

Mastering AWS Serverless

*Architecting, developing, and deploying
serverless solutions on AWS*

Miguel A. Calles



www.bpbonline.com

First Edition 2024

Copyright © BPB Publications, India

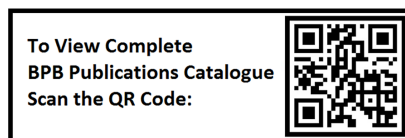
ISBN: 978-93-55516-114

All Rights Reserved. No part of this publication may be reproduced, distributed or transmitted in any form or by any means or stored in a database or retrieval system, without the prior written permission of the publisher with the exception to the program listings which may be entered, stored and executed in a computer system, but they can not be reproduced by the means of publication, photocopy, recording, or by any electronic and mechanical means.

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.



Dedicated to

My wife and children

About the Author

Miguel A. Calles is an AWS Community Builder with a focus on serverless and a published author on serverless security. Miguel works on cloud computing and cybersecurity projects and has been writing on both topics since 2016. He has worked on multiple serverless projects since 2018 as a developer and security engineer. Miguel is currently a Senior Systems Engineer and Security Analyst at Iris Technology and was formerly the Chief Technology Officer at VeriToll building serverless solutions. Miguel has a Bachelor of Science degree in Material Science and Engineering from the Massachusetts Institute of Technology, a Master of Business Administrator degree from the University of Florida, a Cloud Security Alliance's Certificate of Cloud Security Knowledge certification, and an AWS Certified Solution Architect Associate certification. For fun, Miguel enjoys bowling with his family.

About the Reviewer

Syed is a pioneer and **subject matter expert (SME)** in application modernization and cloud computing. He is a holder of multiple industry certifications in Azure, AWS, GCP, Oracle, PMP, CSM. With more than a decade of expertise in product development, architecture, scalability, and modernization, Syed has held positions with Oracle, the Big 4 technology advisory companies like PwC and Ernst & Young, as well as other contemporary application enterprises. Syed has also published articles in leading Journal of Cloud Computing. Syed is a skilled and devoted technology management executive who employs his more than 15 years of expertise in the design and development of high-demand On Premise / Commercial of the Shelf Software / SaaS solutions. He possesses specialized knowledge in designing and constructing highly scalable and highly available systems. In order to assist them in establishing and managing their businesses by taking control of their frequently cumbersome applications, Syed has provided consulting services to some of the world's top corporations. Syed assists businesses in modernizing their applications by emphasizing scale and availability, aiding cloud migration, DevOps transformations, and risk-based problem identification. Syed offers expert guidance, strategy, and thought leadership to assist you in solving application modernization issues. Syed promotes change by collaborating with enterprises of all sizes and executive team

Acknowledgement

I thank my wife and children for supporting and believing in me. Writing a book takes time and energy; sometimes it meant having less family time with you. Thank you for supporting me even when it meant I was sometimes unavailable.

I am grateful to BPB Publications for presenting the opportunity to write on serverless, a topic I enjoy. Their team has been professional and diligent from start to finish. Their review processes provided me with valuable feedback that shaped this book well.

Thank you to my colleagues at Raytheon and Secjuice, who guided me along my writing journey, and my colleagues at Transurban, VeriToll, and Iris Technology, who provided me with opportunities to work on serverless to create solutions for their missions. Thank you to the staff at AWS for building great solutions and providing me with resources to create serverless solutions and accepting me into the AWS Community Builders program.

Thank you to my readers who took an interest in this book. Thank you to the audience of my blog posts who took the time to read my content, share it on social media, and provide feedback. This support encourages me to continue writing more and sharing what I know and learn.

I appreciate my parents, sister, and in-laws for being there for me.

Last but not least, I thank my Creator for another opportunity to continue writing and growing.

Preface

When I pondered the title, **Mastering AWS Serverless**, I tried to determine what it takes to become a master at anything. I envisioned a master wearing a martial arts suit with black belts and teaching a group of young learners. The master would be telling me to execute one single motion repeatedly. The repetition taught by the master would shape these learners to perform those motions with perfection. Over time, the exercises would eventually shape those learners into masters if they continued. Therefore, this book needed to have exercises.

The martial arts dojo is not easily translated into a book, and repeated reading exercises several times can become incredibly dull. Yet, repetition is a crucial activity to master any skill. Thus, I provided as many hands-on exercises as possible. To keep the book interesting, I created a variety of exercises. Yet, there was some inherent repetition: modifying the serverless function settings and defining security permissions. By the end of the book, you will have created a basic serverless application and have done some “repetition exercises” to help you start your journey to become a master in AWS serverless.

Another reason for writing this book is to help you understand AWS serverless. Even with its popularity, only a minority of developers know about it and understand how to use it. This book aims to bridge the gap by explaining concepts and building an application. Understanding concepts is fundamental, but many developers (including myself) learn best with hands-on work. Watching a presentation and reading a book helps to understand why something works. However, *real learning* happens after writing code, running it, and seeing it work. By the end of the book, you will understand how to use serverless. Here is an overview of the book.

Chapter 1: Introduction to AWS Serverless— We will lay the foundation of AWS serverless. We will start with an overview of cloud computing and how that enabled serverless computing. After that, we will learn about the different services used in serverless architectures and designs, including serverless computing, serverless storage, and other services.

Chapter 2: Overview of Serverless Applications— We will provide an overview of different types of serverless applications. There are many ways to use serverless. We will cover the most common ways to build applications using AWS serverless, which include website development, application programming interfaces, mobile app development, data processing, and notifications.

Chapter 3: Designing Serverless Architectures– We will provide an overview of how to design serverless architectures and design topics to consider. We will review monolithic and client-server architectures and how they can migrate to the AWS cloud and transition to AWS serverless. We will discuss using event-driven, microservices, and ad hoc architectures serverless.

Chapter 4: Launching a Website– We will provide hands-on exercises to create a serverless website. The hands-on exercises will cover setting up an AWS account, registering a new domain with Route 53, enabling and disabling S3 website hosting, creating a record in Route 53, requesting a certificate from AWS Certificate Manager (ACM), creating a CloudFront distribution, and create aliases for the distribution in Route 53.

Chapter 5: Creating an API– We will create and manage an API, send requests to a serverless function, and learn about its OpenAPI specification. The hands-on exercise will cover adding a new web page to use the API, invalidating the distribution to clear CloudFront cache, creating an API with API Gateway, integrating a Lambda function with the API, and exporting the API integration.

Chapter 6: Saving and Using Data– We will focus on saving and using data using object storage and serverless databases. The hands-on exercises will cover creating an S3 bucket for object storage, getting the object contents from a Lambda function, creating a DynamoDB database table, and getting the table data from a Lambda function. You will modify the function setting and security permissions multiple times in this chapter and throughout the rest of the book.

Chapter 7: Adding Authentication and Authorization– We will demonstrate how a website could add authentication and authorization to restrict certain pages to logged-in users. We will conduct hands-on exercises to add authentication and authorization using Amazon Cognito by creating a new Cognito user pool, customizing its hosted user interface, and updating the website. We will update the website by adding the login and logout buttons, the auth callback page, the logout page, and the account page.

Chapter 8: Processing Data Using Automation and Machine Learning– We will focus on how to use serverless computing to process data. The hands-on exercises will cover tracking authentication callbacks with an API and transforming that data using a Lambda function and an EventBridge schedule rule. We explore analyzing the transformed data with machine learning.

Chapter 9: Sending Notifications– We will send notifications using email, text messaging, and mobile push notifications. The hands-on exercises will cover configuring SES for emails, sending an email from a Lambda function, setting up an SNS topic to trigger a

Lambda function, sending a topic message from a Lambda function, setting up SNS for text messages, and sending a text message from a Lambda function.

Chapter 10: Additional Automation Topics— We discuss automation tasks that a serverless architecture can support. We will explore automation tasks that include setting up backups, analyzing text using Amazon Comprehend, performing text-to-speech using Amazon Polly, analyzing images using Amazon Rekognition, and auditing security settings using AWS Config.

Chapter 11: Architecture Best Practices— We will focus on architecture best practices when using serverless computing. We will walk through the AWS Well-Architected Framework and apply its Serverless Applications Lens. The Framework and Lens provide various best practices. We will review them to help you better architect and design your serverless applications.

Chapters 12: Next Steps— We will focus on the next steps you can explore to continue mastering AWS serverless. We only covered a few topics in-depth but explored them since it was necessary to gain exposure to them. Some of the following steps will include advanced topics such as attaching layers to Lambda functions, orchestrating function execution using AWS Step Functions, automatically creating resources using infrastructure as code with AWS CloudFormation, AWS Serverless Application Model, and AWS Cloud Development Kit, creating repeatable deployment processes using AWS CodeBuild and AWS CodePipeline, and testing serverless applications.

Code Bundle and Coloured Images

Please follow the link to download the *Code Bundle* and the *Coloured Images* of the book:

<https://rebrand.ly/j2hb61w>

The code bundle for the book is also hosted on GitHub at

<https://github.com/bpbpublications/Mastering-AWS-Serverless>.

In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **<https://github.com/bpbpublications>**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Did you know that BPB offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.bpbonline.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at :

business@bpbonline.com for more details.

At **www.bpbonline.com**, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at business@bpbonline.com with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit www.bpbonline.com. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit www.bpbonline.com.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



Table of Contents

| | |
|---|-----------|
| 1. Introduction to AWS Serverless | 1 |
| Introduction..... | 1 |
| Structure..... | 1 |
| Objectives..... | 2 |
| Introduction to cloud computing..... | 2 |
| Introduction to serverless computing | 8 |
| Introduction to serverless storage..... | 9 |
| Introduction to serverless services..... | 10 |
| Reviewing AWS serverless services..... | 11 |
| <i>Amazon Lambda</i> | 13 |
| <i>Amazon API Gateway</i> | 14 |
| <i>Amazon Simple Storage Service</i> | 16 |
| <i>Amazon CloudFront</i> | 17 |
| <i>Amazon DynamoDB</i> | 18 |
| <i>Amazon CloudWatch</i> | 19 |
| <i>Amazon EventBridge</i> | 20 |
| <i>Amazon Simple Email Service</i> | 20 |
| <i>Amazon Simple Notification Service</i> | 21 |
| <i>Amazon Cognito</i> | 23 |
| <i>AWS Identity and Access Management</i> | 24 |
| Conclusion..... | 25 |
| Points to remember | 26 |
| Multiple choice questions..... | 26 |
| 2. Overview of Serverless Applications | 29 |
| Introduction..... | 29 |
| Structure..... | 29 |
| Objectives..... | 30 |
| Introduction to serverless applications | 30 |
| Website development..... | 31 |
| <i>Amazon Route 53</i> | 36 |

| | |
|--|-----------|
| S3 | 37 |
| CloudFront..... | 38 |
| AWS Certificate Manager | 39 |
| AWS Web Application Firewall | 40 |
| Application Programming Interfaces | 42 |
| API Gateway..... | 43 |
| AWS AppSync..... | 44 |
| Mobile app development | 46 |
| API Gateway..... | 47 |
| AppSync | 47 |
| Data processing..... | 48 |
| S3 | 49 |
| DynamoDB..... | 49 |
| Notifications | 50 |
| Simple Notification Service | 50 |
| Amazon Simple Queue Service | 51 |
| CloudWatch | 53 |
| EventBridge | 53 |
| Simple Email Service | 53 |
| S3 event notification | 53 |
| DynamoDB Streams..... | 54 |
| Conclusion..... | 55 |
| Points to remember | 55 |
| Multiple choice questions..... | 56 |
| 3. Designing Serverless Architectures | 59 |
| Introduction..... | 59 |
| Structure..... | 59 |
| Objectives..... | 59 |
| Introduction to serverless architectures | 60 |
| Reviewing the monolithic architecture | 60 |
| Reviewing the client-server architecture | 62 |
| Migrating our architecture to the AWS cloud | 65 |
| Adding serverless to our architecture..... | 70 |
| Recognizing serverless storage..... | 70 |

| | |
|---|-----------|
| <i>Migrating to a serverless website</i> | 70 |
| <i>Adding a serverless API</i> | 72 |
| <i>Adding serverless computing</i> | 73 |
| <i>Adding or migrating to a serverless database</i> | 75 |
| <i>Additional serverless suggestions</i> | 76 |
| Principles in serverless architectures..... | 77 |
| <i>Principles in event-driven architectures</i> | 77 |
| <i>Principles in microservices architectures</i> | 82 |
| Design considerations and potential drawbacks..... | 85 |
| <i>Considerations for adding serverless to an existing architecture</i> | 85 |
| <i>Considerations for event-driven architectures</i> | 85 |
| <i>Considerations for microservices architectures</i> | 87 |
| <i>Considerations when implementing ad hoc</i> | 88 |
| Conclusion..... | 89 |
| Points to remember | 90 |
| Multiple choice questions..... | 90 |
| 4. Launching a Website | 93 |
| Introduction..... | 93 |
| Structure..... | 93 |
| Objectives..... | 94 |
| Overview of serverless websites | 94 |
| AWS account setup..... | 95 |
| <i>Security credentials</i> | 96 |
| <i>IAM users and Identity Center</i> | 97 |
| Register a new domain with Route 53..... | 97 |
| <i>Hands-on exercise 2: Registering a new domain with Route 53</i> | 98 |
| Enabling S3 website hosting | 101 |
| <i>Hands-on exercise 3: Enabling S3 website hosting</i> | 101 |
| Creating a record in Route 53 | 109 |
| <i>Hands-on exercise 4: Creating a record in Route 53</i> | 109 |
| Disabling S3 website hosting..... | 110 |
| <i>Hands-on exercise 5: Disabling S3 website hosting</i> | 111 |
| Requesting a certificate from AWS Certificate Manager..... | 113 |
| <i>Hands-on exercise 6: Requesting a certificate from ACM</i> | 113 |

| | |
|---|------------|
| Creating a CloudFront distribution | 117 |
| <i>Hands-on exercise 7: Creating a CloudFront distribution</i> | 117 |
| Creating aliases for the distribution in Route 53 | 121 |
| <i>Hands-on exercise 8: Creating aliases for the distribution in Route 53</i> | 122 |
| Conclusion..... | 124 |
| Points to remember | 124 |
| Multiple choice questions..... | 125 |
| 5. Creating an API..... | 127 |
| Introduction..... | 127 |
| Structure..... | 127 |
| Objectives..... | 127 |
| Overview of serverless APIs | 128 |
| Adding a new web page to use the API..... | 130 |
| <i>Hands-on exercise 1: Creating a new web page</i> | 130 |
| Invalidating the distribution to clear CDN cache..... | 134 |
| <i>Hands-on exercise 2: Invalidating the distribution</i> | 134 |
| Creating an API with API Gateway | 136 |
| <i>Hands-on exercise 3: Creating the API</i> | 136 |
| <i>Hands-on exercise 4: Defining resources and methods</i> | 138 |
| <i>Hands-on exercise 5: Configuring the mock integration</i> | 141 |
| <i>Hands-on exercise 6: Deploying the API</i> | 144 |
| <i>Hands-on exercise 7: Creating a custom domain</i> | 145 |
| <i>Hands-on exercise 8: Creating an alias in Route 53</i> | 148 |
| <i>Hands-on exercise 9: Updating the website home page</i> | 150 |
| Integration a Lambda function with the API | 151 |
| <i>Hands-on exercise 10: Creating the function</i> | 151 |
| Python example..... | 156 |
| <i>Hands-on exercise 11: Related resources</i> | 157 |
| <i>Hands-on exercise 12: Updating the API integration</i> | 160 |
| OpenAPI specification | 162 |
| <i>Hands-on exercise 13: Exporting the API integration</i> | 162 |
| Conclusion..... | 165 |
| Points to remember | 165 |

| | |
|---|------------|
| Multiple choice questions..... | 166 |
| 6. Saving and Using Data | 167 |
| Introduction..... | 167 |
| Structure..... | 167 |
| Objectives..... | 167 |
| Overview of saving and using data..... | 168 |
| Creating a new S3 bucket | 169 |
| <i>Hands-on exercise 1: Creating a new S3 bucket</i> | 169 |
| Getting the object contents from a Lambda function..... | 170 |
| <i>Creating an IAM policy</i> | 170 |
| <i>Creating an IAM role</i> | 175 |
| <i>Hands-on exercise 3: Creating an IAM role</i> | 175 |
| <i>Updating the Lambda function settings</i> | 177 |
| <i>Hands-on exercise 4: Updating the Lambda function settings</i> | 177 |
| <i>Updating the Lambda function code</i> | 178 |
| <i>Hands-on exercise 5: Updating the Lambda function code</i> | 178 |
| Creating a DynamoDB table | 180 |
| <i>Hands-on exercise 6: Creating a DynamoDB table</i> | 180 |
| Getting the table data from a Lambda function..... | 182 |
| <i>Updating the IAM policy</i> | 182 |
| <i>Hands-on exercise 7: Updating the IAM policy</i> | 183 |
| <i>Updating the Lambda function code</i> | 184 |
| <i>Hands-on exercise 8: Updating the Lambda function code</i> | 184 |
| Clean up..... | 185 |
| <i>Adding a robots.txt file</i> | 185 |
| <i>Hands-on exercise 9: Adding a robots.txt file</i> | 185 |
| <i>Cleaning up IAM</i> | 186 |
| <i>Hands-on exercise 10: Deleting the old IAM policy and role</i> | 186 |
| Conclusion..... | 186 |
| Points to remember | 187 |
| Multiple choice questions..... | 187 |
| 7. Adding Authentication and Authorization | 189 |
| Introduction..... | 189 |

| | |
|--|-----|
| Structure..... | 189 |
| Objectives..... | 189 |
| Overview of authentication and authorization..... | 190 |
| Creating a new Cognito user pool | 191 |
| <i>Creating a new Cognito user pool</i> | 191 |
| <i>Hands-on exercise 1: Creating a new Cognito user pool</i> | 192 |
| Adding an alias in Route 53 | 201 |
| <i>Hands-on exercise 2: Adding an alias in Route 53</i> | 201 |
| Customizing the Cognito user pool hosted UI..... | 202 |
| <i>Hands-on exercise 3: Customizing the user pool hosted UI</i> | 203 |
| Updating the website..... | 205 |
| Adding the login and log out buttons..... | 205 |
| <i>Hands-on exercise 4: Adding the login and log out buttons</i> | 206 |
| Adding the auth callback page | 210 |
| <i>Hands-on exercise 5: Adding the auth callback page</i> | 210 |
| Creating the logout page..... | 212 |
| <i>Hands-on exercise 6: Creating the logout page</i> | 212 |
| Creating the account page | 213 |
| <i>Hands-on exercise 7: Creating the account page</i> | 213 |
| Uploading the files to S3 bucket | 217 |
| <i>Hands-on exercise 8: Uploading the files to S3 bucket</i> | 217 |
| Invalidating the CloudFront distribution cache | 217 |
| <i>Hands-on exercise 9: Invalidating the CloudFront distribution cache</i> | 218 |
| Testing the authentication | 218 |
| Sign up using the Cognito user pool | 218 |
| Sign out from the Cognito user pool..... | 220 |
| Sign in using the Cognito user pool | 221 |
| <i>Hands-on exercise 12: Signing in using the Cognito user pool</i> | 221 |
| Adding an authorized API endpoint..... | 221 |
| Creating an authorizer..... | 221 |
| <i>Hands-on exercise 13: Creating an authorizer</i> | 221 |
| Creating an IAM role | 223 |
| <i>Hands-on exercise 14: Creating an IAM role</i> | 223 |
| Creating a function..... | 224 |

| | |
|---|------------|
| <i>Hands-on exercise 15: Creating a function</i> | 224 |
| Updating the log group settings | 226 |
| <i>Hands-on exercise 16: Updating the log group settings</i> | 226 |
| Creating an API endpoint | 226 |
| <i>Hands-on exercise 17: Creating an API endpoint</i> | 226 |
| Creating an account information page | 228 |
| <i>Hands-on exercise 18: Creating an account information page</i> | 228 |
| Uploading the files to S3 bucket | 231 |
| Invalidating the CloudFront distribution cache | 232 |
| <i>Hands-on exercise 20: Invalidating the CloudFront distribution cache</i> | 232 |
| Testing the new function | 232 |
| <i>Hands-on exercise 21: Testing the new function</i> | 232 |
| Conclusion | 233 |
| Points to remember | 233 |
| Multiple choice questions | 234 |
| 8. Processing Data Using Automation and Machine Learning | 235 |
| Introduction | 235 |
| Structure | 235 |
| Objectives | 235 |
| Overview of data processing | 236 |
| Tracking authentication callbacks | 237 |
| Create a DynamoDB table | 237 |
| <i>Hands-on exercise 1: Creating a DynamoDB table</i> | 237 |
| Creating an IAM policy | 237 |
| <i>Hands-on exercise 2: Creating an IAM policy</i> | 237 |
| Creating an IAM role | 238 |
| <i>Hands-on exercise 3: Creating an IAM role</i> | 238 |
| Creating an API endpoint | 239 |
| <i>Hands-on exercise 4: Creating an API endpoint</i> | 239 |
| Updating the auth callback web page | 243 |
| <i>Hands-on exercise 5: Updating the auth callback web page</i> | 243 |
| Uploading the files to S3 bucket | 245 |
| <i>Hands-on exercise 6: Uploading the files to S3 bucket</i> | 245 |
| Invalidating the CloudFront distribution cache | 245 |

| | |
|--|------------|
| <i>Hands-on exercise 7: Invalidating the CloudFront distribution cache</i> | 245 |
| <i>Testing the new API endpoint</i> | 246 |
| <i>Hands-on exercise 8: Testing the new API endpoint</i> | 246 |
| Transforming data | 246 |
| <i>Creating SSM parameters</i> | 246 |
| <i>Hands-on exercise 9: Creating SSM parameters</i> | 247 |
| <i>Creating an IAM policy</i> | 248 |
| <i>Hands-on exercise 10: Creating an IAM policy</i> | 249 |
| <i>Creating an IAM role</i> | 250 |
| <i>Hands-on exercise 11: Creating an IAM role</i> | 250 |
| <i>Creating a Lambda function</i> | 251 |
| <i>Hands-on exercise 12: Creating a Lambda function</i> | 251 |
| <i>Updating the log group settings</i> | 256 |
| <i>Hands-on exercise 13: Updating the log group settings</i> | 256 |
| <i>Creating an EventBridge schedule rule</i> | 257 |
| <i>Hands-on exercise 14: Creating an EventBridge schedule rule</i> | 257 |
| <i>Checking the rule is working</i> | 261 |
| Analyzing data using machine learning | 261 |
| <i>Creating IAM policies</i> | 262 |
| <i>Example 1: Creating IAM policies</i> | 262 |
| <i>Creating IAM roles</i> | 265 |
| <i>Example 2: Creating IAM roles</i> | 266 |
| <i>Creating Lambda functions</i> | 267 |
| <i>Example 3: Creating Lambda functions</i> | 267 |
| <i>Creating S3 event triggers</i> | 274 |
| <i>Example 4: Creating S3 event triggers</i> | 274 |
| Conclusion..... | 275 |
| Points to remember | 275 |
| Multiple choice questions..... | 275 |
| 9. Sending Notifications | 277 |
| Introduction..... | 277 |
| Structure..... | 277 |
| Objectives..... | 278 |
| Overview of notifications | 278 |

| | |
|--|-----|
| Setting SES for emails | 278 |
| Set up SES | 278 |
| Send a test email message | 282 |
| Hands-on exercise 2: Sending a test message..... | 282 |
| Set up the DNS records | 283 |
| Hands-on exercise 3: Setting up the DNS records..... | 283 |
| Sending an email from a Lambda function | 284 |
| Create an IAM policy | 284 |
| Hands-on exercise 4: Creating an IAM policy..... | 284 |
| Update an IAM role..... | 285 |
| Hands-on exercise 5: Updating an IAM role | 285 |
| Update a Lambda function..... | 286 |
| Hands-on exercise 6: Updating a Lambda function..... | 286 |
| Test on the website | 289 |
| Hands-on exercise 7: Testing on the website | 289 |
| Using an SNS topic to trigger a Lambda function..... | 289 |
| Create an SNS topic..... | 289 |
| Hands-on exercise 8: Creating an SNS topic..... | 289 |
| Create an IAM policy | 290 |
| Hands-on exercise 9: Creating an IAM policy..... | 290 |
| Create an IAM role | 291 |
| Hands-on exercise 10: Creating an IAM role | 291 |
| Create a Lambda function..... | 292 |
| Hands-on exercise 11: Creating a Lambda function | 292 |
| Update the log group settings..... | 295 |
| Hands-on exercise 12: Updating the log group settings..... | 295 |
| Sending a topic message from a Lambda function..... | 295 |
| Updating the IAM policy | 295 |
| Hands-on exercise 13: Updating the IAM policy | 295 |
| Updating the Lambda function..... | 296 |
| Hands-on exercise 14: Updating the Lambda function..... | 296 |
| Testing on the website..... | 298 |
| Hands-on exercise 15: Testing on the website | 298 |
| Setting up SNS for text messages..... | 299 |

| | |
|--|------------|
| <i>Exiting the SMS sandbox</i> | 299 |
| <i>Example 1: Exiting the SMS sandbox</i> | 299 |
| <i>Create an SNS topic</i> | 299 |
| <i>Example 2: Creating an SNS topic</i> | 299 |
| <i>Request an originating number</i> | 300 |
| <i>Example 3: Requesting an originating number</i> | 300 |
| <i>Adding a Sandbox destination phone number</i> | 300 |
| <i>Example 4: Adding a Sandbox destination phone number</i> | 300 |
| <i>Sending a text message from a Lambda function</i> | 301 |
| <i>Update the Cognito user pool</i> | 301 |
| <i>Example 5: Updating the Cognito user pool</i> | 301 |
| <i>Update the website</i> | 301 |
| <i>Independent exercise 1: Updating the website</i> | 301 |
| <i>Updating an existing Lambda function</i> | 302 |
| <i>Independent exercise 2: Updating an existing Lambda function</i> | 302 |
| <i>Creating new API endpoint and related resources</i> | 302 |
| <i>Independent exercise 3: Creating a new Lambda function and related resources</i> | 302 |
| <i>Setting up SNS for push notifications</i> | 303 |
| <i>Conclusion</i> | 303 |
| <i>Points to remember</i> | 304 |
| <i>Multiple choice questions</i> | 304 |
| 10. Additional Automation Topics | 305 |
| <i>Introduction</i> | 305 |
| <i>Structure</i> | 305 |
| <i>Objectives</i> | 306 |
| <i>Overview of automation</i> | 306 |
| <i>Backups</i> | 306 |
| <i>Text analysis</i> | 307 |
| <i>Text-to-speech</i> | 307 |
| <i>Image resizing</i> | 307 |
| <i>Image analysis</i> | 308 |
| <i>Security</i> | 308 |
| <i>Setting up backups</i> | 308 |
| <i>Enable versioning in the S3 buckets</i> | 309 |

| | |
|--|-----|
| <i>Hands-on exercise 1: Enabling versioning in the S3 buckets</i> | 309 |
| <i>Create an AWS Backup plan</i> | 309 |
| <i>Hands-on exercise 2: Creating an AWS Backup plan</i> | 309 |
| <i>Set up DynamoDB point-in-time recovery</i> | 312 |
| <i>Hands-on exercise 3: Setting up DynamoDB point in time recovery</i> | 312 |
| <i>Set up S3 bucket replication</i> | 313 |
| <i>Hands-on exercise 4: Setting up S3 bucket replication</i> | 313 |
| <i>Use a Lambda function to backup DynamoDB data</i> | 316 |
| <i>Independent exercise 1: Using a Lambda function to backup DynamoDB data</i> | 316 |
| <i>Analyzing text using Amazon Comprehend</i> | 317 |
| <i>Using Amazon Comprehend</i> | 317 |
| <i>Hands-on exercise 5: Using Amazon Comprehend</i> | 317 |
| <i>Create a new DynamoDB table</i> | 318 |
| <i>Hands-on exercise 6: Creating a new DynamoDB table</i> | 318 |
| <i>Update the website</i> | 319 |
| <i>Independent exercise 2: Updating the website</i> | 319 |
| <i>Create new API endpoint and related resources</i> | 319 |
| <i>Independent exercise 3: Creating an API endpoint and related resources</i> | 319 |
| <i>Use a Lambda function to analyze the comments</i> | 320 |
| <i>Independent exercise 4: Using a Lambda function to analyze the comments</i> | 320 |
| <i>Performing text-to-speech using Amazon Polly</i> | 321 |
| <i>Using Amazon Polly</i> | 321 |
| <i>Hands-on exercise 8: Using Amazon Polly</i> | 321 |
| <i>Create a new DynamoDB table</i> | 323 |
| <i>Hands-on exercise 9: Creating a new DynamoDB table</i> | 323 |
| <i>Update the website</i> | 324 |
| <i>Independent exercise 5: Updating the website</i> | 324 |
| <i>Create new API endpoint and related resources</i> | 324 |
| <i>Independent exercise 6: Creating API endpoints and related resources</i> | 324 |
| <i>Use a Lambda function to create the audio files</i> | 325 |
| <i>Independent exercise 7: Using a Lambda function to create the audio files</i> | 325 |
| <i>Analyzing images using Amazon Rekognition</i> | 326 |
| <i>Using Amazon Rekognition</i> | 326 |
| <i>Hands-on exercise 10: Using Amazon Rekognition</i> | 326 |

| | |
|--|------------|
| Update the website..... | 327 |
| Independent exercise 8: Updating the website | 328 |
| Create new API endpoint and related resources..... | 328 |
| Independent exercise 9: Creating an API endpoint and related resources..... | 328 |
| Use a Lambda function to analyze the images..... | 328 |
| Independent exercise 10: Using a Lambda function to analyze the images | 329 |
| Auditing security settings using AWS Config | 330 |
| Set up AWS Config..... | 330 |
| Example 1: Setting up AWS Config..... | 330 |
| Deploying conformance packs | 333 |
| Example 2: Deploying conformance packs..... | 333 |
| Conclusion..... | 335 |
| Points to remember | 335 |
| Multiple choice questions..... | 336 |
| 11. Architecture Best Practices | 337 |
| Introduction..... | 337 |
| Structure..... | 337 |
| Objectives..... | 337 |
| The AWS Well-Architected Framework | 337 |
| Using the Serverless Applications Lens | 338 |
| Design principles | 338 |
| Speedy, simple, singular..... | 339 |
| Think concurrent requests, not total requests | 340 |
| Share nothing | 340 |
| Assume no hardware affinity | 341 |
| Orchestrate your application with state machines, not functions | 341 |
| Using events to trigger transactions..... | 342 |
| Design for failures and duplicates..... | 342 |
| Pillar 1: Operational excellence | 343 |
| Performing operations as code..... | 343 |
| Making frequent, small, reversible changes | 344 |
| Refining operations procedures frequently..... | 345 |
| Anticipating failure..... | 346 |
| Learning from all operational failures..... | 346 |

| | |
|--|------------|
| Using managed services | 347 |
| Implementing observability for actionable insights | 347 |
| Organization | 348 |
| Prepare | 348 |
| Operate | 349 |
| Evolve | 350 |
| Useful services and tools | 351 |
| Pillar 2: Security | 351 |
| Implementing a strong identity foundation | 352 |
| Maintaining traceability | 353 |
| Applying security at all layers | 353 |
| Automating security best practices | 354 |
| Protecting data in transit and at rest | 354 |
| Keeping people away from data | 354 |
| Preparing for security events | 354 |
| Identity and access management | 355 |
| Detective controls or detection | 356 |
| Infrastructure protection | 357 |
| Data protection | 357 |
| Incident response | 358 |
| Useful services and tools | 359 |
| Pillar 3: Reliability | 359 |
| Automatically recover from failure | 360 |
| Testing recovery procedures | 360 |
| Scaling horizontally | 360 |
| Stop guessing capacity | 360 |
| Managing change in automation | 361 |
| Foundations | 361 |
| Managing changes | 362 |
| Failure management | 362 |
| Useful services | 363 |
| Pillar 4: Performance efficiency | 364 |
| Democratizing advanced technologies | 364 |
| Going global in minutes | 365 |

| | |
|---|------------|
| Using serverless architectures..... | 365 |
| Experimenting more often..... | 365 |
| Considering mechanical sympathy..... | 365 |
| Architecture selection..... | 365 |
| Optimizing..... | 366 |
| Review, monitoring, and tradeoffs..... | 366 |
| Useful services..... | 367 |
| Pillar 5: Cost optimization..... | 367 |
| Implementing cloud financial management..... | 367 |
| Adopting a consumption model..... | 368 |
| Measuring overall efficiency..... | 368 |
| Stop spending money on undifferentiated heavy lifting..... | 368 |
| Analyzing and attributing expenditure..... | 368 |
| Cost-effective resources..... | 369 |
| Matching supply and demand..... | 369 |
| Expenditure and usage awareness..... | 369 |
| Optimizing over time..... | 369 |
| Useful services..... | 370 |
| Pillar 6: Sustainability..... | 370 |
| Understanding your impact..... | 371 |
| Establishing sustainability goals..... | 371 |
| Maximizing utilization..... | 371 |
| Adopting new hardware and software offerings..... | 371 |
| Using managed services..... | 372 |
| Reducing the downstream impact of your cloud workloads..... | 372 |
| Region selection..... | 372 |
| Alignment to demand..... | 372 |
| Data..... | 373 |
| Conclusion..... | 373 |
| Points to remember..... | 373 |
| Multiple choice questions..... | 374 |
| 12. Next Steps..... | 375 |
| Introduction..... | 375 |
| Structure..... | 375 |

| | |
|--|-----|
| Objectives..... | 376 |
| AWS Lambda layers..... | 376 |
| <i>Packaging the layer</i> | 376 |
| <i>Example 1: Packaging the layer</i> | 376 |
| <i>Creating a layer</i> | 376 |
| <i>Example 2: Creating a layer</i> | 377 |
| <i>Using the layer</i> | 377 |
| <i>Example 3: Using the layer</i> | 378 |
| AWS step functions | 380 |
| <i>Creating an example state machine</i> | 380 |
| <i>Example 4: Creating an example state machine</i> | 380 |
| <i>Integrating a Step Function with API Gateway</i> | 383 |
| <i>Example 5: Integrating a step function with API Gateway</i> | 383 |
| Infrastructure as Code | 385 |
| AWS CloudFormation | 385 |
| <i>Creating a new IAM policy and role</i> | 385 |
| <i>Example 6: Creating a new IAM policy and role</i> | 385 |
| <i>Creating a stack</i> | 386 |
| <i>Example 7: Creating a stack</i> | 386 |
| <i>Updating a stack</i> | 389 |
| <i>Example 8: Updating a stack</i> | 389 |
| AWS serverless application model | 392 |
| <i>Creating an IAM group and IAM user</i> | 393 |
| <i>Example 9: Creating an IAM group and IAM user</i> | 393 |
| <i>Installing the SAM CLI</i> | 396 |
| <i>Example 10: Installing the SAM CLI</i> | 396 |
| <i>Creating an application</i> | 396 |
| <i>Example 11: Creating an application</i> | 396 |
| <i>Building the app</i> | 397 |
| <i>Example 12: Build the app</i> | 397 |
| <i>Deploying the app</i> | 397 |
| <i>Example 13: Deploying the app</i> | 397 |
| AWS Cloud Development Kit..... | 398 |
| <i>Updating the IAM group</i> | 398 |

| | |
|---|----------------|
| <i>Example 14: Updating the IAM group</i> | 398 |
| Installing CDK | 399 |
| <i>Example 15: Installing CDK</i> | 399 |
| Creating an application | 399 |
| <i>Example 16: Creating an application</i> | 399 |
| Bootstrapping CDK..... | 399 |
| <i>Example 17: Bootstrapping CDK</i> | 399 |
| Deploying an application..... | 400 |
| <i>Example 18: Deploying an application</i> | 400 |
| Updating the application | 400 |
| <i>Example 19: Updating an application</i> | 400 |
| AWS CodeBuild | 402 |
| AWS CodePipeline | 403 |
| Testing serverless applications | 403 |
| Next steps | 404 |
| Conclusion..... | 405 |
| Points to remember | 405 |
| Multiple choice questions..... | 405 |
| Index | 407-414 |

CHAPTER 1

Introduction to AWS Serverless

Introduction

This book aims to help you master serverless on **Amazon Web Services (AWS)**.¹ As a master, you will have the skill, knowledge, and proficiency to build serverless applications in the AWS cloud. With the master knowledge, you will be on your road to becoming a serverless expert.

This chapter will lay the foundation of AWS serverless. We will start with an overview of cloud computing and how that enabled serverless computing. After that, we will learn about the different services used in serverless architectures and designs.

Structure

We will cover the following topics in this chapter:

- Introduction to cloud computing
- Introduction to serverless computing
- Introduction to serverless storage
- Introduction to serverless services
- Reviewing AWS serverless services

¹ Amazon, Amazon Web Services and AWS are registered trademarks of Amazon Web Services, Inc.

Objectives

At the end of the chapter, you will understand AWS's services. You can apply that knowledge when we discuss a serverless application built on AWS.

Introduction to cloud computing

In the early days of computing, organizations hosted equipment in buildings they managed. An organization would buy servers, data storage, router, switches, and racks. They dedicated a room (or sometimes an entire building) to having multiple equipment racks. Cables interconnect connected the equipment to switches and routers that provided network connectivity. Some networks provided local networks that interconnected a select number of devices. Other networks provided connectivity to the organization's Intranet so their staff could connect to their equipment and network services. Some networks provided Internet connectivity where internet-connected devices could connect. These network configurations enabled the *on-premises* hosting and computing in the organization's data warehouse. See *Figure 1.1* for an example of an organization's on-premises infrastructure:

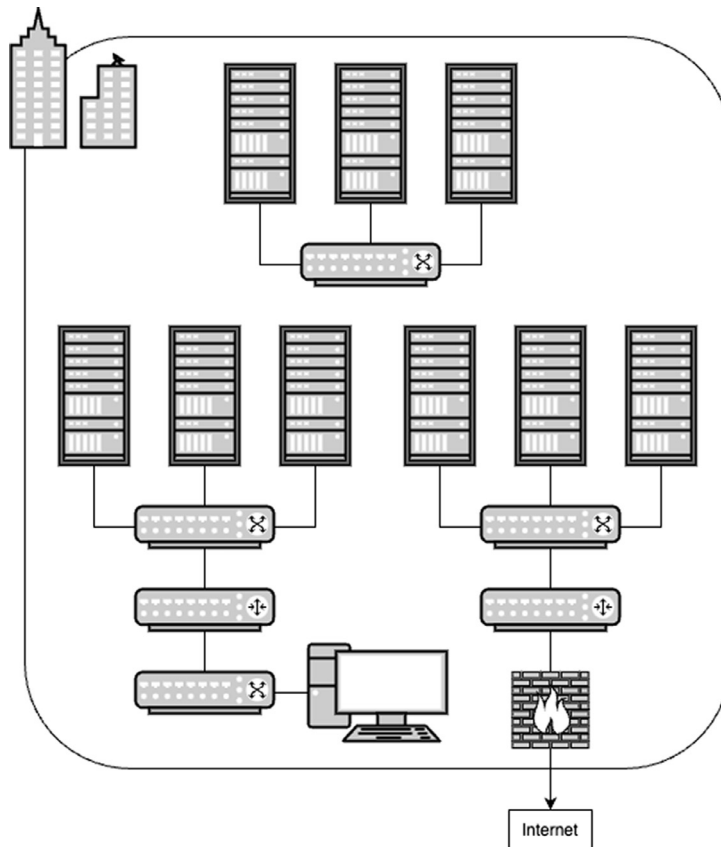


Figure 1.1: An example infrastructure diagram showing on-premises hosting and computing

As Internet speeds and connectivity improved, service providers created data warehouses to provide hosting and computing as a service. For example, an organization would pay a service fee for a web server to be made available. An organization could perform a make-or-buy analysis. It would host the web server when it made sense from business, financial, and legal standpoints. Otherwise, they would *rent* a web server to save on costs (for example, operational, labor costs, and maintenance) and possibly offload some legal, compliance, and security implications to the service provider.

The organization's infrastructure diagrams became simpler as it rented hosting from service providers. Some diagrams started showing a cloud to abstract all the servers and services that were no longer on-premises but provided by a service provider. See *Figure 1.2* for an example of an organization's infrastructure that uses a service provider. In this example figure, a computer that accessed servers using internal networking now accesses them from the service provider, which is depicted as a cloud:

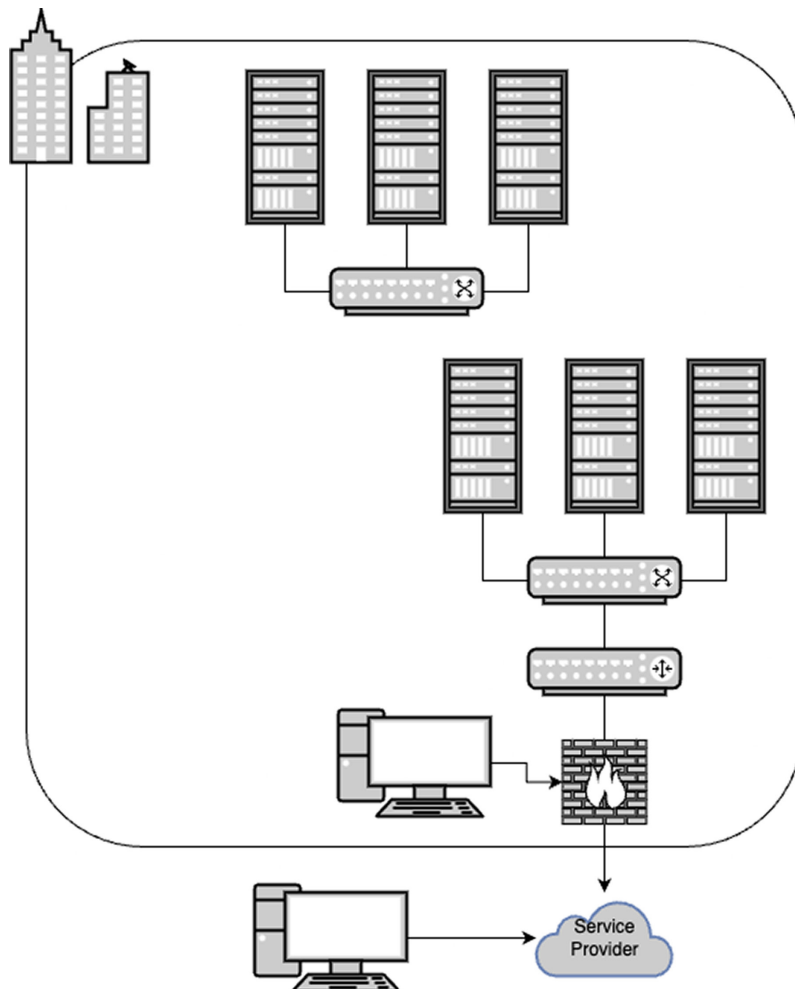


Figure 1.2: An example infrastructure diagram that shows a service provider as a cloud

The term *cloud computing* was eventually accepted. We will refer to cloud computing as a set of services and resources provided by a third-party service provider that is accessed via the Internet and are not physically hosted inside an organization's building. This working definition is essential because some serverless solutions allow an organization to perform serverless computing using on-premises hardware. We are focusing on AWS serverless, which is a set of serverless services that AWS, a cloud computing provider (or cloud provider for short), offers.

Cloud providers became popular because their prices became more cost-effective, and service offerings were more powerful. These improvements were fueled by more affordable equipment, higher capacity data storage, faster Internet speeds, and higher bandwidth. Furthermore, companies created new technologies that allowed multiple servers to run within one physical server. These innovations enabled new cloud providers to emerge.

AWS launched its services to the public in 2006 by offering Internet-based services that customers could configure using a web-based application.² A customer could sign up for an AWS account, use the **AWS web-based console** (**AWS console** for short), create a server, configure the networking, set up the **Domain Name System (DNS)**, and get an **Internet Protocol (IP)** address. A customer could have a live web server within minutes, for example. Since its launch, AWS has provided various cloud-based products: cloud computing (that is, servers), cloud storage (that is, file server equivalents), databases, networking, and many more.

As previously mentioned, the virtual server technologies enabled AWS and other service providers to provide cloud-based services to their customers. AWS created data warehouses (or data centers) in various geographical regions within a country and around the world. They configured their networking and equipment to allow multiple customers to create servers and save their data, so they were *logically* separated. No customer would be able to access another customer's resources. They were logically separated because the software would separate the data and prevent authorized access even though they might be physically hosted on the same equipment. See *Figure 1.3* for an example of logical separation:

² "Overview of Amazon Web Services." Amazon Web Services, Inc. <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html>

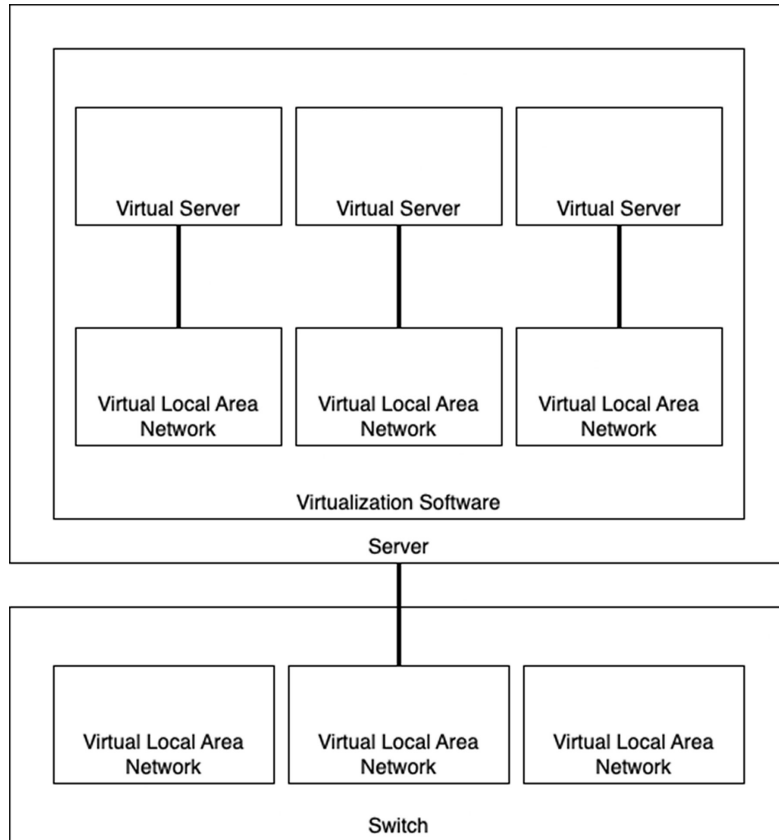


Figure 1.3: An example of logical separation that a cloud provider might implement

With virtualization technologies, AWS can support several customers with the equipment in their data centers. Furthermore, it allows customers to choose:

- The number, speed and type of **central processing unit (CPU)** processors
- The **random access memory (RAM)** size
- The size of the disk space
- The number, speed and type of graphical processing units
- Other specifications to meet their needs.

How the CPU, RAM, and disk space are provisioned on the physical hardware is done for us.

AWS follows the shared responsibility model to ensure our data is stored and accessed safely.³ AWS is committed to the *Security of the Cloud*, and they believe it is the customer's responsibility for the *Security in the Cloud*. This means that AWS will ensure the cloud

3 "Shared Responsibility Model." Amazon Web Services. <https://aws.amazon.com/compliance/shared-responsibility-model/>

services and infrastructure is secure. Any hardware, software, data storage, networking, building, staff, operations, and so on, involved in providing cloud services is secured and maintained securely. This also means that how the customer uses the AWS cloud is their responsibility to secure. Like with any software and service, there are best practices for using them. The customer must know how to and be willing to implement them.

Secure virtualization technologies advanced to another level of virtualization. Initially, the technology provided virtual networks, servers, and disk space. This allowed for the introduction of containers. A container is like a virtual server but extremely lightweight. The container's operating system has a minimum set of libraries and dependencies and no graphical display. The container has no resources but shares them with the server running the container orchestration software. A virtual server needs a virtual CPU, RAM, disk drive, network interfaces, and other resources, whereas a container *borrow*s the resources from the physical server when it needs them.

For example, a physical server may have 4 CPUs, 16 **gigabytes (GB)** of RAM, and one **terabyte (TB)** of disk space. That server can host four virtual servers that each have 1 CPU, 4 GB of RAM, and 250 MB of disk space. We cannot add a fifth virtual server because we are out of resources. Even if the virtual server only uses 2 GB of RAM, it was provisioned with 4 GB of RAM, and they are unusable by another virtual server. See *Figure 1.4* for a drawing illustrating the four virtual servers:

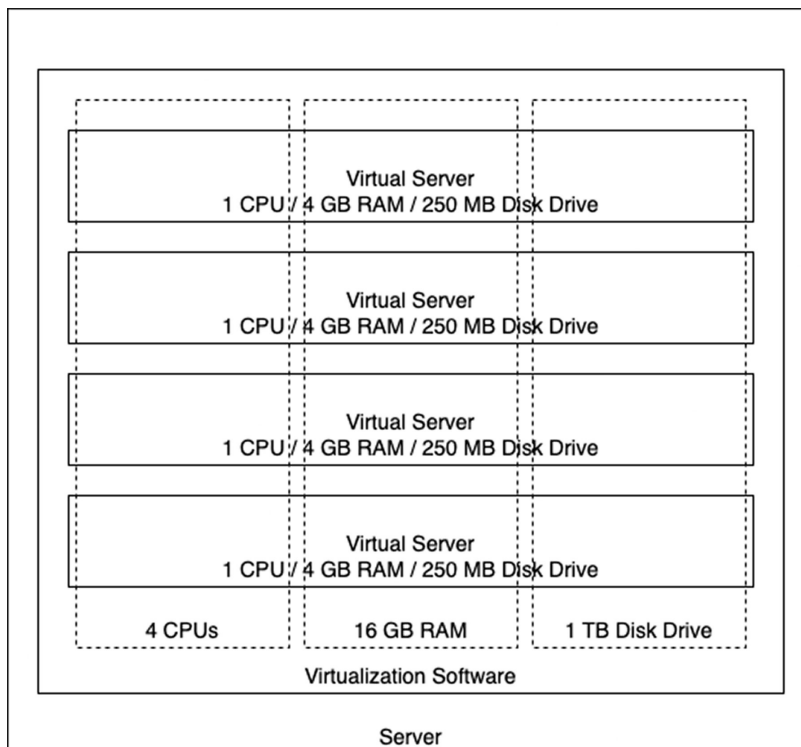


Figure 1.4: An example of four virtual servers running on a physical server

Containers use resources differently. Suppose we create four containers on the physical server. The containers only use 0.5 CPUs and 1 GB of RAM. The containers will use an *image* that has the operating system, and they all share the same image. Any runtime files are stored in a temporary container disk drive that grows as needed and is deleted when the container is stopped. The container can use a volume to store files on the drive permanently. These four containers share resources and use up a total of 2 CPUs, 1 GB of RAM, the size of the container image, and the size of the new files. As long as there are still available resources on the physical server, we can continue creating containers. See *Figure 1.5* for a drawing illustrating the containers:

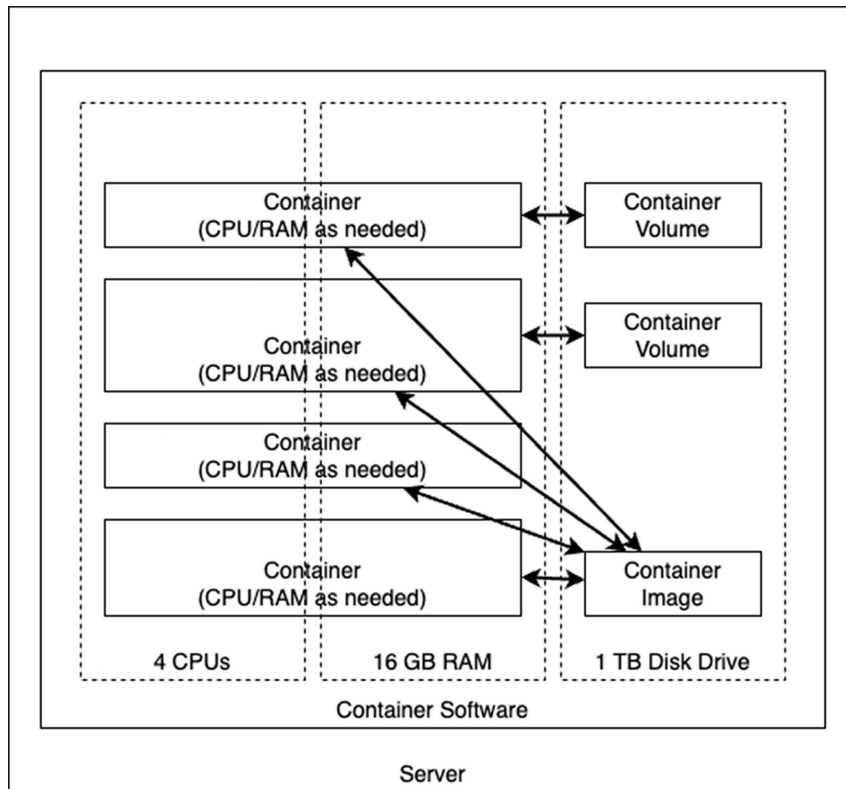


Figure 1.5: An example of containers running on a physical server

Cloud providers were able to improve their service offerings because they maximized the utilization of the physical server resources. Furthermore, it allowed organizations to design their applications to use smaller *servers* that used fewer resources and could be less expensive. The ability to potentially run an application in a single container resulted in the next level of innovation: serverless computing.

Introduction to serverless computing

In 2014, AWS introduced AWS **Lambda**, which is their serverless computing service.⁴ Lambda allows us to upload and run code as a function without configuring a server or a container. The service uses containers to run the code. When we upload the code to the Lambda service, it is stored as a compressed file inside AWS. When the Lambda function needs to run the code, the service will create a new container, create a volume using the compressed file, execute the code, and delete the container after a period of inactivity. How the Lambda service manages that process is beyond our control. We specify the parameters we would like our Lambda function to have:

- Amount of RAM
- Maximum execution time
- Trigger(s) that starts the code execution
- Function code compressed file
- Security permissions

There are other parameters that we will learn in later chapters. The Lambda service uses the Lambda function parameters to create a container. The Lambda service is considered serverless instead of a container service for the following reasons:

- The purpose is to run code
- We cannot directly manage the containers
- The containers are short-lived
- AWS may choose to move away from containers as the underlying technology

Other serverless providers do not use containers for serverless computing, and how AWS operates the Lambda service is out of our control. A container service assumes the containers are running for long periods. Containers are often used as small servers, and the container configuration might impact how the application runs. As a result, defining the Lambda service as a different service offering makes sense.

Moving to a serverless offering provided many benefits to AWS and its customers. In the previous examples, a physical server could host four virtual servers and more than four containers. The physical server can host many more containers because they are created when they are needed and deleted when they are not. Using the previous example, we can potentially host 100s over Lambda function configurations because the underlying containers only exist when they are needed. The likelihood that all the Lambda functions need to exist simultaneously is low. This is a benefit for AWS customers because they can define as many Lambda functions as needed and only pay when they are used, which can be significantly less than provisioning a server or container.

⁴ “AWS Lambda Releases.” Amazon Web Services. <https://docs.aws.amazon.com/lambda/latest/dg/lambda-releases.html>

The serverless computing model provides the following benefits:

- On-demand usage
- Elasticity
- Scalability
- Rapid development
- Reduced infrastructure
- Reduced maintenance
- Lower costs
- Smaller attack surfaces

The success of serverless computing resulted in cloud providers creating new serverless services and updating existing services to work with serverless applications.

Introduction to serverless storage

In 2006, AWS introduced Amazon **Simple Storage Service (S3)**, which is their object storage service.⁵ S3 was one of AWS's early service offerings.⁶ S3 is an object storage where we can store any type of data as an object associated with a key name. We store data in buckets which allows us to create logical groups for our objects. We cannot access objects like files in an operating system or file share server. We access the S3 data using the following AWS capabilities:

- The web-based console
- The **application programming interface (API)**
- The **command line interface (CLI)**
- The **software development kit (SDK)**

AWS and many developers consider S3 as part of the AWS serverless offering. Much like how we do not need to worry about how a Lambda function is created and managed, AWS will manage how we save data to S3. With S3, we can:

- Save an object as big as 5 TB.⁷
- Have no limit on how many objects we can save to a bucket.⁸
- Create up to 1000 buckets

5 "Document history." Amazon Web Services. <https://docs.aws.amazon.com/AmazonS3/latest/userguide/WhatsNew.html>

6 "What is Amazon S3?" Amazon Simple Storage Service User Guide. Amazon Web Services. <https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html>

7 "Amazon Simple Storage Service endpoints and quotas." Reference Guide. Amazon Web Services. <https://docs.aws.amazon.com/general/latest/gr/s3.html>

8 "Bucket restrictions and limitations." Amazon Simple Storage Service User Guide. Amazon Web Services. <https://docs.aws.amazon.com/AmazonS3/latest/userguide/BucketRestrictions.html>

S3 provides flexibility and scalability in storage, similarly to how Lambda provides that for computing.

Since then, AWS has introduced serverless databases or serverless versions of non-serverless databases. Object storage is beneficial for storing large amounts of data but can be inefficient for querying and searching. Services exist that enable querying and searching for object storage, but that can result in a more complex design and increased costs. Rather, we can use serverless databases. A popular serverless database is Amazon DynamoDB.

DynamoDB is a key-value or **Non-Structured Query Language (NoSQL)** database.⁹ Unlike other SQL or NoSQL database services, DynamoDB does not require us to have a dedicated server. AWS fully manages DynamoDB, using multiple servers and disk drives to handle the traffic and store our database table. When we use DynamoDB, we can focus on tables and their data without worrying about configuring and managing the database servers. We work with the tables and data using the AWS console, API, CLI, or SDK.

DynamoDB allows us to configure how much data we can read and write per second (that is, read and write capacity). There are two capacity modes:

- **Provisioned capacity:** We can specify how much *read capacity* and *write capacity* we want as available for our application to use. This provides us with predictability in performance and costs. This mode is best for when we have a consistent number of reads and writes to our table.
- **On-demand capacity:** We can use a pay-per-use mode where we do not need to specify how much capacity we need. This gives us the most flexibility, but we pay higher prices per read and write. This mode is best when we have unpredictable or spiky traffic or when our data sits idle most of the time.

Furthermore, DynamoDB has no limits to the amount of data a table can store. This unlimited storage provides the benefits of object storage with the added features of database operations.¹⁰

Introduction to serverless services

Data storage and application interfaces were the next logical areas to introduce a serverless solution.

9 “What is DynamoDB?” Amazon DynamoDB Developer Guide. Amazon Web Services. <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html>

10 “Service, account, and table quotas in Amazon DynamoDB.” Amazon DynamoDB Developer Guide. Amazon Web Services. <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ServiceQuotas.html>

Serverless data storage services were designed to have the following key benefits:

- Elastic capacities
- Scalable resources

Many serverless data storage services provide *virtually unlimited* or very high data storage. These services do not require provisioning for a specific amount of data storage. Instead, data is stored as needed.

There are different serverless data storage service options which include:

- Object storage
- File systems
- Relational databases
- Non-relational / key-value / document databases

We will explore these services in more detail when we explore the AWS serverless services.

The application interface services can be serverless or not. These services are often categorized as serverless because they work well with serverless computing and provide on-demand, elastic, and scalable characteristics. These application interfaces include:

- API gateways, management, and services
- Messaging and notifications
- Workflow and business logic orchestration
- Event buses and management

We will explore these services in more detail when we explore the AWS serverless services.

Reviewing AWS serverless services

AWS has a web page dedicated to its serverless capabilities.¹¹ See *Figure 1.6*, which has a screen capture of a section of the AWS serverless web page showing some serverless services:

¹¹ “Serverless on AWS.” Amazon Web Services. <https://aws.amazon.com/serverless/>

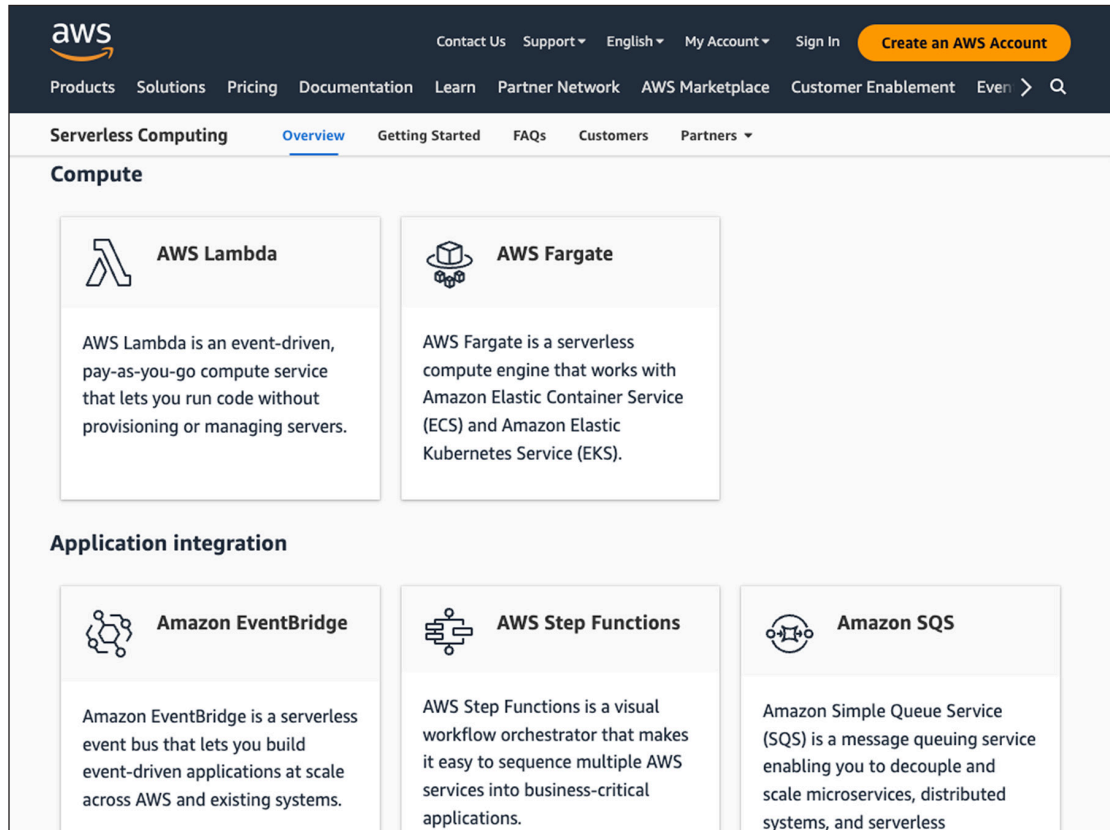


Figure 1.6: A screen capture of the AWS serverless computing web page

AWS provides the following serverless services and services that work well with serverless computing:

- Amazon Aurora Serverless
- Amazon API Gateway
- Amazon CloudFront
- Amazon CloudWatch
- Amazon Cognito
- Amazon DynamoDB (Database)
- Amazon EventBridge
- Amazon **Elastic File System (EFS)**

- Amazon Lambda
- Amazon Neptune Serverless
- Amazon OpenSearch Serverless
- Amazon RedShift Serverless
- Amazon **Relational Database Service (RDS) Proxy**
- Amazon **Simple Email Service (SES)**
- Amazon **Simple Notification Service (SNS)**
- Amazon **Simple Storage Service (S3)**
- Amazon **Simple Queue Service (SQS)**
- AWS AppSync
- AWS Fargate
- AWS Step Functions

These services are well documented in the AWS website and documentation, but we will review some of these services that we will use in this book.

Amazon Lambda

We reviewed Lambda earlier in the chapter. (See the Amazon Lambda icon in *Figure 1.7*.)



Figure 1.7: Amazon Lambda icon

To summarize this service, Lambda allows us to execute functions (code) without configuring physical servers, virtual servers, and containers. We specify the Lambda functions configuration, upload the code, and define what events will start (trigger) the code execution.

We will use Lambda functions as our compute and website backend. See *Figure 1.8* for an example of a Lambda function resource:

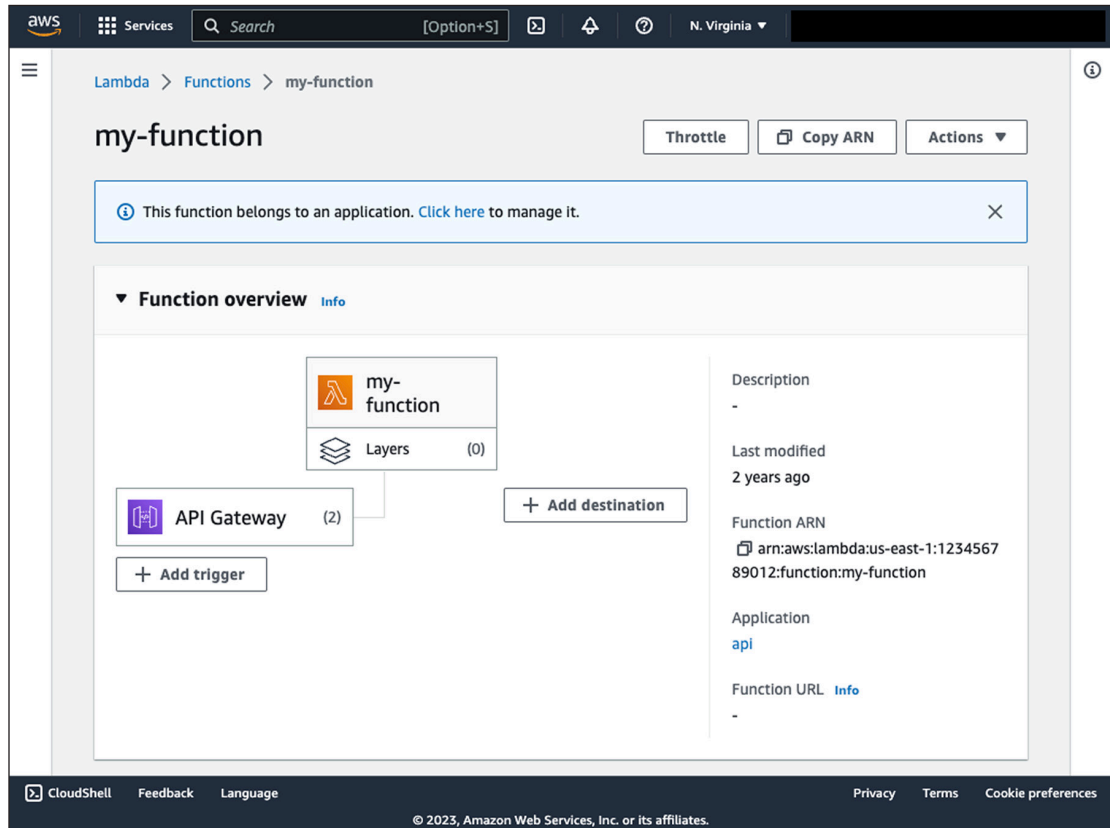


Figure 1.8: An example of a Lambda function resource

Amazon API Gateway

API Gateways allows us to create and manage an API for our applications.¹² (See the Amazon API Gateway icon in Figure 1.9.)



Figure 1.9: Amazon API Gateway icon

We can create APIs that follow the **Representational State Transfer (REST)** designs and WebSocket communications protocol. There are two flavors of RESTful APIs:

12 “What is Amazon API Gateway?” Amazon API Gateway Developer Guide. Amazon Web Services. <https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html>

- **REST APIs:** They provide rich API endpoints that support the **HTTP (Hyper-Text Transfer Protocols)** (for example, **GET, POST, PUT, DELETE OPTIONS**). They use the request-response, where an HTTP request is made to an API endpoint, and it sends a response.
- **HTTP (Hyper-Text Transfer Protocols) APIs:** They provide simpler RESTful APIs and natively support OpenID Connect and **OAuth (Open Authorization) 2.0** protocol.

We can manually create the API or use the OpenAPI¹³ Specification.

We will use API Gateway to create the API endpoints for our application. See *Figure 1.10* for an example of an API:

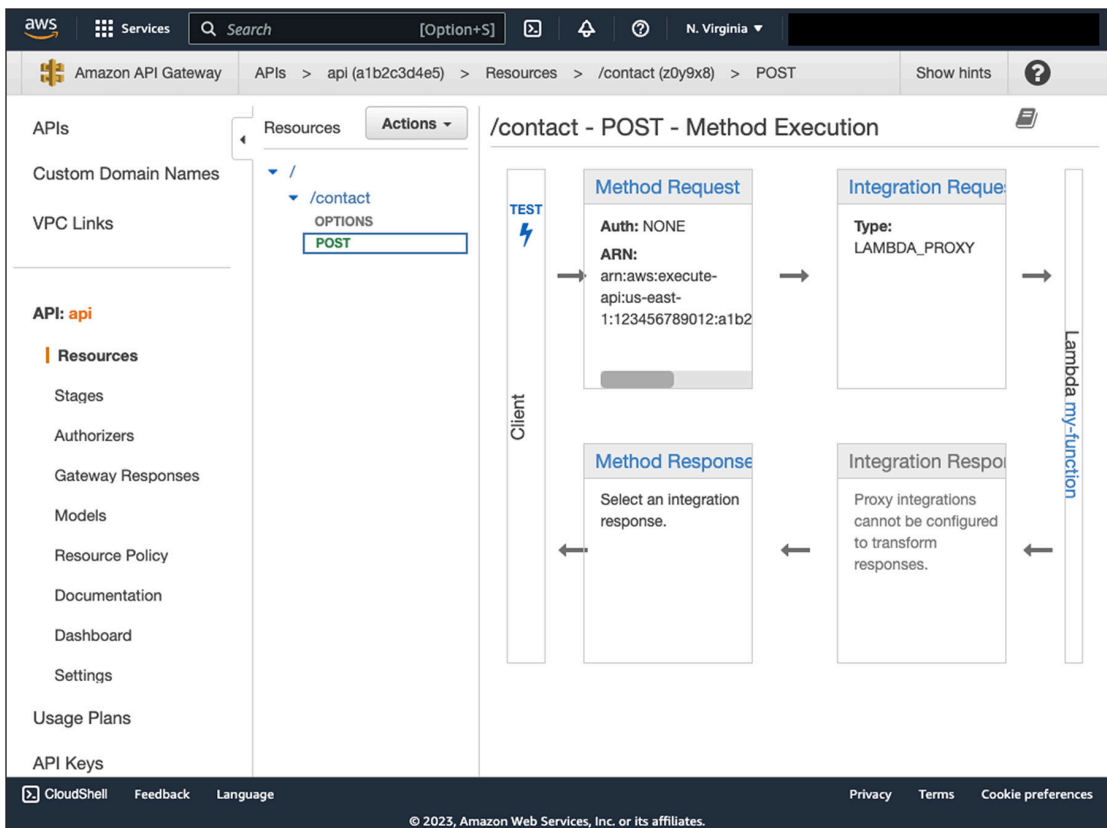


Figure 1.10: An example of an API Gateway API resource

¹³ OpenAPI is a trademark of The Linux Foundation.

Amazon Simple Storage Service

We reviewed the Amazon **Simple Storage Service (S3)** earlier in the chapter. (See the Amazon S3 icon in *Figure 1.11*)



Figure 1.11: Amazon S3 icon

To summarize this service, S3 allows us to upload various data types as objects. We associate an object with a key name and upload it to the bucket. An object can be up to 5 TB in size, and a bucket has no limit on the number of objects or total disk space.

We will use S3 to host our website files and save our application data. See *Figure 1.12* for an example of a bucket:

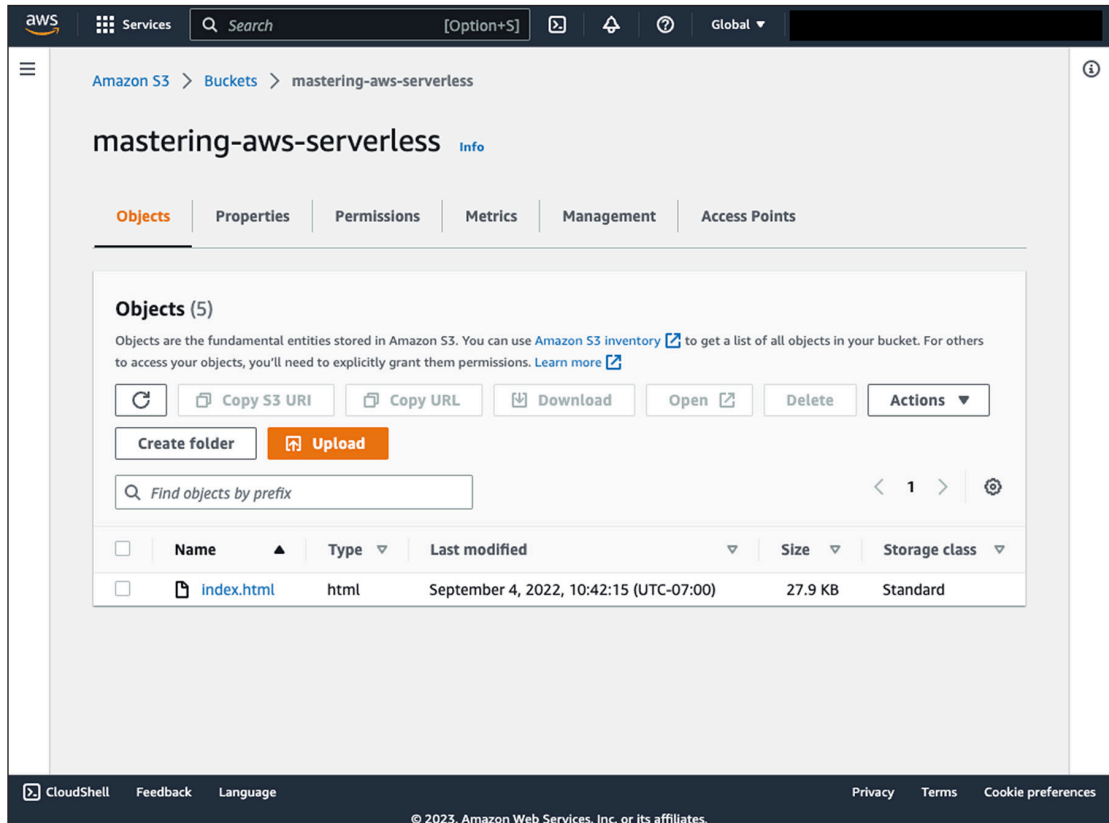
The screenshot shows the Amazon S3 console interface. At the top, there's a navigation bar with the AWS logo, 'Services', a search bar, and a 'Global' dropdown. Below that, the breadcrumb path is 'Amazon S3 > Buckets > mastering-aws-serverless'. The main heading is 'mastering-aws-serverless' with an 'Info' link. There are tabs for 'Objects', 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. The 'Objects' tab is active, showing 'Objects (5)'. A descriptive text explains that objects are fundamental entities and provides links for 'Amazon S3 inventory' and 'Learn more'. Below the text are buttons for 'Refresh', 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', and 'Actions'. There are also buttons for 'Create folder' and 'Upload'. A search bar labeled 'Find objects by prefix' is present. Below the search bar is a table with columns: Name, Type, Last modified, Size, and Storage class. The table contains one row: 'index.html', 'html', 'September 4, 2022, 10:42:15 (UTC-07:00)', '27.9 KB', and 'Standard'. At the bottom of the console, there's a footer with 'CloudShell', 'Feedback', 'Language', '© 2023, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

Figure 1.12: An example of an S3 bucket resource