Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Институт прикладной математики и компьютерных наук

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК
Руководитель ОПОП
д-р техн. наук, профессор
_____ А.В. Замятин
подпись
«03» июня 2023 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

РАСШИРЕНИЕ ДЛЯ ИНТЕГРАЦИИ ИНСТРУМЕНТА ОТЛАДКИ ПРИЛОЖЕНИЙ, РАЗВЕРНУТЫХ В ОБЛАЧНОМ КЛАСТЕРЕ

по направлению подготовки 02.03.02 Фундаментальная информатика и информационные технологии,
направленность (профиль) «Фундаментальная информатика и информационные технологии»

Липатов Александр Сергеевич

Руководитель ВКР
канд. техн. наук, доцент
_____ А.Л. Фукс
«28» мая 2023 г.

Консультант
_____ М.С. Овсянников
«28» мая 2023 г.

Автор работы
студент группы № 931901
_____ А.С. Липатов
«28» мая 2023 г.

Томск – 2023

Министерство науки и высшего образования Российской Федерации.

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Институт прикладной математики и компьютерных наук

УТВЕРЖДАЮ
Руководитель ОПОП
д-р техн. наук, профессор
_____ А.В. Замятин
подпись
«10» ноября 2022 г.

ЗАДАНИЕ

по выполнению выпускной квалификационной работы бакалавра / специалиста /магистра обучающемуся
Липатов Александр Сергеевич
Фамилия Имя Отчество обучающегося
по направлению подготовки 02.03.02 Фундаментальная информатика и информационные технологии, направленность (профиль) «Фундаментальная информатика и информационные технологии»

1 Тема выпускной квалификационной работы

Расширение для интеграции инструмента отладки приложений, развернутых в облачном кластере

2 Срок сдачи обучающимся выполненной выпускной квалификационной работы:
а) в учебный офис / деканат – 28.05.2023 б) в ГЭК – 07.06.2023

3 Исходные данные к работе:
Объект исследования – Разработка настольных приложений
Предмет исследования – Компонент настольного приложения
Цель исследования – разработка расширения для интеграции Telepresence в виде пользовательского интерфейса

Задачи:
Проанализировать возможности Kubernetes и Telepresence, подготовить функциональные требования и архитектурное описание решения, реализовать, отладить и протестировать полученное решение

Методы исследования:
Теоретическое исследование и практическая реализация

Организация или отрасль, по тематике которой выполняется работа, –
ООО «Вдом Ресерч», г. Томск

4 Краткое содержание работы
Исследование возможностей платформы Kubernetes и утилиты Telepresence, проектирование и реализация решения с графическим интерфейсом для интеграции

Руководитель выпускной квалификационной работы
канд. техн. наук, доцент кафедры ТОИ _____ / А.Л. Фукс
должность, место работы *подпись* *И.О. Фамилия*

Задание принял к исполнению
Студент, группа 931901 _____ / А.С. Липатов
должность, место работы *подпись* *И.О. Фамилия*

АННОТАЦИЯ

К выпускной квалификационной работе бакалавра на тему ”Расширение для интеграции инструмента отладки приложений, развернутых в облачном кластере”

Работа включает: 34 страницы, 19 рисунков, 12 использованных источников

Цель работы - разработка расширения для интеграции Telepresence для предоставления удобного пользовательского интерфейса внутри интеграциионной среды разработки.

По результатам исследования было реализовано расширение Telepresence для интегрированной среды разработки Visual Studio Code

Реализованное расширение предлагает разработчикам Visual Studio Code удобную работу с Telepresence в виде графического пользовательского интерфейса.

ОГЛАВЛЕНИЕ

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

1. **Интегрированная среда разработки (IDE)** - это программное приложение, которое помогает программистам эффективно разрабатывать программный код.

2. **Программный интерфейс приложения (API)** - описание способов взаимодействия одной компьютерной программы с другими.

3. **Visual Studio Code (VS Code, Code)** - текстовый редактор, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации в виде расширений.

4. **Расширение (Плагин, дополнение или модуль)** - независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения и/или использования её возможностей. Плагины обычно выполняются в виде библиотек общего пользования.

5. **Kubernetes (k8s)** - открытое программное обеспечение для оркестровки контейнеризированных приложений - автоматизации их развёртывания, масштабирования и координации в условиях кластера. Поддерживает основные технологии контейнеризации, включая Docker, rkt, также возможна поддержка технологий аппаратной виртуализации.

6. **Telepresence** - инструмент в виде консольной утилиты, предоставляющий множество команд для локальной разработки и отладки приложений, развернутых в удаленном кластере Kubernetes.

7. **Minikube** - это упрощённый инструмент для запуска полноценного кластера Kubernetes на локальной машине. Он был разработан, чтобы облегчить разработку, тестирование и отладку приложений, работающих на Kubernetes, в локальной среде.

8. **TypeScript** - язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript.

ВВЕДЕНИЕ

Тема работы ”Расширение для интеграции инструмента отладки приложений, развернутых в облачном кластере”является актуальной, так как исследование связано с относительно новым инструментом для разработки в кластере Kubernetes - Telepresence. Разработчикам, не знакомым с данным инструментом, будет предложено решение с пользовательским интерфейсом в одной из наиболее популярных интегрированных сред разработки. Проблема имеет практическое значение, так как предлагает полезный для разработчика функционал внутри интегрированной среды разработки.

Разработка дополнений для интегрированной среды разработки является регулярной практикой среди желающих расширить функционал IDE разработчиков. Организация, разрабатывающая свой продукт интегрированныой среди разработки, предоставляет публичный интерфейс для того, чтобы разработчики имели возможность расширить функционал продукта за счет модулей динамических модулей. Удобство предоставляемых API и их полная документация значительно определяет успех IDE как программного продукта в настоящее время.

Telepresence является относительно новой технологией: вторая версия инструмента вышла в ноябре 2022 года и была реализована на Go. Существует несколько статьей, содержащих исследование возможностей Telepresence для локальной разработки внутри удаленного кластере Kubernetes с точки зрения прикладного разработчика.

Целью работы является исследование возможности интеграции Telepresence как часть пользовательского интерфейса в одну из наиболее используемых интегрированных сред разработки. Результатом работы должен стать реализованный модуль, предлагающий пользователям IDE основной функционал Telepresence внутри интегрированной среды разработки как часть пользовательского интерфейса.

Объектом исследования является непосредственно инструмент Telepresence - консольная утилита для создания двунаправленного прокси соединения между рабочей станцией разработчика и удаленным кластером Kubernetes. Также предстоит исследовать особенности работы с Kubernetes и API выбранной интегрированной среды разработки, для которого

впоследствии будет разработан модуль.

Предметом исследования является разработка расширения, включая проектирование, реализацию и оценку решения. Целью исследования является предоставление новых возможностей IDE за счет разработки расширения.

Существует несколько популярных IDE для работы с ресурсами Kubernetes: Lens, Monokle, VS Code (расширение Kubernetes), k9s (консольное приложение, имитирующее графический интерфейс) или продукты компании JetBrains. В каждом из них отсутствует интеграция с Telepresence или не поддерживается интеграция модулей.

Предложенное решение поспособствует росту популярности использования в разработке и отладки приложений инструмента Telepresence путем предоставления удобного пользовательского интерфейса внутри одной из популярных IDE для разработки в Kubernetes.

1 Проведение анализа

1.1 Функционал платформы облачного кластера

В качестве облачного кластера рассматривается платформа Kubernetes.

Согласно официальной документации Kubernetes [1]: kubernetes - это портативная и расширяемая платформа для управления контейнеризованными рабочими нагрузками и сервисами, которая предоставляет возможность декларативной настройки состояния и автоматизации в кластере.

Kubernetes обладает огромным потенциалом относительно других традиционных решений и подходов и на настоящее время существует без конкуренции, получая активное развития от сообщества разработчиков [2].

Для необходимого понимания успеха платформы Kubernetes для разработчиков обратимся к истории развертывания приложений, выделив три периода развертывания приложений и их развития.

В первую эру развертывания приложения запускались на физических серверах. Каждое приложение требовало выделенных ресурсов и инфраструктуры, что создавало проблемы с масштабированием, управлением и отказоустойчивостью. При необходимости добавления новых приложений или масштабирования существующих требовалось установка и настройка новых серверов, что было трудоемким и затратным процессом. На рисунке 1 отображены слои системы для развертывания приложений.

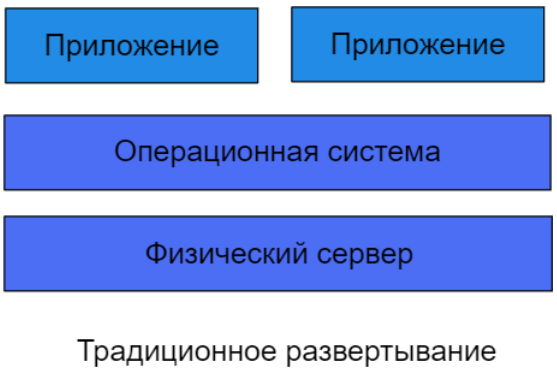


Рисунок 1 – Традиционный подход

С развитием виртуализации стали появляться виртуальные машины (VM), которые позволяли запускать несколько экземпляров операционной

системы на одном физическом сервере. Это снизило требования к физическим ресурсам и упростило управление приложениями. Однако, управление множеством VM все еще было сложным и требовало ручной настройки и масштабирования. На рисунке 2 отображен принцип виртуального подхода.

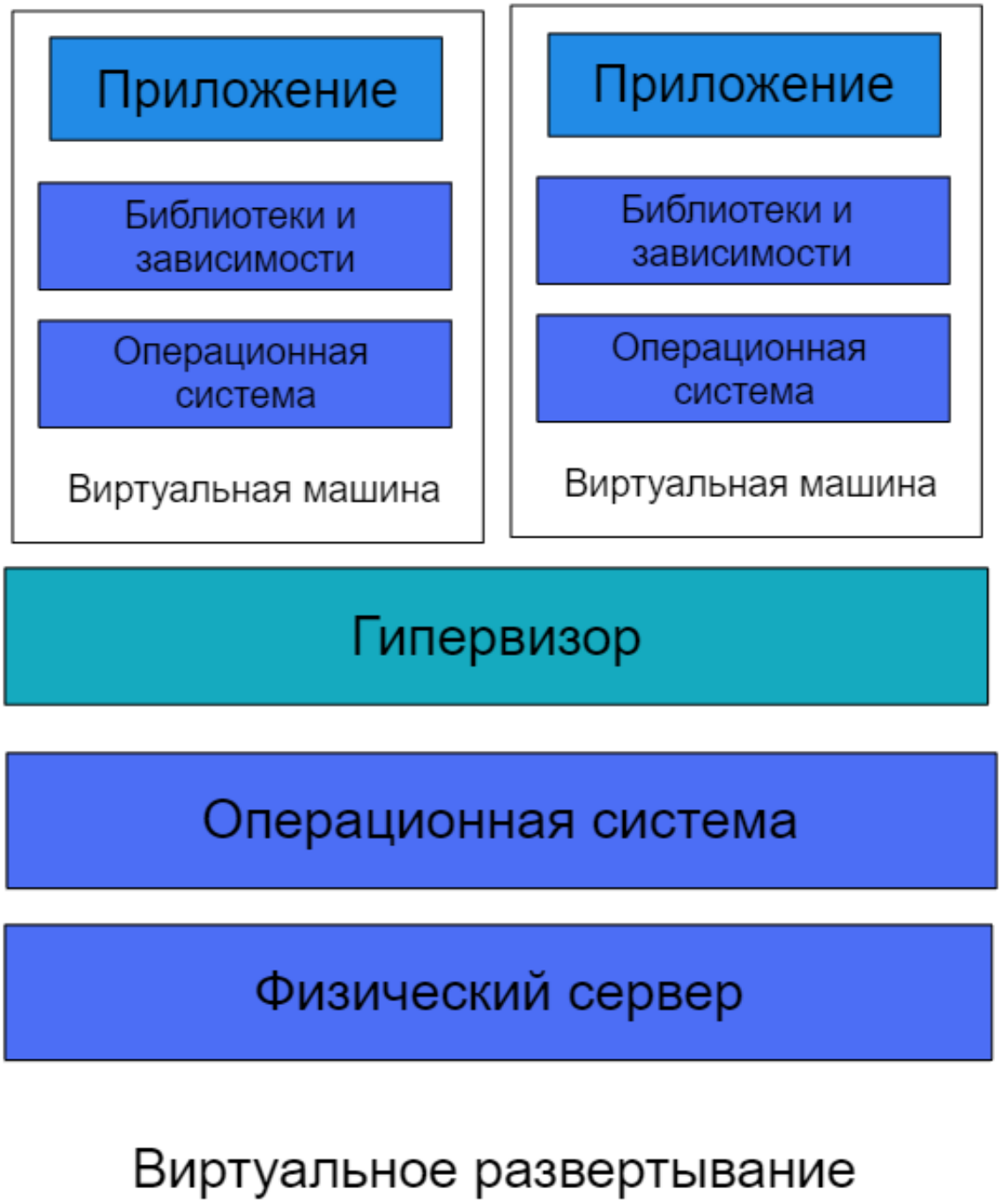


Рисунок 2 – Виртуальное развертывание

С появлением контейнеризации и платформы Kubernetes произошел сдвиг в развертывании и управлении приложениями. Контейнеры обеспечивают изолированную среду для выполнения приложений, что позволяет легко перемещать и масштабировать их на нескольких хостах. Kubernetes предоставляет оркестрацию контейнеров, автоматическое масштабирование, управление высокой доступностью и другие возможности для развертывания и управления приложениями. Он позволяет декларативно описывать требуемое состояние системы и самостоятельно поддерживать это состояние, осуществляя мониторинг, автоматическое восстановление и масштабирование.

Комбинация контейнеризации и Kubernetes обеспечивает упрощенное развертывание, масштабирование и управление приложениями, улучшает отказоустойчивость и обеспечивает более эффективное использование ресурсов. Это делает Kubernetes платформой выбора для разработчиков, которые стремятся создавать современные, масштабируемые и надежные приложения. На рисунке 3 отображены все варианты развертывания приложений упомянутых ранее.

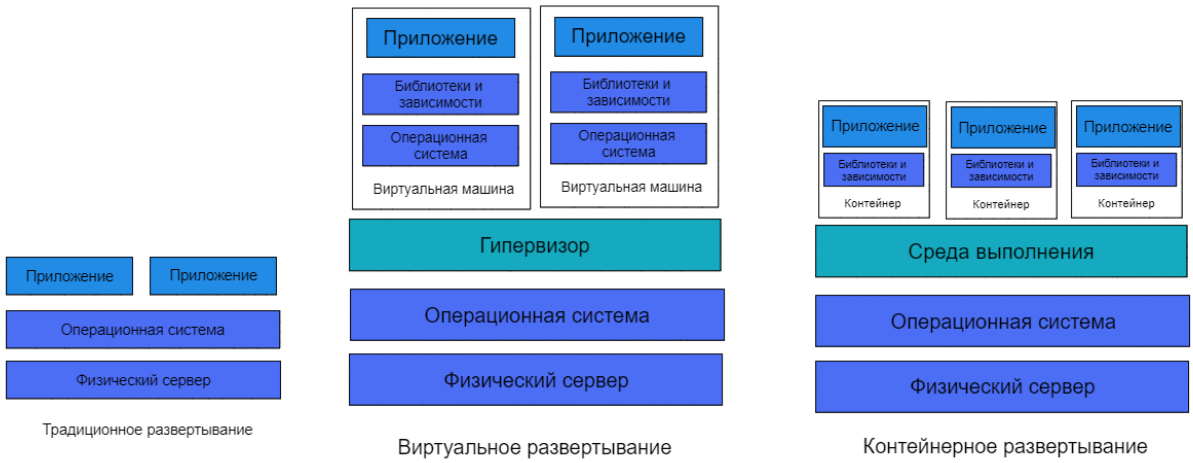


Рисунок 3 – Все варианты развертывания

Kubernetes является ключевой платформой для управления контейнеризованными приложениями. Он обеспечивает автоматизацию, декларативную настройку и оркестрацию контейнеров. Kubernetes позволяет эффективно управлять развертыванием, масштабированием и мониторингом приложений, обеспечивает высокую доступность и отказоустойчивость.

Выбор Kubernetes в качестве платформы для управления контей-

неризованными приложениями становится все более популярным среди разработчиков. Он упрощает развертывание, масштабирование и управление приложениями, предоставляет гибкость в выборе облачной платформы и операционной системы, а также обеспечивает высокую производительность и гибкость в разработке распределенных приложений.

Основные преимущества контейнеров:

1. **Автоматизация**: Kubernetes предоставляет возможность автоматизировать развертывание, масштабирование и управление контейнеризованными приложениями. Это позволяет сократить время и усилия, затрачиваемые на рутинные задачи, и обеспечивает более эффективное использование ресурсов.

2. **Декларативная настройка**: С помощью Kubernetes можно описать желаемое состояние системы в виде декларативных конфигурационных файлов. Кубернетес берет на себя задачу достижения и поддержания этого состояния, что упрощает управление и обеспечивает консистентность среды выполнения.

3. **Оркестрация**: Kubernetes обеспечивает динамическую оркестрацию контейнеров, что позволяет эффективно управлять развертыванием, масштабированием и управлением жизненным циклом приложений. Он автоматически распределяет рабочие нагрузки по узлам кластера, обеспечивает балансировку нагрузки и восстанавливает работу приложений после сбоев.

4. **Высокая доступность и отказоустойчивость**: Kubernetes предлагает механизмы для обеспечения высокой доступности приложений. Он автоматически перезапускает контейнеры или переносит их на другие доступные узлу кластера. Это позволяет обеспечить непрерывную работу приложений даже в случае отказа отдельных компонентов.

5. **Масштабируемость**: Kubernetes обладает возможностью горизонтального масштабирования, что позволяет увеличивать или уменьшать количество экземпляров приложений в зависимости от изменяющейся нагрузки. Это обеспечивает гибкость в управлении ресурсами и позволяет эффективно использовать вычислительные мощности кластера.

6. **Портативность**: Контейнеры, управляемые Kubernetes, являются переносимыми между различными облачными платформами и операционными системами. Это дает разработчикам свободу выбора и упрощает

перенос приложений между различными средами.

7. **Экосистема**: Kubernetes имеет широкую и активную экосистему инструментов и расширений, которые обеспечивают дополнительные возможности и интеграции. Существуют различные плагины, инструменты мониторинга, логирования, управления сетью и многое другое, которые расширяют функциональность и способности Kubernetes.

8. **Сообщество и поддержка**: Kubernetes имеет активное и развитое сообщество разработчиков и пользователей. Это обеспечивает доступ к обширным ресурсам документации, руководствам и форумам поддержки, что упрощает изучение и использование платформы.

Для выполнения поставленной задачи необходимо ознакомиться с декларативной конфигурацией кластера Kubernetes, включая его основные объекты. Декларативная конфигурация позволяет определить различные ресурсы и контроллеры ресурсов, которые управляют работой в кластере. Далее перечислены некоторые из основных объектов:

1. **Service**. Сервис предоставляет абстракцию для доступа к работающим приложениям внутри кластера. Он определяет стабильный конечный точки для связи с подами (pods), предоставляя механизм обнаружения сервиса и балансировки нагрузки.

2. **Deployment**. Развертывание (Deployment) определяет описательную модель для создания и масштабирования реплик подов (replica sets). Он позволяет управлять жизненным циклом приложения, включая создание, обновление и масштабирование.

3. **ReplicaSet**. ReplicaSet представляет собой контроллер, который обеспечивает поддержание желаемого количества работающих реплик подов. Он позволяет масштабировать и управлять репликами приложений в соответствии с определенными параметрами.

4. **ConfigMap**. ConfigMap используется для хранения конфигурационных данных, которые могут быть доступны из контейнеров внутри подов. Он позволяет централизованно управлять конфигурацией приложений и разделить ее от кода.

5. **Pod**. Под (Pod) является наименьшей единицей развертывания в Kubernetes. Он представляет собой группу из одного или нескольких контейнеров, которые разделяют сетевое пространство, хранилище и другие

ресурсы. Поды создаются и управляются контроллерами, такими как ReplicaSet или Deployment.

Для работы с кластером Kubernetes на самом низком уровне используется консольная утилита kubectl. Она предоставляет множество команд, которые обращаются к API контроллера кластера, расположенного на главном узле Kubernetes. Эти команды позволяют взаимодействовать с различными объектами, управлять ресурсами и получать информацию о состоянии кластера. Ознакомление с декларативной конфигурацией и использование утилиты kubectl являются основой для работы с Kubernetes для прикладного разработчика [3].

1.2 Функцинал утилиты для отладки приложений Telepresence

С активным переходом на использование кластеров Kubernetes у разработчиков возникают дополнительные трудности: проблемы отладки и разработки приложений в удаленном кластере. Telepresence является относительно новой технологией, предоставляющей разработчиками возможность выполнять отладку и разработку локально развернутых приложений, подключенных к кластеру Kubernetes за счет создания двунаправленных прокси соединений между кластером и локальной рабочей станцией разработчика[4].

Не углублясь в детали реализации, принцип работы утилиты отображен на рисунке 4.

1. Консольная утилита располагается на рабочей станции разработчика и играет роль клиента.

2. Предварительно в кластере устанавливается менеджер трафика от Telepresence, устанавливающий прокси соединение по запросу клиента.

3. Локальный процесс приложения запускается на рабочей станции разработчика и общается с другими процессами, развернутывами в кластере через прокси соединение в прозрачном режиме при помощи менеджера трафика.

4. Для того чтобы остановить общение локального процесса с приложениями в кластере достаточно прекратить существование прокси соединения через вызов команды клиента Telepresence. [5].

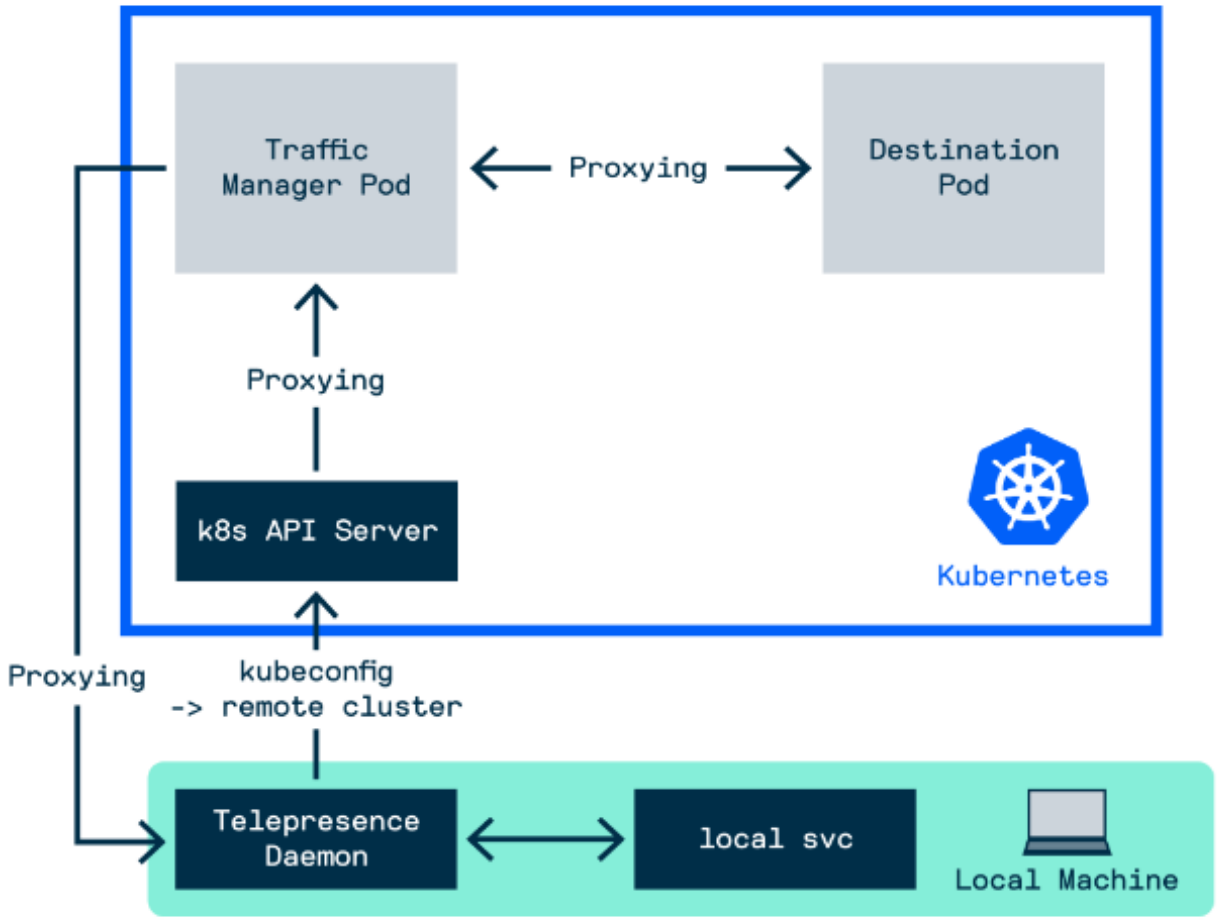


Рисунок 4 – Развертывание Telepresence в Kubernetes и на машине разработчика

Для отображения преимущества использования инструмента отладки Telepresence приведем две блок-схемы отоброжающие различный подход к отладке и тестированию приложений, развернутых в кластере Kubernetes, разработчиком.

Как отображено на рисунке 5, принципиальная разница в процессах заключается в отсутствии длительных этапов для доставки изменений в кластер Kubernetes. Сборка и публикация новой версии, а также процесс развертывания в кластер являются необходимыми этапа, когда выполняется обычная доставка изменений, однако, в рамках процесса отладки и тестирования, разработчика интересует скорость получения обратной связи его изменений в коде приложения. Локальная сборка и запуск приложения значительно проще сборки и публикации первого варианта. Дополнительно к этому, при локальном запуске приложения есть возможность использовать

инструменты отладки и точек останова, для лучшего обозревания работы приложения и устранения дефектов. Telepresence позволит разместить локальную процесс в кластере Kubernetes для отладки в обход основного процесса публикации новых версий.

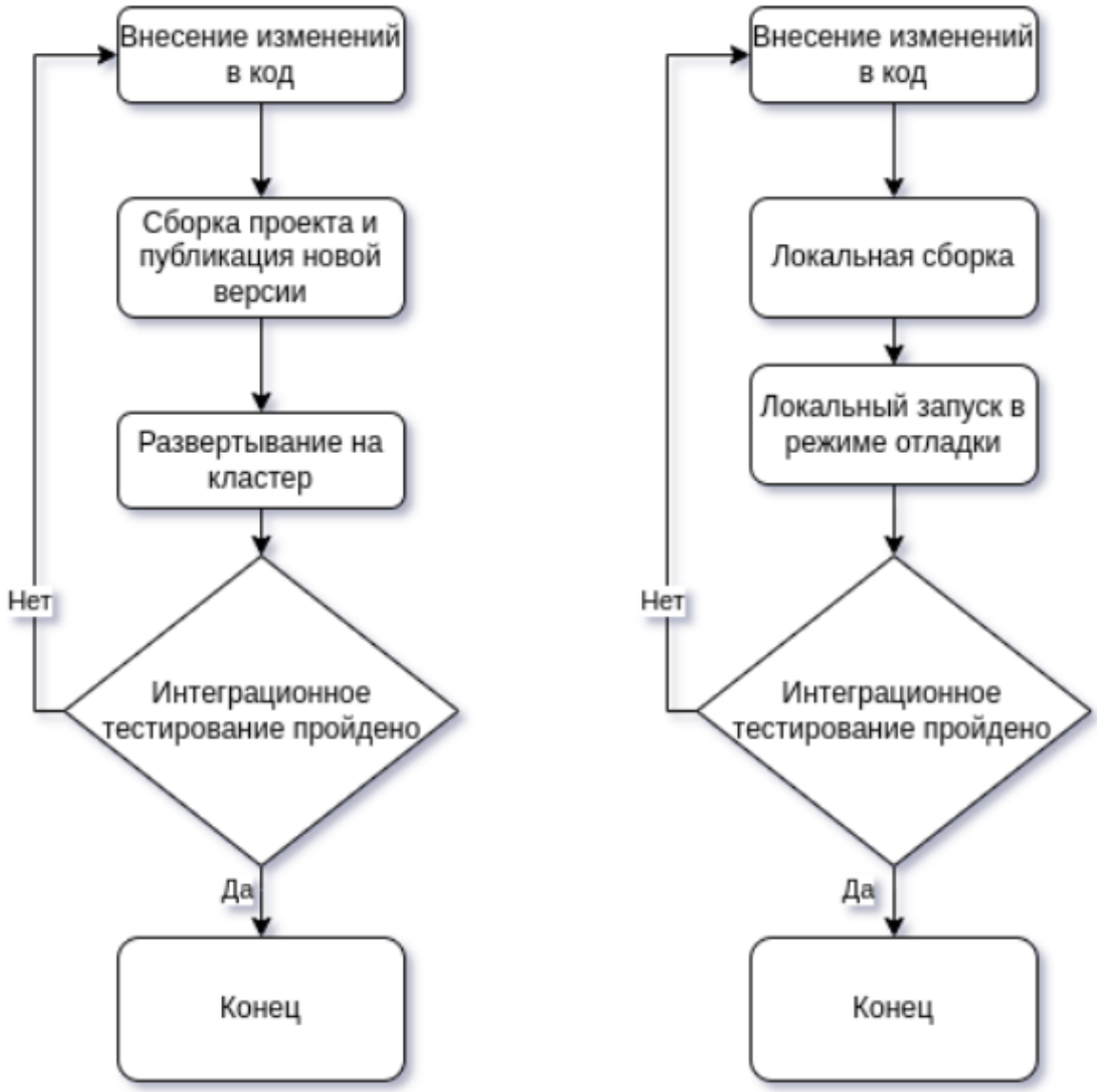


Рисунок 5 – Схема справа отображает процесс отладки с Telepresence, слева - без инструмента

1.3 Сравнение аналогов

Потенциальным конкурентом проектируемого расширения является платформа Ambassador Cloud. Платформа предоставляет множество возможностей для непрерывной отладки приложений, в том числе и через Telepresence. Однако, имеет ряд недостатков и ограничений, делая его не

самым хорошим выбором среди начинающих разработчиков. Основные преимущества сервиса:

1. Облачная платформа. Соответственно сложность развертывания перекладывается на внешние лица.

2. Множество дополнительных функциональных вариантов использования, помимо Telepresence. В том числе потенциальная интеграция с другими сервисами и автоматическое тестирование.

Недостатки платформы относительно проектируемого расширения:

1. Платное, проприетарное решение. Функционал Telepresence является платной услугой данной платформы.

2. Облачное решение. Проблема с контролем для компаний, которые серьезно относятся к информационной безопасности. Платформы и сервисы, которые нельзя развернуть на собственных вычислительных хостах, зачастую отбрасываются политиками компании, ради защиты информации и безопасности.

3. Перегруженность. Платформа выполняет множество задач, которые подходят не для каждого проекта или продукта.

Относительно данного конкурента, расширение для Telepresence позиционирует себя как:

1. Решение для одной конкретной задачи. Задача расширения - предоставлять пользователю функционал инициализации двунаправленных прокси соединений к кластеру.

2. Бесплатное решение с открытым исходным кодом.

3. Простая интеграция. Развертывание расширения ограничено установкой в магазине расширений внутри интеграционной среды разработки.

Таким образом, решение предлагает значительные преимущества по сравнению с платным облачным решением от Ambassador Cloud.

1.4 Платформа разработки в виде IDE

Разрабатывать обособленное приложение для отдельного пользовательского интерфейса нецелесообразно по затратам и доступным ресурсам, поэтому рассматривается платформа для реализации в виде расширений интеграционной среды разработки. В данном контексте, интегрируемой средой разработки является такое программное обеспечение, которое

предоставляет пользователю инструменты и интерфейс для разработки программного обеспечения. На текущий момент существует множество различных сред разработки для различных языков программирования, фреймворков и технологий.

Для того, чтобы определить нужную интегрируемую среду разработки требуется наличие определенных критериев:

1. Поддержка работы с кластером Kubernetes. Среда разработки должна предоставлять возможность обозревать ресурсы кластера и выполнять над ними базовые операции.

2. Поддержка разработки встраивамых модулей, расширений. Предоставляемый разработчиками IDE интерфейс для разработки расширений должен быть хорошо задокументирован и понятен для исполнителя.

3. Популярность и доступность для разработчиков. Интегрируемая среда разработки желательно должна быть бесплатной для использования и быть популярной среди разработчиков.

Путем дополнительного исследования и использования накопленного опыта были отобраны несколько потенциальных кандидатов:

1. Visual Studio Code. Кроссплатформенное решение, поддерживаемое компанией Microsoft. Подходит для огромного множества языков программирования и инструментов и позиционируется как универсальная IDE.

2. JetBrains IDE’s. Каждый отдельный продукт компании JetBrains (Goland, IntelliJ IDEA, Pycharm, Rider, WebStorm) предназначен для отдельных фреймворков и языков программирования, но обладает зачастую идентичными расширениями, интерфейсом и горячими клавишами. Является отличным выбором для профессиональной разработки. Разработка расширений предполагает работу с Java и множеством абстракций объектного-ориентированного программирования [6].

3. Open Lens. Является обозревателем кластера Kubernetes с множеством функциональных возможностей для работы с ресурсами внутри кластера. Функционал ограничен работой с ресурсами кластера и не предназначен для написания программного кода.

Каждый из этих кандидатов имеет свои особенности и преимущества. Выбор конкретной интегрированной среды разработки зависит от предпочтений разработчика и требований проекта.

Проведем небольшую аналитку популярности первых двух IDE за последний год. На рисунке 6 отображена динамика запросов среды Visual Studio Code в России по запросам через поисковый движок Google. Как видно, держится стабильно.

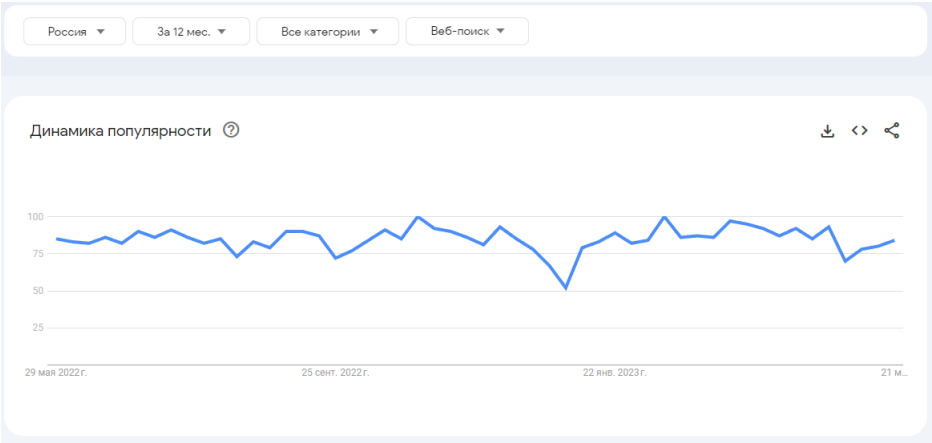


Рисунок 6 – Динамика популярности VS Code

Для сравнения, добавим дополнительно два продукта компании JetBrains: GoLand - среду разработки на Golang и IntelliJ IDEA - среду разработки для Java и Kotlin. На рисунке 7 заметим, что популярность у продуктов JetBrains значительно ниже по отдельности и VS Code определяется как лидер по популярности. Это обусловленно прежде всего его универсальностью, так как среда расчитана на разработку со множеством инструментов, языков программирования и технологий.

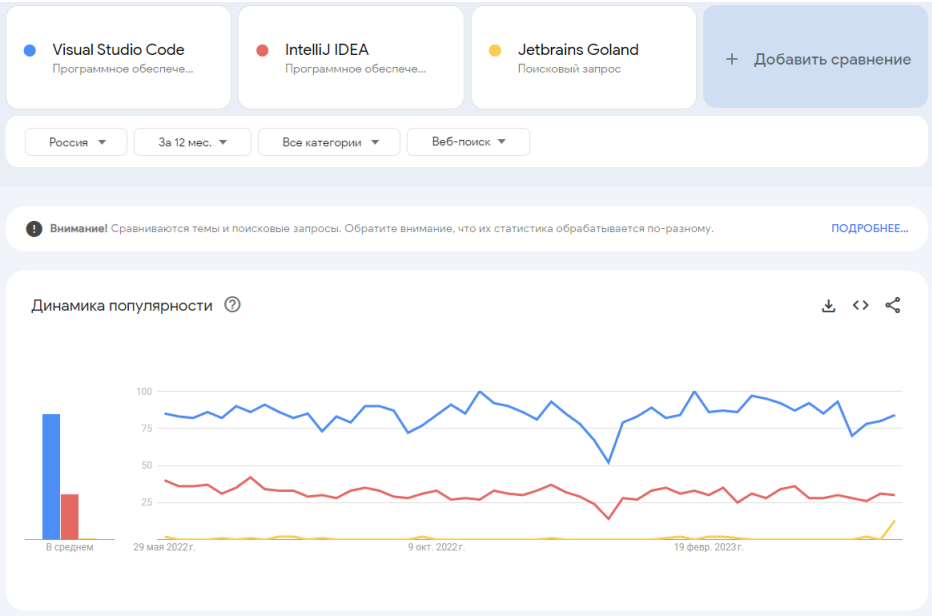


Рисунок 7 – Популярность среди других сред разработки

Беря во внимание перечисленные критерии, аргументы и предпочтения разработчика, предлагается выбрать Visual Studio Code в качестве интегрированной среды разработки для реализации расширения. Visual Studio Code обладает широким спектром поддерживаемых языков программирования, предоставляет удобный интерфейс разработки расширений на основе TypeScript и имеет активное сообщество разработчиков. Более того, Visual Studio Code является кроссплатформенным решением, что позволяет разрабатывать расширение на различных операционных системах.

VS Code обладает современным интерфейсом и получает регулярные обновления и исправления. На рисунке 8 изборажена среда разработки Visual Studio Code с открытым проектом. Для разработки расширений критически важно, чтобы API, предоставляемый разработчикам расширений было устойчиво и не подвергалось ломающим изменениям - данная среда разработки поддерживается Microsoft и сообществом, которое бережно относится к добавлению нового функционала и без ломающих изменений.

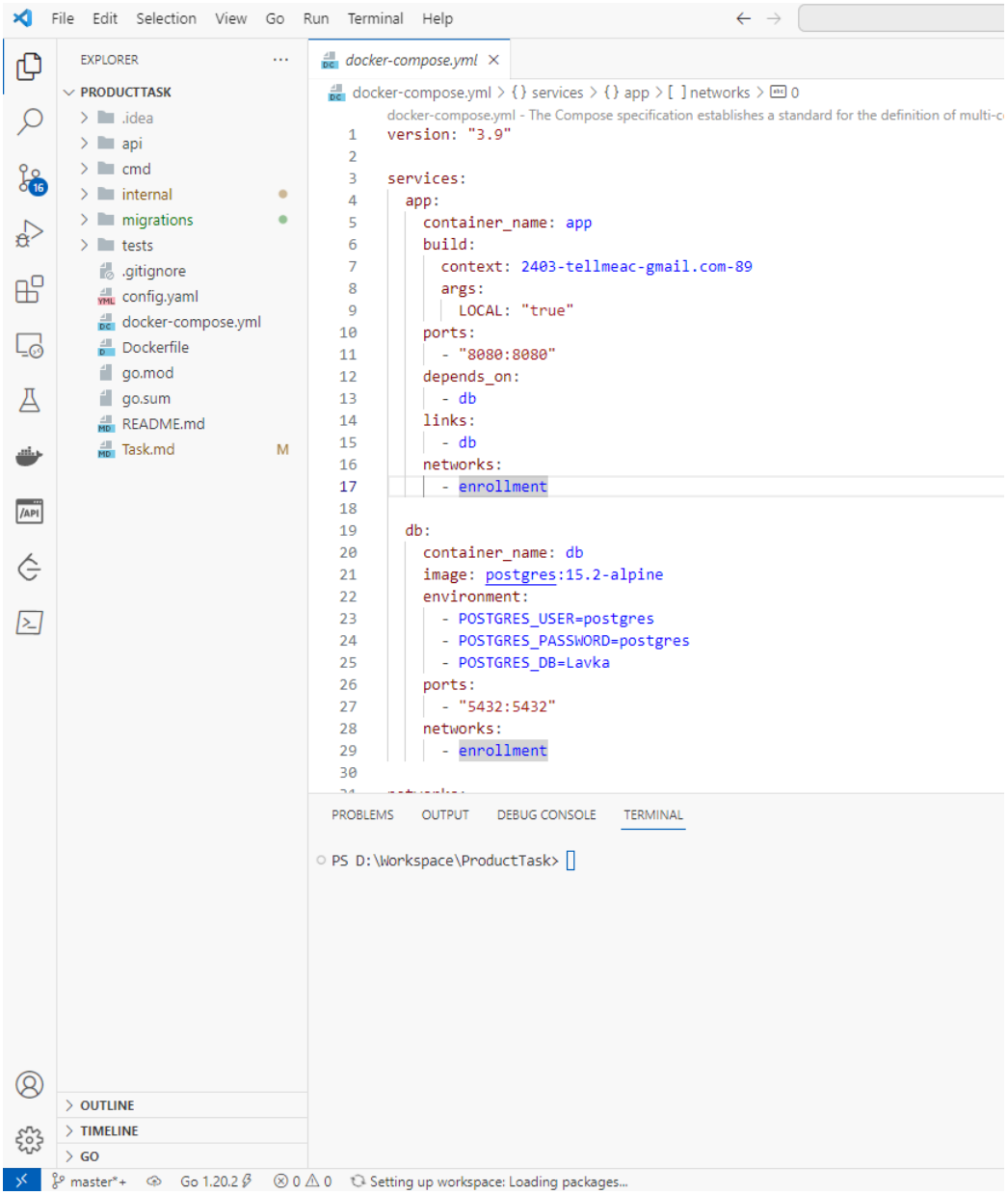


Рисунок 8 – Интеграционная среда разработки VS Code

2 Проектирование и архитектура

Расширение будет разработано для Visual Studio Code. Результат разработки, встраиваемый модуль, должен поддерживать пользовательский интерфейс, предоставляющий необходимый функционал, предоставляемый утилитой Telepresence:

1. Просмотр доступных сервисов.
2. Установление нового соединения.
3. Закрытие активного соединения.
4. Конфигурирование подключения утилиты Telepresence к кластеру Kubernetes.

На основе документации по разработке расширений для VS Code [7] необходимо определить перечень предоставляемых операций и объектов, реализуемых модулем в манифесте расширения.

В контексте расширения для Telepresence в VS Code, следующие команды должны быть задекларированы:

1. OpenConnection: Эта команда отвечает за открытие соединения. При вызове команды, расширение может использовать соответствующий API Telepresence для установки связи между локальной машиной разработчика и кластером Kubernetes. Это позволяет разработчикам локально разрабатывать и отлаживать приложения, взаимодействуя с сервисами внутри кластера.

2. CloseConnection: Данная команда предназначена для закрытия активного соединения. При вызове команды, расширение может использовать соответствующий API Telepresence для завершения соединения между локальной машиной разработчика и кластером Kubernetes. Это особенно полезно, когда разработчик закончил работу с удаленным кластером и хочет прекратить проксирование трафика.

Для конфигурирования доступных подключений к кластеру Kubernetes необходимо ввести конфигурационный параметр, который указывает на определенный кластер Kubernetes и желаемую область имен - namespace.

Данный параметр обычно называется kubeconfig и содержит информацию о кластере, такую как адрес API сервера, учетные данные и контекст подключения. Путем указания конкретного файла конфигурации или используя переменные окружения, Telepresence может использовать эту

информацию для установки соединения с кластером Kubernetes.

Конфигурационный параметр kubeconfig позволяет разработчикам выбирать нужный кластер и область имен, с которыми они хотят работать при разработке и отладке своего расширения с использованием Telepresence. Это обеспечивает гибкость и возможность работы с различными кластерами Kubernetes в разных средах разработки.

Для интеграции объектов просмотра (View) в Telepresence необходимо предоставить возможность вызова этих объектов из контекстного меню, доступного при выборе соответствующего сервиса в обозревателе кластера Kubernetes. Кроме того, необходимо интегрировать команды Telepresence в графический интерфейс интегрированной среды разработки. Для этой цели можно использовать контроллер в виде дополнительного контекстного меню, который будет обеспечивать доступ к соответствующим функциям Telepresence. В качестве примера уже существуют реализации таких контроллеров в виде отдельных расширений, которые могут быть использованы для данной интеграции [8].

Опираясь на подход к реализации модуля для обозревания ресурсов кластера Kubernetes [9] возможно спроектировать простой и расширямый дизайн модуля как отображено на рисунке 9:

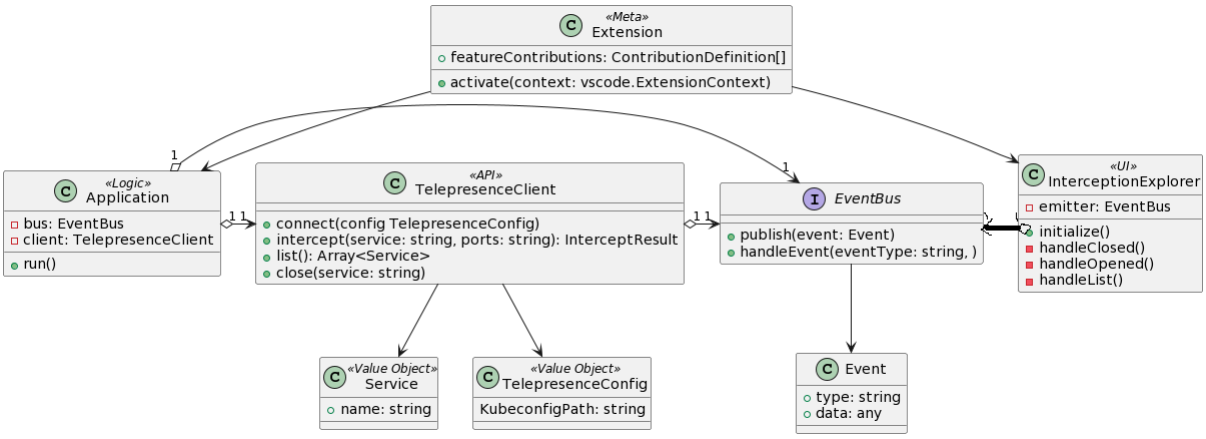


Рисунок 9 – Схема проекта

В кратце разберем, за что отвечает каждый объект:

1. Extension - содержит в себе информацию про доступные для интегрированной среды команды, опции контекстного меню и настройки конфигурации.

2. EventBus - интерфейс для взаимодействия между различными

компонентами через сообщения (события).

3. Event - модель события для интерфейса EventBus.

4. Service - модель сервиса, идентифицируется именем.

5. TelepresenceConfig - объект конфигурации для подключения к Telepresence. Kubeconfig - файл, содержащищй в себе все необходимые параметры для подключения к кластеру Kubernetes [1].

6. TelepresenceClient - клиент-оболочка к утилите отладки Telepresence.

7. InterceptionExplorer - объект, отделяющий слой взаимодействия с пользовательским интерфейсом интеграционной среды разработки и логикой модуля.

8. Application - объект, содержащий в себе логику обработки и публикации событий для InterceptionExplorer.

В рамках расширения Telepresence для VS Code существуют два компонента, которые взаимодействуют через локальные сообщения и события: классы Application и InterceptionExplorer.

Класс Application отвечает за логику расширения. Он поддерживает реализацию операций, связанных с командами расширения, которые были определены ранее. Класс Application также отвечает за обработку изменений конфигурации, связанных с подключением к кластеру Kubernetes и управлением активными соединениями. Внутри класса Application реализованы функции, которые обрабатывают запросы пользователя и взаимодействуют с соответствующими API Telepresence для установки, закрытия соединений и т.д.

Класс InterceptionExplorer отвечает за отображение элементов пользовательского интерфейса и взаимодействие с пользователем. Он обеспечивает представление активных соединений, информацию о них и возможность управления ими через интерфейс расширения. InterceptionExplorer публикует события при взаимодействии пользователя с элементами взаимодействия - опциями контекстного меню, вызов команд или смена настроек расширения.

Такое взаимодействие между Application и InterceptionExplorer, достигаемое через локальные сообщения(события), помогает снизить связность между компонентами и повысить гибкость кода. Компоненты могут взаимодействовать независимо друг от друга, а изменения в одном компоненте не требуют непосредственных изменений в другом компоненте,

что упрощает поддержку и развитие расширения. [10].

Сценарии использования команд расширения Telepresence для VS Code:

1. **Open Connection** - открыть соединение:

– Пользователь выбирает команду *Open Connection* из меню расширения или использует соответствующую комбинацию клавиш.

– Расширение Telepresence открывает окно диалога, где пользователь может указать параметры подключения к кластеру Kubernetes, такие как порт и дополнительные переменные окружения.

– Расширение использует полученные параметры для настройки соединения с кластером Kubernetes через Telepresence API.

– После установки соединения расширение отображает подключенные сервисы и другую информацию в пользовательском интерфейсе или через уведомления.

2. **Close Connection** - закрыть активное соединение:

– Пользователь выбирает команду *Close Connection* из меню расширения или использует соответствующую комбинацию клавиш.

– Расширение Telepresence проверяет наличие активного соединения.

– Если соединение с кластером Kubernetes установлено, расширение закрывает соединение и освобождает все ресурсы, связанные с подключением.

3. **Show Available Service** - показать активные сервисы:

– Пользователь выбирает команду *Show Available Service* из меню расширения или использует соответствующую комбинацию клавиш.

– Расширение Telepresence отображает список всех активных соединений с кластером Kubernetes.

– Для каждого соединения отображается информация о подключенных сервисах, статусе соединения и других деталях.

4. **Setup Settings** *kubeconfig* в VS Code:

– Пользователь открывает настройки VS Code.

– В настройках расширения Telepresence указывается путь к *kubeconfig* файлу, который содержит конфигурацию подключения к кластеру Kubernetes.

– После указания пути к *kubeconfig* файлу, расширение Telepresence использует эту конфигурацию для установки соединения с кластером Kubernetes при выполнении операций открытия и закрытия соединений.

3 Разработка и тестирование

3.1 Подготовка окружения

Для начала разработки необходимо подготовить окружение на рабочей станции разработчика. В частности, необходимо установить Visual Studio Code и NodeJS. Для тестирования и отладки необходимо иметь доступ к облачному кластеру Kubernetes, содержащему несколько тестовых сервисов, над которыми можно свободно проводить любые манипуляции. Кластер Kubernetes можно самостоятельно развернуть в виде одной машины - рабочей станции разработчика. Для отладки и разработки, предпочетают использовать различиные облегченные реализации кластера: Kind, k3s или Minikube[2]. Был выбран Minikube по личному предпочтению разработчика. Установка kubectl и helm является обязательным этапом. Вся настройка окружения сводится к загрузке .deb пакетов и выгрузке бинарных файлов через утилиту curl.

3.2 Инициализация проекта

Согласно официальной документации VS Code [7]. Инициализация расширения для актуальной версии интегрированной среды разработки происходит через генератор шаблона проекта-расширения **yo code**. Установка данного генератора происходит через менеджер Node пакетов npm [11].

Полученная структура проекта отображена на рисунке 10.

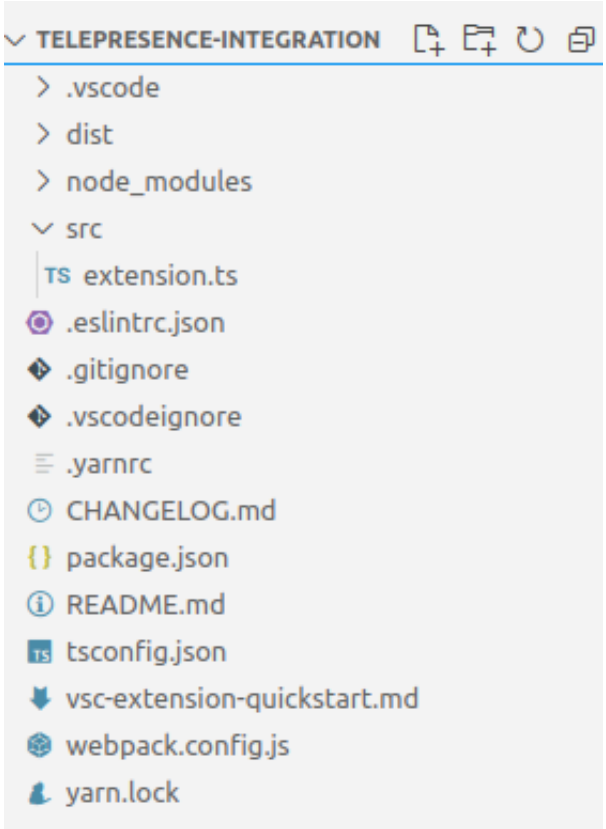


Рисунок 10 – Изначальная структура проекта

Для проверки работоспособности Telepresence и реализуемого расширения, необходимо иметь развернутым на кластере сервисы, взаимодействующие между собой. Для данного тестирвоания был выбран пример от проекта OpenTelemetry в виде helm chart. Установка сервисов происходит следующим образом, как отображено на иллюстрации 11:

Install using Helm (recommended)

Add OpenTelemetry Helm repository:

```
helm repo add open-telemetry https://open-telemetry.github.io/opentelemetry-helm-charts
```

To install the chart with the release name my-otel-demo, run the following command:

```
helm install my-otel-demo open-telemetry/opentelemetry-demo
```

Рисунок 11 – Установка helm chart на локальный кластер Kubernetes

Получаем, как показано на иллюстрации 12 - множество сервисов одного проекта, работающих обособлено в одном кластере. Один из таких сервисов будет заменен на локальную версию, расположенную на рабочей станции разработчика с целью отладки и тестирования.

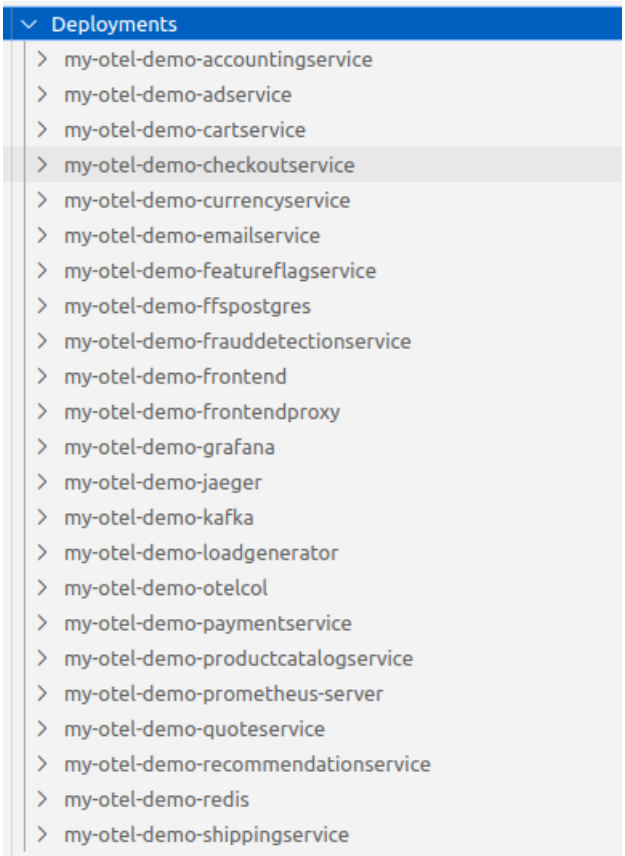


Рисунок 12 – Список развернутых сервисов в кластере Kubernetes (Minikube) для тестирования

3.3 Разработка

Как было заявлено при проектировании, необходимо объявить несколько команд и параметр конфигурации расширения. Согласно документации VS Code [7], это можно сделать через **package.json**, который является своего рода манифестом расширения. Кроме списка зависимостей, содержит такую информацию как: название, версию и описание расширения, включая определения **featureContributions** - объекты, которые предоставляет расширение. Данный манифест описывается в формате json. И определение объектов приведено на рисунке 13.

```
    "contributes": {
    "commands": [
      {
        "command": "telepresence-integration.listServices",
        "title": "List available services for interception"
      },
      {
        "command": "telepresence-integration.connectDefault",
        "title": "Connect with Default Context"
      },
      {
        "command": "telepresence-integration.intercept",
        "title": "Intercept with Telepresence"
      }
    ],
    "configuration": {
      "title": "Telepresence Integration",
      "properties": {
        "telepresence.KubeConfig": {
          "type": "string",
          "default": "~/.kube/config",
          "description": "Kubeconfig file"
        }
      }
    }
  },
```

Рисунок 13 – Объявление команд и объекта конфигурации

Реализуем TelepresenceClient согласно архитектуре как указано на иллюстрации 14. Стоит отметить, что утилита не предоставляет дополнительных флагов для получения ответа в одном из удобных форматов данных - json или yml. Парсинг значений подразумевает использование тяжелых регулярных выражений. Потери скорости незначительны.

```
class TelepresenceClient {
        private executablePath: string = "telepresence";

        public connectDefaultContext() {
                const result = exec(`${this.executablePath} connect`)

                console.log(`Telepresence connect:\n ${result.exitCode}: ${result.stdout?.read()}`)
        }

        public list(): Array<Service> {
                const result = exec(`${this.executablePath} list`)
                if (result.exitCode != 0) {
                        throw Error("Failed to get list of available services");
                }

                console.log(`Telepresence list:\n ${result.exitCode}: ${result.stdout?.read()}`)

                return this.parseList(result);
        }

        public installTrafficManager() {
                const result = exec(`${this.executablePath} helm install`)
                if (result.exitCode != 0) {
                        throw Error("Failed to install traffic manager");
                }

                console.log(`Telepresence install traffic manager:\n ${result.exitCode}: ${result.stdout?.read()}`)
        }

        public intercept(service: string, port: string): InterceptResponse {
                const result = exec(`${this.executablePath} intercept ${service} --port ${port}`)
                if (result.exitCode != 0) {
                        throw Error("Failed to intercept");
                }

                console.log(`Telepresence intercept: ${result.exitCode}:\n ${result.stdout?.read()}`)


                return this.parseIntercept(result);
        }
    // ...
}
```

Рисунок 14 – Реализация Telepresence Client

Интеграция Telepresence в виде графического интерфейса за счет дополнительных объектов управления - опции в контекстном списке меню. API Visual Studio Code предоставляет такой функционал. Команда intercept выполняется над ресурсом Service в кластере Kubernetes. Значит, необходимо предоставлять опцию в контекстном меню при выборе определенного сервиса, ресурс кластера, и не предлагать в иных случаях. Объявить доступ к данном команде необходимо через вышеупомянутый манифест - package.json как показано на рисунке 15.

```
"menus": {
      "editor/title": [
        {
          "when": ""when": "view == extension.vsKubernetesExplorer && viewItem =~ /vsKubernetes\\.resource\\.service/i"",
          "command": "telepresense-integration.intercept",
          "group": "1",
        }
      ]
    }
```

Рисунок 15 – Определение опции для контекстного меню сервиса в Kubernetes

Реализуем команду контекстного меню в InterceptionExplorer, определим метод, выполняющий обработку входных параметров в событие,

публикуемое в шину сообщений для последующей обработки экземпляра класса Application как показано на рисунке 16. InterceptionEvent содержит все необходимые параметры для инициализации прокси соединения утилитой Telepresence, так как подключение к кластеру уже выполнено на раннем этапе. Определим интерфейс EventBus и его реализацию через пакет ts-event-bus. Использовние класса EventBus предполагает сокрытую асинхронную обработку данных, поддерживаемую в TypeScript [12].

```
interface EventBus {
        publish: (event: any) => void;
        handleEvent: (eventType: string, handler: (event: Event) => void) => void;
}

interface ExplorerContext {
        Namespace: string;
        Service?: string;
}

class InterceptionExplorer {
        private emitter: EventBus

        constructor() {
                this.emitter = new LocalEventBus();
        }

        public handleIntercept(ctx: ExplorerContext) {
                // Прочитаем событие из контекста
                const event = this.makeInterceptRequest(ctx)

                // Опубликуем событие через шину сообщениц.
                this.emitter.publish(event)
        }
```

Рисунок 16 – Реализация Interception Explorer

Реализация Application подразумевает ожидание событий в методе run и обработку отдельных событий в приватных методах через вызов клиента Telepresence API и чтение контекста кластера Kubernetes согласно схеме на рисунке 17.

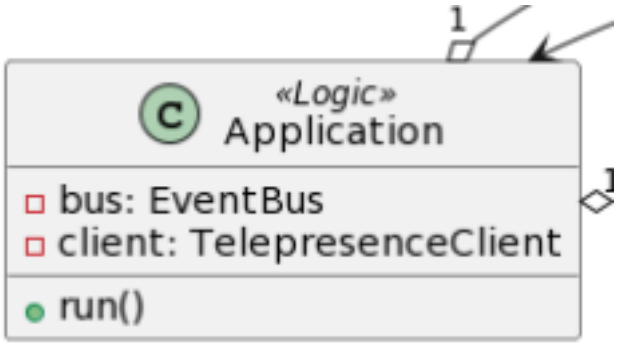


Рисунок 17 – Схема класса Application

3.4 Тестирование и отладка

В рамках отладки был использован Minikube, локальная версия кластера Kubernetes, упомянутая ранее и тестовый сервис, развернутый на нем через менеджеров чартов helm. При активации расширения были пройдены основные сценарии, описанные на этапе проектирования расширения.

Дополнительно, были написаны unit тесты, для тестирования вспомогательных функций.

На рисунке 18 отображен список с контекстом опции, в котором можно выбрать вызов команды над сервисом в кластере Kubernetes - Intercept with Telepresence:

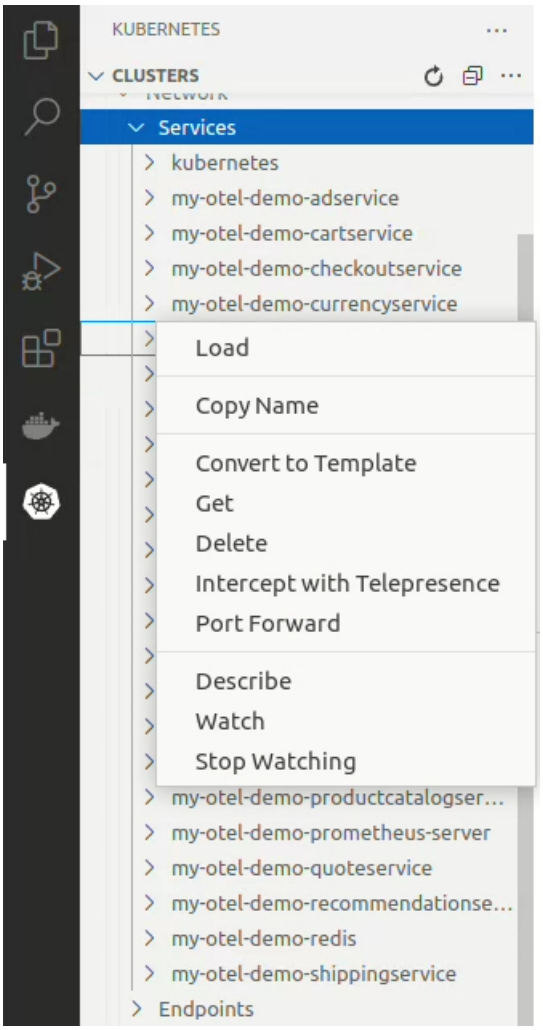


Рисунок 18 – Опция в контекстном меню от расширения

На рисунке 19 отображены настройки в виде определения пути к конфигу подклчюения к кластеру Kubernetes:

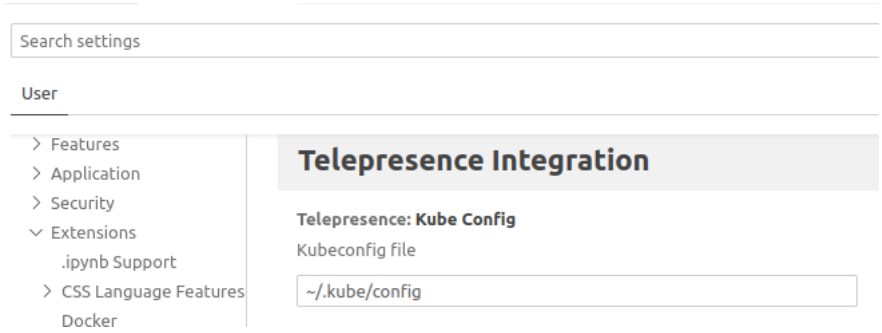


Рисунок 19 – Конфигурирование расширения

3.5 Публикация результата работы

Процесс публикации начинается с установки зависимостей через терминал:

```
$ npm install -g @vscode/vsce
```

Далее используется установленная утилита для публикации.

```
$ vsce package
# telepresence-integration.vsix сгенерирован
$ vsce publish
# LipatovAlexander.telepresence-integration опубилкованна VS Code Marketplace
```

При публикации расширения оно будет доступно для установки другим разработчикам.

3.6 Планирование дальнейших разработок и улучшений

Реализованный модуль предоставляет основной функционал, но стоит отметить количество возможных типов прокси соединений, которые может инициализирвать инструмент отладки Telepresence. В рамках дальнейшего развития расширения предлагаются следующие доработки и функционал:

1. Добавить поддержку работы с дополнительными типами подключения для отладки.
2. Реализовать дополнительные компоненты для мониторинга процесса в виде статуса.
3. Качественная обработка исключений для самостоятельной диагностики конечным пользователем.

4. Реализовать дополнительное окно для отображения состояния в работе клиента Telepresence.

5. Предлагать пользователю дополнительное окно для объявления переменных окружения локального приложения.

6. Подготовить детальную документацию по проекту.

7. Подготовить вводные материалы по использованию.

ЗАКЛЮЧЕНИЕ

В результате работы было спроектировано и реализовано расширение для интеграции инструмента локальной отладки приложений, развернутых в облачном кластере Kubernetes в виде пользовательского интерфейса. Была проведена первичная отладка и тестирование полученного продукта. Составлен перечень доработок функционала, который предстоит реализовать в рамках развития данного проекта. Готовое расширение является входной точкой для начинающих разработчиков серверных приложений, расположенных в облачном кластере и использование реализованного расширения позволяет упростить процесс отладки приложений, развернутых в облачном кластере Kubernetes.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Документация Kubernetes - https://kubernetes.io/ru/docs/home (дата обращения 05.02.2023)
2. Lussa M. (2017) Kubernetes in Action
3. Buchanan, S., Rangama, J., Bellavance, N. (2020). kubectl Overview. In: Introducing Azure Kubernetes Service
4. Документация Telepresence https://www.getambassador.io/docs/telepresence/latest/howtos/intercepts (дата обращения 06.02.2023)
5. Kubernetes tips & tricks: о локальной разработке и Telepresence - https://habr.com/ru/companies/flant/articles/446788 (дата обращения 19.03.2023)
6. Разработка расширений Jetbrains - https://plugins.jetbrains.com/docs/intellij/developing-plugins.html (дата обращения 23.03.2023)
7. Документация расширений VS Code - https://code.visualstudio.com/api (дата обращения 23.03.2023)
8. Пример реализации кнопки как компоненты - https://github.com/seunlanlege/vscode-action-buttons (дата обращения 01.04.2023)
9. Расширение VS Code для обозревания кластера Kubernetes - https://github.com/vscode-kubernetes-tools/vscode-kubernetes-tools (дата обращения 03.04.2023)
10. Гамма Эрих, Хелм Ричард Паттерны объектно-ориентированного проектирования (2022)
11. Документация NodeJS - https://nodejs.org/en/docs (дата обращения 14.04.2023)
12. Документация языка программирования TypeScript - https://www.typescriptlang.org/docs (дата обращения 04.04.2023)

АНТИПЛАГИАТ
ОБНАРУЖЕНИЕ ЗАИМСТВОВАНИЙ

Отчет о проверке на заимствования №1

Автор: Липатов Александр Сергеевич
Проверяющий: Касаткина Анна Наильевна
Организация: Томский Государственный Университет

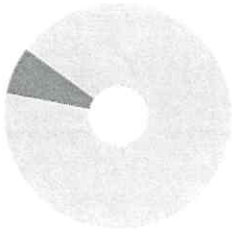
Отчет предоставлен сервисом «Антиплагиат» - http://tsu.antiplagiat.ru

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 287
Начало загрузки: 06.06.2023 06:14:15
Длительность загрузки: 00:00:07
Имя исходного файла: ВКР.pdf
Название документа: ВКР
Размер текста: 42 кБ
Символов в тексте: 42964
Слов в тексте: 4745
Число предложений: 257

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Начало проверки: 06.06.2023 06:14:22
Длительность проверки: 00:01:33
Комментарии: не указано
Поиск с учетом редактирования: да
Проверенные разделы: основная часть с. 2-3,5-37
Модули поиска: ИПС Адилет, Сводная коллекция ЭБС, Интернет Плюс*, Сводная коллекция РГБ, Цитирование, Переводные заимствования (RuEn), Переводные заимствования по eLIBRARY.RU (EnRu), Переводные заимствования по коллекции Гарант: аналитика, Переводные заимствования по коллекции Интернет в английском сегменте, Переводные заимствования по Интернету (EnRu), Переводные заимствования по коллекции Интернет в русском сегменте, Переводные заимствования издательства Wiley , eLIBRARY.RU, СПС ГАРАНТ: аналитика, СПС ГАРАНТ: нормативно-правовая документация, Медицина, Диссертации НББ, Коллекция НБУ, Перефразирования по eLIBRARY.RU, Перефразирования по СПС ГАРАНТ: аналитика, Перефразирования по Интернету, Перефразирования по Интернету (EN), Перефразированные заимствования по коллекции Интернет в английском сегменте, Перефразированные заимствования по коллекции Интернет в русском сегменте, Перефразирования по коллекции издательства Wiley , Патенты СССР, РФ, СНГ, СМИ России и СНГ, Модуль поиска "tsu", Издательство Wiley, Переводные заимствования



СОВПАДЕНИЯ	САМОЦИТИРОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
6,53%	0%	0%	93,47%

Совпадения — фрагменты проверяемого текста, полностью или частично сходные с найденными источниками, за исключением фрагментов, которые система отнесла к цитированию или самоцитированию. Показатель «Совпадения» – это доля фрагментов проверяемого текста, отнесенных к совпадениям, в общем объеме текста.

Самоцитирования — фрагменты проверяемого текста, совпадающие или почти совпадающие с фрагментом текста источника, автором или соавтором которого является автор проверяемого документа. Показатель «Самоцитирования» – это доля фрагментов текста, отнесенных к самоцитированию, в общем объеме текста.

Цитирования — фрагменты проверяемого текста, которые не являются авторскими, но которые система отнесла к корректно оформленным. К цитированиям относятся также шаблонные фразы; библиография; фрагменты текста, найденные модулем поиска «СПС Гарант: нормативно-правовая документация». Показатель «Цитирования» – это доля фрагментов проверяемого текста, отнесенных к цитированию, в общем объеме текста.

Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.

Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.

Оригинальный текст — фрагменты проверяемого текста, не обнаруженные ни в одном источнике и не отмеченные ни одним из модулей поиска. Показатель «Оригинальность» – это доля фрагментов проверяемого текста, отнесенных к оригинальному тексту, в общем объеме текста.

«Совпадения», «Цитирования», «Самоцитирования», «Оригинальность» являются отдельными показателями, отображаются в процентах и в сумме дают 100%, что соответствует полному тексту проверяемого документа.

Обращаем Ваше внимание, что система находит текстовые совпадения проверяемого документа с проиндексированными в системе источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности совпадений или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в тексте	Источник	Актуален на	Модуль поиска	Комментарии
[01]	3,39%	диплом.pdf	02 Июн 2022	Модуль поиска "tsu"	
[02]	2,37%	Силур-девонские кораллы Горного Алтая из коллекции Палеонтол...	05 Июн 2022	Модуль поиска "tsu"	
[03]	2,3%	СТРУКТУРА ЛАНДШАФТОВ И КАЧЕСТВЕННАЯ ОЦЕНКА ТЕРРИТОРИ...	09 Июн 2022	Модуль поиска "tsu"	
[04]	2,25%	ВКР Субач-2.docx	16 Янв 2022	Модуль поиска "tsu"	
[05]	2,25%	VKR_Subach-2.docx	16 Янв 2022	Модуль поиска "tsu"	
[06]	1,53%	https://lib.tsu.ru/win/produkzija/metodichka/pril1.pdf https://lib.tsu.ru	06 Июн 2022	Интернет Плюс*	
[07]	1,53%	https://lib.tsu.ru/win/produkzija/metodichka/NB_Metodichka_2021_g... https://lib.tsu.ru	24 Дек 2021	Интернет Плюс*	
[08]	1,23%	РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ КРИПТОВАЛЮТНЫ... https://elibrary.ru	31 Дек 2022	eLIBRARY.RU	
[09]	1,12%	Разработка и внедрение Web-сайта для компьютерной фирмы с в... https://otherreferats.allbest.ru	30 Июн 2022	Интернет Плюс*	
[10]	1,09%	https://www.tsu.ru/upload/medialibrary/759/rekomenduemyy_shabl... https://tsu.ru	06 Сен 2022	Интернет Плюс*	

С отчётом ознакомлен: Руков.