Санкт-Петербургский государственный университет

ВИЛКОВ Александр Борисович

Выпускная квалификационная работа

Разработка приложения Gateway API для сервиса управления кластерами Kubernetes

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5006.2019 «Математическое обеспечение и администрирование информационных систем»*

Научный руководитель:
доцент кафедры системного программирования, к.ф.-м.н., Д. В. Луцив

Рецензент:
ведущий разработчик ООО "Яндекс Технологии" А. В. Щеглов

Санкт-Петербург
2023

Saint Petersburg State University

Aleksandr Vilkov

Bachelor's Thesis

Development of a Gateway API application for the Managed Kubernetes Service

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5006.2019 "Software and Administration of Information Systems"*

Scientific supervisor:
C.Sc., docent D.V. Lutsiv

Reviewer:
lead developer at "Yandex.Technologies" A.V. Shcheglov

Saint Petersburg
2023

Оглавление

Введение

На сегодняшний день микросервисный подход к разработке архитектуры наиболее распространен при проектировании распределенных приложений [35]. Главная идея этого подхода заключается в декомпозиции приложения на набор сервисов. Каждый сервис из набора организован на основе конкретной бизнес-задачи, он поддерживает возможность независимого развертывания, а также его разработка может производиться одной небольшой командой. Вместе с тем важным аспектом этого подхода является наличие слабой связи между сервисами из набора. Выделяют два типа связи: во время выполнения и во время проектирования. Под связью во время выполнения понимают степень, в которой доступность одного сервиса зависит от доступности другого, а связь во время проектирования заключается в появлении необходимости одновременного изменения сервисов по одной и той же причине [19].

Система состоящая из большого набора сервсивов для обепечения надежности, доступности и масштабируемости нуждается в инструменте оркестрации — программное обеспечение, которое автоматизирует и упрощает управление множеством приложений, контейнеров и инфраструктуры, а также позволяет развернуть, настроить и масштабировать приложения, управлять сетевыми подключениями и балансировкой нагрузки, контролировать доступность и производительность системы. Сейчас популярным оркестратором является Kubernetes, который был разработан Google и теперь поддерживается обширным сообществом [2]. Kubernetes — это современная система управления контейнерами, которая облегчает развертывание, масштабирование и управление приложениями в контейнерах. Многие облачные провайдеры предоставляют услугу PaaS (Platform as a Service) на основе Kubernetes. Данные сервисы позволяют пользователям управлять кластером Kubernetes, скрывая сложности управления инфраструктурой, что, соответственно, дает возможность сосредоточиться на разработке и запуске приложений. В данной работе рассматривается Yandex Managed Service for

Kubernetes [27] — это управляемый сервис, предоставляемый Yandex Cloud для работы с Kubernetes.

Для доступа к микросервисам из внешней сети, требуется настройка сетевых ресурсов, таких как LoadBalancer или Ingress, что может быть сложным и трудоемким процессом. Kubernetes Gateway API появился для упрощения и автоматизации настройки и управления внешними сетевыми ресурсами и сервисами в кластере Kubernetes. Kubernetes Gateway API является открытым стандартом, который поддерживается несколькими облачными провайдерами, такими как Amazon Web Services, Microsoft Azure, Google Cloud Platform, а также Yandex Cloud. Основными функциями Kubernetes Gateway API являются управление внешним доступом и маршрутизация, проксирование и балансировка нагрузки, а такде обеспечение безопасности.

Для настройки маршрутизации и упрощения управлением трафиком в Kubernetes Gateway API используются TypedRoutes — способ определения маршрутов с помощью типизированных объектов, которые позволяют более явно определить параметры маршрутизации и обеспечивают дополнительные возможности валидации и управления трафиком. Одним из представителей TypedRoutes является TLSRoute — объект Kubernetes Gateway API, который представляет собой маршрут, который поддерживает только защищенное соединение, используя протокол TLS (Transport Layer Security). Он позволяет настраивать маршрутизацию трафика на защищенные веб-приложения, включая поддержку сертификатов и конфигурацию настроек TLS, таких как минимальная версия TLS, список поддерживаемых криптографических алгоритмов, и т. д. В данной работе рассматривается разработка поддержки TLSRoute для приложения Gateway API в Managed Service for Kubernetes от Yandex Cloud.

Дата сборки: 15 мая 2023 г.

1. Постановка задачи

Целью работы является разработка поддержки TLSRoute в приложений Gateway API для Yandex Managed Service for Kubernetes. Для её выполнения были поставлены следующие задачи:

1. Сравнить существующие приложения Gateway API у популярных облачных провайдеров;
2. спроектировать и реализовать добавление TLSRoute;
3. провести юнит-тестирование TLSRoute;
4. провести сквозное тестирование TLSRoute.

2. Обзор

2.1. Kubernetes

Kubernetes — это платформа для автоматизации развертывания, масштабирования и управления приложениями в контейнерах. Она была разработана компанией Google и стала одним из наиболее популярных инструментов для управления контейнеризованными приложениями [14].

Основной концепцией Kubernetes является управление контейнерами внутри кластера. Кластер - это группа серверов, которые работают вместе и управляют контейнерами. Kubernetes предоставляет API для управления контейнерами в кластере, таким образом, что приложения могут быть развернуты, масштабированы и обновлены автоматически, без участия операторов [21].

Kubernetes использует декларативную модель, где состояние кластера описывается в виде конфигурационных файлов. Конфигурация включает в себя описание приложений и сервисов, ресурсы, доступные для приложений, политики безопасности и т.д.

Kubernetes также предоставляет механизмы для автоматического масштабирования приложений в зависимости от их нагрузки. Он позволяет автоматически создавать и удалять контейнеры приложений в зависимости от изменения нагрузки на приложение [3].

2.2. Сеть в Kubernetes

Важным аспектом в Kubernetes является сеть, которая необходима для связи между подами и сервисами внутри кластера. При создании кластера Kubernetes, сначала создается виртуальная сеть, к которой подключаются все узлы кластера. Узлы кластера Kubernetes могут быть физическими или виртуальными машинами, а также облачными инстансами. Каждый узел должен иметь доступ к хранилищу данных, используемому для сохранения состояния приложения и общих данных, и сетевые ресурсы, необходимые для обмена данными между уз-

лами и клиентами. Сеть обычно создается на уровне облачного провайдера и может быть настроена для использования различных подсетей. В Kubernetes каждый под имеет свой IP-адрес, а каждый сервис имеет свой DNS-имя. Pod в Kubernetes — это наименьшая единица развертывания и управления приложением, которая может содержать один или несколько контейнеров. Контейнеры внутри Pod'а всегда будут запущены на одном и том же рабочем узле, что позволяет им взаимодействовать друг с другом без необходимости использования сетевых интерфейсов хоста. Каждый Pod имеет свой уникальный IP-адрес в рамках кластера Kubernetes и используется для взаимодействия с другими ресурсами кластера, такими как сервисы, репликации и контроллеры [16]. Pod'ы также являются абстракцией, которая позволяет Kubernetes автоматически масштабировать и управлять приложениями на основе их потребностей в ресурсах, таких как CPU и память. Когда необходимо увеличить масштаб приложения, Kubernetes может автоматически запускать дополнительные экземпляры Pod, распределяя нагрузку между ними. Сервисы в Kubernetes — это абстракция, которая представляет собой стабильный эндпоинт для доступа к группе подов, которые предоставляют определенную функциональность. Когда создается сервис, Kubernetes назначает ему уникальное DNS-имя, которое можно использовать для доступа к сервису из других подов в кластере [5]. Чтобы связать сервисы с группой подов, Kubernetes использует метки (labels) и селекторы (selectors). Метки представляют собой пары ключ-значение, которые могут быть назначены к подам, сервисам и другим ресурсам Kubernetes. Селекторы позволяют выбирать ресурсы на основе их меток. Когда создается новый под, он получает свой IP-адрес из подсети, назначенной для этой сети Kubernetes.

Поды могут взаимодействовать друг с другом, используя свои IP-адреса, а также порты, которые они экспонируют. Настройка сетевого взаимодействия на уровне подов может быть очень сложной и не масштабируемой задачей. Сервисы предоставляют простой и эффективный способ организации сетевого взаимодействия в кластере Kubernetes, упрощая развертывание и масштабирование приложений. Кроме того,

использование сервисов позволяет легко обеспечить балансировку нагрузки между подами, их масштабирование и обнаружение сервисов внутри кластера Kubernetes [22]. Конечным результатом является увеличение отказоустойчивости и надежности приложений, работающих в кластере Kubernetes.

Таким образом, использование сервисов в Kubernetes представляет собой ключевой элемент в организации сетевой инфраструктуры в кластере и помогает сделать приложения более масштабируемыми, отказоустойчивыми и надежными.

В Kubernetes существует несколько типов сервисов, предназначенных для различных целей. Вот некоторые из них:

- ClusterIP: Этот тип сервиса создает виртуальный IP-адрес, который доступен только внутри кластера Kubernetes. Этот тип сервиса часто используется для внутренней коммуникации между различными сервисами в кластере.

- NodePort: Этот тип сервиса открывает порт на каждом узле кластера Kubernetes, который маршрутизируется на соответствующий порт внутри сервиса. Этот тип сервиса может быть использован для обеспечения доступа к сервису извне кластера.

- LoadBalancer: Этот тип сервиса автоматически создает балансировщик нагрузки для сервиса, если он поддерживается инфраструктурой провайдера облачных услуг. Этот тип сервиса может быть использован для обеспечения доступа к сервису извне кластера, подобно типу NodePort.

- ExternalName: Этот тип сервиса дает возможность использовать внешнее DNS-имя для обращения к сервису из кластера Kubernetes.

- Headless: Этот тип сервиса используется для получения доступа к отдельным экземплярам подов внутри сервиса.

2.3. Ingress в Kubernetes

Соответственно, в стандартной конфигурации Kubernetes сервисы доступны только внутри кластера, а для доступа извне требуется настроить внешний балансировщик нагрузки или напрямую открыть порты на узлах кластера. Это может быть сложным и неудобным процессом, особенно при работе с несколькими сервисами и большим количеством пользователей. Эту проблему решает Ingress — это инструмент в Kubernetes, который позволяет управлять доступом к сервисам, работающим внутри кластера, извне. Он работает на уровне HTTP/HTTPS и позволяет управлять маршрутизацией трафика, балансировкой нагрузки, аутентификацией и авторизацией, а также обеспечивает возможность использования SSL-шифрования [11].

Ingress облегчает процесс управления доступом к сервисам, позволяя настраивать маршрутизацию,которая осуществляется с помощью правил маршрутизации, которые определяются в манифестах. Эти правила могут быть основаны на различных критериях, таких как URI, доменное имя, заголовки запросов, методы HTTP и т.д. Каждое правило маршрутизации определяет, какой запрос должен быть направлен на какой сервис или набор сервисов. Эти сервисы определяются внутри правила маршрутизации с помощью соответствующих меток, заданных в манифесте сервиса [12]. Маршрутизация трафика в Ingress может быть реализована различными способами в зависимости от используемого контроллера Ingress. Например, Nginx Ingress Controller может использовать Nginx для маршрутизации трафика, а Traefik Ingress Controller - Traefik. Маршрутизация трафика в Ingress позволяет обеспечить балансировку нагрузки, автоматический маршрутизацию трафика на основе правил и управление доступом к ресурсам на основе политик безопасности и правил аутентификации.

Также для маршрутизации могут использоваться виртуальные имена (Virtual Hosts), которые позволяют обеспечивать доступ к различным сервисам на основе имени хоста, используя один IP-адрес и порт. То есть, если на одном IP-адресе и порту существует несколько веб-

приложений, Ingress может маршрутизировать трафик на соответствующее веб-приложение в зависимости от имени хоста, указанного в запросе [15].

Для обеспечения безопасности виртуальных имен, можно настроить SSL/TLS-сертификаты для каждого имени хоста. SSL/TLS (Secure Sockets Layer/Transport Layer Security) — это протоколы, используемые для обеспечения защищенного соединения между клиентом и сервером в Интернете. Они обеспечивают шифрование трафика и защиту от подмены данных во время передачи между клиентом и сервером [23].

Однако Ingress имеет несколько заметных недостатков:

- Ограниченная поддержка протоколов: Ingress поддерживает только протоколы HTTP и HTTPS
- Ограниченная масштабируемость: Ingress может обрабатывать только трафик в пределах кластера Kubernetes
- Сложность настройки: Настройка Ingress требует большого количества настроек YAML

2.4. Kubernetes Gateway API

Для решения ограничений, которые были присущи Ingress [17], и для обеспечения более гибкого и мощного механизма управления сетевыми сервисами в Kubernetes был создан Kubernetes Gateway API [25], который представляет собой набор CRD (Custom Resource Definition) [4], расширяющий стандартную ресурсную модель Kubernetes. Он добавляет новые типы объектов Kubernetes для описания сетевых элементов, таких как виртуальные хосты, маршруты, группы серверов и т.д., которые используются для управления трафиком, проходящим через кластер Kubernetes.

С помощью ресурсов Gateway API можно описывать сетевые элементы и настраивать их параметры через Kubernetes API, используя те же инструменты и процессы, что и для других объектов Kubernetes. Кроме того, они могут использоваться совместно с другими функциями

Kubernetes, такими как RBAC [18], автомасштабирование и деплойменты, что облегчает управление сетью и трафиком в кластере Kubernetes.

Как и другие объекты Kubernetes, ресурсы Gateway API могут быть настроены и управляться через Kubernetes API, а также через инструменты командной строки, такие как kubectl.

В этом API есть концепция "ролевой ориентированности" (role-oriented), которая помогает облегчить управление сетевыми ресурсами в Kubernetes.

Ролевая ориентированность подразумевает определение ролей пользователей или сервисов, которые могут управлять определенными аспектами сетевой инфраструктуры. Например, администраторы могут иметь права на создание и удаление ресурсов сети, а разработчики могут иметь права на управление трафиком приложений [20].

Ключевыми ресурсами в Kubernetes Gateway API являются GatewayClass, Gateway и TypedRoutes.

GatewayClass - это объект, который предоставляет набор параметров и свойств для создания Gateway. GatewayClass может определять параметры, такие как протоколы, типы балансировки нагрузки и параметры безопасности, которые будут использоваться в Gateway [10].

Gateway - это объект, который представляет собой точку входа для сетевого трафика в кластере Kubernetes. Он определяет, как трафик должен быть маршрутизирован и балансирован между сервисами Kubernetes. Gateway может содержать несколько виртуальных хостов и управлять разными протоколами, такими как HTTP, HTTPS, TCP и т.д [8].

TypedRoutes - это объект, который определяет маршруты и правила маршрутизации трафика для определенного протокола и виртуального хоста внутри Gateway. Он может определять правила маршрутизации трафика на основе пути, заголовков или других параметров запроса [24].

Ролевая ориентированность в Kubernetes Gateway API позволяет упростить управление сетевыми ресурсами в Kubernetes, обеспечивая более гибкий и контролируемый подход к управлению входящим и исходящим сетевым трафиком в кластере.

Таким образом, Kubernetes Gateway API является решением для

управления сетевыми ресурсами в Kubernetes-кластере, которое упрощает настройку и управление сетевыми ресурсами и позволяет использовать широкий спектр шлюзов и режимов работы.

2.5. Kubernetes Gateway Controllers

Важно понимать, что Gateway API является спецификацией, определяющей формат и протокол взаимодействия с API-шлюзами в Kubernetes, поэтому необходим Gateway Controller, который, в свою очередь, является компонентом, реализующий эту спецификацию и обеспечивающий управление API-шлюзами в Kubernetes кластере. Таким образом, Gateway API не может работать самостоятельно без Gateway Controller. Gateway Controller необходим для взаимодействия с API-шлюзами, управления их конфигурацией и обеспечения их функционирования в соответствии с требованиями, определенными в Gateway API. Существует несколько вариантов Gateway Controller для Kubernetes, которые реализуют спецификацию Gateway API и обеспечивают управление API-шлюзами.

2.6. PaaS Kubernetes

Kubernetes как PaaS (платформа как сервис) - это услуга облачных вычислений, которая предоставляет клиентам возможность запускать, управлять и масштабировать приложения в контейнерах на основе Kubernetes, не обладая знаниями и опытом в управлении Kubernetes-кластерами.

PaaS-решения на базе Kubernetes предоставляют удобный способ развертывания, масштабирования и управления приложениями в контейнерах в облаке, позволяя пользователям сосредоточиться на разработке приложений, не задумываясь о поддержке инфраструктуры и управлении кластером Kubernetes. PaaS-решения на базе Kubernetes могут предоставлять следующие возможности: Создание кластеров Kubernetes на основе облачных провайдеров (например, AWS, Google Cloud, Azure), минуя сложности настройки инфраструктуры. Автоматическое масшта-

бирование и управление ресурсами кластера Kubernetes в зависимости от нагрузки, что позволяет экономить время и снижать затраты на ресурсы. Управление версиями приложений и деплоями, позволяющее быстро и безопасно обновлять приложения в кластере Kubernetes. Интеграцию с другими сервисами и инструментами, такими как мониторинг, логирование и автоматизация развертывания. На данный момент практически все крупные облачные провайдеры предоставляют услуги Kubernetes, так как это популярная и востребованная технология. Среди таких провайдеров можно назвать:

1. Amazon Web Services (AWS) — предоставляет услуги Amazon Elastic Kubernetes Service (EKS)
2. Microsoft Azure — предоставляет услуги Azure Kubernetes Service (AKS)
3. Google Cloud Platform (GCP) — предоставляет услуги Google Kubernetes Engine (GKE)
4. DigitalOcean — предоставляет услуги Kubernetes
5. IBM Cloud — предоставляет услуги IBM Cloud Kubernetes Service
6. Alibaba Cloud — предоставляет услуги Alibaba Cloud Container Service for Kubernetes (ACK)
7. Oracle Cloud Infrastructure (OCI) — предоставляет услуги Oracle Kubernetes Engine (OKE)
8. Yandex Cloud — предоставляет услуги Managed Service for Kubernetes

Кроме того, также есть множество других облачных провайдеров и хостинг-провайдеров, которые также предлагают услуги Kubernetes.

2.7. PaaS Kubernetes Gateway API

На данный момент среди крупных публичных облачных платформ поддержка Kubernetes Gateway API присутствует только у Google Cloud Platform и Amazon Web Services.

2.7.1. AWS Gateway API Controller

В Amazon Web Services спецификация Gateway API реализована для Amazon Elastic Kubernetes Service (EKS) — управляемого сервиса, который можно использовать для запуска Kubernetes на AWS без необходимости установки, эксплуатации и обслуживания собственной плоскости управления Kubernetes или узлов. В AWS реализуется Gateway API для интеграции с Amazon VPC Lattice. Контроллер AWS Gateway API — это проект с открытым исходным кодом, полностью поддерживаемый Amazon. После установки контроллера AWS Gateway API в кластере контроллер отслеживает создание ресурсов Gateway API, таких как Gateway и Routes, и создает соответствующие объекты Amazon VPC Lattice [1]. В AWS Gateway API на данный момент из TypedRoute-ресурсов поддерживается только HTTPRoute

2.7.2. Google Kubernetes Engine Gateway controller

Google Kubernetes Engine (GKE) - это управляемая платформа Kubernetes, предлагаемая Google Cloud. Реализация Gateway API в GKE осуществляется через контроллер GKE Gateway, который создает балансировщики нагрузки Google Cloud для подов в кластерах GKE.

Контроллер GKE Gateway поддерживает взвешенное разделение трафика, зеркалирование, продвинутую маршрутизацию, балансировку нагрузки между несколькими кластерами и другие функции [7]. Поддержка TLSRoute отсутствует.

Таким образом, реализацию поддержки TLSRoute в приложении Gateway API для Yandex Managed Service for Kubernetes можно считать преимуществом перед другими облачными платформами.

3. Требования

Необходимо реализовать поддежрку TLSRoute в приложении Gateway API для Yandex Managed Service for Kubernetes. TLSRoute предназначен для обработки защищенного трафика. Он позволяет настроить маршрутизацию трафика до Services в Kubernetes. Необходимо добавить функциональность, которая позволит создавать ресурсы TLSRoute для кластера Managed Service for Kubernetes.

4. Реализация

Важным компонентом приложения является балансировщик нагрузки. Yandex Application Loadbalancer (ALB) [26] — один из продуктов публичной облачной платформы Yandex Cloud, который служит для распределения запросов по бэкендам сетевых приложений и терминирования TLS-шифрования. Данный балансировщик реализован на основе Envoy Proxy [6]. Для популяризации использования продукта Application Loadbalancer среди пользователей Yandex Cloud было решено использовать его для балансировки нагрузки внутри Gateway API. Код проекта закрытый.

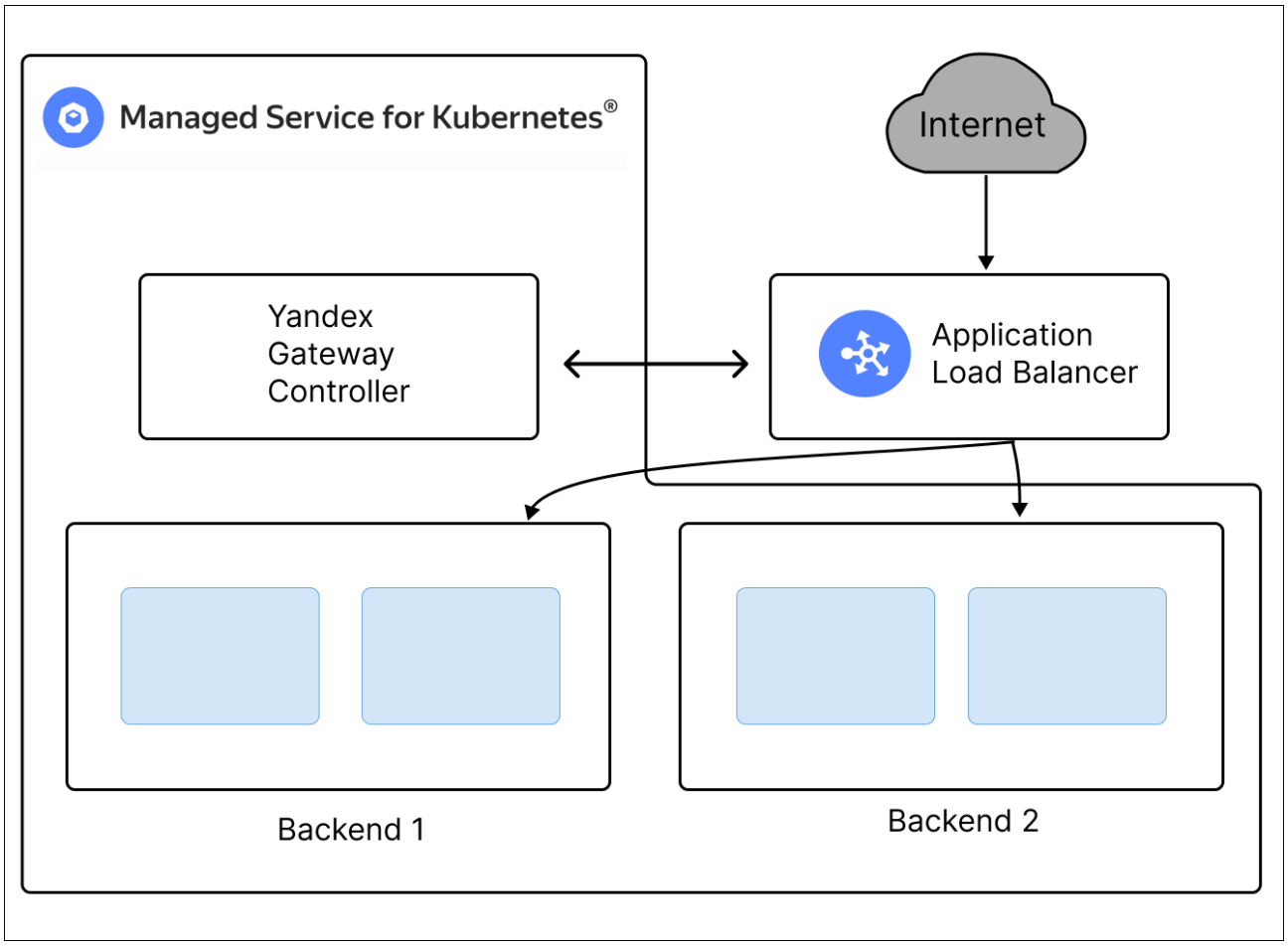

Рис. 1: Схема приложения Gateway API

Приложение Gateway API разрабатывается специально для Yandex Managed Service for Kubernetes [27]. Оно представляет собой инструмент, который позволяет управлять трафиком внутри кластера Kubernetes.

Yandex Gateway Controller является ключевым компонентом этого приложения и находится внутри самого кластера.

Основная задача Yandex Gateway Controller заключается в мониторинге обновлений конфигурации объектов Gateway API в кластере. Как только он получает новую конфигурацию, он применяет соответствующие настройки Application Loadbalancer (ALB). ALB играет роль балансировщика нагрузки и принимает входящий трафик извне кластера. Он осуществляет маршрутизацию этого трафика в соответствии с определенными ресурсами, которые были заданы в конфигурации Gateway API.

Таким образом, при помощи Gateway API и Yandex Gateway Controller можно эффективно управлять трафиком в кластере Kubernetes, оптимизировать его распределение и обеспечить надежную работу приложений. Это позволяет облегчить процесс развертывания и масштабирования приложений, обеспечивая при этом высокую производительность и отказоустойчивость системы.

4.1. Yandex Application Loadbalancer и ресурсы Gateway API

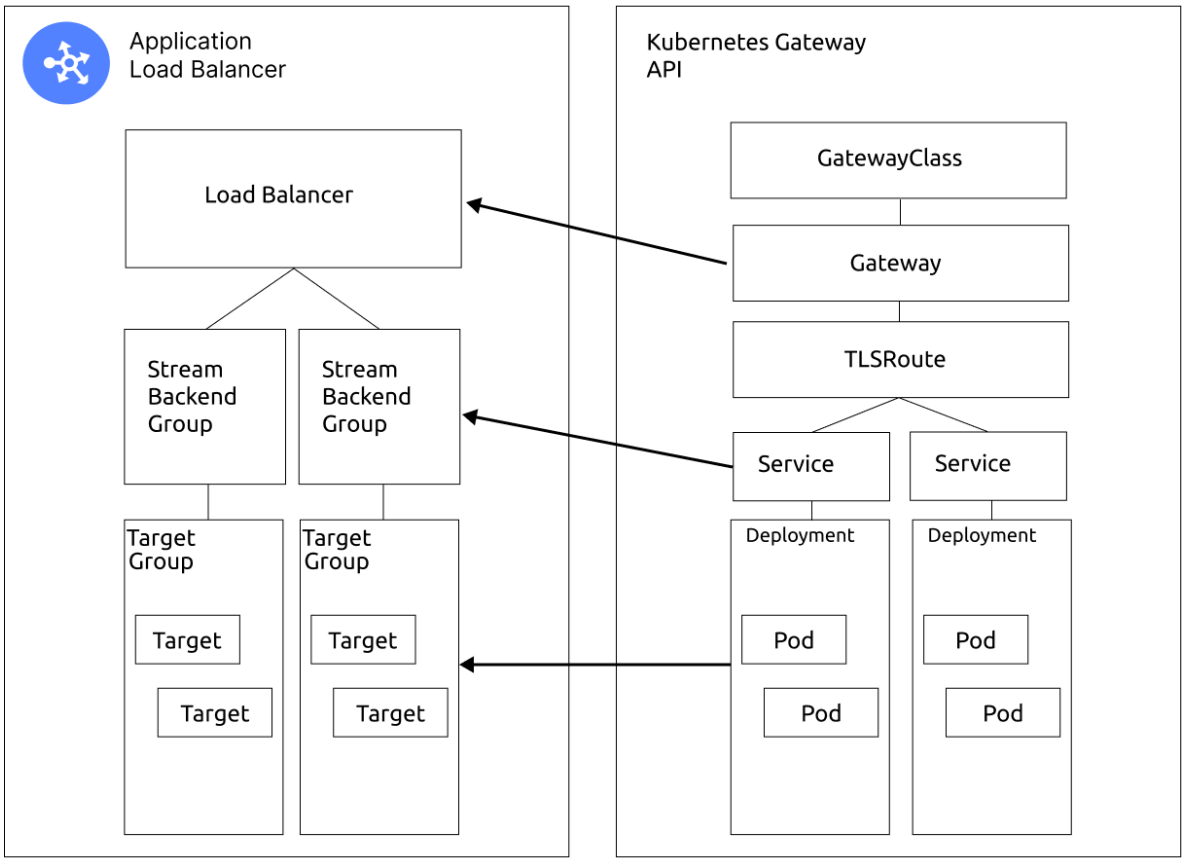

Рис. 2: Сопоставление ресурсов ALB и Gateway API

Проведем сопоставление спецификаций Loadbalancer из ALB и Gateway из GkA. Loadbalancer из ALB и Gateway из GkA выполняют функцию приема входящего трафика и его дальнейшего распределения. Для них настраиваются Listeners, которые определяют по каким адресам и портам они ожидают входящий трафик. Каждый Listeners в ALB состоит из Listener и набора Handler'ов.

Листинг 1: Описание Gateway-ресурса

```
type Gateway struct {
        metav1.TypeMeta   `json:",inline"`
        metav1.ObjectMeta `json:"metadata,omitempty"`
        Spec GatewaySpec  `json:"spec"`
```

```
        Status GatewayStatus `json:"status,omitempty"`
}

type GatewaySpec struct {
        GatewayClassName ObjectName `json:"gatewayClassName"`
        Listeners []Listener `json:"listeners"`
        Addresses []GatewayAddress `json:"addresses,omitempty"`
}

type Listener struct {
        Name SectionName `json:"name"`
        Hostname *Hostname `json:"hostname,omitempty"`
        Port PortNumber `json:"port"`
        Protocol ProtocolType `json:"protocol"`
        TLS *GatewayTLSConfig `json:"tls,omitempty"`
        AllowedRoutes *AllowedRoutes `json:"allowedRoutes,omitempty"`
}
```

Листинг 2: Фрагменты описания Loadbalancer из ALB

```
type CreateLoadBalancerRequest struct {
        ...
        Name string `json:"name,omitempty"`
        Description string `json:"description,omitempty"`
        NetworkId string `json:"network_id,omitempty"`
        ListenerSpecs []*ListenerSpec `json:"listener_specs,omitempty"`
        ...
}

type ListenerSpec struct {
        ...
        Name string `json:"name,omitempty"`
        EndpointSpecs []*EndpointSpec
        // Types that are assignable to Listener:
```

```
        //      *ListenerSpec_Http
        //      *ListenerSpec_Tls
        //      *ListenerSpec_Stream
        Listener isListenerSpec_Listener
        ...
}

type ListenerSpec_Tls struct {
        Tls *TlsListener
}

type TlsListener struct {
        ...
        DefaultHandler *TlsHandler
        SniHandlers []*SniMatch
}
```

Группы бэкендов [33] ALB получают трафик согласно типа Handler. Для приема зашифрованного TCP трафика необходимо для Handler'а использовать тип Stream, а для Listener тип Tls. Группа бэкендов состоит из списка бэкендов. Каждый бэкенд в зависимости от типа указывает на ресурсы, выступающие в роли эндпоинтов приложения. В соответствии с Handler'ом группа бэкендов также должна быть типа Stream. В Kubernetes эндопоинтом является тип Service.

TLSRoute из GkA определяет связи между Service и Listeners ресурса Gateway. С его помощью мы можем настроить связи между Listeners из ALB и группами бэкендов.

Листинг 3: Описание ресурса TLSRoute

```
type TLSRoute struct {
        metav1.TypeMeta   `json:",inline"`
        metav1.ObjectMeta `json:"metadata,omitempty"`
        // Spec defines the desired state of TLSRoute.
```

```
        Spec TLSRouteSpec `json:"spec"`

        // Status defines the current state of TLSRoute.
        Status TLSRouteStatus `json:"status,omitempty"`
}

type TLSRouteSpec struct {
        CommonRouteSpec `json:",inline"`
        Hostnames []Hostname `json:"hostnames,omitempty"`
        Rules []TLSRouteRule `json:"rules"`
}

type TLSRouteRule struct {
        BackendRefs []BackendRef `json:"backendRefs,omitempty"`
}
```

Целевая группа [34] представляет собой набор виртуальных машин. Поскольку ALB находится в одной подсети с Kubernetes возможно с помощью IP-адресов виртуальных машин направлять трафик к нужным Pod'ам.

4.2. Реализация TLSRoute

Пакет main является входной точкой приложения. Он инициализирует сущности других пакетов. Таким образом, происходит инициализиция набора средств разработки Yandex.Cloud Go SDK [28]. Создаются сущности пакета events, реализующую обработку событий происходящих в кластере. Главная сущность пакета events — EventHandler реализует интерфейс из пакета tools/cache [29], определяющий методы, которые должен реализовать обработчик событий ресурсов Kubernetes. EventHandler реагирует на различные события, связанные с ресурсами Kubernetes, такие как добавление, обновление и удаление. EventHandler узнает о событиях в кластера с помощью набора контроллеров, реализованных пакете controller. Main инициалирует контроллеры, управляю-

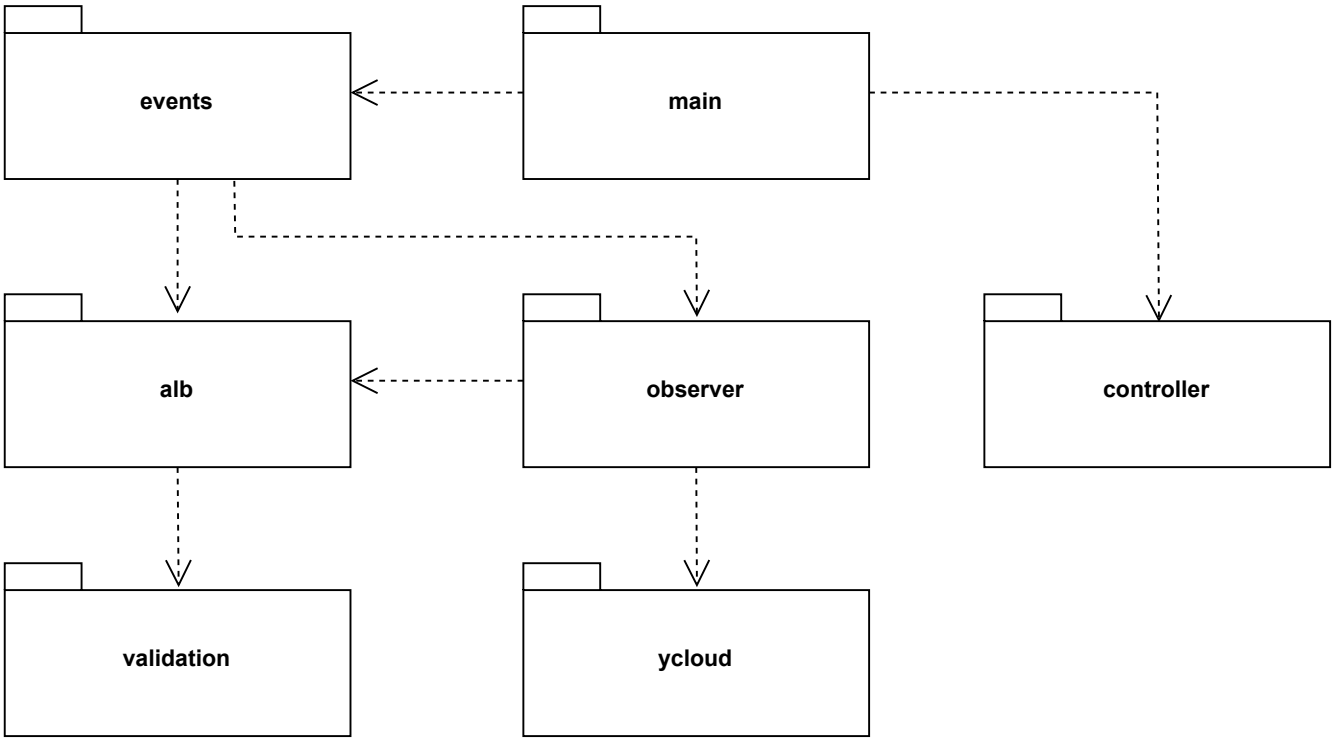

Рис. 3: Диаграмма пакетов приложения Gateway API

щие объектами Gateway API. Автором была добавлена инициализация контроллера объектов TLSRoute, который описан ниже.

Пакет controller содержит в себе логику создания контроллеров, которые предварительно настроены на управление объектами Gateway API во всех пространствах имен. В реализации используется пакет manager [31], который является частью Kubernetes Controller Runtime [13], предоставляющий набор инструментов для разработки контроллеров в Kubernetes. manager представляет собой высокоуровневый компонент, который облегчает создание и запуск контроллеров в кластере. В Controller пакет manager используется для управления жизненным циклом контроллера, обрабатывая добавление, удаление и обновление, тем самым обеспечивая их непрерывную работу. Также пакет manager используется для запуска reconciler'ов. В контекесте Kubernetes reconciler'ы - это обработчики событий, которые отвечают за приведение фактического состояния кластера в соответствие с желаемым состоянием, определенным в конфигурации объектов кластера. Reconciler'ы позволяют определить логику управления объектами Kubernetes и обеспечить автоматическое приведение состояния в соответствие с желаемым состо-

янием.

С помощью пакета controller [30] из Kubernetes Controller Runtime создается экземпляр контроллера, не управляемого сущностями пакета manager и не требующего процедуры Leader Election, что необходимо для полного контроля над жизненным циклом контроллера. Однако экземлпяр контроллера нуждается в сущностях пакета manager, поскольку они позволяют контроллеру взаимодействовать с другими компонентами системы. Например, контроллер предоставляет доступ к клиенту Kubernetes API, позволяющего взаимодействовать с ресурсами кластера.

Основываясь на описанных выше принципах, автором был разработан контроллер, который отвечает за управление объектами TLSRoute в кластере Kubernetes. Для контроллера был реализован Reconciler, который создает события для EventHandler, произошедшие с объектами TLSRoute.

Событием может быть добавление, удаление или обновления объекта Gateway API в кластере. EventHandler получает события от всех контроллеров и накапливает новую конфигурацию. Каждый объект определен с помощью официальной реализации [9] Gateway API для Kubernetes.

В главном цикле происходит обработка событий и запуск перестроения балансировщика нагрузки. Если произошло событие, вызывается метод, который обрабатывает событие и возвращает значение, указывающее изменилась ли конфигурация в такой степени, что требуется запустить процесс перестроения балансировщика нагрузки. Основная идея заключается в том, чтобы отложить обработку событий и объединить их в одну операцию. Если произошло событие и с момента последней перестройки балансировщика нагрузки прошло более 500мс, то обновление выполняется немедленно. Цикл продолжается до получения сигнала о завершении контекста.

В случае вызова операции перестройки балансировщика нагрузки, EventHandler запускает в работу пакет alb. Его главной задачей является представление и моделирование сопоставления конфигурации, опре-

деленной в объектах Gateway API, с конфигурацией необходимой для настройки балансировщика нагрузки Yandex Application Loadbalancer. Сущность DAG представляет собой модель данных балансироваща нагрузки, которая реализована с помощью структуры данных направленный ациклический граф; вся функциональность пакета завязана на ее построении.

Перед построением началом построения DAG вызывается валидация конфигурации из пакета validation. Сначала происходит валидация объектов Gateway. Для каждого такого объекта вызывается функция валидации его слушателей. Для слушателей с типом протокола TLS автор реализовал валидацию. Она основана следующих критериях: уникальность комбинации хоста, порта и протокола для каждого слушателя; наличие ссылок на сертификаты; валидность ссылок на сертификаты. После слушателей валидируется их Gateway с помощью функции, которая проходит по всем состояниям слушателей шлюза и проверяет наличие условия с типом "Ready" и статусом "False". Если найден хотя бы один неготовый слушатель, устанавливается состояние готовности шлюза в "False" и добавляется соответствующее условие.

Также автором была реализована валидация объектов TLSRoute. Валидация проходит в три этапа. На первом этапе отбираются превалидные объекты, на основе проверки значений Hostnames, которая осуществляется с помощью пакета util/validation [32]. На втором этапе из превалидных объектов отбираются те, у кого есть валидные Backends. На третьем этапе формируется окончательный список объектов, а также для каждого слушателя сопоставляются целевые объекты. Далее используется валидация объектов Services, реализованная автором на основе предыдущего этапа.

После успешной валидации происходит возвращение в пакет alb для построения направленного ациклического графа, сопоставленного с моделью данных балансировщика нагрузки. Результаты валидации используются автором для построения BackendGroups и ALBs. ALBs содержат настройки для создания Yandex Application Loadbalancer. В DAG автором быладобавлена поддержка Server Name Indication (SNI)

— это расширение протокола TLS, которое позволяет клиенту указать имя сервера, с которым он планирует установить защищенное соединение.

После успешного построения DAG приложение возвращается обратно в EventHandler, который запускает observer — это пакет, ответственность которого заключается в наблюдение за текущей конфигурацией и взаимодействие с с пакетом Ycloud. Он начинает свою работу вызовом из EventHandler после изменений в DAG. Вызывается метод achieveEverything, который обрабатывает все изменения, связанные с сертификатами, группами целевых объектов, группами бэкэндов, HTTP-маршрутизаторами и ALB.

Метод achieveEverything используется для достижения всех необходимых изменений в конфигурации. Он вызывает ряд методов для разрешения владения и достижения конфигурации сертификатов, групп целевых объектов (Target Group), групп бэкэндов (Backend Group) и балансировщиков нагрузки (Application Load Balancer). Затем вызываются методы для удаления ненужных ресурсов.

Каждый метод, отвечающий за достижение конфигурации для конкретного типа ресурса (например, achieveCerts, achieveTargetGroups и т.д.), выполняет следующие действия: resetFunc — выполняет сброс кэша или других внутренних состояний, связанных с ресурсом; createFunc — выполняет создание ресурса; updateFunc — выполняет обновление существующего ресурса; applyFunc — применяет изменения, связанные с ресурсом, например, сохраняет идентификатор созданного или обновленного ресурса. Автором была добавлена поддержка TLSRoute в этом пакете.

В observer используется пакет ycloud, который содержит интерфейс и его реализацию для взаимодействия с облачной платформой Yandex.Cloud. Он предоставляет набор функций для создания, обновления и удаления различных ресурсов в Yandex.Cloud, таких как сертификаты, группы целевых объектов (Target Group), группы бэкэндов (Backend Group) и балансировщики нагрузки (Application Load Balancer).

Интерфейс YCloud определяет методы для выполнения различных

операций, таких как создание, обновление и удаление ресурсов. В коде представлены методы для создания/обновления сертификатов, групп целевых объектов, групп бэкэндов, HTTP-маршрутизаторов и балансировщиков нагрузки. Также есть методы для получения списков ресурсов и ожидания завершения операций. Реализация интерфейса YCloud в структуре yCloud использует Yandex.Cloud Go SDK [28] для взаимодействия с API облачной платформы. В методах реализации используются соответствующие функции из библиотеки для отправки запросов к API и обработки ответов. В целом, данный пакет предоставляет абстракцию для работы с Yandex.Cloud и упрощает создание, обновление и удаление ресурсов в облаке. Автором была добавлена поддержка создания групп бэкендов типа Stream, а также использование sniMatch для создания обработчиков TLSRoute.

5. Тестирование

5.1. Юнит-тестирование

Было проведено юнит-тестирование кода приложения Gateway API с целью оценки его функциональности и корректности работы. Юнит-тестирование является одним из важных этапов разработки программного обеспечения и позволяет проверить отдельные компоненты системы на соответствие требованиям и ожидаемому поведению.

Для юнит-тестирования Gateway Controller было разработано специальное окружение, которое воссоздало среду выполнения контроллера с учетом всех зависимостей и конфигурации. Данный подход позволяет изолировать тестируемый код от внешних факторов и обеспечить контролируемые условия тестирования.

В ходе тестирования были разработаны и запущены наборы тестовых сценариев, охватывающих различные аспекты функциональности приложения Gateway API. Тестовые сценарии включали в себя создание и удаление ресурсов, обработку входящих запросов, управление состоянием и проверку правильности маршрутизации трафика.

Для каждого тестового сценария были определены ожидаемые результаты и проверочные механизмы, позволяющие автоматически оценить успешность выполнения тестов. В процессе тестирования осуществлялся сбор и анализ результатов, выявление потенциальных ошибок и проблемных ситуаций, а также их исправление.

Результаты юнит-тестирования позволили оценить работу приложения Gateway API в контролируемых условиях и обнаружить возможные дефекты или несоответствия требованиям.

По результатам юнит-тестирования кода Gateway Controller было достигнуто высокое покрытие тестами, составляющее около 86 процентов. Это означает, что большая часть функциональности контроллера была проверена и охвачена тестами, что способствует обнаружению потенциальных ошибок и гарантирует более надежное и стабильное функционирование системы.

5.2. Сквозное тестирование

Для проведения сковозного E2E (end-to-end) тестирования Gateway Controller было подготовлено окружение, в котором осуществлялась развертка контроллера в кластере Kubernetes. В ходе тестовых сценариев, осуществляемых в самом кластере, клиент Kubernetes загружал ингрессы, а затем проверял доступность URL-адресов, создаваемых балансировщиками.

Процесс поднятия окружения для E2E тестирования Gateway Controller включал предварительную настройку кластера Kubernetes с необходимыми ресурсами и конфигурацией. Затем контроллер был развернут в кластере путем применения соответствующих манифестов и установки необходимых зависимостей.

Сами E2E тесты представляли собой сценарии, в которых клиент Kubernetes, используя API, загружал ресурсы, соответствующие конфигурации балансировщиков. Затем выполнялась проверка доступности URL-адресов, которые были созданы балансировщиками, для обеспечения корректной работы маршрутизации трафика.

Эти тесты позволили проверить функциональность и работоспособность Gateway Controller, а также убедиться в том, что создаваемые балансировщики обеспечивают ожидаемый уровень доступности сервисов. Они способствовали выявлению потенциальных проблем и обеспечению надежности работы контроллера при различных условиях и нагрузке.

Заключение

Была реализована и протестирована поддержка TLSRoute в приложении Gateway API для Yandex Managed Service for Kubernetes.

- Проведено сравнение существующих приложений Gateway API у популярных облачных провайдеров;
- спроектировано и реализовано добавление TLSRoute;
- проведено юнит-тестирование;
- проведено сквозное тестирование.

Список литературы

[1] AWS Gateway API Controller. — URL: https://www.gateway-api-controller.eks.aws.dev/.

[2] Aravena Ricardo. Why is Kubernetes getting so popular? — URL: https://stackoverflow.blog/2020/05/29/why-kubernetes-getting-so-popular/ (дата обращения: 2023-05-15).

[3] Balla David, Simon Csaba, Maliosz Markosz. Adaptive scaling of Kubernetes pods // NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium. — Vol. 2. — 2020. — P. 32–33.

[4] Custom Resources | Kubernetes. — URL: https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/.

[5] DNS for Services and Pods | Kubernetes. — URL: https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/.

[6] Envoy Proxy. — URL: https://www.envoyproxy.io/r.

[7] GKE Gateway controller. — URL: https://cloud.google.com/kubernetes-engine/docs/concepts/gateway-api.

[8] Gateway | Gateway API. — URL: https://gateway-api.sigs.k8s.io/concepts/api-overview/#gateway.

[9] Gateway API | Kubernetes. — URL: https://github.com/kubernetes-sigs/gateway-api.

[10] GatewayClass | Gateway API. — URL: https://gateway-api.sigs.k8s.io/concepts/api-overview/#gatewayclass.

[11] Ingress | Kubernetes. — URL: https://kubernetes.io/docs/concepts/services-networking/ingress/.

[12] Ingress | Kubernetes. — URL: https://kubernetes.io/docs/concepts/services-networking/ingress/#ingress-rules.

[13] Kubernetes Controller Runtime. — URL: https://pkg.go.dev/sigs.k8s.io/controller-runtime.

[14] Luksa M. Kubernetes in Action. — Manning, 2017. — ISBN: 9781638355342. — URL: https://books.google.ru/books?id=WTgzEAAAQBAJ.

[15] Name based virtual hosting | Kubernetes. — URL: https://kubernetes.io/docs/concepts/services-networking/ingress/#name-based-virtual-hosting.

[16] Pod networking | Kubernetes. — URL: https://kubernetes.io/docs/concepts/workloads/pods/#pod-networking.

[17] Q: What are the differences between Ingress and Gateway API? | Gateway API. — URL: https://gateway-api.sigs.k8s.io/faq/?h=ingress.

[18] RBAC | Gateway API. — URL: https://gateway-api.sigs.k8s.io/concepts/security-model/#rbac.

[19] Richardson Chris. What are microservices? — URL: https://microservices.io/#:~:text=What%20are%20microservices%3F (дата обращения: 2023-05-15).

[20] Roles And Personas | Gateway API. — URL: https://gateway-api.sigs.k8s.io/concepts/security-model/#roles-and-personas.

[21] Sayfan G. Mastering Kubernetes. — Packt Publishing, 2017. — ISBN: 9781786469854. — URL: https://books.google.ru/books?id=dnc5DwAAQBAJ.

[22] Service | Kubernetes. — URL: https://kubernetes.io/docs/concepts/services-networking/service/.

[23] TLS in Ingress | Kubernetes. — URL: https://kubernetes.io/docs/concepts/services-networking/ingress/#tls.

[24] TypedRoute Resource | Gateway API. — URL: https://gateway-api.sigs.k8s.io/concepts/api-overview/#route-resources.

[25] What is the Gateway API | Gateway API. — URL: https://gateway-api.sigs.k8s.io/#what-is-the-gateway-api.

[26] Yandex Application Loadbalancer. — URL: https://cloud.yandex.ru/services/application-load-balancer.

[27] Yandex Managed Service for Kubernetes. — URL: https://cloud.yandex.ru/services/managed-kubernetes.

[28] Yandex.Cloud Go SDK | Github. — URL: https://github.com/yandex-cloud/go-sdk.

[29] cache | k8s.io/client-go. — URL: https://pkg.go.dev/k8s.io/client-go/tools/cache.

[30] controller | Kubernetes Controller Runtime. — URL: https://pkg.go.dev/sigs.k8s.io/controller-runtime@v0.14.6/pkg/controller.

[31] manager | Kubernetes Controller Runtime. — URL: https://pkg.go.dev/sigs.k8s.io/controller-runtime@v0.14.6/pkg/manager.

[32] validation | Kubernetes Apimachinery. — URL: https://github.com/kubernetes/apimachinery/tree/master/pkg/util/validation.

[33] Группы бэкендов | Yandex Application Loadbalancer. — URL: https://cloud.yandex.ru/docs/application-load-balancer/concepts/backend-group.

[34] Целевые группы | Yandex Application Loadbalancer. — URL: https://cloud.yandex.ru/docs/application-load-balancer/concepts/target-group.

[35] Ю.В. Литвинов. Проектирование распределенных приложений, технические вопросы. — URL: https://github.com/yurii-litvinov/courses/blob/master/software-design/14-distributed-applications-design/14-distributed-applications-design-text.pdf (дата обращения: 2023-05-15).