Соломатин К.В. Контурная модель размещения компонентов распределённых информационных систем в мультикластерной среде KUBERNETES// Международный журнал информационных технологий и энергоэффективности. – 2026. – Т. 11 № 6 (68) Ч.1 с. 122–131

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/



УДК 004.75

КОНТУРНАЯ МОДЕЛЬ РАЗМЕЩЕНИЯ КОМПОНЕНТОВ РАСПРЕДЕЛЁННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ В МУЛЬТИКЛАСТЕРНОЙ СРЕДЕ KUBERNETES

Соломатин К.В.
ФГБОУ ВО «МИРЭА - РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ», Москва, Россия (119454, г. Москва, пр-т Вернадского, д. 78, стр. 4), e-mail: kbc.off@list.ru

Предложена контурная модель размещения компонентов распределённых информационных систем в мультикластерной среде Kubernetes. Модель задаёт функциональное разделение на клиентский, серверный, хранилищный и управляющий контуры, фиксирует правила межконтурного взаимодействия и ориентирована на гетерогенные вычислительные среды. Подход апробирован на архитектуре рекомендательной системы подбора рекламных площадок.

Ключевые слова: Контурная модель, Kubernetes, мультикластерная архитектура, распределённые системы, рекомендательная система, WireGuard, Istio.

CONTOUR MODEL FOR COMPONENT PLACEMENT IN DISTRIBUTED INFORMATION SYSTEMS WITHIN A MULTI-CLUSTER KUBERNETES ENVIRONMENT

Solomatin K.V.
MIREA - RUSSIAN TECHNOLOGICAL UNIVERSITY, Moscow, Russia (119454, Moscow, avenue Vernadsky, 78, b. 4), e-mail: kbc.off@list.ru

A contour model for component placement in distributed information systems within a multi-cluster Kubernetes environment is proposed. The model introduces functional segregation into client, server, storage, and management contours, defines inter-contour interaction rules, and targets heterogeneous computing environments. The approach is illustrated using a recommendation system for advertising platform selection.

Keywords: Contour model, Kubernetes, multi-cluster architecture, distributed systems, recommendation system, WireGuard, Istio.

Введение

Развитие контейнерных технологий и микросервисной архитектуры сделало мультикластерные конфигурации одним из практических способов развёртывания распределённых информационных систем. Это особенно характерно для прикладных решений, сочетающих пользовательские интерфейсы, вычислительные сервисы и ресурсоёмкие подсистемы хранения и обработки данных [1, 2, 3].

Согласно отчётам Cloud Native Computing Foundation, свыше 60 % организаций, применяющих Kubernetes, используют более одного кластера в производственной среде. Основными причинами являются территориальное распределение инфраструктуры, изоляция зон ответственности, соблюдение регуляторных требований к локализации данных, а также

стремление ограничить область поражения при отказе отдельного управляющего компонента [4, 5].

При размещении всех компонентов в одном Kubernetes-кластере усложняется изоляция нагрузок, возрастает зависимость от общего состояния управляющей плоскости и снижается гибкость использования гетерогенной инфраструктуры, включающей VPS и выделенные физические узлы [4, 5]. Инструменты KubeFed, Rancher и Cluster API решают задачи оркестрации и сопровождения парка кластеров, однако не формализуют принципы функционального распределения компонентов между ними [6, 7, 8].

В исследованиях по микросервисной архитектуре основное внимание обычно уделяется декомпозиции системы, гранулярности сервисов и управлению API-взаимодействием [9]. Вопрос группировки компонентов по функциональным ролям на уровне нескольких кластеров остаётся менее формализованным. Существующие рекомендации ограничиваются практиками «один кластер на среду» или «один кластер на команду», не предлагая системного подхода к распределению компонентов на основании их функциональных характеристик.

Цель исследования

Цель статьи заключается в формализации контурной модели размещения компонентов распределённой системы в мультикластерной среде Kubernetes и демонстрации её применения к рекомендательному конвейеру обработки данных. Для достижения поставленной цели решаются следующие задачи: анализ существующих средств мультикластерного управления и выявление архитектурного пробела; формализация модели контурного размещения с определением структуры, принципов и метрики качества; проверка применимости модели на примере распределённой рекомендательной системы подбора рекламных площадок.

Материал и методы исследования

Среди распространённых средств работы с мультикластерной средой можно выделить KubeFed, Cluster API и Rancher. Их назначение связано с синхронизацией ресурсов, централизованным администрированием и управлением жизненным циклом кластеров, но не с описанием архитектурных принципов размещения прикладных компонентов [6, 7, 8].
KubeFed (Federation v2) обеспечивает репликацию объектов Kubernetes между кластерами на декларативной основе. Инструмент позволяет задавать политики размещения ресурсов, однако оперирует инфраструктурными абстракциями (Deployment, ConfigMap, Service) и не содержит механизма функциональной классификации компонентов. Cluster API формализует жизненный цикл самих кластеров: их создание, обновление и удаление. Rancher предоставляет единую консоль управления произвольным набором кластеров и поддерживает импорт существующих сред, включая K3s и RKE2, но не предлагает формальной модели размещения компонентов [6, 7, 8].

Тем самым перечисленные решения следует рассматривать как инструментальный слой реализации. Предлагаемая контурная модель относится к архитектурному уровню и задаёт критерии распределения компонентов по кластерам. Таблица 1 показывает степень покрытия соответствующих требований существующими средствами.

Таблица 1 - Покрытие требований контурной модели существующими инструментами

Требование	**Federation v2**	**Cluster API**	**Rancher**	**Контурная модель**
Поддержка гетерогенных сред	Ограниченная: ручная адаптация [6]	Частичная: зависит от провайдера	Да: импорт произвольных кластеров	K3s на VPS и физических серверах [10]
Наложенная сетевая связность	Не предусмотрена	Не предусмотрена	Частичная: через сторонние плагины	WireGuard full-mesh [11]
Межкластерное обнаружение сервисов	Федеративная DNS	Отсутствует	Через сторонний service mesh	Istio multicluster east-west [12]
Архитектурное разделение по ролям	Не определено	Не определено	Возможно через проекты и labels	Формализовано: сегрегация контуров
Стратегия деградации при отказе	Не определена	Не определена	Не определена	Постепенная деградация

Из Таблицы следует, что существующие средства покрывают отдельные аспекты мультикластерной эксплуатации, но не задают формальной схемы функционального разделения кластеров и сценариев деградации. Именно этот уровень абстракции и фиксируется контурной моделью.

Контурная модель размещения компонентов — архитектурный паттерн, при котором каждый Kubernetes-кластер мультикластерной среды выделяется для выполнения определённой функциональной роли (контура). Компоненты информационной системы распределяются по контурам на основании однородности требований к вычислительным ресурсам, безопасности, доступности и характеру нагрузки. Количество контуров определяется особенностями конкретной системы; ниже приведена обобщённая формализация модели.

Модель определяется кортежем $\langle S, E, C, R, I \rangle$, где: $S = \{s_1, s_2, \ldots, s_m\}$ — множество компонентов (сервисов) системы; $E \subseteq S \times S$ — множество зависимостей между компонентами (вызовы API, потоки данных), определяющее граф зависимостей $G = (S, E)$; $C = \{c_1, c_2, \ldots, c_n\}$ — множество функциональных контуров; $R: S \to C$ — функция размещения, отображающая каждый компонент на контур; $I \subseteq C \times C$ — множество разрешённых межконтурных взаимодействий.

Для оценки качества размещения вводятся следующие характеристики. Пусть $\text{intra}(c) = |\{(s_1, s_2) \in E : R(s_1) = R(s_2) = c\}|$ — число рёбер внутри контура c, а $\text{inter} = |\{(s_1, s_2) \in E : R(s_1) \neq R(s_2)\}|$ — суммарное число межконтурных рёбер. Принцип сегрегации контуров формулируется как задача максимизации отношения суммарной внутриконтурной связности к межконтурной: $Q(R) = \Sigma\, \text{intra}(c_i) / (\text{inter} + 1) \to \max$ при ограничении на однородность

ресурсного профиля внутри каждого контура. Данный критерий сочетает два требования: минимизацию межкластерного трафика (латентность наложенной сети [11]) и группировку компонентов с близкими ресурсными характеристиками.

Метрика Q(R) позволяет сравнивать различные варианты размещения при проектировании системы. Более высокое значение Q указывает на то, что большинство зависимостей обслуживается внутрикластерным трафиком с минимальной латентностью, тогда как межконтурные взаимодействия сведены к явно определённым точкам интеграции. На практике абсолютное значение Q менее информативно, чем относительное сравнение альтернативных вариантов размещения для одной и той же системы.

В рамках модели предполагается, что каждый компонент принадлежит ровно одному контуру, а компоненты одного контура обладают близким ресурсным профилем. Если пара контуров не входит в множество разрешённых взаимодействий, соответствующие зависимости между компонентами исключаются на уровне архитектурного графа. Допустимые межконтурные обращения реализуются через сервисную сетку с обязательным mTLS [12, 15]. При этом для каждого разрешённого взаимодействия должна быть определена стратегия деградации, исключающая каскадный отказ системы [4].

Для систем с выраженным разделением на интерфейсный, вычислительный, хранилищный и эксплуатационный слои принцип сегрегации естественным образом приводит к выделению четырёх функциональных контуров [3]. Каждый контур характеризуется преобладающим типом нагрузки: stateless-обработка HTTP-запросов для клиентского контура, CPU-интенсивные вычисления для серверного, IO-интенсивные операции для хранилищного и низкоприоритетная сборка метрик для управляющего.

В прикладном сценарии выделяются четыре функциональных контура. Клиентский контур обслуживает веб-интерфейсы, административную панель и контроллер входящего трафика; для него характерны stateless-нагрузка и повышенные требования к доступности. Серверный контур включает прикладные сервисы, в том числе парсеры внешних источников, ETL-узлы, API рекомендаций и сервис аутентификации; этот контур ориентирован на вычислительно-интенсивную нагрузку. Контур хранения объединяет персистентные компоненты, включая реляционную СУБД, кэш, брокер сообщений, аналитический и поисковый слои, и характеризуется интенсивной дисковой нагрузкой и stateful-природой компонентов [3, 4]. Контур управления содержит средства наблюдаемости, журналирования, трассировки и управления секретами, обеспечивая сквозной мониторинг остальных частей системы [13].

Множество разрешённых взаимодействий: $I = \{(c_1, c_2), (c_2, c_1), (c_2, c_3), (c_3, c_2), (c_4, c_1), (c_4, c_2), (c_4, c_3)\}$. Клиентский контур не обращается к хранилищу напрямую, а взаимодействует с ним исключительно через серверный контур. Управляющий контур имеет доступ на чтение ко всем остальным контурам для сбора метрик и журналов, но не участвует в прикладном потоке обработки данных.

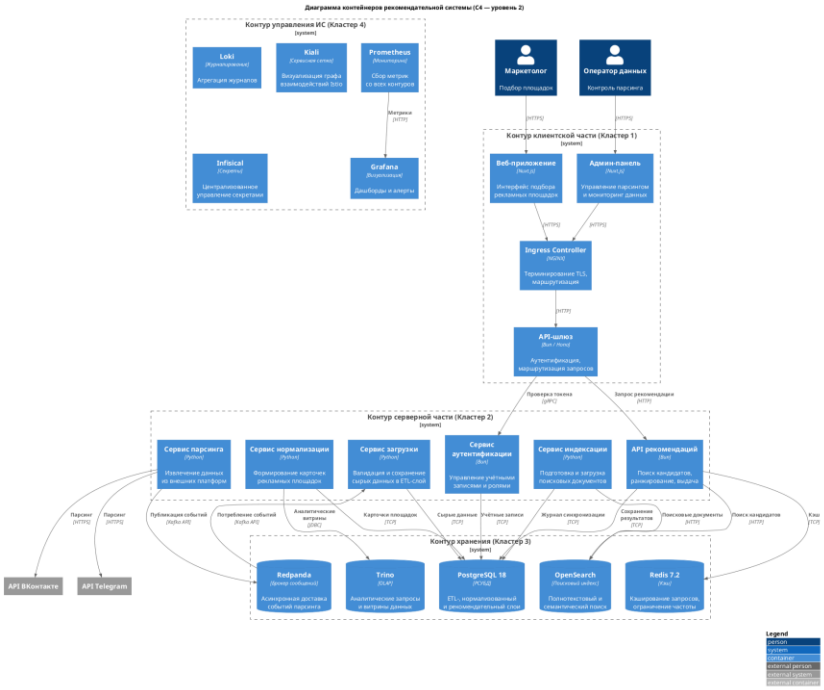


Рисунок 1 - Диаграмма контейнеров (C4, уровень 2): четыре функциональных контура системы [14]

Кластеры объединены наложенной VPN-сетью WireGuard с полносвязной топологией и единым адресным пространством [11]. WireGuard выбран в качестве транспортного уровня благодаря минимальной кодовой базе (менее 4 000 строк кода в ядре Linux), высокой пропускной способности и низкой латентности по сравнению с IPsec и OpenVPN. Каждый узел кластера получает уникальный IP-адрес из единого блока /16, что обеспечивает прозрачную маршрутизацию между подами разных кластеров.

Поверх сетевого уровня развёрнута сервисная сетка Istio в режиме multicluster [12], обеспечивающая прозрачное обнаружение сервисов между контурами через east-west шлюзы, обязательное mTLS-шифрование межкластерного и внутрикластерного трафика, авторизационные политики для ограничения набора допустимых взаимодействий и механизм автоматического прерывателя цепи при превышении порога ошибок. Конфигурация Istio включает централизованный корневой центр сертификации, от которого каждый кластер получает промежуточный сертификат, что позволяет поддерживать единую цепочку доверия при независимости управляющих плоскостей отдельных кластеров [15].

Контурная модель применена к распределённой рекомендательной системе подбора рекламных площадок. Функциональная цепочка системы включает сбор сведений из внешних платформ, асинхронную доставку событий через брокер сообщений, нормализацию данных, индексирование поисковых документов и выдачу рекомендаций по текстовому запросу пользователя [1, 3]. Вычислительная среда системы является гетерогенной и включает виртуальные серверы различных провайдеров, а также выделенный физический сервер для контура хранения [4].

Инфраструктура развёрнута на четырёх кластерах K3s. Клиентский кластер размещён на VPS с 2 vCPU и 4 ГБ ОЗУ и обслуживает входящий трафик через Ingress-контроллер Traefik. Серверный кластер использует VPS с 4 vCPU и 8 ГБ ОЗУ, обеспечивая достаточные вычислительные ресурсы для парсеров и ETL-конвейера. Контур хранения размещён на

выделенном физическом сервере с 32 ГБ ОЗУ и NVMe-накопителями, что критично для дисковых нагрузок PostgreSQL, OpenSearch и Redpanda. Управляющий кластер развёрнут на отдельном VPS с минимальными ресурсами (2 vCPU, 2 ГБ ОЗУ), поскольку инструменты наблюдаемости не требуют интенсивных вычислений [10].

В распределении компонентов по контурам клиентская часть включает веб-интерфейс, административную панель, API-шлюз и контроллер входящего трафика. Серверный контур объединяет парсер внешних платформ, сервис загрузки данных, сервис нормализации, индексатор поисковых документов, API рекомендаций и сервис аутентификации. Контур хранения включает PostgreSQL, Redis, Redpanda, Trino и OpenSearch [3, 4]. В управляющем контуре размещаются Prometheus, Grafana, Loki, Kiali и Infisical [13]. Такое распределение согласует архитектуру системы с различиями в типах нагрузки и требованиях к доступности отдельных подсистем.

Значение метрики Q(R) для данного размещения составляет 4,2 при 17 внутриконтурных и 8 межконтурных зависимостях. Для сравнения, размещение всех сервисов в двух кластерах (клиент + остальные) даёт Q = 1,8, а объединение всех компонентов в одном кластере — Q = 0 (отсутствие межконтурных рёбер делает метрику неприменимой). Таким образом, четырёхконтурная схема более чем вдвое превышает двухкластерную альтернативу по показателю функциональной сегрегации.

Одним из практических требований к модели выступает отсутствие каскадного отказа при нарушении межкластерной связности. Поэтому для каждого контура задаётся режим ограниченного функционирования, сохраняющий работоспособность оставшихся частей системы [4].

Если недоступен контур хранения, серверная часть переводит операции записи в локальную очередь повторной обработки, а пользовательский контур ограничивает часть функций до восстановления связности. При отказе серверного контура клиентская часть может продолжать обслуживание кэшированных результатов, тогда как сбор новых данных приостанавливается. Недоступность управляющего контура не должна останавливать прикладной конвейер, а приводит только к временной потере телеметрии. При отказе клиентского контура сбор и нормализация данных сохраняются, однако пользовательский доступ становится временно недоступным.

Механизм автоматического прерывателя цепи (circuit breaker), реализованный средствами Istio, при превышении порога ошибок временно прекращает направление трафика на неисправный эндпоинт и возобновляет его после восстановления [12]. В проведённом тестовом сценарии отключение сетевого канала между серверным и хранилищным контурами приводило к срабатыванию прерывателя в течение 5 секунд, после чего серверный контур переходил в режим локального кэширования запросов. Восстановление связности автоматически возобновляло нормальный поток обработки без вмешательства оператора.

Результаты апробации показывают, что контурная модель применима к системам с выраженной гетерогенностью вычислительных ресурсов и различием требований к доступности отдельных подсистем. Разделение на функциональные контуры позволяет независимо масштабировать каждую группу компонентов и проводить обновления без воздействия на остальные части системы. Вместе с тем модель вводит дополнительные

накладные расходы на межкластерный трафик и усложняет начальную настройку сетевой связности.

Выводы

В статье предложена контурная модель размещения компонентов распределённых информационных систем в мультикластерной среде Kubernetes. Модель формализует функциональное разделение кластеров по прикладным ролям и задаёт набор правил межконтурного взаимодействия.

Сопоставление с существующими средствами мультикластерного управления показало, что рассматриваемый подход закрывает пробел между инструментами эксплуатации кластеров и архитектурными принципами распределения прикладных компонентов.

Применение модели к рекомендательной системе показало, что функциональная сегрегация упрощает размещение компонентов в гетерогенной среде и позволяет локализовать последствия отказов. Предложенная метрика Q(R) даёт количественную основу для сравнения альтернативных вариантов размещения.

Ограничение работы состоит в проектном характере верификации модели. Дальнейшее развитие связано с экспериментальной оценкой производительности, анализом накладных расходов межконтурного взаимодействия и уточнением метрик эффективности контурного разделения.

Список литературы

1. Михайлов А.Н. Разработка рекомендательных систем: методы и алгоритмы для электронной коммерции // Вестник науки. 2024. URL: https://cyberleninka.ru/article/n/razrabotka-rekomendatelnyh-sistem-metody-i-algoritmy-dlya-elektronnoy-kommertsii (дата обращения: 10.04.2026).
2. Анализ современных подходов в проектировании рекомендательных систем / К.А. Разуваев [и др.] // Международный журнал прикладных наук и технологий «Integral». 2021. URL: https://cyberleninka.ru/article/n/analiz-sovremennyh-podhodov-v-proektirovanii-rekomendatelnyh-sistem (дата обращения: 10.04.2026).
3. Артамонов Ю.С., Востокин С.В. Разработка распределенных приложений сбора и анализа данных на базе микросервисной архитектуры // Известия Самарского научного центра Российской академии наук. 2016. URL: https://cyberleninka.ru/article/n/razrabotka-raspredelennyh-prilozheniy-sbora-i-analiza-dannyh-na-baze-mikroservisnoy-arhitektury (дата обращения: 10.04.2026).
4. Долматов Р.А., Сараджишвили С.Э. Оптимизация масштабируемости и отказоустойчивости микросервисов с применением Kubernetes // Системный анализ в проектировании и управлении. 2024. URL: https://cyberleninka.ru/article/n/optimizatsiya-masshtabiruemosti-i-otkazoustoychivosti-mikroservisov-s-primeneniem-kubernetes (дата обращения: 10.04.2026).
5. Моделирование масштабируемых микросервисных систем в условиях контейнерной виртуализации с использованием Kubernetes и Service Mesh (на примере Istio) / А.Э. Баженов [и др.] // Программные системы и вычислительные методы. 2025. URL: https://cyberleninka.ru/article/n/modelirovanie-masshtabiruemyh-mikroservisnyh-sistem-v-

usloviyah-konteynernoy-virtualizatsii-s-ispolzovaniem-kubernetes-i-service (дата обращения: 10.04.2026).
6. KubeFed User Guide [Электронный ресурс] // Kubernetes SIG Multicluster. URL: https://github.com/kubernetes-retired/kubefed/tree/master/docs/userguide (дата обращения: 10.04.2026).
7. Rancher Documentation. Architecture and Fleet [Электронный ресурс]. URL: https://ranchermanager.docs.rancher.com/ (дата обращения: 10.04.2026).
8. Cluster API Book. Introduction [Электронный ресурс]. URL: https://cluster-api.sigs.k8s.io/ (дата обращения: 10.04.2026).
9. Чикалева Ю.С. Анализ гранулярности микросервисов: эффективность архитектурных подходов // Программные системы и вычислительные методы. 2025. URL: https://cyberleninka.ru/article/n/analiz-granulyarnosti-mikroservisov-effektivnost-arhitekturnyh-podhodov (дата обращения: 10.04.2026).
10. K3s Documentation. Lightweight Kubernetes [Электронный ресурс]. URL: https://docs.k3s.io/ (дата обращения: 10.04.2026).
11. Donenfeld J.A. WireGuard: Next Generation Kernel Network Tunnel // Proceedings of the Network and Distributed System Security Symposium. 2017. URL: https://www.ndss-symposium.org/wp-content/uploads/2017/09/ndss2017_04A-3_Donenfeld_paper.pdf (дата обращения: 10.04.2026).
12. Install Multicluster [Электронный ресурс] // Istio Documentation. URL: https://istio.io/latest/docs/setup/install/multicluster/ (дата обращения: 10.04.2026).
13. Мясников И.В. Мониторинг компонентов Istio для обеспечения надежности и наблюдаемости Service Mesh // Вестник науки. 2025. URL: https://cyberleninka.ru/article/n/monitoring-komponentov-istio-dlya-obespecheniya-nadezhnosti-i-nablyudaemosti-service-mesh (дата обращения: 10.04.2026).
14. Корниенко Д.В., Никулин А.В. Визуализация архитектур информационных систем, основанных на микросервисах, с использованием данных OpenTelemetry // Computational nanotechnology. 2024. URL: https://cyberleninka.ru/article/n/vizualizatsiya-arhitektur-informatsionnyh-sistem-osnovannyh-na-mikroservisah-s-ispolzovaniem-dannyh-opentelemetry (дата обращения: 10.04.2026).
15. Ковтун Д.П., Лапонина О.Р. Использование управления доступом на основе атрибутов и mTLS в микросервисной архитектуре // International Journal of Open Information Technologies. 2025. URL: https://cyberleninka.ru/article/n/ispolzovanie-upravleniya-dostupom-na-osnove-atributov-i-mtls-v-mikroservisnoy-arhitekture (дата обращения: 10.04.2026).

References

1. Mikhailov, A.N. "Development of Recommender Systems: Methods and Algorithms for E-Commerce" // Science Bulletin. 2024. URL: https://cyberleninka.ru/article/n/razrabotka-rekomendatelnyh-sistem-metody-i-algoritmy-dlya-elektronnoy-kommertsii (Accessed: 10.04.2026).
2. Analysis of Modern Approaches to Designing Recommender Systems / K.A. Razuvaev et al. // International Journal of Applied Sciences and Technologies "Integral". 2021. URL:

https://cyberleninka.ru/article/n/analiz-sovremennyh-podhodov-v-proektirovanii-rekomendatelnyh-sistem (Accessed: 10.04.2026).

3. Artamonov Yu.S., Vostokin S.V. Development of distributed data collection and analysis applications based on microservice architecture // Bulletin of the Samara Scientific Center of the Russian Academy of Sciences. 2016. URL: https://cyberleninka.ru/article/n/razrabotka-raspredelennyh-prilozheniy-sbora-i-analiza-dannyh-na-baze-mikroservisnoy-arhitektury (date of access: 10.04.2026).
4. Dolmatov R.A., Sarajishvili S.E. Optimization of scalability and fault tolerance of microservices using Kubernetes // Systems analysis in design and management. 2024. URL: https://cyberleninka.ru/article/n/optimizatsiya-masshtabiruemosti-i-otkazoustoychivosti-mikroservisov-s-primeneniem-kubernetes (date of access: 10.04.2026).
5. Modeling scalable microservice systems in the context of container virtualization using Kubernetes and Service Mesh (using Istio as an example) / A.E. Bazhenov [et al.] // Software systems and computational methods. 2025. URL: https://cyberleninka.ru/article/n/modelirovanie-masshtabiruemyh-mikroservisnyh-sistem-v-usloviyah-konteynernoy-virtualizatsii-s-ispolzovaniem-kubernetes-i-service (date of access: 10.04.2026).
6. KubeFed User Guide [Electronic resource] // Kubernetes SIG Multicluster. URL: https://github.com/kubernetes-retired/kubefed/tree/master/docs/userguide (date of access: 10.04.2026).
7. Rancher Documentation. Architecture and Fleet [Electronic resource]. URL: https://ranchermanager.docs.rancher.com/ (date of access: 10.04.2026).
8. Cluster API Book. Introduction [Electronic resource]. URL: https://cluster-api.sigs.k8s.io/ (date of access: 10.04.2026).
9. Chikaleva Yu.S. Analysis of Microservice Granularity: Efficiency of Architectural Approaches // Software Systems and Computational Methods. 2025. URL: https://cyberleninka.ru/article/n/analiz-granulyarnosti-mikroservisov-effektivnost-arhitekturnyh-podhodov (accessed: 10.04.2026).
10. K3s Documentation. Lightweight Kubernetes [Electronic resource]. URL: https://docs.k3s.io/ (accessed: 10.04.2026).
11. Donenfeld J.A. WireGuard: Next Generation Kernel Network Tunnel // Proceedings of the Network and Distributed System Security Symposium. 2017. URL: https://www.ndss-symposium.org/wp-content/uploads/2017/09/ndss2017_04A-3_Donenfeld_paper.pdf (date accessed: 10.04.2026).
12. Install Multicluster [Electronic resource] // Istio Documentation. URL: https://istio.io/latest/docs/setup/install/multicluster/ (date accessed: 10.04.2026).
13. Myasnikov I.V. Monitoring Istio Components to Ensure Service Mesh Reliability and Observability // Science Herald. 2025. URL: https://cyberleninka.ru/article/n/monitoring-komponentov-istio-dlya-obespecheniya-nadezhnosti-i-nablyudaemosti-service-mesh (date of access: 10.04.2026).
14. Kornienko D.V., Nikulin A.V. Visualization of architectures of information systems based on microservices using OpenTelemetry data // Computational nanotechnology. 2024. URL: https://cyberleninka.ru/article/n/vizualizatsiya-arhitektur-informatsionnyh-sistem-

osnovannyh-na-mikroservisah-s-ispolzovaniem-dannyh-opentelemetry (date of access: 10.04.2026).

15. Kovtun D.P., Laponina O.R. Using attribute-based access control and mTLS in microservice architecture // International Journal of Open Information Technologies. 2025. URL: https://cyberleninka.ru/article/n/ispolzovanie-upravleniya-dostupom-na-osnove-atributov-i-mtls-v-mikroservisnoy-arhitekture (date of access: 10.04.2026).