Гибкая архитектура интеграции

Упрощенные среды интеграции на основе контейнеров и микросервисов

Содержание

Гибкая интеграция, основанная на принципе адаптивности и динамичности, дает возможность быстро удовлетворять потребность в мультиоблаке, децентрализации и микросервисах, возникающую в ходе цифровой трансформации.

Краткая аннотация

Компаниям, вступившим на путь цифровой трансформации, приходится искать новые способы использования и развертывания технологий интеграции, которые давали бы им возможность быстро внедрить мультиоблако, децентрализацию и микросервисы. Уровень интеграции приложений следует изменить так, чтобы организации могли без проблем создавать новые схемы взаимодействия с клиентами, а не подгонять модели к архитектуре и разработке в ущерб эффективности бизнеса.

Многие организации, начавшие внедрять гибкие технологии для приложений (например, архитектуру микросервисов), уже видят первые положительные результаты. Такой подход дополняет и ускоряет корпоративную стратегию в отношении API. Предприятиям, модернизирующим свои инфраструктуры ESB, следует действовать по тому же принципу, так как он помогает эффективнее управлять службами интеграции в частном и гибридном облаках.

Эта брошюра составлена на основе книги, посвященной исследованию преимуществ того, что мы называем **гибкой архитектурой интеграции** – основанной на контейнерах, децентрализованной и состоящей из микросервисов архитектурой для решений интеграции, удовлетворяющих потребностям в адаптивности, масштабируемости и устойчивости, без которых цифровая трансформация попросту невозможна.

Интеграция изменилась

По оценкам IDC, в следующие пять лет расходы на проекты цифровой трансформации составят 20 трлн долларов[1]. Чем же обусловлен такой стремительный рост этих расходов? Неослабевающей и постоянно растущей потребностью в изобретении новых способов взаимодействия с клиентами с помощью связанных интерфейсов в сети приложений, использующих данные всех типов.

Это непростая задача. Свести вместе процессы и источники информации в нужное время и в нужном контексте очень сложно, особенно в случае активного внедрения бизнес-приложений SaaS. Для создания конкурентного преимущества нужно добавлять в бизнес-процессы новые источники данных.

“Для перехода к новым способам обслуживания клиентов организации должны исследовать постоянно растущее количество приложений, процессов и источников информации, а все это, в свою очередь, еще больше усиливает потребность в средствах интеграции и требует вложений”.

Значение интеграции приложений для цифровой трансформации

В процессе разработки новых схем взаимодействия с клиентами приходится продумывать способ доступа к данным и предоставления этих данных для служб и API, на основе которых будут строиться эти схемы. При этом стоит выделить ряд весомых преимуществ интеграции приложений:

- Эффективное решение проблемы разобщенности. Доступ к данным возможен из любой системы и в любом формате. Это позволяет сформировать некую однородность, независимо от того, насколько по-разному развивается ваша мультиоблачная среда.
- Информация из конечных систем. Современная интеграция предполагает не только знание сложных протоколов и форматов данных, но и сбор информации о реальных объектах, бизнесе и функциях в конечных системах.
- Инновации посредством данных. Современные инновационные приложения способны объединять внешние данные и извлекать из них ценную информацию – это качество особенно заметно в архитектуре микросервисов.
- Приложения корпоративного уровня. Процессы интеграции в значительной степени определяет среда выполнения: это и средства корпоративного уровня для восстановления после сбоев, и отказоустойчивость, ведение протоколов, анализ производительности и многое другое.

Ситуация с интеграцией постоянно меняется в соответствии с потребностями рынка и предприятий. Но как мы пришли от SOA и ESB к современной гибкой архитектуре интеграции, основанной на контейнерах?

К чему мы пришли: SOA, ESB и API

Прежде чем заглядывать в будущее гибкой интеграции, давайте разберемся, с чего же все началось. Шаблоны SOA (сервисно-ориентированной архитектуры) появились в начале тысячелетия, и сначала широкая популярность стандартов SOA объяснялась ее блестящими перспективами, обещающими, что каждая система сможет обнаруживать другие системы и общаться с ними посредством шаблонов синхронного взаимодействия.

Если забежать немного вперед, то мы попадем в период активного развития ESB (сервисная шина предприятия) – технологии обеспечения связи с серверными системами, пришедшей из прошлых топологий вида “звезда”. Несмотря на успешное внедрение шаблона ESB на многих предприятиях, этот термин не особо прижился в облачном мире. Эта концепция оказалась громоздкой и недостаточно адаптивной. Как же получилось, что мы перешли от одной крайности к другой?

Ответ можно свести к нескольким факторам (которые, как правило, взаимосвязаны):

- Внедрить SOA оказалось сложнее, чем ESB, особенно для тех, кто финансирует проект в масштабе всего предприятия.
- Шаблоны ESB формировали единую инфраструктуру для всего предприятия, в которую входили десятки или сотни интеграций в производственном серверном кластере. Несмотря на то, что для шаблонов ESB не требовалась жесткая централизация, полученные топологии почти всегда становились ее жертвами.

[1] Тест IDC MaturityScape Benchmark: цифровая трансформация в мире, 2017 г., Shawn Fitzgerald. Golluscio.

- Централизованные шаблоны ESB были не в состоянии обеспечить такую экономию средств, на которую надеялись компании, так как интерфейсы из одного проекта нельзя было повторно использовать в других.
- Межкорпоративные ESB-проекты не укладывались в бюджет, и, как правило, финансировались только те службы, которые можно было использовать столько раз, сколько окупало бы вложенные расходы.

Обеспечение постоянного финансирования межкорпоративных ESB-инициатив было сопряжено с трудностями, так как эти проекты не имели прямого отношения к бизнесу.

В результате создание сервисов этой группой специалистов SOA стало не стимулирующим фактором, как ожидалось, а, наоборот, узким местом. Из-за этого шаблон централизованной ESB получил дурную славу.

Сервисно-ориентированная архитектура на основе шаблона ESB – это проект для всего предприятия, целью которого является создание многократно используемых синхронно доступных сервисов и API с тем расчетом, чтобы можно было быстрее создавать новые приложения, принимающие данные из других систем.

С другой стороны, микросервисная архитектура – это вариант, позволяющий создать приложение таким образом, чтобы оно стало более гибким, масштабируемым и устойчивым.

Аргументы в пользу гибкой архитектуры интеграции

Почему микросервисы становятся столь популярными в сфере, связанной с приложениями? Потому что это совершенно иной подход к структурированию приложений. Приложение представляет собой не большой массив кода, работающий на одном сервере, а набор небольших полностью независимых компонентов.

Микросервисная архитектура имеет три весомых преимущества:

1. Колоссальная **гибкость** – микросервисы достаточно малы, чтобы их кодовую базу можно было полностью понимать и изменять независимо друг от друга.
2. Эластичная **масштабируемость** – расход ресурсов можно полностью привязать к конкретной бизнес-модели.
3. Дискретная **устойчивость** – достаточная обособленность способствует тому, что изменения в одном микросервисе никак не отражаются на других во время выполнения.

Давайте представим, как бы выглядел, с учетом этих преимуществ, совершенно иной подход к интеграции – не централизованный, а на базе архитектуры микросервисов? Вот именно это мы и называем **“гибкой архитектурой интеграции”.**

Определение гибкой архитектуры интеграции звучит так: это “децентрализованная, основанная на контейнерах и микросервисах архитектура, предназначенная для решений интеграции”.

Гибкая архитектура интеграции характеризуется тремя, хоть и перекликающимися, но все же отдельными аспектами:

Аспект 1. Мелкомодульное развертывание интеграции. Что нам дает разбиение интеграций в обособленной ESB на отдельные среды выполнения?

Аспект 2. Автономное владение интеграцией. Как отрегулировать структуру организации, чтобы эффективнее использовать такой мелкомодульный подход?

Аспект 3. Облачная инфраструктура интеграции. Какие еще преимущества можно получить от полностью облачного подхода к интеграции.

Аспект 1. Мелкомодульное развертывание интеграции

Централизованное развертывание узла интеграции или шаблонов ESB, в которых все интеграционные процессы развертываются на одной паре серверов с интенсивным обслуживанием (heavily nurtured, HA), как уже говорилось, оказывается узким местом для проектов. Любое развертывание на общих серверах вносит риск дестабилизации имеющихся ключевых интерфейсов. Ни один отдельный проект не способен обновить версию промежуточного ПО интеграции для получения доступа к новым функциям.

Мы могли бы разбить общекорпоративный компонент ESB на меньшие специализированные фрагменты, которыми легче управлять. Возможно, в некоторых случаях мы даже сможем прийти к единой среде выполнения для каждого экспортируемого интерфейса. Эти шаблоны “мелкомодульного развертывания интеграции” в корне отличаются от прошлых централизованных шаблонов ESB, так как предоставляют специализированные контейнеры нужного размера, что способствует повышению гибкости, масштабируемости и устойчивости. На рис. 1 простыми словами объясняется, в чем разница между централизованной ESB и мелкомодульным развертыванием интеграции.

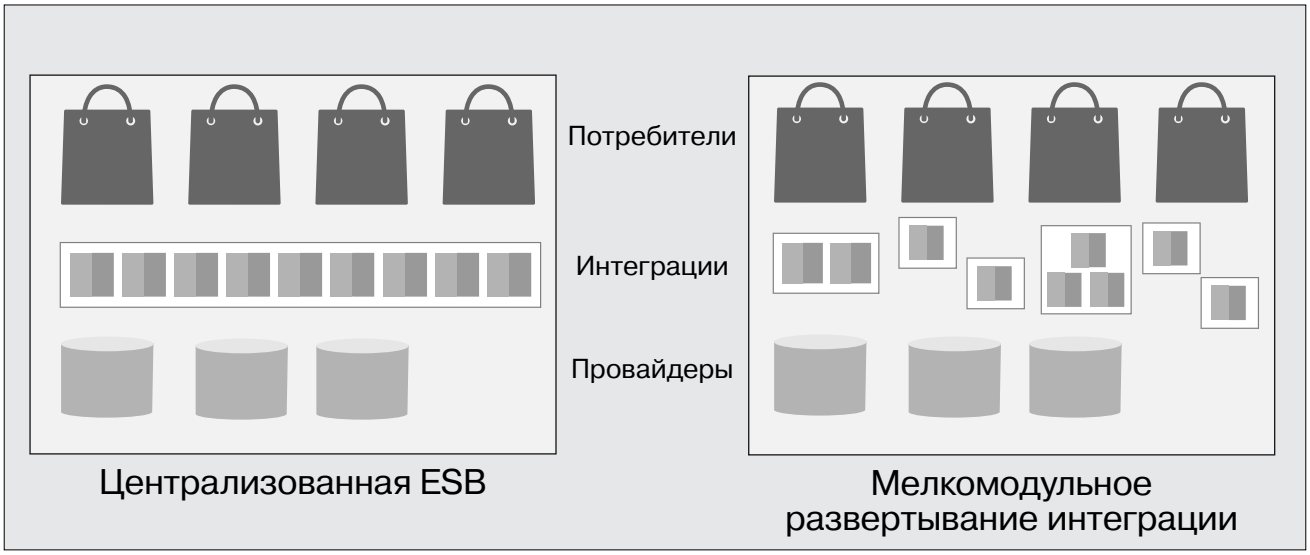

Рис. 1. Упрощенное сравнение централизованной ESB и мелкомодульного развертывания интеграции.

При мелкомодульном развертывании интеграции реализуются преимущества микросервисной архитектуры. Давайте еще раз вернемся к преимуществам микросервисов и рассмотрим их в свете мелкомодульного развертывания интеграции:

- **Гибкость** – разные коллективы могут осуществлять интеграцию независимо друг от друга, не полагаясь на централизованную группу или инфраструктуру, которая может быстро превратиться в слабое звено. Потоки интеграции можно по отдельности изменять, перестраивать и развертывать независимо от других потоков, что способствует более безопасному внедрению изменений и ускоряет ввод решения в эксплуатацию.
- **Масштабируемость** – потоки можно масштабировать по отдельности, что дает вам все преимущества эффективного эластичного масштабирования облачной инфраструктуры.
- **Устойчивость** – изолированные потоки интеграции, развернутые в отдельных контейнерах, не отнимают друг у друга общие ресурсы: память, соединения или процессоры.

Размышляя о гибкости, масштабируемости и устойчивости, важно помнить, что все эти качества мелкомодульной интеграции не проявляются в централизованной инфраструктуре.

Узнать больше о мелкомодульной интеграции можно в нашей книге “Гибкая инфраструктура интеграции”, которую теперь можно загрузить отсюда!

Аспект 2. Автономное владение интеграцией

Серьезной проблемой для сервисно-ориентированной архитектуры была необходимость создания централизованных групп интеграции и инфраструктуры для формирования уровня обслуживания.

В связи с этим темпы выполнения проектов были нестабильными, так как постоянно зависели от центральной группы интеграции. Центральная группа хорошо владела технологией интеграции, но часто не понимала самих интегрируемых приложений, что замедляло передачу требований и порождало множество ошибок.

Многие организации предпочли бы, чтобы коллективы разработки приложений сами отвечали за создание своих сервисов, но в то время технологии и инфраструктуры не имели такой возможности.

Переход к мелкомодульному развертыванию интеграции позволяет распределять владение процессами интеграции и их обслуживания. Так, взяться за работу по интеграции разумнее было бы командам бизнес-приложений: это упростит внедрение новых функций.

Заинтересовались мелкомодульным развертыванием интеграции? Ответы на свои вопросы вы можете получить в нашей книге “Гибкая архитектура интеграции”, которая уже доступна для загрузки!

Аспект 3. Облачная инфраструктура интеграции.

В последние годы среды выполнения интеграции претерпели серьезные изменения. Настолько серьезные, что теперь эти упрощенные среды выполнения можно использовать как чисто облачные. Другими словами, теперь многие запатентованные механизмы управления кластерами, масштабирования, обеспечения готовности и т. д. можно перенести в облачную платформу, в которой они выполняются.

Это дает намного больше, чем просто их выполнение в контейнеризованной среде. Это значит, что они должны иметь возможность выступать как единое “стадо”, а не как обособленные объекты, что способствует максимально эффективному использованию средств координации, таких как Kubernetes и многих других популярных облачных сервисов.

“Стадный” принцип изменит способы взаимодействия групп DevOps со средой и решением в целом: по мере переноса решений в упрощенные архитектуры эффективность их использования будет расти.

Как изменилась современная среда выполнения интеграции в ответ на спрос на гибкую архитектуру интеграции?

Очевидно, что гибкая архитектура интеграции требует совсем другой топологии интеграции. Ключевым фактором здесь выступает современная среда интеграции, которую можно запускать на базе контейнеров и которая отлично подходит для технологий облачного развертывания. Современные среды выполнения интеграции не имеют почти ничего общего со своими прошлыми аналогами. Вот некоторые из основных отличий:

- **Быстрая упрощенная среда выполнения.** Работают в контейнерах типа Docker и достаточно легки, чтобы запускаться и останавливаться за считанные секунды. Помимо этого, эти среды выполнения поддерживают удобное администрирование с помощью таких фреймворков координации, как Kubernetes.
- **Отсутствие зависимостей.** Больше нет необходимости в очередях обращения к базам данных и очередях сообщений, хотя, разумеется, при необходимости их можно подключить без малейших проблем.
- **Инсталляция на основе файловой системы.** Для установки достаточно просто поместить двоичные файлы в файловую систему и запустить их. Такой способ идеально подходит для многоуровневых файловых систем образов Docker.
- **Поддержка инструментария DevOps.** Среда выполнения должна быть готова к непрерывной интеграции и развертыванию. Установка, компоновка, развертывание и настройка посредством сценариев и файлов свойств способствует реализации принципов “инфраструктура как код”. Для ускорения встраивания в конвейеры DevOps должны быть созданы шаблонные сценарии для стандартных инструментов компоновки и развертывания.
- **Сначала API.** Главным протоколом связи должен быть API RESTful. Экспорт интеграций в виде API RESTful должен быть простым и основанным на стандартных соглашениях, например на спецификации Open API. Вызов внутренних APi RESTful тоже должен быть простым, включая обнаружение с помощью файлов определений.
- **Цифровая связь.** Помимо расширенной связи в масштабах предприятия, которую и так предоставляют среды выполнения интеграции, необходимо также подключение к современным ресурсам. К примеру, к базам данных NoSQL (MongoDb и Cloudant и т. п.) и службам обмена сообщениями типа Kafka. Более того, необходим также доступ к обширному каталогу интеллектуальных коннекторов для приложений SaaS (ПО как услуга), например Salesforce.

- **Непрерывная доставка.** Непрерывная доставка осуществляется посредством интерфейсов командной строки и шаблонных сценариев, встроенных в стандартные инструменты конвейера DevOps. Теперь для реализации интерфейсов требуется еще меньше специальных знаний, что заметно ускоряет доставку.
- **Расширенный инструментарий.** Расширенный инструментарий для интеграции предполагает создание большинства интерфейсов только посредством конфигурации, и зачастую это могут делать лица, не имеющие опыта интеграции. Добавление шаблонов для распространенных схем интеграции позволило встроить в инструментарий лучшие методики интеграции и еще больше упростить работу. Высококвалифицированные специалисты по интеграции требуются уже намного реже, а некоторые интеграции могут осуществляться даже группами разработчиков приложений – мы увидим это в следующем разделе о децентрализованной интеграции.

Современные среды выполнения интеграции идеально вписываются во все три качества гибкой архитектуры интеграции: мелкомодульное развертывание, автономное владение и облачную инфраструктуру.

Хотите еще глубже изучить облачную инфраструктуру? Загрузите нашу книгу “Гибкая архитектура интеграции”!

Гибкая архитектура интеграции для платформы интеграции

В этой статье мы главным образом описываем особенности интеграции приложений при развертывании в гибкой архитектуре интеграции. Но некоторые задачи предприятий возможно решить только с применением нескольких важнейших функций интеграции. Эти функции собраны в платформе интеграции (или, как говорят некоторые аналитики, платформе гибридной интеграции), позволяющей организациям эффективно и согласованно создавать бизнес-решения.

Ценность такой платформы признают многие отраслевые специалисты. Так, Gartner отмечает:

Платформа гибридной интеграции (HIP) – это среда локальной и облачной интеграции с функциями управления, позволяющая пользователям с совершенно разным уровнем знаний (как специалистам по интеграции, так и далеким от нее лицам) выполнять самые разные задачи, связанные с интеграцией. …Чтобы иметь возможность решать срочные задачи, встающие перед цифровым предприятием, руководители групп разработки приложений, отвечающие за интеграцию, должны использовать функции среды HIP для модернизации своих стратегий и инфраструктуры интеграции[2].

Один из ключевых моментов, на которых заостряет внимание Gartner, – это то, что платформа интеграции позволяет каждому пользователю организации работать в своем индивидуальном стиле. Это значит, что обычные пользователи могут эффективно работать в простом интерфейсе, в который уже встроены алгоритмы решения несложных задач, тогда как ИТ-специалисты получают высокий уровень контроля и возможность работы с более сложными сценариями корпоративного уровня. Эти пользователи могут работать вместе, многократно используя общие ресурсы, без ущерба для управляемости среды в целом.

Решение задач, возникающих в ходе цифровой трансформации, столь же важно, как и поддержка различных пользовательских сообществ. Мы посвятим этим задачам большую часть статьи, но сначала подробно расскажем о ключевых функциях, которые должны присутствовать в платформе интеграции.

Платформа IBM Cloud Integration

IBM Cloud Integration объединяет основной набор функций интеграции в целостную, простую, надежную и быстродействующую платформу. Она позволяет быстро (за считаные минуты) создавать мощные интеграции и API, демонстрирует высокую производительность и масштабируемость, а также предлагает непревзойденные средства обеспечения сквозной безопасности в масштабах всей организации.

В рамках платформы IBM Cloud Integration мы объединили шесть ключевых компонентов интеграции, реализация каждого из которых является лучшей в своем классе. Компоненты:

Управление API.
Экспорт и администрирование бизнес-сервисов как многократно используемых API для избранных сообществ разработчиков в вашей организации и за ее пределами. Организации реализуют стратегию API для ускорения и повышения эффективности обмена уникальными данными и сервисами с целью содействия разработке новых приложений и реализации новых бизнес-возможностей.

Шлюз безопасности.
Расширение связи и интеграции за пределы предприятия с помощью передовых функций с поддержкой DMZ для защиты API, передаваемых данных и базовых систем

Интеграция приложений.
Подключение приложений и источников данных локально и в облаке с целью координации обмена бизнес-информацией и обеспечения доступа к данным в любое время и из любого места.

Обмен сообщениями.
Обеспечивает доступ к информации в реальном времени когда угодно и из любого места за счет надежной доставки сообщений без потерь и дублирования. Не требует сложных процессов восстановления в случае сетевого или системного сбоя.

Интеграция данных.
Доступ, очистка и подготовка данных в хранилище или озере данных для создания целостного представления о вашем бизнесе для аналитических целей.

Высокоскоростная передача данных.
Перемещение огромных объемов данных между локальными и облачными или только между облачными ресурсами – быстро, предсказуемо и с повышенной степенью защиты. Содействует ускорению внедрения облачных платформ в организациях при очень больших объемах данных.

Хотя это рекламная статья, надеемся, что вы получили представление о различных важнейших функциях, которые должны входить в состав платформы интеграции, разобрались в условиях, которые необходимо соблюдать для успешной совместной работы этих функций, и поняли, как должна внедряться гибкая архитектура интеграции для того, чтобы ваша платформа стала гибкой, масштабируемой и отказоустойчивой.

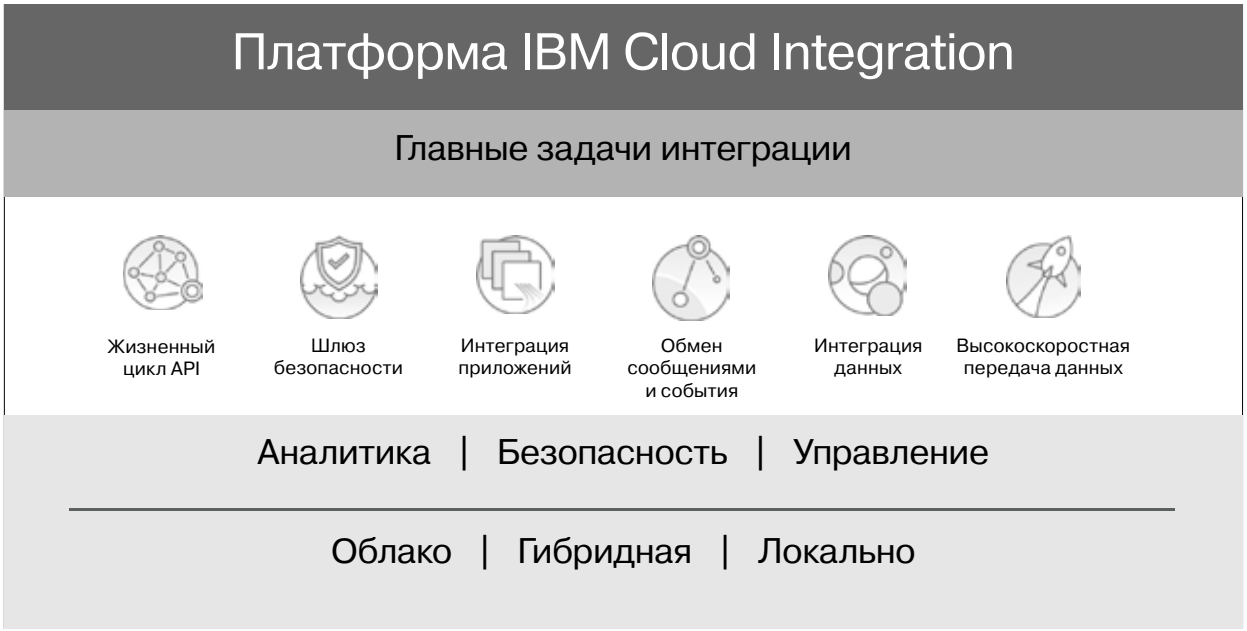

Рис. 2. Платформа IBM Cloud Integration.

Не забудьте загрузить полную версию электронной книги, чтобы детально изучить гибкую архитектуру интеграции.

IBM Восточная Европа/Азия
123317 Москва
Пресненская наб., 10

Веб-сайт IBM:
ibm.com

Подлежит утилизации

82018782-RURU-00