СБЕР

JAVA

«JAVA, СДЕЛАЙ МНЕ БОЛЬНО!»

Егор Зиборов | Сбер

whoami

whoami

СБЕР

2014
МФТИ

11 лет
в IT

5 лет
в Сбере

ПланЪ

СБЕР

01 Немного контекста

02 Немного
про Spring AOP

03 Совсем чуть-чуть
про DataSource

04 GRPC на дорожку

СБЕР

НЕМНОГО КОНТЕКСТА

СБОЛ жи есть

СБЕР

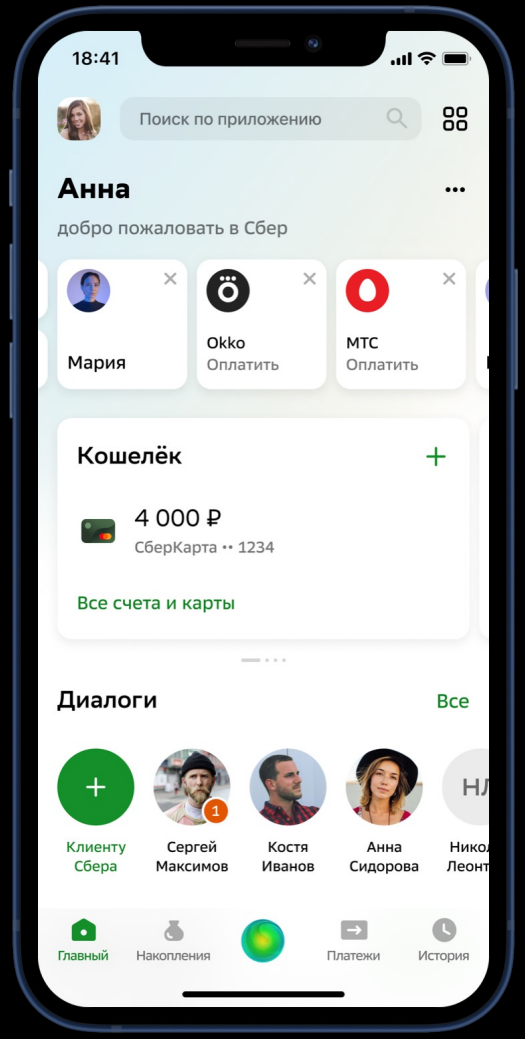
80 млн

Активных клиентов в месяц

MAU

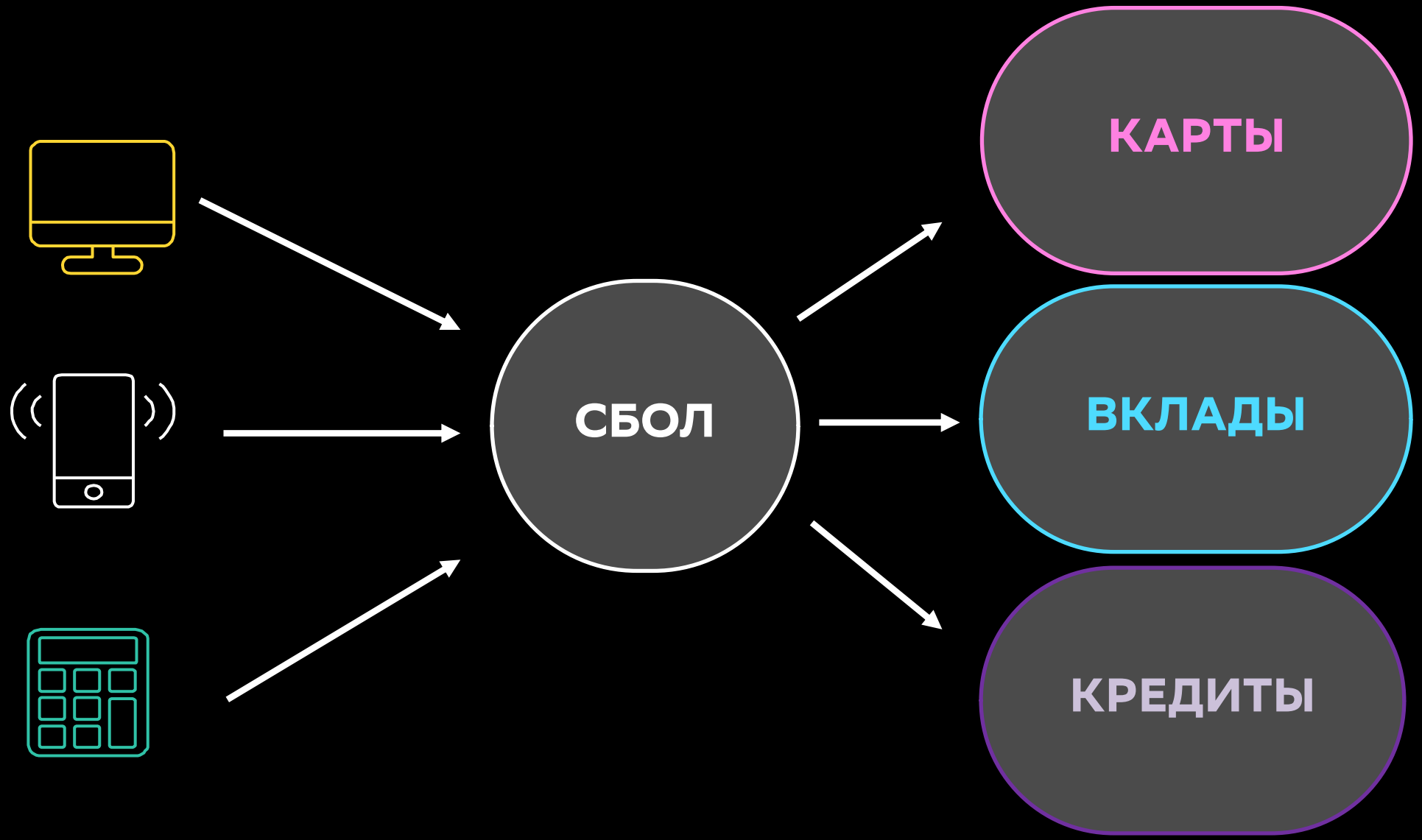
4000 rps

Количество одновременных входов в приложение



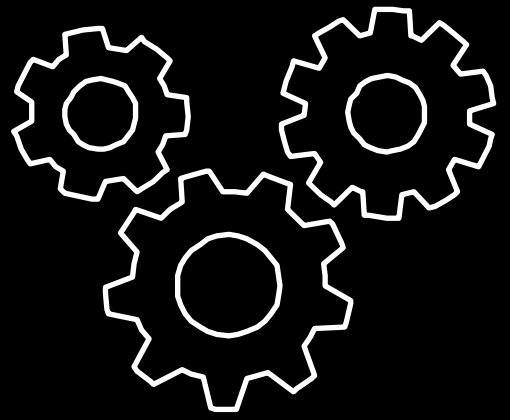
СБОЛ жи есть

СБЕР



TechOverview

СБЕР



Disclaimer

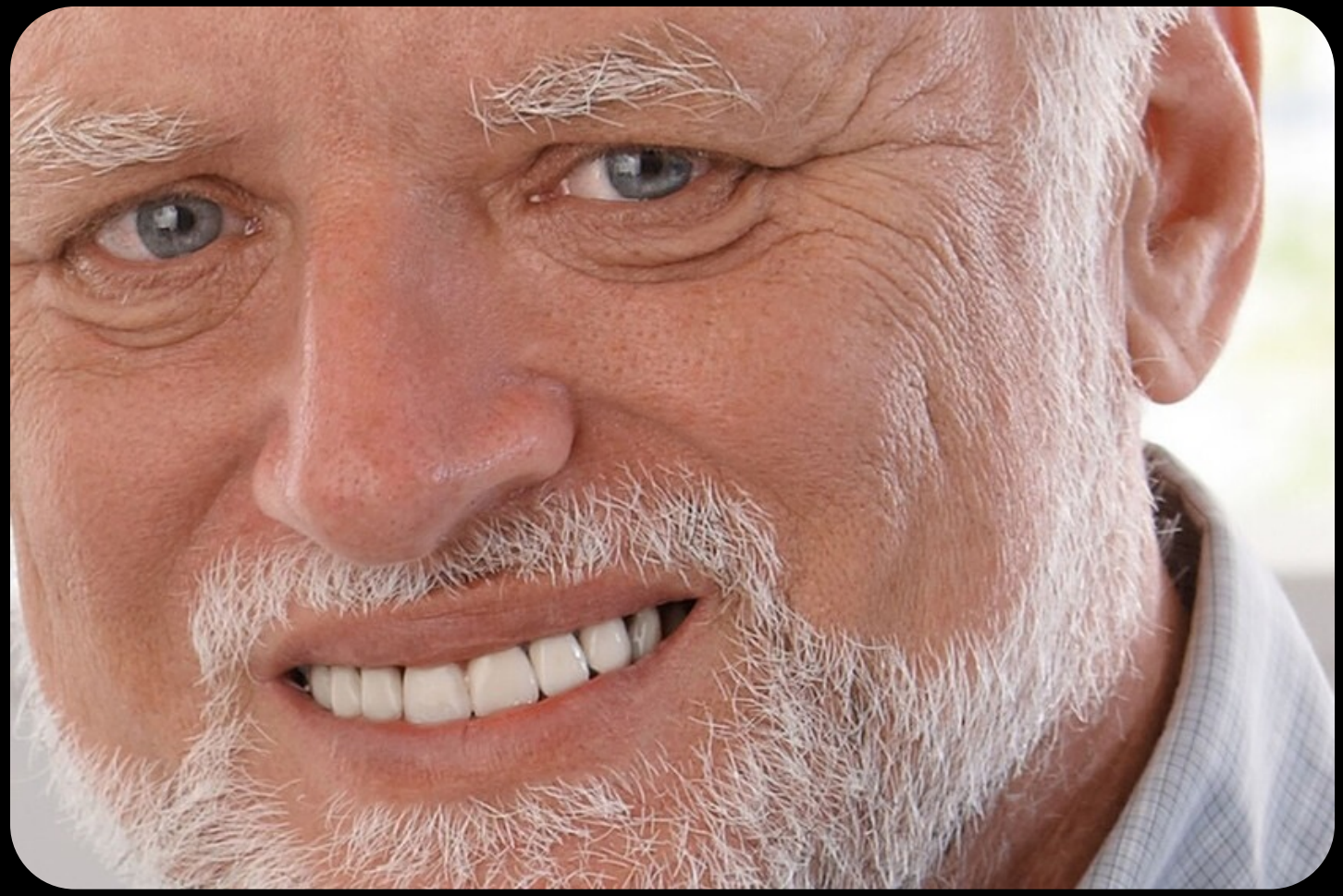
СБЕР

О чем?

- Про опыт
- Про ресерч
- Про боль

Для кого?

- Для разработчиков
- Для Ops инженеров



СБЕР

ОДНАЖДЫ
В SPRING AOP

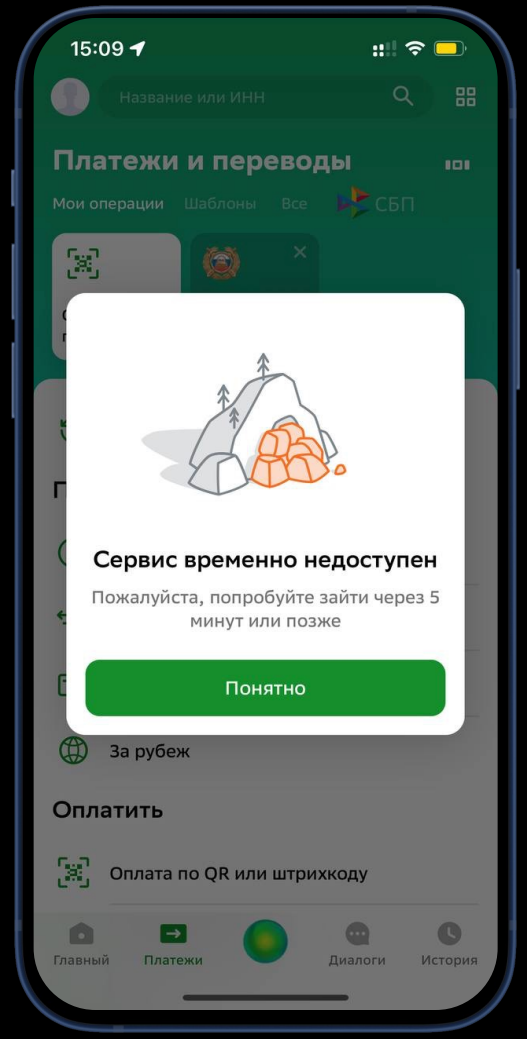
Новый релиз...

СБЕР

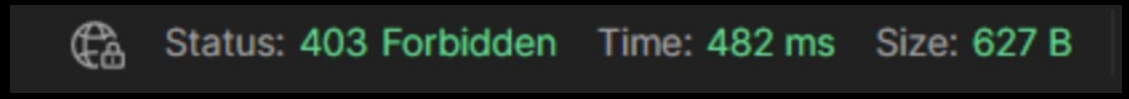
Новые проблемы

Последовательность событий:

- Новый релиз
- Через день начались обращения клиентов
- Мониторинг начал зашкаливать
- Откатываем релиз



Что за ошибки-то?



Что делать-то?

СБЕР

Как исследуем плавающие дефекты?

Локализуем:

- Один VM?
- Один ExecutorService?
- Один Thread?

TL курильщика

СБЕР

```
ThreadLocal<Context> threadLocal = new ThreadLocal();

public void performWithThreadLocal() {
    threadLocal.set(new Context("some state"));
    //my business logic
    threadLocal.remove();
}
```

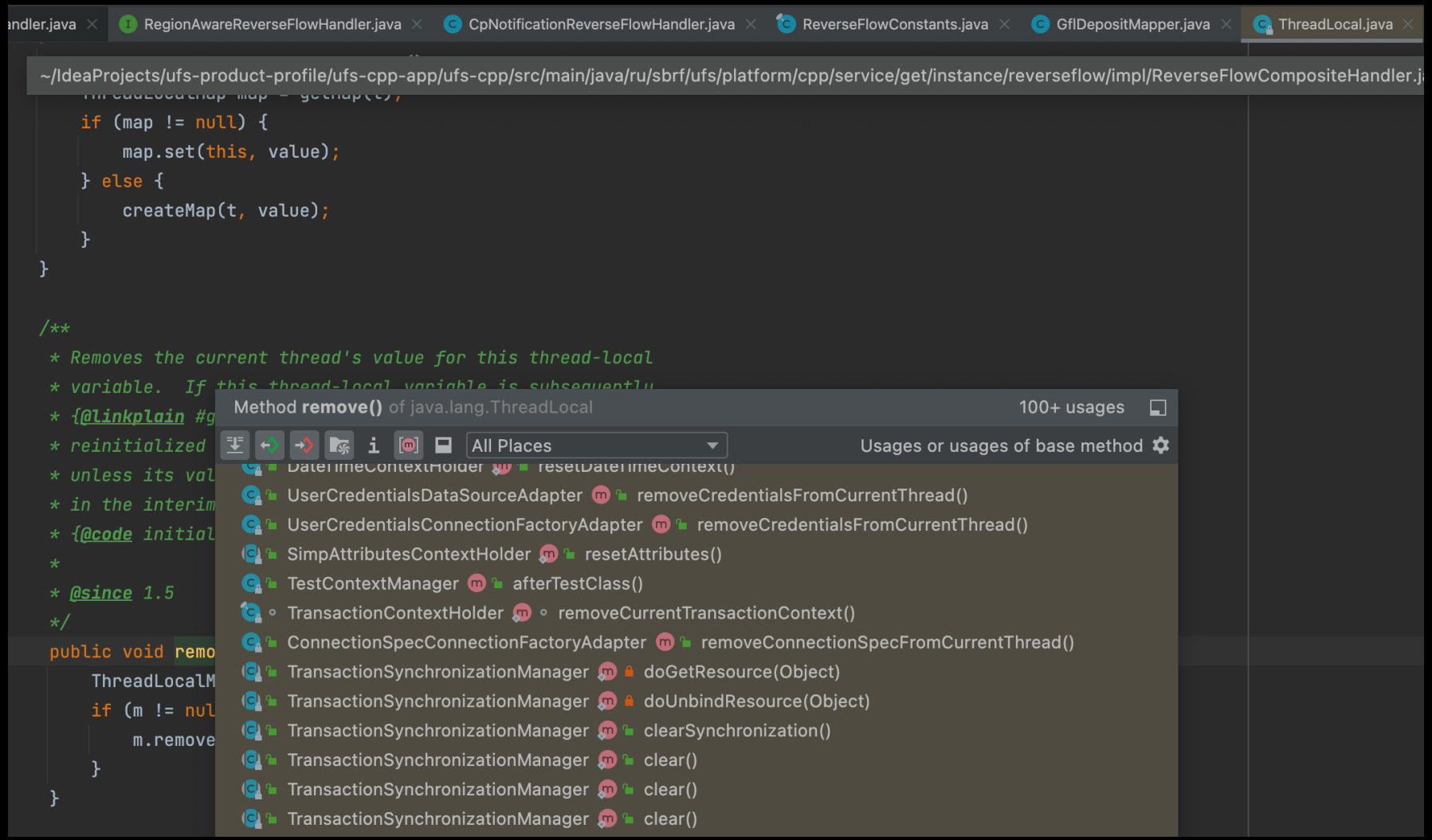
TL здорового человека

СБЕР

```java
ThreadLocal<Context> threadLocal = new ThreadLocal();

public void performWithThreadLocal() {
    threadLocal.set(new Context("some state"));

    try {
        //my business logic
    } finally {
        threadLocal.remove();
    }
}
```

В поисках проблемы

СБЕР



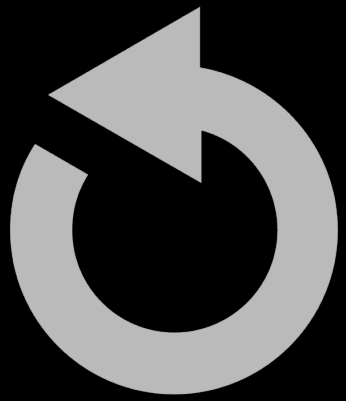
Идем к истоку

СБЕР

Логи заротировались…

Воспроизводим ошибку еще раз

Ищем первую ошибку и…



```
Caused by: java.lang.StackOverflowError: null
    at java.base/java.lang.ThreadLocal.set(ThreadLocal.java:220) ~[na:na]
    at org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(ExposeInvocationInterceptor.java:96)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
    at org.springframework.aop.framework.CglibAopProxy$DynamicAdvisedInterceptor.intercept(CglibAopProxy.java:691)
```


СОВПАДЕНИЕ?
НЕ ДУМАЮ

Что там в Spring’е?

СБЕР

ExposeInvocationInterceptor

```
public Object invoke(MethodInvocation mi) throws Throwable {
    MethodInvocation oldInvocation = (MethodInvocation)invocation.get();
    invocation.set(mi);

    Object var3;
    try {
        var3 = mi.proceed();
    } finally {
        invocation.set(oldInvocation);
    }

    return var3;
}
```

Internal – API

Возвращает предыдущее значение

Разовый провал или...?

СБЕР

TransactionAspectSupport

```
private void bindToThread() {
    this.oldTransactionInfo =
(TransactionAspectSupport.TransactionInfo)TransactionAspectSupport.transactionInfoHolder.get();
    TransactionAspectSupport.transactionInfoHolder.set(this);
}

private void restoreThreadLocalStatus() {
    TransactionAspectSupport.transactionInfoHolder.set(this.oldTransactionInfo);
}
```

AopContext

```
    @Nullable
    static Object setCurrentProxy(@Nullable Object proxy) {
        Object old = currentProxy.get();
        if (proxy != null) {
            currentProxy.set(proxy);
        } else {
            currentProxy.remove();
        }

        return old;
    }
```


РЕБЯТА, НА ЕГО МЕСТЕ
ДОЛЖЕН БЫЛ БЫТЬ Я!

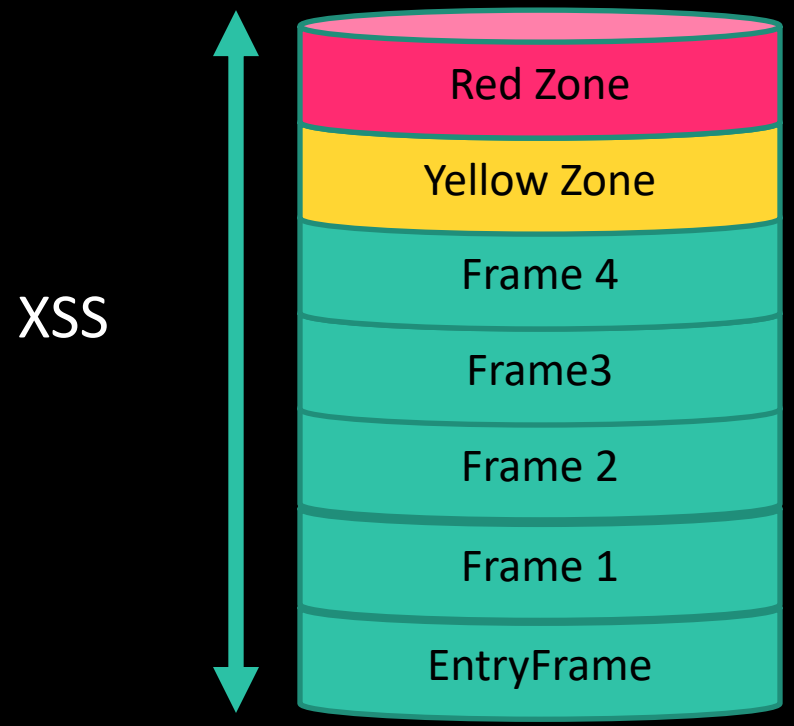
А локи как?

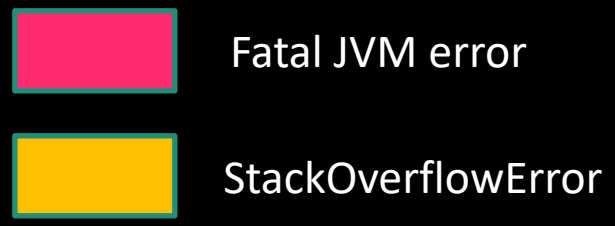
СБЕР

```
Lock someLock = new ReentrantLock();

public void processWithLock() {
  someLock.lock();
  try {
    //do some business logic
  } finally {
    someLock.unlock();
  }
}
```

StackOverflow в Java

СБЕР



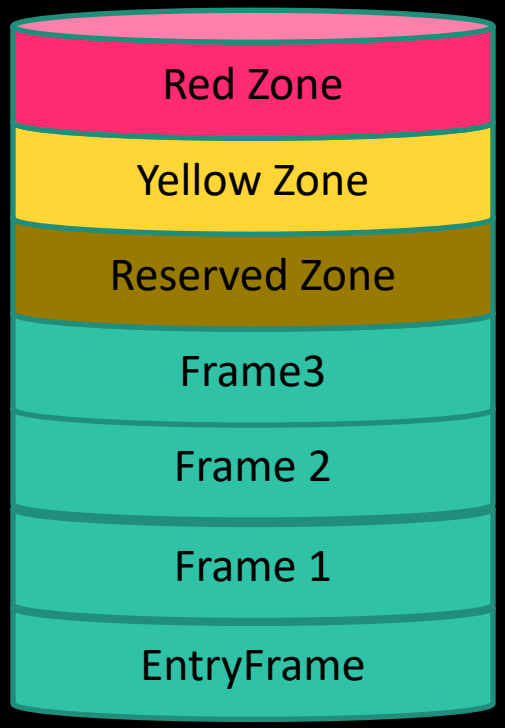


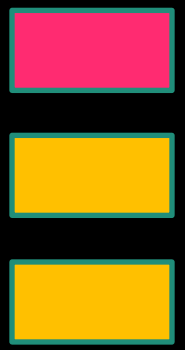
Java решает! (с jdk 9)

СБЕР

JEP 270: Reserved Stack Areas for Critical Sections

Reserve extra space on thread stacks for use by critical sections, so that they can complete even when stack overflows occur.





Fatal JVM error

StackOverflowError

StackOverflowError ПОСЛЕ выхода из секции

https://pangin.pro/posts/stack-overflow-handling

Чудо-аннотация

СБЕР

```
package jdk.internal.vm.annotation;

...
/**
...
 * @since 9
 */
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.METHOD, ElementType.CONSTRUCTOR})
public @interface ReservedStackAccess { }
```

Чудо-фикс

СБЕР

```
package java.lang;

...

public class ThreadLocal<T> {
...
    @ReservedStackAccess
    public void set(T value) {
        ...
    }

    @ReservedStackAccess
    public void remove() {
        ...
    }
...
}
```

А воспроизвести?

СБЕР

Логируй, логируй...

СБЕР

```
private static final Logger logger =  ...

ThreadLocal<Context> threadLocal = new ThreadLocal();

public void performWithThreadLocal() {
    threadLocal.set(new Context("some state"));

    try {
        //my business logic
    } finally {
        log.debug("I'm in finally!!!");
        threadLocal.remove();
    }
}
```

Логируй правильно!

СБЕР

```
private static final Logger logger =  ...

ThreadLocal<Context> threadLocal = new ThreadLocal();

public void performWithThreadLocal() {
    threadLocal.set(new Context("some state"));

    try {
        //my business logic
    } finally {
        threadLocal.remove();
        log.debug("I'm in finally!!!");
    }
}
```

Или даже так

СБЕР

```
ThreadLocal<Context> threadLocal = new ThreadLocal();

public void performWithThreadLocal() {

    try {
        threadLocal.set(new Context("some state"));

        //my business logic

    } finally {
        threadLocal.remove();
    }
}
```

Мораль

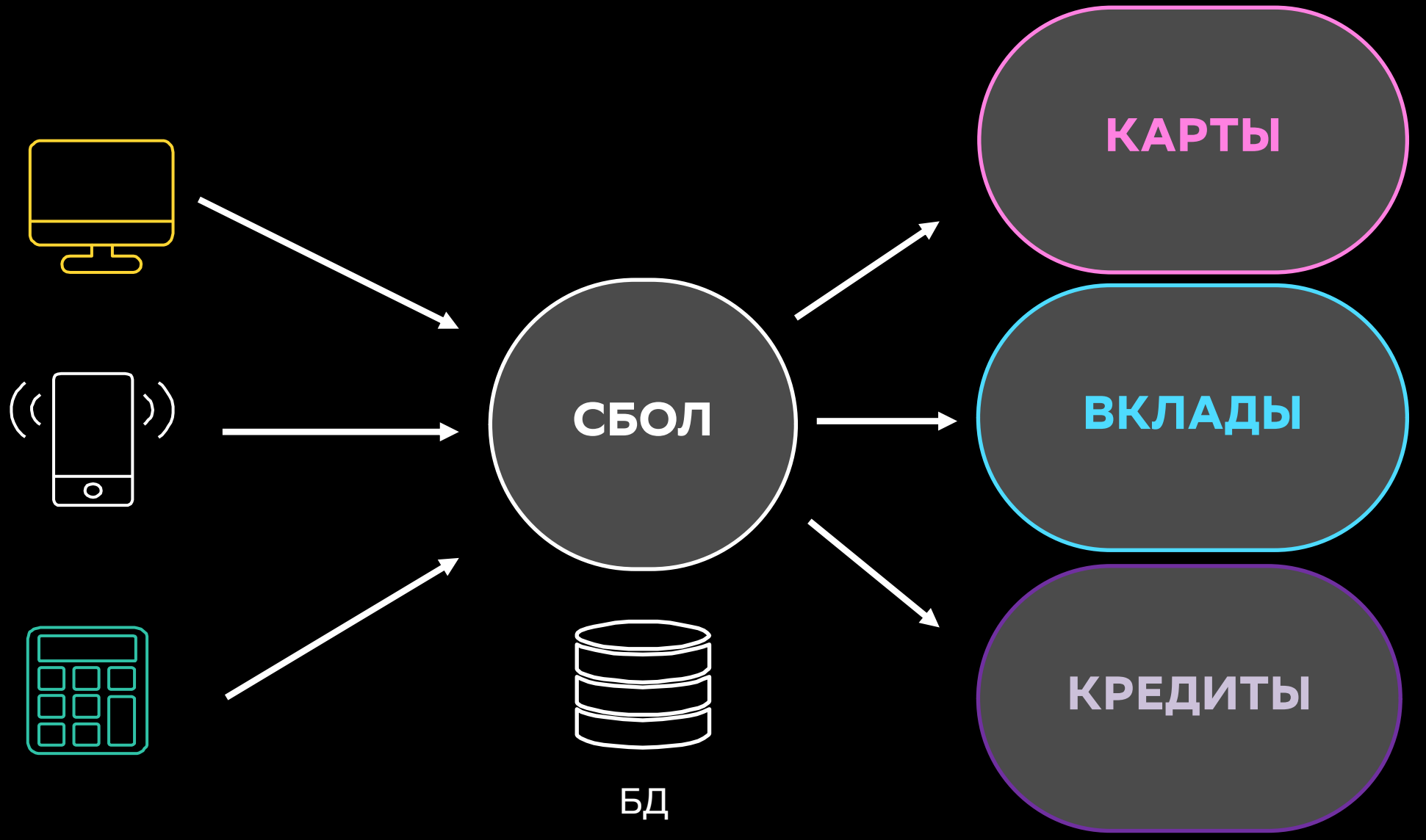
- Нет StackOverFlow – нет проблем!
- Мониторь StackOverflowError и OutOfMemoryError в системе

СБЕР

СОВСЕМ НЕМНОГО DATASOURCE’ОВ

У вас и БД есть?

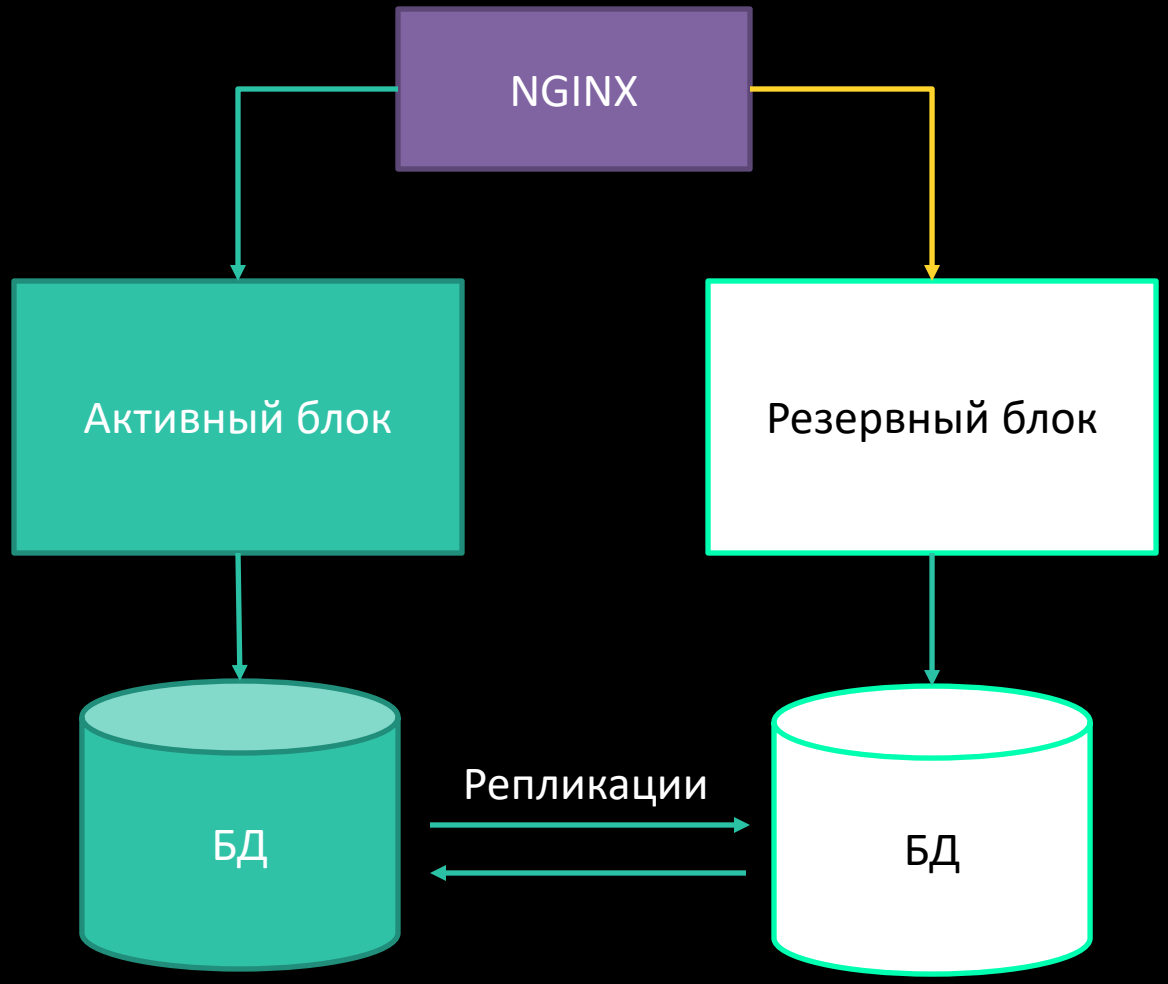
СБЕР



Зачем?

- Кэш
- Мастер-данные

StandIn



Что важно?

- Есть резервный блок, без нагрузки
- В любой момент можно перевести всю нагрузку на него

DataSource – это просто

СБЕР

Как ~~мы~~ все работают с DataSource

```
@Service
public MyService {
  //DataSource variable
  private final DataSource dataSource;

  @Autowired
  public MyService(DataSource dataSource) {
    this.dataSource = dataSource
  }

  public void executeSql(String query) {
    try (Connection conn = dataSource.getConnection()) {
      try (PreparedStatement ps = conn.preparedStatement(query)) {
        ps.execute();
      }
    }
  }
}
```

И вот однажды...

Что произошло?

Переключение в StandBy под нагрузкой

Не прогретые DataSource’ы

Массовое создание Connection’ов

Logon Storm на БД

Но в DataSource же....

СБЕР

General Properties

Scope

cells:tvlda-efs001799Cell01:nodes:tvlda-efs001799Node01:servers:server1

Connection timeout

2 seconds

Maximum connections

30 connections

Minimum connections

1 connections

Reap time

300 seconds

Unused timeout

300 seconds

Aged timeout

600 seconds

Purge policy

EntirePool

Back

Как же так...?

WebSphere::Minimum connections

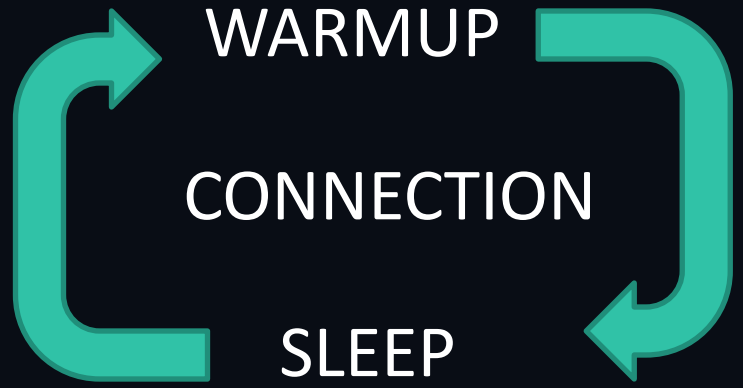
This property defines the minimum number of physical connections that will be kept open in the free pool. *When the JVM is started, WAS does not automatically create the number of connections that is defined by this property*.

HikariCP::minimumIdle

🔢minimumIdle
This property controls the minimum number of *idle connections* that HikariCP tries to maintain in the pool. If the idle connections dip below this value and total connections in the pool are less than maximumPoolSize, HikariCP will make a best effort to add additional connections quickly and efficiently. However, for maximum performance and responsiveness to spike demands, we recommend *not* setting this value and instead allowing HikariCP to act as a *fixed size* connection pool. *Default: same as maximumPoolSize*

Грелка!

СБЕР



Грелка!

СБЕР

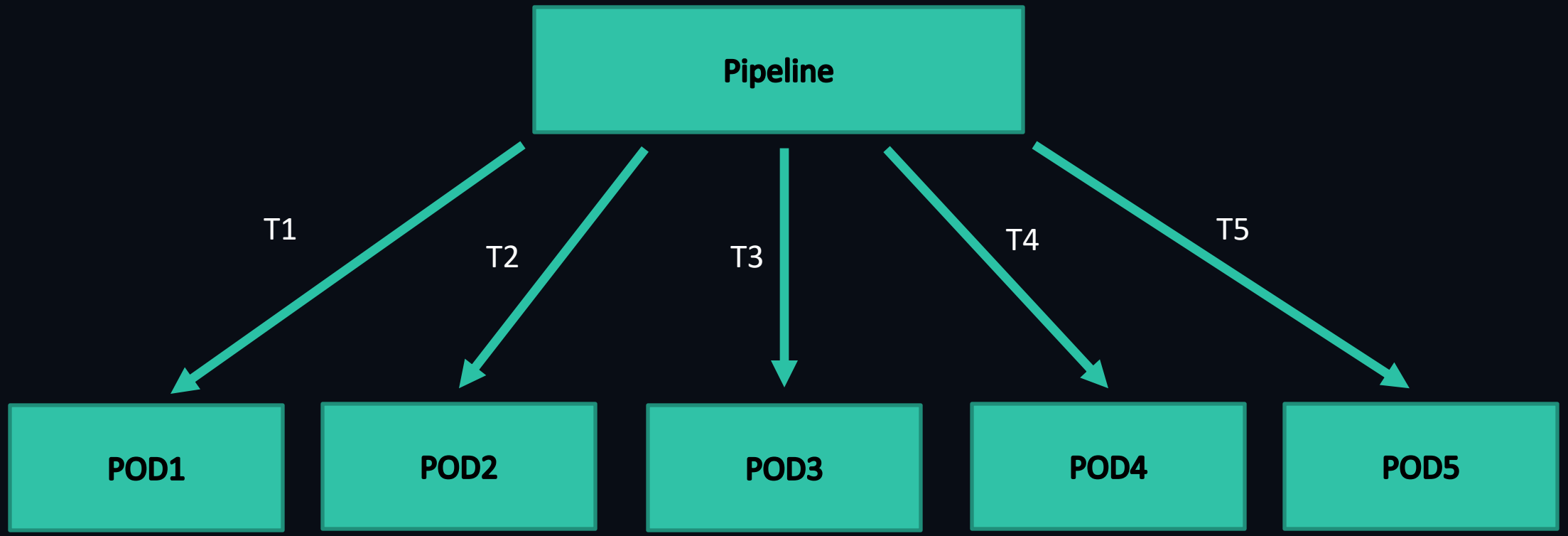
```
public void warmup() {
     List<Future<Connection>> futures = new ArrayList<>(minConnections);
     final RateLimiter rateLimiter = RateLimiter.create(maxOpenRate);

     try {

        for (int i = 0; i < minConnections; i++) {
            rateLimiter.acquire();
            futures.add(executorService.submit(() -> getConnection(minConnections)));
        }

     } finally {
        List<Connection> connections = extractFutures(futures);
        forceOpenConnections(connections);
        releaseOpenedConnection(connections);
     }

}
```

Грелка!

СБЕР

```
@Nullable
private Connection getConnection(int minConnections)
    throws SQLException {
    Connection connection = dataSource.getConnectionUpTo(minConnections);
    ...
    if (connection ≠ null) {
        useConnectionToBypassLazyWrappers(connection);
    }
    return connection;
}

private void useConnectionToBypassLazyWrappers(Connection connection) {
    try {
        connection.createStatement().close();
    } catch (SQLException e) {
        log.warn("Unable to open connection to maintain minConnections", e);
    }
}
```

Немного рандома



Мораль

Есть StandBy?

Проверяй “холодный” переход
на него под полной нагрузкой

СБЕР

GRPC
НА ДОРОЖКУ

Пара слов о

СБЕР

HTTP/2

Создавай

СБЕР

```
protobuf {
    generatedFilesBaseDir = "$projectDir/gen"

    protoc {
        artifact = "com.google.protobuf:protoc:3.12.0"
    }
    plugins {
        grpc {
            artifact = 'io.grpc:protoc-gen-grpc-java:1.37.0'
        }
    }
    generateProtoTasks {
        all()*.plugins {
            grpc {}
        }
    }
}
clean {
    delete protobuf.generatedFilesBaseDir
}
```

gRPC клиент генерируется на основе:

- proto файл
- Google либы
- Gradle таск

Подключай

СБЕР

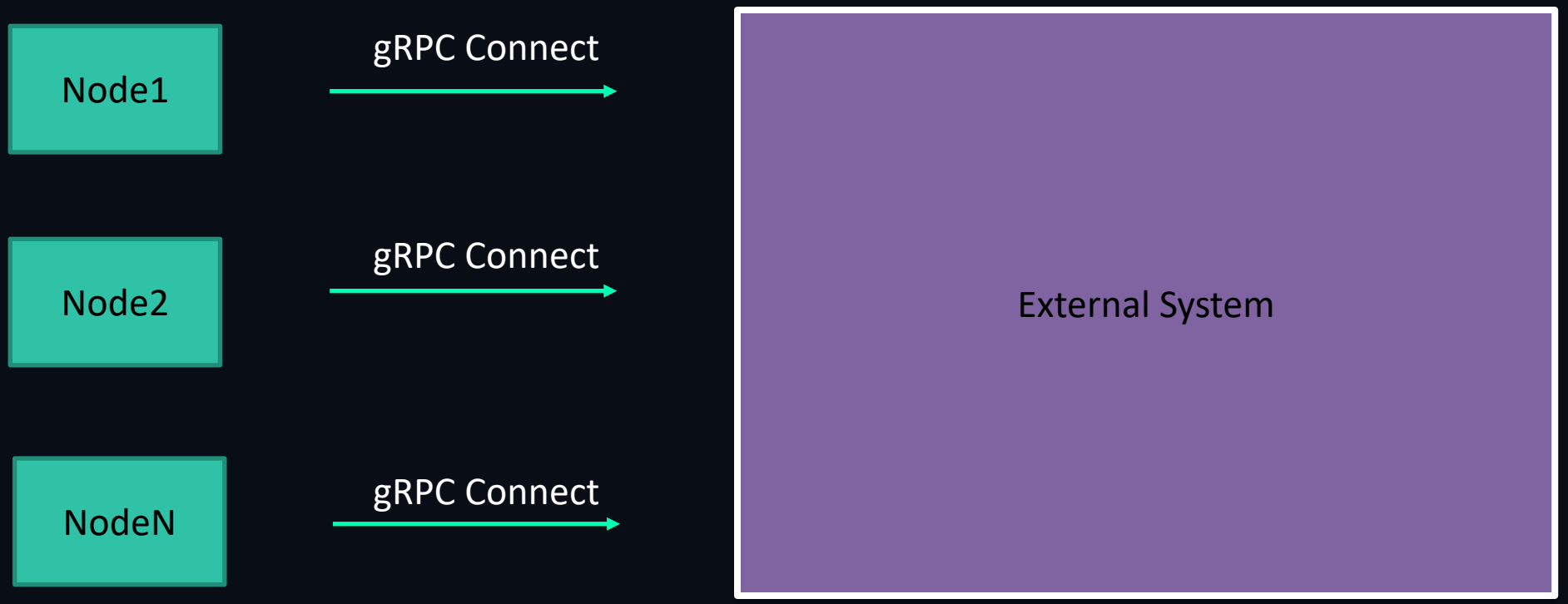
```
    /**
     * Sets stub to client.
     * @param grpcStub grpc stub
     */
    @net.devh.boot.grpc.client.inject.GrpcClient(CHANNEL_NAME)
    public void setStub(SrvgetclientdbopackagesinfoGrpc.SrvgetclientdbopackagesinfoStub grpcStub) {
        this.stub = grpcStub;
    }
```

Используй!

СБЕР

```
@Override
public void callGrpc(FcpGetClientInfoCedboPprbParameters fcpGetClientInfoCedboPprbParameters,
                     GrpcCallback<FcpGetClientInfoCedboPprbResult> grpcCallback) {

    final SrvgetclientdbopackagesinfoGrpc.SrvgetclientdbopackagesinfoStub stubWithHeaders =
        MetadataUtils.attachHeaders(stub, createMetadataWithHeaders(fcpGetClientInfoCedboPprbParameters));

    stubWithHeaders.get(fcpGetClientInfoCedboPprbParameters, new SingleMessageStreamObserver<>(grpcCallback));
}
```

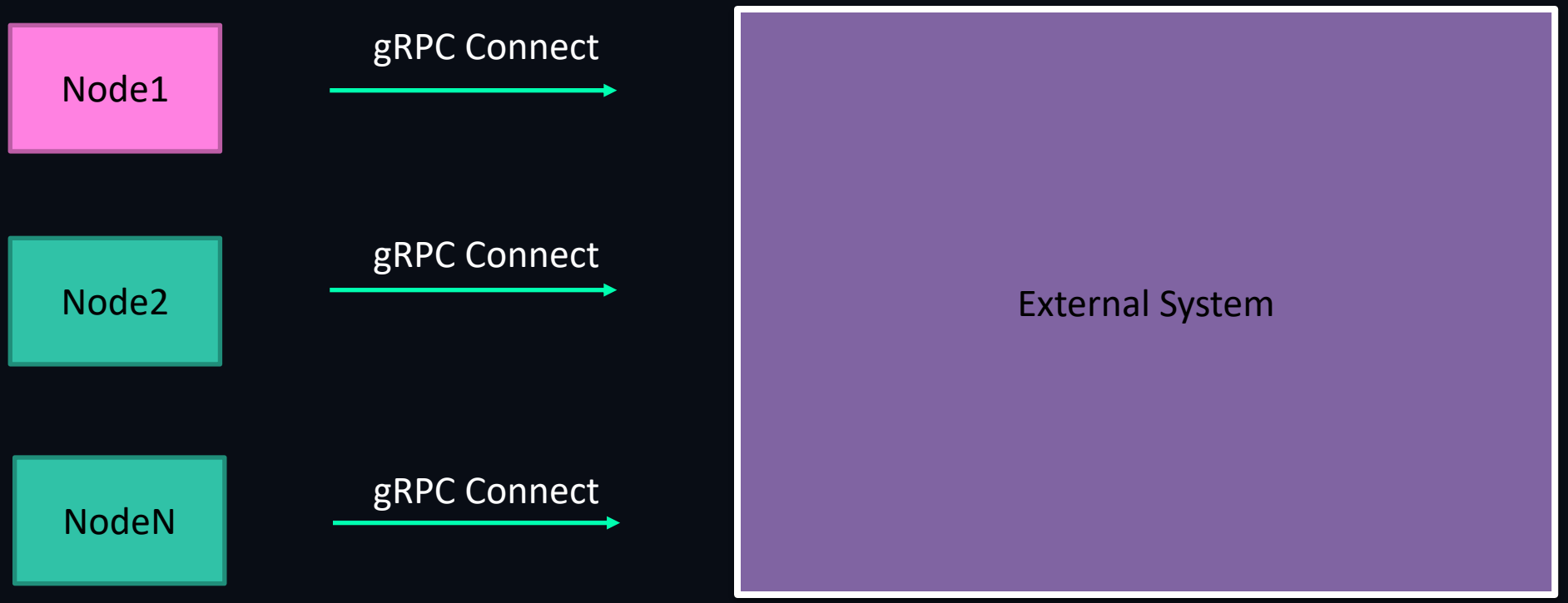
Все работает!



Имеем:

- для всех нод одна точка подключения (LoadBalancer) – ничего не знаем о его структуре
- каждая нода держит один connection

Или нет...

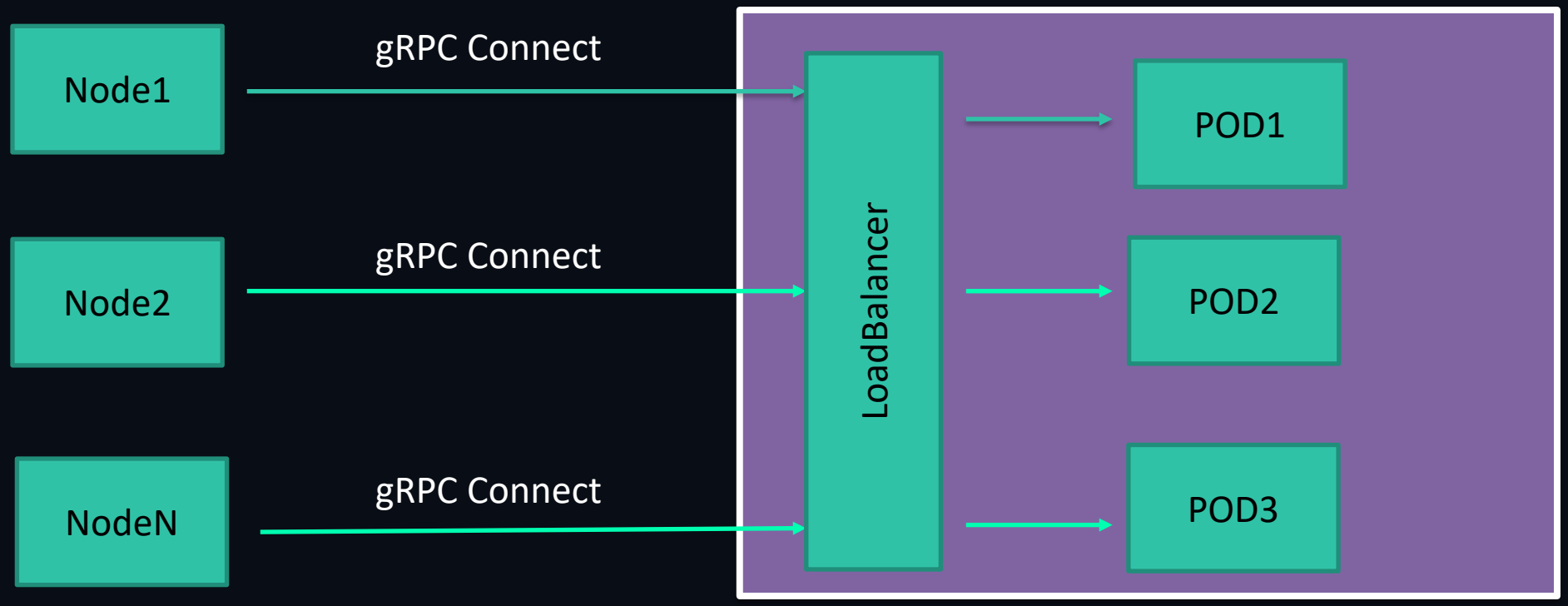


Что-то случилось!

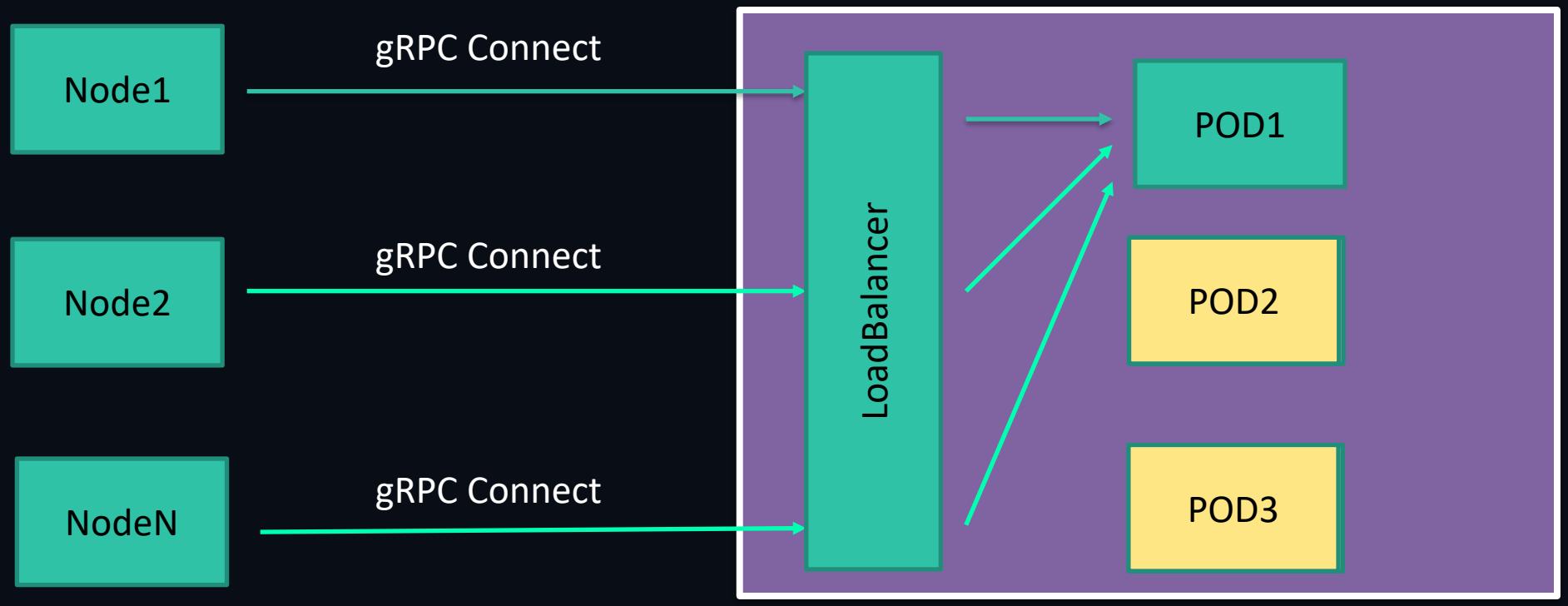
- На одной из нод начались таймауты на 100% запросов

Что делаем? Конечно, ребутаем!

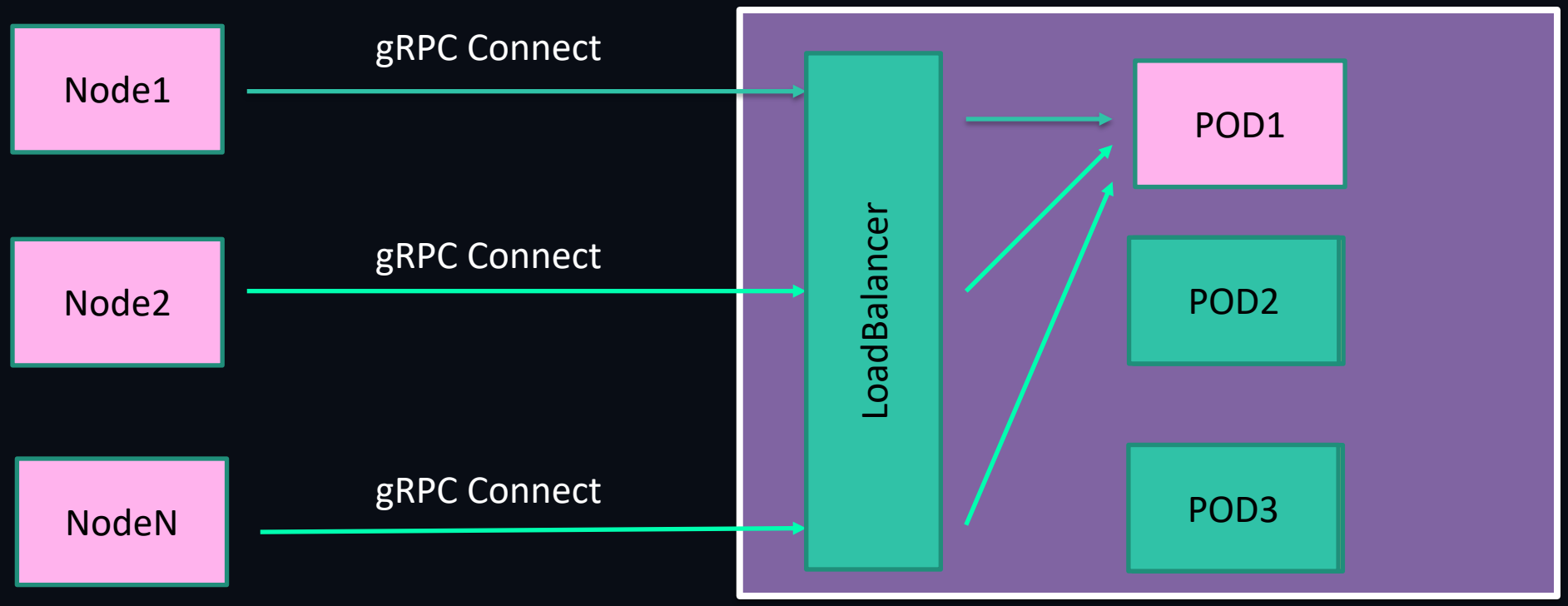
Одним ребутом сыт не будешь



Одним ребутом сыт не будешь



Одним ребутом сыт не будешь



ClientReconnect

СБЕР

io.grpc.ManagedChannel

```java
/**
 * Invoking this method moves the channel into the IDLE state and triggers tear-down of the
 * channel's name resolver and load balancer, while still allowing on-going RPCs on the channel to
 * continue. New RPCs on the channel will trigger creation of a new connection.
 *
 * <p>This is primarily intended for Android users when a device is transitioning from a cellular
 * to a wifi connection. The OS will issue a notification that a new network (wifi) has been made
 * the default, but for approximately 30 seconds the device will maintain both the cellular
 * and wifi connections. Apps may invoke this method to ensure that new RPCs are created using the
 * new default wifi network, rather than the soon-to-be-disconnected cellular network.
 *
 * <p>No-op if not supported by implementation.
 *
 * @since 1.11.0
 */
@ExperimentalApi("https://github.com/grpc/grpc-java/issues/4056")
public void enterIdle() {
  ...
}
```

Иииииспользуем

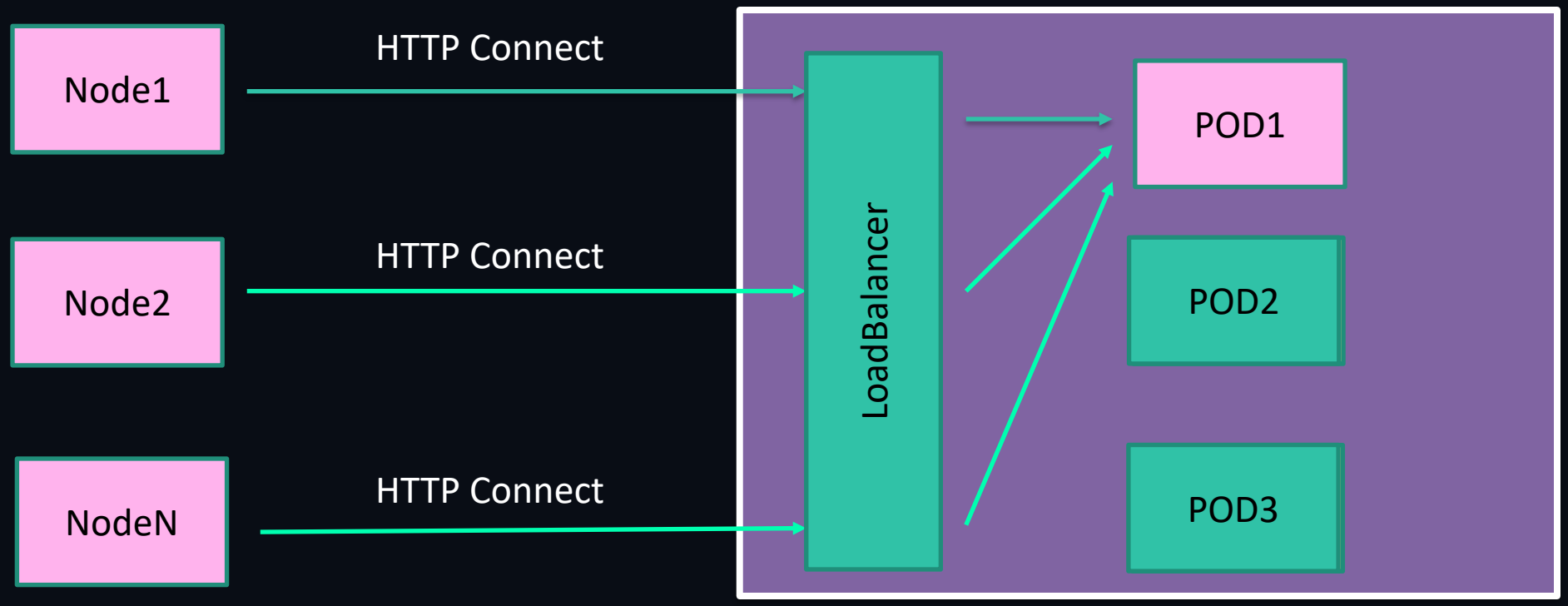
СБЕР

```java
public abstract class AbstractReapable implements Reapable {

    private volatile long reapedAt;

    ...

    @Override
    public void tryReap() {
        final long currentTime = timeSource.currentTimeMillis();
        final long timePassedSinceLastReap = currentTime - reapedAt;

        if (reapByAgedTimeout(timePassedSinceLastReap) || reapByUnusedTimeout(currentTime)) {
            reapedAt = currentTime;
            log.info("Reap procedure has succeeded. [{}] is scheduled for reopen.", resourceName);
        } else {
            log.debug("Reap procedure has succeeded. [{}] should not be scheduled for reopen yet.", resourceName);
        }
    }
```


IT WORKS!
IT FINALLY WORKS!!!

gRPC ли?

СБЕР


Node1
HTTP Connect
Node2
HTTP Connect
NodeN
HTTP Connect
LoadBalancer
POD1
POD2
POD3

Мораль

Есть пул соединений?

Убедись, что в нем есть реконнекты

СБЕР

ИТОГИ

Выводы

1. Что такое СБОЛ
2. StackOverflow в finally блоке
3. Важность проверки переключения на StandBy
4. Важность reconnect’а внутри ConnectionPool’ов

Контакты

СБЕР

@WinZib

winzib@yandex.ru