Оглавление

Предисловие 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**.

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 этой 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 Manning 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 электронной почты **dmkpress@gmail.com** со ссылкой на подозрительные материалы.

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

Предисловие

Amazon навсегда изменила IT-инфраструктуру, упростив подготовку виртуальных машин в 2007 году. После этого совершенствование архитектуры современных приложений носило постепенный характер. Спустя десятилетие, упростив возможность предоставления отдельных функций, платформа Amazon Lambda дала толчок новой волне глубинных перемен. Эта «бессерверная» экосистема кардинально меняет способы проектирования, разработки и эксплуатации интернет-приложений.

Как один из первых, кто начал использовать эту платформу на практике, я имел честь работать со Слободаном и Александаром и воочию убедиться, насколько сильно влияет «бессерверное» мышление на время выхода на рынок и стоимость эксплуатации. В то же время платформа развивается настолько быстро, что в ней легко запутаться. Чтобы получить настоящие преимущества нового способа работы, разработчики должны пересмотреть стратегии аутентификации, управления сеансами, хранения данных, планирования мощностей и распределения вычислений. В своей книге «Бессерверные приложения на Javascript» Слободан и Александар предлагают первый отчет об этой революции и бесценное руководство для разработчиков на JavaScript, желающих воспользоваться преимуществами платформ нового поколения.

В этой книге мне понравилось, как она помогает быстро организовать выполнение простых операций в AWS Lambda, не влияя на организацию и работу проектов. Многие бессерверные фреймворки приложений абстрагируют службы AWS, что увеличивает риск замкнуться на одном фреймворке, так как экосистема продолжает быстро развиваться. Авторы не заставляют нас принять их выбор фреймворков, но объясняют, как легко использовать все связанные службы. Новичков в AWS эта книга знакомит не только с AWS Lambda, но и с целым рядом связанных служб, таких как DynamoDB (хранилище данных), Cognito (аутентификация), API Gateway (доступ к работающим веб-службам) и Cloudwatch (обработка и планирование событий). Даже решив позднее взять на вооружение другие инструменты, вы сможете сохранить весь код и просто развернуть его немного иначе.

Еще одна веская причина прочитать эту книгу – в ней описывается несколько вариантов практического использования бессерверных платформ, включая веб-API, чат-боты, обработку платежей и управление заказами. Постепенно создавая онлайн-магазин для вымышленной пиццерии, авторы представляют практически готовые компоненты, необходимые для запуска современных бизнес-сценариев в облаке. Этот способ постепенного формирования знаний позволяет авторам по мере обсуждения исследовать все более сложные вопросы разработки, такие как организация автоматического тестирования и разработка приложений с прицелом на простоту сопровождения. Последняя часть книги посвящена стратегиям миграции и отвечает на некоторые наиболее распространенные вопросы от людей, у которых уже есть приложения

на какой-либо другой облачной платформе и которые хотят быстро получить конкурентные преимущества, сократить время выхода на рынок или уменьшить стоимость эксплуатации.

Я надеюсь, что вы получите от этой книги столько же удовольствия, сколько и я, и обнаружите эффективные способы предоставления услуг с помощью программного обеспечения в облаке.

Гойко Аджич (Gojko Adzic),
партнер в Neuri Consulting LLP

Вступление

Мы оба были разработчиками более 10 лет, оба начинали с наших первых компьютеров в 90-х годах, когда писали первые функции на Pascal и BASIC и даже участвовали в соревнованиях по программированию. Но все изменилось, когда появился интернет. Мы сразу начали создавать свои веб-приложения и веб-страницы со статическими HTML и CSS. Когда JavaScript и jQuery превратились в новый стандарт, мы почти сразу переключились на них (правда, один из нас продолжил экспериментировать с Flash и ActionScript). С появлением Node.js мы естественно переключились на этот фреймворк с языков, на которых писали в ту пору, таких как Python и C#. Хотя иногда мы можем написать несколько функций на этих языках, наш переход на Node.js состоялся бесповоротно.

Примерно три года назад мы обратили наше внимание на бессерверные архитектуры. Гойко Аджич познакомил нас с AWS Lambda на примере своей разработки Claudia.js – инструмента развертывания. Мы были поражены, насколько просто и быстро разрабатывать и развертывать бессерверные приложения и насколько легко их масштабировать, и мы вместе с ним начали работать над созданием Claudia Bot Builder.

Изучение бессерверной архитектуры постепенно изменило наш взгляд на создание и поддержку веб-приложений. Внутренние службы сменили бессерверные функции, и, вместо того чтобы писать сценарии на bash, входить на наши серверы и распределять вычислительные мощности, мы перестали заботиться об этих проблемах и сосредоточились больше на бизнес-логике и ценности приложений.

Мы опубликовали наши первые бессерверные веб-приложения и разработали сотни чат-ботов. Наша продуктивность увеличилась почти в пять раз. Это было невероятно. Месяцы, потраченные на изучение настройки и обслуживания серверов приложений с помощью bash, ssh, rsync и т. д., потеряли свою важность. Все изменилось. С нашей точки зрения бессерверная экосистема прошла долгий путь: бессерверными услугами стало проще пользоваться, и с каждым годом становится все больше и больше компонентов для бессерверных приложений (с Amazon re:Invent).

Случилось так много и так быстро – мы сделали нашу карьеру бессерверной. Мы начали вести дискуссии о бессерверных архитектурах, проводить семинары и давать консультации. Мы попытались объединить наш опыт и знания из множества других источников и изложить их в удобном для изучения и понятном формате.

Благодарности

Работать над этой книгой было трудно, так как это наш первый опыт. Некоторые главы переписывались по пять и более раз, чтобы вам, уважаемый читатель, легче было понять и усвоить обсуждаемые в них сведения. Наши друзья и семьи оказывали нам большую поддержку в процессе, и мы хотели бы поблагодарить всех, кто помогал нам на этом пути.

Прежде всего мы хотели бы поблагодарить Гойко Аджича (Gojko Adzic). Он ввел нас в мир без серверов несколько лет назад. Отдельное спасибо за его комментарии к этой книге, такие как: «эта страница ничего не стоит, удалите ее», «не лгите своим читателям о шагах» и т. п. Они были бесценны для нас.

Мы хотели бы поблагодарить нашего редактора из издательства Manning Тони Арритола (Toni Arritola). Спасибо, что помогли нам выбраться из тупика, когда мы застряли на первых нескольких главах, что были терпеливы, когда мы отставали от графика, и поддерживали нас всем необходимым. Вы всегда требовали от нас высокого качества и тем помогали сделать книгу лучше для читателей. Также мы хотим поблагодарить Майкла Стивенса (Michael Stephens) и Берта Бейтса (Bert Bates), которые помогли лучше объяснить детали бессерверных архитектур и сосредоточиться на важных темах. Спасибо сотрудникам издательства Manning, работавшим над выпуском и продвижением книги, это была сплоченная команда. Спасибо техническому корректору Валентину Креттазу (Valentin Crettaz) и техническому редактору Костасу Пассадису (Kostas Passadis) за тщательный анализ кода.

Спасибо также рецензентам из Manning, которые нашли время, чтобы прочитать нашу рукопись на разных этапах, и дали ценные отзывы, в том числе: Арно Бейли (Arnaud Bailly), Барнаби Норман (Barnaby Norman), Клаудио Бернардо Родригес (Claudio Bernardo Rodríguez), Дамиан Эстебан (Damian Esteban), Ден Балия (Dane Balia), Дипак Бхаскаран (Deepak Bhaskaran), Джасба Симпсон (Jasba Simpson), Джереми Ланге (Jeremy Lange), Кай Стрем (Kaj Ström), Кэтлин Р. Эстрада (Kathleen R. Estrada), Кумар Унникришнан (Kumar Unnikrishnan), Лука Меццалира (Luca Mezzalira), Мартин Денерт (Martin Dehnert), Рами Абдельвахед (Rami Abdelwahed), Сурджит Манхас (Surjeet Manhas), Томас Пеклак (Thomas Peklak), Умур Йильмаз (Umur Yilmaz) и Ивон Вивилль (Yvon Vieville).

Спасибо сотрудникам Amazon и AWS, что создали такую потрясающую компьютерную службу: AWS Lambda. Ваши усилия меняют мир.

Наконец, спасибо тетушке Марии и всем другим вымышленным героям этой книги!

Об этой книге

Основная цель книги «*Бессерверные приложения на Javascript*» – обучение и помощь в создании бессерверных приложений Node.js. Она отличается прагматическим подходом и рассказывает о вымышленной пиццерии тетушки Марии, чьи проблемы мы будем пытаться решить с помощью бессерверной архитектуры. Книга начинается с объяснения основ бессерверной архитектуры и потом раз за разом описывает решение каждой проблемы, с которой сталкивается тетушка Мария, с применением отдельных идей бессерверных вычислений, помогая тем самым сформировать ясное представление о приемах создания эффективных бессерверных приложений с Node.js.

Кому адресована книга

Книга «*Бессерверные приложения на Javascript*» предназначена для разработчиков веб-приложений на JavaScript, стремящихся узнать, как создавать бессерверные приложения, и понять, как правильно их организовывать, проектировать и тестировать. В интернете можно найти массу информации о Node.js и множество пособий по созданию простых бессерверных приложений, и тем не менее в этой книге мы последовательно расскажем, как применить все эти знания для создания больших бессерверных приложений с Node.js.

Краткое содержание

Книга делится на 3 части и 15 глав.

В первой части описываются основы бессерверных вычислений и как построить бессерверное приложение с базой данных, как подключиться к сторонним службам, как отладить его, как добавить поддержку авторизации и аутентификации и как работать с файлами.

- *Глава 1* знакомит с бессерверной платформой Amazon Web Services и описывает бессерверные вычисления с применением простых аналогий. Здесь вы также познакомитесь с тетушкой Марией, ее пиццерией и проблемами. Наконец, вы увидите, как выглядит типичное бессерверное приложение Node.js, и узнаете, что такое Claudia.js и как этот инструмент помогает развертывать приложения Node.js в AWS Lambda.
- *Глава 2* иллюстрирует разработку простого Pizzeria API с использованием AWS Lambda, API Gateway и Claudia API Builder. Здесь вы также научитесь одной командой развертывать свои API с помощью Claudia.
- *Глава 3* рассказывает, как в бессерверной архитектуре работают базы данных, а также о том, как подключить Pizzeria API к DynamoDB – бессерверной базе данных, предлагаемой AWS.

- *Глава 4* рассказывает, как подключить Pizzeria API к сторонним службам, таким как служба доставки, а также знакомит с некоторыми распространенными проблемами, с которыми можно столкнуться при использовании объектов `Promise` с Claudia API Builder.
- *Глава 5* покажет, как искать ошибки в бессерверных приложениях, как их отлаживать и какие инструменты отладки имеются в вашем распоряжении.
- *Глава 6* показывает, как реализовать аутентификацию и авторизацию в бессерверном приложении. Здесь вы узнаете, чем отличается аутентификация от авторизации в бессерверной среде, как реализовать механизм веб-авторизации с помощью AWS Cognito и как идентифицировать своих пользователей с помощью социальных сетей.
- *Глава 7* описывает возможности хранения файлов в бессерверном окружении и показывает, как создать отдельную функцию для обработки файлов, которая использует хранилище и предоставляет запрошенные файлы другим вашим функциям в AWS Lambda, составляющим ваш бессерверный API.

Во второй части рассказывается, как создавать дополнительные бессерверные приложения, работающие с теми же ресурсами, как создавать чат-ботов, голосовых помощников, SMS-чат-ботов, как добавить обработку естественного языка и как следует организовывать все эти бессерверные приложения вместе.

- *Глава 8* показывает, как создать свой первый чат-бот для Facebook Messenger и как Claudia Bot Builder поможет вам сделать это, написав всего несколько строк.
- *Глава 9* показывает, как добавить простую обработку текстов на естественном языке в свой чат-бот, подключить чат-бот к базе данных DynamoDB и организовать асинхронную отправку отложенных ответов.
- *Глава 10* показывает, как использовать голосового помощника Alexa и SMS-чат-бота Twilio и как Claudia Bot Builder позволяет сделать это невероятно быстро.

Третья часть охватывает более сложные темы: тестирование бессерверных приложений и миграция существующих приложений в бессерверное окружение. Здесь также даются рекомендации, описываются типичные шаблоны программирования, решения распространенных проблем и приводятся ответы на часто задаваемые вопросы. Еще тут будет представлен пример двух компаний, перешедших на бессерверные вычисления.

- *Глава 11* рассказывает о тестировании бессерверных приложений: как писать бессерверные функции, чтобы упростить их тестирование, и как

выполнять автоматизированные тесты локально. Здесь также рассказывается о гексагональной архитектуре и о том, как реорганизовать бессерверные приложения, чтобы упростить их тестирование и устранять потенциальные риски.

- *Глава 12* посвящена обработке платежей с помощью бессерверных приложений, реализации приема платежей в бессерверном API и описанию требований к безопасности при обработке платежей.
- *Глава 13* расскажет все, что вы должны знать о запуске приложений Express.js в AWS Lambda и бессерверной экосистеме, обслуживании статического контента из приложения Express.js, подключении к MongoDB из бессерверного приложения Express.js и об ограничениях и рисках приложения Express.js в бессерверной экосистеме.
- *Глава 14* рассказывает, с чего начать миграцию существующего приложения в бессерверное окружение, как привести структуру приложения в соответствие с характеристиками провайдера услуг бессерверных вычислений, как организовать архитектуру приложения, чтобы она была ориентирована на бизнес и могла развиваться, как учесть архитектурные различия между бессерверными и традиционными серверными приложениями.
- *Глава 15* рассказывает, как CodePen использует преимущества бессерверных вычислений для своих препроцессоров, обеспечивая обработку сотен миллионов запросов, и как MindMup способна обслуживать 400 000 активных пользователей с командой из двух человек благодаря бессерверным технологиям.

Об исходном коде

Эта книга содержит много примеров исходного кода и в виде листингов, и в виде фрагментов в обычном тексте. В обоих случаях исходный код оформляется `моноширинным шрифтом`, чтобы его можно было отличить от обычного текста. Иногда, чтобы подчеркнуть отличия от предыдущего шага и выделить вновь добавленные особенности, код будет оформляться **`жирным моноширинным шрифтом`**.

Во многих случаях оригинальный исходный код был переформатирован; мы добавили переносы строк и изменили ширину отступов, чтобы уместить строки кода по ширине книжной страницы. Кроме того, мы убрали комментарии из кода, если он описывается в тексте книги. Многие листинги сопровождаются дополнительными аннотациями, подчеркивающими наиболее важные идеи.

Исходный код примеров в книге доступен для загрузки на сайте издательства: https://manning.com/books/serverless-apps-with-node-and-claudiajs.

Автор в сети

Одновременно с покупкой книги «*Бессерверные приложения на Javascript*» вы получаете бесплатный доступ к частному веб-форуму, организованному издательством Manning Publications, где можно оставлять комментарии о книге, задавать технические вопросы, а также получать помощь от автора и других пользователей. Чтобы получить доступ к форуму и зарегистрироваться на нем, откройте в веб-браузере страницу https://forums.manning.com/forums/serverless-apps-with-node-and-claudiajs. Кроме того, узнать больше о правилах поведения на форуме можно по адресу: https://forums.manning.com/forums/about.

Издательство Manning обязуется предоставить своим читателям место встречи, где может состояться содержательный диалог между отдельными читателями и между читателями и автором. Но со стороны автора отсутствуют какие-либо обязательства уделять форуму какое-то определенное внимание – его присутствие на форуме остается добровольным (и неоплачиваемым). Мы предлагаем задавать автору стимулирующие вопросы, чтобы его интерес не угасал!

Другие онлайн-ресурсы

Те, кому понадобится дополнительная помощь, могут:

- перейти на страницу проекта Claudia.js в Gitter: https://gitter.im/claudiajs/claudia, где обычно авторы отвечают на вопросы о Claudia.js, Claudia API Builder и Claudia Bot Builder;
- выполнить поиск по тегу *claudiajs* на сайте Stack Overflow (http://stackoverflow.com/questions/tagged/claudiajs) и найти сообщения с вопросами и ответами, касающимися разработки бессерверных приложений с Node.js и Claudia.js. Здесь же вы сможете помочь другим, столкнувшимся с проблемами, решение которых вам уж известно.

Об авторах

Слободан Стоянович (Slobodan Stojanovic) и **Александар Симович** (Aleksandar Simovic) являются обладателями титула AWS Serverless Heroes и основными участниками проекта Claudia.js. Они занимаются разработкой и сопровождением Claudia Bot Builder и являются соавторами книги «Бессерверные приложения на Javascript».

Александар больше семи лет работал старшим консультантом и инженером по программному обеспечению, но не только на JavaScript. Он также увлекается языками Swift, Python и Rust. Живет в Белграде и является соорганизатором встреч JS Belgrade.

Слободан – технический директор Cloud Horizon, студии разработки программного обеспечения, базирующейся в Монреале. Живет в Белграде и является соорганизатором встреч JS Belgrade.

Об иллюстрации на обложке

На обложке книги «*Бессерверные приложения на Javascript*» изображен рисунок с названием «Сербка из Шумадии». Иллюстрация взята из книги «Сербские национальные костюмы» Владимира Кирина (Vladimir Kirin). Кирин (1894–1963) изучал графический дизайн в Лондоне, посещал Академию художеств в Вене, работал художником и иллюстратором. Считается, что его работа способствовала совершенствованию оформления книг в Хорватии.

На протяжении всей своей богатой истории центральный регион Сербии, известный как Шумадия, был культурным центром, и традиционная одежда этого района является стандартом национального костюма Сербии. Как показано на этом изображении, традиционное сербское женское платье состояло из *опанчи*, вышитых шерстяных носков, доходивших до колен. Юбки были разнообразные, плиссированные или собранные в складки и вышитые, с *тканицей*, служившей поясом. Важной частью костюма был *передник*, украшенный цветочными мотивами. Рубашки были в форме туники, богато украшенные серебряной нитью, а поверх рубашки надевались шнуры. Девушки также носили нашейные украшения или монисто из золотых монет вокруг горла, сережки, браслеты, а их волосы были украшены металлическими монетами или цветами.

Мы в издательстве Manning славим изобретательность, предприимчивость и радость компьютерного бизнеса обложками книг, изображающими богатство региональных различий двухвековой давности, оживших благодаря Кирин.

Часть I

Бессерверная пиццерия

Тетушка Мария – волевой человек. Вот уже тридцать лет она управляет своей пиццерией, местом сбора людей из разных поколений, живущих неподалеку: многие проводили там время со своими семьями, смеялись и даже ходили на романтические свидания. Но в последнее время в ее пиццерии наступили тяжелые времена. Число посетителей постепенно уменьшается. Развитие технологий привело к тому, что ее клиенты предпочитают делать заказы онлайн через веб-сайты или телефоны в пиццериях конкурентов.

У ее пиццерии уже есть веб-сайт, но для него нужно написать приложение, обрабатывающее и хранящее информацию о пиццах и заказах.

Наша задача в первой части этой книги – помочь тетушке Марии наверстать упущенное, создав для нее бессерверный API. Но так как вы все еще новичок в разработке бессерверных приложений, сначала мы расскажем вам, что такое бессерверные вычисления и как они могут помочь в создании Pizzeria API (глава 1). Затем вы добавите маршруты в свой API и развернете его в AWS Lambda с помощью Claudia (глава 2). Чтобы сохранить и выполнить все заказы, необходимо связать новый API с таблицей DynamoDB (глава 3) и установить связь со сторонней службой доставки (глава 4).

Во время разработки вы столкнетесь с некоторыми проблемами и узнаете, как отлаживать бессерверные приложения (глава 5).

Чтобы сделать API полностью функциональным, нужно научиться аутентифицировать и авторизовать пользователей (глава 6), а также сохранять и манипулировать изображениями пиццы (глава 7).

Глава 1

Введение в бессерверные вычисления с Claudia

Эта глава охватывает следующие темы:

- что такое бессерверные вычисления;
- основные понятия бессерверных вычислений;
- различия между бессерверными и серверными веб-приложениями;
- назначение Claudia;
- преимущества бессерверных вычислений.

Бессерверные вычисления – это способ развертывания и выполнения приложений в облачной инфраструктуре с оплатой за фактическое использование, без аренды или покупки серверов. За планирование, масштабирование, балансировку и мониторинг вычислительных мощностей отвечает поставщик услуг бессерверных вычислений. Кроме того, поставщик может рассматривать ваши приложения как функции.

Что значит *бессерверные*? Создается впечатление, что это еще одно новомодное словцо, обещающее улучшить вашу жизнь.

В этой книге объясняется, что такое бессерверные вычисления, какие проблемы они решают и где могут или не могут использоваться для ваших приложений, без рекламной шелухи и попытки втюхать вам бессерверные подходы как какой-то модный облачный культ, которому должен следовать каждый. Мы предпримем более прагматический подход и объясним понятия, попутно демонстрируя приемы создания надежных и масштабируемых бессерверных приложений с Node.js и Claudia.js.

В этой главе основное внимание уделяется понятию «бессерверные вычисления»: что это такое, почему важно иметь представление о них и в чем их отличие и сходство с серверными вычислениями. Ваша главная цель в этой

главе – получить хорошее представление об основных понятиях бессерверных вычислений и заложить прочный фундамент для будущих глав.

1.1. Серверы и стиральные машины

Чтобы понять идею бессерверных вычислений, рассмотрим пример со стиральными машинами. Устройство стирки одежды может показаться странной аналогией, но владение сервером в настоящее время похоже на владение стиральной машиной. Всем нужна чистая одежда, и покупка стиральной машины кажется вполне логичным решением. Но большую часть времени стиральная машина простаивает без дела. В лучшем случае она используется от 5 до 15 часов в неделю. То же самое касается серверов. В большинстве случаев средний сервер приложений просто ждет получения запроса, ничего не делая.

Самое интересное, что у серверов и стиральных машин много общих проблем. И те, и другие имеют предельный вес или объем, который они могут обработать. Владение небольшим сервером аналогично владению небольшой стиральной машиной; если накапливается большая куча белья, машина не сможет обработать все сразу. Вы можете купить большую машину, способную выстирать сразу до 8 кг одежды, но тогда возникнет другая проблема – запуск большой машины для стирки единственной рубашки окажется слишком расточительным. Кроме того, настроить единственный сервер для безопасного и надежного выполнения всех имеющихся приложений сложно, а иногда невозможно. Правильная настройка для одного приложения может совершенно не подходить для другого. Точно так же перед стиркой требуется рассортировать одежду по цвету, а затем выбрать правильную комбинацию программы, моющего средства и смягчителя. Если выбрать неправильные настройки или моющие компоненты, машина может испортить вашу одежду.

Эти проблемы, а также проблема, заключающаяся в том, что не каждый может купить стиральную машину, привели к росту количества прачечных самообслуживания или прачечных-автоматов, предлагающих платные услуги стиральных машин для стирки одежды. Аналогичная потребность в отношении серверов привела к появлению компаний, предоставляющих услуги аренды серверов, как локальных, так и в облаке. Вы можете арендовать сервер, а поставщик услуги позаботится об их сохранности, электропитании и основных настройках. Но как прачечные, так и прокатные серверы решают лишь часть проблем.

Арендуя стиральные машины или серверы, вы все равно должны знать, как сочетать одежду или приложения и выбирать программу стирки, подходящие моющие средства или настраивать среду выполнения на серверах. Вы также должны учесть количество машин и их ограничения по размеру, заранее планируя, сколько машин арендовать.

В мире бытовых услуг во второй половине XX века появилась новая услуга – услуга по вызову. Суть ее заключается в следующем: вы собираете белье в кучу и звоните в прачечную, а служащие прачечной заберут его, постирают, высушат, сложат в аккуратную стопку и, если вы пожелаете, доставят постиранное

белье вам на дом. Сдавать белье в стирку часто можно поштучно, то есть вам не нужно ждать, пока накопится определенный объем грязного белья, и не нужно беспокоиться о стиральных машинах, моющих средствах и программах стирки.

В отличие от индустрии бытовых услуг, индустрия программного обеспечения все еще находится на стадии прачечных самообслуживания, так как многие из нас все еще арендуют серверы или пользуются услугами PaaS (платформа как услуга). Мы по-прежнему должны оценить количество потенциальных запросов (количество одежды), которые собираемся обработать, и резервируем достаточное число серверов, чтобы (мы надеемся) справиться с нагрузкой, часто тратя наши деньги на серверы, которые либо не работают на полную мощность, либо перегружены и не могут обработать все запросы наших клиентов.

1.2. Основные понятия

Итак, что меняет внедрение бессерверных вычислений? Судя по названию технологии, она подразумевает отсутствие серверов, что выглядит более чем нелогично. Вернемся к определению, которое было дано в начале главы:

> **Что такое бессерверные вычисления?**
>
> *Бессерверные вычисления* – это способ развертывания и выполнения приложений в облачной инфраструктуре с оплатой за фактическое использование, без аренды или покупки серверов.

Вопреки своему названию, технология бессерверных вычислений не исключает существования серверов; программному обеспечению нужна аппаратура, на которой оно будет выполняться. Под словом «бессерверный» просто подразумевается, что компаниям, организациям или разработчикам не требуется приобретать либо арендовать физический сервер.

Возможно, вам интересно узнать, почему было выбрано такое название. Потому что эта технология основывается на абстрагировании от понятия «сервер». Вместо приобретения или аренды сервера для своего приложения, настройки и развертывания среды выполнения вы просто выгружаете приложение в облако поставщика услуг бессерверных вычислений, который сам позаботится о том, чтобы выделить серверы, хранилища, настроить среду выполнения для приложения и запустить его.

> **ПРИМЕЧАНИЕ.** Кому-то из вас может быть интересно, избавляет ли технология бессерверных вычислений от необходимости иметь в компаниях свои подразделения сопровождения и эксплуатации программного обеспечения. В большинстве случаев ответ на этот вопрос: да, избавляет.

Точнее, поставщик услуг сохранит ваше приложение в некотором контейнере. Контейнер представляет изолированное окружение, содержащее все, необходимое вашему приложению для работы. Контейнер можно представить как горшок для комнатного растения. Он содержит грунт со всеми питательными веществами, необходимыми вашему растению.

Как и горшок с растением, контейнер позволяет поставщику услуги бессерверных вычислений безопасно перемещать и хранить ваше приложение, а также выполнять его и копировать в зависимости от ваших потребностей. Но главное преимущество бессерверной технологии – отсутствие необходимости выполнять какие-либо настройки, балансировать, масштабировать серверы, то есть решать любые задачи управления сервером. Провайдер сам управляет всем этим за вас, а также гарантирует, что с увеличением нагрузки на ваше приложение он создаст достаточное количество копий контейнера с приложением для обработки всех вызовов, и каждый контейнер будет точной копией исходного. Если потребуется, провайдер создаст тысячи копий. Решение о запуске еще одной копии контейнера принимается провайдером, только когда количество запросов к вашему приложению становится настолько большим, что текущее число действующих контейнеров не успевает обработать их все.

Если к вашему приложению вообще не поступает запросов, провайдер остановит все экземпляры, соответственно, оно не будет расходовать память и процессорное время сервера. Провайдер услуги бессерверных вычислений отвечает за все детали, касающиеся работы: он знает, где хранится ваше приложение, как и куда его копировать, когда запускать новые контейнеры и когда уменьшать количество контейнеров при снижении нагрузки.

Продолжая аналогию со стиральными машинами, процесс предоставления услуги бессерверных вычислений напоминает услуги прачечной по вызову; служащий прачечной появляется у вашей двери, чтобы забрать грязное белье, затем оно стирается в прачечной и потом возвращается к вам. Независимо от того, сколько у вас одежды и каких видов (шерсть, хлопок, кожа и т. д.), прачечная берет на себя всю ответственность за сортировку белья, выбор моющих средств и программ.

Бессерверные вычисления и FaaS

Первоначально термин «бессерверные вычисления» интерпретировался иначе, чем сейчас. Подразумеваемая под ним технология называлась *сервер как услуга* (Backend as a Service, BaaS) и предназначалась для приложений, которые частично или полностью зависят от сторонних услуг по предоставлению серверной логики. Позднее эта технология стала называться *функция как услуга* (Function as a Service, FaaS), поскольку провайдеры услуг бессерверных вычислений интерпретируют приложения как функции, вызывая их только по запросу.

1.3. Как работают бессерверные вычисления?

Как отмечалось выше, провайдеры услуг бессерверных вычислений предлагают изолированные контейнеры для приложений. Контейнер управляется событиями, поэтому активируется только при появлении определенного события.

События – это конкретные внешние воздействия, подобные физическим выключателям. Возьмем в качестве примера освещение в доме: события, включающие его, могут отличаться. Свет может быть включен человеком, щелкнувшим обычным выключателем; датчиком движения; датчиком освещенности, включающим свет, когда садится солнце. Но контейнеры не ограничиваются приемом определенных событий и вызовом содержащихся в них функций; они также позволяют вашим функциям самим создавать события или, точнее, их генерировать. В техническом смысле в бессерверных вычислениях контейнеры функций являются и *приемниками*, и *источниками событий*.

Наконец, провайдеры предлагают различные события, способные запускать ваши функции. Список событий зависит от провайдера и реализации, но часто их роль играют HTTP-запросы, выгрузка файлов в хранилище, обновление базы данных, события интернета вещей (Internet of Things, IoT) и множество других.

> **ПРИМЕЧАНИЕ.** Бессерверные функции запускаются только по событиям, и вы платите лишь за время выполнения. После выполнения провайдер отключает функцию, сохраняя возможность повторного ее запуска по следующему событию.

1.4. Бессерверные вычисления на практике

Ландшафт бессерверных вычислений содержит множество движущихся частей, поэтому далее мы познакомимся с ними поближе. С этой целью мы создадим пример приложения, разрабатывая его поэтапно, чтобы вы могли видеть, как оно конструируется. По мере знакомства с новыми понятиями мы будем расширять пример приложения.

В этой книге мы напишем пример совершенно нового приложения (оно будет создано «с нуля»), решающего проблемы небольшой компании, а точнее – пиццерии. Пиццерия управляется вашей вымышленной тетушкой Марией. В течение книги тетушка Мария столкнется со множеством реальных проблем, и наша цель – помочь ей в этом, попутно усваивая понятия бессерверных вычислений. Бессерверные вычисления, как и любая новая технология, вводят множество новых понятий, с которыми сложно справиться, если рассматривать их все сразу.

> **ПРИМЕЧАНИЕ.** В случае проблем при переносе существующего приложения на бессерверную платформу не стесняйтесь обращаться к последней части книги. Если вы пока незнакомы с бессерверными вычислениями, прочитайте хотя бы несколько первых глав, прежде чем переходить к последней части книги.

1.4.1. Бессерверная пиццерия тетушки Марии

Тетушка Мария – волевой человек. Вот уже тридцать лет она управляет своей пиццерией, местом сбора людей из разных поколений, живущих неподалеку: многие проводили там время со своими семьями, смеялись и даже ходили на романтические свидания. Но в последнее время в ее пиццерии наступили тяжелые времена. Число посетителей постепенно уменьшается. Многие из ее клиентов теперь предпочитают делать заказы онлайн, через веб-сайты или телефоны в пиццериях конкурентов. Некоторые новые компании начали переманивать ее клиентов. Например, в новой пиццерии Chess запустили мобильное приложение с предварительным просмотром пиццы и возможностью сделать заказ онлайн, а также чат-бота для заказов через различные приложения мгновенного обмена сообщениями. Клиенты нашей тетушки любят ее пиццерию, но многие предпочитают заказывать пиццу с доставкой на дом, поэтому ее тридцатилетний бизнес начал угасать. В пиццерии уже есть веб-сайт, но для обработки и хранения информации о пиццах и заказах требуется внутреннее (серверное) приложение.

1.4.2. Распространенный подход

Учитывая ограниченность ресурсов тетушки Марии, самым простым решением является создание небольшого API с использованием популярного фреймворка Node.js, такого как Express.js или Hapi, и настройка базы данных (скорее всего, MongoDB, MySQL или PostgreSQL).

Код типичного API делится на несколько уровней и напоминает трехуровневую архитектуру. То есть код должен делиться на такие уровни, как представление, бизнес-логика и данные.

> **Трехуровневая архитектура**
>
> *Трехуровневая архитектура* – это шаблон архитектуры клиент-серверного программного обеспечения, в котором пользовательский интерфейс (представление), логика работы («бизнес-правила»), хранение данных и доступ к ним разрабатываются и поддерживаются как независимые модули, чаще всего на отдельных платформах.
>
> Узнать больше о трехуровневой архитектуре можно на странице Википедии https://ru.wikipedia.org/wiki/Трехуровневая_архитектура.

На рис. 1.1 изображен дизайн типичного трехуровневого приложения с отдельными маршрутами для пиццы, заказов и пользователей. В нем также должны иметься точки входа для чат-ботов и обработчика платежей. Все маршруты должны запускать некоторые функции-обработчики на уровне бизнес-логики, а результаты обработки – отправляться на уровень данных, в базу данных и хранилище файлов и изображений.

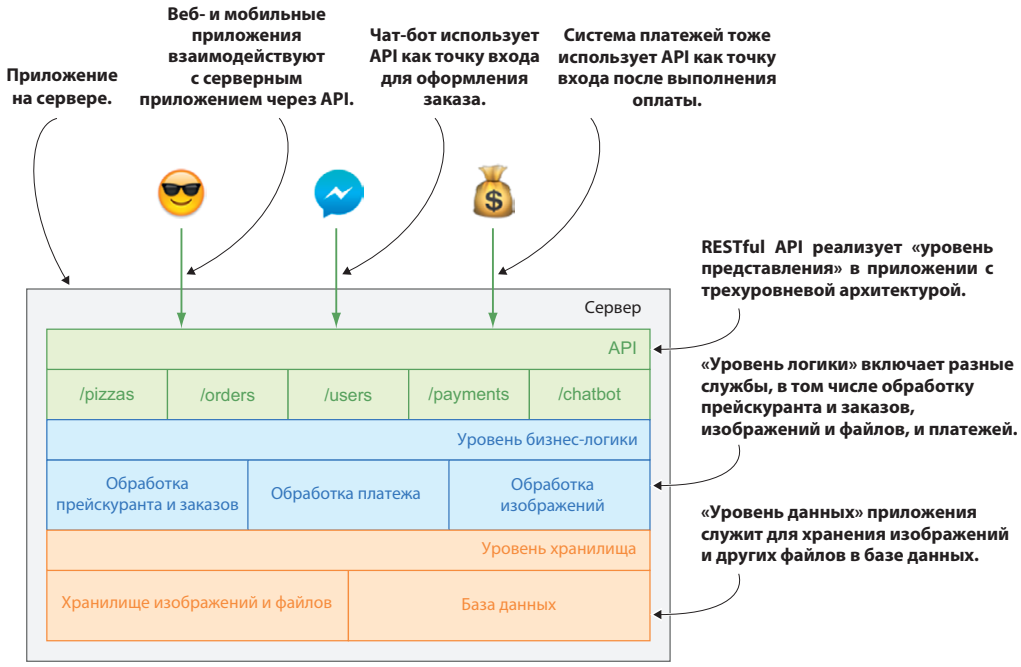

Рис. 1.1. Pizza API с типичным трехуровневым дизайном

Этот подход идеален для небольших приложений, в том числе и для нашего Pizza API, по крайней мере до тех пор, пока количество заказов на пиццу не вырастет до определенного уровня. После этого вам потребуется масштабировать инфраструктуру.

Но, чтобы масштабировать монолитное приложение, необходимо отделить слой данных (чтобы не копировать базу данных ради согласования данных). После этого приложение будет выглядеть так, как показано на рис. 1.2. И у нас все равно остается единый конгломерат со всеми маршрутами и бизнес-логикой. Такое приложение можно реплицировать (запускать дополнительные копии, чтобы увеличить пропускную способность), если у вас слишком много пользователей, но в каждом экземпляре будут присутствовать все службы приложения, независимо от интенсивности их использования.

Монолитное приложение

Монолитным называют приложение, в котором интерфейс пользователя и код доступа к данным объединены в одну программу на одной платформе. Монолитное приложение является автономным и независимым от других приложений.

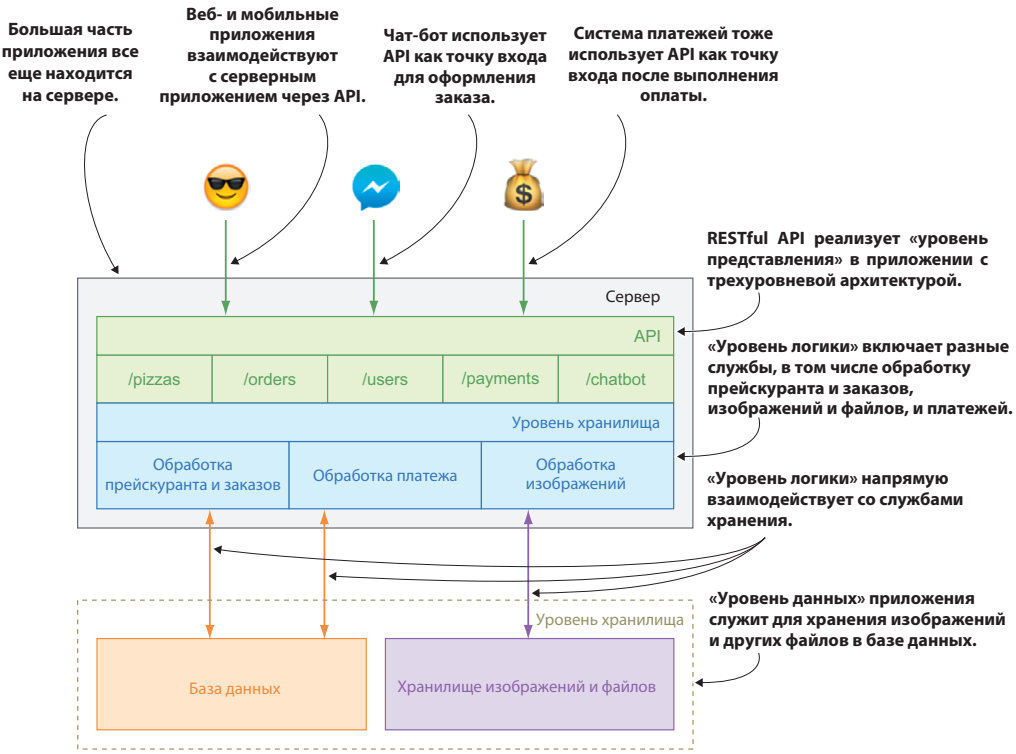

Рис. 1.2. Типичная архитектура с внешней базой данных и хранилищем файлов

1.4.3. Бессерверное решение

Для использования бессерверных вычислений требуется иной подход, так как приложения в этом случае управляются событиями и являются распределенными.

Все части бессерверного приложения с конечными точками API и бизнес-логикой изолируются в независимых и автоматически масштабируемых контейнерах.

В бессерверном приложении запросы обрабатываются на уровне маршрутизатора API, который решает единственную задачу: принимает HTTP-запросы и направляет их в службы уровня бизнес-логики. Маршрутизатор API в бессерверной архитектуре всегда действует независимо. Это означает, что от разработчиков приложений не требуется поддерживать маршрутизацию API, и поставщик услуг бессерверных вычислений автоматически масштабирует приложение, чтобы обеспечить своевременную обработку всех HTTP-запросов, поступающих в адрес вашего API. Также вы платите только за запросы, которые обрабатываются.

В примере нашего Pizza API маршрутизатор будет принимать все запросы от мобильных и веб-приложений и, если необходимо, обслуживать точки входа для чат-ботов и системы обработки платежей.

После получения запроса маршрутизатор передает его для обработки в другой контейнер со службой уровня бизнес-логики.

В бессерверных приложениях бизнес-логика часто разбивается на более мелкие единицы. Размер каждой единицы зависит от предпочтений разработчика. Единицей может быть единственная функция или целое монолитное приложение. В большинстве случаев размер единицы не влияет напрямую на сумму оплаты услуг инфраструктуры, поскольку вы платите за выполнение функции. Кроме того, единицы масштабируются автоматически, и вам не придется платить за единицы, которые ничего не обрабатывают, поэтому владение одной или дюжиной единиц обходится одинаково.

Однако в ситуациях с небольшими приложениями или когда обрабатывается не очень большой объем информации, можно сэкономить на хостинге и обслуживании, объединив функции, связанные с одной службой, в одну бизнес-единицу. Для Pizza API вполне разумно будет создать одну единицу для обработки прейскуранта и заказов, одну для обработки платежей, одну для обработки сообщений от чат-бота и одну для обработки изображений и файлов.

Последняя часть нашего бессерверного API – уровень данных, который мало чем отличается от уровня данных в масштабируемом монолитном приложении с отдельно масштабируемой базой данных и службой хранения файлов. Было бы лучше, если бы база данных и хранилище файлов были также независимыми и автоматически масштабируемыми.

Еще одно преимущество бессерверных приложений – уровень данных может вызывать бессерверную функцию «из коробки». Например, когда в хранилище выгружается изображение пиццы, есть возможность запустить службу обработки изображений, которая изменит размер фотографии и свяжет ее с конкретной пиццей в прейскуранте.

Потоки обработки данных в бессерверном Pizza API можно видеть на рис. 1.3.

1.5. Бессерверная инфраструктура – AWS

Нашему бессерверному Pizza API нужна инфраструктура для работы. Технология бессерверных вычислений еще очень молода, и на данный момент имеется лишь несколько вариантов инфраструктуры. Большинство вариантов принадлежит крупным поставщикам, поскольку для бессерверных вычислений требуется большая и развитая инфраструктура с поддержкой масштабирования. Самыми известными и наиболее продвинутыми инфраструктурами являются бессерверный контейнер Amazon AWS Lambda, Microsoft Azure Functions и Google Cloud Functions.

В этой книге мы будем использовать AWS Lambda, потому что это самая зрелая из доступных на рынке бессерверных инфраструктур, она имеет стабильный API и множество успешных историй использования.

AWS Lambda – это бессерверная вычислительная платформа, управляемая событиями, которая предлагается компанией Amazon как часть Amazon Web

Services. Это вычислительная служба, которая запускает код в ответ на события и автоматически управляет вычислительными ресурсами, необходимыми этому коду.

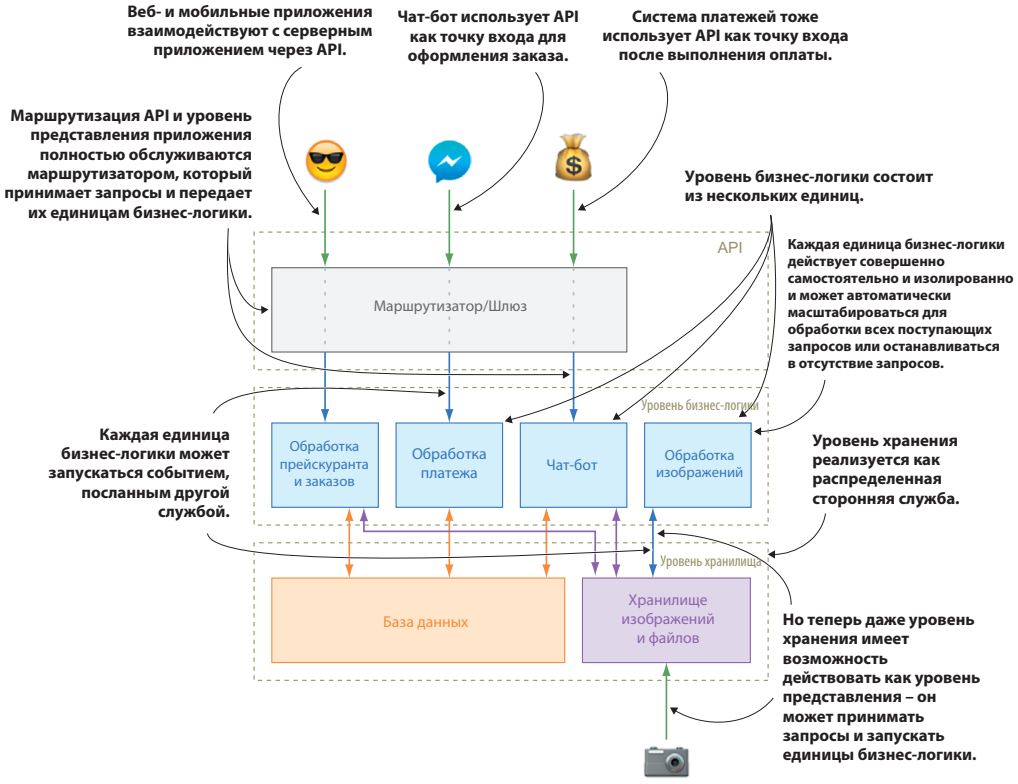

Рис. 1.3. Бессерверная реализация Pizza API

Google Cloud Functions и Microsoft Azure Functions

Компания Google запустила услугу Google Cloud Functions в середине 2016 года, в ответ на появление Amazon AWS Lambda. Инфраструктура Google Cloud Functions позиционируется как набор легковесных микросервисов, управляемых событиями, которые позволяют запускать функции на JavaScript в среде выполнения Node.js. Ваша функция может быть вызвана в ответ на HTTP-запрос, событие из Google Cloud Storage и других служб Google Cloud Pub/Sub. На момент написания этих строк инфраструктура Google Cloud Functions еще находилась в стадии разработки, поэтому цены не были известны. Дополнительные подробности вы можете узнать на официальном сайте: https://cloud.google.com/functions/.

Реализация бессерверных вычислений от корпорации Microsoft – Azure Functions – является частью ее платформы облачных вычислений Azure.

Microsoft описывает ее как инфраструктуру бессерверных вычислений, управляемых событиями, которая ускоряет разработку, осуществляет масштабирование в зависимости от спроса и взимает плату только за фактически потребленные ресурсы. Инфраструктура Azure Functions позволяет писать функции на JavaScript, C#, F#, Python и других языках сценариев. Цены на услуги Azure аналогичны ценам на AWS Lambda: вы платите 20 центов за 1 миллион выполнений и 0,000016 доллара США за гигабайт потребленных ресурсов в месяц, при этом обработка первого миллиона запросов и 400 000 Гбайт каждый месяц предоставляются бесплатно. За дополнительной информацией обращайтесь на официальный веб-сайт: https://azure.microsoft.com/ru-ru/services/functions/.

ПРИМЕЧАНИЕ. Большинство из того, с чем вы познакомитесь в этой книге, также можно реализовать с помощью других поставщиков услуг бессерверных вычислений, но некоторые услуги могут отличаться, поэтому для отдельных решений может потребоваться использовать немного иной подход.

На платформе Amazon слово *бессерверные* обычно напрямую связано с AWS Lambda. Но для бессерверных приложений, таких как Pizza API, AWS Lambda является лишь одним из строительных блоков. Чтобы создать полноценное приложение, часто нужны другие службы, такие как службы хранения, маршрутизации и базы данных. В табл. 1.1 перечислены все необходимые службы, предлагаемые платформой AWS:

- Lambda – используется для вычислений;
- API Gateway (шлюз API) – выполняет маршрутизацию, принимая HTTP-запросы и вызывая другие службы в зависимости от маршрута;
- DynamoDB – автоматически масштабируемая база данных;
- Simple Storage Service (S3) – служба хранилища, реализующая абстракцию обычного жесткого диска и предлагающая неограниченное пространство для хранения.

Таблица 1.1. Строительные блоки бессерверных приложений в AWS

Назначение	Служба AWS	Краткое описание
Вычисления	Lambda	Вычислительный компонент для бизнес-логики
Маршрутизация	API Gateway	Компонент маршрутизации, используется для маршрутизации HTTP-запросов к функциям Lambda
База данных	DynamoDB	Автоматически масштабируемая документ-ориентированная база данных
Хранилище	S3	Служба автоматически масштабируемого хранилища файлов

Lambda – это самая важная часть инфраструктуры бессерверных вычислений, с которой вы обязательно должны разобраться, потому что она содержит вашу бизнес-логику. Lambda – это бессерверный вычислительный контейнер AWS, который запускает вашу функцию по событию. Он автоматически масштабируется, если в функцию поступает сразу большое число событий. Чтобы создать Pizza API в виде бессерверного приложения, необходимо использовать вычислительный бессерверный контейнер AWS Lambda.

Когда возникает определенное событие, такое как HTTP-запрос, служба Lambda запускает функцию и передает ей аргументы с данными из события, контекстом и функцию обратного вызова для отправки ответа. Функция в терминологии Lambda – это самая обычная функция-обработчик, написанная на одном из поддерживаемых языков. На момент написания этих строк AWS Lambda поддерживала следующие языки:

- Node.js;
- Python;
- Java (Java 8) и другие языки JVM;
- C# (.NET Core).

В Node.js данные события, контекст и функция обратного вызова передаются как объекты JSON. Объект контекста `context` содержит подробную информацию о вашей функции и ее текущем выполнении, такую как время выполнения, причина вызова функции и т. д. Третий аргумент, который получает ваша функция, – это функция обратного вызова, которая позволит вам вернуть ответ с некоторой полезной информацией или ошибкой, которая будет отправлена обратно, службе, сгенерировавшей событие. В листинге 1.1 показан пример небольшой функции AWS Lambda, которая возвращает текст *Hello from AWS Lambda*.

Листинг 1.1. Пример минимальной действующей функции Lambda на Node.js

```
function lambdaFunction(event, context, callback) {
    callback(null, 'Hello from AWS Lambda')
}

exports.handler = lambdaFunction
```

Функция принимает событие, контекст и функцию обратного вызова.

Функция обратного вызова вызывается для возврата сообщения об успехе.

Функция экспортируется как обработчик.

ПРИМЕЧАНИЕ. Как показано в листинге 1.1, функция должна экспортироваться присваиванием ссылки на нее свойству `exports.handler`, а не `module.exports`, стандартному свойству для Node.js. Это объясняется тем, что в AWS Lambda экспортируемый элемент должен быть объектом с методом `handler`, а не функцией непосредственно.