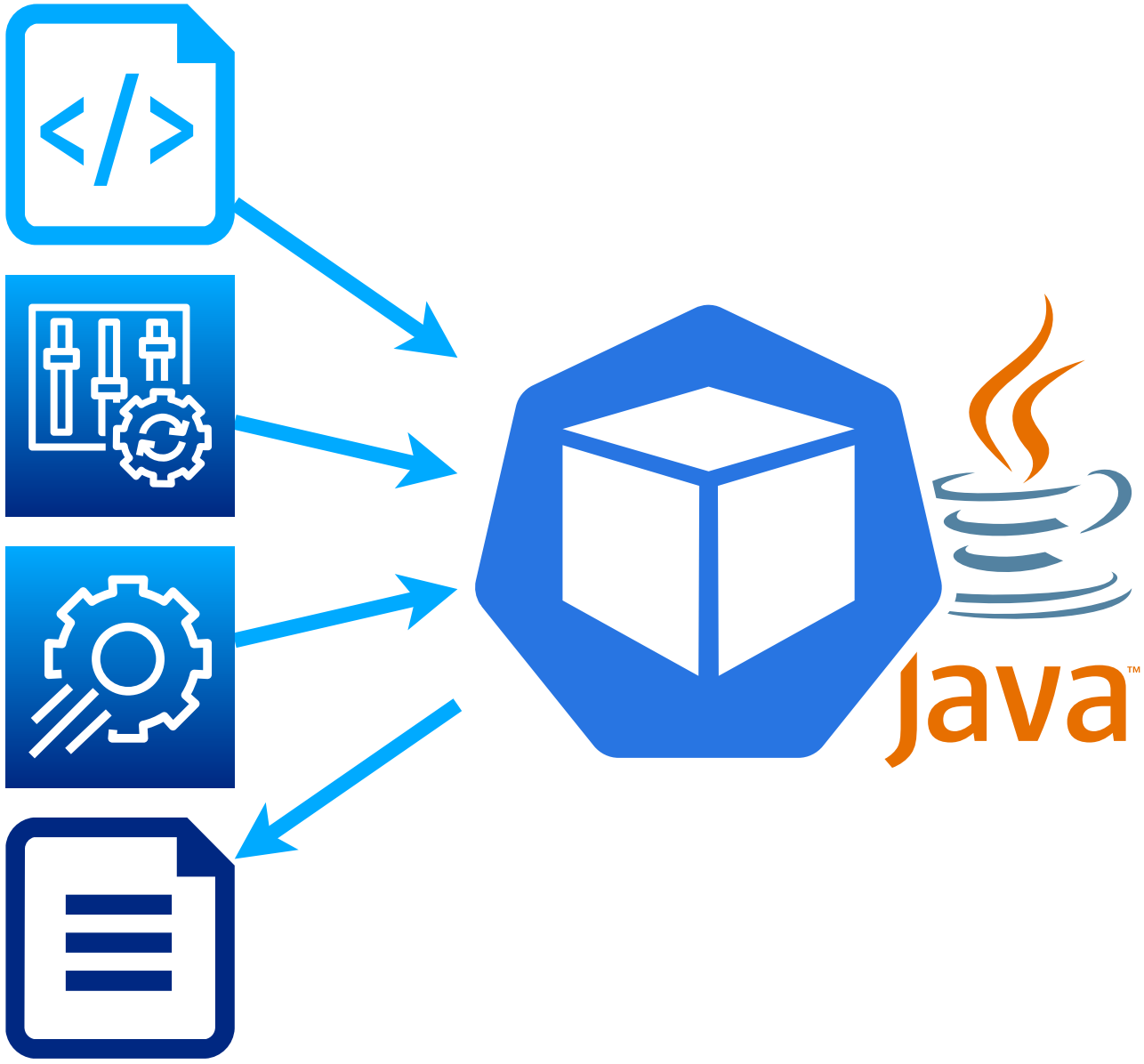
Пишем скрипты автоматизации профилирования

Масштабирование

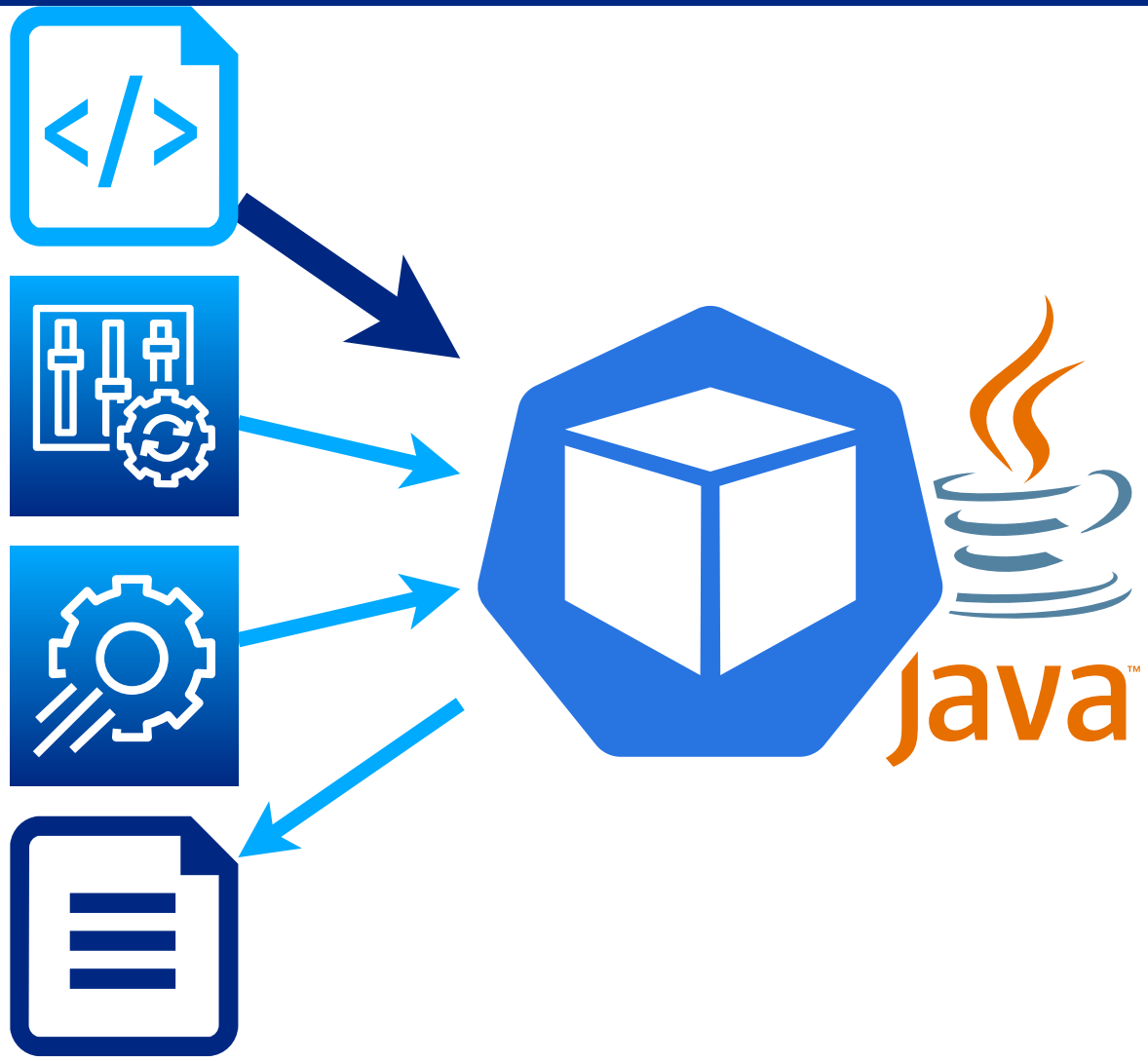


Пишем скрипты автоматизации профилирования

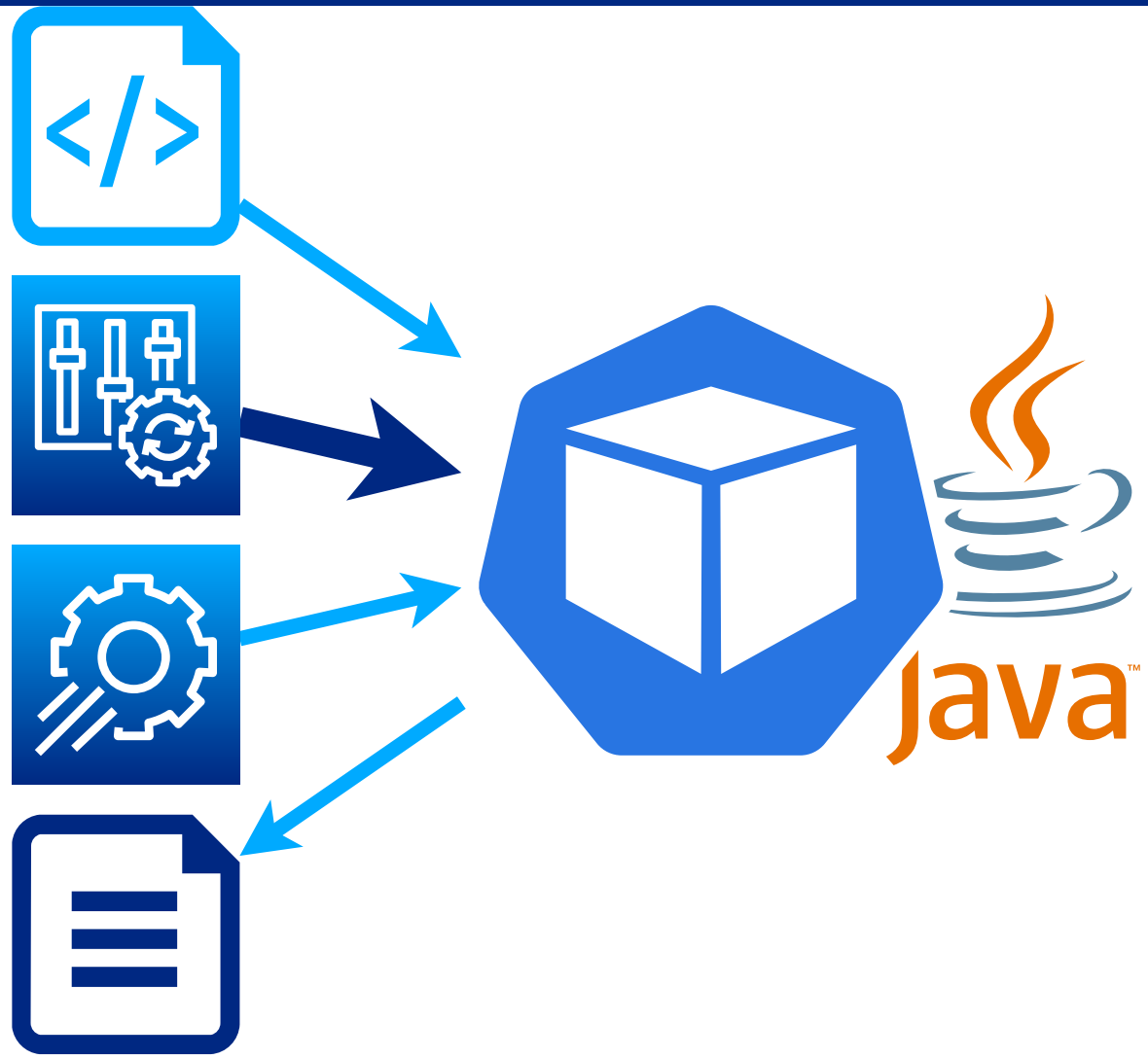
Масштабирование



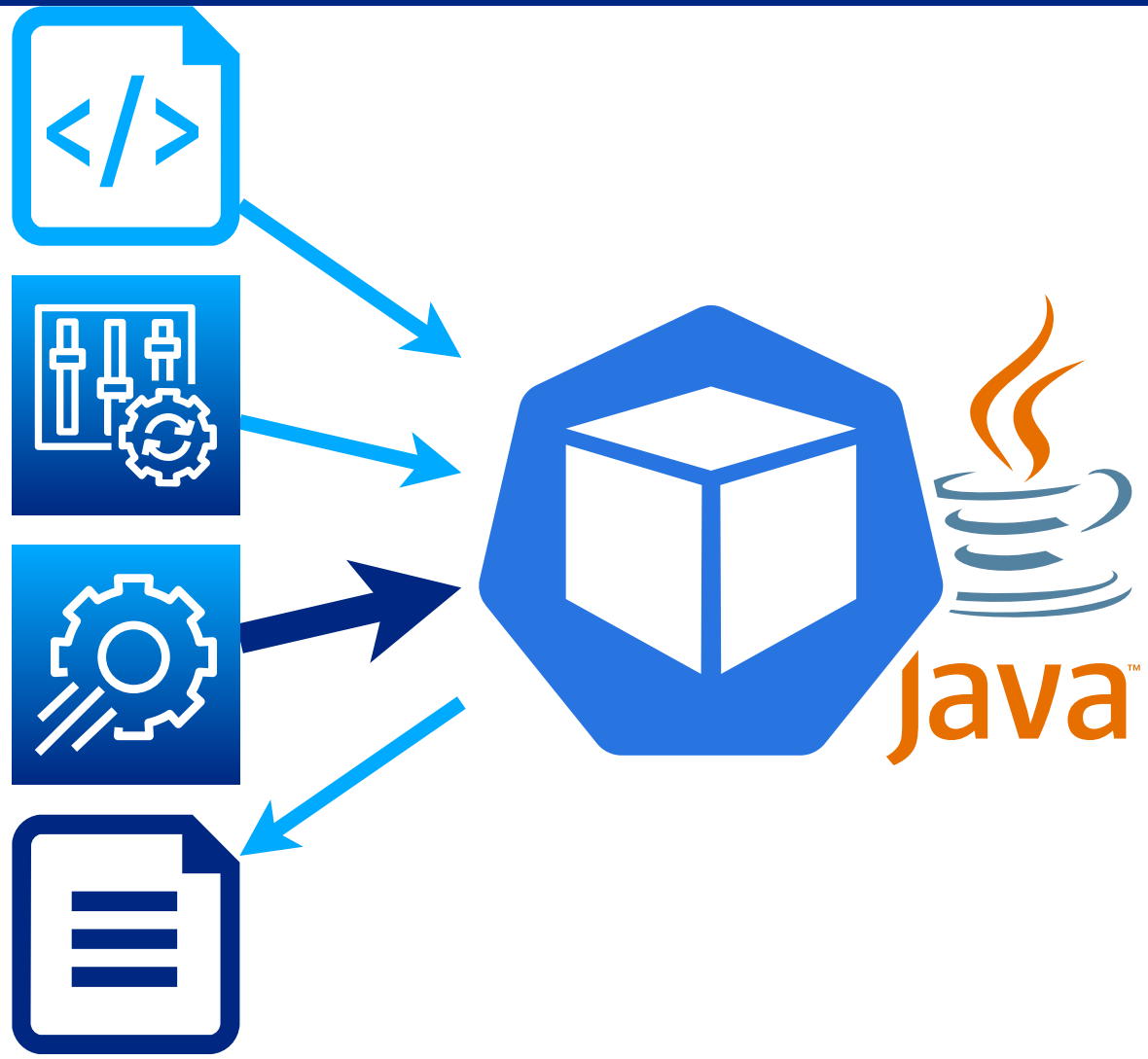
Примонтировать/загрузить профайлер



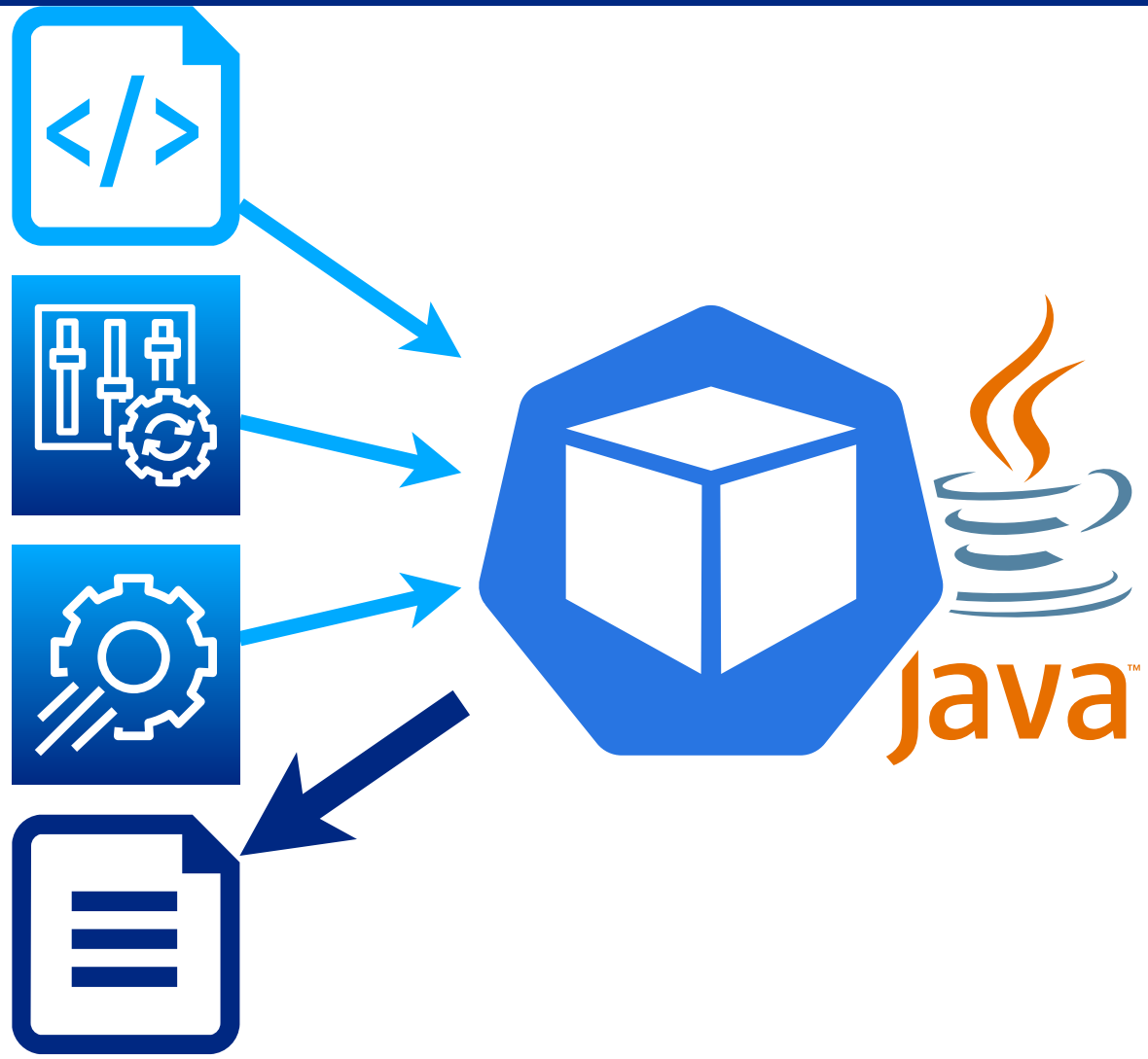
Настроить JAVA_OPTIONS на профилирование



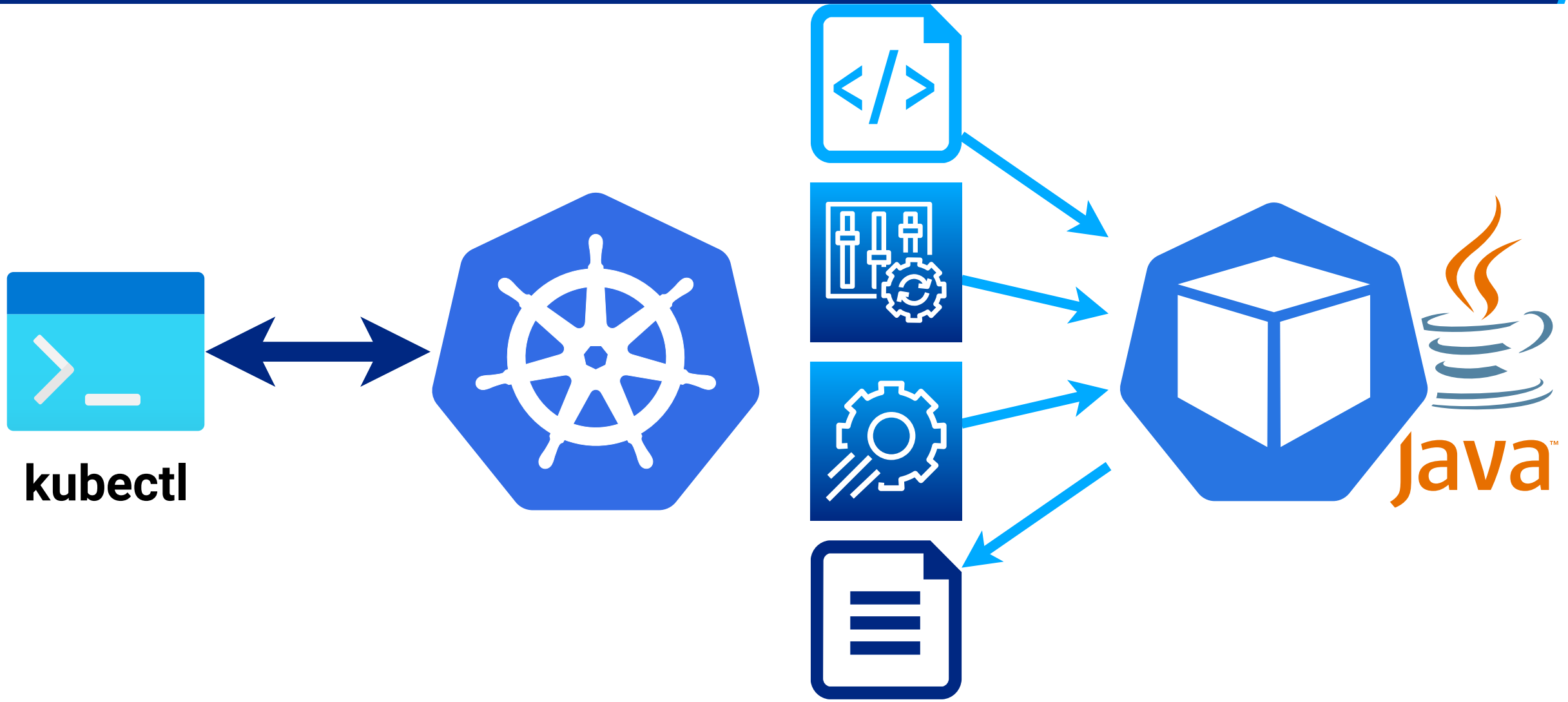
Запустить профилирование



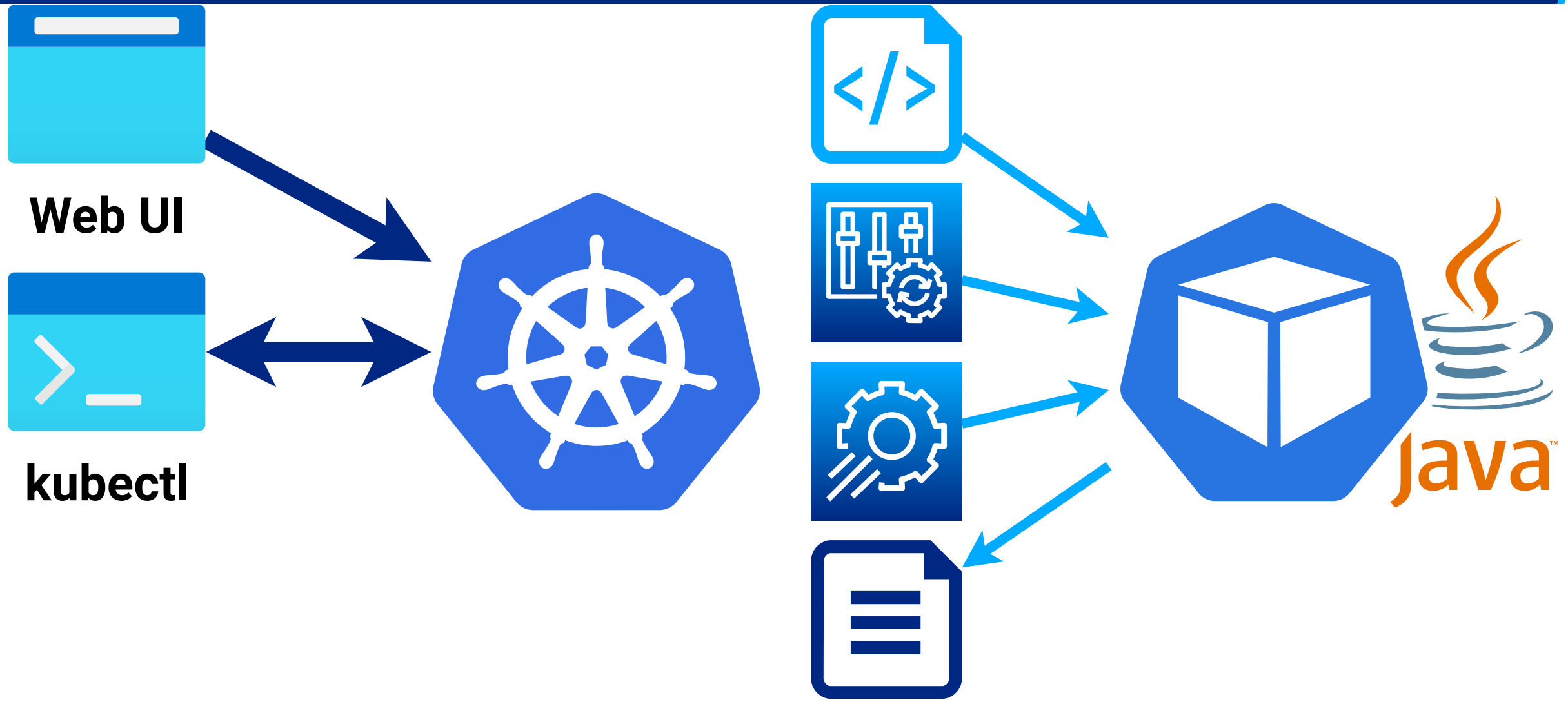
Скачать результаты профилирования



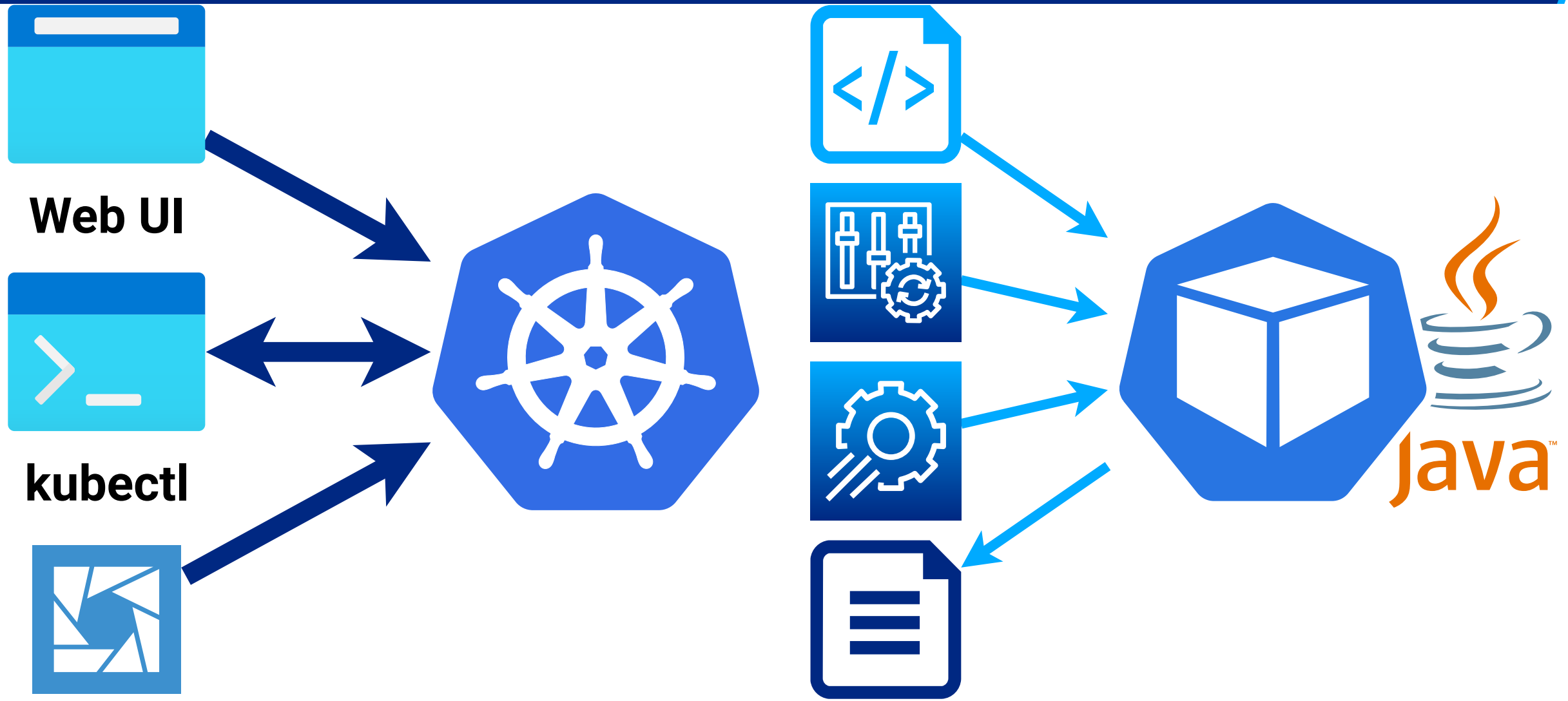
kubectl



kubectl, Web UI (Dashboard)



kubectl, Web UI (Dashboard), Lens, ...



- Автоматизированное тестирование
- Нагрузочное тестирование
  - Автоматизация в НТ
  - Архитектура и Ресурсы стендов НТ и Пром
  - База знаний
    - 0. Подготовка рабочего места
    - 1. Методика, профиль нагрузки, требования
    - 2. Настройка стенда
    - 3. Изучение работы с системой
    - 4. Разработка тестов и заглушек
    - 5. Генерация тестовых данных
    - 6. Запуск тестов
    - 7. Анализ метрик и логов
    - 8. Оформление отчёта и инструкций
  - Заглушки
  - Инфраструктура НТ
  - Методология
  - НТ Мигратора
  - НТ интеграционной шины IIB
  - Последние результаты НТ
  - Предложения


```
Выберите набор сервисов:
```

Например нажать 1 + <Enter> для выбора сервисов rupayment и rupayment-async.

Доступные наборы сервисов хранятся в файле select_pods.sh в функции select_service. При необходимости можно добавить новый набор сервисов.

В ответ получится такой вывод (kubectl по имени сервиса найдет поды):

```
Выбранные поды сервисов:
[Стенд] [Набор]         [Сервис]         [Под]
kpe     rupayment       rupayment        rupayment-6d9b79fb6d-
kpe     rupayment       rupayment-async  rupayment-async-655b7
```

Выбрать команду:

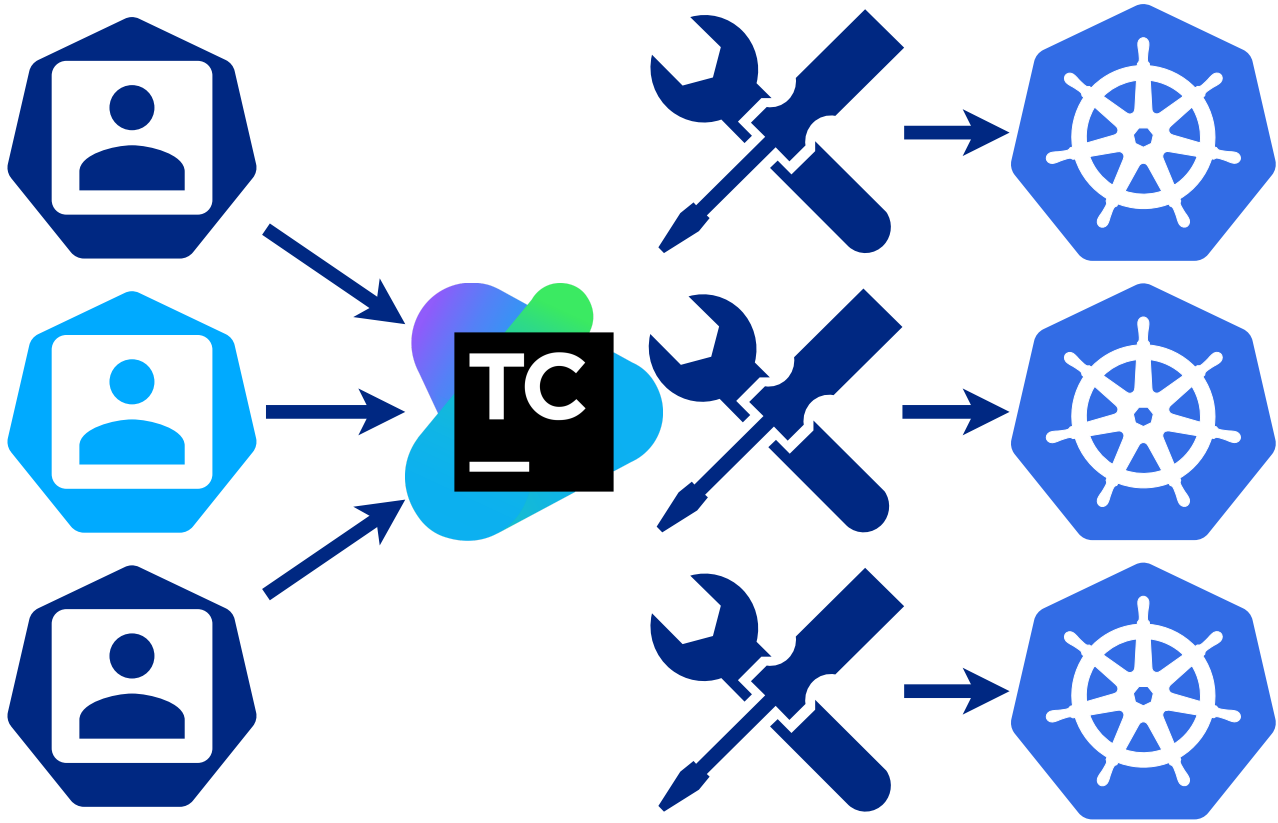
```
Команды:
1) quit                      5) clear_files
2) check_files               6) start_profile_sjk_jmx
3) check_jmx_port            7) download_service_log
4) upload_files              8) download_profile_results
Выберите команду:
```

Нужные команды:

- 4 - загрузить sjk.jar в поду
- 6 - запустить профилирование выбрав длительность профилирования, например 600s
- подождать 600s
- 7 - скачать логи сервиса в локальный каталог target

Помещаем скрипты в CI/CD окружение: добавляем Web UI

Масштабирование



Для микросервисов задач не стало меньше

Особенности Kubernetes

Профайлер

Подключение профайлера к JVM в k8s

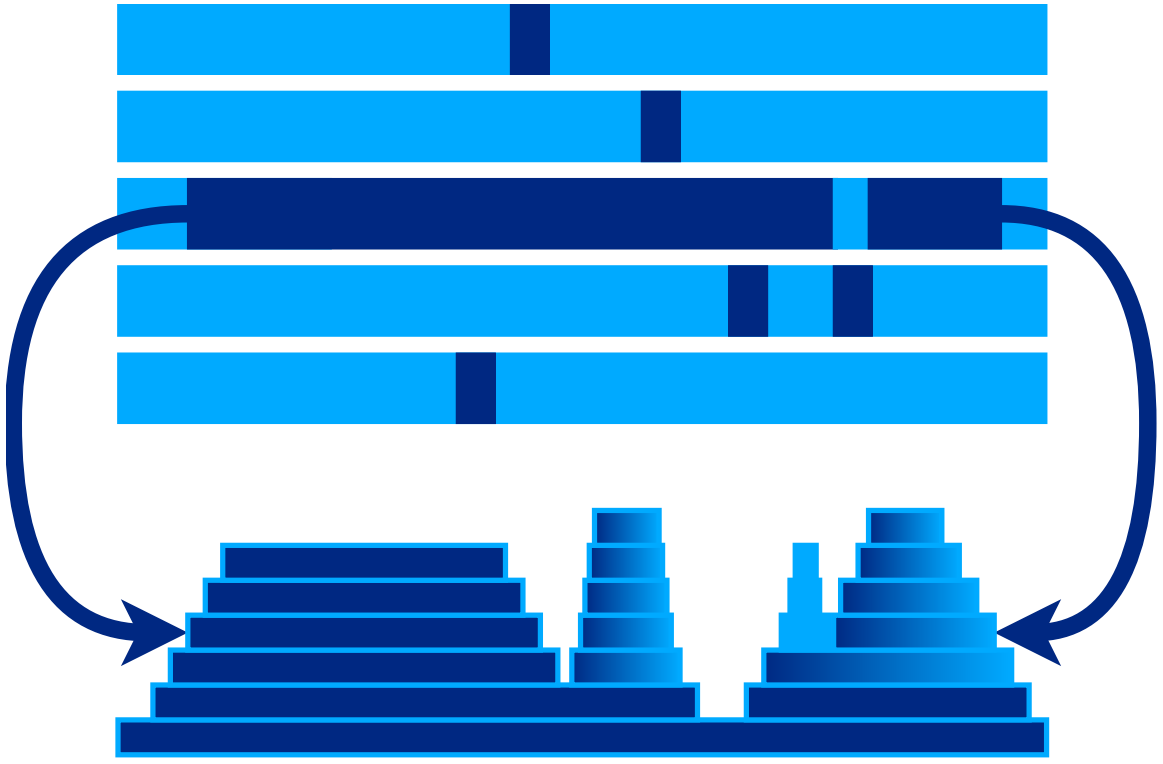
Добавь ресурсов +1 CPU и если JavaAgent, то и +1 GiByte HEAP

При большой нагрузке профилируй локально, семплированием

Для Alpine Linux выбирай musl реализации инструментов

Как выполнять анализ: от потоков к коду

Анализ

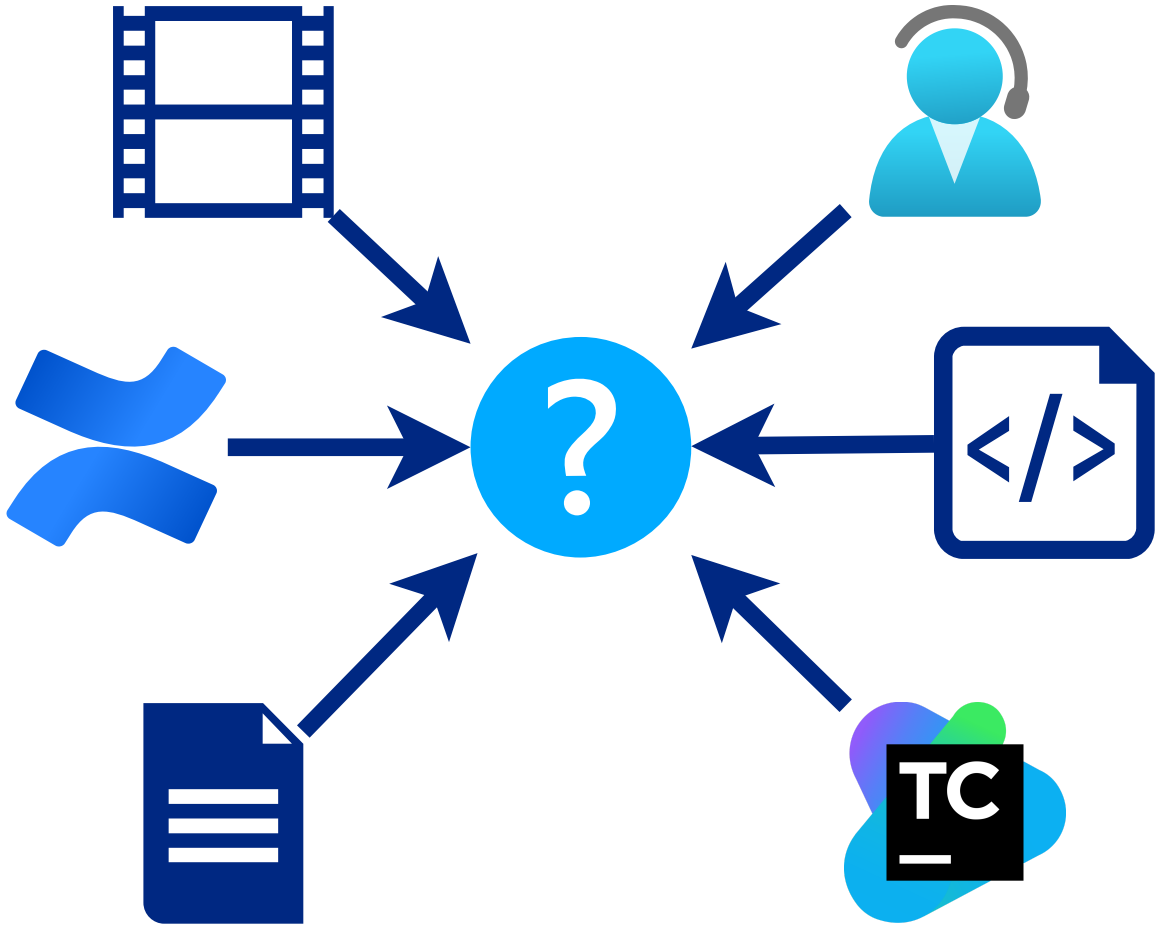


Анализ результатов профилирования

- Посмотреть на потоки визуально
- Собрать статистику по потокам
- Детализировать работу потоков
- JProfiler и YourKit также перехватывают SQL и HTTP-запросы
- С SJK несложно автоматизировать формирование отчета
- JDK Flight Recorder собирает огромное количество метрик

Обмен знаниями, передача опыта, автоматизация

Масштабирование



Обмен знаниями, передача опыта, скрипты

- Документировать результат
- Доброжелательность и терпение
- Стремиться к автоматизации и регрессионному профилированию

Профилирование JVM в Kubernetes : три больших шага

Вопросы и ответы