Содержание

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги, и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com на странице с описанием соответствующей книги.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии данной книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и O'Reilly очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли применить санкции.

Пожалуйста, свяжитесь с нами по адресу dmkpress@gmail.com со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, помогающую нам предоставлять вам качественные материалы.

Об авторах

Ли Калькот – лидер в области инновационных продуктов и технологий, стремящийся обеспечить инженеров эффективными и действенными решениями. Являясь основателем Layer5, Ли находится в авангарде движения за развитие облачных технологий. Работая в компаниях SolarWinds, Seagate, Cisco и Schneider Electric, он постоянно уделяет внимание открытому исходному коду, передовым и новейшим технологиям. Советник, автор и оратор, Ли активно работает в сообществе как Docker Captain, Cloud Native Ambassador и Google Summer of Code Mentor.

Зак Бутчер – инженер-основатель компании Tetrate и основной разработчик проекта Istio. Его всегда привлекали трудные задачи, начиная с разработки веб-приложений для IE6 и заканчивая управлением сервисами, контролем доступа и иерархической организацией ресурсов Google Cloud Platform. Tetrate – небольшая компания, и Зак выполняет в ней широкий круг обязанностей, в том числе как системного архитектора, коммерсанта, автора и оратора.

Колофон

На обложке книги «*Istio: запуск и работа*» изображена вилохвостая чайка (*Xema sabini*). После завершения сезона размножения в арктической тундре на севере Аляски, в Нунавате и Гренландии популяции этой маленькой чайки рассеиваются по прибрежным районам на севере Южной Америки и Юго-Западной Африки, богатым кормом.

У взрослых особей темно-серая голова с черным кольцом в основании шеи; удлиненные крылья сверху бледно-серые с черным передним краем и треугольником белого цвета в центре. У взрослых особей красные глаза, темно-серые лапки, а также черный клюв с желтым кончиком. Молодые птицы первого года жизни имеют более тусклое оперение с коричневым окрасом. Длина птиц в среднем чуть больше 30 см (маленькая по сравнению с более привычной серебристой чайкой со средней длиной полметра).

В период гнездования эта маленькая чайка питается насекомыми, но в остальное время года, проводимое в прибрежных водах, ее рацион состоит в основном из мелких рыб, ракообразных и планктона. Эти птицы тянутся к местам, где восходящие холодные течения, такие как Перуанское течение у северо-западного побережья Перу и Бенгела, поднимающиеся к юго-западу от Африки, выносят на поверхность глубоководные питательные вещества, создавая питательный рай для многих видов морской флоры и фауны.

Название этой птице дал английский ученый Уильям Элфорд Лич в честь Эдварда Сабина, впервые описавшего ее. Лич посчитал эту птицу достаточно уникальной, чтобы отнести ее к отдельному роду (она остается единственным членом рода *Xema*), однако некоторые до сих пор оспаривают это мнение. Будучи важной фигурой в британской зоологии и таксономии начала XIX в., Лич был известен экстравагантным и нестандартным подходом к выбору названий, иногда применяя анаграммы из имен коллег и знакомых. Хотя он также использовал имена из классической мифологии, что в бóльшей степени соответствует культурным и научным традициям.

Многие животные, изображенные на обложках O’Reilly, находятся под угрозой полного исчезновения; все они важны для мира.

Иллюстрацию для обложки нарисовала Карен Монтгомери, основываясь на черно-белых гравюрах из каталога *British Birds*.

Введение

Кому стоить прочесть эту книгу?

Сервисная сетка (service mesh) является важным инструментом организации любой облачной инфраструктуры. Эта книга предназначена для тех, кто хочет начать работать с Istio. Хорошо, если читатель уже знаком с Docker и Kubernetes, но для изучения Istio по этой книге вполне достаточно базовых знаний о работе сети и Linux. Знание языка программирования Go или другого не требуется и не ожидается.

Здесь описаны многие облачные инструменты итехнологии, такие как Prometheus, Jaeger, Grafana, Meshery, Envoy и OpenTracing. Знакомство читателя с ними было бы идеальным, но для усвоения содержимого этой книги достаточно упомянутых выше базовых знаний.

Почему мы написали эту книгу?

Эпоха сервисных сеток выводит на новый уровень интеллектуальные сетевые сервисы, меняющие архитектуру современных приложений и увеличивающие надежность обслуживания. Istio – лишь одна из многих сервисных сеток, но, обладая огромным набором функций и возможностей, нуждается в исчерпывающем руководстве.

Цель этой книги – рассказать шаг за шагом, как начать работать с Istio. Она проведет вас за собой, крепко держа за руку. Все понятия описываются в четком логическом порядке, когда объяснение каждого следующего понятия основывается на предыдущих. Учитывая сложность обсуждаемой темы и активность сообщества, книга просто не в состоянии охватить все возможные варианты использования, поэтому акцент сделан на основных структурных элементах и неустаревающих аспектах проекта. По мере необходимости указываются дополнительные ресурсы.

Прочитав «*Istio: запуск и работа*», вы познакомитесь с основными возможностями Istio и сможете уверенно развертывать ее в своих облачных окружениях.

Соглашения по оформлению

В этой книге используются следующие соглашения по оформлению:

Курсив
: Используется для обозначения новых терминов, адресов URL и электронной почты, имен файлов и расширений имен файлов.

`Моноширинный шрифт`

Применяется для оформления листингов программ и программных элементов внутри обычного текста, таких как имена переменных и функций, баз данных, типов данных, переменных окружения, инструкций и ключевых слов.

`Моноширинный полужирный`

Обозначает команды или другой текст, который должен вводиться пользователем.

`Моноширинный курсив`

Обозначает текст, который должен замещаться фактическими значениями, вводимыми пользователем или определяемыми из контекста.

Так выделяются советы и предложения.

Так обозначаются советы, предложения и примечания общего характера.

Так обозначаются предупреждения и предостережения.

Благодарности

Спасибо Никки Макдональд (Nikki McDonald), Джону Дэвинсу (John Devins), Вирджинии Уилсон (Virginia Wilson), Корбину Коллинзу (Corbin Collins), Деборе Бейкер (Deborah Baker) и остальной команде O'Reilly.

И отдельное спасибо всем, кто рецензировал нашу рукопись по мере сборки этой книги, особенно нашим техническим рецензентам Майлсу Штайнхаузеру (Myles Steinhauser), Гириш Ранганатан (Girish Ranganathan) и Джесс Мэйлс (Jess Males).

Ли хотел бы лично сказать:

«Джилл, твои стойкость и любовь – моя опора в жизни. Ты подарила мне величайшую ценность – наших двойняшек.

Доктор Джи, мой друг, путешествие только началось. Спасибо, что сопровождали меня.

Кит, я жажду встречи с тобой, чтобы вновь испытать радость настоящей мужской дружбы».

Глава 1

Введение в сервисные сетки

Что такое сервисная сетка?

Сервисные сетки обеспечивают обслуживание сетевых рабочих нагрузок на основе политик, гарантируя требуемое поведение сети при постоянном изменении условий и топологии. Изменяются нагрузки, конфигурации, ресурсы (включая те, которые влияют на инфраструктуру и топологию приложений, в том числе внутри- и межкластерные ресурсы, то появляющиеся, то исчезающие), и развертываются новые рабочие нагрузки.

Основы

Сервисные сетки – это адресуемый инфраструктурный уровень, позволяющий управлять как модернизацией существующих монолитных (или иных) рабочих нагрузок, так и разрастанием микросервисов. Сервисные сетки – это задействованный в полную силу адресуемый инфраструктурный уровень. Они выгодны в монолитных средах, но главной причиной их появления является быстрое развитие микросервисов и контейнеров – истинно облачного подхода к организации масштабируемых и независимых сервисов. Микросервисы превратили внутренние коммуникации приложений в сетку, сплетенную из вызовов удаленных процедур (RPC) между сервисами, передаваемых по сетям. Среди преимуществ микросервисов – демократизация выбора языка и технологий в независимых сервисных группах – командах, быстро создающих новые функции, итеративно и непрерывно поставляющих программное обеспечение (как правило, в виде сервисов).

Область сетевых взаимодействий очень обширна. Неудивительно, что существует много тонких, почти незаметных различий между похожими понятиями. По своей сути сервисная сетка – это сеть, конструируемая разработчиками исключительно для взаимодействий между сервисами: она избавляет разработчиков от необходимости заниматься сетевыми проблемами (например, обеспечением надежности) и дает администраторам возможность декларативно определять поведение сети, идентификацию узлов и политики управления трафиком.

Это может выглядеть как реинкарнация программно определяемой сети (SDN), но сервисные сетки отличаются в первую очередь ориентацией на разработчика, а не на администратора сети. По бóльшей части сегодняшние сер-

висные сетки полностью основаны на программном обеспечении (хотя аппаратные реализации могут появиться в будущем). Термин «*целевой нетворкинг*» (intent-based networking) используется в основном в физических сетях, но, учитывая декларативный контроль на основе политик, предоставляемый сервисными сетками, справедливо сравнить их с истинно облачными SDN. На рис. 1.1 показана обобщенная архитектура сервисной сетки (в главе 2 мы описываем, что значит быть истинно облачным).

Рис. 1.1 ❖ Если нет уровня управления, это не сервисная сетка

Сервисные сетки строятся с использованием прокси сервисов. Прокси сервисов находятся на уровне данных и передают трафик. Трафик прозрачно перехватывается с помощью правил iptables в пространстве имен подов (pod – группа контейнеров).

Такой унифицированный слой инфраструктуры в сочетании с развернутыми сервисами обычно называют *сервисной сеткой* (*service mesh*). Istio превращает разрозненные микросервисы в интегрированную сервисную сетку, внедряя прокси сервисов во все сетевые пути, устанавливая соединения между прокси и ставя их под централизованный контроль, таким образом формируя сервисную сетку.

Путешествие в сервисную сетку

Не важно, чем вы занимаетесь: управляете ли флотилией микросервисов или модернизируете существующие неконтейнерные сервисы, рано или поздно вы все равно окажетесь перед необходимостью организации сервисной сетки. Чем больше будет развернуто микросервисов, тем быстрее вы окажетесь в этой ситуации.

Клиентские библиотеки: первые сервисные сетки?

Чтобы справиться со сложной задачей управления микросервисами, некоторые компании в качестве основы для стандартизации разработки начали использовать *клиентские библиотеки*. Некоторые считают эти библиотеки первыми сервисными сетками. Использование библиотеки требует, чтобы в архитектуре был прикладной код, расширяющий или использующий примитивы выбранных библиотек, как показано на рис. 1.2. Кроме того, архитектура должна учитывать потенциальное использование фреймворков и/или серверов приложений для разных языков программирования.

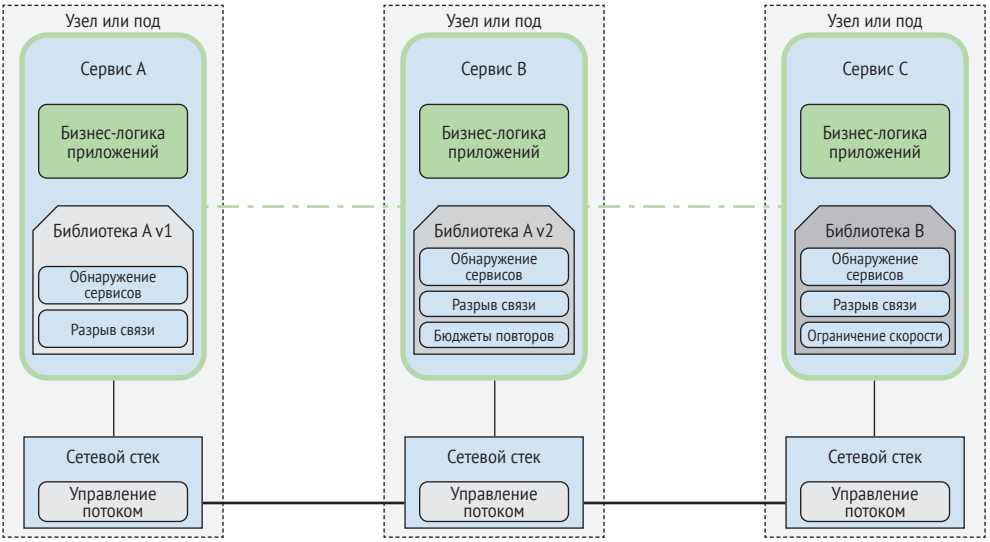

Рис. 1.2 ❖ Сервисная архитектура использует клиентские библиотеки, связанные с логикой приложений

Выгода создания клиентских библиотек, во-первых, состоит в том, что потребляемые ресурсы учитываются локально по каждому конкретному сервису, и, во-вторых, разработчики могут самостоятельно выбрать существующую или создать новую библиотеку для определенного языка. Однако недостатки использования клиентских библиотек со временем привели к появлению сервисных сеток. Наиболее значительным недостатком библиотек является тесная связь инфраструктуры с прикладным кодом. Неоднородный дизайн клиентских библиотек, часто зависящий от языка, делает их функциональность и поведение непоследовательными, что сужает возможности мониторинга, требует применения специфических методов для расширения сервисов, часто сильно зависящих друг от друга, и влечет потенциальные риски безопасности. Такие специфические *библиотеки отказоустойчивости* (*resilience libraries*) могут оказаться слишком дорогостоящими для широкого использования в организациях из-за сложности внедрения в действующие приложения или полной нецелесообразности интеграции в существующие архитектуры.

Сетевые взаимодействия – это сложно. Создание клиентской библиотеки, устраняющей конфликты между клиентами путем добавления случайных задержек и использования экспоненциального алгоритма расчета времени следующей повторной попытки – сложная задача, и не всегда удается обеспечить одинаковое поведение различных клиентских библиотек (на разных языках и в разных версиях этих библиотек). Координация обновления клиентских библиотек затруднена в больших окружениях, поскольку обновления требуют внесения изменений в код, установки новой версии приложения и, возможно, простоя приложения.

Рисунок 1.3 показывает, как размещение прокси возле каждого экземпляра приложения избавляет от необходимости иметь специфические библиотеки отказоустойчивости для разрыва цепи, тайм-аутов, повторных попыток, обнаружения сервисов, балансировки нагрузки и т. д. Сервисные сетки дают организациям, внедряющим микросервисы, возможность использовать лучшие фреймворки и языки и избавляют от хлопот с выбором библиотек и шаблонов проектирования для каждой конкретной платформы.

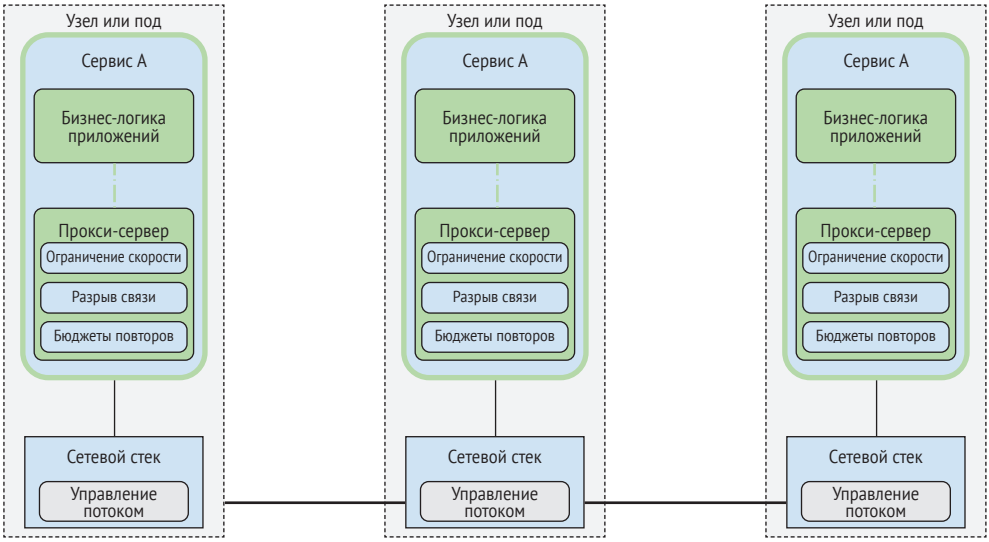

Рис. 1.3 ❖ Архитектура сервисов, использующих прокси, отделенные от логики приложений

Зачем они нужны?

Здесь можно задаться вопросом: «Есть же оркестратор контейнеров. Зачем городить еще один инфраструктурный уровень?» Сегодня, при массовом использовании микросервисов и контейнеров, оркестраторы контейнеров обеспечивают бóльшую часть потребностей кластера (узлов и контейнеров). Они осуществляют функции планирования, обнаружения и поддержания работоспособности главным образом на уровне инфраструктуры (что и требуется), оставляя неудовлетворенными потребности на уровне сервисов. Сервисная

сетка – это выделенный слой инфраструктуры для обеспечения безопасной, быстрой и надежной связи между сервисами, иногда зависящий от оркестратора контейнера или интеграции с другой системой обнаружения сервисов. Сервисные сетки могут развертываться отдельным уровнем поверх оркестраторов контейнеров, но не требуют их, поскольку компоненты уровней управления и данных могут быть развернуты независимо от инфраструктуры контейнеров. В главе 3 мы увидим, что агент узла (включая прокси), как компонент уровня данных, часто используется в неконтейнерных окружениях.

Сервисная сетка Istio обычно адаптируется под конкретные потребности. Сотрудники организаций, с которыми мы беседовали, внедряют сервисные сетки в первую очередь для контроля с применением средств мониторинга сетевого трафика. Многие учреждения, особенно финансовые, используют сервисные сетки прежде всего для управления шифрованием межсервисного трафика.

Что бы ни было катализатором, организации внедряют их сломя голову... Сервисные сетки полезны не только в истинно облачных окружениях, они помогают решать сложные задачи эксплуатации микросервисов. Многие организации, использующие монолитные сервисы (работающие на физических или виртуальных машинах, локально или за пределами организации), остро желают внедрения сервисных сеток, потому что это позволит ускорить модернизацию существующих архитектур.

На рис. 1.4 показаны возможности оркестраторов контейнеров (звездочками отмечены наиболее важные из них). Сервисные сетки, как правило, полагаются на нижележащие слои. В данном случае нижний уровень образуют оркестраторы контейнеров.

Разве на контейнерных платформах такого еще нет?

Контейнеры предлагают простой и универсальный механизм упаковки приложений, не зависящий от выбранного языка и обеспечивающий управление их жизненным циклом . Будучи универсальными по своей природе, *оркестраторы контейнеров* отвечают за формирование кластеров, эффективное распределение своих ресурсов и высокоуровневое управление приложениями (развертывание, обслуживание, оценка близости/удаленности, проверка исправности, масштабирование и т. д.). Как показано на рис. 1.4, оркестраторы имеют механизмы обнаружения сервисов и балансировки нагрузки со встроенными виртуальными IP-адресами. Поддерживаемые алгоритмы балансировки нагрузки, как правило, просты по своей природе (циклические, случайные) и действуют под одним виртуальным IP-адресом для взаимодействия с внутренними подами.

Kubernetes занимается регистрацией/вытеснением экземпляров в группе на основании их работоспособности и соответствия предикату группы (меткам и селекторам). Далее, сервисы могут использовать DNS для обнаружения сервисов и балансировки нагрузки вне зависимости от их реализации. Нет необходимости в специальных библиотеках, зависящих от языка, или в регистрации. Контейнерные оркестраторы позволили переместить рутинные сетевые задачи из приложений в инфраструктуру, освободив общую технологическую экосистему инфраструктуры и переместив акцент на более высокие уровни.

Рис. 1.4 ❖ Возможности и фокус оркестраторов контейнеров в сравнении с потребностями уровня сервисов

Теперь ясно, как сервисные сетки дополняют нижележащие слои. Перейдем к другим слоям.

Ландшафт и экосистема

Ландшафт сервисных сеток (https://oreil.ly/57P0j) представляет собой растущую экосистему инструментов, не относящуюся к истинно облачным приложениям; на самом деле он также обеспечивает большое преимущество для неконтейнеризированных, немикросервисных нагрузок. По мере осмысления преимуществ и роли, которую сервисная сетка играет в развертывании, можно начинать выбор сетки и ее интеграцию с имеющимися инструментами.

Ландшафт

Как выбрать сервисную сетку? Большое разнообразие доступных в настоящее время сервисных сеток не позволяет людям легко определить, что на самом

деле является сервисной сеткой, а что – нет. Со временем они обретают все больше похожих возможностей, что облегчает их описание и сравнение.

Интересно, но не удивительно, что многие сервисные сетки основаны на одних и тех же прокси, таких как Envoy и NGINX.

Экосистема

Как сервисная сетка соотносится с другими технологическими экосистемами, мы уже видели на примере клиентских библиотек и оркестраторов контейнеров. API-шлюзы удовлетворяют ряд схожих потребностей и обычно развертываются в оркестраторах в качестве пограничного прокси. Пограничные прокси предоставляют управление уровнями с 4 (L4) по 7 (L7) и используют оркестраторы контейнеров для обеспечения надежности, доступности и масштабируемости контейнерной инфраструктуры.

API-шлюзы взаимодействуют с сервисными сетками способом, озадачивающим многих, поскольку API-шлюзы (и прокси, на которых они основаны) варьируются от традиционных до облачных API-шлюзов и API-шлюзов микросервисов. Последние могут быть представлены коллекцией API-шлюзов с открытым исходным кодом для микросервисов, которые обертывают существующие прокси уровня L7, интегрированные с оркестраторами контейнеров и средствами самообслуживания разработчика (например, HAProxy, Traefik, NGINX или Envoy).

Главной задачей API-шлюзов в сервисных сетках является прием трафика извне и его распределение внутри. API-шлюзы образуют управляемый API для доступа к сервисам и ориентированы на передачу вертикального трафика (входящего и исходящего из сервисной сетки). Они не так хорошо подходят для управления горизонтальным трафиком (внутри сервисной сетки), так как им требуется, чтобы трафик проходил через центральный прокси, а это добавляет лишний сетевой переход. Сервисные сетки, напротив, предназначены в первую очередь для управления горизонтальным трафиком внутри сервисной сетки.

Учитывая их взаимодополняющий характер, API-шлюзы и сервисные сетки часто устанавливаются совместно. API-шлюзы используют другие функции системы управления для работы с аналитикой, бизнес-данными, вспомогательными сервисами и механизмами управления версиями. Сегодня существуют как перекрытие, так и разрыв между возможностями сервисных сеток, API-шлюзов и систем управления. Сервисные сетки по мере развития получают новые возможности, и перекрытие областей применения увеличивается.

Критическая, ненадежная сеть

Как уже отмечалось, в комплексе действующих микросервисов сеть непосредственно вовлечена в каждую транзакцию, в каждое обращение к бизнес-логике и в каждый запрос, сделанный к приложению. Надежность сети и задержки являются одними из главных проблем современных облачных приложений. Одно истинно облачное приложение может включать сотни микросервисов,

со множеством экземпляров каждого, постоянно меняющихся оркестратором контейнеров по расписанию.

Учитывая центральную роль сети, желательно, чтобы она была как можно более интеллектуальной и отказоустойчивой.

Сеть должна:

- маршрутизировать трафик в обход отказов для повышения совокупной надежности кластера;
- избегать нежелательных издержек, возникающих, например, при выборе маршрутов с высокой задержкой или серверов с холодным кешем;
- обеспечить защиту межсервисного трафика от тривиальной атаки;
- помогать в выявлении проблем, выделяя неожиданные зависимости и первопричины сбоев в коммуникациях;
- разрешать определять политики не только на уровне соединений, но и на уровне поведения сервисов.

Однако вряд ли разработчик будет гореть желанием вносить всю эту логику в свое приложение.

Необходимо управление на уровне L5, то есть сеть, ориентированная на сервисы; проще говоря – сервисная сетка.

Преимущества сервисной сетки

Сервисные сетки предлагают единообразие способов подключения, защиты, управления и мониторинга микросервисов.

Наблюдаемость

Сервисные сетки обеспечивают видимость, отказоустойчивость и контроль трафика, а также контроль безопасности распределенных сервисов приложений. Это весомые преимущества. Сервисные сетки разворачиваются прозрачно и обеспечивают видимость и контроль трафика без необходимости внесения каких-либо изменений в код приложения (более подробно см. главу 2).

В текущем, первом поколении сервисные сетки имеют большой потенциал, в том числе и Istio. Остается подождать и посмотреть, какие появятся возможности у второго поколения, когда сервисные сетки будут использоваться так же широко, как контейнеры и их оркестраторы.

Управление трафиком

Сервисные сетки обеспечивают детальный, декларативный контроль над сетевым трафиком, например позволяя определить направление запроса на выполнение канареечного развертывания. В число функций поддержки надежности обычно входят: разрыв цепи, балансировка нагрузки с учетом задержек, согласованное обнаружение сервисов, повторы, тайм-ауты и критические сроки (более подробно см. главу 8).

Безопасность

В лице сервисных сеток организации получают мощный инструмент управления безопасностью, политиками и требованиями. Большинство сервисных