Бессерверные многоуровневые архитектуры AWS

Использование Amazon API Gateway и AWS Lambda

Ноябрь 2015 г.

© Amazon Web Services, Inc. или ее дочерние организации, 2015. Все права защищены.

Уведомления

Этот документ предоставляется исключительно в информационных целях. В документе представлены текущие предложения продуктов и практики AWS, актуальные на дату публикации, которые могут меняться без предварительного уведомления. Клиентам необходимо провести независимую оценку представленной в документе информации и возможности использования продуктов и услуг AWS любым способом. Указанные информация, продукты и услуги предоставляются «как есть», без какой-либо явной или подразумеваемой гарантии. Данный документ не создает никаких гарантий, контрактных обязательств и иных обязательств, условий или заверений от AWS, ее дочерних организаций, поставщиков или лицензиатов. Обязанности и финансовые обязательства AWS в отношении клиентов компании регулируются соглашениями AWS. Данный документ не является таким соглашением, а также не вносит изменения в какие-либо соглашения, заключенные между компанией AWS и ее клиентами.

Содержание

Резюме

В этом техническом описании показано, как инновационные возможности Amazon Web Services (AWS) могут изменить подход к проектированию многоуровневых архитектур популярных решений, таких как микросервисы, серверные системы для мобильных приложений и публичные веб-сайты. Теперь специалисты по архитектуре и разработчики могут использовать шаблон реализации, включающий в себя Amazon API Gateway и AWS Lambda, чтобы сократить цикл разработки и операционный цикл, необходимые для создания многоуровневых приложений и управления ими.

Введение

Многоуровневые приложения (трехуровневые, n-уровневые и т. д.) уже многие годы используются в качестве фундаментального шаблона проектирования. Многоуровневый шаблон служит удобным ориентиром для контроля и обслуживания разнородных и масштабируемых компонентов приложения по отдельности (зачастую даже разными группами). Нередко многоуровневые приложения создаются с использованием сервис-ориентированного подхода для применения интернет-сервисов. При этом сеть служит границей между уровнями. Однако существует множество типичных аспектов при создании уровня интернет-сервиса в рамках приложения. Большая часть кода, написанная для многоуровневого интернет-приложения – это прямой результат использования шаблона. В качестве примера можно привести код для интеграции разных уровней, код с определением API и модели данных, используемых уровнями для взаимодействия, а также код, защищающий точки интеграции уровней от несанкционированного доступа.

Amazon API Gateway[1], сервис для создания API и управления ими, и AWS Lambda[2], сервис для выполнения произвольных функций в коде, позволяют при совместном использовании упростить создание надежных многоуровневых приложений.

Благодаря интеграции Amazon API Gateway с AWS Lambda определенные пользователем функции можно вызывать непосредственно с помощью пользовательского HTTPS-запроса. И API Gateway, и Lambda автоматически масштабируются в соответствии с требованиями вашего приложения, независимо от объема запроса. При их совместном использовании можно создать уровень для приложения, предназначенный для написания основного кода приложения, и *не* тратить время на другие типовые аспекты реализации многоуровневой архитектуры, такие как оптимизация с целью повышения доступности, разработка клиентских SDK, управление сервером и операционной системой, масштабирование и реализация механизма авторизации клиентов.

Недавно AWS объявила о возможности создавать функции Lambda, которые выполняются в вашем облаке Amazon Virtual Private Cloud (Amazon VPC)[3]. Это дает еще больше преимуществ при совместном использовании API Gateway и Lambda и позволяет реализовать сценарии, в которых требуются частные сети, например интегрировать интернет-сервис с реляционной базой данных, которая содержит конфиденциальную информацию. Интеграция Lambda и Amazon VPC косвенно расширила возможности Amazon API Gateway, так как теперь разработчики могут определить собственный набор HTTPS API, доступных из Интернета, вне серверной части, которая остается закрытой и защищенной в Amazon VPC. Преимущества этого мощного подхода очевидны на каждом уровне многоуровневой архитектуры. В этом техническом описании основное внимание уделено самому популярному примеру многоуровневой архитектуры – **трехуровневому** интернет-приложению. Однако этот шаблон можно применять и в других, более сложных сценариях.

Обзор трехуровневой архитектуры

Трехуровневая архитектура – это популярный шаблон для приложений, взаимодействующих с пользователями. Она состоит из **уровня представления**, **логического уровня** и **уровня данных**. Уровень представления – это компонент, с которым непосредственно взаимодействуют пользователи (например, веб-страница, интерфейс мобильного приложения и т. д.). Логический уровень содержит код для преобразования действий пользователя на уровне представления в функции, на которых основано поведение приложения. Уровень данных состоит из хранилища (базы данных, хранилища объектов, кэши, файловые системы и т. д.), где размещаются данные приложения. На рис. 1 показан пример простого трехуровневого приложения.

Рис. 1. Шаблон архитектуры простого трехуровневого приложения

В Интернете есть множество прекрасных ресурсов, где можно подробнее ознакомиться с *общим* шаблоном трехуровневой архитектуры. В этом техническом описании рассматривается специальный шаблон реализации данной архитектуры с использованием Amazon API Gateway и AWS Lambda.

Бессерверный логический уровень

Логический уровень трехуровневой архитектуры представляет собой «мозг» приложения. Поэтому интеграция Amazon API Gateway и AWS Lambda для создания логического уровня может дать очень много преимуществ. Используя эти два сервиса, вы сможете создать бессерверное производственное приложение с высоким уровнем доступности, масштабирования и безопасности. Приложение может использовать тысячи серверов, но если выбрать этот шаблон, вам не придется управлять ни одним сервером. Кроме того, применяя эти управляемые сервисы вместе, вы получите следующие преимущества:

- нет необходимости выбирать, защищать, обновлять операционные системы или управлять ими;
- нет необходимости выбирать размер серверов, отслеживать или масштабировать их;
- нет риска переплаты за лишние ресурсы;
- нет риска спада производительности из-за недостаточного объема ресурсов.

Кроме того, в каждом сервисе доступны специальные возможности, которые дополняют трехуровневую архитектуру.

Amazon API Gateway

Amazon API Gateway – это полностью управляемый сервис для определения, развертывания и обслуживания API. Клиенты могут интегрировать API, используя стандартные HTTPS-запросы. Очевидно, что Amazon API Gateway легко может применяться в сервис-ориентированной многоуровневой архитектуре. Однако возможности и качества сервиса делают его серьезным подспорьем для логического уровня.

Интеграция с AWS Lambda

Amazon API Gateway позволяет приложениям легко использовать инновационные возможности AWS Lambda напрямую с помощью HTTPS-запросов. API Gateway служит мостом, который соединяет уровень представления и функции в AWS Lambda. После определения клиент-серверного отношения с помощью API содержимое HTTPS-запросов клиента передается функции Lambda для выполнения. Это могут быть метаданные, заголовки и текст запроса.

Стабильная производительность API во всем мире

С каждым развертыванием Amazon API Gateway связана база раздачи Amazon CloudFront[4]. Amazon CloudFront – это интернет-сервис доставки контента, который использует глобальную сеть периферийных местоположений Amazon в качестве точек подключения для клиентов, интегрирующих ваши API. Это позволяет уменьшить время ответа API. Благодаря использованию множества периферийных местоположений по всему миру с помощью Amazon CloudFront также можно бороться с DDoS-атаками. Дополнительные сведения см. в техническом описании AWS Best Practices for Combatting DDoS Attacks[5].

Вы можете повысить производительность определенных запросов API, используя Amazon API Gateway для хранения ответов в дополнительном кэше в памяти. Это не только увеличит производительность повторяющихся запросов API, но и сократит использование серверной части – тем самым вы сможете снизить расходы в целом.

Стимулирование развития

Ресурсы для разработки любого нового приложения – это инвестиции. Вам необходимо обосновать их перед началом проекта. Снижая объем финансовых и временных затрат, требуемых для разработки, вы получите больше пространства для экспериментов и инноваций.

Для множества многоуровневых приложений на основе интернет-сервисов уровень представления легко фрагментируется среди пользователей (отдельные мобильные устройства, веб-браузеры и т. д.). Зачастую эти пользователи также находятся в разных географических регионах. Однако отдельный логический уровень не фрагментируется физически пользователями. Все они зависят от одной инфраструктуры, поддерживающей логический уровень, что только подчеркивает ее значимость. Нередко при начальной реализации логического уровня для ускорения выпуска приложения разработчики экономят на различных аспектах («нам не нужны показатели эффективности для первого запуска», «в начале решение не будет использоваться интенсивно, позаботимся о масштабировании потом» и т. д.). Это может привести к технической ограниченности решения и создать операционные риски, ведь в этом случае придется развертывать изменения в уже работающем производственном приложении. Amazon API Gateway позволяет *вам* воспользоваться этими преимуществами и быстро выпустить приложение, так как они уже реализованы в сервисе. При этом вы не будете подвержены рискам, описанным выше.

Общий жизненный цикл приложения может быть неизвестен или может быть коротким. Поэтому сформировать экономическое обоснование для нового многоуровневого приложения довольно сложно. Можно упростить задачу, если уже на первом этапе у вас под рукой есть управляемые возможности Amazon API Gateway, при этом вы начнете нести расходы на инфраструктуру только после того, как API начнут получать запросы. Дополнительные сведения см. в разделе Цены Amazon API Gateway.[6]

Быстрые итерации и маневренность

Для новых приложений пользовательская база может быть определена недостаточно четко (размер, шаблоны использования и т. д.). Логический уровень должен сохранять маневренность по мере ее формирования. Вашему приложению и компании необходима возможность адаптироваться к меняющимся ожиданиям первых пользователей. Amazon API Gateway снижает число циклов разработки, требуемых для полного развертывания API. Amazon API Gateway позволяет создавать эмулируемые интеграции[7] для формирования ответов API, которые могут создавать клиентские приложения, напрямую от API Gateway и параллельно дорабатывать полноценную логику серверной части. Эту возможность можно использовать не только при первом развертывании API, но и после того как компания решит изменить приложение (и существующий API) в ответ на реакцию пользователей. API Gateway и AWS Lambda поддерживают управление версиями, поэтому существующие функциональные возможности и клиентские зависимости не будут затронуты при выпуске новых возможностей в виде отдельной версии API или функции.

Безопасность

При реализации логического уровня публичного трехуровневого интернет-приложения всегда возникает вопрос о безопасности. Приложение должно гарантировать, что только авторизованные клиенты могут получить доступ к логическому уровню (предоставляемому по сети). Сервис Amazon API Gateway реализует надежные механизмы безопасности для полной защиты серверной части приложения. Не следует полагаться на статические строки ключей API для управления доступом клиентских приложений, так как их можно извлечь из клиентов и использовать в других компонентах. Вы можете воспользоваться несколькими способами защиты логического уровня Amazon API Gateway:

- все запросы к вашим API могут выполняться по протоколу HTTPS для шифрования во время передачи;
- функции AWS Lambda могут ограничивать доступ так, чтобы отношение доверия существовало только между определенным API в Amazon API Gateway и определенной функцией в AWS Lambda. Вызвать эту функцию Lambda можно будет только с помощью выбранного API;

- Amazon API Gateway позволяет создавать клиентские SDK для интеграции с вашими API. Пакет SDK также будет контролировать подпись запросов, если для API требуется аутентификация. Данные для доступа API, используемые на стороне клиента для аутентификации, передаются непосредственно функции AWS Lambda, а дальнейшая аутентификация может происходить в вашем коде, если это необходимо;
- каждому сочетанию ресурса и метода, которое вы создаете в рамках API, назначается собственное имя ресурса Amazon (ARN), на которое можно ссылаться в политиках AWS Identity and Access Management (IAM)[8].
  - Это значит, что ваши API получают приоритет при обработке вместе с другими API AWS. Политики IAM можно настраивать в соответствии с вашими потребностями. Кроме того, они могут ссылаться на определенные ресурсы и методы API, созданного с помощью Amazon API Gateway.
  - Доступ к API регулируется политиками IAM, которые вы создаете вне контекста кода приложения. Это значит, что вам не нужно писать код для реализации этих уровней доступа. Если код не существует, он не будет содержать ошибок и не сможет быть взломан.
  - Авторизация клиентов с помощью подписи AWS версии 4 (SigV4)[9] и политик IAM для API позволяет использовать одни и те же данные для доступа, чтобы ограничить или разрешить доступ к другим сервисам и ресурсам AWS (например, к корзинам Amazon S3 или таблицам Amazon DynamoDB).

AWS Lambda

Сервис AWS Lambda позволяет выполнять произвольный код на любом поддерживаемом языке (Node, JVM и Python с ноября 2015 г.) в ответ на событие. Таким событием может быть один из нескольких программных триггеров, предоставляемых AWS. Их называют **источниками событий** (поддерживаемые сейчас источники см. здесь[10]). Многие популярные примеры использования AWS Lambda связаны с рабочими процессами обработки данных на основе событий, такими как обработка файлов, хранимых в Amazon Simple Storage Service (Amazon S3)[11], или потоковая передача записей данных из Amazon Kinesis[12].

При совместном использовании с Amazon API Gateway функция AWS Lambda может существовать в контексте типичного интернет-сервиса и может быть вызвана непосредственно HTTPS-запросом. Amazon API Gateway служит входной точкой для логического уровня, но теперь вам необходимо применить логику, лежащую в основе этих API. Здесь в дело вступает AWS Lambda.

Здесь могла бы быть ваша бизнес-логика

AWS Lambda позволяет писать функции, которые называют **обработчиками**, выполняющиеся в ответ на событие. Например, можно создать обработчик, который срабатывает при HTTPS-запросе вашего API. С помощью Lambda можно создавать модульные обработчики с выбранным уровнем детализации (один для отдельного API или один для отдельного метода API), которые можно обновлять, вызывать и изменять по отдельности. Обработчик может связываться с любыми другими зависимыми компонентами (например, другими функциями в коде, библиотеками, собственными двоичными файлами и даже внешними интернет-сервисами). Lambda позволяет упаковать все необходимые зависимые компоненты в определении функции во время ее создания. При этом вы указываете, какой метод в пакете развертывания будет служить обработчиком запроса. Вы можете повторно использовать пакет развертывания для определения множества функций Lambda, и у каждой функции может быть уникальный обработчик в одном и том же пакете развертывания. В бессерверной многоуровневой архитектуре каждый из API, создаваемый в Amazon API Gateway, интегрируется с функцией Lambda (и обработчиком в ней), которая выполняет необходимую бизнес-логику.

Интеграция Amazon VPC

AWS Lambda, фундамент логического уровня, напрямую интегрируется с уровнем данных. Так как на этом уровне часто располагаются конфиденциальные бизнес-данные или сведения пользователя, для него требуется надежная защита. Вы можете управлять доступом к сервисам AWS, которые можно интегрировать с помощью функции Lambda, используя политики IAM. К этим сервисам относятся Amazon S3, Amazon DynamoDB, Amazon Kinesis, Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), функции AWS Lambda и многие другие. Однако у вас может быть компонент, который самостоятельно управляет доступом, например реляционная база данных. Для улучшения защиты таких компонентов их можно развернуть в частной сетевой среде – Amazon Virtual Private Cloud (Amazon VPC)[13].

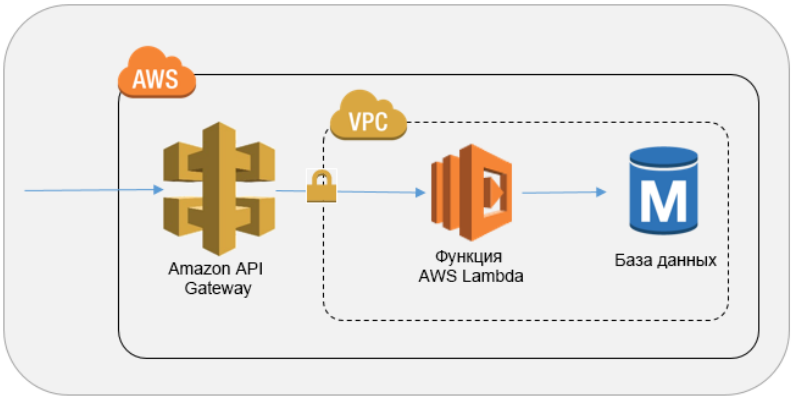

Рис. 2. Шаблон архитектуры с использованием VPC

При использовании VPC можно отключить доступ к базам данных и другим хранилищам, от которых зависит бизнес-логика, через Интернет. VPC также гарантирует, что взаимодействовать с данными из Интернета можно *только* через заданные вами API и написанные вами функции Lambda.

Безопасность

Для выполнения функции Lambda она должна быть вызвана событием или сервисом, которым это разрешает политика IAM. Можно создать функцию Lambda, которую *совсем* нельзя выполнить, если она не вызывается запросом API Gateway, который вы определили. Ваш код обрабатывает только часть допустимого примера использования, заданного созданным API.

Каждая функция Lambda предполагает роль IAM, которая назначается в соответствии с отношением доверия IAM. Роль IAM определяет другие сервисы и ресурсы AWS, с которыми функция Lambda сможет взаимодействовать (например, таблицу Amazon DynamoDB или корзину Amazon S3). Эти сервисы определяются и контролируются за рамками функции. Это звучит кратко, на за этими словами лежит море возможностей. Вы можете писать код, не сохраняя и не извлекая данные для доступа AWS, т. е. вам не нужно жестко кодировать ключи API и писать код для их извлечения и сохранения в памяти. Контроль вызова функцией Lambda сервисов, разрешенных ролью IAM, осуществляется самим сервисом автоматически.

Уровень данных

Используя AWS Lambda в качестве логического уровня, вы получаете более широкий выбор носителей данных для уровня данных. Все они разделены на две категории: хранилища данных, размещенные в Amazon VPC, и хранилища данных с поддержкой IAM. AWS Lambda поддерживает безопасную интеграцию с обоими типами хранилища.

Хранилища данных, размещенные в Amazon VPC

Интеграция AWS Lambda с Amazon VPC позволяет функциям безопасно использовать различные технологии хранения данных.

- Amazon RDS[14]

  Используйте любые ядра базы данных, предоставляемые Amazon Relational Database Service (Amazon RDS). Подключайтесь к Amazon RDS непосредственно из кода, написанного в Lambda, так же, как за пределами Lambda, но используя все преимущества простой интеграции AWS Key Management Service (AWS KMS) для шифрования данных доступа к базе данных.

- Amazon ElastiCache[15]

  Интегрируйте функции Lambda с управляемым кэшем в памяти, чтобы повысить производительность приложения.

- Amazon RedShift[16]

  Вы можете создавать функции для безопасного выполнения запросов к корпоративному хранилищу данных и формирования отчетов, панелей управления или получения специализированных результатов.

- Частный интернет-сервис, размещенный в Amazon Elastic Compute Cloud (Amazon EC2)[17]

  Существующие приложения могут работать как частный интернет-сервис в VPC. Выполняйте HTTP-запросы по логической частной сети VPC из функции Lambda.

Хранилища данных с поддержкой IAM

Так как сервис AWS Lambda интегрирован с IAM, он может использовать IAM для безопасной интеграции с любым сервисом AWS, который можно применять с помощью API AWS.

- Amazon DynamoDB[18]

  Amazon DynamoDB – это база данных AWS NoSQL с бесконечным масштабированием. Если вам необходимо извлекать записи данных (не больше 400 КБ на момент написания данной статьи) с задержкой несколько десятков миллисекунд независимо от масштаба, используйте Amazon DynamoDB. Благодаря детализированной системе управления доступом Amazon DynamoDB функции Lambda могут следовать принципу минимума доступных полномочий при запросе определенных данных в DynamoDB.

- Amazon S3[19]

  Amazon Simple Storage Service (Amazon S3) предоставляет масштабируемое хранилище объектов. Сервис Amazon S3 обеспечивает надежность хранения объектов на уровне 99,999999999 %, поэтому используйте его, если вам необходимо недорогое хранилище с высокой степенью надежности. Кроме того, сервис Amazon S3 предусматривает доступность объектов на уровне 99,99 %, поэтому воспользуйтесь им, если вам требуется хранилище с высокой степенью доступности. Доступ к объектам, хранимым в Amazon S3 (файлам, образам, журналам и двоичным данным), можно получать по протоколу HTTP. Функции Lambda могут безопасно взаимодействовать с Amazon S3 через частные конечные точки, а доступ к данным в S3 можно ограничить политикой IAM, связанной с данной функцией Lambda.

- Amazon Elasticsearch Service[20]

  Amazon Elasticsearch Service (Amazon ES) – это управляемая версия популярной поисковой и аналитической системы Elasticsearch. Amazon ES предоставляет управляемые возможности подготовки кластеров, обнаружения сбоев и замены узлов. Кроме того, вы можете ограничить доступ к Amazon ES API с помощью политик IAM.

Уровень представления

Amazon API Gateway предоставляет множество возможностей для уровня представления. HTTPS-интерфейсы API могут использоваться любыми клиентами, поддерживающими протокол HTTPS. В следующем списке приводятся распространенные примеры использования на уровне представления приложения.

- Мобильное приложение: помимо интеграции с пользовательской бизнес-логикой с помощью Amazon API Gateway и AWS Lambda, вы можете использовать Amazon Cognito[21] в качестве механизма для создания удостоверений пользователей и управления ими.
- Контент статических веб-сайтов (например, файлы, размещенные в Amazon S3): интерфейсы API, созданные в Amazon API Gateway, могут быть совместимыми с CORS. При этом веб-браузеры смогут напрямую вызывать API со статических веб-страниц.
- Любое другое клиентское устройство с поддержкой HTTPS: многие устройства, подключенные к сети, могут использовать HTTPS для связи с сервисами. В механизмах взаимодействия клиентов с API, созданными с помощью Amazon API Gateway, нет ничего уникального или особенного – это просто HTTPS. Специальное клиентское программное обеспечение и лицензии не требуются.

Примеры шаблона архитектуры

Вы можете реализовать следующие популярные шаблоны архитектуры, используя Amazon API Gateway и AWS Lambda в качестве основы для логического уровня. В каждом примере мы применяем только сервисы AWS, не требующие от пользователей управлять собственной инфраструктурой.

Серверная система для мобильного приложения

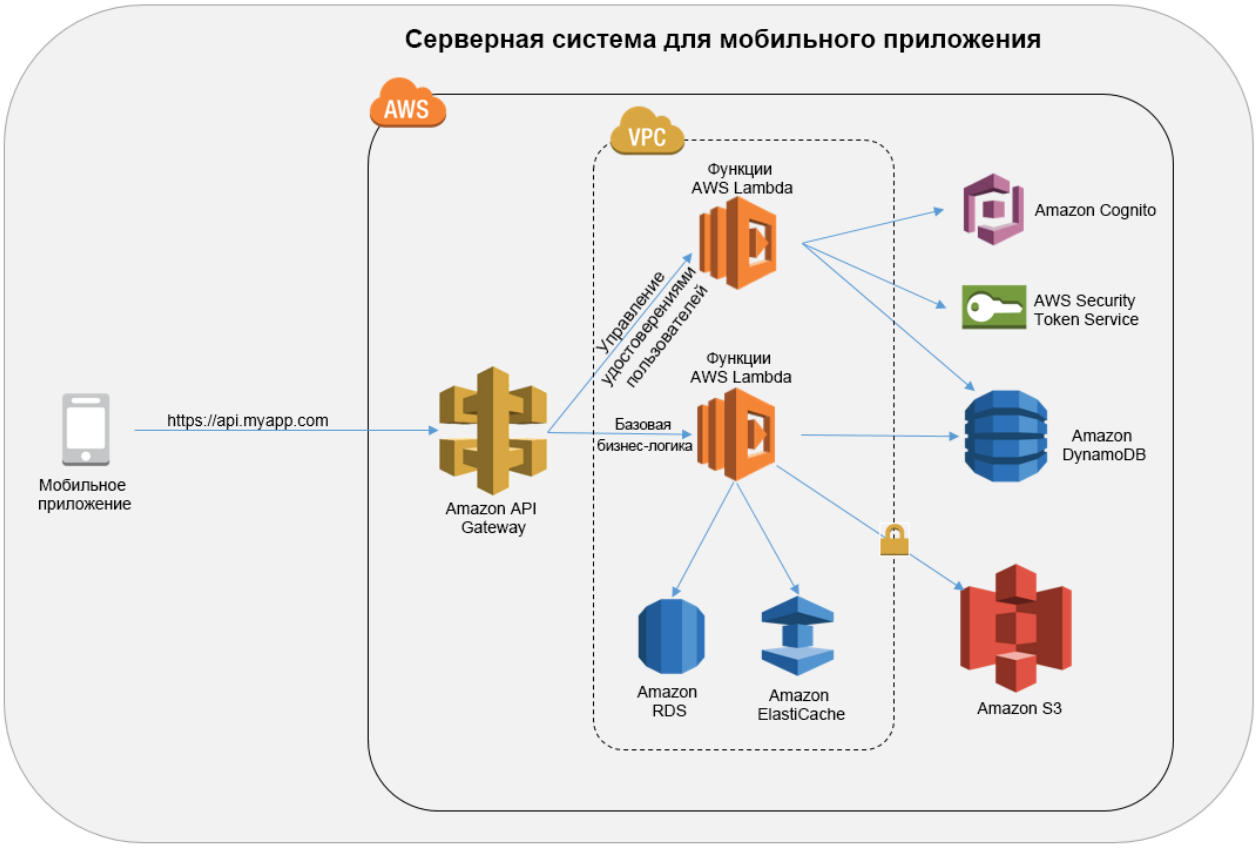

Рис. 3. Шаблон архитектуры серверной системы для мобильных приложений

- **Уровень представления:** мобильное приложение, работающее на смартфонах пользователей.
- **Логический уровень:** Amazon API Gateway и AWS Lambda. Логический уровень глобально распределен базой раздачи Amazon CloudFront, созданной для каждого API Amazon API Gateway. Разные наборы функций Lambda можно использовать для управления удостоверениями пользователей и устройств, а также для аутентификации. При этом управление осуществляет сервис Amazon Cognito, который интегрирован с IAM для предоставления временных данных для доступа пользователей, а также с популярными сторонними поставщиками удостоверений. В других функциях Lambda можно определить базовую бизнес-логику для серверной системы мобильного приложения.

- **Уровень данных:** можно использовать различные хранилища данных, которые описаны ранее в этом документе.

Веб-сайт, размещенный в Amazon S3

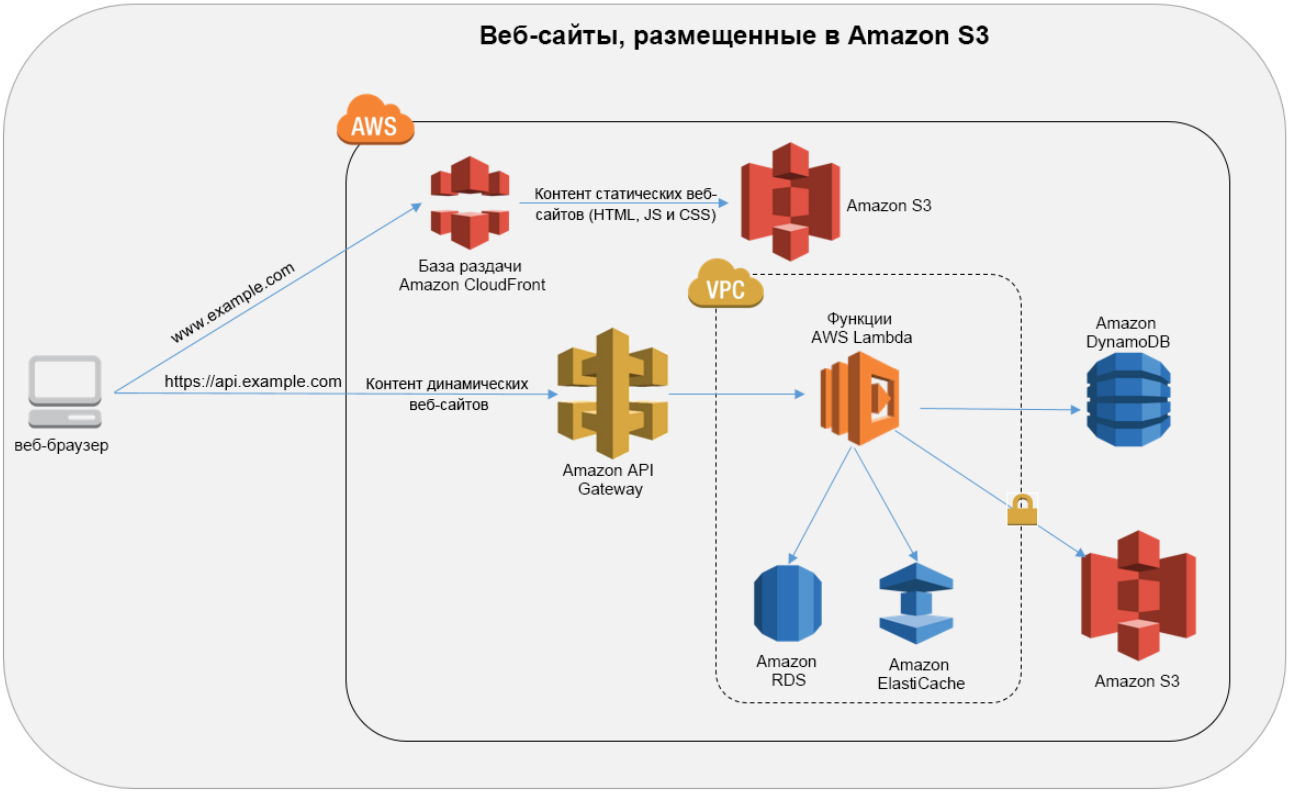

Рис. 4. Шаблон архитектуры статического веб-сайта, размещенного в Amazon S3

- **Уровень представления:** статический контент веб-сайтов, размещенный в Amazon S3 и распределенный Amazon CloudFront. Размещение контента статического веб-сайта в Amazon S3 – это недорогая альтернатива размещению контента в серверной инфраструктуре. Однако если веб-сайт должен поддерживать расширенные возможности, статический контент должен интегрироваться с динамической серверной системой.
- **Логический уровень:** Amazon API Gateway и AWS Lambda. Статический контент веб-сайтов, размещенный в Amazon S3, можно напрямую интегрировать с Amazon API Gateway, что позволит использовать поддержку CORS.

- **Уровень данных:** можно использовать различные хранилища данных, которые описаны ранее в этом документе.

Среда микросервисов

Рис. 5. Шаблон архитектуры среды микросервисов

Шаблон архитектуры **микросервисов** не связан с типичной трехуровневой архитектурой, рассмотренной в этом техническом описании. В архитектуре микросервисов существуют разъединенные программные компоненты, поэтому преимущества многоуровневой архитектуры получают дополнительный импульс. Для создания микросервиса требуется всего лишь API, созданный в Amazon API Gateway, и функции, выполняемые в AWS Lambda. Вы можете использовать эти сервисы для разъединения и фрагментирования среды до требуемого уровня детализации.

В целом, среда микросервисов может вызывать следующие трудности: повторяющиеся затраты на создание каждого микросервиса, проблемы с оптимизацией плотности и использования серверов, сложности при одновременном выполнении нескольких микросервисов и профилирование требований клиентского кода для интеграции с различными отдельными сервисами.

Однако при создании микросервисов на основе бессерверного шаблона AWS эти проблемы гораздо легче устранить, а в некоторых случаях они сразу же отпадают. Шаблон микросервисов AWS упрощает создание каждого последующего микросервиса (Amazon API Gateway даже позволяет клонировать существующие API). Вручную оптимизировать использование серверов с этим шаблоном не требуется. И API Gateway, и Lambda предоставляют простые возможности управления версиями. Наконец, Amazon API Gateway предоставляет программно созданные клиентские SDK для ряда популярных языков, что сокращает затраты на интеграцию.

Заключение

Шаблон многоуровневой архитектуры позволяет создавать простые в обслуживании, разъединенные и масштабируемые компоненты приложения, соответствующие рекомендациям. При создании логического уровня, на котором осуществляется интеграция с Amazon API Gateway и происходят вычисления в AWS Lambda, вы сможете добиться поставленных целей с меньшими затратами. Вместе эти сервисы формируют интерфейсную часть HTTPS API для клиентов и создают безопасную среду в VPC для выполнения бизнес-логики. Это позволяет воспользоваться преимуществами многих популярных сценариев, в которых можно применять управляемые сервисы, а не управлять типичной серверной инфраструктурой самостоятельно.

Авторский коллектив

Данный документ был подготовлен при участии следующих лиц и организаций.

Эндрю Бэйрд (Andrew Baird), специалист по архитектуре решений, AWS

Стефано Бульяни (Stefano Buliani), старший менеджер по продуктам, Tech, AWS Mobile

Вайом Награни (Vyom Nagrani), старший менеджер по продуктам, AWS Mobile

Аджай Нэйр (Ajay Nair), старший менеджер по продуктам, AWS Mobile

Примечания

[1] http://aws.amazon.com/api-gateway/

[2] http://aws.amazon.com/lambda/

[3] https://aws.amazon.com/vpc/

[4] https://aws.amazon.com/cloudfront/

[5] https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf

[6] https://aws.amazon.com/api-gateway/pricing/

[7] http://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-mock-integration.html

[8] http://aws.amazon.com/iam/

[9] http://docs.aws.amazon.com/general/latest/gr/signature-version-4.html

[10] http://docs.aws.amazon.com/lambda/latest/dg/intro-core-components.html#intro-core-components-event-sources

[11] https://aws.amazon.com/s3/

[12] https://aws.amazon.com/kinesis/

[13] https://aws.amazon.com/vpc/

[14] https://aws.amazon.com/rds/

[15] https://aws.amazon.com/elasticache/

[16] https://aws.amazon.com/redshift/

[17] https://aws.amazon.com/ec2/

[18] https://aws.amazon.com/dynamodb/

[19] https://aws.amazon.com/s3/storage-classes/

[20] https://aws.amazon.com/elasticsearch-service/

[21] https://aws.amazon.com/cognito/

amazon web services