

LBRIS

We know
books

*Amazon Web Services in
Action, Second Edition*

MICHAEL WITTIG

ANDREAS WITTIG

FOREWORD BY BEN WHALEY



MANNING
Shelter Island

brief contents

PART 1	GETTING STARTED	1
	1 ■ What is Amazon Web Services? 3	
	2 ■ A simple example: WordPress in five minutes 36	
PART 2	BUILDING VIRTUAL INFRASTRUCTURE CONSISTING OF COMPUTERS AND NETWORKING	57
	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	
PART 3	STORING DATA IN THE CLOUD	233
	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 STARTED1**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 NETWORKING57

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

LBRIS | We know

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 CLOUD233

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

Part 1

Getting started

Have you watched a blockbuster on Netflix, bought a gadget on 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%.¹ Another impressive number: AWS reported net sales of \$4.1 billion USD for the quarter ending in June 2017.² 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.³ Being recognized as a leader attests AWS's high speed and high quality of innovation.

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

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

³ 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?

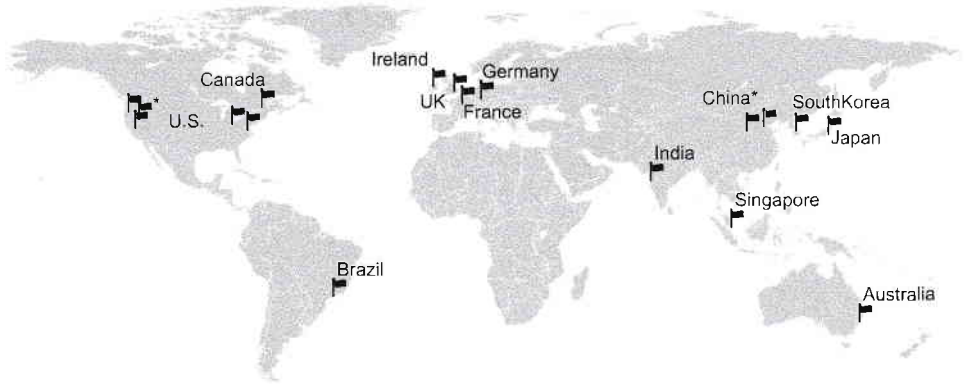
This chapter covers

- 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.

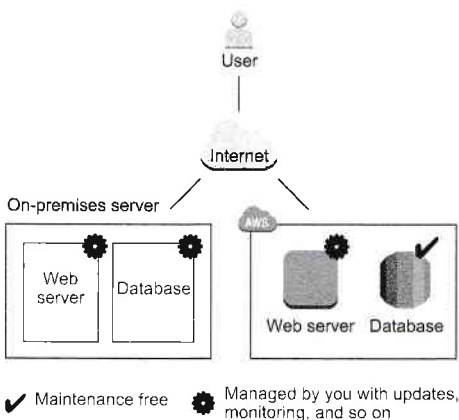


Figure 1.2 Running a web shop on-premises vs. on AWS