
contents

[Front matter](#)

[preface](#)

[acknowledgments](#)

[about this book](#)

[about the authors](#)

[about the cover illustration](#)

Part 1. [Getting started](#)

[1 What is Amazon Web Services?](#)

1.1 What is Amazon Web Services (AWS)?

1.2 What can you do with AWS?

Hosting a web shop

Running a Java EE application in your private network

Implementing a highly available system

Profiting from low costs for batch processing infrastructure

1.3 How you can benefit from using AWS

Innovative and fast-growing platform

Services solve common problems

Enabling automation

Flexible capacity (scalability)

Built for failure (reliability)

Reducing time to market

Benefiting from economies of scale

Global infrastructure

Professional partner

1.4 How much does it cost?

Free Tier

Billing example

Pay-per-use opportunities

1.5 Comparing alternatives

1.6 [Exploring AWS services](#)

1.7 [Interacting with AWS](#)

[Management Console](#)

[Command-line interface](#)

[SDKs](#)

[Blueprints](#)

1.8 [Creating an AWS account](#)

[Signing up](#)

[Signing in](#)

1.9 [Creating a budget alert to keep track of your AWS bill](#)

2 [A simple example: WordPress in 15 minutes](#)

2.1 [Creating your infrastructure](#)

2.2 [Exploring your infrastructure](#)

[Virtual machines](#)

[Load balancer](#)

[MySQL database](#)

[Network filesystem](#)

2.3 [How much does it cost?](#)

2.4 [Deleting your infrastructure](#)

Part 2. [Building virtual infrastructure consisting of computers and networking](#)

3 [Using virtual machines: EC2](#)

3.1 Exploring a virtual machine

Launching a virtual machine

Connecting to your virtual machine

Installing and running software manually

3.2 Monitoring and debugging a virtual machine

Showing logs from a virtual machine

Monitoring the load of a virtual machine

3.3 Shutting down a virtual machine

3.4 Changing the size of a virtual machine

3.5 Starting a virtual machine in another data center

3.6 Allocating a public IP address

3.7 Adding an additional network interface to a virtual machine

3.8 Optimizing costs for virtual machines

Commit to usage, get a discount

Taking advantage of spare compute capacity

4 Programming your infrastructure: The command line, SDKs, and CloudFormation

4.1 Automation and the DevOps movement

Why should you automate?

4.2 Using the command-line interface

Installing the CLI

Configuring the CLI

Using the CLI

Automating with the CLI

4.3 Programming with the SDK

Controlling virtual machines with SDK: nodecc

How nodecc creates a virtual machine

How nodecc lists virtual machines and shows virtual machine details

How nodecc terminates a virtual machine

4.4 Infrastructure as Code

Inventing an infrastructure language: JIML

4.5 Using AWS CloudFormation to start a virtual machine

Anatomy of a CloudFormation template

Creating your first template

Updating infrastructure using CloudFormation

5 Securing your system: IAM, security groups, and VPC

5.1 Who's responsible for security?

5.2 Keeping the operating system up-to-date

5.3 Securing your AWS account

Securing your AWS account's root user

AWS Identity and Access Management (IAM)

Defining permissions with an IAM identity policy

Users for authentication and groups to organize users

Authenticating AWS resources with roles

5.4 Controlling network traffic to and from your virtual machine

Controlling traffic to virtual machines with security groups

Allowing ICMP traffic

Allowing HTTP traffic

Allowing HTTP traffic from a specific source IP address

Allowing HTTP traffic from a source security group

5.5 Creating a private network in the cloud: Amazon Virtual Private Cloud (VPC)

Creating the VPC and an internet gateway (IGW)

Defining the public proxy subnet

Adding the private backend subnet

Launching virtual machines in the subnets

Accessing the internet from private subnets via a NAT gateway

6 Automating operational tasks with Lambda

6.1 Executing your code with AWS Lambda

What is serverless?

Running your code on AWS Lambda

Comparing AWS Lambda with virtual machines (Amazon EC2)

6.2 Building a website health check with AWS Lambda

Creating a Lambda function

Use CloudWatch to search through your Lambda function's logs

Monitoring a Lambda function with CloudWatch metrics and alarms

Accessing endpoints within a VPC

6.3 Adding a tag containing the owner of an EC2 instance automatically

Event-driven: Subscribing to EventBridge events

Implementing the Lambda function in Python

Setting up a Lambda function with the Serverless Application Model (SAM)

Authorizing a Lambda function to use other AWS services with an

IAM role

Deploying a Lambda function with SAM

6.4 **What else can you do with AWS Lambda?**

What are the limitations of AWS Lambda?

Effects of the serverless pricing model

Use case: Web application

Use case: Data processing

Use case: IoT backend

Part 3. **Storing data in the cloud**

7 **Storing your objects: S3**

7.1 [What is an object store?](#)

7.2 [Amazon S3](#)

7.3 [Backing up your data on S3 with AWS CLI](#)

7.4 [Archiving objects to optimize costs](#)

7.5 [Storing objects programmatically](#)

[Setting up an S3 bucket](#)

[Installing a web application that uses S3](#)

[Reviewing code access S3 with SDK](#)

7.6 [Using S3 for static web hosting](#)

[Creating a bucket and uploading a static website](#)

[Configuring a bucket for static web hosting](#)

[Accessing a website hosted on S3](#)

7.7 [Protecting data from unauthorized access](#)

7.8 [Optimizing performance](#)

8 [Storing data on hard drives: EBS and instance store](#)

8.1 Elastic Block Store (EBS): Persistent block-level storage attached over the network

Creating an EBS volume and attaching it to your EC2 instance

Using EBS

Tweaking performance

Backing up your data with EBS snapshots

8.2 Instance store: Temporary block-level storage

Using an instance store

Testing performance

Backing up your data

9 Sharing data volumes between machines: EFS

9.1 Creating a filesystem

Using CloudFormation to describe a filesystem

Pricing

9.2 Creating a mount target

9.3 Mounting the EFS filesystem on EC2 instances

9.4 Sharing files between EC2 instances

9.5 Tweaking performance

Performance mode

Throughput mode

Storage class affects performance

9.6 Backing up your data

10 Using a relational database service: RDS

10.1 Starting a MySQL database

Launching a WordPress platform with an RDS database

Exploring an RDS database instance with a MySQL engine

Pricing for Amazon RDS

10.2 Importing data into a database

10.3 Backing up and restoring your database

Configuring automated snapshots

Creating snapshots manually

Restoring a database

Copying a database to another region

Calculating the cost of snapshots

10.4 Controlling access to a database

Controlling access to the configuration of an RDS database

Controlling network access to an RDS database

Controlling data access

10.5 Building on a highly available database

Enabling high-availability deployment for an RDS database

10.6 Tweaking database performance

Increasing database resources

Using read replication to increase read performance

10.7 Monitoring a database

11.1 Creating a cache cluster

Minimal CloudFormation template

Test the Redis cluster

11.2 Cache deployment options

Memcached: Cluster

Redis: Single-node cluster

Redis: Cluster with cluster mode disabled

Redis: Cluster with cluster mode enabled

MemoryDB: Redis with persistence

11.3 Controlling cache access

Controlling access to the configuration

Controlling network access

Controlling cluster and data access

11.4 Installing the sample application Discourse with

CloudFormation

VPC: Network configuration

Cache: Security group, subnet group, cache cluster

Database: Security group, subnet group, database instance

Virtual machine: Security group, EC2 instance

Testing the CloudFormation template for Discourse

11.5 Monitoring a cache

Monitoring host-level metrics

Is my memory sufficient?

Is my Redis replication up-to-date?

11.6 Tweaking cache performance

Selecting the right cache node type

Selecting the right deployment option

Compressing your data

12 Programming for the NoSQL database service: DynamoDB

12.1 Programming a to-do application

12.2 Creating tables

Users are identified by a partition key

Tasks are identified by a partition key and sort key

12.3 Adding data

Adding a user

Adding a task

12.4 Retrieving data

Getting an item by key

Querying items by key and filter

Using global secondary indexes for more flexible queries

Creating and querying a global secondary index

Scanning and filtering all of your table's data

Eventually consistent data retrieval

12.5 Removing data

12.6 Modifying data

12.7 Recap primary key

Partition key

Partition key and sort key

12.8 SQL-like queries with PartiQL

12.9 DynamoDB Local

12.10 Operating DynamoDB

12.11 Scaling capacity and pricing

Capacity units

12.12 Networking

12.13 Comparing DynamoDB to RDS

12.14 NoSQL alternatives

Part 4. Architecting on AWS

13 Achieving high availability: Availability zones, autoscaling, and CloudWatch

13.1 Recovering from EC2 instance failure with CloudWatch

How does a CloudWatch alarm recover an EC2 instance?

13.2 Recovering from a data center outage with an Auto Scaling group

Availability zones: Groups of isolated data centers

Recovering a failed virtual machine to another availability zone with the help of autoscaling

Pitfall: Recovering network-attached storage

Pitfall: Network interface recovery

Insights into availability zones

13.3 Architecting for high availability

RTO and RPO comparison for a single EC2 instance

AWS services come with different high availability guarantees

14 Decoupling your infrastructure: Elastic Load Balancing and Simple Queue Service

14.1 Synchronous decoupling with load balancers

Setting up a load balancer with virtual machines

14.2 Asynchronous decoupling with message queues

Turning a synchronous process into an asynchronous one

Architecture of the URL2PNG application

Setting up a message queue

Producing messages programmatically

Consuming messages programmatically

Limitations of messaging with SQS

15 Automating deployment: CodeDeploy, CloudFormation, and Packer

15.1 In-place deployment with AWS CodeDeploy

15.2 Rolling update with AWS CloudFormation and user data

15.3 Deploying customized AMIs created by Packer

Tips and tricks for Packer and CloudFormation

15.4 Comparing approaches

16 Designing for fault tolerance

16.1 Using redundant EC2 instances to increase availability

Redundancy can remove a single point of failure

Redundancy requires decoupling

16.2 Considerations for making your code fault tolerant

Let it crash, but also retry.

Idempotent retry makes fault tolerance possible

16.3 Building a fault-tolerant web application: Imagery

The idempotent state machine

Implementing a fault-tolerant web service

Implementing a fault-tolerant worker to consume SQS messages

Deploying the application

17 Scaling up and down: Autoscaling and CloudWatch

17.1 Managing a dynamic EC2 instance pool

17.2 Using metrics or schedules to trigger scaling

Scaling based on a schedule

Scaling based on CloudWatch metrics

17.3 Decoupling your dynamic EC2 instance pool

Scaling a dynamic EC2 instance pool synchronously decoupled by a load balancer

Scaling a dynamic EC2 instances pool asynchronously decoupled by a queue

18 Building modern architectures for the cloud: ECS, Fargate, and App Runner

18.1 Why should you consider containers instead of virtual machines?

18.2 Comparing different options to run containers on AWS

18.3 The ECS basics: Cluster, service, task, and task definition

18.4 AWS Fargate: Running containers without managing a cluster of virtual machines

18.5 Walking through a cloud-native architecture: ECS, Fargate, and S3

index