

# Amazon Web Services IN ACTION

SECOND EDITION

Michael Wittig  
Andreas Wittig

Foreword by Ben Whaley



 MANNING

## Compute & Networking

Abbr.	Name	Description	Where
EC2	Amazon Elastic Compute Cloud	Virtual machines with Linux and Windows	3
	AWS Lambda	Run code without the need for virtual machines	7
EIP	Elastic IP Address	Fixed public IP address for EC2 instances	3.6
ENI	Amazon EC2 Elastic Network Interface	Virtual network interface for EC2 instances	3.7
VPC	Amazon Virtual Private Cloud	Private network inside the cloud	6.5
	Amazon EC2 Security Group	Network firewall	6.4

## Deployment & Management

Abbr.	Name	Description	Where
	AWS Elastic Beanstalk	Deployment tool for simple applications	5.4
	AWS OpsWorks	Deployment tool for multilayer applications	5.5
	AWS CloudFormation	Infrastructure automation and deployment tool	5.3
IAM	AWS Identity and Access Management	Secure access to your cloud resources (authentication and authorization)	6.3
CLI	AWS command-line interface	AWS in your terminal	4.2
SDK	AWS software development kits	AWS in your applications	4.3

## Praise for the First Edition

*Fantastic introduction to cloud basics with excellent real-world examples.*

—Rambabu Posa, GL Assessment

*A very thorough and practical guide to everything AWS ... highly recommended.*

—Scott M. King, Amazon

*Cuts through the vast expanse of official documentation and gives you what you need to make AWS work now!*

—Carm Vecchio, Computer Science Corporation (CSC)

*The right book to program AWS from scratch.*

—Javier Muñoz Mellid, Senior Computer Engineer, Igalia



*Amazon Web Services in  
Action, Second Edition*

MICHAEL WITTIG  
ANDREAS WITTIG  
FOREWORD BY BEN WHALEY



MANNING  
Shelter Island

For online information and ordering of this and other Manning books, please visit [www.manning.com](http://www.manning.com). The publisher offers discounts on this book when ordered in quantity. For more information, please contact

Special Sales Department  
Manning Publications Co.  
20 Baldwin Road  
PO Box 761  
Shelter Island, NY 11964  
Email: [orders@manning.com](mailto:orders@manning.com)

©2019 by Manning Publications Co. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.


Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in the book, and Manning Publications was aware of a trademark claim, the designations have been printed in initial caps or all caps. The following are trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries: Amazon Web Services, AWS, Amazon EC2, EC2, Amazon Elastic Compute Cloud, Amazon Virtual Private Cloud, Amazon VPC, Amazon S3, Amazon Simple Storage Service, Amazon CloudFront, CloudFront, Amazon SQS, SQS, Amazon Simple Queue Service, Amazon Simple Email Service, Amazon Elastic Beanstalk, Amazon Simple Notification Service, Amazon Route 53, Amazon RDS, Amazon Relational Database, Amazon CloudWatch, AWS Premium Support, ElastiCache, Amazon Glacier, AWS Marketplace, AWS CloudFormation, Amazon CloudSearch, Amazon DynamoDB, DynamoDB, Amazon Redshift, and Amazon Kinesis.

The icons in this book are reproduced with permission from Amazon.com or under a Creative Commons license as follows:

- AWS Simple Icons by Amazon.com (<https://aws.amazon.com/architecture/icons/>)
- File icons by Freepik (<http://www.flaticon.com/authors/freepik>) License: CC BY 3.0
- Basic application icons by Freepik (<http://www.flaticon.com/authors/freepik>) License: CC BY 3.0

All views expressed in this book are of the authors and not of AWS or Amazon.

- © Recognizing the importance of preserving what has been written, it is Manning's policy to have the books we publish printed on acid-free paper, and we exert our best efforts to that end. Recognizing also our responsibility to conserve the resources of our planet, Manning books are printed on paper that is at least 15 percent recycled and processed without the use of elemental chlorine.

 Manning Publications Co.  
20 Baldwin Road  
PO Box 761  
Shelter Island, NY 11964

Development editor: Frances Lefkowitz  
Technical development editor: John Hyaduck  
Review editor: Aleksandar Dragosavljević  
Project editor: Deirdre Hiam  
Copy editor: Benjamin Berg  
Proofreader: Elizabeth Martin  
Technical proofreader: David Fombella Pombal  
Typesetter: Gordan Salinovic  
Cover designer: Marija Tudor

ISBN 9781617295119

Printed in the United States of America

1 2 3 4 5 6 7 8 9 10 – DP – 23 22 21 20 19 18

# *brief contents*

---

<b>PART 1</b>	<b>GETTING STARTED .....</b>	<b>1</b>
	1 ■ What is Amazon Web Services? .....	3
	2 ■ A simple example: WordPress in five minutes .....	36
<b>PART 2</b>	<b>BUILDING VIRTUAL INFRASTRUCTURE CONSISTING OF COMPUTERS AND NETWORKING .....</b>	<b>57</b>
	3 ■ Using virtual machines: EC2 .....	59
	4 ■ Programming your infrastructure: The command-line, SDKs, and CloudFormation .....	102
	5 ■ Automating deployment: CloudFormation, Elastic Beanstalk, and OpsWorks .....	135
	6 ■ Securing your system: IAM, security groups, and VPC .....	165
	7 ■ Automating operational tasks with Lambda .....	199
<b>PART 3</b>	<b>STORING DATA IN THE CLOUD .....</b>	<b>233</b>
	8 ■ Storing your objects: S3 and Glacier .....	235
	9 ■ Storing data on hard drives: EBS and instance store .....	258

- 10 ■ Sharing data volumes between machines: EFS 274
- 11 ■ Using a relational database service: RDS 294
- 12 ■ Caching data in memory: Amazon ElastiCache 321
- 13 ■ Programming for the NoSQL database service: DynamoDB 349

## **PART 4 ARCHITECTING ON AWS.....381**

- 14 ■ Achieving high availability: availability zones, auto-scaling, and CloudWatch 383
- 15 ■ Decoupling your infrastructure: Elastic Load Balancing and Simple Queue Service 413
- 16 ■ Designing for fault tolerance 431
- 17 ■ Scaling up and down: auto-scaling and CloudWatch 463



# contents

---

*foreword xvii*  
*preface xix*  
*acknowledgments xxi*  
*about this book xxiii*  
*about the author xxvii*  
*about the cover illustration xxviii*

## PART 1 GETTING STARTED .....1

**1** *What is Amazon Web Services? 3*

1.1 *What is cloud computing? 4*

1.2 *What can you do with AWS? 5*

*Hosting a web shop 5* ▪ *Running a Java EE application in your private network 7* ▪ *Implementing a highly available system 8*  
*Profiting from low costs for batch processing infrastructure 9*

1.3 *How you can benefit from using AWS 10*

*Innovative and fast-growing platform 10* ▪ *Services solve common problems 10* ▪ *Enabling automation 10* ▪ *Flexible capacity (scalability) 11* ▪ *Built for failure (reliability) 11* ▪ *Reducing time to market 11* ▪ *Benefiting from economies of scale 12*  
*Global infrastructure 12* ▪ *Professional partner 12*

- 1.4 How much does it cost? 12
  - Free Tier* 13 ▪ *Billing example* 13 ▪ *Pay-per-use opportunities* 15
- 1.5 Comparing alternatives 15
- 1.6 Exploring AWS services 16
- 1.7 Interacting with AWS 19
  - Management Console* 19 ▪ *Command-line interface* 20
  - SDKs* 21 ▪ *Blueprints* 22
- 1.8 Creating an AWS account 22
  - Signing up* 23 ▪ *Signing In* 28 ▪ *Creating a key pair* 29
- 1.9 Create a billing alarm to keep track of your AWS bill 33

## 2 *A simple example: WordPress in five minutes* 36

- 2.1 Creating your infrastructure 37
- 2.2 Exploring your infrastructure 44
  - Resource groups* 44 ▪ *Virtual machines* 45 ▪ *Load balancer* 47 ▪ *MySQL database* 49 ▪ *Network filesystem* 50
- 2.3 How much does it cost? 52
- 2.4 Deleting your infrastructure 54

## PART 2 BUILDING VIRTUAL INFRASTRUCTURE CONSISTING OF COMPUTERS AND NETWORKING .....57

### 3 *Using virtual machines: EC2* 59

- 3.1 Exploring a virtual machine 60
  - Launching a virtual machine* 60 ▪ *Connecting to your virtual machine* 72 ▪ *Installing and running software manually* 75
- 3.2 Monitoring and debugging a virtual machine 76
  - Showing logs from a virtual machine* 76 ▪ *Monitoring the load of a virtual machine* 77
- 3.3 Shutting down a virtual machine 78
- 3.4 Changing the size of a virtual machine 79
- 3.5 Starting a virtual machine in another data center 82
- 3.6 Allocating a public IP address 86
- 3.7 Adding an additional network interface to a virtual machine 88
- 3.8 Optimizing costs for virtual machines 92
  - Reserve virtual machines* 93 ▪ *Bidding on unused virtual machines* 95

## 4 *Programming your infrastructure: The command-line, SDKs, and CloudFormation* 102

- 4.1 Infrastructure as Code 104
  - Automation and the DevOps movement* 104 ▀ *Inventing an infrastructure language: JIML* 105
- 4.2 Using the command-line interface 108
  - Why should you automate?* 108 ▀ *Installing the CLI* 109
  - Configuring the CLI* 110 ▀ *Using the CLI* 113
- 4.3 Programming with the SDK 117
  - Controlling virtual machines with SDK: nodecc* 118 ▀ *How nodecc creates a virtual machine* 119 ▀ *How nodecc lists virtual machines and shows virtual machine details* 120 ▀ *How nodecc terminates a virtual machine* 121
- 4.4 Using a blueprint to start a virtual machine 121
  - Anatomy of a CloudFormation template* 122 ▀ *Creating your first template* 126

## 5 *Automating deployment: CloudFormation, Elastic Beanstalk, and OpsWorks* 135

- 5.1 Deploying applications in a flexible cloud environment 136
- 5.2 Comparing deployment tools 137
  - Classifying the deployment tools* 138 ▀ *Comparing the deployment services* 138
- 5.3 Creating a virtual machine and run a deployment script on startup with AWS CloudFormation 139
  - Using user data to run a script on startup* 140 ▀ *Deploying OpenSwan: a VPN server to a virtual machine* 140 ▀ *Starting from scratch instead of updating* 145
- 5.4 Deploying a simple web application with AWS Elastic Beanstalk 145
  - Components of AWS Elastic Beanstalk* 146 ▀ *Using AWS Elastic Beanstalk to deploy Etherpad, a Node.js application* 146
- 5.5 Deploying a multilayer application with AWS OpsWorks Stacks 151
  - Components of AWS OpsWorks Stacks* 152 ▀ *Using AWS OpsWorks Stacks to deploy an IRC chat application* 153

## 6 *Securing your system: IAM, security groups, and VPC* 165

- 6.1 Who's responsible for security? 167
- 6.2 Keeping your software up to date 168
  - Checking for security updates* 168
  - *Installing security updates on startup* 169
  - *Installing security updates on running virtual machines* 170
- 6.3 Securing your AWS account 171
  - Securing your AWS account's root user* 172
  - *AWS Identity and Access Management (IAM)* 173
  - *Defining permissions with an IAM policy* 174
  - *Users for authentication, and groups to organize users* 176
  - *Authenticating AWS resources with roles* 177
- 6.4 Controlling network traffic to and from your virtual machine 179
  - Controlling traffic to virtual machines with security groups* 181
  - Allowing ICMP traffic* 182
  - *Allowing SSH traffic* 183
  - Allowing SSH traffic from a source IP address* 184
  - *Allowing SSH traffic from a source security group* 185
- 6.5 Creating a private network in the cloud: Amazon Virtual Private Cloud (VPC) 189
  - Creating the VPC and an internet gateway (IGW)* 190
  - *Defining the public bastion host subnet* 192
  - *Adding the private Apache web server subnet* 194
  - *Launching virtual machines in the subnets* 195
  - Accessing the internet from private subnets via a NAT gateway* 196

## 7 *Automating operational tasks with Lambda* 199

- 7.1 Executing your code with AWS Lambda 200
  - What is serverless?* 201
  - *Running your code on AWS Lambda* 201
  - Comparing AWS Lambda with virtual machines (Amazon EC2)* 202
- 7.2 Building a website health check with AWS Lambda 203
  - Creating a Lambda function* 204
  - *Use CloudWatch to search through your Lambda function's logs* 210
  - *Monitoring a Lambda function with CloudWatch metrics and alarms* 212
  - Accessing endpoints within a VPC* 217
- 7.3 Adding a tag containing the owner of an EC2 instance automatically 218
  - Event-driven: Subscribing to CloudWatch events* 219
  - *Implementing the Lambda function in Python* 222
  - *Setting up a Lambda function with the Serverless Application Model (SAM)* 223
  - *Authorizing a Lambda function to use other AWS services with an IAM role* 224
  - Deploying a Lambda function with SAM* 226

- 7.4 What else can you do with AWS Lambda? 227  
*What are the limitations of AWS Lambda?* 227 ▪ *Impacts of the serverless pricing model* 228 ▪ *Use case: Web application* 229  
*Use case: Data processing* 230 ▪ *Use case: IoT back end* 231

## PART 3 STORING DATA IN THE CLOUD .....233

### 8 *Storing your objects: S3 and Glacier* 235

- 8.1 What is an object store? 236
- 8.2 Amazon S3 237
- 8.3 Backing up your data on S3 with AWS CLI 238
- 8.4 Archiving objects to optimize costs 241  
*Creating an S3 bucket for the use with Glacier* 241 ▪ *Adding a lifecycle rule to a bucket* 242 ▪ *Experimenting with Glacier and your lifecycle rule* 245
- 8.5 Storing objects programmatically 248  
*Setting up an S3 bucket* 249 ▪ *Installing a web application that uses S3* 249 ▪ *Reviewing code access S3 with SDK* 250
- 8.6 Using S3 for static web hosting 252  
*Creating a bucket and uploading a static website* 253  
*Configuring a bucket for static web hosting* 253 ▪ *Accessing a website hosted on S3* 254
- 8.7 Best practices for using S3 255  
*Ensuring data consistency* 255 ▪ *Choosing the right keys* 256

### 9 *Storing data on hard drives: EBS and instance store* 258

- 9.1 Elastic Block Store (EBS): Persistent block-level storage attached over the network 259  
*Creating an EBS volume and attaching it to your EC2 instance* 260 ▪ *Using EBS* 261 ▪ *Tweaking performance* 263  
*Backing up your data with EBS snapshots* 266
- 9.2 Instance store: Temporary block-level storage 268  
*Using an instance store* 271 ▪ *Testing performance* 272  
*Backing up your data* 272

### 10 *Sharing data volumes between machines: EFS* 274

- 10.1 Creating a filesystem 277  
*Using CloudFormation to describe a filesystem* 277 ▪ *Pricing* 277
- 10.2 Creating a mount target 278

- 10.3 Mounting the EFS share on EC2 instances 280
- 10.4 Sharing files between EC2 instances 283
- 10.5 Tweaking performance 284
  - Performance mode* 285 ▪ *Expected throughput* 285
- 10.6 Monitoring a filesystem 286
  - Should you use Max I/O Performance mode?* 286 ▪ *Monitoring your permitted throughput* 287 ▪ *Monitoring your usage* 288
- 10.7 Backing up your data 289
  - Using CloudFormation to describe an EBS volume* 290 ▪ *Using the EBS volume* 290

## 11 *Using a relational database service: RDS* 294

- 11.1 Starting a MySQL database 296
  - Launching a WordPress platform with an RDS database* 297
  - Exploring an RDS database instance with a MySQL engine* 299
  - Pricing for Amazon RDS* 300
- 11.2 Importing data into a database 300
- 11.3 Backing up and restoring your database 303
  - Configuring automated snapshots* 303 ▪ *Creating snapshots manually* 304 ▪ *Restoring a database* 305 ▪ *Copying a database to another region* 307 ▪ *Calculating the cost of snapshots* 308
- 11.4 Controlling access to a database 308
  - Controlling access to the configuration of an RDS database* 309
  - Controlling network access to an RDS database* 310 ▪ *Controlling data access* 311
- 11.5 Relying on a highly available database 311
  - Enabling high-availability deployment for an RDS database* 313
- 11.6 Tweaking database performance 314
  - Increasing database resources* 314 ▪ *Using read replication to increase read performance* 316
- 11.7 Monitoring a database 318

## 12 *Caching data in memory: Amazon ElastiCache* 321

- 12.1 Creating a cache cluster 327
  - Minimal CloudFormation template* 327 ▪ *Test the Redis cluster* 328

- 12.2 Cache deployment options 330
  - Memcached: cluster* 330 ▪ *Redis: Single-node cluster* 331
  - Redis: Cluster with cluster mode disabled* 332 ▪ *Redis: Cluster with cluster mode enabled* 332
- 12.3 Controlling cache access 334
  - Controlling access to the configuration* 334 ▪ *Controlling network access* 334 ▪ *Controlling cluster and data access* 335
- 12.4 Installing the sample application Discourse with CloudFormation 336
  - VPC: Network configuration* 337 ▪ *Cache: Security group, subnet group, cache cluster* 338 ▪ *Database: Security group, subnet group, database instance* 339 ▪ *Virtual machine—security group, EC2 instance* 340
  - Testing the CloudFormation template for Discourse* 342
- 12.5 Monitoring a cache 344
  - Monitoring host-level metrics* 344 ▪ *Is my memory sufficient?* 345 ▪ *Is my Redis replication up-to-date?* 345
- 12.6 Tweaking cache performance 346
  - Selecting the right cache node type* 347 ▪ *Selecting the right deployment option* 347 ▪ *Compressing your data* 348

## 13 Programming for the NoSQL database service: DynamoDB 349

- 13.1 Operating DynamoDB 351
  - Administration* 352 ▪ *Pricing* 352 ▪ *Networking* 353
  - RDS comparison* 353 ▪ *NoSQL comparison* 354
- 13.2 DynamoDB for developers 354
  - Tables, items, and attributes* 354 ▪ *Primary key* 355
  - DynamoDB Local* 356
- 13.3 Programming a to-do application 356
- 13.4 Creating tables 358
  - Users are identified by a partition key* 358 ▪ *Tasks are identified by a partition key and sort key* 360
- 13.5 Adding data 361
  - Adding a user* 363 ▪ *Adding a task* 363
- 13.6 Retrieving data 364
  - Getting an item by key* 365 ▪ *Querying items by key and filter* 366 ▪ *Using global secondary indexes for more flexible queries* 368 ▪ *Scanning and filtering all of your table's data* 371
  - Eventually consistent data retrieval* 372

- 13.7 Removing data 373
- 13.8 Modifying data 374
- 13.9 Scaling capacity 375
  - Capacity units* 375 ▪ *Auto-scaling* 377

## PART 4 ARCHITECTING ON AWS.....381

### 14 *Achieving high availability: availability zones, auto-scaling, and CloudWatch* 383

- 14.1 Recovering from EC2 instance failure with CloudWatch 385
  - Creating a CloudWatch alarm to trigger recovery when status checks fail* 387 ▪ *Monitoring and recovering a virtual machine based on a CloudWatch alarm* 388
- 14.2 Recovering from a data center outage 392
  - Availability zones: groups of isolated data centers* 392 ▪ *Using auto-scaling to ensure that an EC2 instance is always running* 396
  - Recovering a failed virtual machine to another availability zone with the help of auto-scaling* 399 ▪ *Pitfall: recovering network-attached storage* 402 ▪ *Pitfall: network interface recovery* 407
- 14.3 Analyzing disaster-recovery requirements 411
  - RTO and RPO comparison for a single EC2 instance* 411

### 15 *Decoupling your infrastructure: Elastic Load Balancing and Simple Queue Service* 413

- 15.1 Synchronous decoupling with load balancers 415
  - Setting up a load balancer with virtual machines* 416
- 15.2 Asynchronous decoupling with message queues 420
  - Turning a synchronous process into an asynchronous one* 421
  - Architecture of the URL2PNG application* 422 ▪ *Setting up a message queue* 423 ▪ *Producing messages programmatically* 423
  - Consuming messages programmatically* 425 ▪ *Limitations of messaging with SQS* 428

### 16 *Designing for fault tolerance* 431

- 16.1 Using redundant EC2 instances to increase availability 434
  - Redundancy can remove a single point of failure* 434
  - Redundancy requires decoupling* 436



- 16.2 Considerations for making your code fault-tolerant 437
  - Let it crash, but also retry* 437
  - *Idempotent retry makes fault tolerance possible* 438
- 16.3 Building a fault-tolerant web application: Imagery 440
  - The idempotent state machine* 443
  - *Implementing a fault-tolerant web service* 444
  - *Implementing a fault-tolerant worker to consume SQS messages* 452
  - *Deploying the application* 455

## 17 *Scaling up and down: auto-scaling and CloudWatch* 463

- 17.1 Managing a dynamic EC2 instance pool 465
- 17.2 Using metrics or schedules to trigger scaling 469
  - Scaling based on a schedule* 471
  - *Scaling based on CloudWatch metrics* 472
- 17.3 Decouple your dynamic EC2 instance pool 475
  - Scaling a dynamic EC2 instance pool synchronously decoupled by a load balancer* 476
  - *Scaling a dynamic EC2 instances pool asynchronously decoupled by a queue* 480
- index* 487



## *foreword*

---

Throughout the late 1990s and early 2000s I worked in the rank and file of system administrators endeavoring to keep network services online, secure, and available to users. At the time, administration was a tedious, onerous affair involving cable slinging, server racking, installing from optical media, and configuring software manually. It was thankless work, often an exercise in frustration, requiring patience, persistence, and plenty of caffeine. To participate in the emerging online marketplace, businesses of the era bore the burden of managing this physical infrastructure, accepting the associated capital and operating costs and hoping for enough success to justify those expenses.

When Amazon Web Services emerged in 2006, it signaled a shift in the industry. Management of compute and storage resources was dramatically simplified, and the cost of building and launching applications plummeted. Suddenly anyone with a good idea and the ability to execute could build a global business on world-class infrastructure at a starting cost of just a few cents an hour. The AWS value proposition was immediately apparent, ushering in a wave of new startups, data center migrations, and third-party service providers. In terms of cumulative disruption of an established market, a few technologies stand above all others, and AWS is among them.

Today, the march of progress continues unabated. In December 2017 at its annual re:Invent conference in Las Vegas, Werner Vogels, CTO of Amazon, announced to more than 40,000 attendees that the company had released 3,951 new features and services since the first conference in 2012. AWS has an \$18 billion annual run rate and 40% year-over-year growth. Enterprises, startups, and governments alike have adopted the AWS cloud en masse. The numbers are staggering, and AWS shows no signs of slowing down.

Needless to say, this growth and innovation comes at the expense of considerable complexity. The AWS cloud is composed of scores of services and thousands of features, enabling powerful new applications and highly efficient designs. But it is accompanied by a brand-new lexicon with distinct architectural and technical best practices. The platform can bewilder the neophyte. How does one know where to begin?

*Amazon Web Services in Action, Second Edition*, slices through the complexity of AWS using examples and visuals to cement knowledge in the minds of readers. Andreas and Michael focus on the most prominent services and features that users are most likely to need. Code snippets are sprinkled throughout each chapter, reinforcing the programmable nature of the cloud. And because many readers will be footing the bill from AWS personally, any examples that incur charges are called out explicitly throughout the text.

As a consultant, author, and at heart an engineer, I celebrate all efforts to introduce the bewildering world of cloud computing to new users. *Amazon Web Services in Action, Second Edition* is at the head of the pack as a confident, practical guide through the maze of the industry's leading cloud platform.

With this book as your sidekick, what will you build on the AWS cloud?

—BEN WHALEY, AWS COMMUNITY HERO AND AUTHOR

## *preface*

---

When we started our career as software developers in 2008, we didn't care about operations. We wrote code, and someone else was responsible for deployment and operations. There was a huge gap between software development and IT operations. On top of that, releasing new features was a huge risk because it was impossible to test all the changes to software and infrastructure manually. Every six months, when new features needed to be deployed, we experienced a nightmare.

Time passed, and in 2012 we became responsible for a product: an online banking platform. Our goal was to iterate quickly and to be able to release new features to the product every week. Our software was responsible for managing money, so the quality and security of the software and infrastructure was as important as the ability to innovate. But the inflexible on-premises infrastructure and the outdated process of deploying software made that goal impossible to reach. We started to look for a better way.

Our search led us to Amazon Web Services, which offered us a flexible and reliable way to build and operate our applications. The possibility of automating every part of our infrastructure was fascinating. Step by step, we dove into the different AWS services, from virtual machines to distributed message queues. Being able to outsource tasks like operating an SQL database or a load balancer saved us a lot of time. We invested this time in automating testing and operations for our entire infrastructure.

Technical aspects weren't the only things that changed during this transformation to the cloud. After a while the software architecture changed from a monolithic application to microservices, and the separation between software development and operations disappeared. Instead we built our organization around the core principle of DevOps: you build it, you run it.

We have worked as independent consultants since 2015, helping our clients get the most out of AWS. We've accompanied startups, mid-sized companies, and enterprises on their journey to the cloud. Besides designing and implementing cloud architectures based on AWS services, we are focusing on infrastructure as code, continuous deployment, Docker, serverless, security, and monitoring.

We enjoyed writing the first edition of our book in 2015. The astonishing support from Manning and our MEAP readers allowed us to finish the whole book in only nine months. Above all, it was a pleasure to observe you—our readers—using our book to get started with AWS or deepen your knowledge.

AWS is innovating and constantly releases new features or whole new services. Therefore, it was about time to update our book in 2017. We started to work on the second edition of our book in June. Within six months we updated all chapters, added three more chapters, and improved the book based on the feedback of our readers and our editors.

We hope you enjoy the second edition of *Amazon Web Services in Action* as much as we do!

## *acknowledgments*

---

Writing a book is time-consuming. We invested our time, and other people did as well. We think that time is the most valuable resource on Earth, and we want to honor every minute spent by the people who helped us with this book.

To all the readers who bought the first edition of our book: thanks so much for your trust and support. Watching you reading our book and working through the examples boosted our motivation. Also, we learned quite a bit from your feedback.

Next, we want to thank all the readers who bought the MEAP edition of this book. Thanks for overlooking the rough edges and focusing on learning about AWS instead. Your feedback helped us to polish the version of the book that you are now reading.

Thank you to all the people who posted comments in the Book Forum and who provided excellent feedback that improved the book.

In addition, thanks to all the reviewers of the second and first edition who provided detailed comments from the first to the last page. The reviewers for this second edition are Antonio Pessolano, Ariel Gamino, Christian Bridge-Harrington, Christof Marte, Eric Hammond, Gary Hubbart, Hazem Farahat, Jean-Pol Landrain, Jim Amrhein, John Guthrie, Jose San Leandro, Lynn Langit, Maciej Drozdowski, Manoj Agarwal, Peeyush Maharshi, Philip Patterson, Ryan Burrows, Shaun Hickson, Terry Rickman, and Thorsten Höger. Your feedback helped shape this book—we hope you like it as much as we do.

Special thanks to Michael Labib for his input and feedback on chapter 12 covering AWS ElastiCache.

Furthermore, we want to thank John Hyaduck, our technical developmental editor. Your unbiased and technical view on Amazon Web Services and our book helped to perfect the second edition. Thanks to Jonathan Thoms, the technical editor of the first edition as well.

David Fombella Pombal and Doug Warren made sure all the examples within our book are working as expected. Thanks for proofing the technical parts of our book.

We also want to thank Manning Publications for placing their trust in us. Especially, we want to thank the following staff at Manning for their excellent work:

- Frances Lefkowitz, our development editor, who guided us through the process of writing the second edition. Her writing and teaching expertise is noticeable in every part of our book. Thanks for your support.
- Dan Maharry, our development editor while writing the first edition. Thanks for taking us by the hand from writing the first pages to finishing our first book.
- Aleksandar Dragosavljević, who organized the reviews of our book. Thanks for making sure we got valuable feedback from our readers.
- Benjamin Berg and Tiffany Taylor, who perfected our English. We know you had a hard time with us, but our mother tongue is German, and we thank you for your efforts.
- Candace Gillhoolley, Ana Romac, and Christopher Kaufmann, who helped us to promote this book.
- Janet Vail, Deirdre Hiam, Elizabeth Martin, Mary Piergies, Gordan Salinovic, David Novak, Barbara Mirecki, Marija Tudor, and all the others who worked behind the scenes and who took our rough draft and turned it into a real book.

Many thanks to Ben Whaley for contributing the foreword to our book.

Last but not least, we want to thank the significant people in our lives who supported us as we worked on the book. Andreas wants to thank his wife Simone, and Michael wants to thank his partner Kathrin, for their patience and encouragement.



## *about this book*

---

Our book guides you from creating an AWS account to building fault-tolerant and auto-scaling applications. You will learn about services offering compute, network, and storage capacity. We get you started with everything you need to run web applications on AWS: load balancers, virtual machines, file storage, database systems, and in-memory caches.

The first part of the book introduces the principles of Amazon Web Services and gives you a first impression of the possibilities in the cloud. Next, you will learn about fundamental compute and network services. Afterward, we demonstrate six different ways to store your data. The last part of our book focuses on highly available or even fault-tolerant architectures that allow you to scale your infrastructure dynamically as well.

Amazon offers a wide variety of services. Unfortunately, the number of pages within a book is limited. Therefore, we had to skip topics such as containers, big data, and machine learning. We cover the basic or most important services, though.

Automation sneaks in throughout the book, so by the end you'll be comfortable with using AWS CloudFormation, an infrastructure-as-code tool that allows you to manage your cloud infrastructure in an automated way; this will be one of the most important things you will learn from our book.

Most of our examples use popular web applications to demonstrate important points. We use tools offered by AWS instead of third-party tools whenever possible, as we appreciate the quality and support offered by AWS. Our book focuses on the different aspects of security in the cloud, for example by following the “least privilege” principle when accessing cloud resources.

We focus on Linux as the operating system for virtual machines in the book. Our examples are based on open source software.

Amazon operates data centers in geographic regions around the world. To simplify the examples we are using the region US East (N. Virginia) within our book. You will also learn how to switch to another region to exemplarily make use of resources in Asia Pacific (Sydney).

## **Roadmap**

Chapter 1 introduces cloud computing and Amazon Web Services. You'll learn about key concepts and basics, and you'll create and set up your AWS account.

Chapter 2 brings Amazon Web Services into action. You'll spin up and dive into a complex cloud infrastructure with ease.

Chapter 3 is about working with a virtual machine. You'll learn about the key concepts of the Elastic Compute Service (EC2) with the help of a handful of practical examples.

Chapter 4 presents different approaches for automating your infrastructure: the AWS command-line interface (CLI) from your terminal, the AWS SDKs to program in your favorite language, as well as AWS CloudFormation, an infrastructure-as-code tool.

Chapter 5 introduces three different ways to deploy software to AWS. You'll use each of the tools to deploy an application to AWS in an automated fashion.

Chapter 6 is about security. You'll learn how to secure your networking infrastructure with private networks and firewalls. You'll also learn how to protect your AWS account and your cloud resources.

Chapter 7 is about automating operational tasks with AWS Lambda. You will learn how to execute small code snippets in the cloud without the need of launching a virtual machine.

Chapter 8 introduces Amazon Simple Storage Service (S3), a service offering object storage, and Amazon Glacier, a service offering long-term storage. You'll learn how to integrate object storage into your applications to implement a stateless server by creating an image gallery.

Chapter 9 is about storing data from your virtual machines on hard drives with Amazon Elastic Block Storage (EBS) and instance storage. In order to get an idea of the different options available, you will take some performance measurements.

Chapter 10 explains how to use a networking filesystem to share data between multiple machines. Therefore, we introduce the Amazon Elastic File System (EFS).

Chapter 11 introduces Amazon Relational Database Service (RDS), which offers managed relational database systems like MySQL, PostgreSQL, Oracle, and Microsoft SQL Server. You will learn how to connect an application to an RDS database instance, for example.

Chapter 12 is about adding a cache to your infrastructure to speed up your application and save costs due to minimizing load on the database layer. Specifically, you will learn about Amazon ElastiCache, which provides Redis or memcached as a service.

Chapter 13 introduces Amazon DynamoDB, a NoSQL database offered by AWS. DynamoDB is typically not compatible with legacy applications. You need to rework your applications to be able to make use of DynamoDB instead. You'll implement a to-do application in this chapter.

Chapter 14 explains what is needed to make your infrastructure highly available. You will learn how to recover from a failed virtual machine or even a whole datacenter automatically.

Chapter 15 introduces the concept of decoupling your system to increase reliability. You'll learn how to use synchronous decoupling with the help of Elastic Load Balancing (ELB). Asynchronous decoupling is also part of this chapter; we explain how to use the Amazon Simple Queue Service (SQS), a distributed queuing service, to build a fault-tolerant system.

Chapter 16 dives into building fault-tolerant applications based on the concepts explained in chapter 14 and 15. You will create a fault-tolerant image processing web services within this chapter.

Chapter 17 is all about flexibility. You'll learn how to scale the capacity of your infrastructure based on a schedule or based on the current load of your system.

### **Code conventions and downloads**

You'll find four types of code listings in this book: Bash, YAML, Python, and Node.js/JavaScript. We use Bash to create tiny scripts to interact with AWS in an automated way. YAML is used to describe infrastructure in a way that AWS CloudFormation can understand. In addition, we use Python to manage our cloud infrastructure. Also, we use the Node.js platform to create small applications in JavaScript to build cloud-native applications.

This book contains many examples of source code both in numbered listings and in line with normal text. In both cases, source code is formatted in a `fixed-width font like this` to separate it from ordinary text. Code annotations accompany many of the listings, highlighting important concepts. Sometimes we needed to break a line into two or more to fit on the page. In our Bash code we used the continuation backslash. In our YAML, Python, and Node.js/JavaScript code, an artificial line break is indicated by this symbol: `➤`.

The code for the examples in this book is available for download from the publisher's website at <https://www.manning.com/books/amazon-web-services-in-action-second-edition> and from GitHub at <https://github.com/awsinAction/code2>.

### **Book forum**

Purchase of *Amazon Web Services in Action, Second Edition* includes free access to a private web forum run by Manning Publications where you can make comments about the book, ask technical questions, and receive help from the author and from other users. To access the forum, go to <https://forums.manning.com/forums/amazon-web-services-in-action-second-edition>. You can also learn more about Manning's forums and the rules of conduct at <https://forums.manning.com/forums/about>.

Manning's commitment to our readers is to provide a venue where a meaningful dialogue between individual readers and between readers and the authors can take place. It is not a commitment to any specific amount of participation on the part of the authors, whose contribution to the forum remains voluntary (and unpaid). We suggest you try asking the authors some challenging questions lest their interest stray! The forum and the archives of previous discussions will be accessible from the publisher's website as long as the book is in print.

## *about the authors*

---



Andreas Wittig and Michael Wittig are software and DevOps engineers focusing on Amazon Web Services. The brothers started building on AWS in 2013 when migrating the IT infrastructure of a German bank to AWS—the first bank in Germany to do so. Since 2015, Andreas and Michael have worked as consultants helping their clients to migrate and run their workloads on AWS. They focus on infrastructure-as-code, continuous deployment, serverless, Docker, and security. Andreas and Michael build SaaS products on top of the Amazon’s cloud as well. Both are certified as AWS Certified Solutions Architect - Professional and AWS Certified DevOps Engineer - Professional. In addition, Andreas and Michael love sharing their knowledge and teaching how to use Amazon Web Services through this book, their blog ([cloudonaut.io](https://cloudonaut.io)), as well as online- and on-site trainings (such as *AWS in Motion* [<https://www.manning.com/livevideo/aws-in-motion>]).



## *about the cover illustration*

---

The figure on the cover of *Amazon Web Services in Action, Second Edition* is captioned “Pay-san du Canton de Lucerne,” or a peasant from the canton of Lucerne in central Switzerland. The illustration is taken from a collection of dress costumes from various countries by Jacques Grasset de Saint-Sauveur (1757-1810), titled *Costumes de Différent Pays*, published in France in 1797. Each illustration is finely drawn and colored by hand.

The rich variety of Grasset de Saint-Sauveur’s collection reminds us vividly of how culturally apart the world’s towns and regions were just 200 years ago. Isolated from each other, people spoke different dialects and languages. In the streets or in the countryside, it was easy to identify where they lived and what their trade or station in life was just by their dress.

The way we dress has changed since then and the diversity by region, so rich at the time, has faded away. It is now hard to tell apart the inhabitants of different continents, let alone different towns, regions, or countries. Perhaps we have traded cultural diversity for a more varied personal life—certainly for a more varied and fast-paced technological life.

At a time when it is hard to tell one computer book from another, Manning celebrates the inventiveness and initiative of the computer business with book covers based on the rich diversity of regional life of two centuries ago, brought back to life by Grasset de Saint-Sauveur’s pictures.

# Part 1

## Getting started

**H**ave you watched a blockbuster on Netflix, bought a gadget on [Amazon.com](https://www.amazon.com), or booked a room on Airbnb today? If so, you have used Amazon Web Services (AWS) in the background. Because Netflix, Amazon.com, and Airbnb all use Amazon Web Services for their business.

Amazon Web Services is the biggest player in the cloud computing markets. According to analysts, AWS maintains a market share of more than 30%.<sup>1</sup> Another impressive number: AWS reported net sales of \$4.1 billion USD for the quarter ending in June 2017.<sup>2</sup> AWS data centers are distributed worldwide in North America, South America, Europe, Asia, and Australia. But the cloud does not consist of hardware and computing power alone. Software is part of every cloud platform and makes the difference for you, as a customer who aims to provide a valuable experience to your services's users. The research firm Gartner has yet again classified AWS as a leader in their Magic Quadrant for Cloud Infrastructure as a Service in 2017. Gartner's Magic Quadrant groups vendors into four quadrants: niche players, challengers, visionaries, and leaders, and provides a quick overview of the cloud computing market.<sup>3</sup> Being recognized as a leader attests AWS's high speed and high quality of innovation.

---

<sup>1</sup> Synergy Research Group, "The Leading Cloud Providers Continue to Run Away with the Market," <http://mng.bz/qDYo>.

<sup>2</sup> Amazon, 10-Q for Quarter Ended June 30 (2017), <http://mng.bz/1LAX>.

<sup>3</sup> AWS Blog, "AWS Named as a Leader in Gartner's Infrastructure as a Service (IaaS) Magic Quadrant for 7th Consecutive Year," <http://mng.bz/0W1W>.

The first part of this book will guide you through your initial steps with AWS. You will get an impression of how you can use AWS to improve your IT infrastructure.

Chapter 1 introduces cloud computing and AWS. This will get you familiar with the big-picture basics of how AWS is structured.

Chapter 2 brings Amazon Web Service into action. Here, you will spin up and dive into a complex cloud infrastructure with ease.



# What is Amazon Web Services?

---

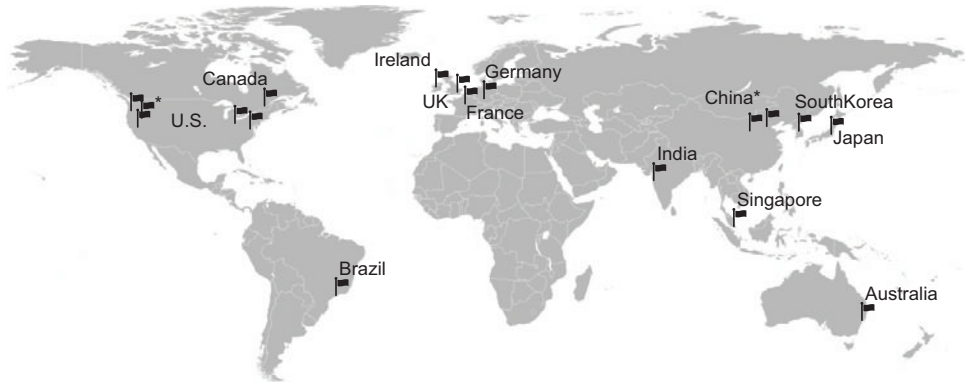
## ***This chapter covers***

- Overview of Amazon Web Services
- The benefits of using Amazon Web Services
- What you can do with Amazon Web Services
- Creating and setting up an AWS account

*Amazon Web Services* (AWS) is a platform of web services that offers solutions for computing, storing, and networking, at different layers of abstraction. For example, you can use block-level storage (a low level of abstraction) or a highly distributed object storage (a high level of abstraction) to store your data. You can use these services to host websites, run enterprise applications, and mine tremendous amounts of data. *Web services* are accessible via the internet by using typical web protocols (such as HTTP) and used by machines or by humans through a UI. The most prominent services provided by AWS are EC2, which offers virtual machines, and S3, which offers storage capacity. Services on AWS work well together: you can use them to replicate your existing local network setup, or you can design a new setup from scratch. The pricing model for services is pay-per-use.

As an AWS customer, you can choose among different *data centers*. AWS data centers are distributed worldwide. For example, you can start a virtual machine in Japan in exactly the same way as you would start one in Ireland. This enables you to serve customers worldwide with a global infrastructure.

The map in figure 1.1 shows AWS's data centers. Access is limited to some of them: some data centers are accessible for U.S. government organizations only, and special conditions apply for the data centers in China. Additional data centers have been announced for Bahrain, Hong Kong, Sweden, and the U.S..



\* Limited access

**Figure 1.1** AWS data center locations

In more general terms, AWS is known as a *cloud computing platform*.

## 1.1 **What is cloud computing?**

Almost every IT solution is labeled with the term *cloud computing* or just *cloud* nowadays. Buzzwords like this may help sales, but they're hard to work with in a book. So for the sake of clarity, let's define some terms.

Cloud computing, or the cloud, is a metaphor for supply and consumption of IT resources. The IT resources in the cloud aren't directly visible to the user; there are layers of abstraction in between. The level of abstraction offered by the cloud varies, from offering virtual machines (VMs) to providing software as a service (SaaS) based on complex distributed systems. Resources are available on demand in enormous quantities, and you pay for what you use.

The official definition from the National Institute of Standards and Technology:

*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (networks, virtual machines, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*

—National Institute of Standards and Technology, *The NIST Definition of Cloud Computing*

Clouds are often divided into three types:

- *Public*—A cloud managed by an organization and open to use by the general public.
- *Private*—A cloud that virtualizes and distributes the IT infrastructure for a single organization.
- *Hybrid*—A mixture of a public and a private cloud.

AWS is a public cloud. Cloud computing services also have several classifications:

- *Infrastructure as a service (IaaS)*—Offers fundamental resources like computing, storage, and networking capabilities, using virtual machines such as Amazon EC2, Google Compute Engine, and Microsoft Azure.
- *Platform as a service (PaaS)*—Provides platforms to deploy custom applications to the cloud, such as AWS Elastic Beanstalk, Google App Engine, and Heroku.
- *Software as a service (SaaS)*—Combines infrastructure and software running in the cloud, including office applications like Amazon WorkSpaces, Google Apps for Work, and Microsoft Office 365.

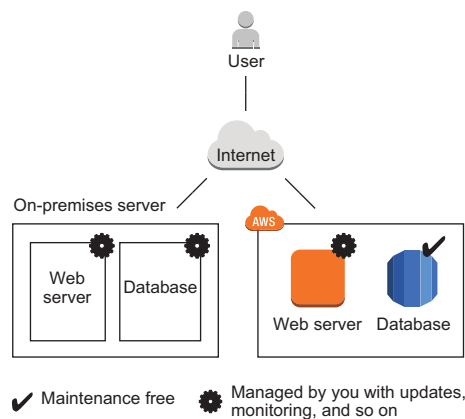
The AWS product portfolio contains IaaS, PaaS, and SaaS. Let's take a more concrete look at what you can do with AWS.

## 1.2 What can you do with AWS?

You can run all sorts of application on AWS by using one or a combination of services. The examples in this section will give you an idea of what you can do.

### 1.2.1 Hosting a web shop

John is CIO of a medium-sized e-commerce business. He wants to develop a fast and reliable web shop. He initially decided to host the web shop on-premises, and three years ago he rented machines in a data center. A web server handles requests from customers, and a database stores product information and orders. John is evaluating how his company can take advantage of AWS by running the same setup on AWS, as shown in figure 1.2.

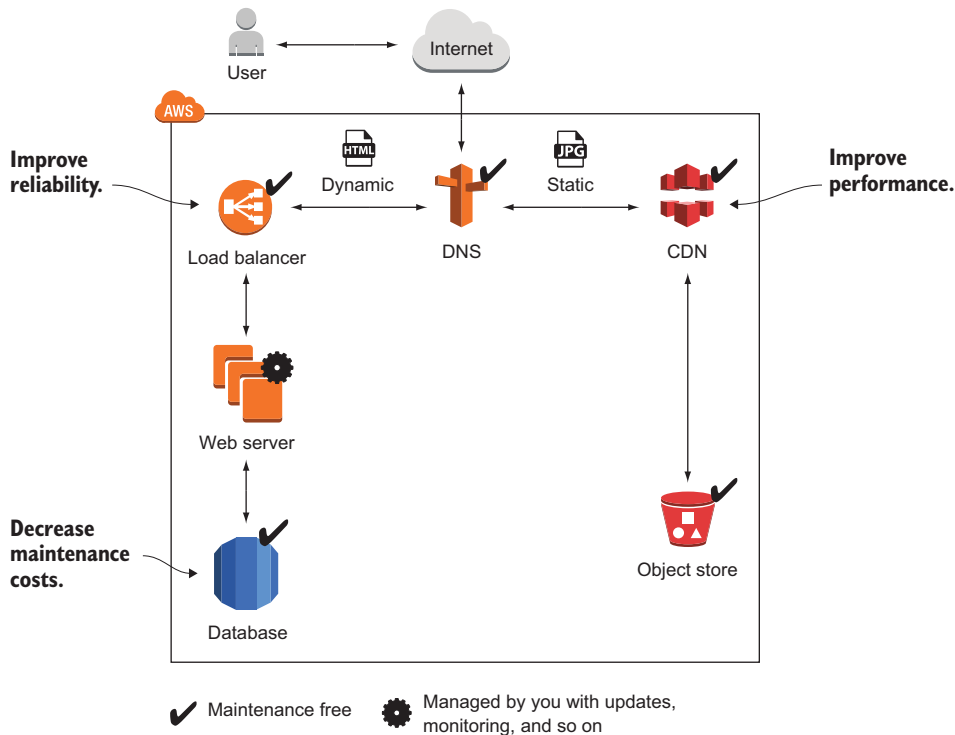


**Figure 1.2** Running a web shop on-premises vs. on AWS

John not only wants to lift-and-shift his current on-premises infrastructure to AWS; he wants to get the most out of the advantages the cloud is offering. Additional AWS services allow John to improve his setup.

- The web shop consists of dynamic content (such as products and their prices) and static content (such as the company logo). Splitting these up would reduce the load on the web servers and improve performance by delivering the static content over a content delivery network (CDN).
- Switching to maintenance-free services including a database, an object store, and a DNS system would free John from having to manage these parts of the system, decreasing operational costs and improving quality.
- The application running the web shop can be installed on virtual machines. Using AWS, John can run the same amount of resources he was using on his on-premises machine, but split into multiple smaller virtual machines at no extra cost. If one of these virtual machines fails, the load balancer will send customer requests to the other virtual machines. This setup improves the web shop's reliability.

Figure 1.3 shows how John enhanced the web shop setup with AWS.



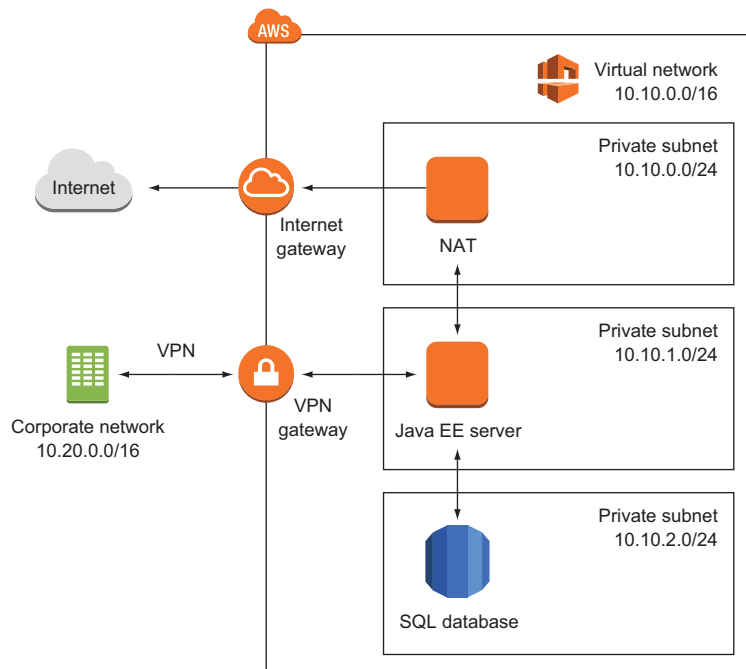
**Figure 1.3** Running a web shop on AWS with CDN for better performance, a load balancer for high availability, and a managed database to decrease maintenance costs

John is happy with running his web shop on AWS. By migrating his company's infrastructure to the cloud, he was able to increase the reliability and performance of the web shop.

### 1.2.2 Running a Java EE application in your private network

Maureen is a senior system architect in a global corporation. She wants to move parts of her company's business applications to AWS when the data-center contract expires in a few months, to reduce costs and gain flexibility. She wants to run enterprise applications (such as Java EE applications) consisting of an application server and an SQL database on AWS. To do so, she defines a virtual network in the cloud and connects it to the corporate network through a Virtual Private Network (VPN) connection. She installs application servers on virtual machines to run the Java EE application. Maureen also wants to store data in an SQL database service (such as Oracle Database Enterprise Edition or Microsoft SQL Server EE).

For security, Maureen uses subnets to separate systems with different security levels from each other. By using access-control lists, she can control ingoing and outgoing traffic for each subnet. For example, the database is only accessible from the JEE server's subnet which helps to protect mission-critical data. Maureen controls traffic to the internet by using Network Address Translation (NAT) and firewall rules as well. Figure 1.4 illustrates Maureen's architecture.



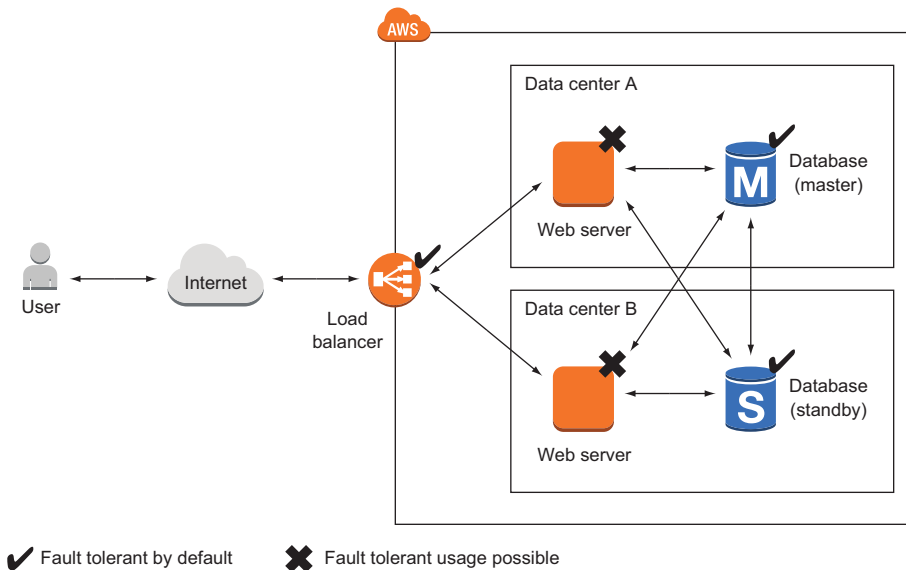
**Figure 1.4** Running a Java EE application with enterprise networking on AWS improves flexibility and lowers costs.

Maureen has managed to connect the local data center with a private network running remotely on AWS to enable clients to access the JEE server. To get started, Maureen uses a VPN connection between the local data center and AWS, but she is already thinking about setting up a dedicated network connection to reduce network costs and increase network throughput in the future.

The project was a great success for Maureen. She was able to reduce the time needed to set up an enterprise application from months to hours, as AWS can take care of the virtual machines, databases, and even the networking infrastructure on demand within a few minutes. Maureen's project also benefits from lower infrastructure costs on AWS, compared to using their own infrastructure on-premises.

### 1.2.3 Implementing a highly available system

Alexa is a software engineer working for a fast-growing startup. She knows that Murphy's Law applies to IT infrastructure: anything that can go wrong *will* go wrong. Alexa is working hard to build a highly available system to prevent outages from ruining the business. All services on AWS are either highly available or can be used in a highly available way. So, Alexa builds a system like the one shown in figure 1.5 with a high availability architecture. The database service is offered with replication and fail-over handling. In case the master database instance fails, the standby database is promoted as the new master database automatically. Alexa uses virtual machines acting as web servers. These virtual machines aren't highly available by default, but Alexa launches multiple virtual machines in different data centers to achieve high availability. A load balancer checks the health of the web servers and forwards requests to healthy machines.



**Figure 1.5** Building a highly available system on AWS by using a load balancer, multiple virtual machines, and a database with master-standby replication

So far, Alexa has protected the startup from major outages. Nevertheless, she and her team are always planning for failure and are constantly improving the resilience of their systems.

### 1.2.4 Profiting from low costs for batch processing infrastructure

Nick is a data scientist who needs to process massive amounts of measurement data collected from gas turbines. He needs to generate a report containing the maintenance condition of hundreds of turbines daily. Therefore, his team needs a computing infrastructure to analyze the newly arrived data once a day. Batch jobs are run on a schedule and store aggregated results in a database. A business intelligence (BI) tool is used to generate reports based on the data stored in the database.

As the budget for computing infrastructure is very small, Nick and his team have been looking for a cost effective solution to analyze their data. He finds a way to make clever use of AWS's price model:

- *AWS bills virtual machines per minute.* So Nick launches a virtual machine when starting a batch job, and terminates it immediately after the job finished. Doing so allows him to pay for computing infrastructure only when actually using it. This is a big game changer compared to the traditional data center where Nick had to pay a monthly fee for each machine, no matter how much it was used.
- *AWS offers spare capacity in their data centers at substantial discount.* It is not important for Nick to run a batch job at a specific time. He can wait to execute a batch job until there is enough spare capacity available, so AWS offers him a virtual machine with a discount of 50%.

Figure 1.6 illustrates how Nick benefits from the pay-per-use price model for virtual machines.

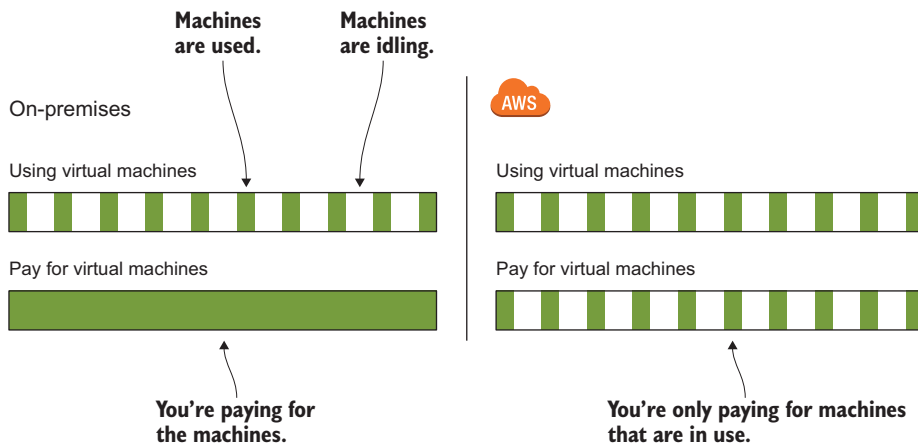


Figure 1.6 Making use of the pay-per-use price model of virtual machines