

УДК 004.451Istio
ББК 32.972.1
К17

Калькот Л., Бутчер З.

К17 Istio: приступаем к работе / пер. с англ. А. Л. Бриня. – М.: ДМК Пресс, 2020. – 236 с.: ил.

ISBN 978-5-97060-863-0

Вне зависимости от того, управляете ли вы микросервисами или модернизируете существующие неконтейнерные сервисы, рано или поздно вы все равно окажетесь перед необходимостью организации сервисной сетки. Этот момент наступит тем быстрее, чем больше будет развернуто микросервисов.

В предлагаемой вашему вниманию книге Ли Калькот и Зак Бутчер показывают, как сервисная сетка Istio вписывается в жизненный цикл распределенного приложения. Вы изучите ее архитектуру, узнаете об инструментах и API для управления многими функциями Istio, рассмотрите вопросы безопасности и управления трафиком. Особое внимание уделяется устранению неисправностей и отладке.

Книга предназначена для IT-специалистов, в задачу которых входит обеспечение безопасной, быстрой и надежной связи между сервисами.

УДК 004.451Istio
ББК 32.972.1

Authorized Russian translation of the English edition of Istio: Up and Running ISBN 9781492043782 © 2020 Lee Calcote and Zack Butcher This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same. Russian language edition copyright © 2020 by DMK Press. All rights reserved.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

ISBN 978-1-492-04378-2 (англ.)

ISBN 978-5-97060-863-0 (рус.)

Copyright © Lee Calcote and Zack
Butcher, 2020

© Оформление, издание, перевод,
ДМК Пресс, 2020

Содержание

От издательства	10
Об авторах	11
Колофон	12
Введение	13
Благодарности	15
Глава 1. Введение в сервисные сетки	16
Что такое сервисная сетка?	16
Основы	16
Путешествие в сервисную сетку	17
Клиентские библиотеки: первые сервисные сетки?	18
Зачем они нужны?	19
Разве на контейнерных платформах такого еще нет?	20
Ландшафт и экосистема	21
Ландшафт	21
Экосистема	22
Критическая, ненадежная сеть	22
Преимущества сервисной сетки	23
Сервисная сетка Istio	25
Происхождение Istio	26
Текущее состояние Istio	26
Активность развития	27
Выпуски	28
Классификация версий	29
Будущее	29
То, чем Istio не является	30
Речь идет не только о микросервисах	30
Терминология	31
Глава 2. Истинно облачный подход к равномерной наблюдаемости	34
Что значит быть <i>истинно облачным</i> ?	34
Путь к истинной облачности	35
Упаковка и развертывание	37
Архитектура приложений	37
Процессы разработки и эксплуатации	38
Истинно облачная инфраструктура	38
Что такое наблюдаемость?	39
Источники телеметрии	40
Журналы	40

Метрики	41
Трассировка	41
Комбинирование источников телеметрии.....	42
Почему так важна наблюдаемость в распределенных системах?.....	43
Равномерная наблюдаемость с сервисной сеткой	44
Клиентские библиотеки.....	45
Взаимодействие с системами мониторинга	45
Глава 3. Istio на первый взгляд.....	47
Архитектура сервисной сетки.....	47
Уровни	48
Компоненты уровня управления Istio	49
Прокси сервисов.....	51
Компоненты уровня данных Istio	52
Шлюзы.....	53
Расширяемость	56
Замена прокси	56
Замена адаптеров.....	57
Масштабируемость и производительность	58
Модели развертывания	59
Глава 4. Развертывание Istio	61
Подготовка окружения для Istio	61
Docker Desktop как среда установки	61
Конфигурирование Docker Desktop	62
Установка Istio	65
Параметры установки Istio.....	67
Регистрация нестандартных ресурсов Istio.....	68
Установка компонентов уровня управления	70
Развертывание образца приложения Bookinfo	73
Развертывание примера приложения с автоматическим внедрением прокси	74
Работа примера приложения в сети	76
Деинсталляция Istio.....	77
Установка с помощью Helm	77
Установка Helm.....	77
Установка с помощью Helm.....	78
Проверка сетки после установки	79
Деинсталляция с помощью Helm.....	79
Другие окружения.....	79
Глава 5. Прокси для сервисов	80
Что такое прокси для сервисов?	81
Коротко о iptables.....	82
Обзор Envoy Proxy.....	83
Почему Envoy?.....	84
Envoy в Istio	85
Внедрение в сетку.....	85
Внедрение вручную	86
Выборочное внедрение.....	88
Автоматическое внедрение.....	88

Init-контейнеры в Kubernetes.....	91
Выделение ресурсов для прокси	91
Функциональные возможности Envoy	91
Основные конструкции	92
Сертификаты и защита трафика.....	93
Глава 6. Безопасность и идентичность.....	96
Контроль доступа.....	97
Аутентификация	97
Авторизация.....	97
Идентичность.....	98
SPIFFE.....	98
Архитектура управления ключами	100
Citadel.....	101
Агенты узлов	102
Envoy	103
Pilot	104
mTLS	104
Настройка политик аутентификации и авторизации в Istio.....	105
Политика аутентификации: конфигурирование mTLS	105
Политика авторизации: настройка разрешений	108
Глава 7. Pilot.....	111
Настройка Pilot	111
Конфигурация сетки.....	111
Сетевая конфигурация.....	113
Обнаружение сервисов	114
Обслуживание конфигурации	114
Отладка и устранение неисправностей в Pilot	116
istioctl	116
Отладка Pilot.....	117
Трассировка конфигурации.....	119
Приемники	119
Маршруты.....	122
Кластеры	124
Глава 8. Управление трафиком	127
Как движется трафик в Istio?	127
Работа сетевых API Istio	128
ServiceEntry.....	129
DestinationRule	132
VirtualService	135
Gateway	139
Управление трафиком и маршрутизация.....	147
Устойчивость.....	152
Стратегия балансировки нагрузки	153
Обнаружение аномалий	154
Повторные попытки	154
Тайм-ауты.....	155
Имитация ошибок.....	156

Входные и выходные шлюзы	157
Входной шлюз	158
Выходной шлюз	158
Глава 9. Mixer и политика в сетке	161
Архитектура	161
Обеспечение политики	163
Как работают политики Mixer	164
Передача телеметрии	165
Атрибуты	166
Отправка отчетов	167
Кэширование результатов проверок	167
Адаптеры	167
Внутрипроцессные адаптеры	168
Внепроцессные адаптеры	168
Создание политики Mixer и использование адаптеров	169
Конфигурация Mixer	169
Адаптер открытого агента политик	170
Адаптер Prometheus	171
Глава 10. Телеметрия	175
Модели адаптеров	175
Отчеты телеметрии	176
Метрики	176
Настройка Mixer для сбора метрик	176
Настройка сбора и запроса метрик	177
Трассировка	178
Отключение трассировки	180
Журналы	181
Метрики	183
Визуализация	184
Глава 11. Отладка Istio	185
Поддержка интроспекции в компонентах Istio	185
Отладка с использованием уровня администрирования	186
С использованием kubectl	187
Готовность рабочих нагрузок	189
Конфигурация приложения	189
Сетевой трафик и порты	189
Сервисы и развертывание	190
Поды	191
Istio: установка, обновление и удаление	191
Установка	192
Обновление	192
Отладка Mixer	193
Отладка Pilot	194
Отладка Galley	194
Отладка Envoy	195
Административная консоль Envoy	196
Ответы 503 или 404	196

Внедрение прокси	196
Совместимость версий	198
Глава 12. Вопросы развертывания приложений	199
Соображения об уровне управления	199
Galley	200
Pilot	202
Mixer	204
Citadel	207
Пример из практики: канареечное развертывание	208
Кросс-кластерное развертывание	214
Глава 13. Продвинутое сценарии	216
Типы продвинутых топологий	216
Однокластерные сетки	216
Мультикластерные сетки	217
Рабочие примеры	220
Выбор топологии	221
Кросс-кластер или мультикластер?	221
Настройка кросс-кластера	224
Настройка DNS и развертывание Bookinfo	226
Предметный указатель	232

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.

Введение

Кому стоит 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 и технологии, 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 сетей выводит на новый уровень интеллектуальные сетевые сервисы, меняющие архитектуру современных приложений и увеличивающие надежность обслуживания. Istio – лишь одна из многих сервисных сетей, 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 разворачивать 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.

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 большей 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.

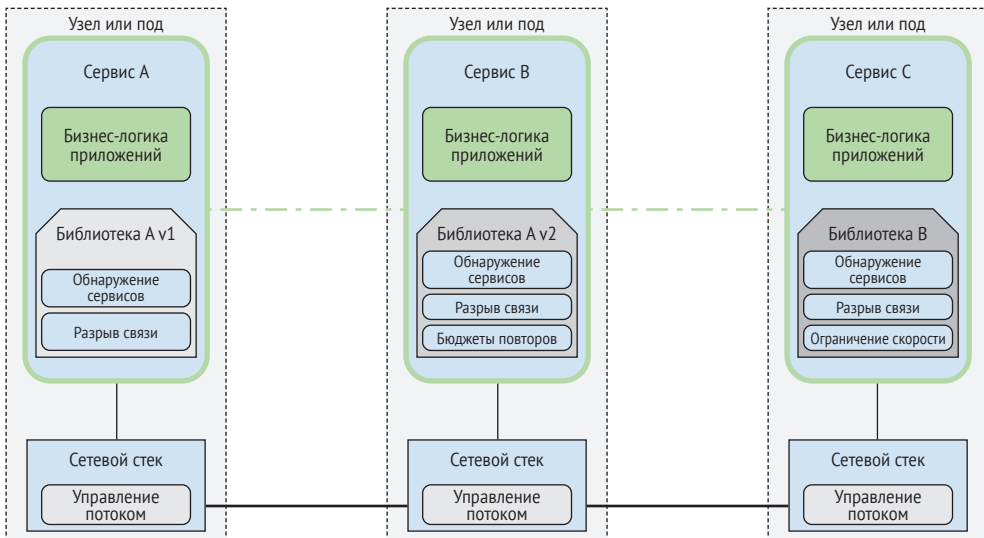


Рис. 1.2 ❖ Сервисная архитектура использует клиентские библиотеки, связанные с логикой приложений

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 удастся 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.

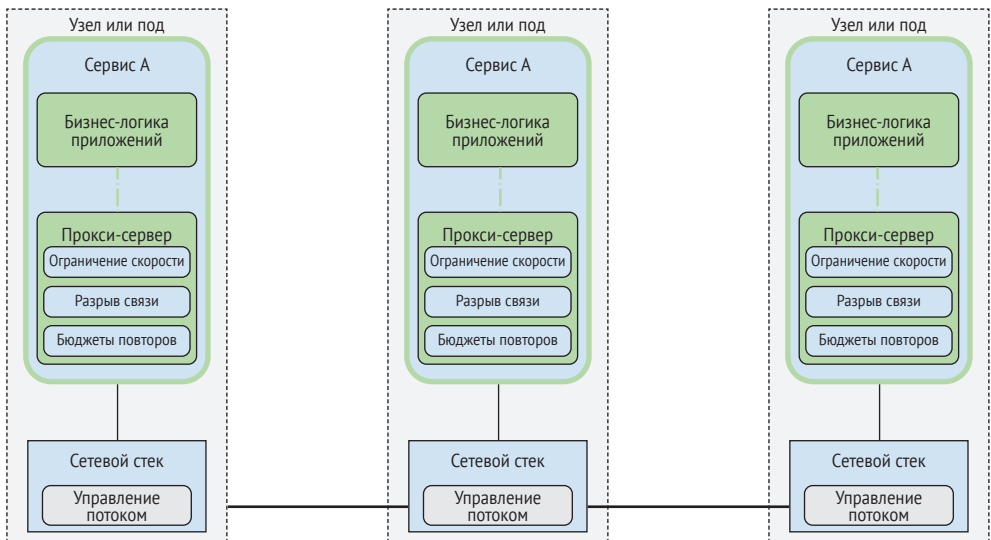


Рис. 1.3 ❖ Архитектура сервисов, использующих прокси, отделенные от логики приложений

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 большую 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 сетей, 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 неконтнеризированных, 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

сеток предоставляют центр сертификации (CA, Certificate Authority) для управления ключами и сертификатами в обеспечение связи между сервисами. Присвоение каждому сервису в сетке проверяемой идентичности является ключом к определению клиентов, имеющих право выполнять запросы к различным сервисам, а также для шифрования трафика, порождаемого этими запросами. Сертификаты генерируются для каждого сервиса и представляют его уникальную идентичность. Обычно для идентификации сервисов и управления жизненным циклом сертификатов (генерация, распределение, обновление и отзыв) от их имени используются соответствующие прокси (подробнее об этом см. главу 6).

Модернизация существующей инфраструктуры (реновация развертывания)

Многие считают, что при небольшом количестве сервисов нет смысла добавлять сервисную сетку в свою архитектуру. Это ошибочное суждение. Сервисные сетки остаются ценным инструментом независимо от числа сервисов. Просто их ценность увеличивается с ростом числа запущенных сервисов и мест их развертывания.

Некоторые новые проекты имеют роскошную возможность внедрить сервисную сетку с самого начала, но большинству организаций придется перестраивать существующие сервисы (монолитные или другие) для этого. Имеющиеся сервисы могут работать не в контейнерах, а на виртуальных или физических машинах. Сервисные сетки помогают в модернизации, позволяя организациям обойтись без переписывания приложений, внедрения микросервисов и новых языков или перехода на облако.

Для разбиения монолитов можно использовать *фасадные сервисы*. Также можно использовать *шаблон Заслонка* (Strangler), чтобы окружить устаревший монолит сервисами с более дружелюбным API.

Внедрение сервисной сетки дает организациям поддержку наблюдаемости (например, метрики, журналы и трассировку), а также оптимизацию зависимостей и обслуживания каждого сервиса (микросервиса или обычного). Например, для трассировки в сервисы потребуется добавить только пересылку определенных HTTP-заголовков. Сервисные сетки полезны для дооснащения существующих инфраструктур единообразными и универсальными средствами мониторинга с минимальным количеством изменений кода.

Развязка на уровне 5

Важными факторами, которые необходимо учитывать при оценке ценности сервисной сетки, являются устранение зависимости между командами разработчиков и увеличение скорости доставки, как показано на рис. 1.5.

Подобно тому как микросервисы помогают разделить разработчиков на команды по функциональным направлениям, сервисные сетки помогают отделить администраторов от процессов разработки и выпуска приложений, предоставляя администраторам декларативный контроль над работой сервисов. Сервисная сетка не просто отделяет команды друг от друга – она исклю-

чает размывание ответственности между ними и обеспечивает единообразие рабочих стандартов между организациями в одной отрасли.

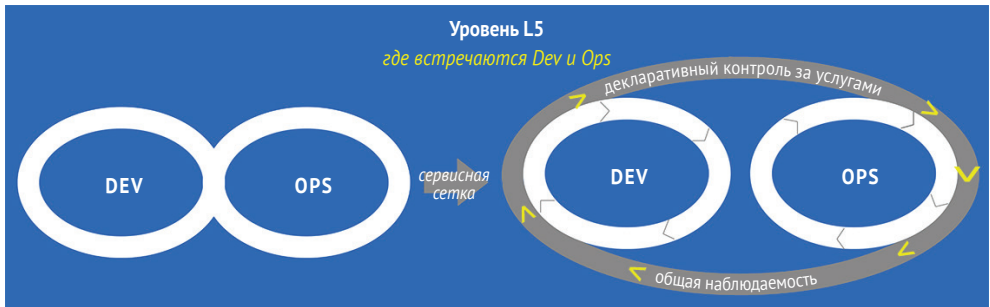


Рис. 1.5 ❖ Уровень 5 (L5), где встречаются Dev (разработчики) и Ops (операторы)

Рассмотрим следующий список заданий:

- определить, когда разрывать связь и упростить процесс;
- установить предельные сроки обслуживания;
- гарантировать генерирование распределенных трассировок и их передачу в системы мониторинга;
- запретить пользователям аккаунта «Рога и копыта» доступ к бета-версиям сервисов.

Кто несет ответственность за их реализацию – разработчик или администратор? Ответы, вероятно, будут зависеть от организации; в настоящее время нет общепринятой практики. Сервисные сетки помогают исключить распыление обязанностей или ситуации, когда одна команда обвиняет другую в недостаточной ответственности.

СЕРВИСНАЯ СЕТКА ISTIO

Начнем рассмотрение сервисной сетки Istio.

Этимология проекта

Поскольку фреймворк Kubernetes является значимой частью облачной экосистемы, его греческое этимологическое наследие привело к распространению проектов с названиями, начинающимися с «К», и других букв, ассоциирующихся с известными греческими совами. *Kubernetes* (греч. κυβερνήτης (<https://oreil.ly/azC7B>)) – греческое слово «рулевой» или «пилот», однокоренное со словами «кибернетика» и «губернаторский», имеющими важное значение в теории управления, а контроль находится в центре внимания Kubernetes.

Istio (греч. ἴστιο) – это греческое слово «парус» и произносится как *ис-ти-о*. И конечно же, название интерфейса командной строки (CLI) Istio, *istioctl*, произносится как *ис-ти-о-си-ети-эль*, потому что он используется для управления Istio, а не для шуток.

Происхождение Istio

Istio – это реализация сервисной сетки с открытым исходным кодом, созданная компаниями Google, IBM и Lyft. Начавшийся как совместный проект этих компаний, он быстро обогатился вкладом многих других организаций и отдельных лиц. Istio – обширный проект; в истинно облачной экосистеме по масштабам задач он занимает второе место после Kubernetes. Он включает в себя ряд проектов, разрабатываемых энтузиастами под эгидой Cloud Native Computing Foundation (CNCF), таких как Prometheus, OpenTelemetry, Fluentd, Ambassador, Jaeger, Kiali и многие другие.

Подобно другим сервисным сеткам, Istio помогает повысить отказоустойчивость и прозрачность архитектуры сервисов. Сервисные сетки не требуют, чтобы приложения знали о существовании сетки, и в этом отношении дизайн Istio ничем не отличается от других сервисных сеток. Работая с входящим, внутренним и исходящим трафиками, Istio прозрачно перехватывает и обрабатывает сетевой трафик от имени приложения.

Используя Envoy в качестве компонента уровня данных, Istio поможет настроить приложение так, чтобы рядом с ним разворачивался экземпляр прокси. Уровень управления Istio состоит из нескольких компонентов, управляющих конфигурацией прокси уровня данных, API для администраторов, настройками безопасности, проверкой политик и многим другим. Эти компоненты уровня управления мы рассмотрим в последующих главах данной книги.

Хотя изначально Istio создавалась для работы в Kubernetes, ее дизайн не зависит от платформы развертывания. То есть сервисную сетку на основе Istio можно развернуть и на платформах OpenShift, Mesos и Cloud Foundry, а также в таких традиционных окружениях, как виртуальные и физические машины. Взаимодействие Consul с Istio может помочь при развертывании на виртуальных и физических машинах. Istio принесет пользу в любом случае, используется ли она для поддержки монолита или группы микросервисов, и чем больше сервисов, тем больше пользы.

Текущее состояние Istio

Как развивающийся проект, Istio имеет разумную частоту выпуска новых версий, что является одним из показателей активности проекта с открытым исходным кодом. На рис. 1.6 представлена статистика сообщества за период с мая 2017 г., когда было публично объявлено об образовании проекта Istio, до февраля 2019 г. За этот период сделано порядка 2400 форков (копий проекта с целью участия в разработке проекта или для использования его кода в качестве основы для других проектов) и получено около 15 000 звезд (проект получает звезду, когда пользователь помещает его в свои закладки, чтобы получить информацию о его обновлениях в своих новостях).

Простое количество звезд, форков и коммитов может служить индикатором жизнеспособности проекта, отражая скорость его развития, уровень интереса и поддержки. Каждую из этих исходных метрик можно усовершенствовать. От-

ношение числа коммитов, обзоров и слияний ко времени отражает скорость развития проекта. При определении жизнеспособности проекта следует обращать внимание на то, как изменяется активность, насколько регулярно выходят новые версии и как часто появляются исправления ошибок и улучшения между выходами версий.

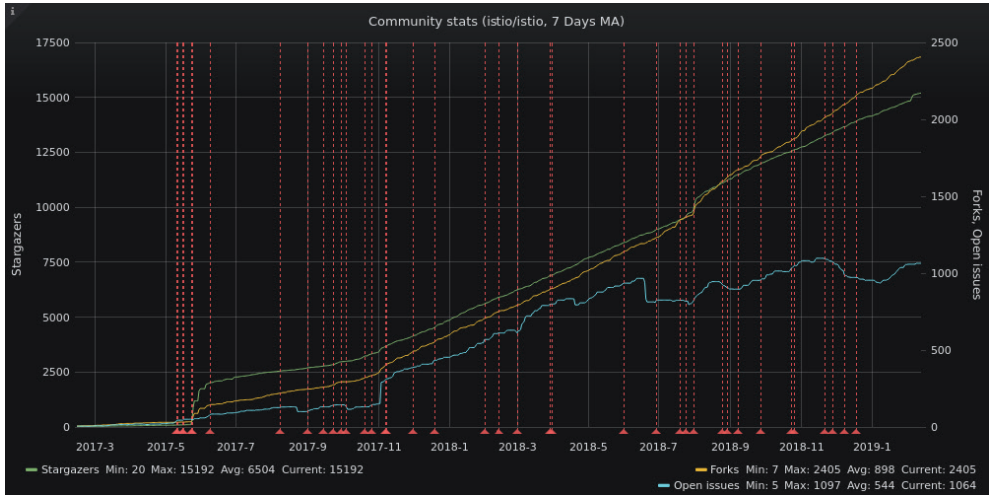


Рис. 1.6 ❖ Статистика проекта Istio

Активность развития

Нумерация версий Istio осуществляется с использованием привычной семантики (<https://semver.org/lang/ru/>): например, версия 1.1.1. Как и другие проекты, Istio учитывает множество нюансов, выбирая частоту выпуска новых версий и устанавливая срок поддержки (см. табл. 1.1). Несмотря на доступность ежедневных и еженедельных выпусков, они не поддерживаются и могут быть ненадежными. Как показывает табл. 1.1, ежемесячные выпуски относительно безопасны и, как правило, содержат новые функции. Если вы хотите использовать Istio в производстве, ищите выпуски, отмеченные как LTS (Long-Term Support – долгосрочная поддержка). На момент написания этой книги последней LTS-версией была версия 1.2.x.

Для справки: минорные версии Kubernetes выходят примерно каждые три месяца, поэтому каждая такая версия поддерживается в течение примерно девяти месяцев.

К примеру, разработчики дистрибутива Ubuntu во главу угла ставят стабильность, а не скорость добавления новых функций, поэтому выпускают LTS-версии раз в два года в апреле. Стоит отметить, что LTS-версии используются более интенсивно (около 95 % всех установок Ubuntu — LTS-версии).

Таблица 1.1

Тип	Уровень поддержки	Качество и рекомендованное использование
Ежедневные сборки	Не поддерживаются	
Промежуточные (snapshot) версии	Поддержка предоставляется только для последней snapshot-версии	Предположительно стабильные версии, но их использование в производстве должно быть ограничено. Обычно принимаются только начинающими или пользователями, требующими специфических возможностей
LTS-версии	Поддержка предоставляется в течение трех месяцев после выхода следующей LTS-версии	Безопасно использовать в производстве. Пользователям рекомендуется как можно скорее обновляться до этих версий
Исправления	Такой же, как LTS/Snapshot-версий	Пользователям рекомендуется применять исправления по мере их выхода для данной версии

В проекте Docker используется немного другой график выпуска версий:

- версии Docker CE Edge выходят ежемесячно;
- версии Docker CE Stable выходят ежеквартально, а исправления к ним выпускаются по мере необходимости;
- версии Docker EE выпускаются дважды в год, а исправления к ним выпускаются по мере необходимости.

Обновления и исправления выпускаются в следующем режиме:

- исправления и обновления вносятся в версию Docker EE в течение не менее одного года после выпуска;
- исправления и обновления вносятся в версию Docker CE Stable в течение одного месяца после выхода следующей версии Docker CE Stable;
- исправления и обновления вносятся в версию Docker CE Edge только до выхода следующей версии Docker CE Edge или Stable.

Выпуски

Первоначально планировалось, что ежеквартально будет выходить один точечный выпуск Istio, за которым последуют n выпусков с исправлениями. Snapshot-версии, качественно не уступающие точечным выпускам, предполагалось выпускать раз в месяц, но их не планировалось поддерживать, и они могли включать изменения, нарушающие обратную совместимость. История всех релизов доступна на странице истории выпусков проекта Istio в GitHub (<https://oreil.ly/2fZ4x>). В табл. 1.2 представлены данные о выходе новых версий Istio за 10-месячный период.

Таблица 1.2. Скорость выпуска новых версий Istio с августа 2018 г. по апрель 2019 г.

Дата выпуска	Номер выпуска	Дней с предыдущего выпуска
4/16/19	1.1.3	11
4/5/19	1.1.2	0
4/5/19	1.0.7	11
3/25/19	1.1.1	6
3/19/19	1.1.0	3
3/16/19	1.1.0-rc.6	2

Таблица 1.2 (окончание)

Дата выпуска	Номер выпуска	Дней с предыдущего выпуска
3/14/19	1.1.0-rc.5	2
3/12/19	1.1.0-rc.4	4
3/8/19	1.1.0-rc.3	4
3/4/19	1.1.0-rc.2	5
2/27/19	1.1.0-rc.1	6
2/21/19	1.1.0-rc.0	8
2/13/19	1.1.0-snapshot.6	0
2/13/19	1.0.6	19
1/25/19	1.1.0-snapshot.5	0
1/25/19	1.1.0-snapshot.4	48
12/8/18	1.0.5	11
11/27/18	1.1.0-snapshot.3	6
11/21/18	1.0.4	26
10/26/18	1.0.3	3
10/23/18	1.1.0-snapshot.2	26
9/27/18	1.1.0-snapshot.1	21
9/6/18	1.0.2	8
8/29/18	1.1.0-snapshot.0	5

Классификация версий

В полном соответствии с принципами гибкой разработки каждый выпуск Istio проходит свой жизненный цикл (разработка / альфа-версия / бета-версия / стабильная версия). Одни выпуски достигают стабильного состояния, а другие только появляются или совершенствуются, как показано в табл. 1.3.

Таблица 1.3. Характеристики разных версий Istio (см. <https://oreil.ly/qV8b0>)

	Альфа-версия	Бета-версия	Стабильная версия
Цель	Демонстрация возможностей; работает от начала до конца, но имеет ограничения	Готова к использованию в производстве, больше не игрушка	Надежная, проверенная в производстве
API	Без гарантий обратной совместимости	Поддерживается версионирование API	Надежный, пригодный к использованию в производстве. Поддерживается версионирование API с автоматическим преобразованием версий для обратной совместимости
Производительность	Не измеряется и не гарантируется	Не измеряется и не гарантируется	Производительность (задержка/масштаб) измерима, документирована, с гарантиями от регрессии
Политика устаревания	Нет	Слабая: 3 месяца	Надежная, стабильная. Уведомления рассылаются за год до изменений

Будущее

Рабочие группы разрабатывают проекты архитектуры v2 с учетом уроков, извлеченных из массового применения Istio и отзывов пользователей об удоб-

стве применения. В будущем все больше и больше людей узнают о сервисных сетках, поэтому простота внедрения будет ключевым фактором, помогающим массам успешно достичь третьей фазы их собственного облачного путешествия → контейнеры → оркестраторы → сетки.

То, чем Istio не является

Istio не учитывает специфические возможности, имеющиеся в других сервисных сетках, или предлагаемые программным обеспечением уровня управления. Это связано с изменчивостью и с использованием дополнительного стороннего программного обеспечения.

Istio лишь упрощает распределенную трассировку, но не является решением для мониторинга производительности *white box* (полностью известных) приложений (application performance monitoring, APM). Способы генерации дополнительной телеметрии, сопровождающей и анализирующей сетевой трафик и сервисные запросы, доступные в Istio, обеспечивают дополнительную видимость *black box*, объектов с неизвестными/необъявленными параметрами. Из всех метрик и журналов, доступных в Istio, эта телеметрия позволяет получить представление о потоках сетевого трафика, включая источник, приемник, задержки и ошибки; метрики сервисов высшего уровня, но нестандартные метрики приложений, которые генерируют рабочие нагрузки по отдельности, а также журналы уровня кластера остаются недоступными.

В Istio имеются плагины, интегрирующие журналы сервисов в систему мониторинга, которая используется для протоколирования на уровне кластера (например, Fluentd, Elasticsearch, Kibana). Кроме того, Istio может использовать метрики и оповещения, которые собираются уже применяемыми утилитами, такими как Prometheus.

Речь идет не только о микросервисах

Kubernetes этого всего не делает. Будет ли инфраструктура будущего полностью базироваться на использовании Kubernetes? Вряд ли. Не все приложения, особенно разработанные для работы вне контейнеров, хорошо подходят для Kubernetes (во всяком случае, пока). Хвост информационных технологий достаточно длинный – вспомним, что созданные десятилетия назад мэйнфреймы используются и по сей день.

Никакие технологии не являются панацеей. Монолиты легче понять, потому что большая часть приложения находится в одном месте. Можно проследить взаимодействие различных его частей в рамках одной системы (либо более или менее ограниченного набора систем). Однако монолиты не масштабируются с точки зрения команд разработчиков и строк кода.

Нераспределенные монолиты будут существовать еще долгое время. Сервисные сетки помогают их модернизировать, позволяя создавать фасады, упрощающие эволюционное развитие архитектур. Прием развертывания перед монолитом сервисной сетки, играющей роль интеллектуального фасада, используется многими для постепенного замещения монолита путем перехвата запросов по адресам назначения (или иным способом). Этот подход позволяет постепенно перенести функции монолита в современные микросервисы,

а также может использоваться в качестве временной меры в ожидании полной реорганизации структуры облачных вычислений.

ТЕРМИНОЛОГИЯ

Некоторые важные термины, связанные с Istio, следует знать и помнить:

Облако (cloud)

Специализированный провайдер облачных сервисов.

Кластер (cluster)

Набор узлов Kubernetes с общим API.

Хранилище конфигурации (config store)

Система, хранящая конфигурацию вне уровня управления, например etcd в сервисной сетке Istio, развернутой в Kubernetes, или даже в простой файловой системе.

Управление контейнерами (container management)

Программные стеки виртуализации операционных систем, такие как Kubernetes, OpenShift, Cloud Foundry, Apache Mesos и другие.

Окружение (environment)

Вычислительная среда от поставщиков инфраструктуры как услуги (IaaS), таких как Azure Cloud Services (<https://oreil.ly/mDkY5>), AWS (<https://aws.amazon.com/>), Google Cloud Platform (<https://oreil.ly/39fXT>), IBM Cloud (<https://oreil.ly/N-1xd>), Red Hat Cloud Computing (https://oreil.ly/ZCMj_), или группа виртуальных/физических машин, работающих в локальных/удаленных центрах обработки данных.

Сетка (mesh)

Ряд рабочих нагрузок с общим административным управлением в рамках одного и того же руководящего органа (например, уровня управления).

Мультисреда (гибрид) (Multienvironment, hybrid)

Неоднородный набор окружений, каждое из которых может отличаться от других реализацией и способом развертывания следующих инфраструктурных компонентов:

Границы сети (network boundaries)

Пример: один компонент доступен в местной сети, а другой – в облаке.

Системы идентификации (Identity systems)

Пример: один компонент использует LDAP, другой – учетные записи сервисов.

Системы разрешения имен, такие как DNS (Naming systems)

Пример: локальный DNS, DNS на базе Consul.

VM / контейнер / фреймворки управления процессами (VM / container / process orchestration frameworks)

Пример: один компонент имеет локально управляемые VM, а другой – контейнеры, управляемые Kubernetes.

Множественная аренда (Multitenancy)

Логически изолированные, физически интегрированные сервисы, работающие под одним уровнем управления сервисной сетки Istio.

Сеть (Network)

Набор непосредственно связанных между собой конечных точек (может включать виртуальную частную сеть [VPN]).

Безопасное разрешение имен (Secure naming)

Обеспечивает сопоставление между именем сервиса и субъектами рабочих нагрузок, уполномоченными на выполнение рабочих нагрузок, реализующих сервис.

Сервис (Service)

Определенная группа взаимосвязанных линий поведения в рамках сервисной сетки. Сервисы имеют имена, а политики Istio, такие как балансировка нагрузки и маршрутизация, применяются к именам сервисов. Обычно сервис имеет одну или несколько конечных точек, доступных извне.

Конечная точка сервиса (Service endpoint)

Достижимое по сети представление сервиса. Конечные точки экспортируются рабочими нагрузками. Не все сервисы имеют конечные точки, доступные извне.

Сервисная сетка (Service mesh)

Общий набор имен и идентификационных данных, позволяющий обеспечить применение общих политик и сбор телеметрии. Имена сервисов и субъект рабочей нагрузки уникальны в пределах одной сетки.

Имя сервиса (Service name)

Уникальное имя сервиса, идентифицирующее его в пределах сети сервисов. Сервис нельзя переименовать, он сохраняет свою идентичность: каждое имя сервиса уникально. Сервис может иметь несколько версий, но имя сервиса не зависит от версии. Имена сервисов доступны в конфигурации Istio в виде атрибутов `source.service` и `destination.service`.

Прокси сервиса (Service proxy)

Компонент уровня данных, управляющий трафиком от имени прикладных сервисов.

Шаблон «Прицеп» (Sidecar)

Методология совместного планирования служебных и прикладных контейнеров, сгруппированных в одну логическую единицу планирования. В случае с Kubernetes – *под* (pod).

Рабочая нагрузка (Workload)

Процесс / двоичный код, развернутый в Istio, обычно представленный такими объектами, как контейнеры, поды или виртуальные машины. Рабочая нагрузка может содержать ноль или более конечных точек; рабочая нагрузка может потреблять ноль или более сервисов. Каждая рабочая нагрузка имеет одно каноническое имя сервиса, связанное с ней, но может также иметь дополнительные имена сервисов.

Имя рабочей нагрузки (Workload name)

Уникальное имя для рабочей нагрузки, идентифицирующее ее в пределах сервисной сетки. В отличие от имени сервиса и субъекта рабочей нагрузки, имя рабочей нагрузки не является строго контролируемым свойством и не должно использоваться в определениях списков управления доступом

(ACL). Имена рабочих нагрузок доступны в конфигурации Istio в виде атрибутов `source.name` и `destination.name`.

Субъект (учетная запись) рабочей нагрузки (*Workload principal*)

Определяет контролируемые полномочия, под которыми выполняется *рабочая нагрузка*. Для проверки субъектов рабочих нагрузок в Istio используется аутентификация сервис–сервис. По умолчанию субъекты рабочих нагрузок соответствуют формату SPIFFE ID (<http://spiffe.io/>). Множественные рабочие нагрузки могут совместно использовать одного и того же субъекта, но каждая рабочая нагрузка имеет один субъект. Они доступны в конфигурации Istio в виде атрибутов `source.user` и `destination.user`.

Зона, уровень управления Istio (*Zone, Istio control plane*)

В набор компонентов, необходимых для работы сервисной сетки Istio, входят Galley, Mixer, Pilot и Citadel.

- Одна зона представлена одним логическим хранилищем Galley.
- Все компоненты Mixer и Pilot, подключенные к одному хранилищу Galley, считаются частью одной и той же зоны, независимо от того, где они работают.
- Одна зона может работать независимо, даже если все другие зоны находятся вне сети или недоступны.
- Одна зона может содержать только одно *окружение*.
- Зоны не используются для идентификации *сервисов* или *рабочих нагрузок*. Каждое *имя сервиса* и *каждый субъект рабочей нагрузки* принадлежит сервисной сетке в целом, а не отдельной зоне.
- Каждая зона относится к одной сервисной сетке. Сервисная сетка охватывает одну или несколько зон.
- В отношении кластеров (например, кластеров Kubernetes) и поддержки мультисред одна зона может иметь несколько экземпляров таких кластеров. Однако пользователям Istio лучше выбрать более простые конфигурации. Запуск компонентов уровня управления в каждом кластере или окружении и ограничение конфигурации зоны единственным кластером являются относительно простой задачей.

Администраторам необходимы независимый контроль и гибкий набор инструментов для обеспечения безопасности, совместимости, доступности и отказоустойчивости микросервисов. Разработчикам требуются свобода от проблем с инфраструктурой и возможность экспериментировать с различными производственными функциями, а также выполнять канареечное развертывание новых версий без ущерба для всей системы. Istio добавляет в микросервисы поддержку управления трафиком и создает основу для важнейших функций, таких как безопасность, мониторинг, маршрутизация, управление связью и управление политиками.