

AWS Certified Solutions Architect - Associate Guide

The ultimate exam guide to AWS Solutions Architect certification



Packt

www.packt.com

Gabriel Ramirez and Stuart Scott

AWS Certified Solutions Architect – Associate Guide

The ultimate exam guide to AWS Solutions Architect certification

Gabriel Ramirez
Stuart Scott

Packt

BIRMINGHAM - MUMBAI

AWS Certified Solutions Architect – Associate Guide

Copyright © 2018 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Commissioning Editor: Vijin Boricha

Acquisition Editor: Heramb Bhavsar

Content Development Editor: Abhishek Jadhav

Technical Editor: Mohd Riyam Khan

Copy Editor: Safis Editing

Project Coordinator: Jagdish Prabhu

Proofreader: Safis Editing

Indexer: Tejal Daruwale Soni

Graphics: Tom Scaria

Production Coordinator: Nilesh Mohite

First published: October 2018

Production reference: 1311018

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-78913-066-9

www.packtpub.com

To my family, with love!

– Gabriel Ramirez

To my loving wife Lisa, for her support and encouragement throughout this book!

– Stuart Scott



mapt.io

Mapt is an online digital library that gives you full access to over 5,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Mapt is fully searchable
- Copy and paste, print, and bookmark content

Packt.com

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.packt.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packt.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 Packt books and eBooks.

Contributors

About the authors

Gabriel Ramirez is a passionate technologist with a broad experience in the Software Industry, he currently works as an Authorized Trainer for Amazon Web Services and Google Cloud.

He is holder of 9/9 AWS Certifications and does community work by organizing the AWS User Groups in Mexico. He can be found on LinkedIn at linkedin.com/in/gramirezm/.

Stuart Scott is the AWS content lead at Cloud Academy where he has created over 40 courses reaching tens of thousands of students. His content focuses heavily on cloud security and compliance, specifically on how to implement and configure AWS services to protect, monitor and secure customer data in an AWS environment.

He has written numerous cloud security blogs Cloud Academy and other AWS advanced technology partners. He has taken part in a series of cloud security webinars to share his knowledge and experience within the industry to help those looking to implement a secure and trusted environment.

In January 2016 Stuart was awarded 'Expert of the Year' from Experts Exchange for his knowledge share within cloud services to the community.

About the reviewer

Yohan Wadia is a client-focused evangelist and technologist with more than 8 years of experience in the cloud industry, focused on helping customers succeed with cloud adoption.

As a technical consultant, he provides guidance and implementation services to customers looking to leverage cloud computing through either Amazon Web Services, Windows Azure, or Google Cloud Platform by helping them come up with pragmatic solutions that make practical as well as business sense.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit authors.packtpub.com and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight 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.

Table of Contents

Preface	1
Chapter 1: Introducing Amazon Web Services	7
Technical requirements	8
Minimizing complexity	8
Conway's law	9
Cloud computing	10
Architecting for AWS	10
Cloud design principles	11
Cloud design patterns – CDP	28
AWS Cloud Adoption Framework – AWS CAF	29
AWS Well-Architected Framework – AWS WAF	30
Shared security model	31
Identity and Access Management	38
User creation	40
Designing an access structure	44
Create an administration group	45
Business case	47
Inline policies	53
IAM cross-account roles	57
Summary	61
Further reading	62
Chapter 2: AWS Global Infrastructure Overview	63
Technical requirements	64
Introducing AWS global infrastructure	64
Becoming a service company	64
Data centers	65
10,000-feet view	66
Regions	67
100,000-feet view	67
Latency	67
Compliance	67
Supported services	68
Cost	68
Connectivity	68
Endpoint access	68
Global CDN	69
Amazon CloudFront	69
Single region / multi-region patterns	69

Rationale	69
Active-active	70
Active-passive	70
Network-partitioning tolerance	70
Complexity	71
CloudFront	71
Data replication and redundancy with managed services	71
Exercise	72
Replicating tags	80
Replicating ACLs	81
Distributed nature of S3	84
Metadata replication	86
Encryption replication	89
Hosting a static website with S3 and CloudFront	91
Summary	105
Further reading	106
Chapter 3: Elasticity and Scalability Concepts	107
Technical requirements	108
Sources of failure	108
The cause	109
Dividing and conquering	110
Serial configuration	111
Parallel configuration	112
Reactive and proactive scalability	112
Horizontal scalability	113
Vertical scalability	113
Exercise	114
Virtualization technologies	115
LAMP installation	120
Scaling the web server	123
Resiliency	123
EC2 persistence model	125
Disaster recovery	127
Cascading deletion	131
Bootstrapping	132
Scaling the compute layer	135
Proactive scalability	136
Scaling a database server	136
Summary	138
Further reading	138
Chapter 4: Hybrid Cloud Architectures	139
Effective migration to the cloud	139
Extending your data center	142

All in the cloud	142
VPC	142
Tenancy	143
Sizing	143
The default VPC	144
Public traffic	149
Private traffic	152
Security groups	155
Creating a security group	156
Chaining security groups	156
Bastion host	157
Hybrid deployment	159
Software VPNs	159
Static hardware VPNs	160
Dynamic hardware VPNs	161
Direct Connect (DX)	161
Storage gateway use cases	162
Network filesystems with file gateways	162
Block storage iSCSI with volume gateway – stored	162
Block storage iSCSI with volume gateway – cached	163
Virtual tape library iSCSI with a tape gateway	163
The Database Migration Service	163
Homogeneous migration	164
The AWS Schema Conversion tool	164
Heterogeneous migrations	165
Summary	166
Further reading	166
Chapter 5: Resilient Patterns	167
 Technical requirements	167
 Route 53	167
Health checks	169
Record types	173
 Summary	178
 Further reading	178
Chapter 6: Event Driven and Stateless Architectures	179
 Technical requirements	179
 Web application hosting	180
Route 53	180
 Serverless application architecture	181
 Streaming data architecture	183
 Summary	184
 Further reading	184
Chapter 7: Integrating Application Services	185
 Technical requirements	186

SQS as a reliable broker	187
Asynchrony	187
Creating a queue	189
Security	189
Durability	190
Message delivery	193
Message reception	196
Messaging patterns	197
Managing 1:N communications with SNS	198
Subscriber	199
Fanout	200
Authenticating your web and mobile apps with Cognito	204
Cognito user pools	205
Federated identities	206
API Gateway integration	208
Request flow	215
WebSockets in AWS	219
AWS IoT	219
AWS AppSync	219
Web app demo	219
Summary	220
Further reading	221
Chapter 8: Disaster Recovery Strategies	222
 Technical requirements	222
 Availability metrics	222
The business perspective	223
Business impact analysis	223
Recovery Time Objective (RTO)	224
Recovery Point Objective (RPO)	224
Availability monitoring	225
 Backup and restore	225
Preparation phase	226
In the case of a disaster	226
Trade-offs	227
 Pilot light	227
The preparation phase	228
In the case of a disaster	229
Trade-offs	229
 Warm standby	230
The preparation phase	231
In the case of a disaster	231
Trade-offs	231
 Multi-site active-active	232
The preparation phase	233
In the case of a disaster	234

Trade-offs	235
Best practices	235
Summary	235
Further reading	236
Chapter 9: Storage Options	237
Technical requirements	238
Relational databases	238
RDS	238
Managed capabilities	238
Instances	239
Parameter groups	240
Option groups	240
Snapshots	240
Events	241
Multi-AZ	241
Read replicas	242
Caching	243
Object storage	244
Simple storage service	244
Data organization	245
Integrity	246
Availability	246
Cost dimensions	247
Reducing cost	247
Durability	248
Maximum durability	248
Limited durability	248
Use cases	249
Consistency	249
Storage optimization	250
Creating objects from the CLI	250
Copy an existing object	251
Using a lifecycle policy	251
Lifecycle policies	252
Archiving with Glacier	257
Retrieval options	257
Workflow	258
NoSQL	260
DynamoDB	260
Control plane	260
Managed capabilities	267
Consistency	267
Local secondary index	268
Global secondary index	270
DynamoDB Streams	272
Global tables	272
Summary	273

Further reading	273
Chapter 10: Matching Supply and Demand	274
Technical requirements	274
Elastic Load Balancing	275
Classic Load Balancer – CLB	275
Network Load Balancer – NLB	276
Application Load Balancer – ALB	278
Creating an Application Load Balancer	280
ELB attributes	284
Stateless versus stateful	284
Internet-facing versus internal-facing	285
TCP passthrough	285
Cross-zone load balancing	285
Connection draining	285
AWS Auto Scaling	286
Alternate flow	290
Create a launch configuration	291
Auto Scaling groups	293
Resiliency	296
Summary	298
Further reading	299
Chapter 11: Introducing Amazon Elastic MapReduce	300
Technical requirements	301
Clustering in AWS	301
High performance computing	302
CfnCluster	302
Enhanced networking	305
Jumbo frames	305
Placement groups	307
Creating a placement group	307
Benchmarking	309
Elastic MapReduce	310
MapReduce	311
Analyzing a public dataset	312
Summary	320
Further reading	321
Chapter 12: Web Scale Applications	322
Technical requirements	322
AWS Lambda	323
Summary	328
Further reading	328
Chapter 13: Understanding Access Control	329

Technical requirements	329
Authentication, authorization, and access control	330
Authentication	330
Authorization	331
Access control	331
Authenticating via access control methods	332
Usernames and passwords	332
Multi-factor authentication	332
Programmatic access	337
Key pairs	338
IAM roles	341
Cross-account roles	342
Web identity and SAML federation	343
Federation of access	343
Web identity federation	343
SAML 2.0 federation	344
IAM authorization	346
Users	346
Groups	346
Roles	347
Identity-based policies	347
Managed policies versus inline policies	350
Writing policies from scratch by using a JSON policy editor	351
Using the visual editor within IAM	351
Copying an existing managed policy	352
Inline policies	353
Summary	353
Further reading	353
Chapter 14: Encryption and Key Management	354
 Technical requirements	355
 An overview of encryption	355
Symmetric key cryptography	356
Asymmetric key cryptography	356
 EBS encryption	357
Encrypting a new EBS volume	358
Encrypting a new EBS volume during the launch of a new EC2 instance	360
Encrypting an existing EBS volume	360
 Amazon S3 encryption	363
Server-side encryption with S3 managed keys (SSE-S3)	364
Server-side encryption with KMS managed keys (SSE-KMS)	366
Server-side encryption with customer managed keys (SSE-C)	368
Client-side encryption with KMS managed keys (CSE-KMS)	370
Client-side encryption with KMS managed keys (CSE-C)	372
 RDS encryption	374

How to enable encryption	374
Steps to encrypt an existing database	379
Key Management Service (KMS)	381
So, what is KMS?	381
Customer master keys	382
Data encryption keys (DEK)	383
Key policies	383
Grants	384
Key rotation	385
Manual key rotation	386
Summary	386
Further reading	387
Chapter 15: An Overview of Security and Compliance Services	388
 Technical requirements	389
 AWS CloudTrail	389
 Amazon Inspector	395
Installing the agent	397
Assessment templates, runs, and findings	398
 AWS Trusted Advisor	400
Yellow warning under service limits	402
Red warning under service limits	404
 AWS Systems Manager	405
Resource groups	408
Creating a resource group	408
Actions	411
Insights	413
Shared resource	415
 AWS Config	416
Configuration item	418
Configuration streams	419
Configuration history	419
Configuration snapshot	420
Configuration recorder	420
Config rules	421
Resource relationship	421
High-level process overview	422
 Summary	423
 Further reading	423
Chapter 16: AWS Security Best Practices	424
 Technical requirements	425
 Shared responsibility model	425
 Data protection	429
Using encryption at rest for sensitive data	429
Taking advantage of encryption features built into AWS services	430

Using encryption in transit for sensitive data	431
Protecting against unexpected data loss	432
Using S3 MFA delete to prevent accidental deletion	433
Using S3 lifecycle policies	433
Implementing S3 versioning to protect against unintended actions	433
Virtual Private Cloud	434
Using security groups to control access at an instance level	434
Using NACLs to control access at a subnet level	434
Implementing the rule of least privilege	435
Implementing layers in your VPC	435
Creating Flow Logs to obtain deeper analysis of network traffic	435
Identity and Access Management	435
Avoid sharing identities	436
Using MFA for privileged users	436
Using roles	436
Password policy	437
Assigning permissions to groups instead of to individual users	438
Rotating your access keys	438
Assigning permissions according to the rule of least privilege	438
Re-evaluating permissions and deleting accounts	439
Do not use the root account as an operational user	439
EC2 security	439
Implementing a patching strategy	440
Controlling access with security groups	440
Encrypting sensitive data on persistent storage	440
Harden the operating system	441
Using Bastion hosts to connect to your EC2 instances	441
Security services	442
Summary	444
Further reading	445
Chapter 17: Web Application Security	446
Technical requirements	446
AWS web application firewall	447
Conditions	447
Rules	454
Web ACL	455
Monitoring	457
AWS Shield	458
DDoS	459
Shield plans	459
AWS Firewall Manager	460
Before using AWS Firewall Manager	461
Amazon CloudFront security features	462
Summary	463

Further reading	463
Chapter 18: Cost Effective Resources	464
Technical requirements	465
Reserved Instances	465
Standard Reserved Instances	465
Convertible Reserved Instances	466
Billing and cost management	467
Billing alarms	469
Service level alarms	470
Billing reports	474
Cost Explorer	477
Reserved Instances recommendations	478
QuickSight visualization	479
Cost Allocation Tags	482
AWS Organizations	482
Summary	485
Further reading	486
Chapter 19: Working with Infrastructure as Code	487
Technical requirements	487
AWS CloudFormation	488
Template anatomy	491
Resources	491
Stack updates	492
Deletion policy	494
Outputs	494
Template reusability	495
Parameters	496
Mappings	500
Depends on	502
Helper scripts	504
Multi-tier web app	507
Best practices	509
Summary	509
Further reading	509
Chapter 20: Automation with AWS	510
Technical requirements	510
Incident Response	510
CloudWatch Logs Agent	511
CloudWatch Metric Filters	515
Summary	520
Further reading	520
Chapter 21: Introduction to the DevOps practice in AWS	521

Table of Contents

Technical requirements	521
CI / CD pipeline	521
AWS CodeDeploy	524
AppSpec file	527
Summary	536
Further reading	536
Chapter 22: Mock Test 1	537
Chapter 23: Mock Test 2	544
Assessment	565
<u>Another Book You May Enjoy</u>	584
Index	586

Preface

Amazon Web Services (AWS) is currently the leader in the public cloud market. With an increasing global interest in leveraging cloud infrastructure, the AWS Cloud from Amazon offers a cutting-edge platform for architecting, building, and deploying web-scale cloud applications.

As more the rate of cloud platform adoption increases, so does the need for cloud certification. The AWS Certified Solution Architect – Associate Guide is your one-stop solution to gaining certification. Once you have grasped what AWS and its prerequisites are, you will get insights into different types of AWS services such as Amazon S3, EC2, VPC, SNS, and more to get you prepared with core Amazon services. You will then move on to understanding how to design and deploy highly scalable applications. Finally, you will study security concepts along with the AWS best practices and mock papers to test your knowledge.

By the end of this book, you will not only be fully prepared to pass the AWS Certified Solutions Architect – Associate exam but also capable of building secure and reliable applications.

Who this book is for

The *AWS Certified Solutions Architect – Associate Guide* is for you if you are an IT professional or Solutions Architect wanting to pass the AWS Certified Solution Architect – Associate 2018 exam. This book is also for developers looking to start building scalable applications on AWS.

What this book covers

Chapter 1, *Introducing Amazon Web Services*, in this chapter, the Amazon Web Services provides a very rich feature set of services and this chapter will take the readers through fundamentals concepts of AWS concepts. This will include information about what AWS Cloud is, how it enables large organizations and small start-ups to leverage enterprise class infrastructure.

Chapter 2, *AWS Global Infrastructure Overview*, this chapter will teach the readers about the AWS Global infrastructure, the service endpoints and partitions, availability zones, regions, edge locations and how they interact with high availability patterns and resilient designs. This chapter will also cover replication and synchronization of data at a global scale with a special focus on security.

Chapter 3, *Elasticity and Scalability Concepts*, this chapter will teach the readers how to match capacity and demand, design cost efficient solutions and understand how these two concepts play a role in Cloud Architecture. We'll focus on Demand, Buffer and Time based approaches, automation and serverless implementations.

Chapter 4, *Hybrid Cloud Architectures*, this chapter will teach the readers how to integrate cloud services, deploy new applications and interconnect and extend existing infrastructure to the cloud. Use application services as message queues, publisher subscriber, API Gateway and lambda as a bridge as an adapter.

Chapter 5, *Resilient Patterns*, this chapter will teach the readers how to avoid complete service failures by absorbing the operational impact of a service failure by loosely coupling components and services. To inject failure in our systems to make them fault tolerant and exposing failure paths. To design reactive autonomous monitoring systems in the cloud.

Chapter 6, *Event Driven and Stateless Architectures*, this chapter will teach the readers how to design workflows like ETL and image processing leveraging storage and processing using lambda. You will understand the pros and cons about maintaining servers and using PaaS and abstract services like S3 and DynamoDB.

Chapter 7, *Integrating Application Services*, the chapter will teach the readers how to integrate services like authentication, mobile backends, messaging and persistence to their apps. You will use Backend as a Service (BaaS) to decouple front end and middleware and to use 3rd party service providers.

Chapter 8, *Disaster Recovery Strategies*, the chapter will teach the readers what are the main patterns in DR strategies using the cloud. The reader will learn to implement successfully backup and restore, use pilot light and multi site active - active scenarios. You will be guided on how to implement a full DR exercise in a hybrid environment.

Chapter 9, *Storage Options*, the chapter will teach the readers the different storage options available, to evaluate durability, cost, performance size and management tasks of each one. You will compare hot and cold solutions and examples of EBS, S3, Glacier, RedShift and DynamoDB.

Chapter 10, *Matching Supply and Demand*, the chapter will teach the readers how to optimize for cost, use optimal resources on every layer. Work with AutoScaling and resize RDS databases with CloudWatch alarms.

Chapter 11, *Introducing Amazon Elastic MapReduce*, this chapter will teach the readers get insight about Elastic Map Reduce, the use cases and how to design clusters for High Performance Computing on EC2. Profiling your instances to maximize throughput and optimize network resources.

Chapter 12, *Web Scale Applications*, the chapter will teach the readers how to build massive applications that reach millions of users, with high levels of concurrency. Offload your backends with cache technologies like CloudFront and ElasticCache and NoSQL datastores.

Chapter 13, *Understanding Access Control*, the chapter will teach the readers to get familiar about the main security objectives, use granular control access for your users and applications. Learn about the security best practices and permission management through IAM.

Chapter 14, *Encryption and Key Management*, the chapter will teach the readers how encryption works in the cloud, use custom means to encrypt sensitive information and leverage encryption mechanisms from different AWS services and how to integrate with the Marketplace solutions to design robust security schemes and be compliant with several international standards, regulations and frameworks.

Chapter 15, *An Overview of Security and Compliance Services*, the chapter will provide an overview of some of the key AWS services that are used to secure, protect and govern data and resources within an AWS environment. It will define what each service is used for and the components that are used within each.

Chapter 16, *AWS Security Best Practices*, the chapter will teach the readers how to implement the AWS security reference model and get an in depth analysis of every service and configuration used to protect your application and data.

Chapter 17, *Web Application Security*, the chapter will teach the readers how to protect web applications, take a proactive standpoint for application design. You will learn how to avoid Cross Site Scripting, Man in the middle attacks and data integrity loss.

Chapter 18, *Cost Effective Resources*, the chapter will teach the readers how to design cost efficient resources and optimize services to improve ROI. Build custom cost reports with custom tags and use consolidated billing with multiple accounts. Create budgets and alarms to avoid unexpected charges.

Chapter 19, *Working with Infrastructure as Code*, the chapter will teach the readers how manage infrastructure using a set of tools, practices and thinking as software to gain consistency, flexibility, reusability and many advantages of this paradigm. We will work with CloudFormation and introduce you to OpsWorks, also will talk about many of the tools available in the industry that will help you manage configurations and infrastructure.

Chapter 20, *Automation with AWS*, the chapter will continue on the previous one demonstrating how automation help industries achieve more with less, how deployment strategies can help in consistency, availability and continuity of business. We show how to automate response to application logs, CloudTrail and Configuration Changes through AWS Config.

Chapter 21, *Introduction to DevOps in AWS*, the chapter will explain the principles, processes, toolchain and culture behind this practice. We'll take a holistic approach to apply SCM, Continuous Integration (CI) and Continuous Delivery (CD).

Chapter 22, *Mock Test 1*, in this chapter, readers will get the hands-on experience of the real time certification exam which will cover questions from the above stated services and which will make them confident about clearing the associate exam with the help of important tips and tricks.

Chapter 23, *Mock Test 2*, in this chapter, readers will get the hands-on experience of the real time certification exam which will cover questions from the above stated services and which will make them confident about clearing the associate exam with the help of important tips and tricks.

To get the most out of this book

You should have access to an AWS account.

The detailed requirement for each chapter can be found in the *Technical requirement* section of the chapters.

Download the example code files

You can download the example code files for this book from your account at www.packt.com. If you purchased this book elsewhere, you can visit www.packt.com/support and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at www.packt.com.
2. Select the **SUPPORT** tab.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at <https://github.com/PacktPublishing/AWS-Certified-Solutions-Architect-Associate-Guide>. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at <https://github.com/PacktPublishing/>. Check them out!

Conventions used

There are a number of text conventions used throughout this book.

CodeInText: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "The public key is installed in the `~/.ssh/authorized_keys` in the filesystem of the instance."

A block of code is set as follows:

```
{  
    "Tenancy": "default",  
    "GroupName": "",  
    "AvailabilityZone": "us-east-1a"  
}
```

Any command-line input or output is written as follows:

```
mkdir webApp && cd $_
```

Bold: Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "In the EC2 console choose **Launch Instance**."



Warnings or important notes appear like this.



Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at customercare@packtpub.com.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit www.packt.com/submit-errata, selecting your book, clicking on the Errata Submission Form link, and entering the details.

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 copyright@packt.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 authors.packtpub.com.

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 Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit packt.com.

1

Introducing Amazon Web Services

Welcome to the journey of becoming an **Amazon Web Services (AWS)** solutions architect. A path full of challenges, but also a path full of knowledge awaits you. To begin, I'd like to start by defining the role of a solutions architect in the software-engineering context. Architecture has a lot to do with technology, but it also has a lot to do with everything else; it is a discipline responsible for the nonfunctional requirements, and a model to design the **Quality of Service (QoS)** of the information systems.

Architecture is about finding the right balance and the midpoint of every circumstance. It is about understanding the environment in which problems are created, involving the people, the processes, the organizational culture, the business capabilities, and any external drivers that can influence the success of a project.

We will learn that part of our role as solutions architects is to evaluate several trade-offs, manage the essential complexity of things, their technical evolution, and the inherent entropy of complex systems.

The following topics will be covered in this chapter:

- Understanding cloud computing
- Cloud design patterns and principles
- Shared security model
- Identity and access management

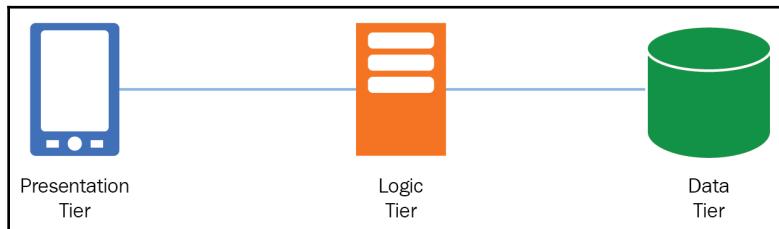
Technical requirements

Solution scripts are available in the book's repositories at the following URLs, if you get stuck with the examples:

- <https://github.com/PacktPublishing/AWS-Certified-Solutions-Architect-Associate-Guide>
- <https://github.com/gabanox/Certified-Solution-Architect-Associate-Guide>

Minimizing complexity

A widely used strategy to solve difficult problems is to use **functional decomposition**, that is, breaking a complex system or process into manageable parts; a pattern for this is the **multilayered architecture**, also known as the **n-tier architecture**, by which we decompose big systems into logical layers focused only on one responsibility, leveraging characteristics such as scalability, flexibility, reusability, and many other benefits. The three-layer architecture is a popular pattern used to decompose monolithic applications and design distributed systems by isolating their functions into three different services:

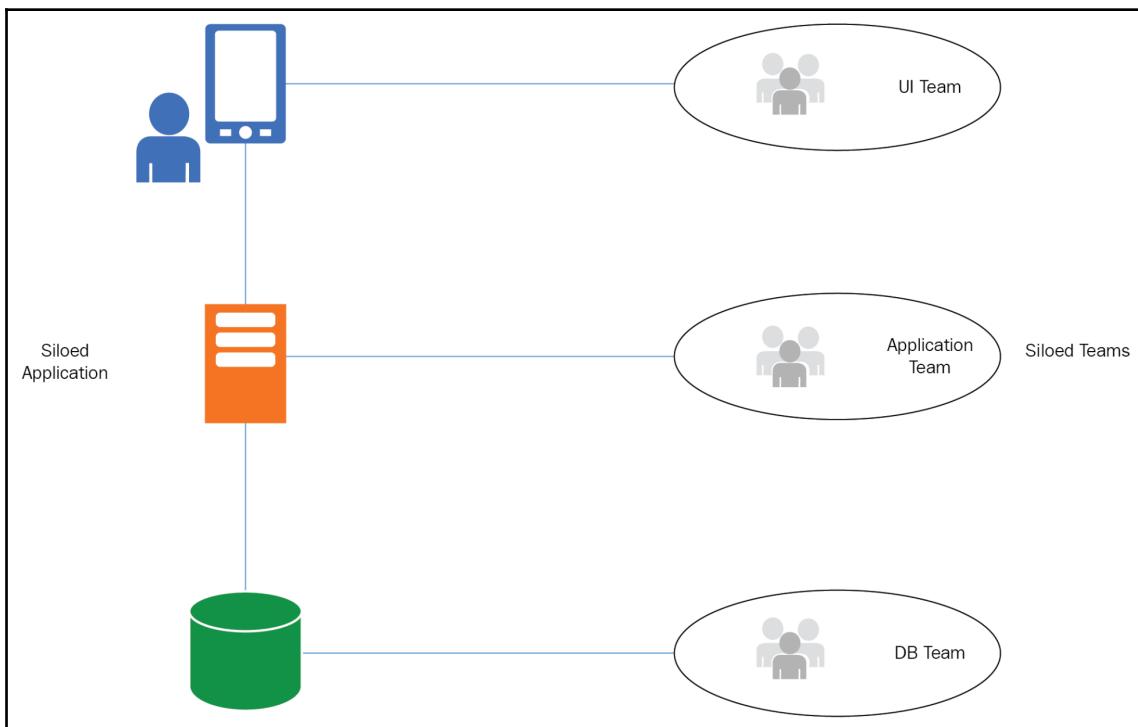


- **Presentation Tier:** This represents the component responsible for the user interface, in which user actions and events are generated via a web page, a mobile application, and so on.
- **Logic Tier:** This is the middleware, the middle tier where the business logic is found. This tier can be implemented via **web servers** or **application servers**; here, every presentation tier event gets translated into service methods and business functions.
- **Data Tier:** Persistence means will interact with the logic tier to maintain user state and behavior; this is the central repository of data for the application. Examples of this are **database management systems (DBMS)** or **distributed memory-caching systems**.

Conway's law

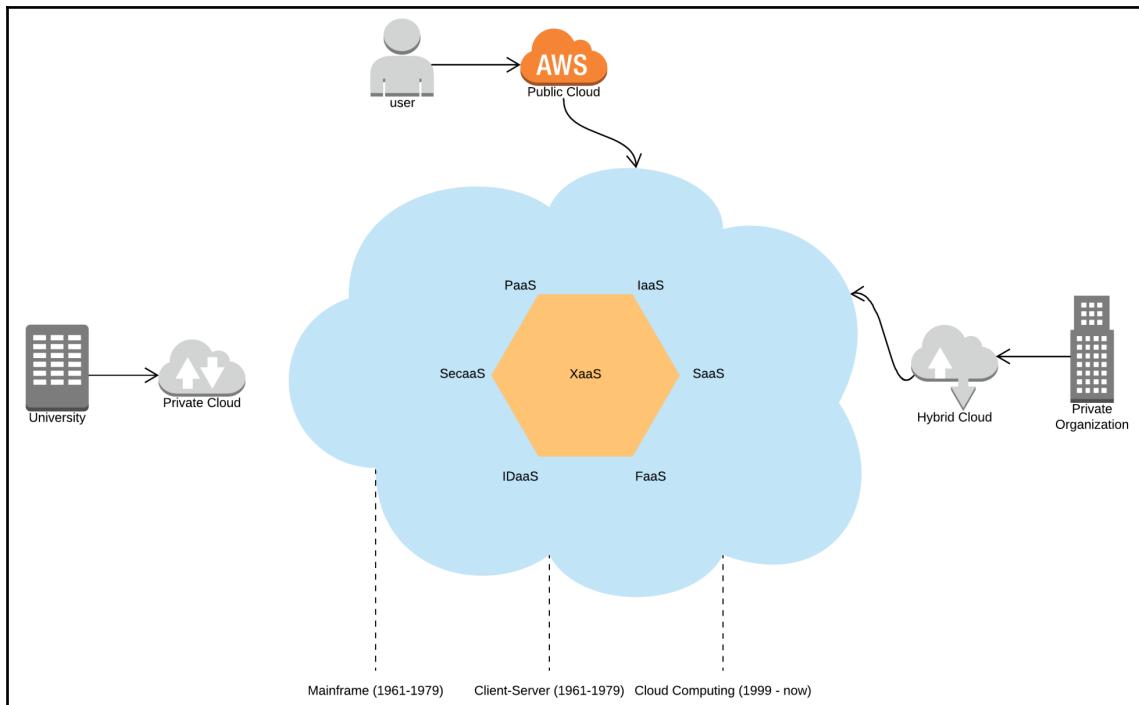
"organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations."

This sentence shows the relevance of the way people organize to develop systems, and how this impacts every design decision we make. We will get into depth in the later chapters discussing microservices architectures, about how we can decouple and remove the barriers that prevent systems from evolving. Bear in mind that this book will show you a new way of systems thinking, and with AWS you have the tools to solve any kind of problem and create very sophisticated solutions.



Cloud computing

Cloud computing is a service model based on large pools of resources exposed through web interfaces, with the objective being to provide shareable, elastic, and secure services on demand with low cost and high flexibility:



Architecting for AWS

Designing cloud-based architectures, carries a different approach than traditional solutions, because physical hardware and infrastructure are now treated as software. This brings many benefits, such as reusability, high cohesion, a uniform service interface, and flexible operations.

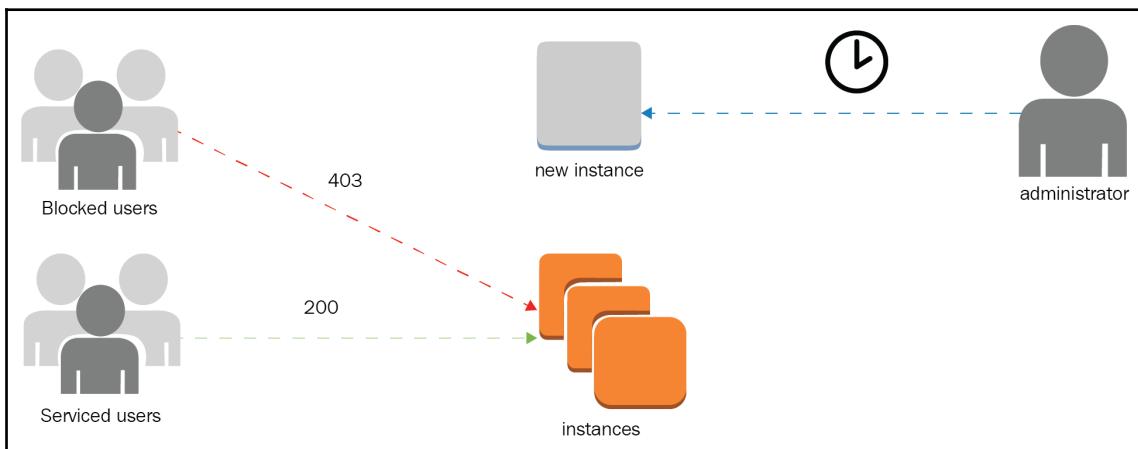
It's easy to make use of on-demand resources when they are needed to modify its attributes in a matter of minutes. We can also provision complex structures declaratively and adapt services to the demand patterns of our users. In this chapter, we will be discussing the design principles that will make the best use of AWS.

Cloud design principles

These principles confirm the fundamental pillars on which well-architected and well-designed systems must be made:

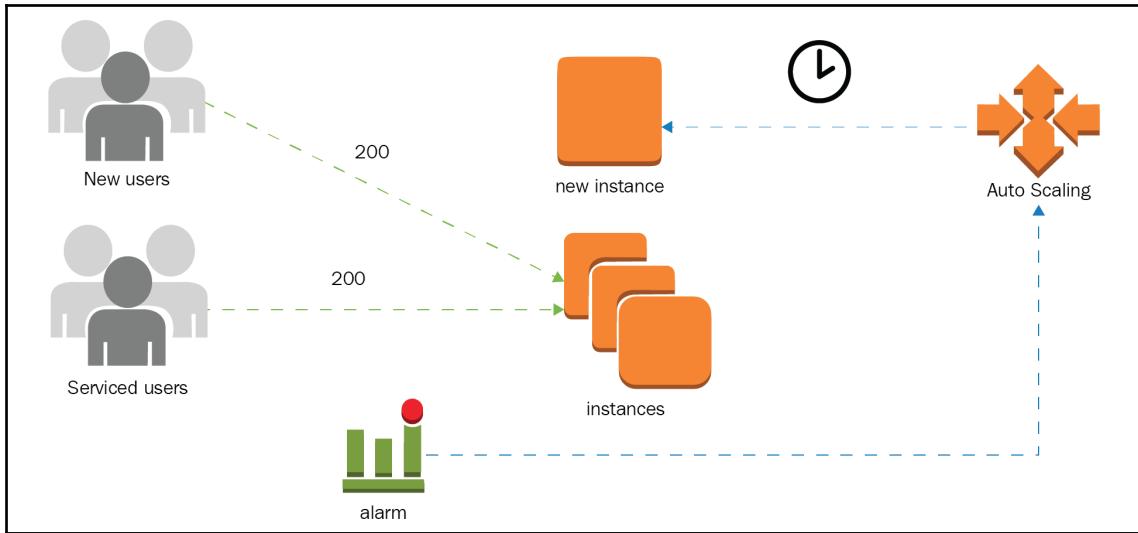
- **Enable scalability:**

- **Antipattern:** Manual operation to aggregate capacity reactively and not proactively. Passive detection of failures and service limits can result in downtimes for applications and is prone to human errors due to limited reaction timespans:



From the diagram, we can see that instances take time to be fully usable, and the process is human-dependent.

- **Best practice:** The elastic nature of AWS services makes it possible to manage changes in demand and adapt to the consumer patterns with the possibility to reach global audiences. When a resource is not elastic, it is possible to use Auto Scaling or serverless approaches:



Auto Scaling automatically spins up instances to compensate for demand.

- **Automate your environment:**

- **Antipattern:** Ignoring configuration changes and the lack of a **configuration management database (CMDB)** can result in erratic behavior, visibility loss, and have a high impact on critical production systems. The absence of robust monitoring solutions results in fragile systems and slow responses to change requests compromising security and the system's stability:

The screenshot shows the AWS Config console for an EC2 VPC named 'vpc-'. The timeline at the top displays three events: a creation event on August 22nd at 10:14:50 AM, a change event on August 22nd at 4:08:37 PM, and a second change event on October 8th at 12:04:36 PM. Below the timeline, the 'Configuration Details' section provides resource metadata, including its VPC ID, state, and CIDR range. The 'Relationships' and 'Changes' sections are also visible.

EC2 VPC vpc-

on October 31, 2018 12:56:28 PM Greenwich Mean Time (UTC+00:00)

Manage resource [Edit](#)

Configuration timeline [Compliance timeline](#)

22nd August 2018 10:14:50 AM Events 22nd August 2018 4:08:37 PM Changes 08th October 2018 12:04:36 PM Change Event Now View Details

▼ Configuration Details

Amazon Resource Name	VPC ID
Resource type	AWS::EC2::VPC
Resource ID	State
Resource name	available
Availability zone	VPC CIDR
Created on	10.0.0.0/16
Tags (1)	DHCP Options Set
	Default VPC
	Instance tenancy

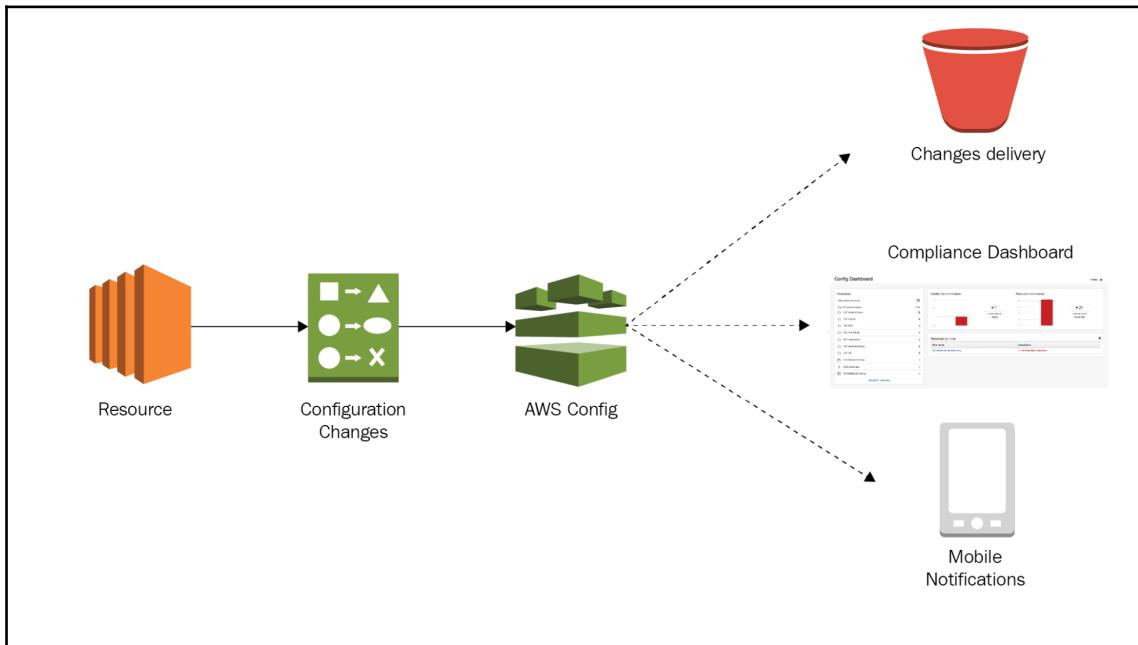
Resource type: AWS::EC2::VPC
Resource ID: vpc-
Resource name: null
Availability zone: Multiple Availability Zones
Created on: Not available
Tags (1): Name:CA-D...

▼ Relationships [Edit](#)

▼ Changes [Edit](#) (1)

AWS Config records every change in the resources and provides a unique source of truth for configuration changes.

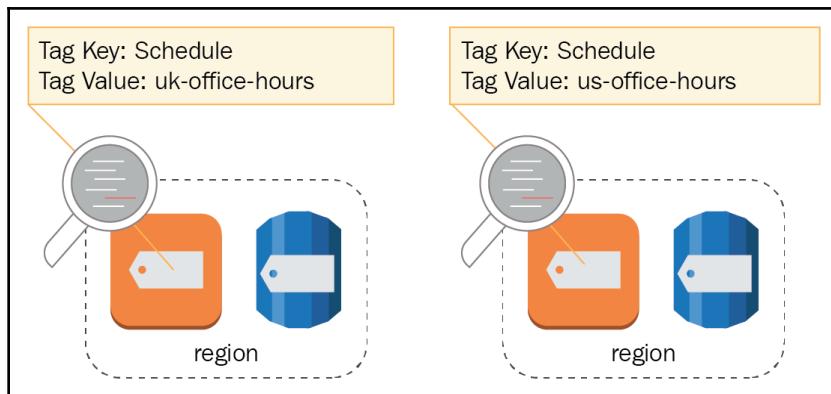
- **Best practice:** Relying on automation, from scripts to specialized monitoring services, will help us to gain reliability and make every cloud operation secure and consistent. The early detection of failures and deviation from normal parameters will support in the process of fixing bugs and avoiding issues to become risks. It is possible to establish a monitoring strategy that accounts for every layer of our systems. Artificial intelligence can be used for analyzing the stream of changes in real time and reacting to these events as they occur, facilitating agile operations in the cloud:



Config can be used to durably store configuration changes for later inspection, react in real time with push notifications, and visualize real-time operations dashboards.

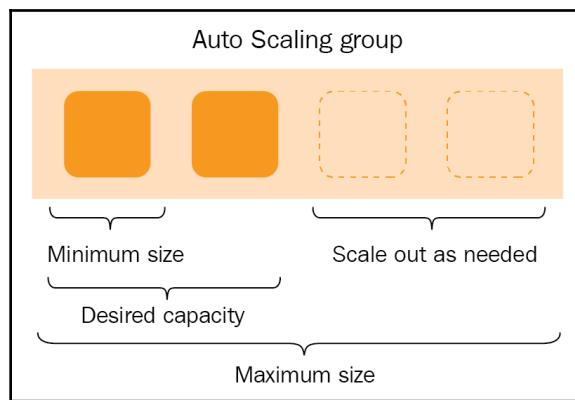
- **Use disposable resources:**

- **Antipattern:** Running instances with low utilization or over capacity can result in higher costs. The poor understanding of every service feature and capability can result in higher expenses, lower performance, and administration overhead. Maybe you are not using the right tool for the job. Immutable infrastructure is a determinant aspect of using disposable resources, so you can create and replace components in a declarative way:



Tagging will help you gain control, and provide means to orchestrate change management. The previous diagram shows how tagging can help to discover compute resources to stop and start only in office hours for different regions.

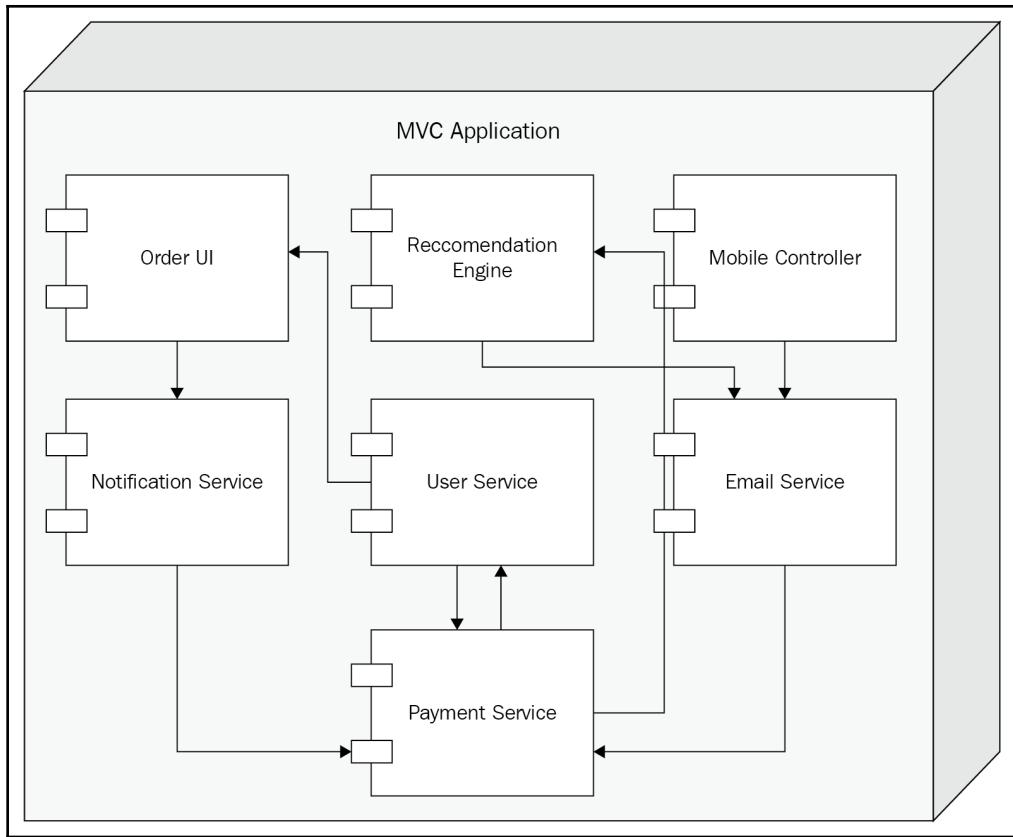
- **Best practice:** Using reserved instances promotes a fast **Return on Investment (ROI)**. Running instances only when they are needed or using services such as Auto Scaling and AWS Lambda will strengthen usage only when needed, thus optimizing costs and operations. Practices such as **Infrastructure as Code (IaC)** give us the ability to create full-scale environments and perform production-level testing. When tests are over, you can tear down the testing environment:



Auto Scaling lets the customer specify the scaling needs without incurring additional costs.

- **Loosely couple your components:**

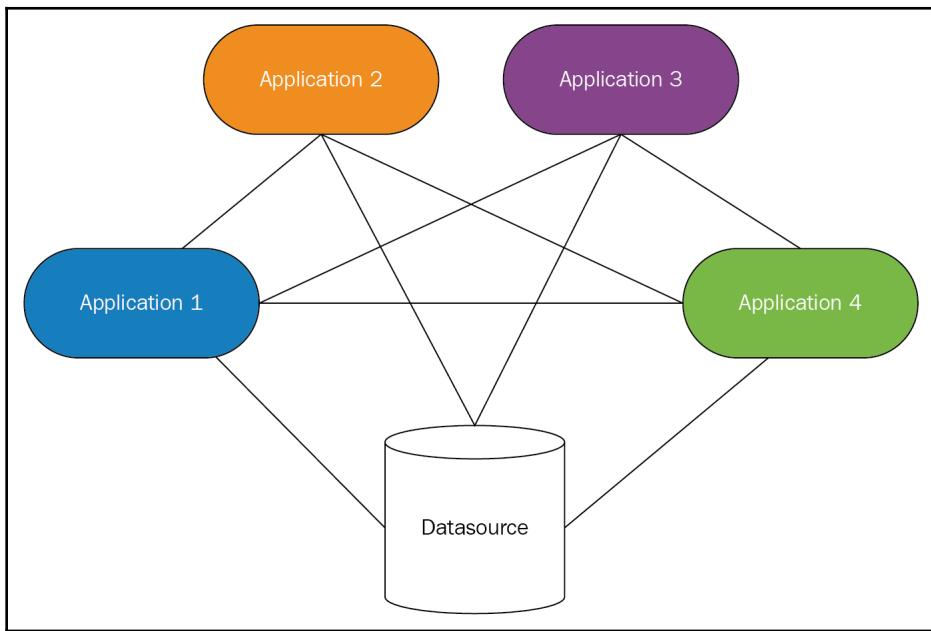
- **Antipattern:** Tightly coupled systems avoid scalability and create a high degree of dependencies at the component and service levels. Systems become more rigid, less portable, and it is very complicated to introduce changes and improvements. Coupling not only happens at the software or infrastructure levels, but also with service providers by using proprietary solutions or services that create dependencies with a brand forcing us to consume their products with the inherent restrictions of the maker:



Software changes constantly, and we need to keep ourselves updated to avoid the erosion of the operating systems and technology stacks.

- **Best practice:** Using indirection levels will avoid direct communications and less state sharing between components, thus keeping low coupling and high cohesion. Extracting configuration data and exposing it as services will permit evolution, flexibility, and adaptability to changes in the future.

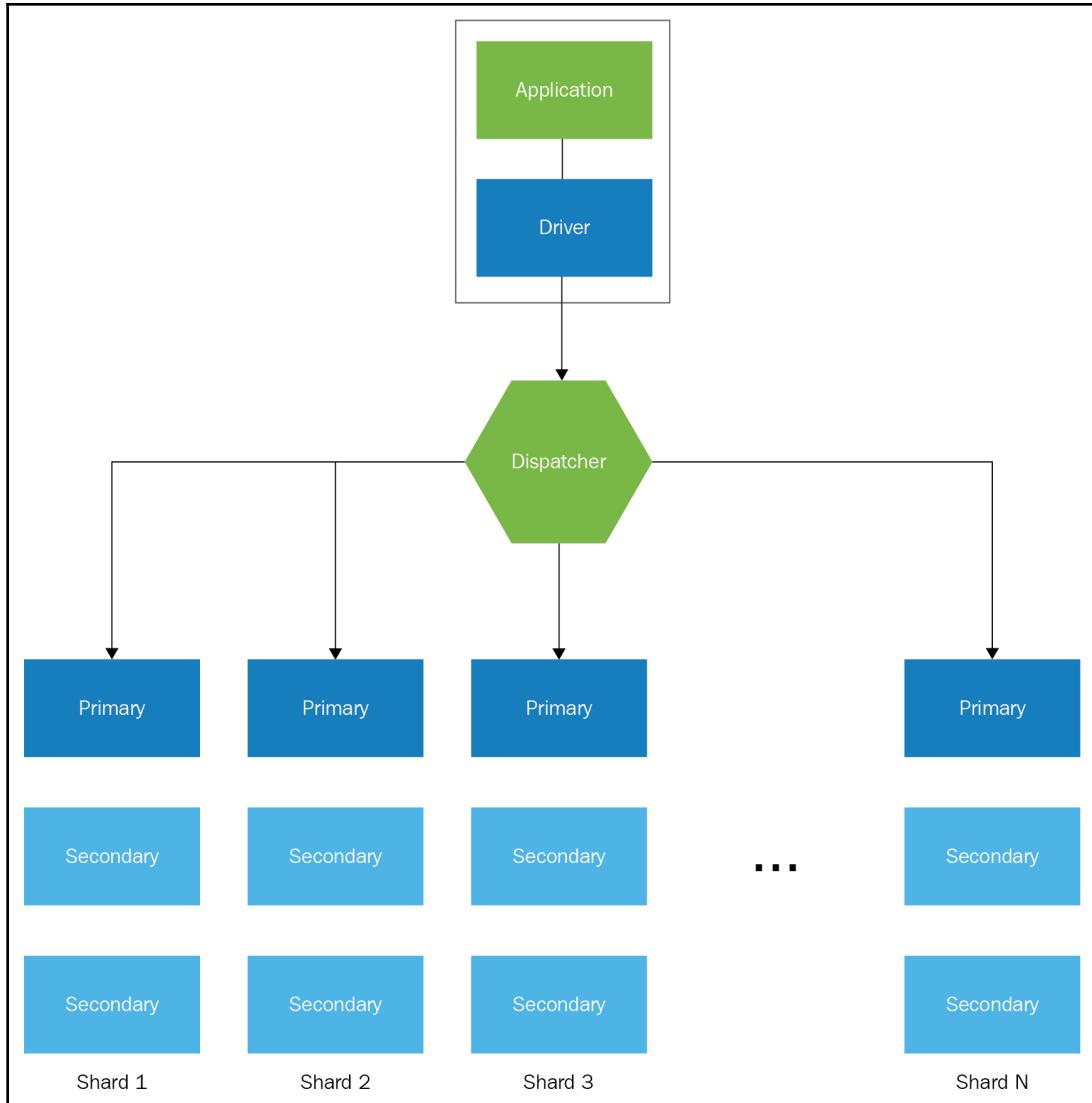
It is possible to replace a component with another if the interface has not changed significantly. It is fundamental to use standards-based technologies and protocols that bring interoperability such as HTTP and RESTful web services:



A good strategy to decouple is to use managed services.

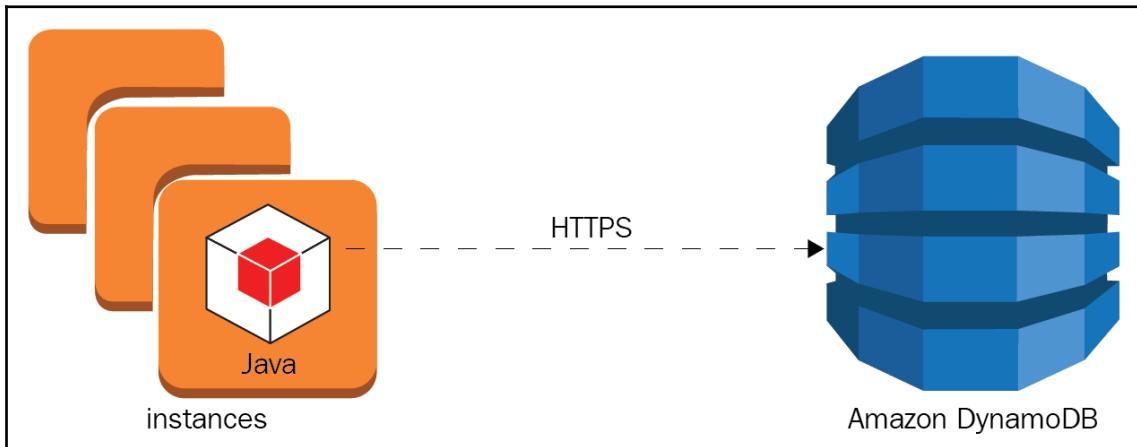
- **Design services, not servers:**

- **Antipattern:** Investing time and effort in the management of storage, caching, balancing, analysis, streaming, and so on, is time-consuming and deviates us from the main purpose of the business: the creation of valuable solutions for our customers. We need to focus on product development and not on infrastructure and operations. Large-scale in-house solutions are complicated, and they require a high level of expertise and budget for research and fine-tuning:



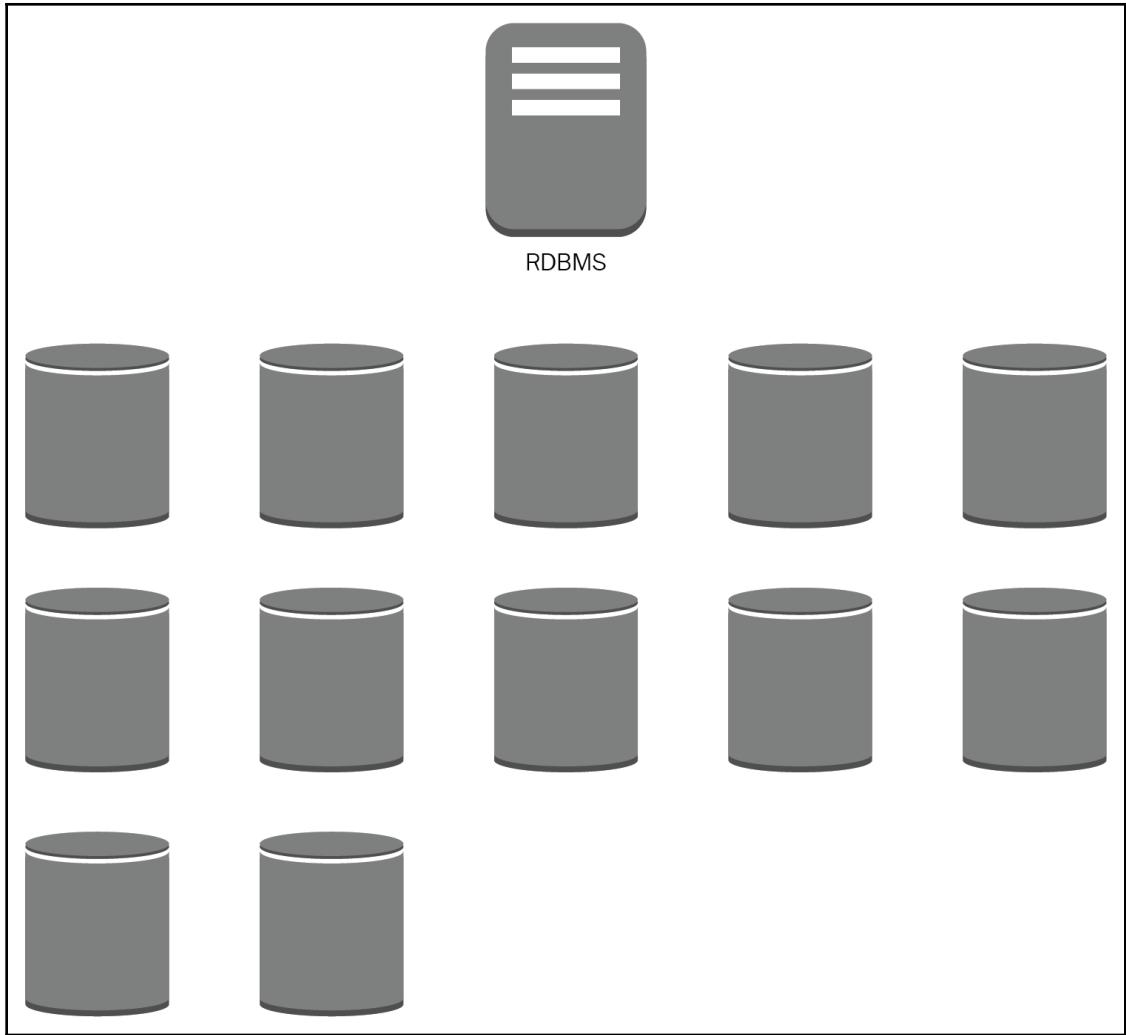
This image depicts the architecture needed for a scalable NoSQL database

- **Best practice:** Cloud-based services abstract the problem of the operation of this specialized services on a large-scale and offload the operations management to a third party. Many AWS services are backed by **Service Level Agreements (SLAs)**, and these SLAs are passed down to our customers. In the end, this improves the brand's reputation and our security posture; also, this will enable organizations to reach higher levels of compliance:



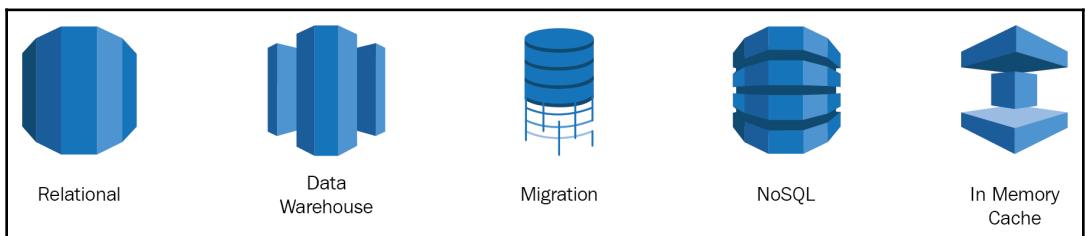
Communication in-transit is secure between AWS services

- **Choose the right database solutions:**
 - **Antipattern:** Taking a **relational database management system (RDBMS)** as a silver bullet and using the same approach to solve any kind of problem, from temporal storage to Big Data. Not taking into account the usage patterns of our data, not having a governance data model, and an incorrect classification can leave data exposed to third parties. Working in silos will result in increased costs associated with the **extract, transform, and load (ETL)** pipelines from dispersed data avoiding valuable knowledge:



Big Data and analytics workloads will require a lot of effort and capacity from an RDBMS datastore.

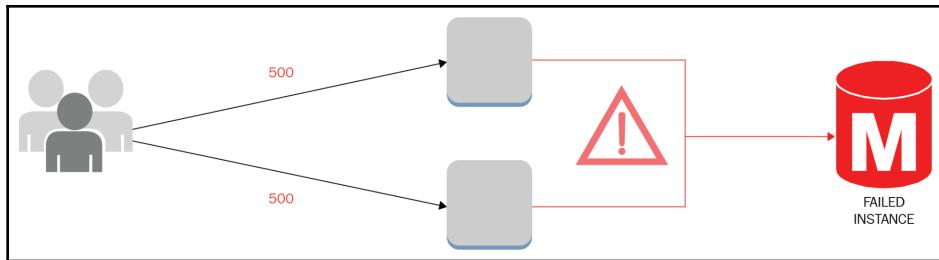
- **Best practice:** Classify information according to its level of sensibility and risk, to establish controls that allow clear security objectives, such as **confidentiality, integrity, and availability (CIA)**. Designing solutions with specific-purpose services and ad hoc use cases such as caching and distributed search engines and non-relational databases. All of these will contribute to the flexibility and adaptability of the business. Understanding data temperature will improve the efficiency of storage and recovery solutions while optimizing costs. Working with managed databases will make it possible to analyze data at petabyte scale, process huge quantities of information in parallel, and create data-processing pipelines for batch and real-time patterns:



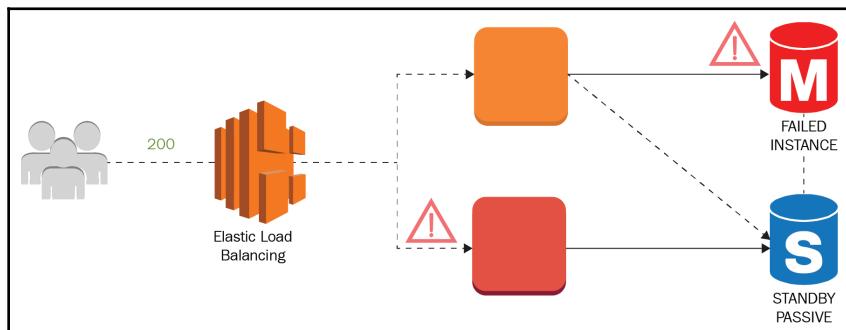
Some specific-purpose storage services in AWS

- **Avoid single points of failure:**

- **Antipattern:** A chain is as strong as its weakest link. Monolithic applications, a low throughput network card, and web servers without enough RAM memory can bring a whole system down. Non-scalable resources can represent single points of failure or even a database license that prevents the use of more CPU cores. Points of failure can also be related to people by performing unsupervised activities and processes without the proper documentation; also the lack of agility could represent a constraint in critical production operations:



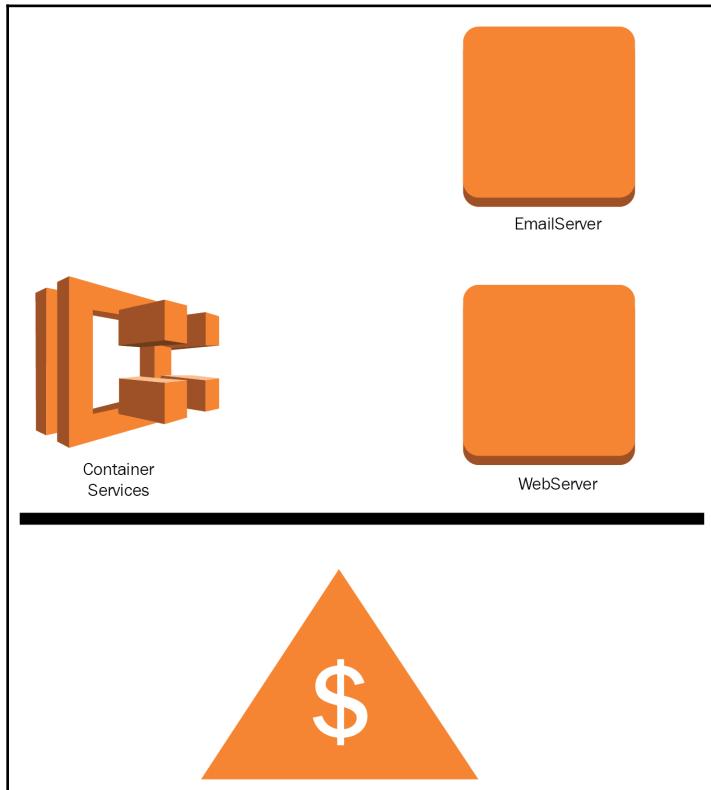
- **Best practice:** Active-passive architectures avoid complete service outages, and redundant components enable business continuity by performing switchover and failover when necessary. The data and control planes must take this N+1 design paradigm, avoiding bottlenecks and single component failures that compromise the full operation. Managed services offer up to 99.95% availability regionally, offloading responsibilities from the customer. Experienced AWS solutions architects can design sophisticated solutions with SLAs of up to five nines:



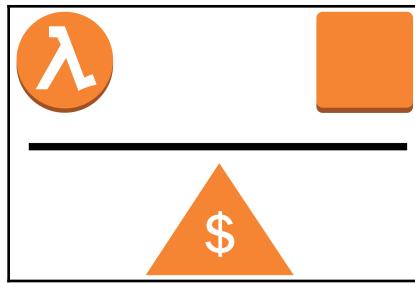
- **Optimize for cost:**

- **Antipattern:** Using big servers for simple compute functions, such as authentication or email relay, could lead to elevated costs in the long term and keeping instances running 24/7 when traffic is intermittent. Adding bigger instances to improve performance without a performance objective and proper tuning won't solve the problem. Even poorly designed storage solutions will cost more than expected. Billing can go out of control if expenses are not monitored in detail.

It is common to provision resources for testing purposes or to leave instances idle, forgetting to remove them. Sometimes, instances need to communicate with resources or services in another geographic region increasing complexity and costs due to inter-region transfer rates:



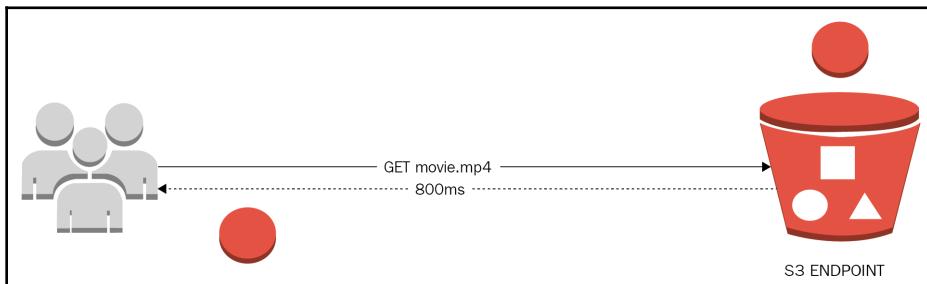
- **Best practice:** Replacing traditional servers with containers or serverless solutions. Consider using Docker to maximize the instance resources and AWS Lambda; use recurring compute resources only when needed. Reserving compute capacity can decrease your costs significantly, by up to 95%. Leverage managed services features that can store transient data such as sessions, messages, streams, system metrics, and logs:



- **Use caching:**

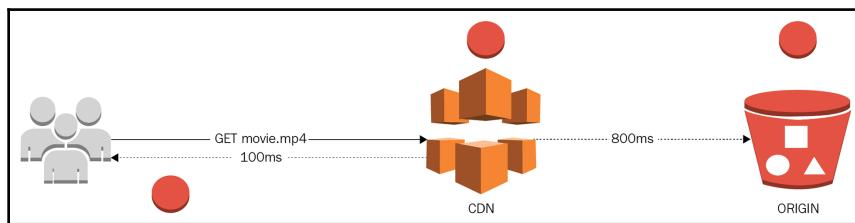
- **Antipattern:** Repeatedly accessing the same group of data or storing this data in a medium not optimized for reading workloads, applications dealing with the physical distance between the client, and the service endpoint. The user receives a pretty bad experience when the network is not available.

Increasing costs due to redundant read requests and cross-region transfer rates, also not having a life cycle for storage and no metrics that can warn about the usage patterns of data:



- **Best practice:** Identify the most used queries and objects to optimize this information by transferring a copy of this data to the closest location to your end users. By using memory storage technologies it is possible to achieve microsecond latencies and the ability to retrieve huge amounts of data. It is necessary to use caching strategies in multiple levels, even storing commonly accessed data directly on the client, for example mobile applications using search catalogs.

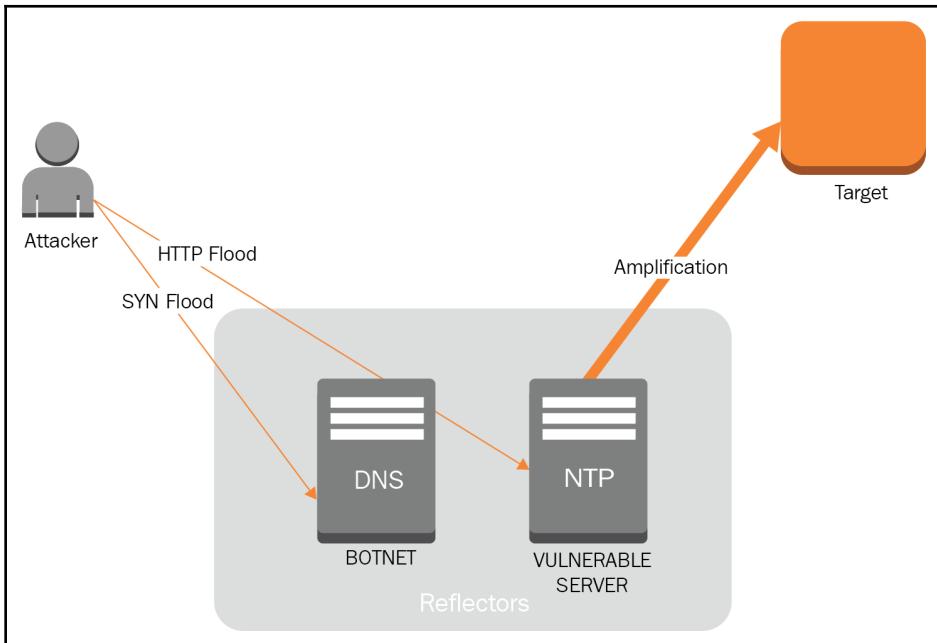
Caching services can lower your costs and offload backend stress by moving this data closer to the consumer. It is even possible to offer degraded experiences without the total service disruption in the case of a backend failure:



- **Secure your infrastructure everywhere:**

- **Antipattern:** Trusting in the operating system's firewalls and being naive about the idea that every workload in the cloud is 100% secure out of the box. Waiting until a security breach is made to take corrective measures maybe thinking that only Fortune 500 companies are victims of **Distributed Denial of Service (DDoS)** attacks. Implementing HTTPS but no security at-rest compromise greatly the organization assets and become an easy prey of ransomware attacks.

Using the root account and the lack of logs management structures will take visibility and auditability away. Not having an InfoSec department or ignoring security best practices can lead to a complete loss of credibility and of our clients:



- **Best practice:** Security must be an holistic labor. It must be implemented at every layer using a systemic approach. Security needs to be automated to be able to react immediately when a breach or unusual activity is found; this way, it is possible to remediate as detected. In AWS, it is possible to segregate network traffic and use managed services that help us to protect valuable assets with complete visibility of operations and management.

At-rest data can be safeguarded by using encryption semantics using cryptographic keys generated on demand delegating the management and usage of these keys to users designated for these jobs. Data in transit must be protected by standard transmission protocols such as IPSec and TLS:



The CIA triad is a commonly used model to achieve information security

Cloud design patterns – CDP

Cloud design patterns are a collection of solutions for solving common problems using AWS technologies. It is a library of recurrent problems that solutions architects deal with frequently when implementing and designing for the cloud. This is curated by the **Ninja of Three (NoT)** team.

These patterns will be used throughout this book as a reference, so you can get acquainted with them and understand their rationale and implementation in detail.



You can find detailed information at this URL: http://en.clouddesignpattern.org/index.php/Main_Page.

AWS Cloud Adoption Framework – AWS CAF

The **Cloud Adoption Framework** offers six perspectives to help business and organizations to create an actionable plan for the change management associated with their cloud strategies. It is a way to align businesses and technology to produce successful results:



The CAF Action Plan template

The six perspectives are grouped as follows:

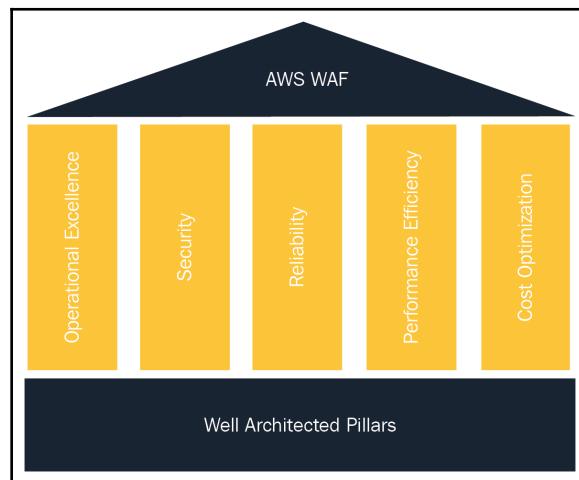
- CAF business perspectives:
 - **Business perspective:** Aligns business and IT into a single model
 - **People perspective:** Personnel management to improve their cloud competencies
 - **Governance perspective:** Follows best practices to improve enterprise management and performance
- CAF technical perspectives:
 - **Platform perspective:** Includes the strategic design, the principles, patterns, and tools to help you with the architecture transition process
 - **Security perspective:** Focuses on compliance and risk management to define security controls

- **Operations perspective:** This perspective helps to identify the processes and stakeholders relevant to execute change management and business continuity

AWS Well-Architected Framework – AWS WAF

The **AWS Well-Architected Framework** takes a structured approach to design, implement, and adopt cloud technologies, and it works around five perspectives so you can look at a problem from different angles. These areas of interest are sometimes neglected because of time constraints and misalignment with compliance frameworks. This book relies strongly on the pillars that are listed here:

- Operational excellence
- Security
- Reliability
- Performance efficiency
- Cost optimization

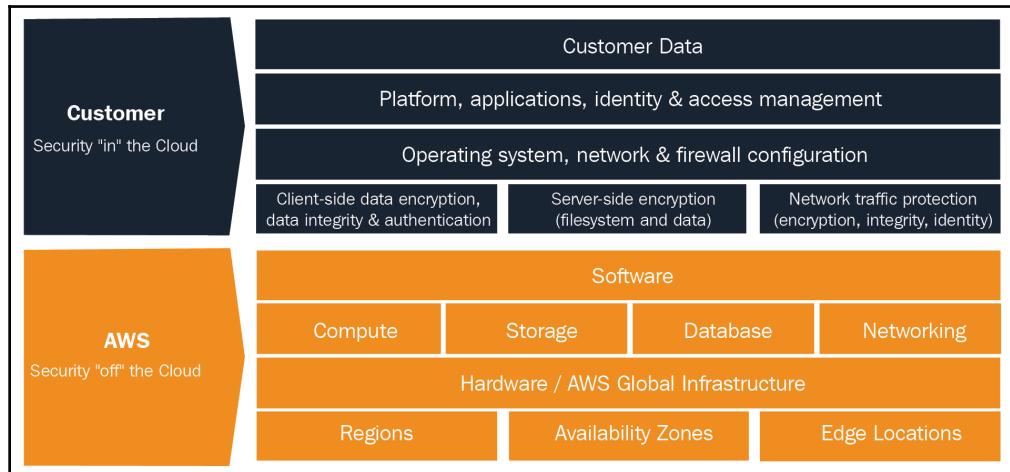


More information and WAF whitepapers can be found here: <https://aws.amazon.com/architecture/well-architected/>.



Shared security model

This model is the way AWS frees the customer from the responsibility of establishing controls at the infrastructure, platform, and services levels by implementing them through their services. In this sense, the customer must provide full control of implementation in some cases, or work in a hybrid model where the customer provides their own solutions by complementing existing ones in the cloud:

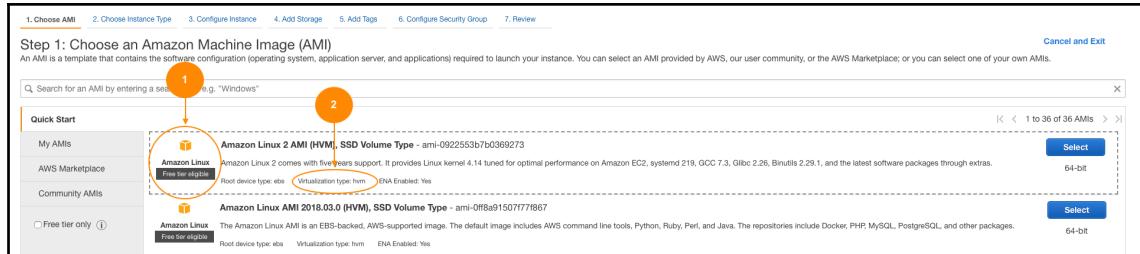


The previous diagram shows that AWS is responsible for the security *of the cloud*; this involves software and hardware infrastructure and core services. The customer is responsible for everything *in the cloud* and the data they are the owner of.

To clarify this model, we will use a simple web server example and explain for every step which controls are in place for the customer and for AWS:

To create our web server, we will create an instance.

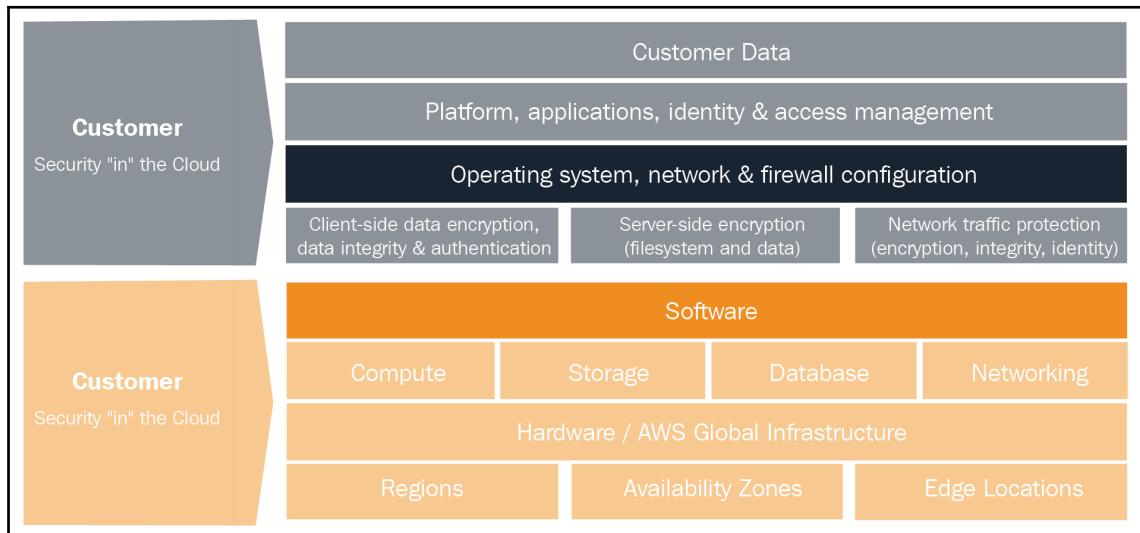
In the EC2 console choose **Launch Instance**:



Following are the details of the instance:

AWS/customer	<ul style="list-style-type: none"> In this example, let's create an instance (1); this image (Amazon Linux AMI) is managed by AWS, and it is security hardened, and it comes preconfigured from software packages from only trusted sources Instances run isolated from other clients by virtual interfaces that run on a custom version of the Xen hypervisor Every disk block is zeroed and RAM memory is randomized
---------------------	---

The previous example is an example of an inherited control (virtualization type) and a shared control (virtual image).





The highlighted components represent the ones relevant for this example.

The next screen is for the configuration of the network attributes and the tenancy mode:

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of Instances: 1 Launch into Auto Scaling Group

Purchasing option: Request Spot instances

Network: vpc-493eb132 (default) Create new VPC

Subnet: No preference (default subnet in any Availability Zone) Create new subnet

Auto-assign Public IP: Use subnet setting (Enable)

Placement group: Add instance to placement group

IAM role: EC2AccessToS3Role Create new IAM role

Shutdown behavior: Stop

Enable termination protection: Protect against accidental termination

Monitoring: Enable CloudWatch detailed monitoring
Additional charges apply.

Tenancy: Shared - Run a shared hardware instance Additional charges will apply for dedicated tenancy.

The following are the details of instance configuration:

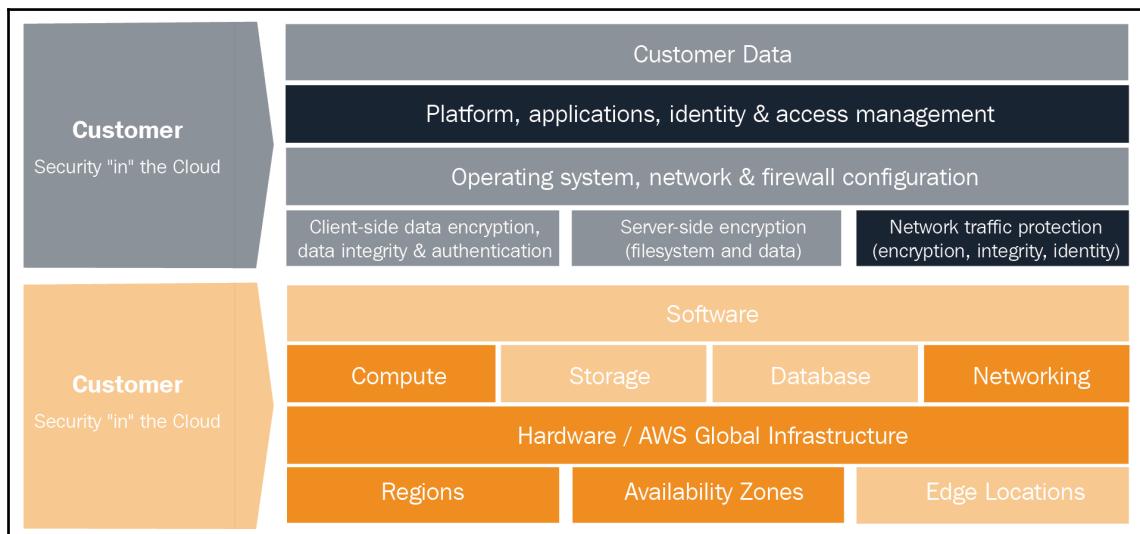
AWS	Every instance runs in a virtual private cloud (Network) (1); the network is an infrastructure-protected service, and the customer inherits this protection, which enables workload isolation to the account level.
Customer	Is possible to segregate the network by means of public and private subnetting, route tables function as a traffic control mechanism between networks, service endpoints, and on-premises networks.
Customer	Identity and Access Management is the service dedicated to user management and account access. IAM Roles are meant to improve security from the customer perspective by establishing trust relationships between services and other parties. EC2AccessToS3Role (2) will allow an instance to invoke service actions on S3 securely to store and retrieve data.
AWS/customer	The Tenancy property (3) is a shared control by which AWS implements security at some layers and the customer will implement security in other layers. It is common to run your instance in shared hosts (multi-tenant), but it can be done on a dedicated host (single tenant); this will make your workloads compliant with FIPS-140 and PCI-DSS standards.

The **virtual private cloud (VPC)** is an example of an inherited control, since AWS runs the network infrastructure; nevertheless, segmentation and subnet configuration is an example of a hybrid control, because the client is responsible for the full implementation by performing a correct configuration and resource distribution.

IAM operations are customer-related, and this represents a specific customer control. IAM roles and all the account access must be managed properly by the client.

Making use of dedicated resources is an example of shared controls. AWS will provide the dedicated infrastructure and the client provides all the management from the hypervisor upwards (operating system, applications).

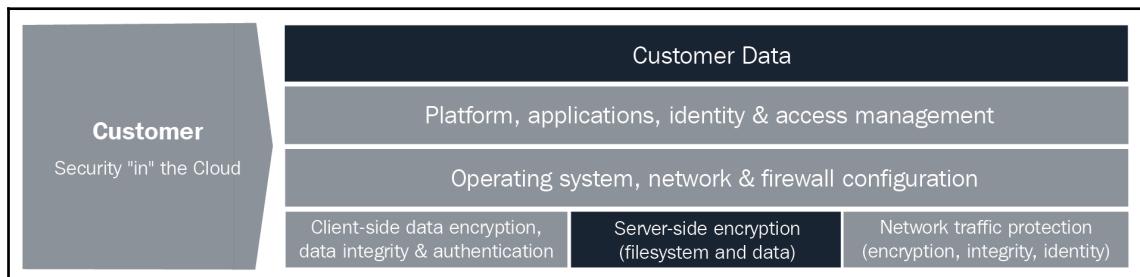
The highlighted components represent the ones relevant for this example. Add a persistent EBS volume to our EC2 instance:



Security at rest for EBS with KMS cryptographic keys

AWS/customer	EBS volumes can be ciphered on demand by using cryptographic keys provided by the Key Management Service (KMS) ; this way all data at rest will be kept confidential
---------------------	---

The EBS encryption attribute is an example of a shared control, because AWS will provide these facilities as part of EBS and KMS services, but the client must enable this configuration properties because by default, disks are not encrypted. The customer has the ability to use specific controls such as **Linux Unified Key Setup (LUKS)** to encrypt EBS volumes with third-party tools:



The highlighted components represent the ones relevant for this example.

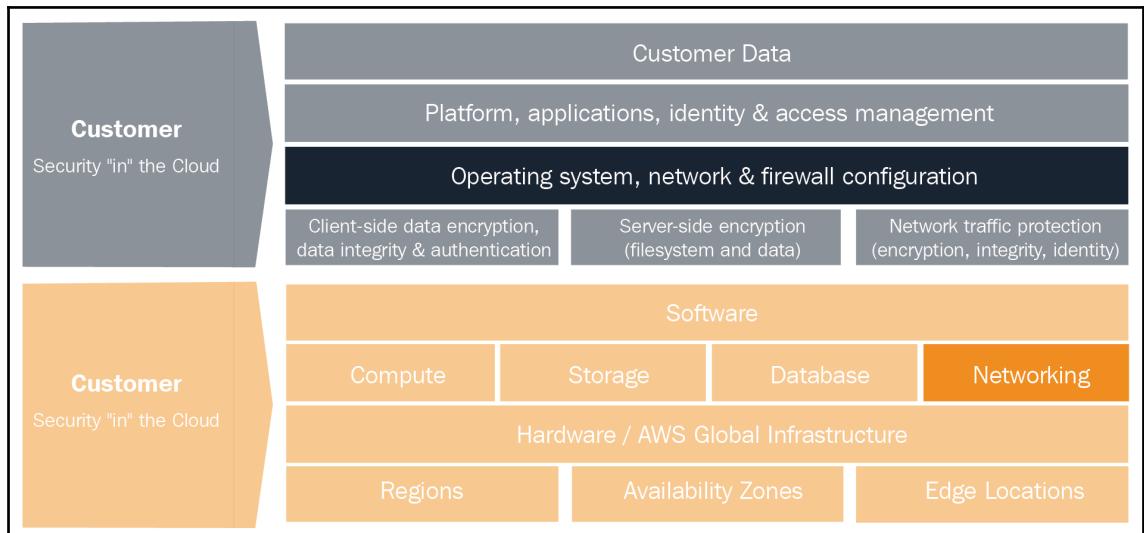


Create a security group to filter the network traffic:

Detail:

AWS/customer	Security groups act as firewalls at the instance level, denying all inbound traffic and opening access only by customer-specified IPs, networks, ports, and protocols. It is a best practice to compartmentalize access by chaining multiple security groups restricting access on every layer. In this example, we create only one security group for the web server in which will be allowed HTTP traffic from any IP address (0 . 0 . 0 / 0) and restricted access via SSH only from a management machine—in this case, my IP.
--------------	---

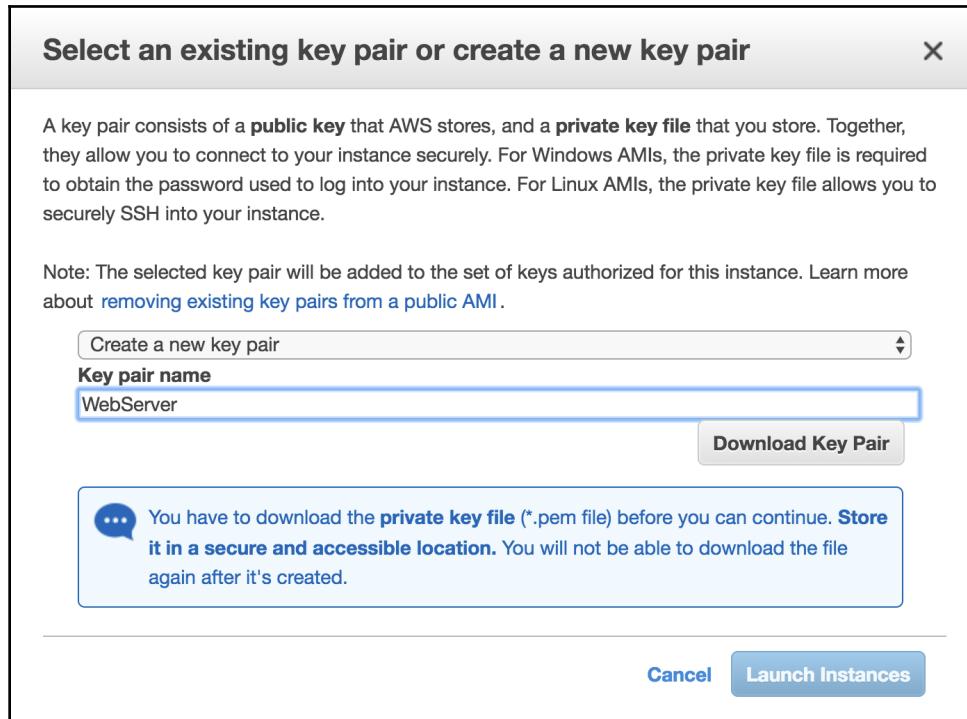
This is a hybrid control because the function of network traffic filtering is from AWS, but the full implementation is given by the customer through the service API:



The highlighted components represent the ones relevant for this example.



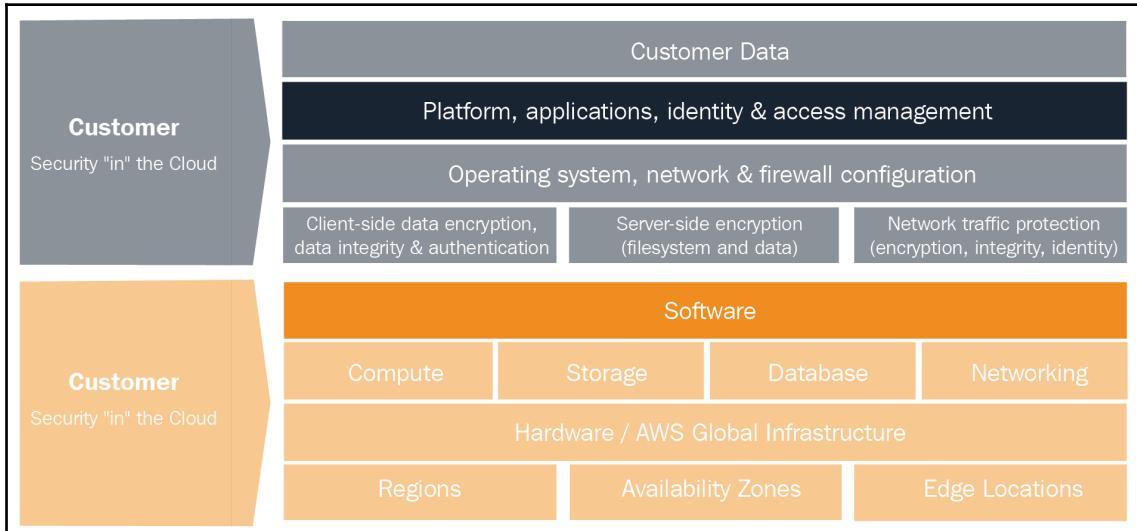
Create a key pair to access the EC2 instance:



Detail:

AWS/Customer	Every compute instance in EC2, whether Linux or Windows, is associated with a key pair, one public key and one private key. The public key is used to cipher the login information of a specific instance. The private key is guarded by the customer so they can provide their identity through SSH for Linux instances. Windows instances use the private key to decrypt the administrator's password.
--------------	--

This is a shared control because the customer and AWS keep responsibility for the guarding of these keys and avoid third-party access that does not have the private key in their possession:



The last step has a dual responsibility:

- The customer must protect the platform on which the application will be running, their applications, and everything related to the identity and access management from the app of the middleware perspective.
- AWS is responsible for the storage and protection of the public key and the instance configuration.

Identity and Access Management

Let's discuss the core services to manage security in the AWS account scope. **Identity and Access Management (IAM)** and **CloudTrail**. IAM is the service responsible for all the user administration, and their credentials, access, and permissions with respect to the AWS service APIs. CloudTrail will give us visibility on how these accesses are used, since CloudTrail records all the account activity at the API level.

1. To enable CloudTrail, you must access the AWS console, find CloudTrail in the services pane and then click on **Create trail**:

The screenshot shows the AWS CloudTrail Trails page. On the left, there's a sidebar with 'CloudTrail', 'Dashboard', 'Event history', and 'Trails' (which is selected). The main area has a heading 'Trails' with a sub-instruction: 'Deliver logs to an Amazon S3 bucket. CloudTrail events can be processed by one trail for free. There is a charge for processing events with additional trails. For more information, see [AWS CloudTrail Pricing](#)'. Below this is a 'Create trail' button with a red circle containing the number '1' over it. A table lists the existing trail: Name (sandbox-account-trail), Region (All), S3 bucket (sandbox-account-trail), Log file prefix (empty), CloudWatch Logs Log group (CloudTrail/DefaultLogGroup), and Status (Off). To the right, there's a 'Learn more' section with links to 'Pricing', 'Documentation', 'Forums', and 'FAQs'.

2. The configuration is flexible enough to record events in one region only, or cross-region in the same account, and you can even record CloudTrail events with multiple accounts; it is recommended to choose **All** for **Read/Write events**:

The screenshot shows the 'Create Trail' page. The sidebar on the left includes 'CloudTrail', 'Dashboard', 'Event history', and 'Trails'. The main form has a 'Trail name*' field containing 'audit-trail' with a red circle containing the number '1' over it. Below it is a 'Apply trail to all regions' section with a radio button set to 'Yes'. Under 'Management events', there's a note about management events providing insights into operations, with a 'Learn more' link. At the bottom, there's a 'Read/Write events' section with a radio button set to 'All'.

It is important to enable this service initially in newly created accounts and always keep it active because it also helps in troubleshooting when configuration problems occur, when there are production service outages, or to attribute actions to IAM users.

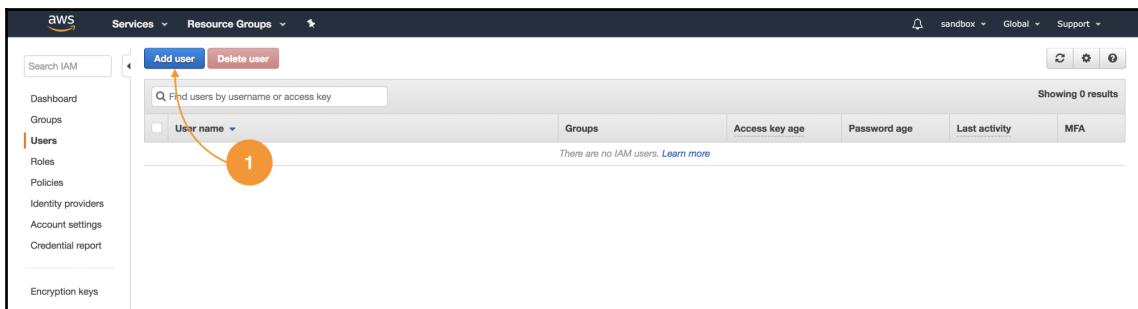
CloudTrail is considered a best practice associated with the operation of any AWS account.



User creation

We will create an IAM user and this user can be used by a person for the purpose of everyday operations. But it can also be used by an application invoking service APIs.

1. Let's navigate to the IAM service in the console, choose the left menu **Users**, and then **Add user**:



2. The user we will create can only access the web console, and for this, we will create credentials that consist of a username and a password:



3. In this case, check the **AWS Management Console access** (1) as shown in the next screenshot. You have the possibility to assign a custom password for the user or to generate one randomly. It is a good practice to enable password resetting on the next sign-in. This will automatically assign the policy **IAMUserChangePassword** (2) that will allow the user to change their own password as shown here:

Add user

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name* gabriel

Add another user

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Access type* Programmatic access
Enables an access key ID and secret access key for the AWS API, CLI, SDK, and other development tools.

AWS Management Console access
Enables a password that allows users to sign-in to the AWS Management Console.

Console password* Autogenerated password Custom password
***** Show password

Require password reset User must create a new password at next sign-in
Users automatically get the IAMUserChangePassword policy to allow them to change their own password.

* Required

[Cancel](#) [Next: Permissions](#)

4. Choose **Next: Permissions**; on this screen, we will leave it as is. Select **Create User** to demonstrate the default user level that every new user has. The last screen shows us the login data, such as the URL. This URL is different than root access:

Add user

Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://aws-sa-associate-guide.sigin.aws.amazon.com/console>

[Download .csv](#)

User	Access key ID	Secret access key	Password	Email login instructions
gabriel	AKIAIKSJ5JAU2TC47ARA	***** Show	***** Show	Send email

5. Copy the URL, username, and password shown in the previous screenshot. The access URL can be customized; by default, it has the account number but it is up to the administrator to generate an alias by clicking **Customize**:

Welcome to Identity and Access Management

IAM users sign-in link:
<https://aws-sa-associate-guide.signin.aws.amazon.com/console>

1

IAM Resources

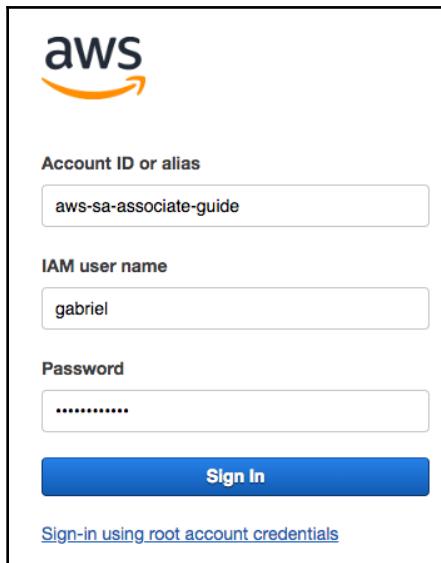
Users: 1 Roles: 1
Groups: 0 Identity Providers: 0
Customer Managed Policies: 14

Security Status

2 out of 5 complete.

<input checked="" type="checkbox"/> Delete your root access keys	▼
⚠ Activate MFA on your root account	▼
<input checked="" type="checkbox"/> Create individual IAM users	▼
⚠ Use groups to assign permissions	▼
⚠ Apply an IAM password policy	▼

6. Close your current session and use the new access URL. You will see a screen like the following:



7. Use your new IAM administrator username and password; once logged in, search for the EC2 services. You will get the following behavior:

The screenshot shows a list of denied permissions for EC2 resources in the US East (N. Virginia) region. The items listed are:

- You are not authorized to describe Running Instances
- You are not authorized to describe Dedicated Hosts
- You are not authorized to describe Volumes
- You are not authorized to describe Key Pairs
- You are not authorized to describe Placement Groups
- You are not authorized to describe Elastic IPs
- You are not authorized to describe Snapshots
- Error retrieving resource count
- You are not authorized to describe Security Groups

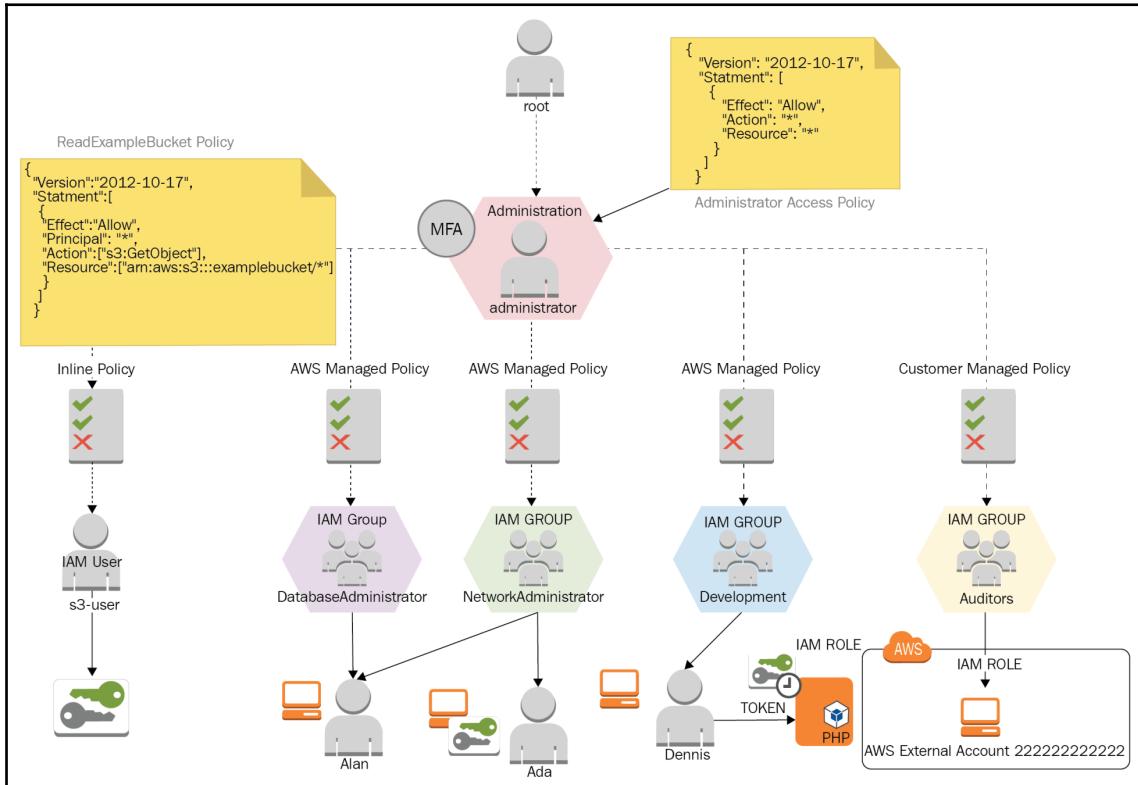
8. Let's validate the current access scope by trying to list the account buckets in S3:

The screenshot shows the AWS S3 console interface. At the top, there is a search bar labeled "Search for buckets". Below the search bar are three buttons: "+ Create bucket", "Delete bucket", and "Empty bucket". To the right of these buttons, it shows "Buckets 0" and "Regions 0". A red horizontal bar highlights an error message: "Error Access Denied". Below this, there is a table header with columns: "Bucket name", "Access", "Region", and "Date created". A note at the bottom states: "* Objects might still be publicly accessible due to object ACLs. [Learn more](#)".

This simple test shows up something fundamental about AWS security. Every IAM user, once created, has no permissions; only when permissions are assigned explicitly will the IAM user be able to use APIs, including the AWS console. Let's close this session and log in again; this time, with the root account to create a secure access structure.

Designing an access structure

We will work on the access structure using the following model:

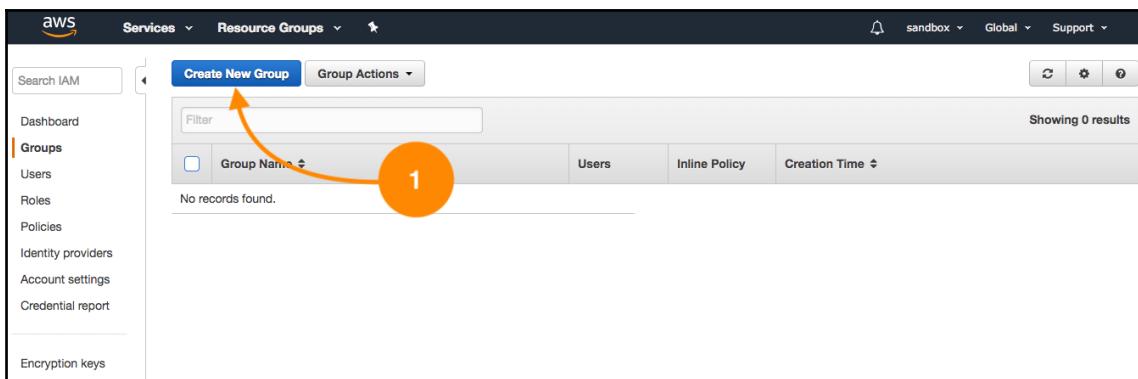


This diagram shows different use cases for **IAM Users**, **IAM Groups**, **IAM Roles**, and the **IAM Policies** for everyone.

Create an administration group

We will create IAM groups solely for administration purposes, and a unique user that has wide access to this account.

1. Navigate to **IAM | Groups** and then choose the action **Create New Group**:

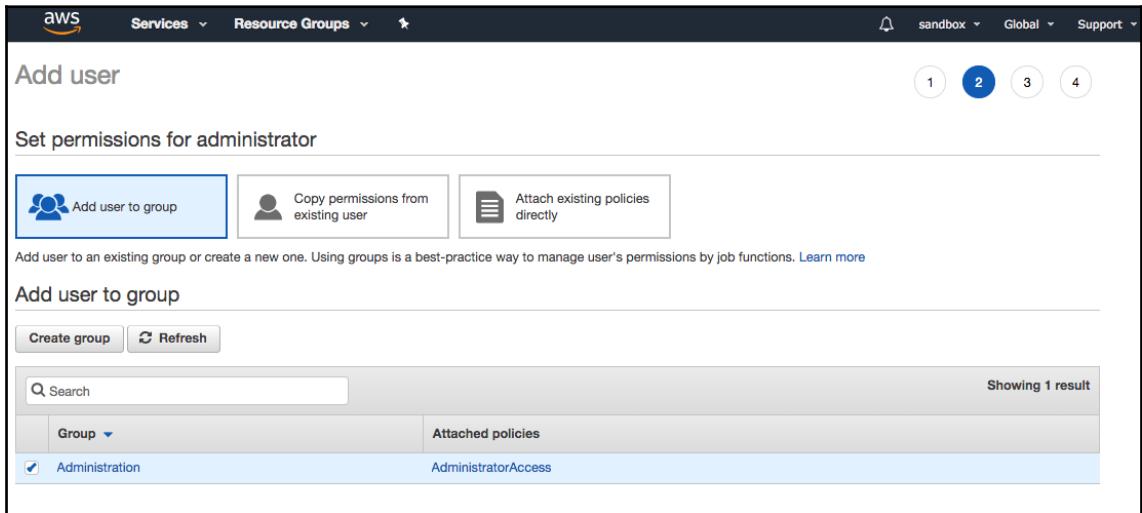


2. Use the name **Administration**, and select **Next Step**. In this screen, we will add an IAM policy. IAM policies are JSON formatted documents that specify granularly the permissions that an entity has (user, group, role).
3. In the search bar of the step, write the word **administrator** to filter some of the options available for administration policies; the policy we need is called **Administrator Access**. This policy is an AWS-managed policy and is available for every AWS customer for their use when a full administration policy is required for a principle.
4. Select **Next Step**. This will lead us to the review screen, where we can visually confirm that the administrator policy has been added. This policy has many AWS resources and has a unique Amazon Resource Name as follows:

`arn:aws:iam::aws:policy/AdministratorAccess`

5. Add the group; in the group detail, we can observe that it doesn't have any users in it. Let's add our administrator user. This user must be used instead of the root account.
6. Choose the left menu **IAM users** and create a new user with the name **administrator**; in this case, we will only choose **AWS Management Console** access, because, at this time, this user won't be using access keys, only the username and the password, to log into the console.

7. In step 2 of the **Add User** screen, choose the administration group; this way, the user will inherit the **AdministratorAccess IAM** policy from the group **Administration**:



8. Select **Next: Review and Create User**. It is now possible to download the **comma-separated value (CSV)** file with the access data for this user. Again, copy the access URL for this user and close the root session; from now on, we will be using only the administrator user created. The login screen must be similar to this:

The screenshot shows the AWS sign-in page. It has fields for 'Account ID or alias' (filled with 'aws-sa-associate-guide'), 'IAM user name' (filled with 'administrator'), and 'Password' (a masked password). At the bottom is a large blue 'Sign In' button. Below the button is a link: 'Sign-in using root account credentials'.

9. Once the login data is entered, we will be required to change the temporary password, guaranteeing confidentiality. Now that we are logged in as an IAM user, we can confirm, because the top bar has changed to administrator@account as shown:



Business case

Ada is a network administrator; part of her activities consist of provisioning **virtual networks (VPCs)**, subnets, configuring route tables, managing hosted zones for domains in Route 53, and managing logs of the network traffic.

Ada supports database and development areas in the provisioning of infrastructure and network resources, but she should not be able to create compute resources as EC2 instances and EBS volumes.

Alan is a DBA, and most of his tasks are performance monitoring, checking database logs, provisioning databases, and performing full database backups on S3. This user must be able to read network parameters for each database instance and troubleshoot issues in production databases.

Alan will work with the AWS console because most of the monitoring operations are made from a web environment and other DBAs just using a vendor-specific driver. We will create a database administrator group, because, this way, the administration is centralized. By doing this, we can control and maintain all the security aspects in one spot; this is a better approach than having to manage tens or hundreds of users independently.

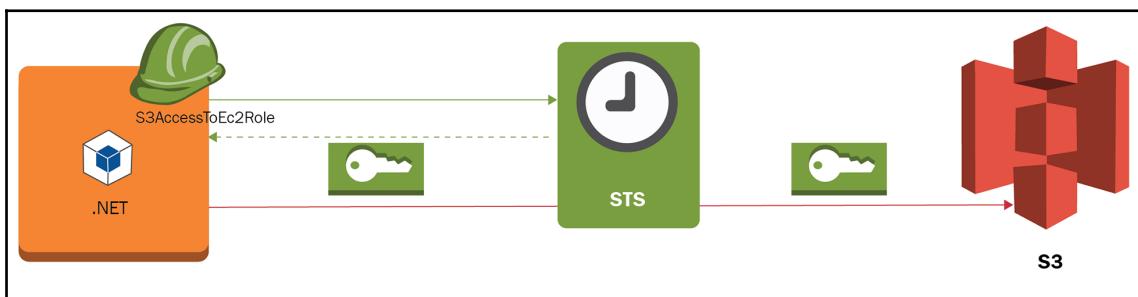
Let's create four groups and three users with the following details:

Function	IAM group name	IAM policy	IAM user
IAM User	DatabaseAdministrator	DatabaseAdministrator	Alan
IAM User	NetworkAdministrator	NetworkAdministrator	Ada
IAM Service Role	Development	-	Dennis
IAM Cross Account	Auditors	SecurityAudit	-

Alan must support Ada's activities temporarily while she is out of the office for vacations, so we need to add Alan to the network administrator group. The set of permissions Alan will have are the union of both groups.

IAM groups allow a user to become a member of one or more groups, as groups exist at the same level. It's impossible to nest a group inside another group.

Dennis is the member of the development team and he must be able to create EC2 instances. He must also be able to create EC2 service roles that permit cross-service access. These IAM roles give the ability to impersonate the access to services and avoid the usage of explicit access keys when applications interact via CLI or programmatically to authenticate APIs. The application Dennis will deploy consists of a .NET application that uses the AWS SDK and accesses an S3 bucket to store and retrieve data. To achieve this, the IAM role **S3AccessToEC2Role** must be created via IAM. This instance role and profile will internally require temporal credentials via the **security token service (STS)**:



The practice of not using `ACCESS_KEY` and `SECRET_ACCESS_KEY` credentials will prevent other parties from using them if they are found in the filesystem or in the application configuration files.

Our user Dennis must be able to create instances and IAM roles for applications. Dennis will make use of an IAM policy inherited via the development group and will be given the permissions needed to create instance roles and pass IAM roles to services. Let's create this policy, so it can be attached to the development group.

1. Navigate to **IAM | Policies | Create policy** and choose the **JSON** tab and paste the document available under this file: <https://github.com/gabanox/Certified-Solution-Architect-Associate-Guide/blob/master/chapter00/RoleCreatorPolicy.json>:

```

1: {
2:   "Version": "2012-10-17",
3:   "Statement": [
4:     {
5:       "Effect": "Allow",
6:       "Action": [
7:         "ec2:RunInstances",
8:         "ec2:AssociateIamInstanceProfile",
9:         "ec2:ReplaceIamInstanceProfileAssociation"
10:      ],
11:      "Resource": "*"
12:    },
13:    {
14:      "Effect": "Allow",
15:      "Action": [
16:        "iam:PassRole",
17:        "iam:CreateRole",
18:        "iam>ListPolicies"
19:      ],
20:      "Resource": "*"
21:    }
22:  ]
23: }

```

Feedback English (US) © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

2. We are required to specify a **Name** and **Description** for the new IAM policy:

Service	Access level	Resource	Request condition
EC2	Limited: Write	All resources	None
IAM	Limited: Write	All resources	None

* Required Cancel Previous Create policy

3. Once we have created our policy, let's associate it with the development group. On the IAM Policies listing, we have the option to filter by name (1); in this case, this policy was created globally in this account and can be reused. Choose **Customer managed** or use the search box to find it:

The screenshot shows the AWS IAM Policies list. The left sidebar has 'Policies' selected. The main area shows a single result: 'RoleCreatorPolicy'. The policy details show it allows EC2 to run instances and associate IAM instance profiles. The JSON code is displayed below.

```

    "Effect": "Allow",
    "Action": [
        "ec2:RunInstances",
        "ec2:AssociateIamInstanceProfile",
        "ec2:ReplaceIamInstanceProfileAssociation"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "*"
}
]
}

```

4. Once we choose the policy, select **Policy Actions** and use the function attach (2), as shown in the preceding screenshot; this will redirect us to the detail, where we can use attached entities and associate them with the development group as shown:

The screenshot shows the IAM Policy Summary page for 'RoleCreatorPolicy'. It lists one entity attached: 'Development'. The 'Attached entities' tab is selected, showing the 'Development' group.

5. Again, let's use the policy filter choosing **AWS managed** until we find **AmazonEC2FullAccess**. Repeat the process to attach this policy to the development group. This way, Dennis will be able to create EC2 instances and associate the appropriate instance role:

The screenshot shows the AWS IAM Policies search results. A filter is applied for 'AWS managed' policies, and the search term 'amazonec2full' is entered. One result is found: 'AmazonEC2FullAccess'. The details show it is an 'AWS managed' policy with 0 attachments and a description stating it provides full access to Amazon EC2 via the AWS Management Console.

6. Close the current session and log in with Dennis' credentials. Let's navigate to the IAM section and proceed to create a new role. The service that will use this role is EC2:

The screenshot shows the 'Create role' wizard, Step 1: Select type of trusted entity. It lists four options: 'AWS service' (selected), 'Another AWS account', 'Web identity', and 'SAML 2.0 federation'. Below the options, a note states: 'Allows AWS services to perform actions on your behalf.' with a 'Learn more' link. Step 1 is highlighted with a blue circle.

Select type of trusted entity

- AWS service** EC2, Lambda and others
- Another AWS account Belonging to you or 3rd party
- Web identity Cognito or any OpenID provider
- SAML 2.0 federation Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

Create role

Choose the service that will use this role

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

API Gateway	CodeDeploy	EMR	IoT	Rekognition
AWS Support	Config	ElastiCache	Kinesis	S3
AppSync	DMS	Elastic Beanstalk	Lambda	SMS
Application Auto Scaling	Data Pipeline	Elastic Container Service	Lex	SNS
Auto Scaling	DeepLens	Elastic Transcoder	Machine Learning	SWF
Batch	Directory Service	Elastic Load Balancing	Macie	SageMaker
CloudFormation	DynamoDB	Glue	MediaConvert	Service Catalog
CloudHSM	EC2	Greengrass	OpsWorks	Step Functions
CloudWatch Events	EC2 - Fleet	GuardDuty	RDS	Storage Gateway
CodeBuild	EKS	Inspector	Redshift	Trusted Advisor

* Required [Cancel](#) [Next: Permissions](#)

7. Under permissions, search for s3 in the search box and add the policy **AmazonS3FullAccess**:

Create role

Attach permissions policies

Choose one or more policies to attach to your new role.

[Create policy](#) [Refresh](#)

Filter: All ▾ Showing 4 results

	Policy name ▾	Attachments ▾	Description
<input type="checkbox"/>	AmazonDMSRedshiftS3Role	0	Provides access to manage S3 settings for Redshift endpoint...
<input checked="" type="checkbox"/>	AmazonS3FullAccess	0	Provides full access to all buckets via the AWS Management...
<input type="checkbox"/>	AmazonS3ReadOnlyAccess	0	Provides read only access to all buckets via the AWS Manag...
<input type="checkbox"/>	QuickSightAccessForS3StorageManagement...	0	Policy used by QuickSight team to access customer data pr...

8. Name the role **EC2ToS3InstanceRole** and click **Create**:

Create role

Review

Provide the required information below and review this role before you create it.

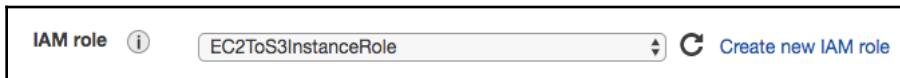
Role name*
Use alphanumeric and '+=-, @-_.' characters. Maximum 64 characters.

Role description Allows EC2 instances to call AWS services on your behalf.
Maximum 1000 characters. Use alphanumeric and '+=-, @-_.' characters.

Trusted entities AWS service: ec2.amazonaws.com

Policies [AmazonS3FullAccess](#)

9. Spin a new EC2 instance in step 3 of the EC2 wizard, under **Configure Instance**, and make sure to select **EC2toS3InstanceRole**:



At this point, if you have some problem creating this users structure, you can run the following script from the command line.

- First, make sure that you have access keys configured; if they aren't, use `aws configure`
- Download the following script and set execution permissions with `chmod +x checkpoint.sh` from your terminal: <https://github.com/gabanox/Certified-Solution-Architect-Associate-Guide/blob/master/chapter00/checkpoint1.sh>

Inline policies

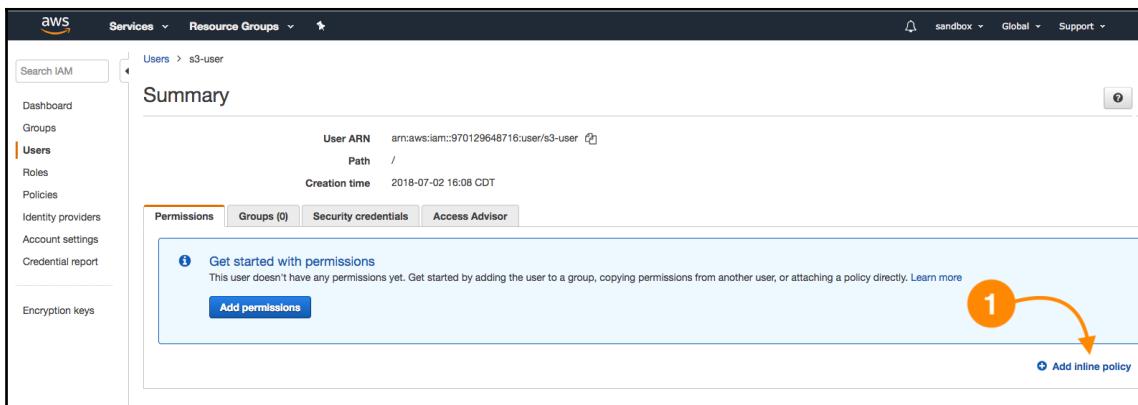
As the administrator, navigate to the S3 service in the console, and create a bucket with a unique name. After that, upload a text file with any kind of content using the upload button.

Write down the bucket name, as you will use it later.

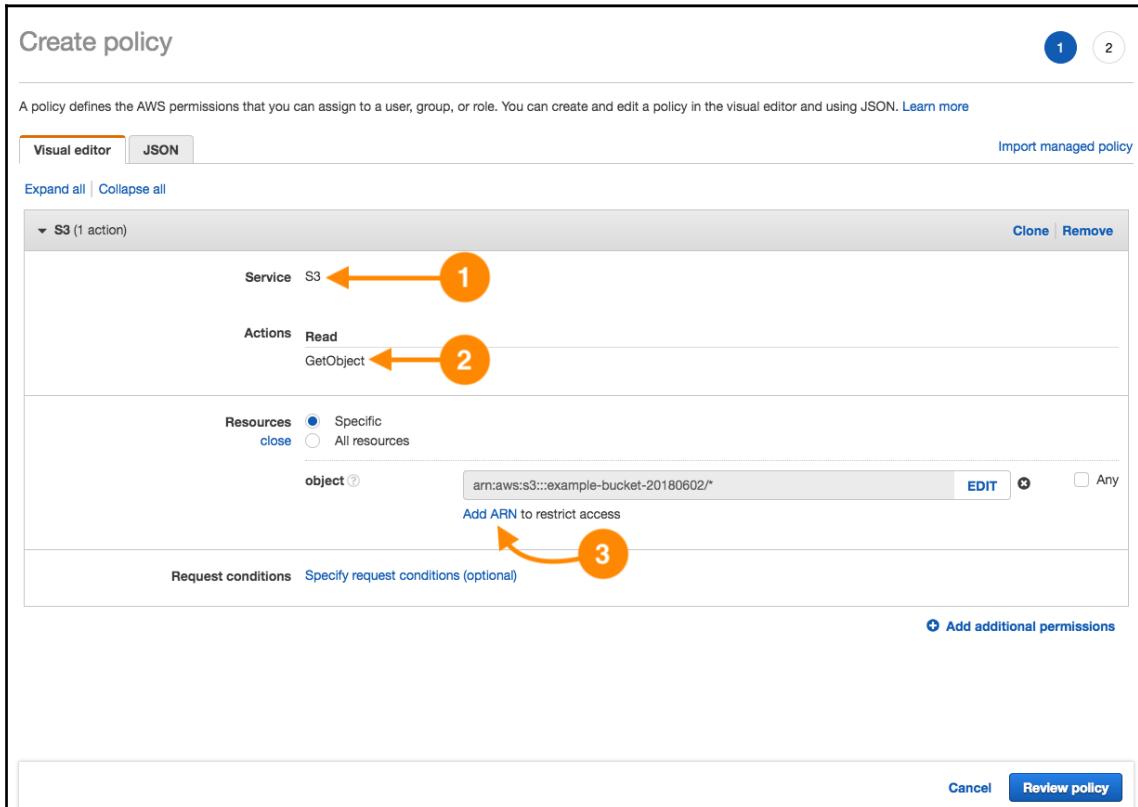
1. We will create an IAM user with CLI-only access and a policy that only allows reading objects. Use the name `s3-user` as shown in the next screenshot, and in the fourth step of the **Add user** wizard, download the access credentials CSV file:

The screenshot shows the 'Add user' wizard in the AWS IAM console. Step 1: Set user details. The 'User name*' field contains 's3-user'. Below it is a link to 'Add another user'. Under 'Select AWS access type', the 'Access type*' dropdown has 'Programmatic access' selected (indicated by a checked checkbox). A detailed description follows: 'Enables an access key ID and secret access key for the AWS API, CLI, SDK, and other development tools.' There is also an unselected checkbox for 'AWS Management Console access'. At the bottom, there is a note: '* Required' and two buttons: 'Cancel' and 'Next: Permissions'.

2. Make sure **Programmatic access** is enabled; we will create this user without any kind of permissions at first. Once created, the user will be assigned an **inline policy**; these policies cannot be reused with other users because they are defined only once and attached directly to users and groups.
3. Under the **Users** section in **IAM**, select the **s3-user** and **Add inline policy** as shown here:



4. Use the visual editor, in this case by choosing **S3** as the service (1). Under **Actions**, select only **GetObject** (2), and for the resources section, select **Specific** and **Add ARN** (3), writing down the bucket name and the option **Any**:



5. To configure the access credentials via CLI, open a terminal and paste the following command replacing the values for `aws_access_key_id` and `_aws_secret_access_key` with your own (the `s3-user` CSV file values):

```
aws configure set profile.s3-user.aws_access_key_id  
"AKIAXXXXXXXXXXXXXX"  
aws configure set profile.sqs-user.aws_secret_access_key  
"ZuEVD4DDyK1TsmNp/Pa6toR/Qf3FFUN0t/XXXXXX"
```

6. The following command will allow the download operation (a read) to the local filesystem:

```
aws s3 cp s3://example-bucket-20180602/test-object.txt --profile s3-user .
```

The structure of this command is as follows:

```
aws s3 cp <LocalPath> <S3Uri> or <S3Uri> <LocalPath> or <S3Uri> <S3Uri> ...
```

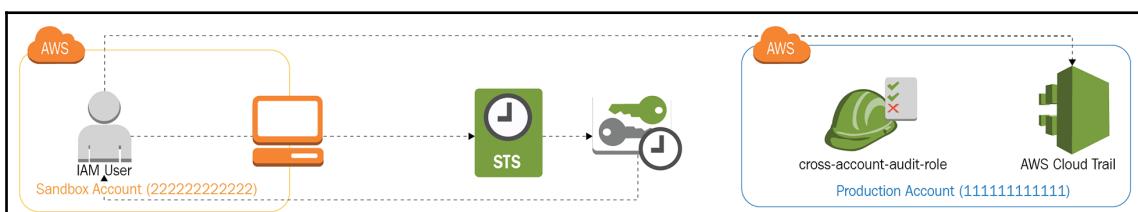
This way, we are using the `copy` subcommand and validating that the S3 user has read-only access to objects.

IAM cross-account roles

An external auditors group requires read-only access to the account so they can inspect API activity on the account. These auditors are a company that has an AWS account, but to follow the less-privileged principle, we will use a role with an AWS Managed Policy called **SecurityAudit**, plus the **AWSCloudTrailReadOnlyAccess**. This way, there won't be a necessity to create an additional user in the audited account, but this account can define the level of permissions necessary to let external auditors perform their task.

This task requires an additional AWS account, so this is only demonstrative. Also, take into account that only IAM users are allowed to perform the `AssumeRole` action.

1. As an administrator, navigate to IAM and create a role. On the type of trusted entity, choose **Another AWS account**. For this demo, I will use a hypothetical account number. The IAM role used in this scenario is called **delegation**:



We have two accounts, the **Production Account (11111111111)**, which will have the **cross-account role** created, and the access policy for this external audit user, and the **Sandbox Account (222222222222)**, the one that will assume the external role to access AWS APIs; in this case, Cloud trail.

2. The next step is to create the cross-account role in the production account. The type of trusted entity is **Another AWS account**, and we will specify the trusted account number (sandbox):

Create role

Select type of trusted entity

1 2 3

AWS service EC2, Lambda and others

Another AWS account Belonging to you or 3rd party

Web identity Cognito or any OpenID provider

SAML 2.0 federation Your corporate directory

Allows entities in other accounts to perform actions in this account. [Learn more](#)

Specify accounts that can use this role

Account ID*

Options Require external ID (Best practice when a third party will assume this role)
 Require MFA 

External auditors will also have read-only access to Cloud trail:

Create role

Review

Provide the required information below and review this role before you create it.

Role name*
Use alphanumeric and '+,-,@,_-' characters. Maximum 64 characters.

Role description
Maximum 1000 characters. Use alphanumeric and '+,-,@,_-' characters.

Trusted entities The account 970129648716

Policies  SecurityAudit  AWSCloudTrailReadOnlyAccess

Permissions boundary Permissions boundary is not set

3. Select the role created to review the details; by default, it is configured to allow broad access using the root principal as the trusted entity.
4. Edit this trust relationship as shown next, and specify our IAM user in this case administrator (it could be a member of the audit group):

Edit Trust Relationship

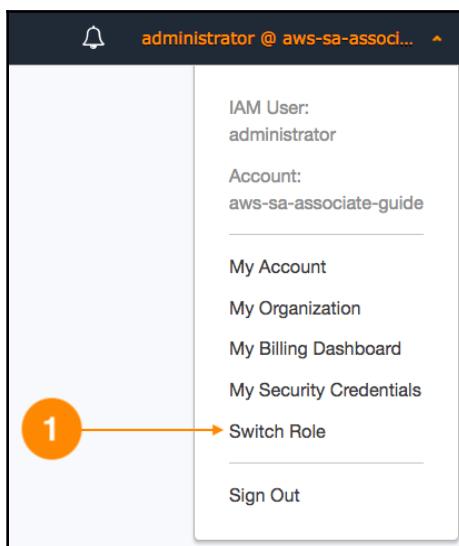
You can customize trust relationships by editing the following access control policy document.

Policy Document

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Principal": {  
7         "AWS": "arn:aws:iam::111111111111:user/administrator"  
8     },|  
9       "Action": "sts:AssumeRole",  
10      "Condition": {}  
11    }  
12  ]  
13 }
```

Cancel Update Trust Policy

Now, in the sandbox account, we need to configure the audit-role this account will assume. This can be done in the current user upper menu as follows:



In the next screen, you will be prompted to specify the requested account (production), the name of the external role to assume, and a friendly name:

The dialog box contains the following fields:

- Account***: 111111111111
- Role***: cross-account-audit-role
- Display Name**: audit-role
- Color**: A color palette showing six squares: pink, orange, yellow, green, blue, and black.

At the bottom, there is a note: ***Required**, a **Cancel** button, and a **Switch Role** button.

5. Now, we can see our role-friendly name listed under **Role History**; once we assume this identity, our web console will display the production account data and the top menu will change to display the current assumed role:

The AWS Home page displays the following information in the top right corner:

- Logged in as: administrator
- Account: aws-sa-associate-guide
- Currently active as: cross-account-audit-role
- Account: hipermediasoft

The **Role History** section shows:

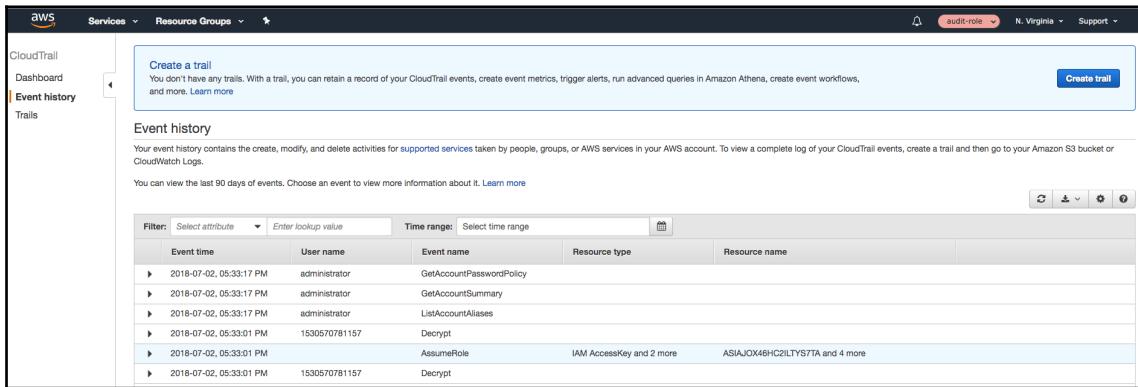
- audit-role
- development
- security
- sandbox-arch
- sandbox

Other links in the top right include:

- My Account
- My Organization
- My Billing Dashboard
- Back to administrator

The main content area includes sections for **AWS services**, **Build a solution**, and a sidebar with **Helpful tip** and **Explore AI**.

6. To end this lesson under Cloud trail, it is possible to see all the API events since logging was activated for the account:



The screenshot shows the AWS CloudTrail console. In the left sidebar, 'Event history' is selected. At the top, there's a 'Create a trail' button and a note about trails. Below that is a section titled 'Event history' with a note about viewing the last 90 days of events. A table lists several API events from July 2, 2018, at 05:33:17 PM, performed by the user 'administrator'. The events include 'GetAccountPasswordPolicy', 'GetAccountSummary', 'ListAccountAliases', 'Decrypt', and 'AssumeRole' for IAM AccessKey and 2 more.

Event time	User name	Event name	Resource type	Resource name
2018-07-02, 05:33:17 PM	administrator	GetAccountPasswordPolicy		
2018-07-02, 05:33:17 PM	administrator	GetAccountSummary		
2018-07-02, 05:33:17 PM	administrator	ListAccountAliases		
2018-07-02, 05:33:01 PM	1S30570781157	Decrypt		
2018-07-02, 05:33:01 PM		AssumeRole	IAM AccessKey and 2 more	ASIAJ0X46HC2ILTY57TA and 4 more
2018-07-02, 05:33:01 PM	1S30570781157	Decrypt		

As a complementary activity, it is recommended to enable **Multi-Factor Authentication (MFA)**; you can provision this 2FA by downloading Google Authenticator from your mobile app store.



For detailed information check the documents here: <https://aws.amazon.com/iam/details/mfa/>.

Summary

This chapter is the formal introduction to Amazon Web Services; you have learned about cloud design principles, patterns, frameworks, and best practices for AWS architecture. You will put security in practice by designing an access structure for a business case for several kinds of users, manage them via groups, and enable cross-account roles.

Further reading

- **Understanding Cloud Design Patterns:** http://en.clouddesignpattern.org/index.php/Main_Page
- **The AWS Cloud Adoption Framework:** <https://aws.amazon.com/es/professional-services/CAF/>
- **AWS architecture well framework:** <https://aws.amazon.com/es/architecture/well-architected/>
- **Architecting for the Cloud (AWS Best Practices):** https://d1.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf
- **AWS - Overview of Security Processes:** https://d1.awsstatic.com/whitepapers/Security/AWS_Security_Whitepaper.pdf

2

AWS Global Infrastructure Overview

In this chapter, you will learn the most important concepts about the AWS global infrastructure, and by the end of this chapter, you will be able to design highly available and resilient solutions in the cloud using multiple data centers, replicate data to a second region to perform backups and disaster recovery, and understand the trade-offs associated with the election of a geographic region. You will create a web app and deploy it using a **content delivery network (CDN)** with CloudFront, and manage the hosted zone in Route 53. You will also work with the AWS and S3 low-level API CLIs, and you will understand the consistency model and the security aspects of encryption of data at rest and in transit.

The following topics will be covered in this chapter:

- Introducing AWS global infrastructure
- Single-region/multi-region patterns
- Global CDN
- Data replication and redundancy with managed services

Technical requirements

1. You will need an AWS account. If you have not already done so, you can create one free (<https://aws.amazon.com/free/>) or, if you have one already, make sure you have enough privileges to create IAM users.
2. To perform the S3 exercises, it is recommended to download the **AWS Command Line Interface**, which can be found here, for a getting started guide(<https://aws.amazon.com/cli/>).
3. For the CLI, you need Python 2.6.5 or later, and the package manager **PIP**. Once installed, configure your client with the next command:

```
aws configure
```

This command will prompt you for the ACCESS KEY and SECRET ACCESS KEY from your IAM user, and for a working region. The region code can be obtained from the following resource: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html>.

Introducing AWS global infrastructure

The AWS infrastructure is one of the most critical aspects of cloud computing, since many of the architectural and design decisions involve the use of **Infrastructure as a Service (IaaS)**. The infrastructure layer comprehends physical locations where these locations operate abstracted by a service interface. As solutions architects, we must have a clear understanding of service design that improves the overall availability compared to traditional environments. You will learn how to implement recovery procedures and strategies that allow for the automatic self-healing of components to achieve business continuity, enabling the delivery of secure and reliable applications to end users.

Becoming a service company

The distributed services technological world now requires a global scope, and users think about the cloud as a service provider; no matter where they are physically located, they need to be able to access their information at any time, wherever they are. The cloud market demands infrastructure so robust that it allows an increasing number of users to consume information efficiently.

Data centers have a fixed capacity, and this represents a restriction on the amount of resources available; you will not be able to provision more resources beyond the last rack available in the physical facilities.

"This means that the next server will cost 10M because you will need to buy a new data center"

- Jon Jenkins "Velocity Culture" (<https://www.youtube.com/watch?v=dxk8b9rSK0o>)

Service clouds offer elastic services that in some way abstract this problem. In the year 2000, Amazon was dealing with big issues pertaining to the scaling of their infrastructure, which led to the creation of strategies that could support this hyper growth, and the result was the basis of what is now **Amazon Web Services (AWS)**. During this time, AWS has learned a lot from its experience, as it is not an easy endeavor to pioneer a new market and remain an industry leader.

"So very quietly around 2000, we became a services company with really no fanfare."

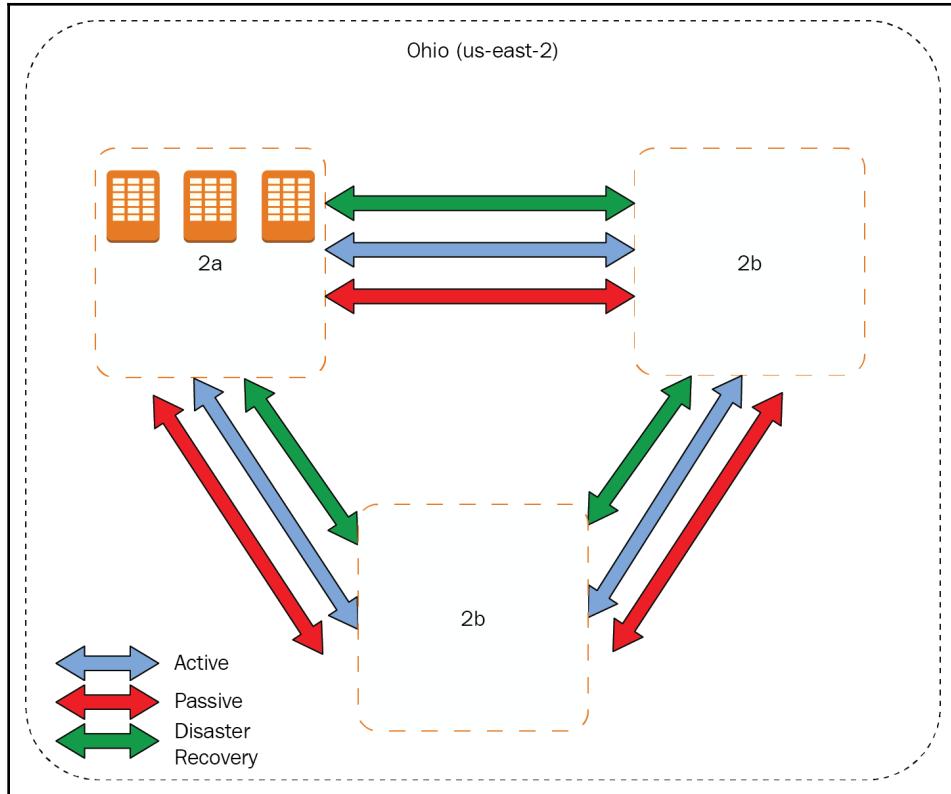
- Andy Jassy, AWS CEO (<https://techcrunch.com/2016/07/02/andy-jassys-brief-history-of-the-genesis-of-aws/>)

At the time of the writing of this book, AWS has 15 geographic regions and a total of 54 Availability Zones and three private regions; one in the US (GovCloud) and two more in China (Beijing and Ningxia). So, AWS is positioned as the cloud provider with the most significant infrastructure in the world when it comes to data centers.

Data centers

AWS takes into account environmental and geographic considerations when building their data centers and avoids sharing the same line of failure; they are entirely autonomous and isolated among them and interconnected in full mesh with low-latency private links.

They are designed to provide $N+1$ redundancy and have automatic recovery procedures in the cases of disaster; this provides end users have a continuity of operations in the case of a natural disaster or any other kind of service failure. Some of the AWS services have procedures to replicate data to a redundant data center providing reliability:



10,000-feet view

To refer to a data center, we use the term **Availability Zone (AZ)**, which contains a collection of computing resources designed to provide high availability within a region; some AWS services maintain automatic replication and synchronization between AZs; on the other hand, services require us to architect for data replication and synchronization.

Regions

AWS Regions are a means to deliver applications and services with a global footprint, allowing the elasticity of resources and information synchronization with a high level of control and isolation. They make it possible to execute multiple workloads, satisfying granular compliance programs in multiple countries. We shall discuss different points of view to assess the election of a geographic region to leverage every characteristic in our cloud deployments.

100,000-feet view

A group of AZs are included in a geographic region (two or more) and provide additional requirements besides availability. They are completely isolated from other regions, bringing more tolerance under failure. Management capabilities operate by isolating workloads and having unique resources in every region, for example, operating system images (Amazon Machine Images).



Choosing a geographic region is the first decision to make when designing in AWS and this is not trivial.

Latency

The closest regions to end users have the advantage of delivering services with low latency without generating additional costs for the use of caching services.

Compliance

Some governments require that public service providers and data storage exist contained within the geographic limits of their nation. In this sense, some countries can be compliant by isolating workloads into their jurisdiction. The countries that do not have an AWS region in their territory can use additional security mechanisms such as encryption, obfuscation, and the use of secure communication channels that complement the required security objectives.

Supported services

Historically, the first AWS region was built in North Virginia, and most of the services and features were first released here, followed by Oregon, which is a more recent region with modern hardware and has latterly been the favorite region for public releases. Amazon has an impressive development pace and it is usual that every month new services, functionalities, and regulatory compliances get announced.

Cost

Costs may vary per region, and this is mostly due to local duties and the economic position of every country. This fact can play to our favor because we can execute more cost-efficient workloads in other regions rather than production, for example, development environments archiving and cold backups.

Connectivity

AWS Direct Connect is a service that makes it possible to access data centers throughout a dedicated connection from a **Point of Presence (PoP)** to AWS directly; this service can be acquired via a third-party provider by using dedicated fiber private links.

Endpoint access

Direct Connect Gateway enables global access to private VPC resources and AWS services at a low cost; this was formerly known as **inter-region connectivity**, which allowed users to operate in multiple AWS Regions across the US including GovCloud, with limited access elsewhere. This new model allows users to gain interregional access to data and services, making efficient infrastructure operations with better throughput, improving the whole connectivity experience, and reducing costs significantly, compared to multiple ISP links.

Global CDN

Edge Locations are the last component in the AWS global infrastructure. They are collocated resources on the edge of AWS Regions. CloudFront uses this type of infrastructure for content delivery, to access services and data efficiently in the main cities of the world. They expand the functionality of the AZs, arranging an edge location cache providing low latency for applications, and it is a **Content Delivery Network (CDN)**. It is also used to propagate DNS and Amazon Route 53 anycast data; you can even execute AWS Lambda directly on the border.

Amazon CloudFront

This service is responsible for maintaining multiple copies of your data in different locations for a fast search, depending on where the request is being made. Amazon CloudFront can be thought as a cache load balancer. It is also possible to use it for streaming data and an upload optimization for services like Amazon S3 with transfer acceleration enabled.

Single region / multi-region patterns

Regularly, we can find two recurring patterns on every cloud deployment; the first consists of using a single region maintaining communications and resources very closely, distributed to the AZ level. The second one is to deploy services and applications globally using multiple regions with a higher reach even in different AWS accounts. These two patterns are discussed next, to understand the logic behind these decisions.

Rationale

As architects, we need to make decisions based on information and we must have a business justification. For example, to extend our customer base by offering services in different countries, we need to provide data replication to improve durability and integrity of data, and synchronization for tasks such as batch, and reporting or traffic segregation for better application management.

Active-active

A computing cluster operates with a minimum of two nodes, and each one brings the same kind of service simultaneously; this is called **active-active**. The purpose of this topology is to load balance and avoid **single points of failure (SPOF)**.

Active-passive

As its name suggests, this kind of cluster maintains an active primary service and a passive secondary waiting to replace the primary in the case of a failure; these clusters are commonly used in disaster recovery strategies and consist of a means to balance requests between primary and secondary services as necessary to maintain business operations, even in the case of maintenance.

Network-partitioning tolerance

Regional failures are rare, but remember *anything that can go wrong will go wrong*; to build resiliency, we need to take into account that networking components could fail at any time. When a regional failure happens, in which clients get disconnected for connectivity issues, we can reroute current traffic to a second region in which systems are up and running and to some extent synchronize data concerning the primary region.

Let's imagine this scenario: a user purchases an e-commerce website, and distributed systems of this application create a transaction that propagates from the middleware to a DBMS. Before the database updates, a network switch fails, leading to an inconsistent state for the business and database tiers due to a mismatch. The application cannot prove the system consistency, because at this moment we have two versions of the system until a rollback gets fired at the middleware. One half of the system has not finished the update operation (**data tier**), and the other half is waiting for a commit and acknowledges this (**business tier**); this leads to a split-brain scenario, where the two parts of a brain know nothing about their failing counterpart.

This idea is the essence of the CAP theorem, which tells us that any distributed system in the presence of a network partition must, by nature, choose between consistency and availability.

Some of the AWS services are designed to be highly available and tolerant to network partitioning giving on strong consistency in favor of future eventual consistency, maybe we are talking about milliseconds for a system to be strongly consistent again until it converges.

Complexity

Active-active and active-passive solutions are patterns that make significant improvements in the **Quality of Service (QoS)** aspects with a trade-off in complexity. They require further monitoring tooling and failover scripts for automatic recovery and orchestration capabilities; this results in higher **Total Cost of Ownership (TCO)**.

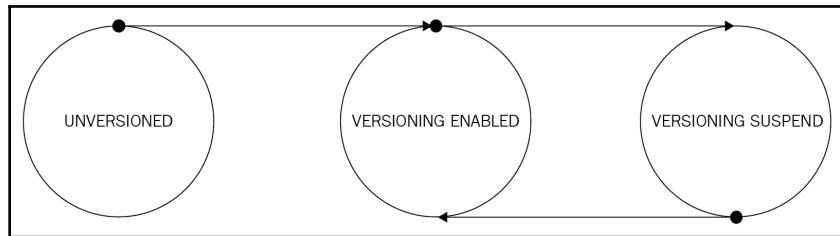
CloudFront

The **CloudFront** service is designed to be resilient in the face of multiple failure types and has global reach, capable of operating in conjunction with AWS Lambda and AWS Shield to execute custom logic at the edge and offer DDoS protection for your applications.

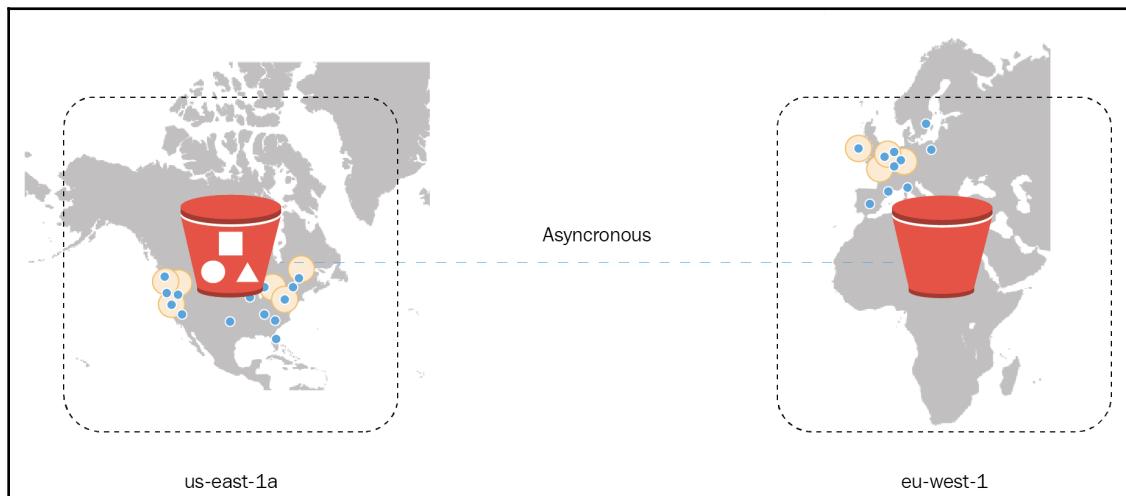
Data replication and redundancy with managed services

- **Problem:** We need replication of PNG images to a second region in the same AWS account to improve data durability, and in the case of a disaster, recover 100% of data to the primary region.
- **Possible solution:** Two S3 buckets need to be configured; one in each region to be completely isolated and avoid sharing the same failure line. For this exercise, versioning must be enabled and **cross-region replication (CRR)** must be configured.

This diagram shows that an S3 bucket can be found in one of three states:



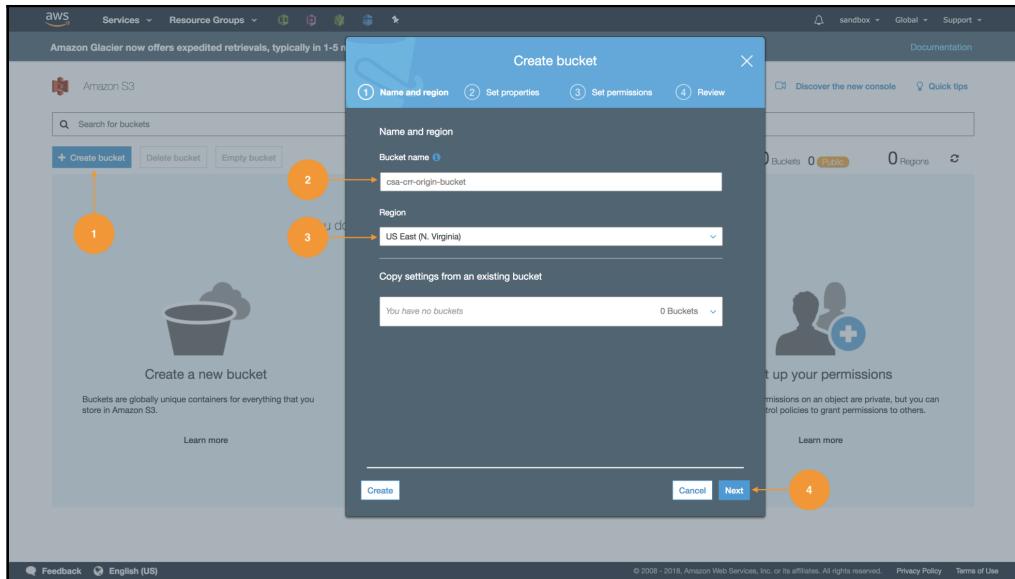
Let's create an interregional deployment, using two S3 buckets with automatic versioning and replication, to synchronize objects from North America to Europe, just as depicted in the following diagram:



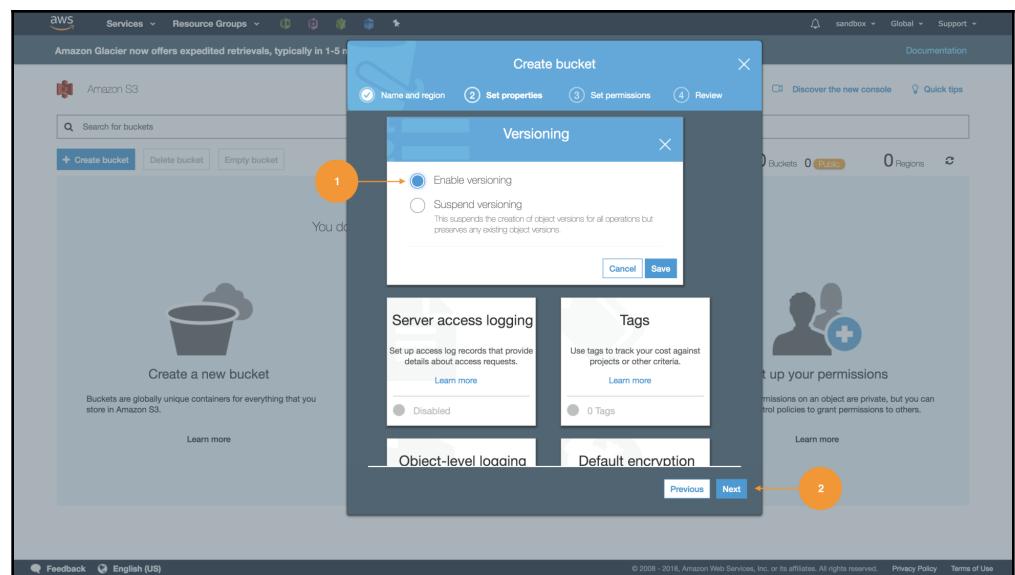
Exercise

Follow the given steps to configure a bucket in different regions in the same AWS account.

1. Let's create the origin bucket (us-east-1):



2. Enable versioning on this bucket (this will replicate all object operations including ACLs, tags, and metadata). Replication is a useful feature when it comes to preventing data loss from accidental deletes:





In the confirmation screen, validate that the region is North Virginia (`us-east-1`) and versioning is enabled.

Create bucket

X

1 Name and region 2 Set properties 3 Set permissions 4 Review

Name and region

Bucket name `cse-crr-origin-bucket` Region US East (N. Virginia)

Properties

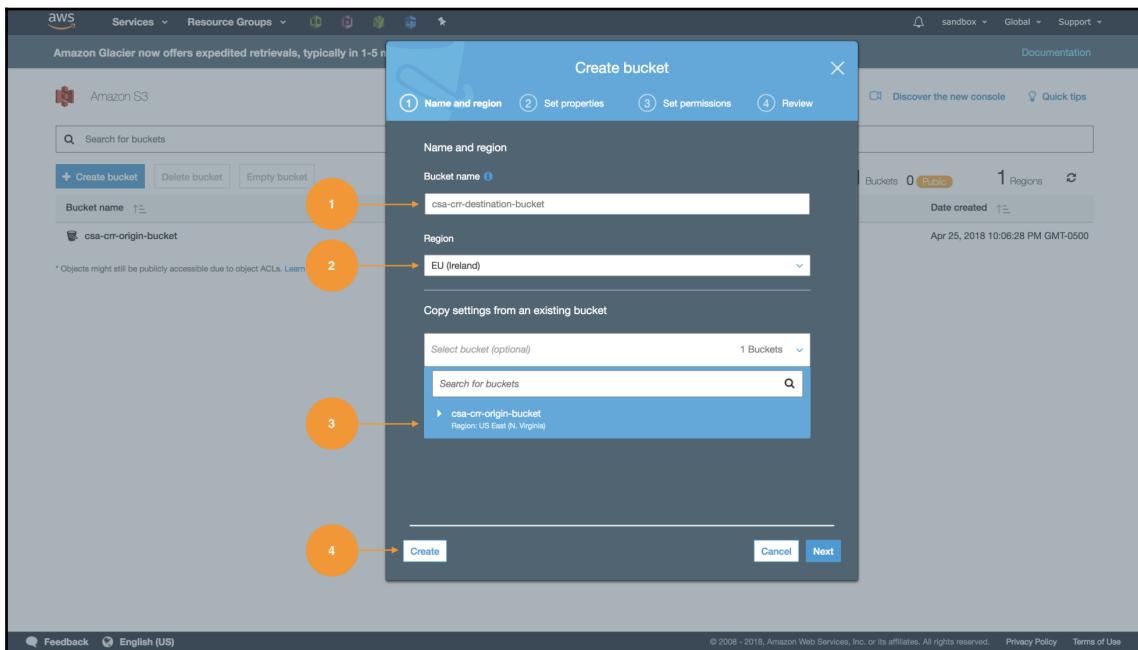
Versioning	Enabled
Server access logging	Disabled
Tagging	0 Tags
Object-level logging	Disabled
Default encryption	None

Permissions

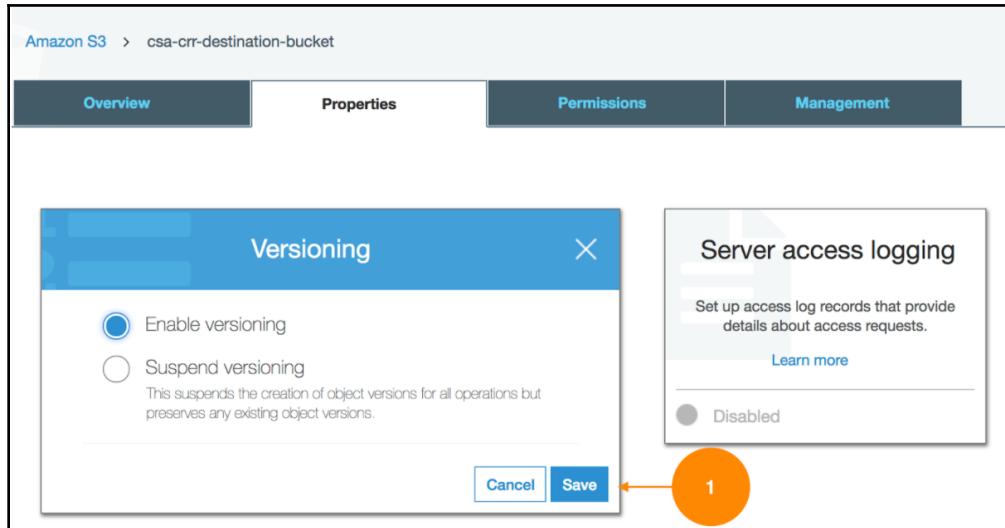
Users	1
Public permissions	Disabled
System permissions	Disabled

Previous Create bucket

3. Let's provision the destination bucket in the secondary region (eu-west-1):



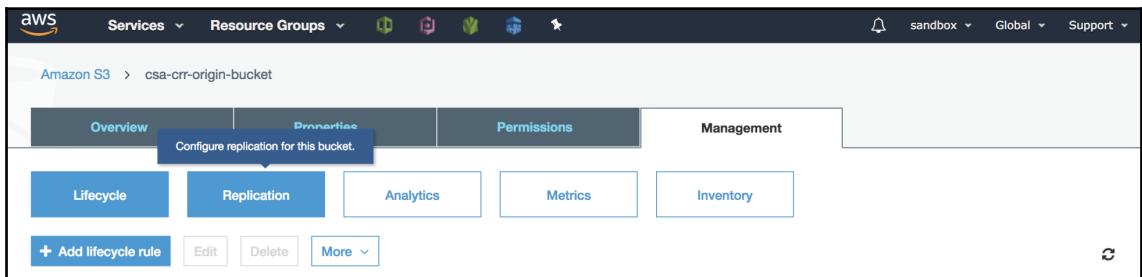
4. A copy of the existing configurations on this bucket can be enabled, and once the bucket has been created, we will need to make sure versioning is active; if not, the case enables this feature in the bucket properties:



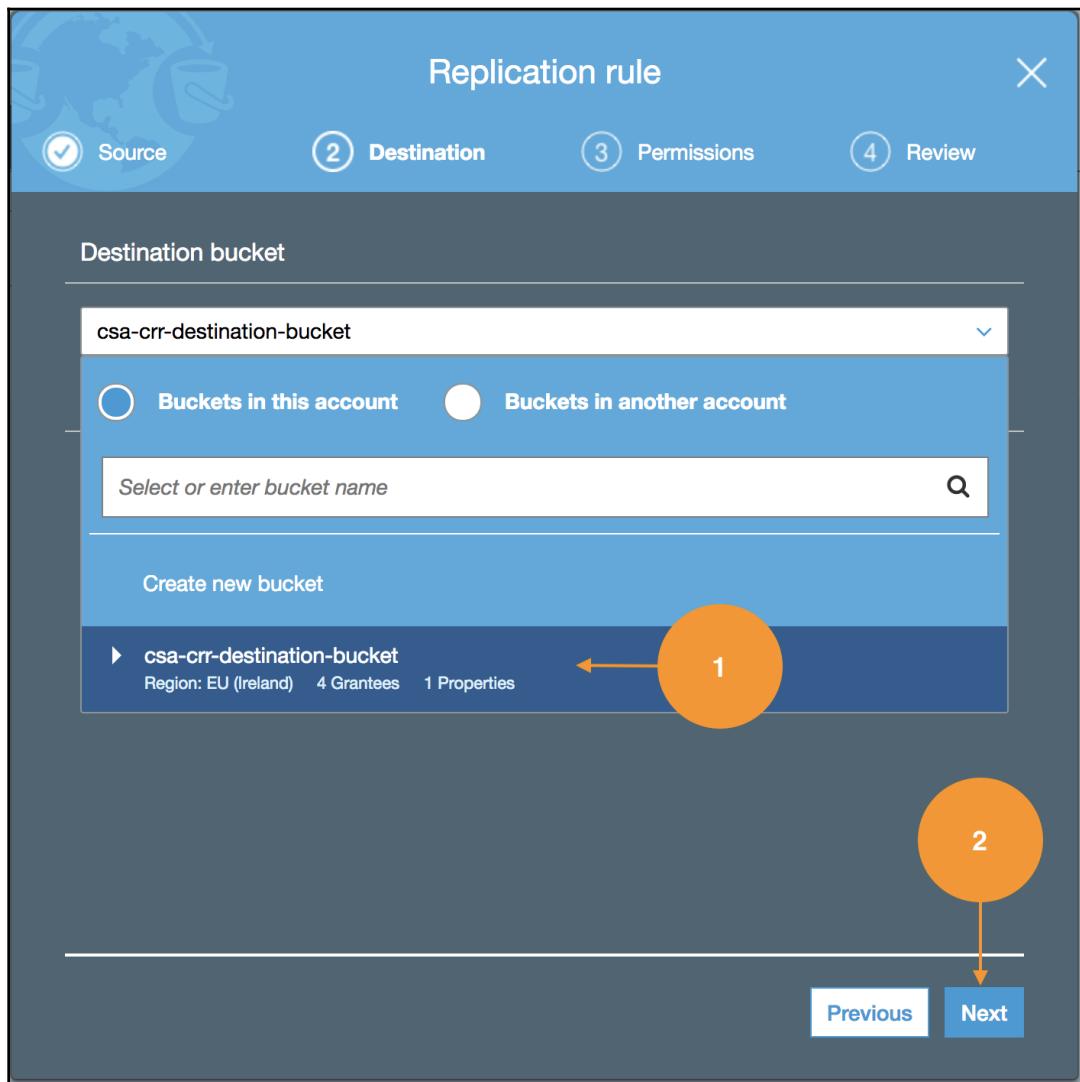
As a result, we have both buckets in the same account but in different regions:

Bucket name	Access	Region	Date created
csa-crr-destination-bucket	Not public *	EU (Ireland)	Apr 25, 2018 10:55:42 PM GMT-0500
csa-crr-origin-bucket	Not public *	US East (N. Virginia)	Apr 25, 2018 10:06:28 PM GMT-0500

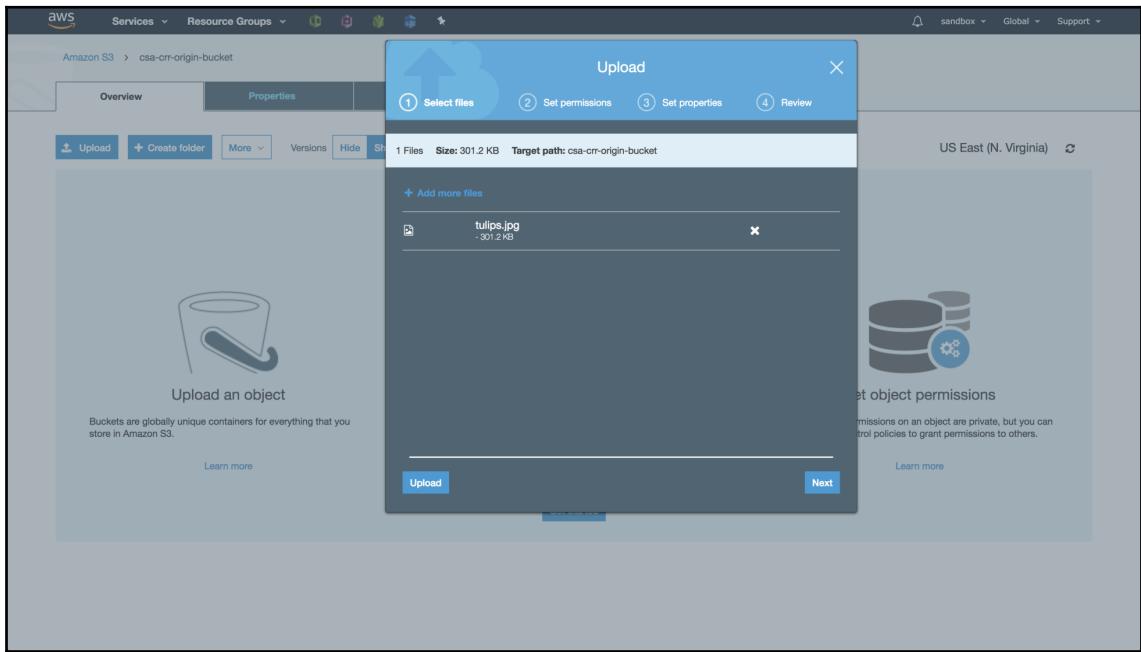
5. Let's configure automatic replication; we will need to select **Replication** under **Management** in the origin bucket (csa-crr-replication-bucket) to create a replication rule. This replication rule is adding file prefixes to filter the objects to be replicated:



6. It is possible to choose a different storage option and inherit the ownership of the objects to the bucket destination owner. To confirm the current screen, a new role must be created, which will assume the identity to gain write access to the resource:



7. The next step is to upload an image to the origin bucket:



8. Now, we can check the replication status. We can observe in (1) the corresponding ETag on the original image. ETag is an HTTP mechanism to perform cache validations; in AWS, S3 is used to generate a unique hash of the object and maintain a change registry. In this case, the replication preserves the original metadata, for example, the **Etag**, **Last Modified**, and **Version ID**.



On the replication status, we can see that the original bucket shows COMPLETED, while the destination bucket is a REPLICA, accounting that the copy has been made successfully (3).

The screenshot displays two side-by-side AWS S3 object details pages. Both pages show an object named 'tulips.jpg'.

Left Bucket (csa-rrr-origin-bucket):

- Key: tulips.jpg
- Size: 308429
- Expiration: N/A
- ETag: 0f082e027d410ec1dfae83a4148af6
- Last modified: Apr 25, 2018 11:42:42 PM GMT-0500
- Link: https://s3.amazonaws.com/csa-rrr-origin-bucket/tulips.jpg
- Replication Status: COMPLETED (circled with orange #3)

Right Bucket (csa-rrr-destination-bucket):

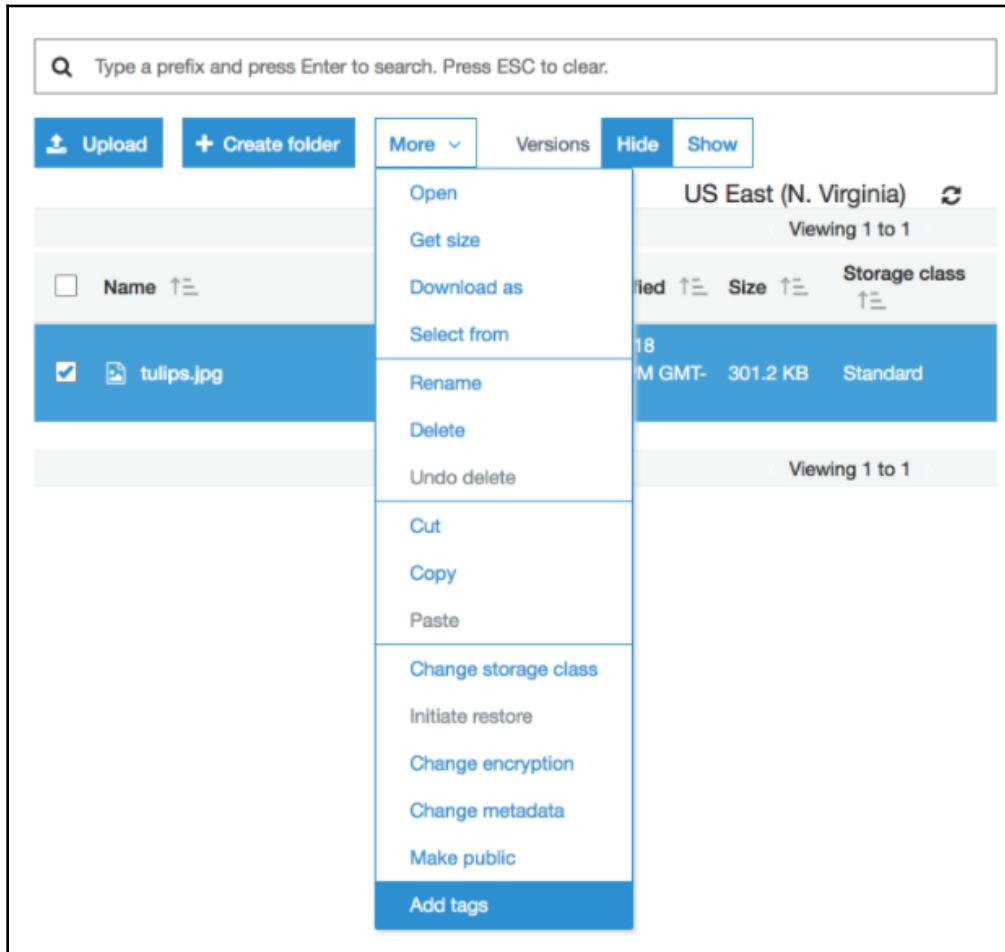
- Key: tulips.jpg
- Size: 308429
- Expiration: N/A
- ETag: 0f082e027d410ec1dfae83a4148af6
- Last modified: Apr 25, 2018 11:42:42 PM GMT-0500
- Link: https://s3.amazonaws.com/csa-rrr-destination-bucket/tulips.jpg
- Replication Status: REPLICA (circled with orange #3)

Both pages also show the object's properties and permissions, which are identical for both objects.

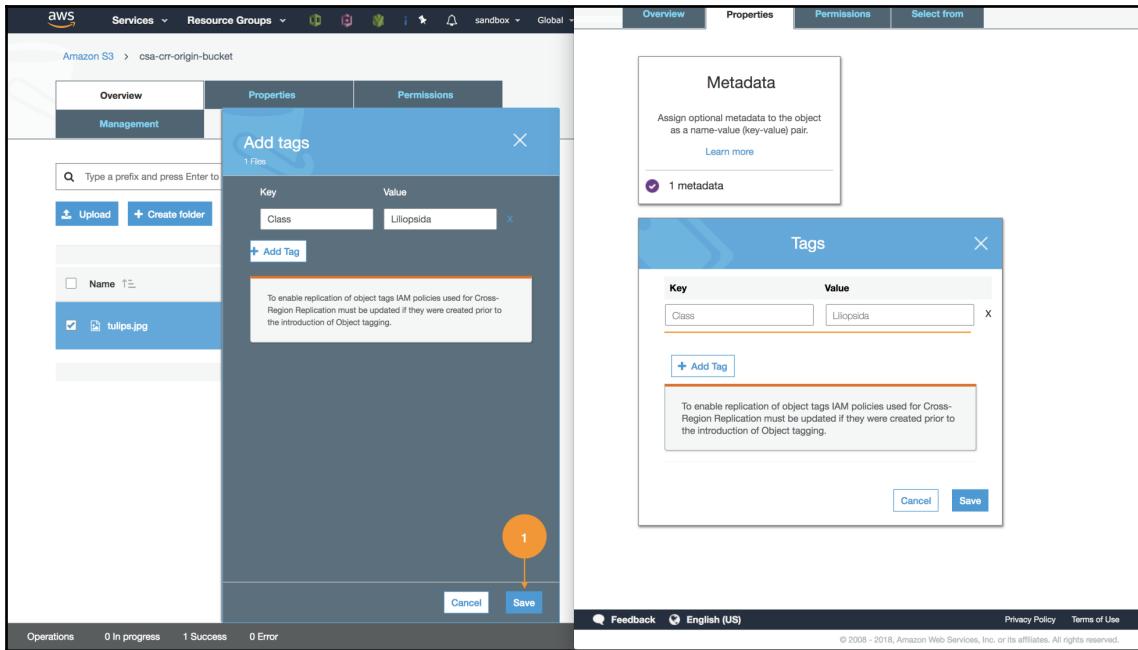
Something significant to note is that objects with the **REPLICA** status cannot be replicated again into another bucket; this operation can only be done by using S3 APIs with the **CopyObject** action, which involves a GET after a PUT operation.

Replicating tags

Tags are a way to add contextual information in the form of key=value to classify objects in S3. We only need to select the object checkbox and choose the option **Add tags** under **More**:



Changes propagate almost immediately asynchronously to the destination bucket:



Our applications can use this metadata as a means to query additional information about the object, such as the EmployeeID=123456 or Environment=Production. So, it becomes important to know the following limit; to overcome this situation, you will need to build a secondary index to save and query this metadata (for example a database table):

Hard Limit	
Tags per object	10

Replicating ACLs

By default, each object created in a bucket is owned by the root user of the account, this is known as the **resource owner**. S3 gives us the option to use two different models of permissions depending on the actions to be performed. To clarify this, use the next scenario.

A hosting company is proprietary of a storage bucket for their customers, and it wants to bring forward a confidential service in which the customer is the only owner of their objects, writing an ACL denegating all other user access, even root.

	ACLs	IAM Policy
Other accounts access	X	
Anonymous access (public)	X	
IAM user access		X
Explicit deny		X
Conditional permissions		X

1. Enable public access to the object modifying its ACL in the origin bucket:

Amazon S3 > csa-crr-origin-bucket

tulips.jpg Latest version ▾

Overview Properties Permissions Select from

Access for your AWS account

Account	Read object	Read object permissions	Write object permissions
<input type="radio"/> sandbox	Yes	Yes	Yes

Access for other AWS accounts

+ Add account	Delete	Read object	Read object permissions	Write object permissions

Public access

Group	Read object	Read object permissions	Write object permissions
<input type="radio"/> Everyone	-	-	-

2. In the **Public access** group, check the **Read object** checkbox:

The screenshot shows the AWS S3 console with the path 'Amazon S3 > csa-crr-origin-bucket' and the object 'tulips.jpg'. The 'Properties' tab is selected. On the right, a modal dialog titled 'Everyone' is displayed, containing a warning message: 'This object will have public access. Everyone will have access to one or all of the following: read this object, read and write permissions.' Below the message, there are checkboxes for 'Access to the object': 'Read object' (which is checked) and 'Access to this object's ACL' (which has two sub-options: 'Read object permissions' and 'Write object permissions'). At the bottom of the modal are 'Cancel' and 'Save' buttons.

3. Let's validate the ACL replicated into the destination bucket:

The screenshot shows the AWS S3 console with the path 'Amazon S3 > csa-crr-destination-bucket' and the object 'tulips.jpg'. The 'Properties' tab is selected. The 'Permissions' section shows the following configuration:

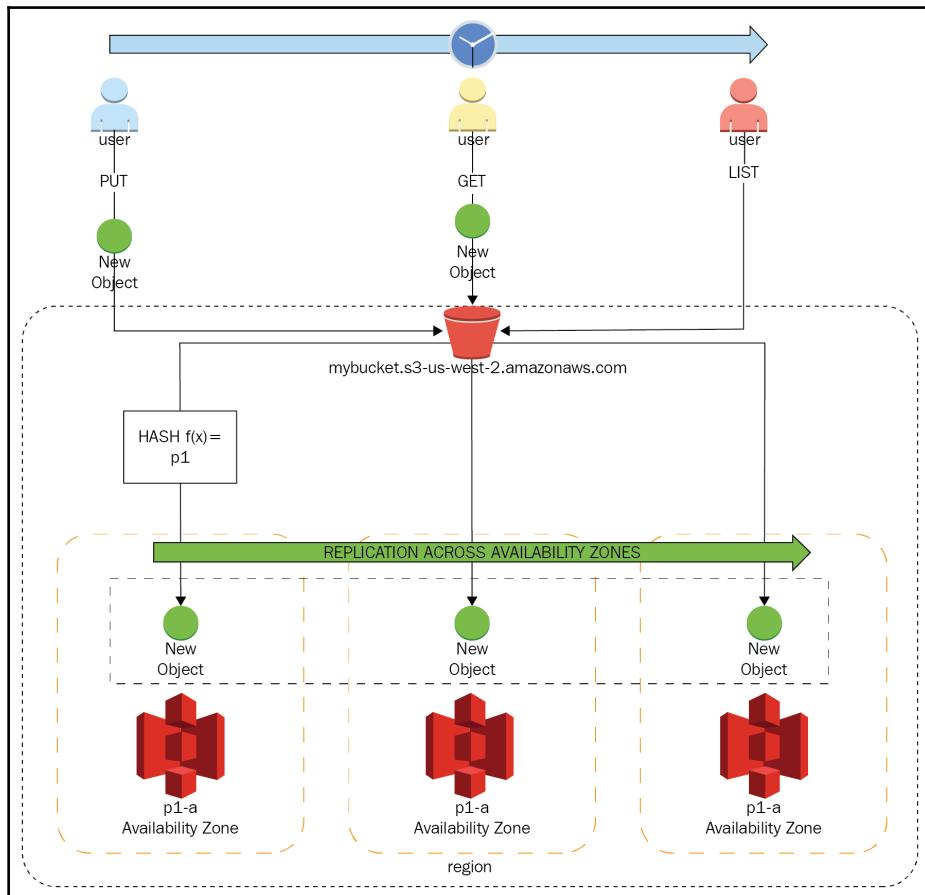
Account	Read object	Read object permissions	Write object permissions
sandbox	Yes	Yes	Yes
Everyone	Yes	-	-

An orange oval highlights the 'Read object' column for the 'Everyone' group row.

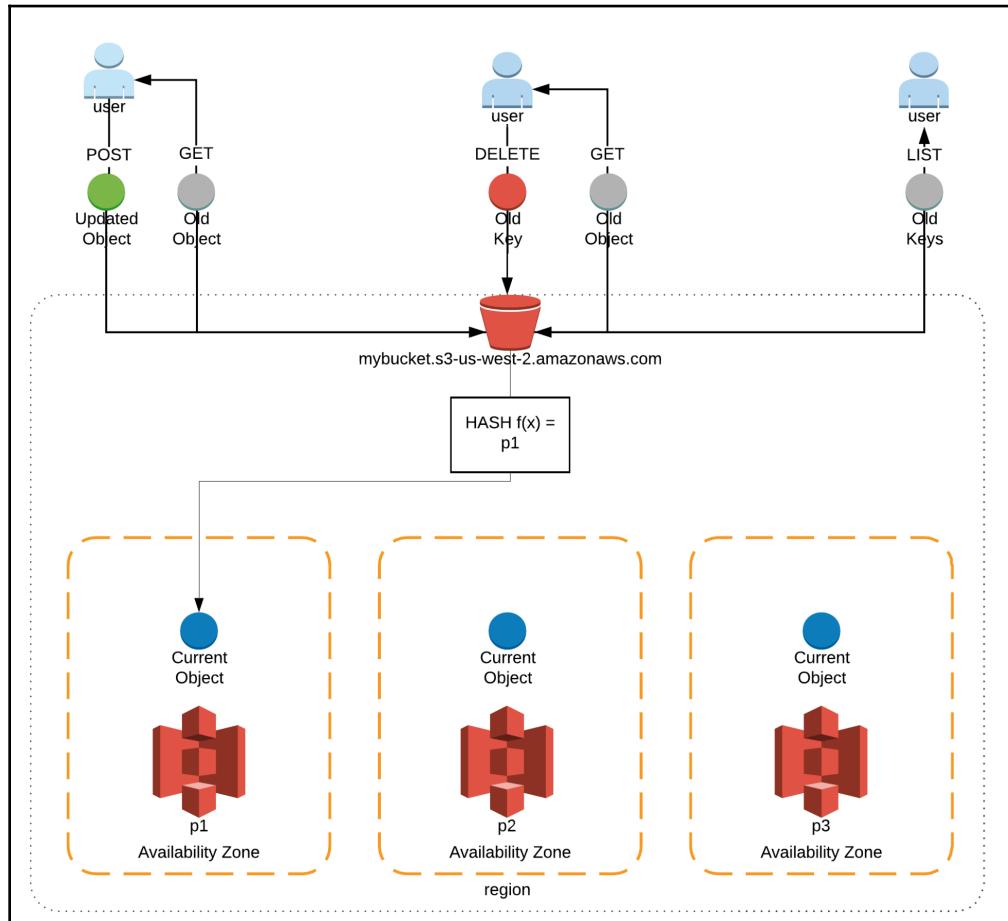
Now it is possible to access the object via the DNS name in both regions, it is relevant to mention that every object in S3 is opaque, being that each extension and file type is meaningless. S3 is a service optimized to store vast amounts of data with high availability and amazing durability, so it does not have hierarchical capabilities like a regular file system. Imagine S3 as a considerable HashMap data structure as a service.

Distributed nature of S3

Each new object works on a strong consistency model called read-after-write for which every object is written in at least three different AZs before the **SUCCESS** code is returned, avoiding stale reads on other clients. Nevertheless, if a client immediately does a **LIST** keys operation, they could have an inconsistent read, and the new object will be invisible to this partition:

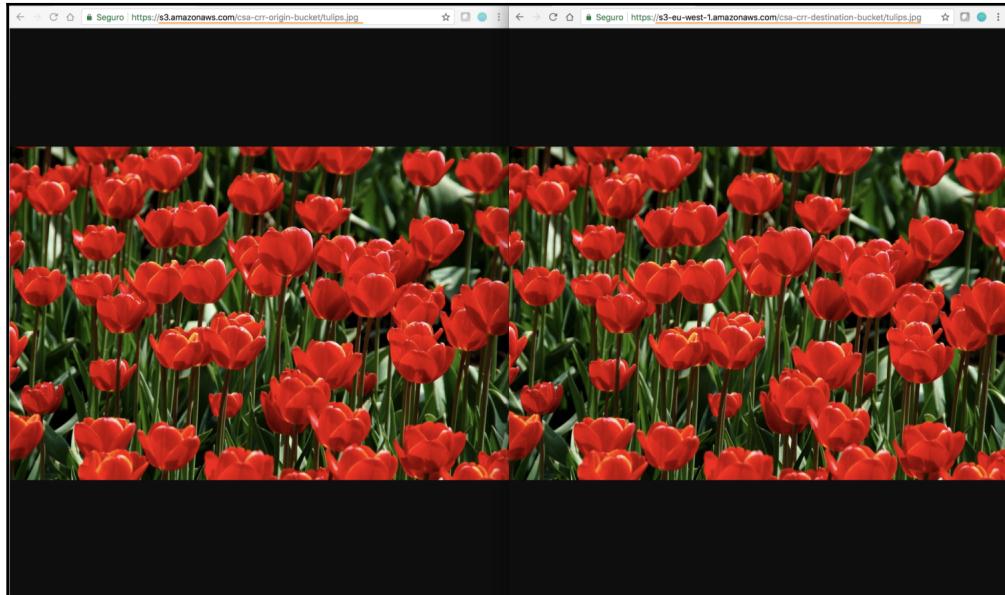


Each modified object has an eventual consistency model for **PUT** overwrites and **DELETES** on existing objects, leading to probable stale reads if a **GET** or **LIST** operation is immediately performed. This model is less strict and yields strong consistency in favor of availability, performance, and network partition tolerance (two AZs failing simultaneously):





Navigate both buckets and click the **Open** button so we can validate that the original object has now been replicated successfully.



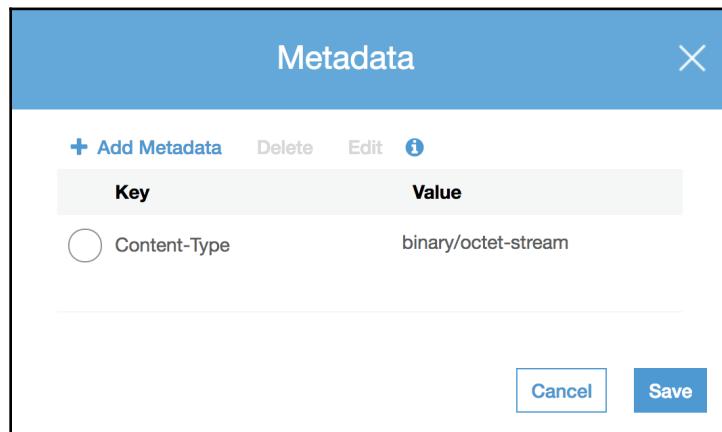
We can verify the object in both buckets (origin and destination) by selecting the object checkbox, **More | Open**, as shown in the previous screenshot.

Metadata replication

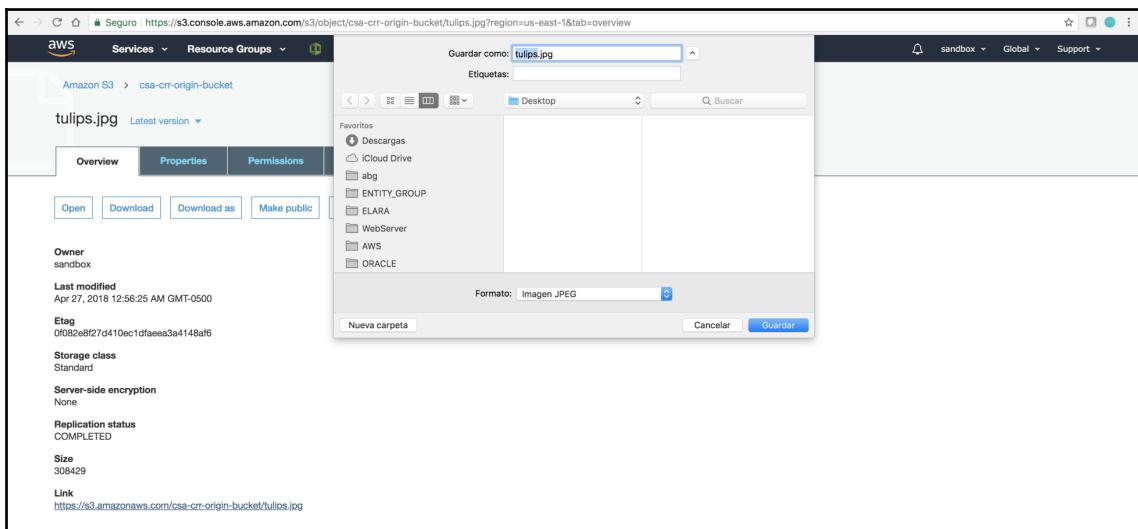
When we uploaded this `tulips.jpg`, S3 automatically read the file type inferring the MIME type `image/jpeg` and stored this information as a metadata. Every S3 object can have two kinds of metadata:

	Defined by the user	Example
System-Defined Metadata	Only some attributes	<code>Last-Modified</code>
User-defined metadata	Yes	<code>x-amz-meta-</code>

Let's modify the system-defined metadata Content-Type to binary/octet-stream so we can force a web browser to download:



The result of this is due to the fact that the web browser does not have a plugin to handle the kind of file received, and the default behavior is to download the file to the client:



Let's validate that the metadata replication was successful, using the destination bucket:

The screenshot shows the AWS S3 console interface. At the top, the navigation bar includes 'Services', 'Resource Groups', and various AWS service icons. The current view is for the 'csa-crr-destination-bucket' under 'Amazon S3'. A file named 'tulips.jpg' is selected, with its 'Latest version' being viewed.

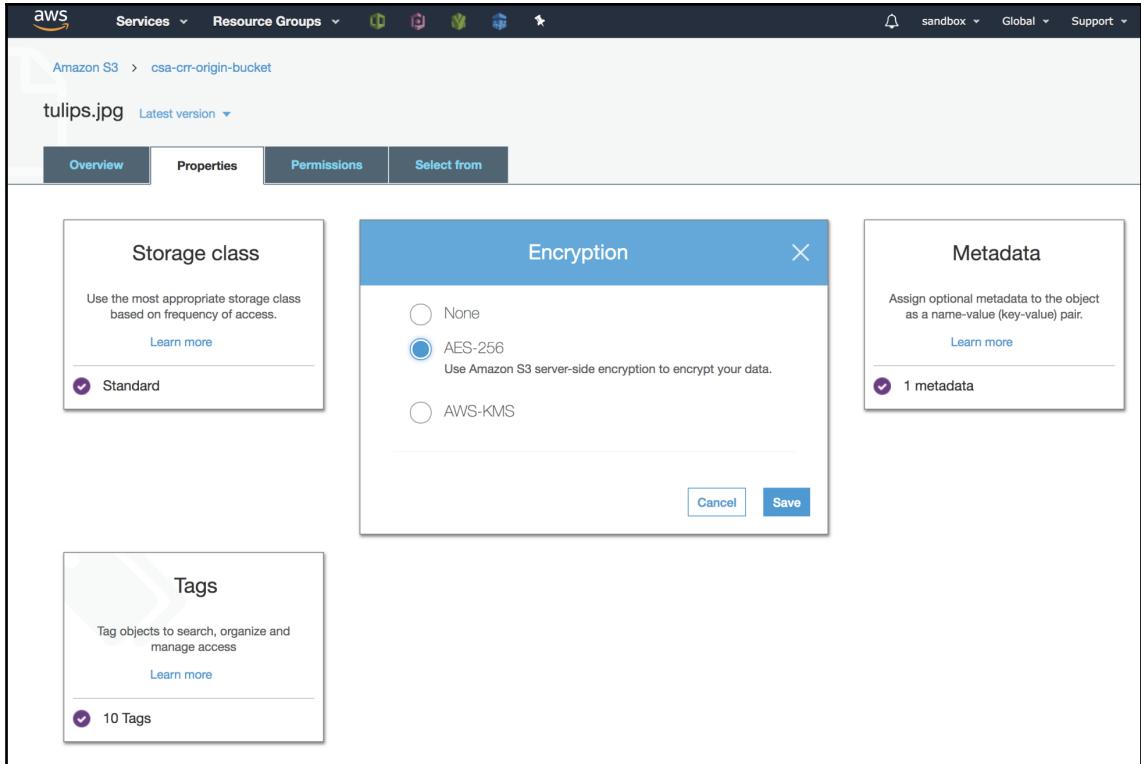
The main content area has four tabs: 'Overview' (selected), 'Properties', 'Permissions', and 'Select from'. Below these tabs are three cards:

- Storage class**: Describes using the most appropriate storage class based on frequency of access. It shows a 'Standard' option selected.
- Encryption**: Describes using encryption to protect data in-transit and at rest. It shows 'None' selected.
- Tags**: Allows tagging objects for search, organization, and access management. It shows '10 Tags' listed.

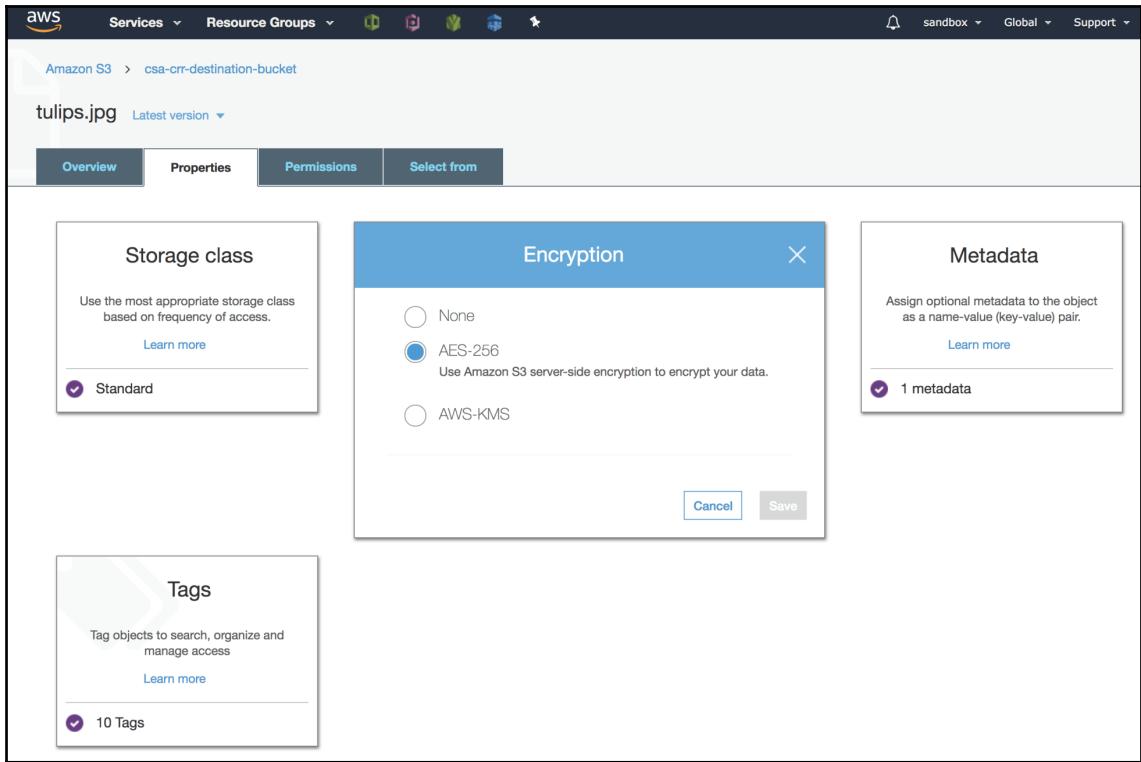
A modal window titled 'Metadata' is open over the main content. It contains a table with two columns: 'Key' and 'Value'. There is one entry: 'Content-Type' with the value 'binary/octet-stream'. The modal includes buttons for '+ Add Metadata', 'Delete', 'Edit', 'Cancel', and 'Save'.

Encryption replication

S3 can use a standard symmetric encryption algorithm on the server side (S3 servers), AES-256, to improve the at-rest security of our data and ensure information confidentiality:



Follow up the replication of this feature in the second region:





Service Limits (Certification objective)

Hard Limit	
Maximum object size	5 TB
Minimum object size	0 bytes
Put operation payload	5 GB



Objects with more than 100 MB of size must be uploaded using the multi-part API

Hosting a static website with S3 and CloudFront

The **Simple Storage Service (S3)** is a great option to host a static website because it gives us the following benefits:

- Hosts every file with a 99.99999999% durability [DONE]
- High availability with an SLA of 99.99%
- Low cost for usage calculated for every 1,000 requests, plus storage, and data transfer fees

Our first step is to provision a bucket where we will be hosting our website; for DNS resolution, our bucket name must have the same name as the domain name; for this purpose, I have registered the domain `s3websitehosting` using Route 53.

Latency is an important aspect, so we need to choose a region closer to our end users so we can measure response times for HTTP endpoints using [cloudping.info](#):

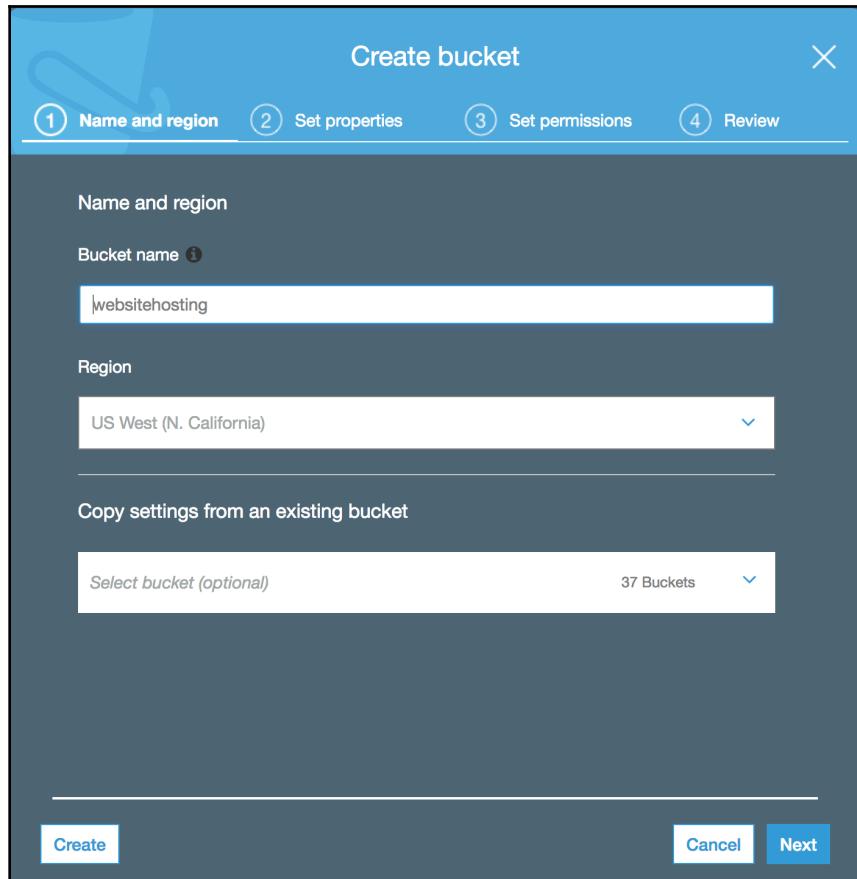
CloudPing.info

Amazon Web Services™ are available in several regions. Click the button below to estimate the latency from your browser to each AWS™ region.

Region	Latency
US-East (Virginia)	77 ms
US East (Ohio)	79 ms
US-West (California)	79 ms
US-West (Oregon)	93 ms
Canada (Central)	82 ms
Europe (Ireland)	154 ms
Europe (London)	559 ms
Europe (Frankfurt)	161 ms
Asia Pacific (Mumbai)	307 ms
Asia Pacific (Seoul)	199 ms
Asia Pacific (Singapore)	246 ms
Asia Pacific (Sydney)	307 ms
Asia Pacific (Tokyo)	169 ms
South America (São Paulo)	173 ms
China (Beijing)	500 ms
AWS GovCloud (US)	101 ms

HTTP Ping

We will choose California because it makes sense from my current location:



Once we have created our bucket, let's create our web app locally. An excellent choice is to use a generator to bootstrap the web application; personally, I like `yeoman.io`. Install the dependencies required by the project (Node.js and NPM). For this app, I will use the Angular JS generator:

```
npm install -g grunt-cli bower yo generator-karma generator-angular
```

Now, let's create the project directory:

```
mkdir webApp && cd $_
```

Lastly, specify the project name:

```
yo angular static_website
```

The terminal window shows the Yeoman generator process for creating an Angular static website. It starts with a welcome message: "Welcome to Yeoman, ladies and gentlemen!". The generator then asks several questions:

- ? Would you like to use Gulp (experimental) instead of Grunt? No
- ? Would you like to use SASS (with Compass)? No
- ? Would you like to include Bootstrap? Yes
- ? Which modules would you like to include? angular-animate.js, angular-cookies.js, angular-resource.js, angular-route.js, angular-sanitize.js, angular-touch.js

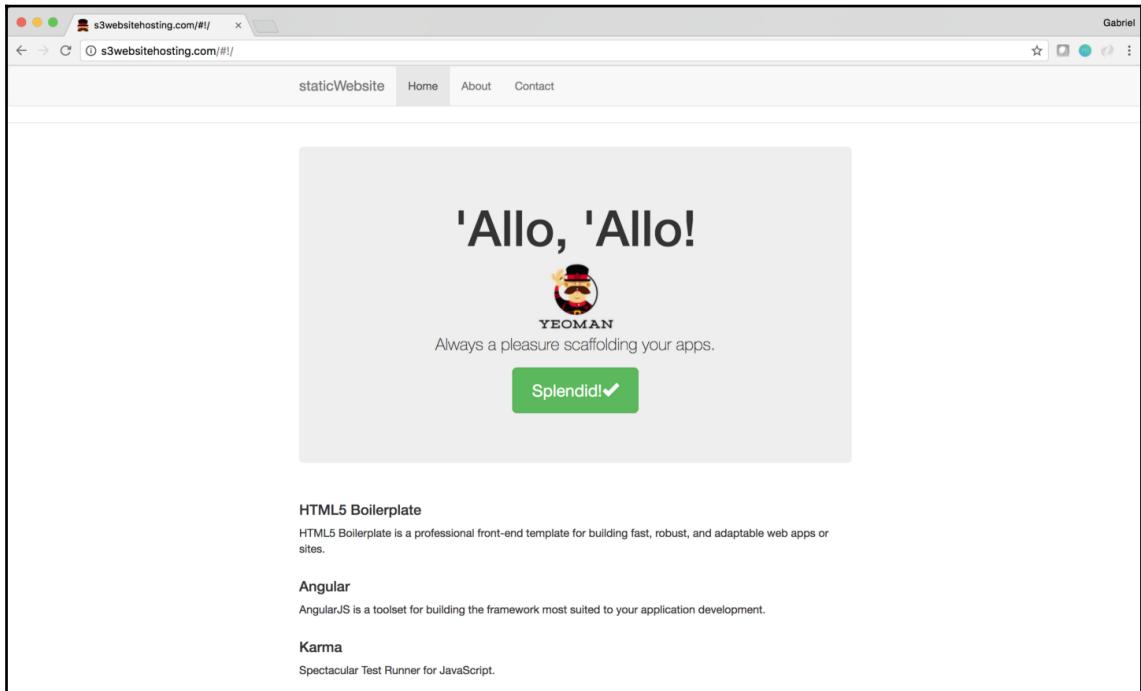
It then lists the files being created, including app/styles/main.css, create app/index.html, create bower.json, create .bowerrc, create package.json, create Gruntfile.js, create README.md, invoke angular:common:/usr/local/lib/node_modules/generator-angular/app/index.js, create .editorconfig, create .gitattributes, create .jssrc, create .jshintrc, create .yo-rc.json, create .gitignore, create test/.jshintrc, create app/404.html, create app/favicon.ico, create app/robots.txt, create app/views/main.html, create app/images/yohan.png, invoke angular:main:/usr/local/lib/node_modules/generator-angular/app/index.js, invoke angular:controller:/usr/local/lib/node_modules/generator-angular/app/index.js, create app/scripts/app.js, create app/controllers/main.js, create test/spec/controllers/main.js, invoke karma:app.

At the bottom, it says "I'm all done. Running `bower install & npm install` for you to install the required dependencies. If this fails, try running the command yourself."

You must receive a message output such as *done, without errors*; if not the case, fix any dependencies issues regularly associated with dependencies or versions not being upgraded. After this, try again. You could choose other generators such as JQuery.

Let's execute grunt server to run the web application locally, and grunt-only to prepare the distribution of the web app running all the pipeline tasks configured for this generator.

A great advantage of using these kinds of frameworks is the ability to use workflows that maximize our productivity, because multiple tasks are performed with our code, such as compression and minification of the files and images. When we use CloudFront to accelerate our website, using Edge Locations is essential to have a strategy that allows us to invalidate previous versions of our application; a very efficient way is to version every file that needs to be changed, so CloudFront can identify when a change has been made and put the latest version in the cache:



Now, let's proceed to upload our website to the created S3 bucket (use the `webApp/dist` directory as our productive version):

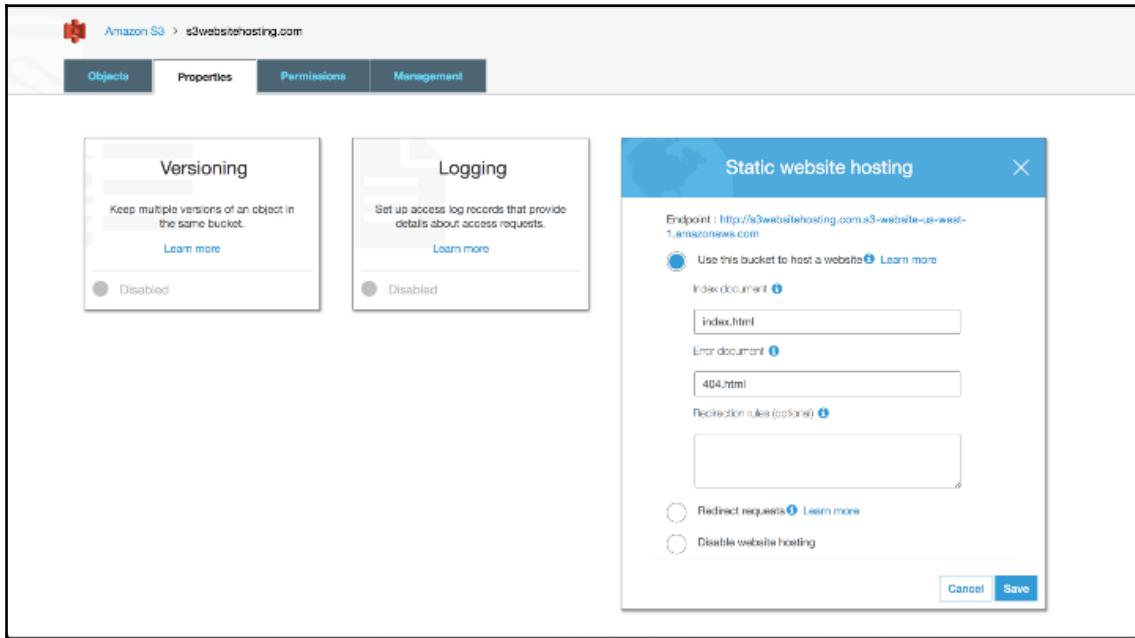
```
aws s3 sync . s3://s3websitehosting --acl public-read
```

The previous command finds any differences between origin and destination to perform synchronization; in this case, the origin is the actual work directory, and the destination is the bucket `s3websitehosting`.

Now, let's update the bucket configuration to enable static website hosting:

```
aws s3 website s3://s3websitehosting.com -- index-document index.html --  
error-document 404.html
```

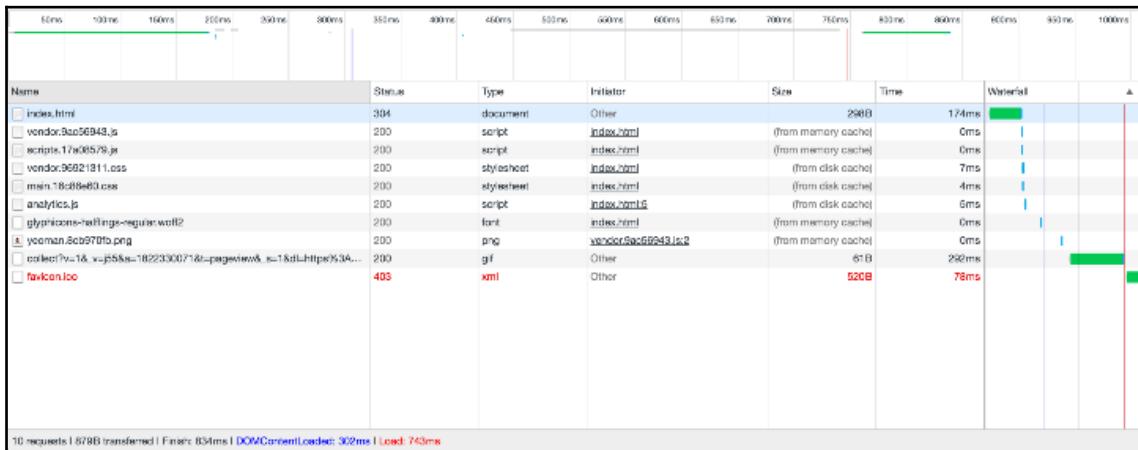
We can make sure these changes have been applied (this can be done via the console too):



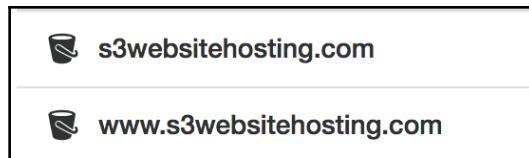
It is available as a low-level CLI called S3-API client and is much more robust at making S3 API calls where you can; for example, specifying redirect rules such as HTTP to HTTPS traffic redirection.

We now have a published website searchable from the following URL: <https://s3-us-west-1.amazonaws.com/s3websitehosting.com/index.html#/>.

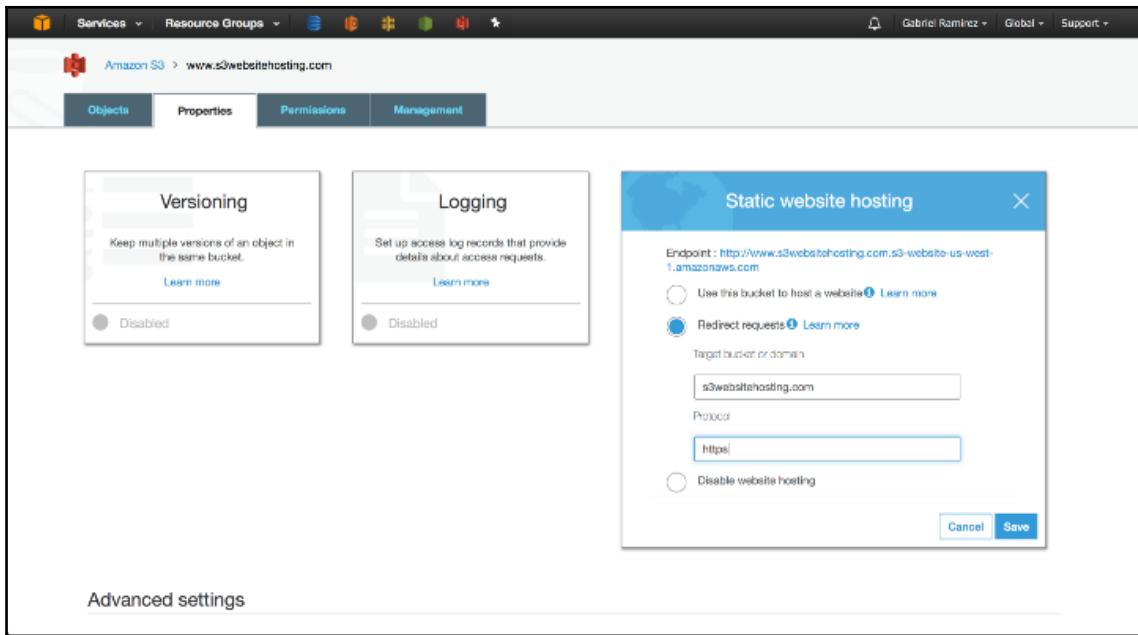
Now that we have uploaded our files, it is a good idea to make a performance benchmark; in my case, I have **743 ms** as the response time from my current location:



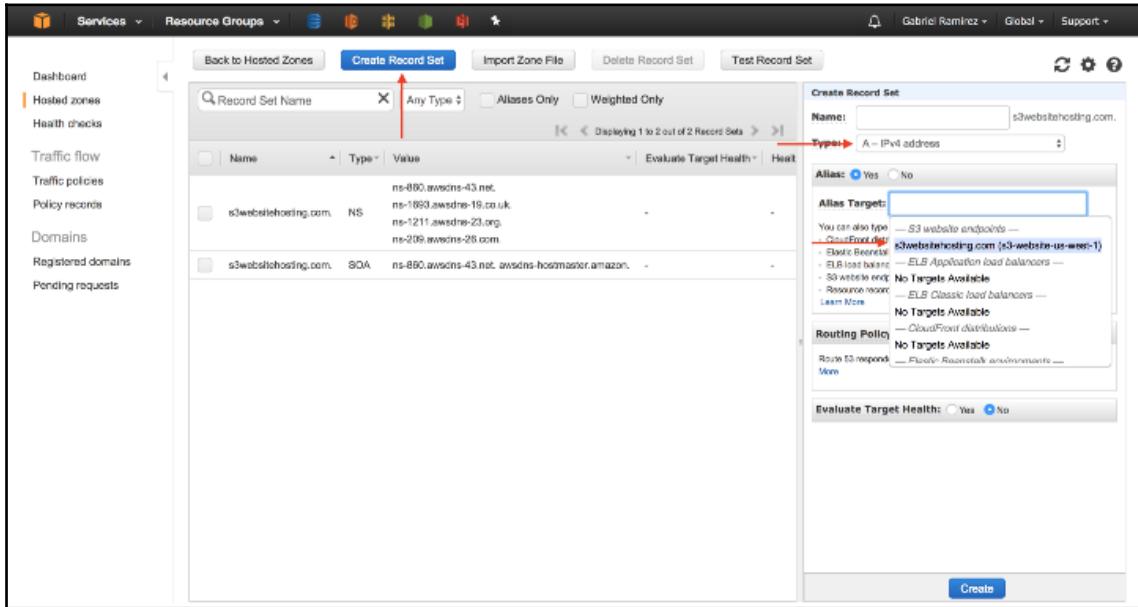
To establish routing rules on our website and be able to redirect requests from <http://www.s3websitehosting.com> to <http://s3websitehosting.com>, we will need two buckets. Only one of them will host our website, and the other will act as an alias to the origin:



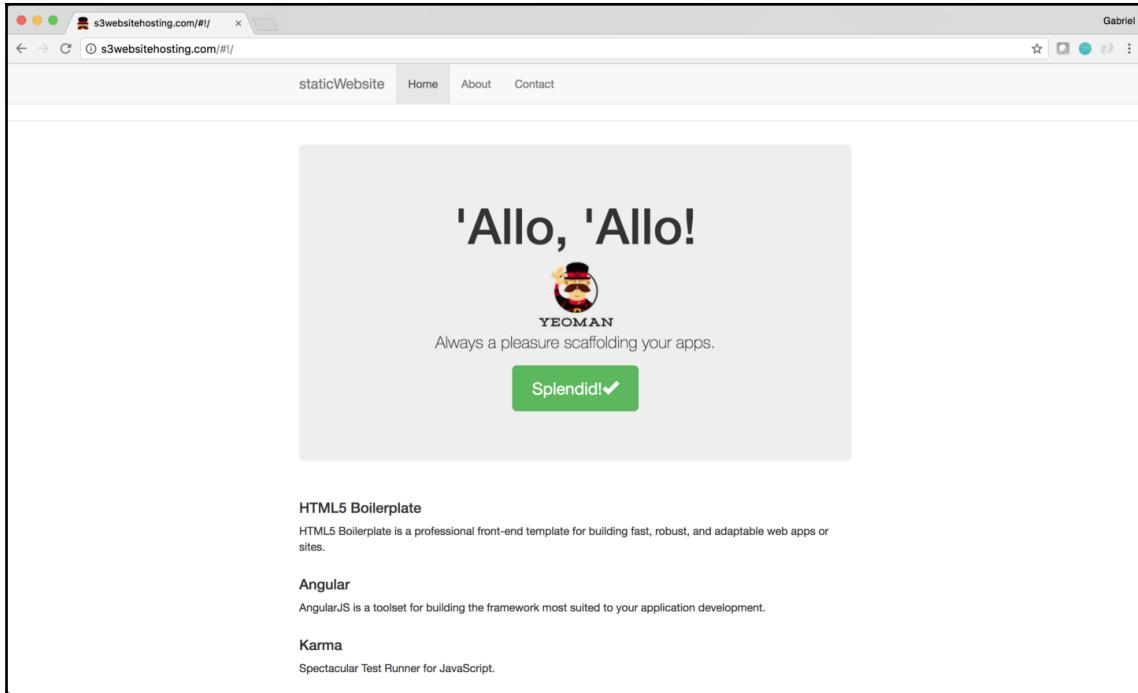
In this case, the www prefixed bucket will point to <https://www.domain.com/>:



Proceed to **create alias registries** so we can associate our domain name to the static website in the S3 bucket; in **Route 53**, we must edit the public-hosted zone file so we can create an alias registry:



Route 53 DNS propagation is pretty fast, and, in just a matter of seconds, the database records must be propagated and the service will be able to solve the domain; let's check in a web browser by querying the name:



However, if I look up www.s3websitehosting.com we will have an error:



This can be resolved very easily by establishing another alias record, but this time by adding the `www` prefix (repeat the previous process, but now use `www` as the **Name** value):

Create Record Set

Name: `www` .s3websitehosting.com.

Type: A – IPv4 address

Alias: Yes No

Alias Target: s3-website-us-west-1.amazonaws.com

Alias Hosted Zone ID: Z2F56UZL2M1ACD

You can also type the domain name for the resource. Examples:
- CloudFront distribution domain name: d11111abceef8.cloudfront.net
- Elastic Beanstalk environment CNAME: example.elasticbeanstalk.com
- ELB load balancer DNS name: example-1.us-east-1.elb.amazonaws.com
- S3 website endpoint: s3-website.us-east-2.amazonaws.com
- Resource record set in this hosted zone: www.example.com

[Learn More](#)

Routing Policy: Simple

Route 53 responds to queries based only on the values in this record. [Learn More](#)

Evaluate Target Health: Yes No

Create

If we filter the alias registries, they must have alike entries but with their domain name:

Record Set Name Any Type Aliases Only Weighted Only

Displaying 1 to 2 out of 2 Record Sets

<input type="checkbox"/>	Name	Type	Value	Evaluate Target Health	H
<input type="checkbox"/>	s3websitehosting.com.	A	ALIAS dwtyj2rqtpuli.cloudfront.net. (z2fdtndataqyw2)	No	-
<input type="checkbox"/>	www.s3websitehosting.com.	A	ALIAS dwtyj2rqtpuli.cloudfront.net. (z2fdtndataqyw2)	No	-

This will generate an automatic redirect for queries to `www` for the APEX of the hosted zone. Now, it is time to accelerate our website by using a CloudFront web distribution in front of it to our web page that will be copied to multiple Edge Locations and improve the user experience dramatically.

Choose CloudFront from the **Services** console option, then **Create Distribution**, and for delivery method, select **Web**. Use the following values:

The screenshot shows the AWS CloudFront 'Create Distribution' configuration screen. Key settings include:

- Alternate Domain Names (CNAMEs):** s3websitehosting.com, www.s3websitehosting.com (highlighted with red arrows)
- SSL Certificate:** Default CloudFront Certificate (*.cloudfront.net) (highlighted with red arrows)
- Default Root Object:** index.html (highlighted with a red arrow)
- Comment:** s3websitehosting.com (highlighted with a red arrow)
- Distribution State:** Enabled

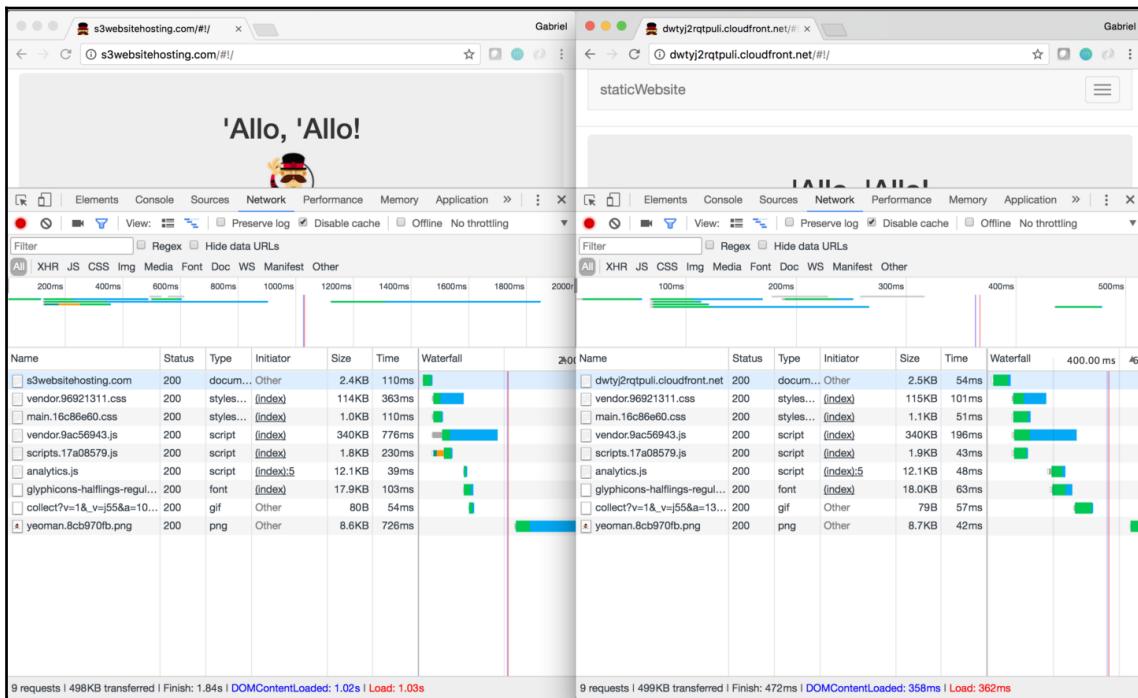
Wait until the distribution status has changed to **Deployed**:



Copy the DNS-generated domain name and test it in your web browser; it must have a structure similar to XXXXXXXXXXXX.cloudfront.net.

Ready! Now that you have your website cached in multiple Edge Locations around the world when a DNS query arrives, the service will determine the best location candidate to serve this request. Think of CloudFront as a global network load balancer.

This image shows a benchmark between the origin (S3 bucket) versus the web distribution (CloudFront):



Now we need to associate a new alias record in the hosted zone, but this time to the CNAME of the CloudFront web distribution and not the S3 bucket name:

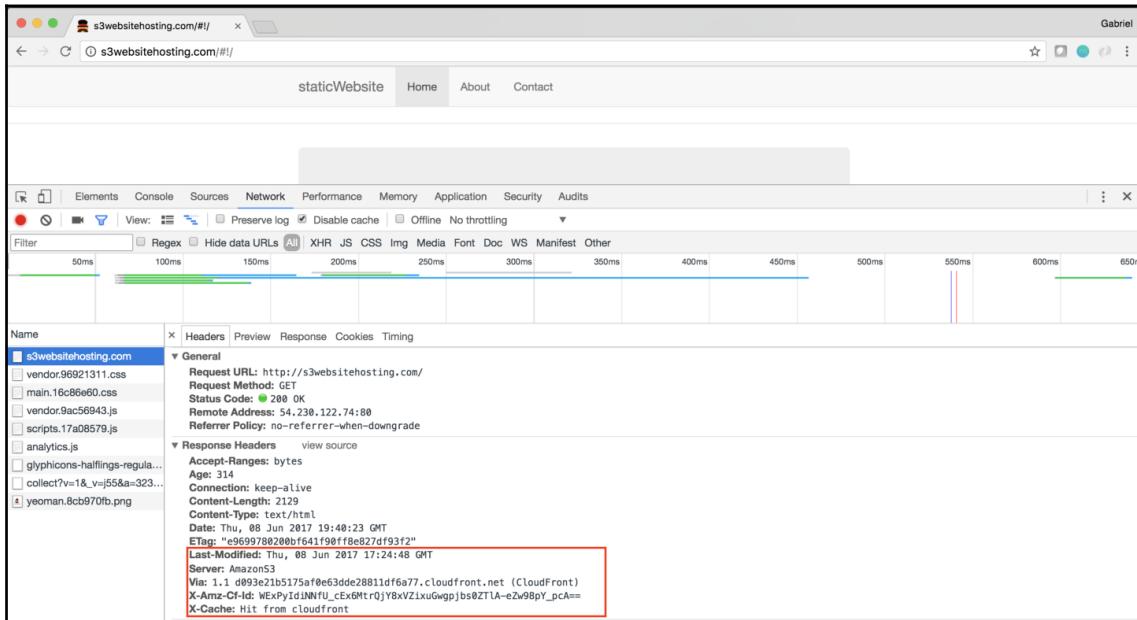
The screenshot shows the AWS Route 53 service dashboard. On the left, there's a sidebar with options like Dashboard, Hosted zones (which is selected), Health checks, Traffic flow, Traffic policies, Policy records, Domains, Registered domains, and Pending requests. The main area has tabs for Back to Hosted Zones, Create Record Set, Import Zone File, Delete Record Set, and Test Record Set. A search bar at the top says "Record Set Name" with "www.s3websitehosting.com." entered. Below it, there's a dropdown for "Any Type" and checkboxes for "Aliases Only" and "Weighted Only". A table lists four record sets, with the last one being edited. The edited record set is for "www.s3websitehosting.com." with type "A" and value "ALIAS s3-website-us-west-1.amazonaws.com. (z2f5)". The "Type" dropdown shows "A - IPv4 address" and the "Alias" checkbox is checked. The "Alias Target" dropdown is empty. The "Routing Policy" section shows "Route 53 responds" and "More" with a note "Download route in this hosted zone". At the bottom, there's an "Evaluate Target Health" section with "Yes" and "No" radio buttons.

Proceed to validate the alias registries updated with the CNAME:

This screenshot shows the same AWS Route 53 interface, but the search results have changed. The table now displays two records. The first is for "s3websitehosting.com." with type "A" and value "ALIAS dwtyj2rqtpuli.cloudfront.net. (z2fdtndataqyw2)". The second is for "www.s3websitehosting.com." with type "A" and value "ALIAS dwtyj2rqtpuli.cloudfront.net. (z2fdtndataqyw2)". The "Type" dropdown now shows "ALIAS" and the "Aliases Only" checkbox is checked. The rest of the interface remains the same, including the "Evaluate Target Health" section at the bottom.

Performing a new test, we can check the developer tools in our web browser to see that the web app content is being served by the cache servers:

X-Cache: Hit from cloudfront



Summary

In this chapter, we learned the most important concepts about the AWS global infrastructure, the related aspects when choosing a geographic region, how Availability Zones work, and how to create a web distribution using CloudFront.

We hosted a static website in an S3 bucket, studied the consistency model under S3, and interacted with the CLI interface. We also implemented a disaster recovery solution replicating data between buckets using two AWS Regions. We tackled security aspects related to S3, such as ACLs, IAM policies, and versioning.

It is up to the student to recover data by simulating a loss of integrity in the origin bucket by deleting the original object, all the available versions, and the `DELETE_MARKER` (first delete the original object, then go to **Versions | Show**). Recover the object from the secondary region; you can help yourself using the `aws s3 cp` command.

Further reading

- **Overview of Security Processes Whitepaper:** <https://aws.amazon.com/whitepapers/overview-of-security-processes/>
- **Data Center Security Controls:** <https://aws.amazon.com/compliance/data-center/controls/>

3

Elasticity and Scalability Concepts

Public cloud providers often have the ability to scale their services indefinitely to provide unlimited computing, storing, and networking functionalities. In this chapter, we will cover basic failure scenarios and how to overcome them by using vertical scaling and the virtual management of resources. We will also look at how to manage persistent configurations and failover by using decoupled resources, such as virtual IPs.

We will introduce EBS as a resilient service by managing the persistent state of our data with high durability, and look at how it can be used as a business continuity model for disaster recovery. We will explain the vertical scaling of an RDS database and will provide you with a deep understanding of how availability is managed and calculated.

The following topics will be covered in this chapter:

- Availability metrics and their associated dependencies
- Implementing vertical scaling with EC2 and RDS
- Image versioning and image life cycles, illustrated with snapshots
- How to bootstrap a web server with scripting
- Provisioning persistent EBS disks and operating them via the CLI
- Performing a stress test to benchmark different instance sizes

Technical requirements

We will use the AWS CLI in this chapter's exercises in order to install and configure the clients. Detailed information can be found at <https://aws.amazon.com/cli/>. Having a basic knowledge of the Linux shell is encouraged, and familiarity with the Terminal will be useful.

Sources of failure

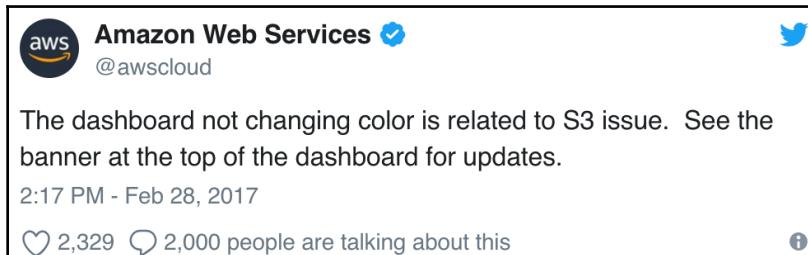
Once, I was in Lima, Peru, heading an AWS architecting course. During their first practice, the students started to complain about their labs issuing 500 errors, timeouts, and general inconsistencies. After a closer look, I realized that something uncommon was happening. I opened my personal production web console and navigated directly to the alerts center, and I found a large number of events involving S3, and many other systems, in northern Virginia. In the media, you could read messages like the following:

- *No, you're not crazy. Part of the internet broke* (<https://www.cnet.com/news/amazon-web-services-aws-s3-service-problem/>)
- *How a typo took down S3, the backbone of the internet* (<https://www.theverge.com/2017/3/2/14792442/amazon-s3-outage-cause-typo-internet-server>)
- *Amazon S3 Outage Has Broken a Large Chunk of the Internet* (<https://www.forbes.com/sites/ryanwhitwam/2017/02/28/amazon-s3-outage-has-broken-a-large-chunk-of-the-internet/#51ebb502c467>)

The @awscloud Twitter account disclosed the information in the following screenshot:



The Personal Health Dashboard in AWS was affected, too:



So many systems are dependent on S3, which is a central part of AWS operations. **Simple Storage Service (S3)** was designed to provide high levels of availability, and it had never had an outage like this before; at that moment, Moore's law was made present to teach us valuable lessons.

The cause

After a few days, AWS announced the root cause of the issue, which gave us a better idea of the real problem:

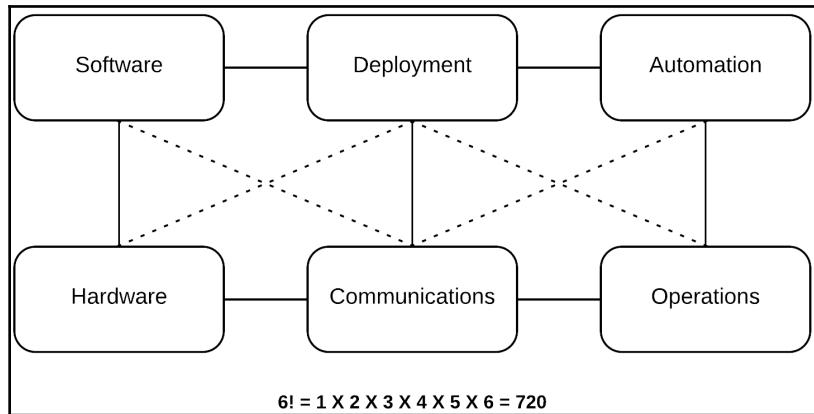
"At 9:37 am PST, an authorized S3 team member using an established playbook executed a command which was intended to remove a small number of servers for one of the S3 subsystems that is used by the S3 billing process. Unfortunately, one of the inputs to the command was entered incorrectly and a larger set of servers was removed than intended."

The cause was a typo - a deployment mistake from one of the site engineers that was made while they were releasing a change into production. How can this kind of regression be fixed in the future?

"We are making several changes as a result of this operational event. While removal of capacity is a key operational practice, in this instance, the tool used allowed too much capacity to be removed too quickly. We have modified this tool to remove capacity more slowly and added safeguards to prevent capacity from being removed when it will take any subsystem below its minimum required capacity level. This will prevent an incorrect input from triggering a similar event in the future."

Failure should be our teacher; as Thomas A. Edison said, "I have not failed. I've just found 10,000 ways that won't work." Throughout this book, I will teach you how to build resilient systems that can tolerate many lines of failure.

Several interrelated factors can affect availability, as shown in the following diagram:



By looking at the preceding diagram, you can grasp how many combinations need to be tested to fully cover every combination (six factorial). Producing a recovery strategy for each failure path is a non-deterministic and expensive; to make things easier, we can make use of **Recovery Oriented Computing (ROC)**, which aims to find the primary failure spots and remediate them as soon as possible by performing exhaustive testing.

In this chapter, we will focus on ways to prevent data loss by snapshotting our volumes, using AMI versioning (to be fully operative in the event of an instance loss), and using elastic network interfaces (which help us to mask failing instance, by moving the interface to a higher capacity or replacement failover instance).

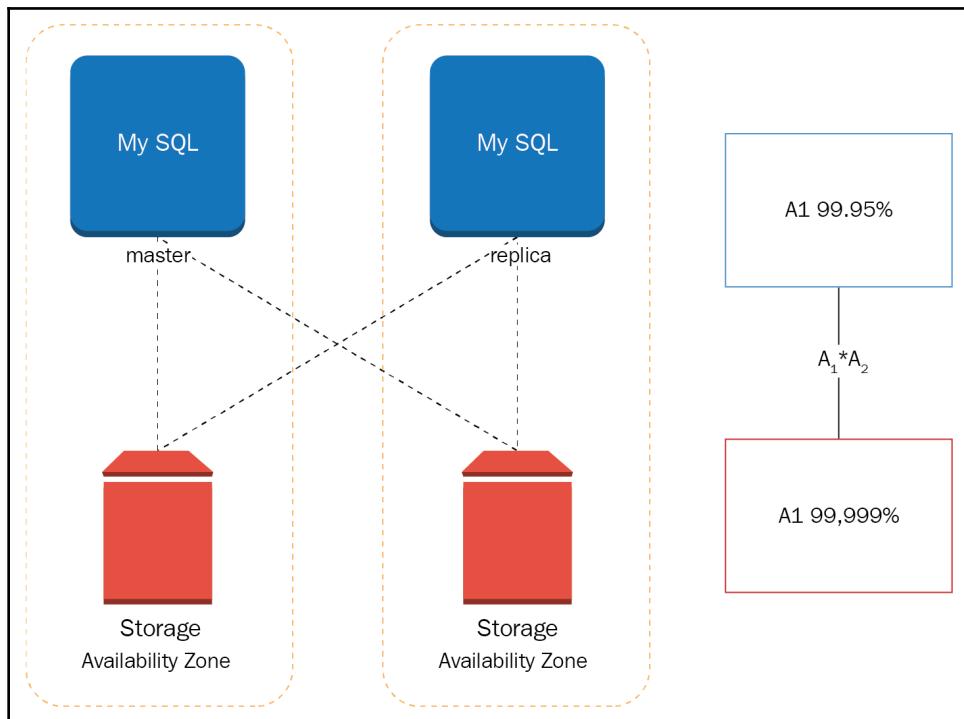
Dividing and conquering

If you have a complex problem, decompose it into individual, manageable parts; isolate them, and focus on unique strategies to avoid failure. This way, a component crash will not affect other components since you will be avoiding dependencies and providing insulating mechanisms to manage communication between layers; this is an architectural best practice.

Serial configuration

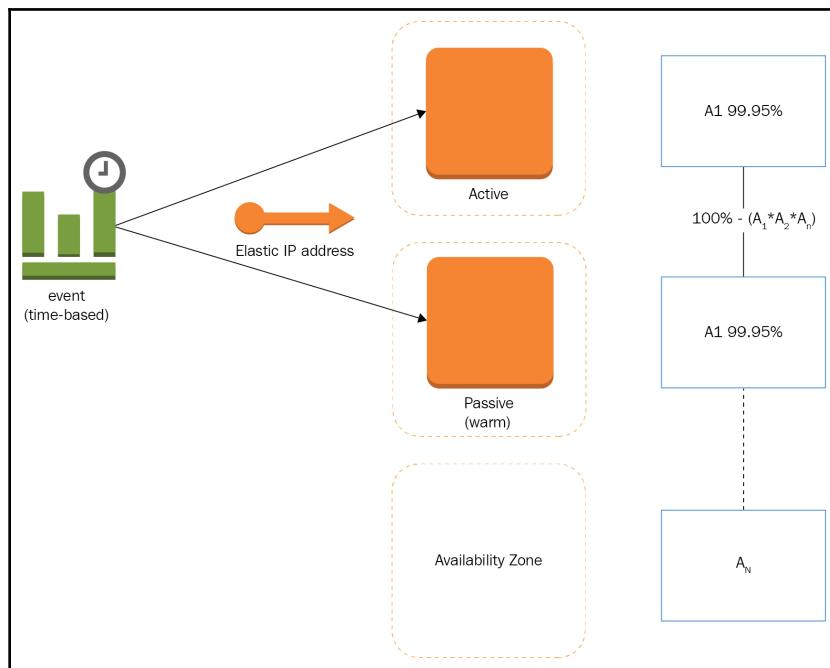
Serial configuration is a technique that provides a reference to calculate coupling when they have strong dependencies between each other. In order to understand the availability of two components, we have to calculate the individual value for each component; for example, a database instance (RDS) and its persistent storage (EBS). A reliable service would have to have both components fully operational, producing a serial model.

The following diagram shows how a high dependency from the database server, downstream of the EBS volume, can be calculated:



Parallel configuration

We can make sure that we design our architecture with component redundancy in mind (for example, with AZs). If the components are independent and do not share anything (the **shared nothing** approach), even the network, then it becomes possible to increase the system availability. To this end, we can have an orchestrator node, or an arbiter that supervises the system. A quorum can be agreed upon between the nodes to fail over the next redundant node in service. The following diagram shows an example of a third node observing the system and reacting to eventualities (CloudWatch monitoring):



Reactive and proactive scalability

Scalability is the ability to add capacity to a system dynamically, without modifying its architecture. There are two types of scalability, as follows:

- Horizontal
- Vertical

Proactive scalability is when automatic means are in place to compensate the system capacity via monitoring (before the event occurs).

Reactive scalability is when the event has occurred and manual intervention is put in place to compensate the capacity (when the alarm threshold has been reached).

Horizontal scalability

Horizontal scalability uses redundancy (homogeneous nodes) to avoid **single points of failure (SPOFs)**. This is the preferred way to increase availability, but it sometimes requires application re-engineering to externalize data sources in a stateless fashion.

Vertical scalability

Vertical scalability consists of a change in hardware specifications in order to acquire more RAM, a higher storage capacity, more virtual CPUs, and any performance improvements.

The **Elastic Compute Cloud (EC2)** has different instance types, families, and sizes, which allows for the vertical scalability of a single compute node, as shown in the following screenshot:

Name	API Name	Memory	vCPUs
M5 General Purpose Large	m5.large	8.0 GiB	2 vCPUs
M5 General Purpose Extra Large	m5.xlarge	16.0 GiB	4 vCPUs
M5 General Purpose Double Extra Large	m5.2xlarge	32.0 GiB	8 vCPUs
M5 General Purpose Quadruple Extra Large	m5.4xlarge	64.0 GiB	16 vCPUs
M5 General Purpose 12xlarge	m5.12xlarge	192.0 GiB	48 vCPUs
M5 General Purpose 24xlarge	m5.24xlarge	384.0 GiB	96 vCPUs

In the preceding screenshot, you can see how the different instance sizes scale almost directly with their RAM capacity (**large**, **xlarge**, **2xlarge**, and so on).

The instances pertaining to the **M** family are designed for multipurpose scenarios; these instances have a balance between the CPU, memory, and network throughput. They are candidates for workloads as web servers, small to medium databases, development environments, and even enterprise applications.

Exercise

Now, we will launch an `m1.small` instance and set up an Apache web server. Execute **ApacheBench (ab)** to perform a performance test, measure the results, and then change the underlying hardware to an `m4.large` instance, and compare both of the results.



For more information about the LAMP stack on an Amazon Linux image, follow the tutorial at <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/install-LAMP.html>.

We will be using the command-line interface because of its concise nature. The API action for EC2 is called `RunInstances`. It is advised that you familiarize yourself with the main API actions of every API of the AWS core services.

Instances are launched from an instance template, called a virtual image (or an **Amazon Machine Image (AMI)**, in the Amazon lingo). These images follow an incremental approach, in which we add customization and bake another one to be reused, as follows:

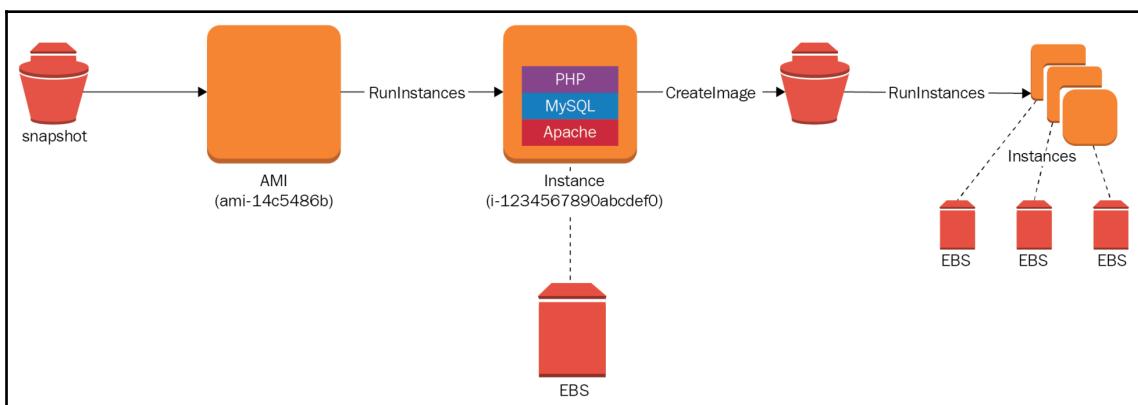
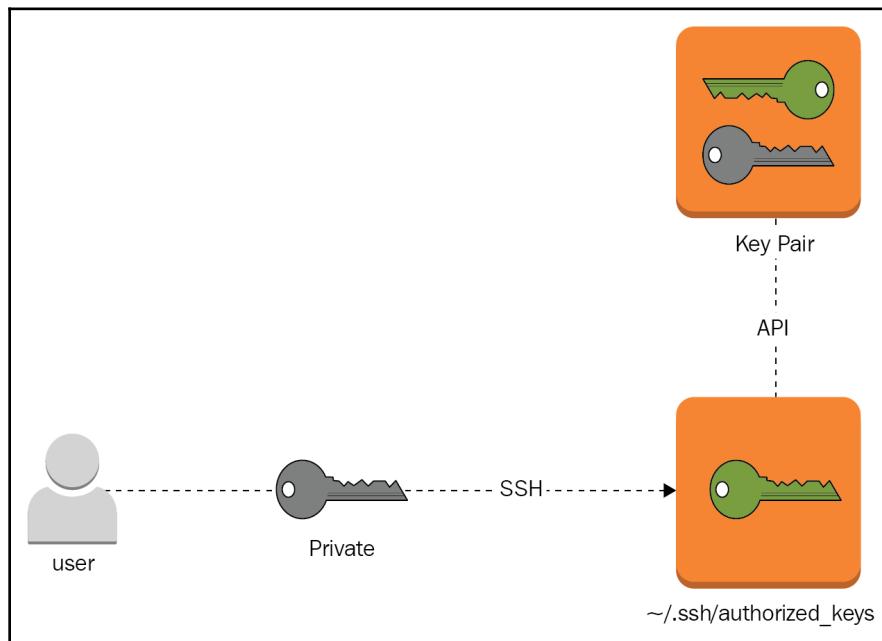


Image versioning is a best practice for immutable infrastructures, where new releases can be upgraded via full system images (OS, technology stacks, and code) and quickly released into production or controlled environments. Images play a critical role in disaster recovery strategies, where you can replace services and applications at well-known points in time. Amazon Linux images are a good place to start, because these images are hardened, optimized for EC2, audited for repositories and system software (to improve security and stability), and production ready.

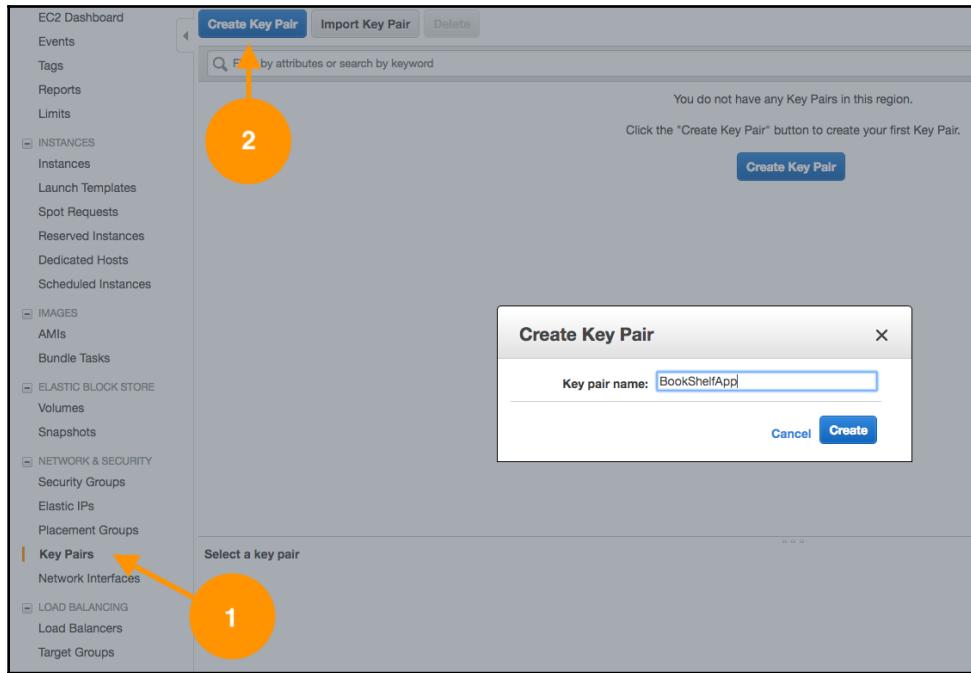
Virtualization technologies

A hardware virtual machine is the de facto virtualization method for EC2, and it is the successor of **paravirtualization (PV)**. The custom XEN hypervisor design efforts improve the performance of the virtual functions of HVM while cutting down multiple layers between the hypervisor and the operating system.

Linux instances are automatically provisioned with a public key in order to encrypt login information. They use the **Secure Shell (SSH) V2** protocol under the **public key infrastructure (PKI)**. This provides a mechanism in which a secret is broken down into two parts: the public key and the private key. The private key is available to the user, and is used in the SSH tunnel to authenticate and encrypt the communication channel for confidentiality between parties. The public key is installed in the `~/.ssh/authorized_keys` in the filesystem of the instance, as shown in the following diagram:



To prepare the environment so that we can execute run-instances against the EC2 endpoint, we will create a key pair from the EC2 console. You will be asked for a key name, as shown in the following screenshot:



Once this has been done, a one-time download will be available for the private key, locally wrapped in an X.509 certificate in a PEM format, as shown in the following screenshot. For demonstrative purposes, open this file in a text editor, and you will see that the user only holds the private key. Make sure that this file has restrictive access privileges by changing its mask with `chmod 400`:

```

BookshelfApp.pem
1 -----BEGIN RSA PRIVATE KEY-----
2 MIIEpQIBAAKCAQEAtzsHPs86yKD0zoq9/EHDTYmEHxozSwpt8lJARBhv430jj0KCjrYH4zYb5M
3 /fjE+QQN7okSyyb/54WmYVksMSi7U06PHhvDQmWEElChS1KV2jcSZMr2hCJkAywlIlhcdBMsFcP
4 ZxwcIpVny7jXDN8JJ2ULLelSrtfq8GbVzestCn02wDqngs9B1LsAzbv5nzUp6L6D6BnM08Hxmea/m
5 n/R30mckuNQXkh9KjQzaSi3fjCfRMahtoTRR94BGtzqcS3iWTQg4vquEaUTGFJwBLopNkqX/fzL
6 OV0dd00V1191Vay22YatuqoVIn/7v/Ad/CAa/Qm//nIIpy/ZZEvsMwIDAQABoIBAF9Lxv/mvZb9
7 B3WoeRkKhu37uIiIkuX7AYXHgp0X/2dQbh804d9FM031lPmgB0E0ejvE3Q4Ton+gH/Yk75ysZ7wF
8 iC6WdD4H0dM/gwn/pBWpsYF2myN09Jisgy810l12egS8pjYmgYLJ+GQQL07oiog6M80WbNMLB6LB
9 z4IKqeY3KaNek5iuml9hvyR0mmQk0qGIyRH8sr+a198s7tWs4KnM03kXY10nY4EVZIBafquVs+Eo
10 LduTbhKwN6B10sKrFyIiTKgj7d49iprEGPLt00dKAH0yinwV2Kx7YGi24y95dRXUS9nJPL/Dot
11 d/JZZDGYm5Gn0NRLozWDTbKoazkCgYEAx4xBEMzydb3TgYTGdHSAJ1oeVgt/bI09DD2rBdvt2Sp7
12 wdeK15wjju0/ifVknZuHlkxgePiyBmZrTSH6y/oA7ghhFziyG07B+MDau9HcN6Rh/pEuLW5PEFR
13 /0tSLV8e10mPyCAPxpcf+iGy809gxGvmbRmj+eQskChe3CosgdcUCgYEAzpTA6duIKX4sF97zwSAI
14 gl1w521HZLIN3VYu5GpykKJAH3i9Zift08Ugt/rIVlpqx/AaTdyVrCLTy0j/ldeA04Rce3Kcd0
15 07MhVK2I5u2YEAKDzD8pYj2fDcvbaZ312jMR8Kdm/RZZUrJXkrF0d+PChVnmRX80tUIFEyo8ZcC
16 gYEArS5dwo+Ska1ToQhk2LN2fw0j8Bqq4fGG0z5o1lgiM1loK3FzrbfJQinASrx6e1HTEI02vp
17 PxcPd+SF+iI6woNwx0SXb6ioC1bh53CCu5p1wxFl0r9/fW6QnLeVANf4ywksycVjYsc18Dnl1A
18 7sML4/4BWk7d+PinedotMsUcgYEAK3ALd1zdg2+vnKhDkrroF9ME38dYqv+BcbUu5osqlRuX1F5i
19 HuTkJKJ9JZHxJeB8UrnhA7csCFekOpD71rnNAfj1rafqjADZTZ71Wm9ScPNQbsNZ1q0+I6uVDRo
20 tqqsaz0D0tRHZt5qheEPK30swdj0XkfazKGqUXrltzc3ZKUCgYEAsI6PFYJYad0nIVDur/wefmE4
21 +PfdBoKRFxLIXfp0XpDaiBNQ/XmJbWo5BNwGwDyxWpIdLQq0SnFwIXVu+e5W15VYEftMGYiGV2Cb
22 du0SnlSF54i9ERU/Q0hudtH7vWfbijM9inD4gKCWoJCx4GbcBBfg7QXXXXXXXXXXXXXXXXXXXX=_
23 -----END RSA PRIVATE KEY-----

```

An AMI is a regionally scoped resource. To successfully launch an instance from the CLI, we need the base image ID. Use the console to search for the EC2 service, locate the **Launch Instance** blue button, and make sure that you are in the **North Virginia** region. The next screen will prompt you for the image type; use the first option, as this list is sorted by release, and make a note of the instance ID, as shown in following screenshot:

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start

- My AMIs
- AWS Marketplace
- Community AMIs
- Free tier only

Image	Name	Description	Root device type	Virtualization type	ENI Enabled	Select
	Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-14c5486b	The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.	ebs	hvm	Yes	Select
	Amazon Linux 2 LTS Candidate 2 AMI (HVM), SSD Volume Type - ami-afd15ed0	Amazon Linux 2 LTS Candidate 2 provides an updated version of the Linux Kernel (4.14) tuned for EC2, systemd support, a newer compiler (gcc 7.3), an updated C runtime (glibc 2.26), newer tooling (binutils 2.29.1), and the latest software packages through the extras mechanisms.	ebs	hvm	Yes	Select

Execute the following command, replacing the appropriate `image-id` and `key-name`:

```
aws ec2 run-instances --image-id ami-14c5486b --key-name BookshelfApp
```

Look at the following error:

```
An error occurred (InvalidAMIID.NotFound) when calling the RunInstances operation: The image id '[ami-xxxxxxxx]' does not exist
```

If you receive this error, it could be because you are in the wrong region. Remember that AMIs are tied to specific regions. Also, make sure that your CLI client is using the `us-east-1` region by issuing `aws configure`:

The output should be similar to the following (note that my client is configured with the `table` output option):

RunInstances		
OwnerId		970129648716
ReservationId		r-06699510e093e568b

Scroll down in the output and you will find information related to the state transition reason (`pending`), as shown in the following screenshot. This means that our instance is not yet able to accept inbound connections:

StateReason		
Code		pending
Message		pending

This command is the simplest possible option to launch an instance in which we did not specify any network details or the availability zone. This is called the default VPC, which is automatically created for the AWS account in every region. Our newly created instance now appears in the instances section, with the **available** state.

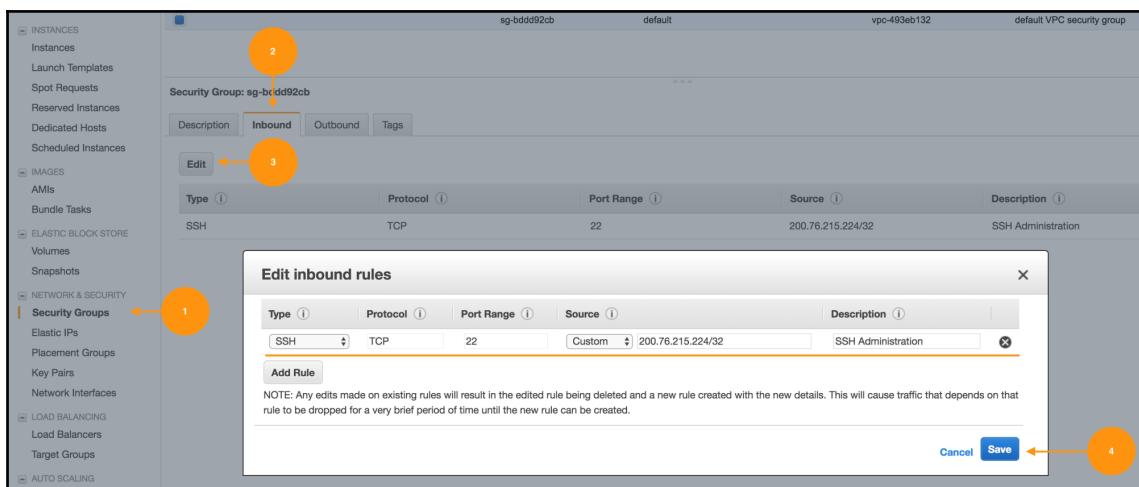
In order to remotely connect to the instance via ssh, copy the public IP or use the public DNS name. Use ssh and replace @IP with your IP, as follows:

```
ssh -i ~/.path/to/file/BookshelfApp.pem ec2-user@54.172.31.74
```

Now, inbound SSH traffic is blocked, and every instance is associated with a virtual firewall, called the **security group**. This firewall will follow the least-privilege principle, restricting all inbound traffic and allowing outbound traffic from the instance, by default. To allow inbound SSH traffic, we will apply the following configuration to our default security group.

As administrators, you should add your IP addresses as exceptions; please remember to update the value every time it is needed. You can check your current IP at <http://checkip.amazonaws.com/>.

Navigate to the EC2 console, under **NETWORK & SECURITY | Security Groups**. Find the security group with the default group name and use the checkbox to select it. Then, go to the **Inbound** tab and edit the rules, as follows:



You can also update the security group rule with the following command:

```
CLIENT_IP=$(curl -s http://checkip.amazonaws.com)"/32"
aws ec2 authorize-security-group-ingress --group-name default --protocol
tcp --port 22 --cidr $CLIENT_IP
```

Next, you will be prompted on whether to continue with the connection; type yes, and you will be logged in to the instance.

LAMP installation

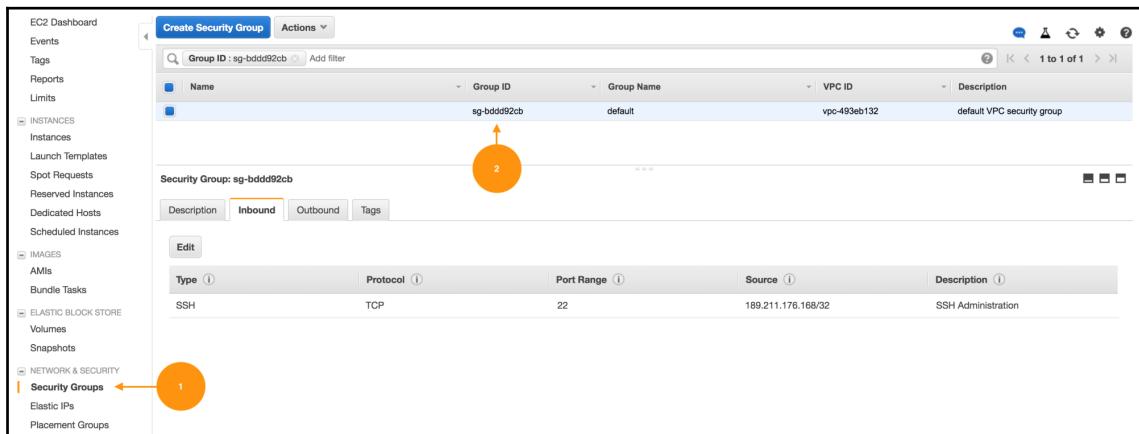
The **LAMP** stack is well known because it is a web server on Linux, Apache, MySQL, and PHP. You can use the tutorial at <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/install-LAMP.html> to learn more about every command and configuration concept in LAMP. Please take a moment to read it.

The following command will install the LAMP stack and create a PHP settings page, for reference:

```
sudo yum update -y
sudo yum install -y httpd24 php70 mysql56-server php70-mysqlnd
sudo usermod -a -G apache ec2-user
sudo chown -R ec2-user:apache /var/www
sudo chmod 2775 /var/www
find /var/www -type d -exec sudo chmod 2775 {} \;
find /var/www -type f -exec sudo chmod 0664 {} \;
echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php
```

Now that we have configured our web server, assigned proper permissions, and created a configuration web page for PHP, let's allow public traffic ingress via HTTP, with the following configuration.

You can find the `security group-id` navigating from the instances list, details, and associated security group, or directly from the security groups. Once it has been identified, copy the **Group ID** value, as shown in the following screenshot:



With the following command, we will authorize traffic ingress for our web server:

```
aws ec2 authorize-security-group-ingress --group-id sg-bddd92cb --protocol tcp --port 80 --cidr 0.0.0.0/0
```

Validate this change in the console; you will see it listed as the **HTTP** type, as shown in the following screenshot:

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	
SSH	TCP	22	189.211.176.168/32	SSH Administration

Navigate in your web browser by using the public IP address that was used before (in my case, 54.172.31.74). This should take you to the page shown in the following screenshot:

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly, but has not yet been configured.

If you are a member of the general public:
The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".

The [Amazon Linux AMI](#) is a supported and maintained Linux image provided by [Amazon Web Services](#) for use on [Amazon Elastic Compute Cloud \(Amazon EC2\)](#). It is designed to provide a stable, secure, and high performance execution environment for applications running on [Amazon EC2](#). It also includes packages that enable easy integration with [AWS](#), including launch configuration tools and many popular AWS libraries and tools. [Amazon Web Services](#) provides ongoing security and maintenance updates to all instances running the [Amazon Linux AMI](#). The Amazon Linux AMI is provided at no additional charge to [Amazon EC2](#) users.

If you are the website administrator:
You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

You are free to use the images below on Apache and Amazon Linux AMI powered HTTP servers. Thanks for using Apache and the Amazon Linux AMI!

Powered by ELECTRICITY

Verify the PHP Apache configuration, under `54.172.31.74/phpinfo.php`, as shown in the following screenshot

PHP Version 7.0.30	
System	Linux ip-172-31-209-4.14.33-51.37.amzn1.x86_64 #1 SMP Thu May 3 20:07:43 UTC 2018 x86_64
Build Date	May 10 2018 17:39:56
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php-7.0.conf:/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php-7.0.d
Additional .ini files parsed	/etc/php-7.0.d/20-bz2.ini, /etc/php-7.0.d/20-calendar.ini, /etc/php-7.0.d/20-ctype.ini, /etc/php-7.0.d/20-curl.ini, /etc/php-7.0.d/20-dom.ini, /etc/php-7.0.d/20-exif.ini, /etc/php-7.0.d/20-finfo.ini, /etc/php-7.0.d/20-ftp.ini, /etc/php-7.0.d/20-gettext.ini, /etc/php-7.0.d/20-iconv.ini, /etc/php-7.0.d/20-json.ini, /etc/php-7.0.d/20-mysqlind.ini, /etc/php-7.0.d/20-pdo.ini, /etc/php-7.0.d/20-phar.ini, /etc/php-7.0.d/20-posix.ini, /etc/php-7.0.d/20-shmop.ini, /etc/php-7.0.d/20-simplexml.ini, /etc/php-7.0.d/20-sockets.ini, /etc/php-7.0.d/20-sqlite3.ini, /etc/php-7.0.d/20-sysvmsg.ini, /etc/php-7.0.d/20-sysvsem.ini, /etc/php-7.0.d/20-sysvshm.ini, /etc/php-7.0.d/20-tokenizer.ini, /etc/php-7.0.d/20-xml.ini, /etc/php-7.0.d/20-xmlwriter.ini, /etc/php-7.0.d/20-xsl.ini, /etc/php-7.0.d/30-mysqli.ini, /etc/php-7.0.d/30-pdo_mysql.ini, /etc/php-7.0.d/30-pdo_sqlite.ini, /etc/php-7.0.d/30-wddx.ini, /etc/php-7.0.d/30-xmlreader.ini, /etc/php-7.0.d/php.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, compress.bzip2, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, bzip2.*, convert.iconv.*
This program makes use of the Zend Scripting Language Engine: Zend Engine v3.0.0, Copyright (c) 1998-2017 Zend Technologies	
zend engine	

The web page in the preceding screenshot confirms that we have successfully installed a web server using the LAMP stack on EC2.

Scaling the web server

Once our instance has been provisioned and the web server is running, we will add a greater capacity to permit more concurrent users to interact with the web service. We will do this by going through the following steps:

1. Obtain the `instance-id` with the following expression:

```
export CURRENT_INSTANCE=$(aws ec2 describe-instances --query
'Reservations[*].Instances[*].InstanceId' --filters 'Name=instance-
state-name,Values=running' --output text)
```

2. We must stop the instance to change the `instance-type` attribute to `m4.large`, as follows:

```
aws ec2 stop-instances --instance-id $CURRENT_INSTANCE --output
json
```

3. Once stopped, modify the attribute via the CLI, as follows:

```
aws ec2 modify-instance-attribute --instance-id $CURRENT_INSTANCE --
instance-type m4.large
```

4. Restart the instance, as follows:

```
aws ec2 start-instances --instance-id $CURRENT_INSTANCE --output
json
```

Well done! Now, your web server is running on hardware with a larger capacity.

Resiliency

To build resilience, let's shut down and start up the web server to make sure that MySQL and Apache automatically start with the operating system. We first define the `CURRENT_INSTANCE` variable to make our commands easier, as follows:

```
CURRENT_INSTANCE=$(aws ec2 describe-instances --query
'Reservations[*].Instances[*].InstanceId' --filter
'Name=tag:Name,Values=WebServer' --output text)

aws ec2 reboot-instances --instance-ids $CURRENT_INSTANCE
```

Again, check `phpinfo.php`, to verify that it is still working. Now, issue a `stop instance` command, wait a few seconds, and start the instance again, as follows:

```
aws ec2 stop-instances --instance-id $CURRENT_INSTANCE  
aws ec2 start-instances --instance-id $CURRENT_INSTANCE
```

If you try to reach the informational web page, `54.172.31.74/phpinfo.php`, it will probably no longer be there. Why did this happen? Every time you shut down an instance, it will run on another, random hardware machine with different network attributes, such as IPs and DNSes.

EC2 has a resource called Elastic IP; this resource is associated with the account level, not the subnet level. It is not an ephemeral resource, and it can be used to achieve greater levels of flexibility. Let's create one with the following command:

```
aws ec2 allocate-address --domain vpc
```

The result will return the allocation ID; this value can be used to associate the public IP with the instance, as shown in the following screenshot:

AllocateAddress		
AllocationId	Domain	PublicIp
eipalloc-d300d8db	vpc	52.44.105.242

Now, associate this static resource with the instance, as follows:

```
aws ec2 associate-address --allocation-id eipalloc-d300d8db --instance-id  
$CURRENT_INSTANCE
```

Excellent! Navigate to the web page verify that our web server is associated with this static resource (in my case, `52.44.105.242`). This IP will remain attached to the instance, even when the instance is stopped. The only case in which the Elastic IP will be detached automatically is when the instance gets terminated.

Quota: You can only have five Elastic IPs associated with your account across all regions. This is a soft limit.



EC2 persistence model

Connect to the EC2 instance again using the Elastic IP via SSH and create a file with the following command:

```
tar -cvf /var.tar.gz /var/
```

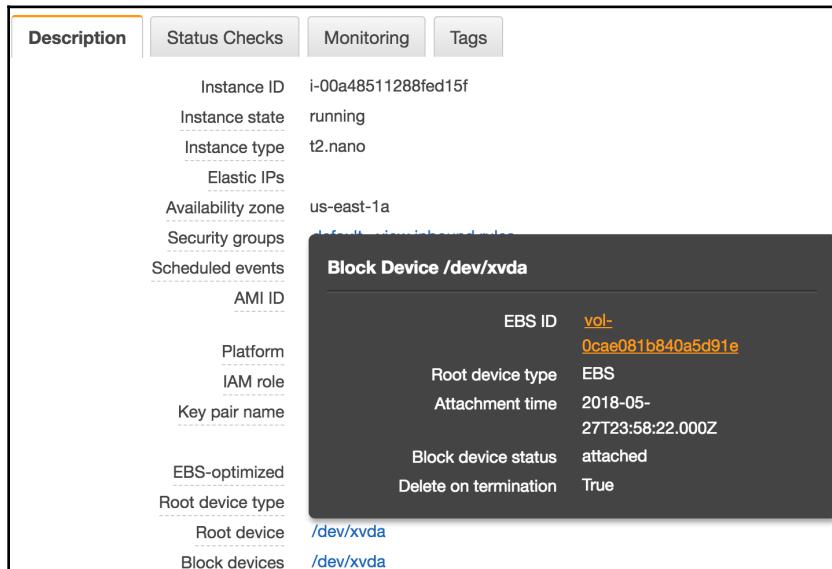
You have created a compressed file, with all of the contents of `/var`. We are interested in the virtual disk mapping; for this, issue the `lsblk` command, as shown in the following screenshot:

```
[ec2-user@ip-172-31-29-175 ~]$ lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda   202:0    0 8G  0 disk
`-xvda1 202:1    0 8G  0 part /
[ec2-user@ip-172-31-29-175 ~]$ curl http://169.254.169.254/latest/meta-data/block-device-mapping/ami/
/dev/xvda[ec2-user@ip-172-31-29-175 ~]$
```

In this output, we have the root filesystem, `/`, mounted on the device called `xvda1`. This naming convention is a common one for Xen hypervisors using HVM virtualization. Every EC2 instance has private access to a DNS metadata server within the VPC at the `169.254.169.254` canonical address. This metadata server can be used to read information about the instance itself, along with the surrounding infrastructure in which it is running. This is valuable when you are writing bootstrapping scripts, applying application configurations, and even performing service authentication techniques. With a simple `curl` command, we can access the `block-device-mapping` information from this image.

EC2 instances can be categorized as persistent when they are backed by an EBS volume, and as ephemeral when they use the instance store media (root volume). The block devices list shows that all of the `/` filesystem is mounted in the root device volume. This direct storage capacity is borrowed from a **Direct Attached Storage (DAS)** disk, upon which the image is stored and runs. Let's perform a simple test: stop the instance, detach the root volume, attach it again, and then start the instance; this will force a reset on every block available to the root volume, and our data will be lost (`/var.tar.gz`).

Retrieve the volume-id associated with the root device, as shown in the following screenshot:



We will now detach the root volume for this instance, as follows:

```
aws ec2 stop-instances --instance-id $CURRENT_INSTANCE  
aws ec2 detach-volume --volume-id vol-0cae081b840a5d91e
```

Now, attach the root volume again, as follows:

```
aws ec2 attach-volume --volume-id vol-0cae081b840a5d91e --instance-id  
aws ec2 start-instances --instance-id $CURRENT_INSTANCE --output json
```

Our file no longer exists. What we can learn from this test is that ephemeral storage cannot be used as a durable mechanism because of its volatile nature. The EC2 architecture decouples the computing and storage layers by providing block storage as a service via volumes. These volumes have a life cycle independent of the instance.

Instance store backed AMIs have the advantage of using this temporary area, without cost and with high performance—up to 315,000 IOPS. Common use cases include virtual memory paging, buffers, and local caches.

Disaster recovery

In order to have persistence, we must have an EBS backed instance, with one (or many) EBS volume attached to it. Use the desired format and mount the filesystems. These EBS volumes are available as **Network Attached Storage (NAS)**; their life cycles are independent of the instance.

To demonstrate the use of this service, let's provision an EBS volume and then attach it to the instance. EBS volumes have AZ scope, and they are automatically replicated inside of the data center to achieve high durability out of the box. Let's go through the following steps:

1. To properly associate the volume, let's query the AZ in which this instance is currently running, as follows:

```
aws ec2 describe-instances --instance-ids $CURRENT_INSTANCE --output json --query 'Reservations[0].Instances[0].Placement'
```

The result is as follows:

```
{  
    "Tenancy": "default",  
    "GroupName": "",  
    "AvailabilityZone": "us-east-1a"  
}
```

2. Create the volume by using the AvailabilityZone information, as follows:

```
aws ec2 create-volume \  
--size 80 \  
--availability-zone $(aws ec2 describe-instances --instance-ids  
$CURRENT_INSTANCE --query  
'Reservations[0].Instances[0].Placement.AvailabilityZone' --filter  
'Name>tag:Name,Values=WebServer' --output text) \  
--volume-type gp2
```

3. Now, describe the volumes that are available in order to find the status information, as follows:

```
aws ec2 describe-volumes
```

The preceding command generates the following output:

DescribeVolumes	
Volumes	
AvailabilityZone	us-east-1a
CreateTime	2018-05-27T23:58:22.757Z
Encrypted	False
Iops	100
Size	8
SnapshotId	snap-086b6d892c1edb9b2
State	in-use
VolumeId	vol-0cae081b840a5d91e
VolumeType	gp2
Attachments	
AttachTime	2018-05-28T00:15:17.000Z
DeleteOnTermination	False
Device	/dev/xvda
InstanceId	i-00a48511283fed15f
State	attached
VolumeId	vol-0cae081b840a5d91e
Volumes	
AvailabilityZone	us-east-1a
CreateTime	2018-05-28T00:50:35.674Z
Encrypted	False
Iops	240
Size	80
SnapshotId	
State	available
VolumeId	vol-080c266f654bca621
VolumeType	gp2

4. The volume state for the first volume is `in-use` and `attached`. The second volume is `available` for use, so we need to attach it to our instance with the following command:

```
aws ec2 attach-volume --volume-id $(aws ec2 describe-volumes --query 'Volumes[0].VolumeId' --output text) --instance-id $CURRENT_INSTANCE --device /dev/xvda
```

5. Log in to the instance via SSH and list the available devices with `lsblk`, as follows:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
xvda	202:0	0	8G	0	disk	
`-xvda1	202:1	0	8G	0	part	/
xvdb	202:16	0	80G	0	disk	

The xvdb is the new device that is listed, but it lacks a mount point in the filesystem. Create an ext4 filesystem on this device, on the /dev/xvdb partition with the following command:

```
sudo mkfs -t ext4 /dev/xvdb
```

6. The filesystem creation process will look as follows:

```
[ec2-user@ip-172-31-18-40 ~]$ sudo mkfs -t ext4 /dev/xvdb
mke2fs 1.42.12 (29-Aug-2014)
Creating filesystem with 20971520 4k blocks and 5242880 inodes
Filesystem UUID: f2787091-a85e-4fb7-b1b1-400940117d96
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
     4096000, 7962624, 11239424, 20480000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

7. Once you are finished, create a work directory for the volume, as follows:

```
sudo mkdir /data
```

Then, mount the volume as follows:

```
sudo mount /dev/xvdb /data
```

8. Finally, list the block devices as follows:

```
[ec2-user@ip-172-31-18-40 ~]$ sudo mkdir /data
[ec2-user@ip-172-31-18-40 ~]$ sudo mount /dev/xvdb /data
[ec2-user@ip-172-31-18-40 ~]$ lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
xvda      202:0    0   8G  0 disk
`-xvda1   202:1    0   8G  0 part /
xvdb      202:16   0  80G  0 disk /data
```

9. We will set the configuration to automount the volume every time the operating system boots. To achieve this, edit `/etc/fstab` with the editor of your preference (I am using `vi` as the editor) to resemble the following:

```
vi /etc/fstab
```

Make sure that your configuration file has the highlighted line shown in the following screenshot:

```
#  
LABEL=/      /          ext4    defaults,noatime 1  1  
tmpfs       /dev/shm   tmpfs   defaults        0  0  
devpts      /dev/pts   devpts  gid=5,mode=620 0  0  
sysfs       /sys       sysfs   defaults        0  0  
proc         /proc      proc    defaults        0  0  
/dev/xvdb   /data     ext4    defaults,noatime 1  2
```

10. Mount all of the devices with the `mount -a`, and show all disks, filesystems, and mounts using `df -h`, as follows:

```
[ec2-user@ip-172-31-18-40 ~]$ sudo mount -a  
[ec2-user@ip-172-31-18-40 ~]$ df -h  
Filesystem      Size  Used Avail Use% Mounted on  
devtmpfs        232M   60K  232M   1% /dev  
tmpfs           242M     0  242M   0% /dev/shm  
/dev/xvda1      7.8G  1.1G  6.7G  14% /  
/dev/xvdb       79G   56M  75G   1% /data
```

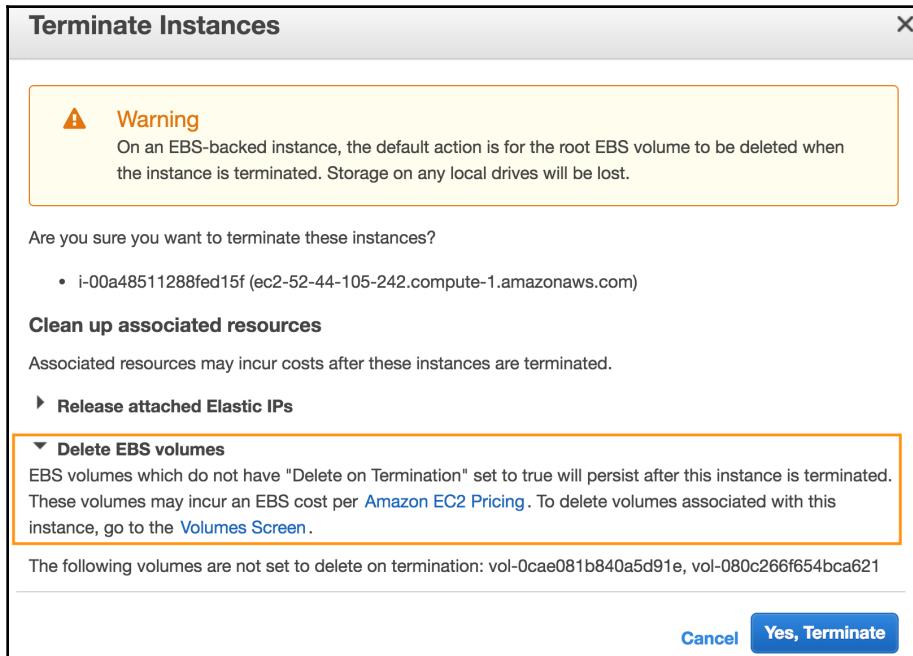
This exercise serves as a demonstration of how to add persistent storage. Think about the result of a volume detach and attach operation. That's right: all of the `/etc/fstab` configurations will be lost because all of the OS runs on the root volume, and not the additional EBS volume (`xvdb`). To solve this, we must create a disk image that will copy the root volume, plus all of the EBS attached volumes. This information will be read from the EC2 metadata service, as would be illustrated via a simple `curl` test.

Let's look at some of the differences between instance storage and EBS persistence, as shown in following table:

	Instance Storage	EBS
Persistence model	Volatile	Persistent Block Service
Storage Architecture	DAS	NAS
Performance	High (up to 315,000 IOPS)	Medium (20,000 IOPS)
Cost	Free	Storage, Use, Capacity
Mounting	Automatic	Manual
Lifecycle	Instance dependent	Independent of Instance
Use Case	Buffers, swap, logs	Databases, file servers

Cascading deletion

- Terminate your instance via the console and observe the default behavior, as shown in following screenshot:



This is a dependency that must be managed manually in order to avoid unnecessary costs. If you wish to preserve existing data for later use, a snapshot must be created, which will automatically be stored on S3. Next, delete the disk as follows:

```
aws ec2 create-snapshot --volume-id vol-080c266f654bca621 --  
description "Data volume first snapshot"
```

```
gabanox-2:chapter02 gabanox$ aws ec2 create-snapshot --volume-id vol-080c266f654bca621 --description "Data volume first snapshot"  
-----  
| CreateSnapshot |  
+-----+-----+  
| Description | Data volume first snapshot |  
| Encrypted | False |  
| OwnerId | 970129648716 |  
| Progress | |  
| SnapshotId | snap-069045fa5b22b7409 |  
| StartTime | 2018-05-28T02:11:48.000Z |  
| State | pending |  
| VolumeId | vol-080c266f654bca621 |  
| VolumeSize | 80 |  
+-----+-----+
```

2. Verify the status of the snapshot, as follows:

```
aws ec2 describe-snapshots --owner-ids self
```

3. Now, delete the root and EBS volume (remember to replace the following code with your own `volume-id`):

```
aws ec2 delete-volume --volume-id vol-080c266f654bca621  
aws ec2 delete-volume --volume-id vol-0cae081b840a5d91e
```

If successful, this command will not return any output.

Bootstrapping

The next exercise will be to spin off a new instance using the user data feature. This way, we will have a means to execute a series of instructions on the operating system automatically, as a post-creation process.

We will use a text file as a reference for the bootstrapping scripts. This set of instructions will be executed internally by the `cloud-init` open source package.

This command will also reference the default security group (you can find the ID by navigating within the console and selecting the current instance properties). To set this up, let's go through the following steps:

1. We will provision our new instance by using the following user data input file. You can find this file in the GitHub repository under chapter02/bootstrap.txt:

```
aws ec2 run-instances --image-id ami-14c5486b --key-name BookshelfApp --instance-type t2.medium --security-group-ids sg-bddd92cb --user-data file://bootstrap.txt
```

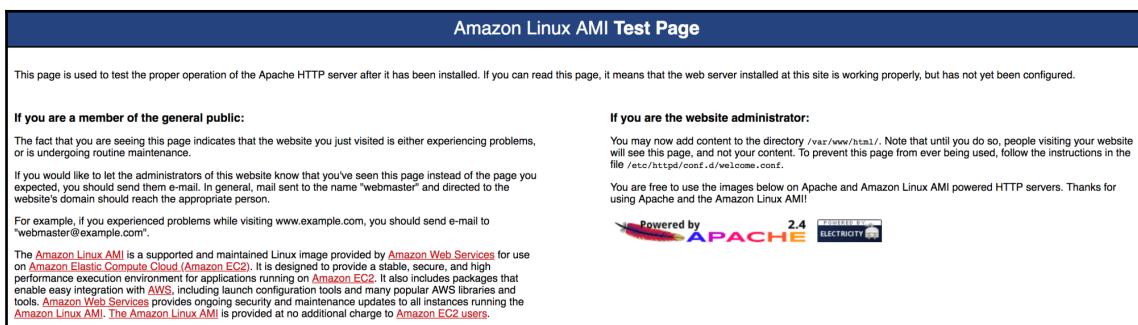
2. Refresh the current instance variable, as follows:

```
export CURRENT_INSTANCE=$(aws ec2 describe-instances --query 'Reservations[*].Instances[*].InstanceId' --filters 'Name=instance-state-name,Values=running' --output text)
```

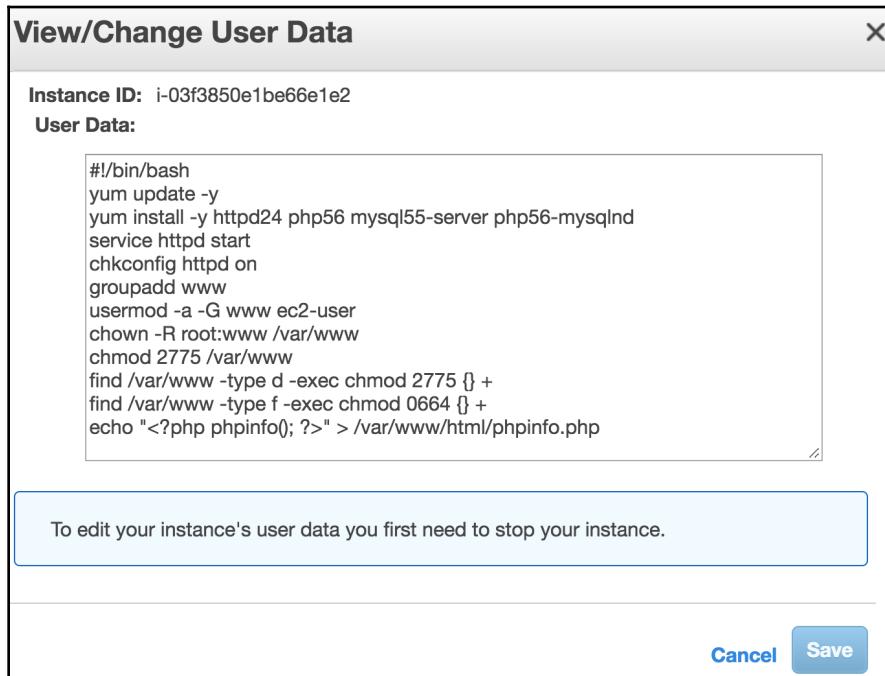
3. Now, associate the Elastic IP as follows:

```
aws ec2 associate-address --allocation-id eipalloc-d300d8db --instance-id $CURRENT_INSTANCE
```

4. Navigate to your Elastic IP address by using your web browser (in my case, it's 52.44.105.242); we can now validate that our web server was created from scratch, as shown in the following screenshot:



5. Inside of the EC2 console, select the running instance, then **Actions | Instance Settings | View/Change User Data**, as shown in the following screenshot:



6. Perform a stress test with Apache Benchmark using the following code; if you don't have it installed on your *nix system by default, then follow the installation guide at <http://httpd.apache.org/>:

```
ab -n 1000 -c 10 -k -H "Accept-Encoding: gzip, deflate"
http://52.44.105.242/
```

Store the results on a spreadsheet.



Scaling the compute layer

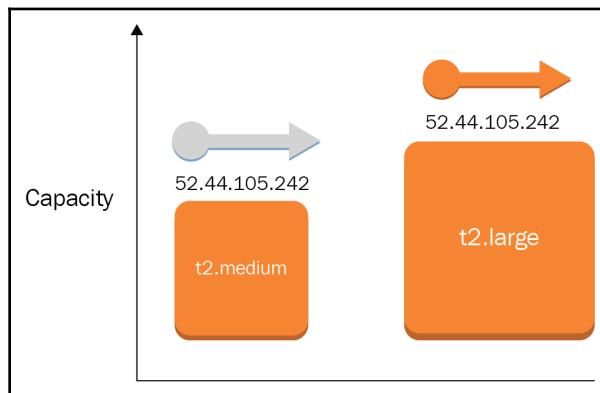
1. Let's create a second instance with a larger capacity (`t2.large`), as follows:

```
aws ec2 run-instances --image-id ami-14c5486b --key-name  
BookshelfApp --instance-type t2.large --security-group-ids sg-  
bddd92cb --user-data file://bootstrap.txt
```

Store the results on a spreadsheet.



2. To switch traffic over to our larger instance, we will implement the **floating IP cloud pattern** (http://en.clouddesignpattern.org/index.php/CDP:Floating_IP_Pattern) by reassociating the virtual IP from the active to the passive instance, as shown in the following diagram:



3. Issue the following command by using your new instance ID and the previous IP's ID allocation:

```
aws ec2 associate-address --instance-id i-096a8c337e10e9edf --  
allocation-id eipalloc-d300d8db
```

4. Perform the load test again and write down the results, in order to establish a quantitative benchmark, as follows:

```
ab -n 1000 -c 10 -k -H "Accept-Encoding: gzip, deflate"  
http://52.44.105.242/
```

5. My results are depicted in the following table:

Apache Web Server 2.4	t2.medium	t2.large
Time taken for tests	20.290 seconds	12.944 seconds
Requests per second	49.28 [#/sec] (mean)	77.25 [#/sec] (mean)
Time per request	20.290 [ms] (mean)	12.944 [ms] (mean)

Proactive scalability

Proactive scalability is used to remove human intervention. It is possible to create a latency metric and measure the requests per second in a time interval, or perhaps the CPU use percentage, in order to create an alarm and automatically execute a script that moves the virtual IP to a more efficient and robust node. Using this same rationale, we can make a fully-autonomous system that adds redundant copies of our web server, and a load balancer that homogeneously distributes requests, increasing the overall system capacity that adapts to different traffic patterns. This is the core responsibility of Auto Scaling, a service that will be discussed in depth in the following chapters.

Scaling a database server

Hosting a relational database server based on MySQL using the AWS RDS service choosing a t2.micro instance (1 GB RAM and 1vCPU), the engine automatically configures the maximum number of concurrent connections, based on the available memory, by using the formula for MySQL 5.7.21, as follows:

```
max_connections = (Available RAM - Global Buffers) / Thread Buffers
```

The preceding command will result in the following configuration for a db.t2.micro instance:

RDS > Instances

Instances (1)

DB instance Instances (1) Instance actions Restore from S3 Launch DB Instance

mysql-instance MySQL backing-up 1 Connections db.t2.micro

```
mysql> show global variables like '%max_connections%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 66   |
+-----+-----+
1 row in set (0.06 sec)

mysql> |
```

Nevertheless, if we scale the database instance vertically, to a db.m4.large with 8 GB RAM and two vCPUs, we will increase the concurrent connections available for this node, as following screenshot:

RDS > Parameter groups > default.mysql5.7

default.mysql5.7

Parameters Edit parameters

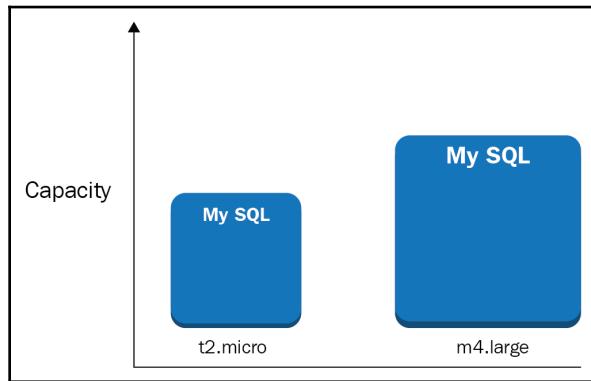
connections

Name	Values	Allowed values	Modifiable	Source	Apply type	Data type	Description
max_connections	{DBInstanceStateMemory/12582880}	1-100000	true	system	dynamic	integer	The number of simultaneous client connections allowed.

```
mysql> show global variables like '%max_connections%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 648   |
+-----+-----+
1 row in set (0.06 sec)

mysql> $
```

This change can be applied to a maintenance window as part of DBA operations, as shown in the following diagram:



The ability to scale the reading capacity is also available for databases on RDS using what are called **read replicas**. You can provision up to five read replicas for every database engine, and up to 15 for Aurora clusters.

As a challenge, try to create a read replica by choosing your current instance, going to the **Actions** menu, and then selecting **Create Read Replica**. Use a database client with the `read` endpoint and perform some `select` queries.

Summary

In this chapter, we learned to integrate cloud services, deploy new applications, and interconnect and extend existing infrastructure to the cloud. We also used application services as message queues, publisher subscriber, API Gateway, and Lambda as a bridge as a adapter.

Further reading

- Automating Elasticity: <https://docs.aws.amazon.com/aws-technical-content/latest/cost-optimization-automating-elasticity/introduction.html>

4

Hybrid Cloud Architectures

In this chapter, you will learn about networking on AWS with the **Virtual Private Cloud (VPC)**, and how every object interacts to design a network architecture for every use case. The VPC service is flexible enough to let network administrators and developers create the underlying infrastructure to support all kinds of applications and workloads. The networking infrastructure has to be secure, scalable, and highly flexible, to provide the foundation structure for any use case.

We will cover the following topics in this chapter:

- Effective migration to the cloud
- Extending your data center
- Storage gateway use cases
- The Database Migration Service

Effective migration to the cloud

Several factors can be taken into account when creating a migration strategy. If you have multiple applications with different levels of complexity, start with the easiest ones (the applications that have portable architectures, with a smaller business impact) and move forward to more complex scenarios (for example, proprietary, licensed, or highly coupled monolithic applications).

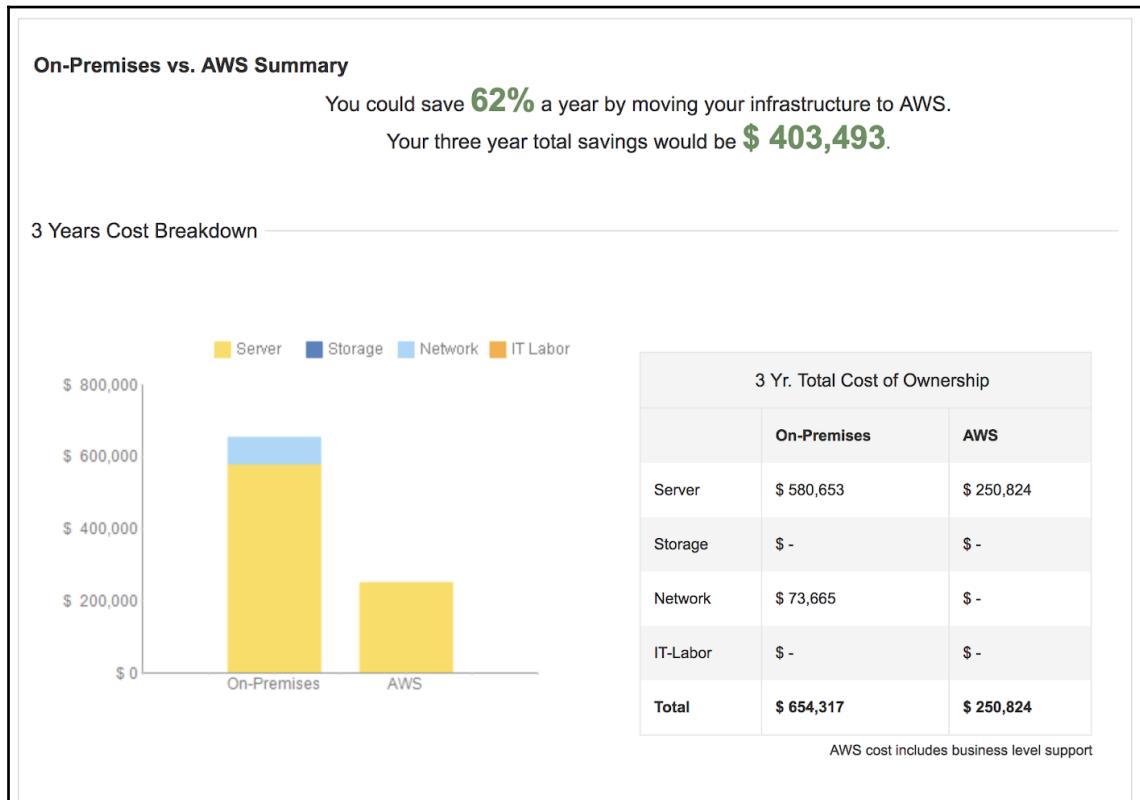
AWS provides a reference to effectively migrate to the cloud via the following steps (ordered from simple to complex):

- **Rehost:** This can be done with portable software stacks, like Java virtual machines, Docker containers or VMware, Microsoft Hyper-V, and Citrix Xen virtualization technologies, that go back and forth to EC2, with the VM Import/Export service.

- **Replatform:** The amount of effort to configure and tune applications is higher than rehosting, but only takes a moment to switch from highly coupled stacks to AWS managed services or platform as a service. An example of this is switching from an Oracle WebLogic Java application to an Apache Tomcat standalone Spring Boot application running in Elastic Beanstalk, or from a LAMP on-premise server to a three-layer web app running with ELB, EC2, and RDS.
- **Refactor/Rearchitect:** This strategy is aimed at improving the scale, business continuity, or performance of the current architecture; design strategies and patterns, like multi-tier, serverless, or managed service, can apply. Stateless applications with ancillary services, like RDS, S3, or DynamoDB, can improve the overall performance of your applications.
- **Repurchase:** If your licensed software, or **Software as a Service (SaaS)**, does not scale or fit to your business needs, you need either need to customize or spend less on that specific IT function. You can use an open source software or a new service provider, or you can build your own custom solution.
- **Retire:** The application's retirement can be done when a new solution or product covers the previous functionality. Hard dependencies and non-portable software can make it unlikely to migrate successfully (for example, a mainframe application). Ending the life of these applications is the best strategy, focusing on new projects that can align with contemporary business objectives.
- **Retain:** This keeps applications that represent critical business components, or the core of many systems. Working at a large bank, I saw the problems that complex systems encounter upon migration; business cannot be paused until the migration of millions of lines of code, written in COBOL, has finished. In parallel SOA, efforts were put in place to provide a new service API and gradually swift to the newer implementations. A useful design pattern for this is the **StranglerApplication** (<https://www.martinfowler.com/bliki/StranglerApplication.html>).

You can build a business case, which is a prioritized backlog used by imperatives like cost, risk, **Return on Investment (ROI)** and dependencies. The cost opportunity can be calculated using the TCO calculator (<https://awstcoccalculator.com/>) with standard models.

The following diagram shows three years of savings, by migrating 100 VMware instances with 16 GB RAM to AWS:



Some projects need objective validation from a technical perspective. To make experimentation easier, it is common to build a **proof of concept (POC)** or a which can validate our assumptions. Another tool to calculate the average expenses of a planned solution is the **Simple Monthly Calculator** (<http://aws.amazon.com/calculator>). This tool can also compare different EC2 pricing models, on demand, versus reserved instances.

As a best practice, the migration process should conform to the following phases:

1. Migration preparation and business planning
2. Portfolio discovery and planning
3. Application design
4. Migration and validation
5. Operation

Extending your data center

Many users and companies, across industries, see AWS as a cloud strategy that's used to gain strong competencies, data center expansion, or disaster recovery options. It gives companies the best of both IT worlds, and will improve performance. You should be prepared for traffic spikes, and should increase your company agility by extending or moving workloads to AWS.

There are two common models for deployment: all in the cloud, or hybrid deployment. AWS provides all of the necessary components to enable customers to improve the efficiency of their operations, while keeping local infrastructure and internal systems that can burst to the cloud.

All in the cloud

Efforts are put in place so that developers have all their IT operation in the AWS cloud. It is common to see new companies and startups adopt this model because it leverages all the good parts of cloud computing like enterprise agility, less technical debt, effort, and cost. Companies like Netflix do not have any on-premises infrastructure, and run 100% in the cloud.

VPC

The **virtual private cloud (VPC)** service provides a networking environment for your applications and data. This service gives the customer the ability to isolate workloads and be in full control of the network topology and capabilities.

The VPC can be created with the following objects:

- CIDR block
- A route table
- Subnets
- A network gateway

Every EC2 instance must run in a VPC, and every AWS account has a default VPC in every region, pre-created and ready to be used, with several default configurations. VPCs are regional resources that will provide management capabilities by isolating and partitioning the VPC IP space in a secure way. In the following sections, we will analyze the default VPC components in order to understand their behavior.

Tenancy

The VPC can be designed for shared **tenancy**, which is the default mode and dedicated mode in which your computing resources (like dedicated instances) can run in dedicated hardware, isolated from other AWS customers. This enforces compliance at the networking level.

Sizing

VPCs can be designed to fit a CIDR space (from /16 to /28, in slash notation), indicating the size in bits, the former being the biggest size available for networking by VPC. There are a total of 32 bits in IPv4 address space, calculated by the power of two. A /16 network has 65,536 IPs; a /17 network has 32,738 IPs; and so on, until we reach the smallest network available on VPC: /28, with 16 IPs. As a rule of thumb, always use the biggest address space available for private VPCs (/16), because that way, you will be able to allocate more computing resources and gain flexibility. Not doing so will constrain the usable IP space, exhausting your resources. The VPC cannot be resized, so you will have to migrate your instances to a bigger VPC:

The screenshot shows a web browser window displaying the '3. Private Address Space' section of RFC 1918. The text describes three reserved blocks for private internets: the 24-bit block (10.0.0.0/8), the 20-bit block (172.16.0.0/12), and the 16-bit block (192.168.0.0/16). Below this, it explains the historical context of these blocks in pre-CIDR notation.

3. Private Address Space

The Internet Assigned Numbers Authority (IANA) has reserved the following three blocks of the IP address space for private internets:

10.0.0.0	-	10.255.255.255	(10/8 prefix)
172.16.0.0	-	172.31.255.255	(172.16/12 prefix)
192.168.0.0	-	192.168.255.255	(192.168/16 prefix)

We will refer to the first block as "24-bit block", the second as "20-bit block", and to the third as "16-bit" block. Note that (in pre-CIDR notation) the first block is nothing but a single class A network number, while the second block is a set of 16 contiguous class B network numbers, and third block is a set of 256 contiguous class C network numbers.

RFC 1918 Address allocation for private internet

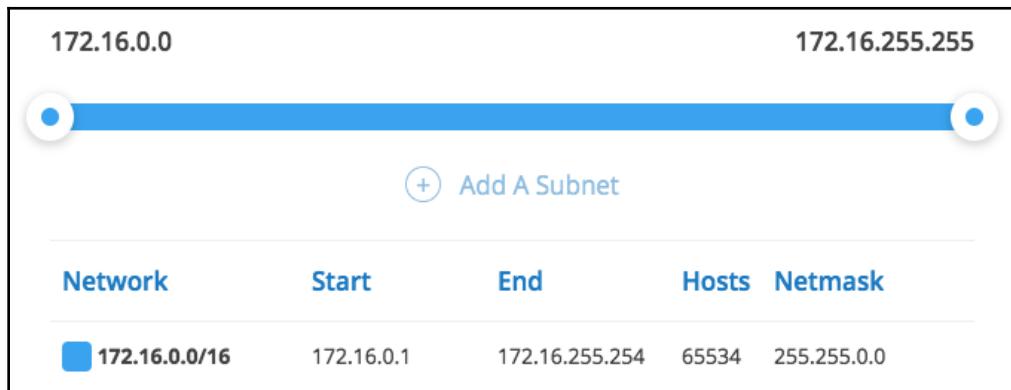
It is important to know that five IPs are reserved, and cannot be assigned for EC2 instances. The first IP is used for the network, the second one is used by the VPC DNS server, the third is reserved for future use, and the last is reserved as well, to avoid broadcast traffic (only unicast traffic is allowed in the VPC).

Assuming a /16 VPC for administrative and security purposes, we will opt to partition the address space by subnetting. It is a good design practice to segregate public traffic from private traffic; so, it is a good idea to have one public subnet and one private subnet for every AZ, to have high availability at the network level.

The default VPC

This network uses 20 bits for the host allocation, and a 12-bit network mask ($32-20=12$). The default VPC can be identified because it uses the CIDR address space of 172.31.0.0/16; this kind of network allows for 16 contiguous class B networks, and the following combinations are available:

One big subnet with the full space, with 65,534 hosts, as shown in the following screenshot:

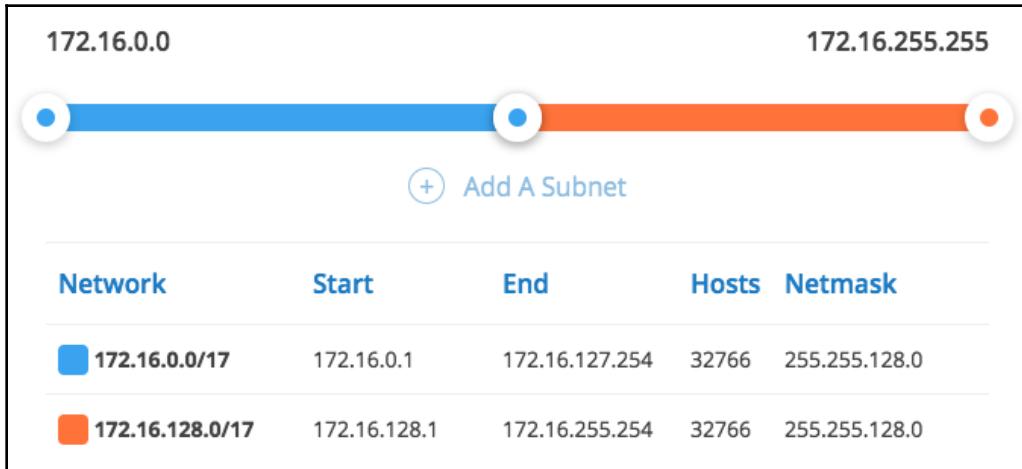


In the VPC, this subnet is usable with 65,531 IPs, as shown in the following screenshot:

Name	Subnet ID	State	VPC	IPv4 CIDR	Available IPv4
one-big-subnet	subnet-0a77981fa8adbc617	available	vpc-067d929cb7ca7e938 ...	172.16.0.0/16	65531

This is a poorly designed VPC, because the subnet will use only one AZ, and won't be resilient.

Two subnets, each in one availability zone, as shown in the following screenshot:

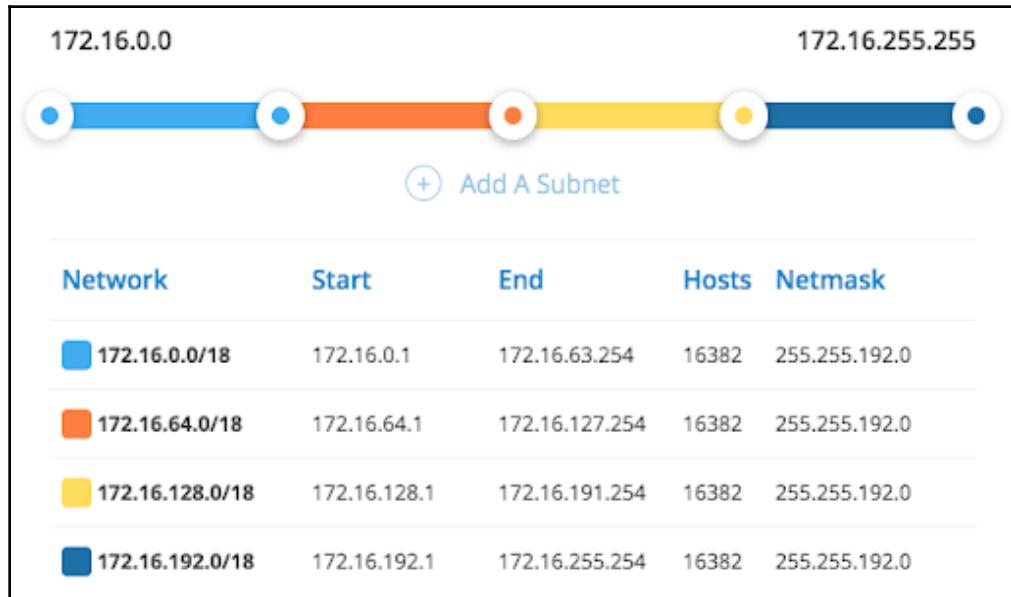


Now, we have two subnets that are distributed across two AZs, which will be resilient if both subnetworks share the same functionality (for example, public servers):

The screenshot shows the AWS Lambda console. At the top, there is a search bar with the text 'vpc-067d929cb7ca7e938' and a 'Add filter' button. Below the search bar is a table with the following data:

Name	State	IPv4 CIDR	Available IPv4	Availability Zone
us-east-1b-subnet	available	172.16.128.0/17	32763	us-east-1b
us-east-1a-subnet	available	172.16.0.0/17	32763	us-east-1a

Let's redesign the network space. This time, we will use one subnet for public traffic, and one for private traffic. For resiliency, we will use two availability zones, with a total of four subnets:



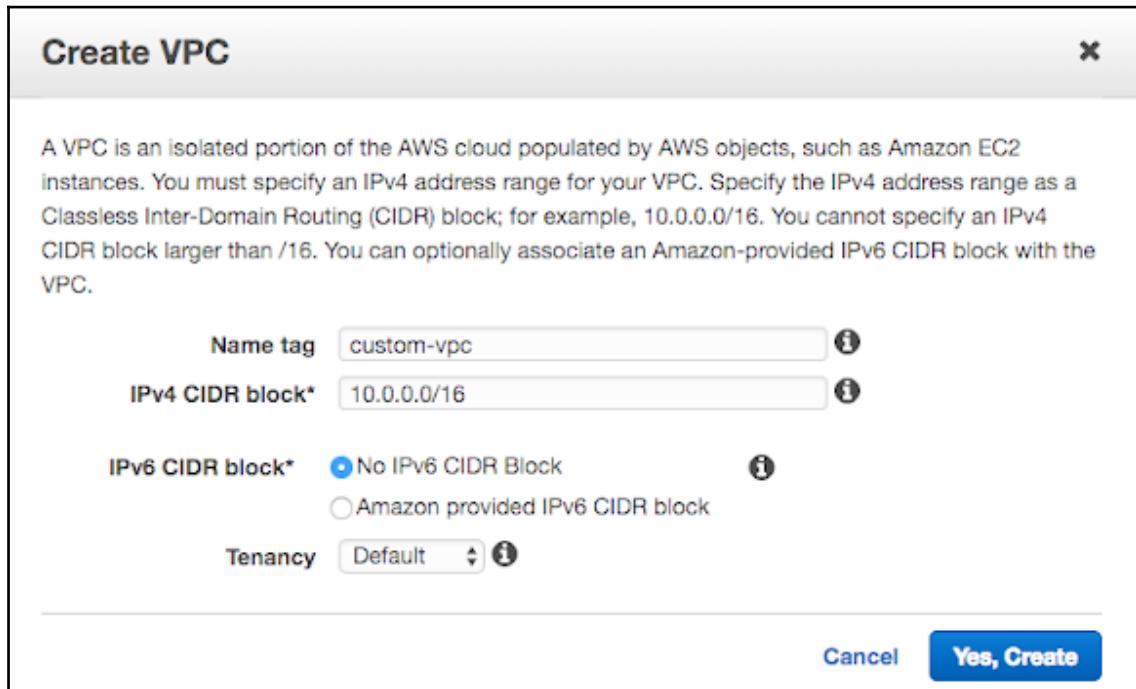
Arbitrarily, we can assign each subnet to be in a different AZ, and we can use public and private routing, as shown in the following screenshot:

	Name	State	IPv4 CIDR	Available IPv4	Availability Zone
	us-east-1a-public-subnet	available	172.16.0.0/18	16379	us-east-1a
	us-east-1a-private-subnet	available	172.16.64.0/18	16379	us-east-1a
	us-east-1b-public-subnet	available	172.16.128.0/18	16379	us-east-1b
	us-east-1b-private-subnet	available	172.16.192.0/18	16379	us-east-1b

This kind of network architecture doesn't leave room for growth, because the hosts have been exhausted by segregating public traffic from private traffic. VPCs can be used as an administrative tool so that you can segregate spaces for different kinds of applications, environments, or clients. Multi-tier architectures will benefit from networking layer topologies in which each functional tier is isolated.

Rearchitect the network with 24-bit block subnets. The network will host three functional tiers, using the same public/private pattern, and leave room for additional subnets. This is more of an industry standard CIDR range that's widely used across organizations. You can have up to 256 subnets in the /16 VPC.

Navigate to the VPC service and choose **Create VPC**, with the CIDR set as 10.0.0.0/16, as shown in the following screenshot:



Once this has been created, we will have to create six smaller subnets, each with 251 IPs, using the /24 CIDR range.

On the VPC dashboard, navigate to **Subnets**, and then **Create subnet**, as shown in the following screenshot:

The screenshot shows the 'Create subnet' form within the AWS VPC Subnets section. It includes fields for Name tag (webtier-us-east-1a-public), VPC (vpc-0480bbd34ef62045c), CIDR (10.0.0.0/16), Availability Zone (us-east-1a), and IPv4 CIDR block (10.0.0.0/24). A note at the bottom indicates that * Required fields must be filled.

Name tag	webtier-us-east-1a-public	i	
VPC*	vpc-0480bbd34ef62045c	i	
VPC CIDRs	CIDR	Status	Status Reason
	10.0.0.0/16	associated	
Availability Zone	us-east-1a	i	
IPv4 CIDR block*	10.0.0.0/24	i	

* Required Cancel Create

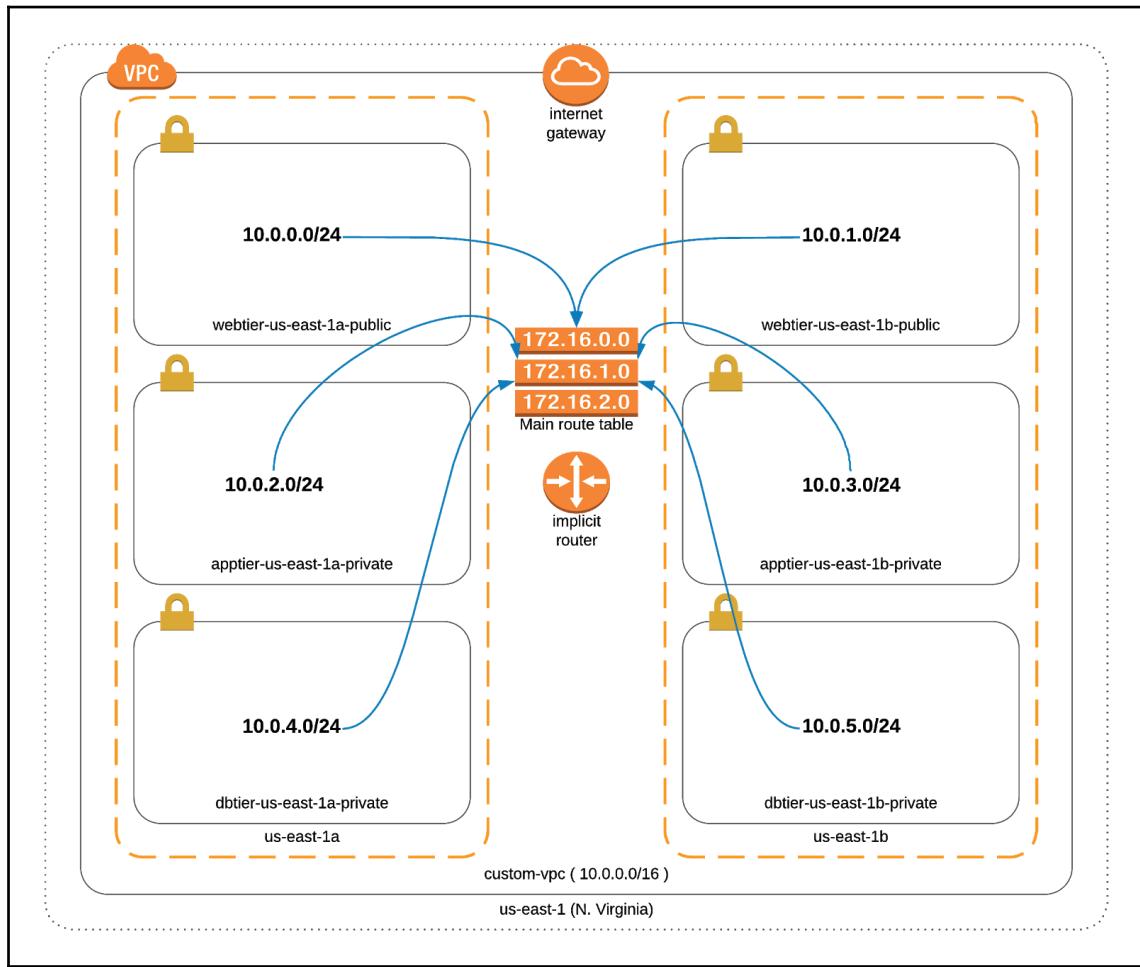
Repeat this process until you have the following structure:

Name	State	VPC	IPv4 CIDR	Available IPv4	Availability Zone
webtier-us-east-1a-public	available	vpc-0edd6d5450de65e8a custom-vpc	10.0.0.0/24	251	us-east-1a
webtier-us-east-1b-public	available	vpc-0edd6d5450de65e8a custom-vpc	10.0.1.0/24	251	us-east-1b
apptier-us-east-1a-private	available	vpc-0edd6d5450de65e8a custom-vpc	10.0.2.0/24	251	us-east-1a
apptier-us-east-1b-private	available	vpc-0edd6d5450de65e8a custom-vpc	10.0.3.0/24	251	us-east-1b
dbtier-us-east-1a-private	available	vpc-0edd6d5450de65e8a custom-vpc	10.0.4.0/24	251	us-east-1a
dbtier-us-east-1b-private	available	vpc-0edd6d5450de65e8a custom-vpc	10.0.5.0/24	251	us-east-1b

Every subnet is associated with the main route table by default. This main route table only permits local VPC traffic, as shown in the following screenshot:

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No

The following is the current VPC architecture, where every subnet is associated with the main route table:



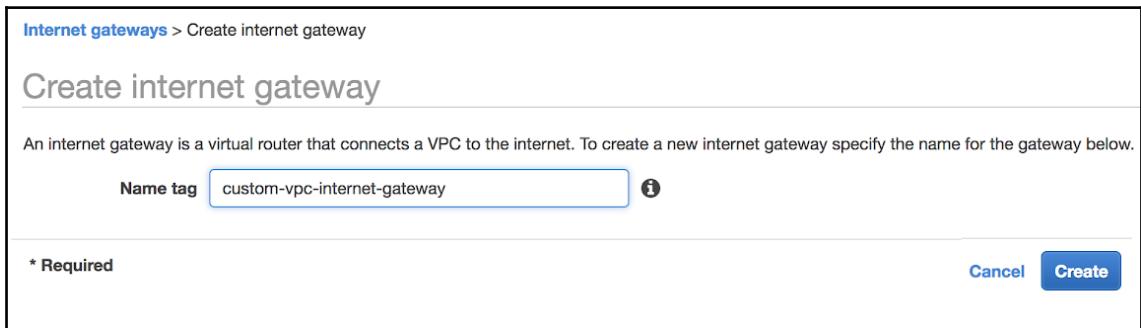
We want to specify custom routing. Let's create custom route tables, to control the routing schemes for public and private traffic.

Public traffic

To enable internet traffic, we have to provision a virtual public internet gateway. This internet gateway is a redundant, horizontally-scaled service that the VPC provides in order to route internet traffic from and to the VPC subnets.

Provision an internet gateway, as follows:

1. To create an internet gateway, navigate to **VPC | Internet gateways | Create internet gateway**, as shown in the following screenshot:



Once it has been created, you can verify that the resource is not attached to any VPC, and that it is currently in the detached state.

2. Let's select the internet gateway resource and choose **Actions | Attach to VPC**. This resource can only be attached to one VPC at a time; once you have selected the current VPC, you will see the state as **attached**, as shown in the following screenshot:

Name	ID	State	VPC
custom-vpc-internet-gateway	igw-03b44c3df0ed09c2b	attached	vpc-0edd6d5450de65e8a custom-vpc

Up to this point, we have associated an internet gateway to the VPC, but to successfully allow traffic from and to the instances to the internet, we have to create a custom route table, as follows:

1. Navigate to **Route Tables** and choose **Create Route Table**. This route table can be assigned to multiple subnets that need public network routing:

Create Route Table

A route table specifies how packets are forwarded between the subnets within your VPC, the Internet, and your VPN connection.

Name tag: public-route-table vpc-0edd6d5450de65e8a | custom-vpc i

VPC: vpc-0edd6d5450de65e8a | custom-vpc i

Cancel Yes, Create

- Once it has been created, select the newly created route table, and, under **Routes**, edit the route table entries. Use `0.0.0.0/0` to designate any IP as the traffic origin destination, and use the internet gateway ID for the target of the traffic:

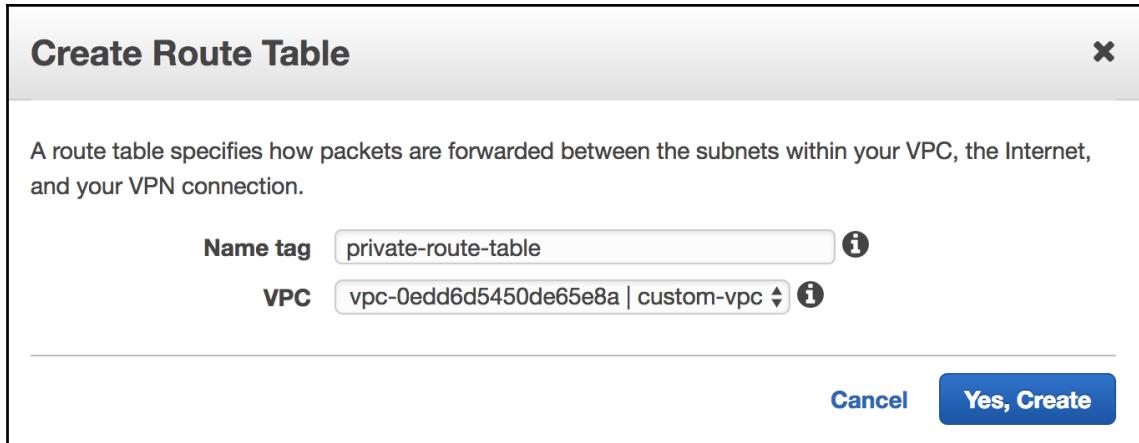
The screenshot shows the AWS Route Table configuration interface. At the top, there's a header with tabs for Name, Route Table ID, Explicitly Associated, Main, and VPC. Below the header, the route table is listed with its ID (rtb-0ba...), name (public-route-table), and associated VPC (vpc-0480bbd34ef62045c | custom-vpc-2). A sub-header "rtb-0ba... | public-route-table" is present. Below this, there are tabs for Summary, Routes (which is selected), Subnet Associations, Route Propagation, and Tags. Under the Routes tab, there's a "View: All rules" section. A table lists routes with columns: Destination, Target, Status, Propagated, and Remove. There are two rows: one for "10.0.0.0/16" with Target "local" and Status "Active" (highlighted in green); and another for "0.0.0.0/0" with Target "igw-0acd8904a3add4314 | vpc2-internet-gateway" and Status "Active" (highlighted in green). A "Remove" button is next to the second row. At the bottom left is a "Save" button, and at the bottom right is a "Cancel" button.

Destination	Target	Status	Propagated	Remove
10.0.0.0/16	local	Active	No	×
0.0.0.0/0	igw-0acd8904a3add4314 vpc2-internet-gateway	Active	No	×

This rule will forward all network traffic to the internet gateway of our VPC.

Private traffic

Create another route table and name it `private-route-table`. Leave the route table entries as-is. Currently, the route table only permits local (private) traffic, as shown in the following screenshot:



This time, the destination is the VPC CIDR space (`10.0.0.0/16`), and the target is **local**, meaning that only local traffic is allowed for any host.

Middleware, databases, and sensitive information must be protected in the private subnet, because doing so will avoid exposing unnecessary resources to the public. By using **Network Address Translation (NAT)** services, the flow of traffic from the private space to the internet can be enabled and controlled. You can use a custom AMI to perform this network function by running an EC2 instance. The more services deployed onto the private subnets, the greater the network bandwidth necessary to communicate to the internet through this proxy service. From an architectural point of view, this single server will represent a single point of failure, and the network throughput will be restricted by the network interface of the instance, depending on the instance's size.

To improve the design, you can opt to use a **NAT gateway** (a managed NAT service that maps private IPs to a public address and prevents the internet from initiating a connection with your instances). This service can burst up to 10 Gbps, and it is highly available:

1. To create a NAT gateway, navigate to **NAT Gateways** and choose **Create NAT Gateway**.

To create the gateway service, you will need an Elastic IP to specify the public subnet on which the service will be configured:

NAT Gateways > Create NAT Gateway

Create NAT Gateway

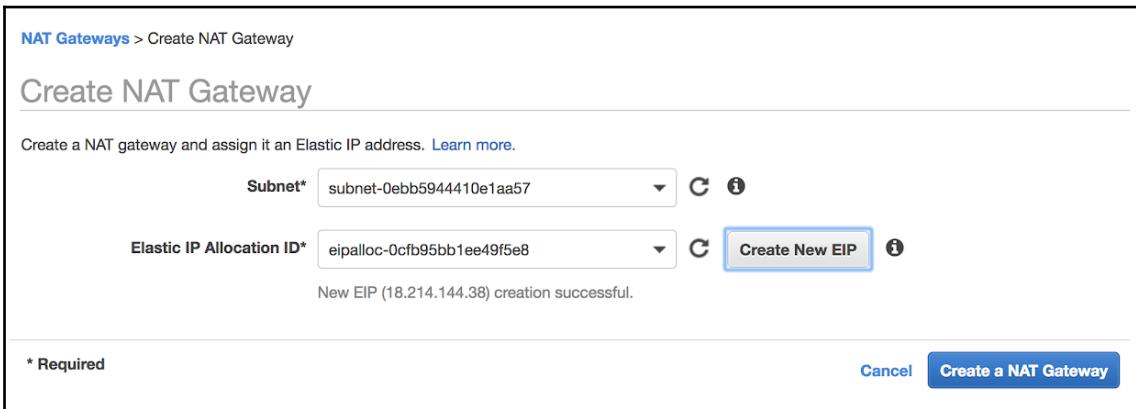
Create a NAT gateway and assign it an Elastic IP address. [Learn more.](#)

Subnet* C ⓘ

Elastic IP Allocation ID* C ⓘ [Create New EIP](#) ⓘ

New EIP (18.214.144.38) creation successful.

* Required [Cancel](#) [Create a NAT Gateway](#)



Once the NAT gateway has been created, the user must modify the private route tables for the private subnets, making sure that the destination is 0.0.0.0/0 and that the **Target** is the NAT ID, as shown in the following screenshot:

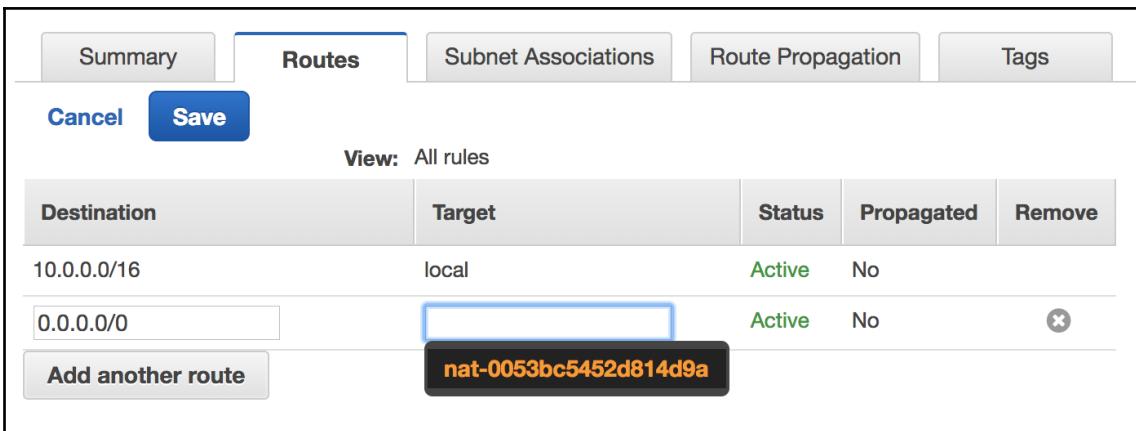
Summary Routes Subnet Associations Route Propagation Tags

[Cancel](#) [Save](#)

View: All rules

Destination	Target	Status	Propagated	Remove
10.0.0.0/16	local	Active	No	
0.0.0.0/0	<input type="text" value="nat-0053bc5452d814d9a"/>	Active	No	

[Add another route](#)



Now, we will have to update the subnet associations to make sure that the private subnets are related to the private route table.

2. Select the **private-route-table**, go to the **Subnet Associations** tab, and then choose edit. Use the checkbox for every private subnet, as follows:

The screenshot shows the AWS Route Table Associations page for the route table `rtb-0b6868f05b5c71f4d | private-route-table`. The **Subnet Associations** tab is selected. A modal dialog is open, showing the **Edit** button highlighted. Below it, a table lists subnets and their associations:

Associate	Subnet	IPv4 CIDR	IPv6 CIDR	Current Route Table
<input type="checkbox"/>	subnet-022f21f36ab0980d2 webtier-us-east-1a-public	10.0.0.0/24	-	Main
<input type="checkbox"/>	subnet-0cbe1369482033b54 webtier-us-east-1b-public	10.0.1.0/24	-	Main
<input checked="" type="checkbox"/>	subnet-0ebb5944410e1aa57 apptier-us-east-1a-private	10.0.2.0/24	-	Main
<input checked="" type="checkbox"/>	subnet-00c6c9e3eaeb056fd apptier-us-east-1b-private	10.0.3.0/24	-	Main
<input checked="" type="checkbox"/>	subnet-05ee87db010837cd0 dbtier-us-east-1a-private	10.0.4.0/24	-	Main
<input checked="" type="checkbox"/>	subnet-053a4227e1916373a dbtier-us-east-1b-private	10.0.5.0/24	-	Main

Repeat the same process for the public route table, adding the public subnets, as shown in the following screenshot:

The screenshot shows the AWS Route Table Associations page for the route table `rtb-0bc801f1b3b3b4eda | public-route-table`. The **Subnet Associations** tab is selected. A modal dialog is open, showing the **Edit** button highlighted. Below it, a table lists subnets and their associations:

Subnet	IPv4 CIDR	IPv6 CIDR
subnet-022f21f36ab0980d2 webtier-us-east-1a-public	10.0.0.0/24	-
subnet-0cbe1369482033b54 webtier-us-east-1b-public	10.0.1.0/24	-

A message at the bottom states: "The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:"

Subnet	IPv4 CIDR	IPv6 CIDR
--------	-----------	-----------

All your subnets are associated with a route table.

Security groups

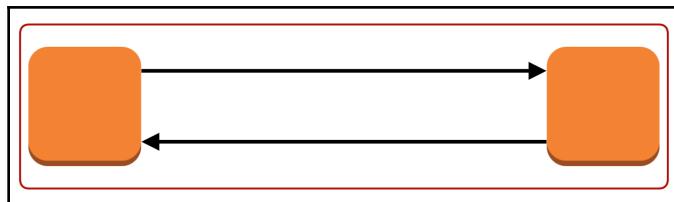
Security groups act as virtual firewalls at the instance level by restricting all inbound traffic and only allowing relevant traffic to the instance. These rules work at the instance level, and are associated with the virtual network interface.

If an instance is not assigned to a custom security group, it will be assigned to the default security group. This security group has the following rules:

Inbound		
Source	Protocol	Port range
The same security ID (sg-0axxx)	All	All
Outbound		
Source	Protocol	Port range
0.0.0.0/0	All	All

Default security group

This means that two instances associated with the same security group can communicate with each other, denying any kind of ingress traffic.



Security groups are stateful, meaning that any rule allowed for ingress is also allowed to egress the instance. Only allow rules can be specified, because anything that is not permitted is denied by default.

To test this behavior, you can launch an EC2 instance in the public subnet, log in to the instance, and perform a `PING` to `aws.amazon.com`; you will receive a `PING` response, but if you try to `PING` from your local computer, the response will be denied.

Creating a security group

Navigate to EC2 and create a security group. You must specify **SSH traffic** from the drop-down menu from your current IP address to restrict management connectivity to only your workstation. Additionally, you can add a rule, like ICMP echo requests, so that the instance can respond to external PING commands:

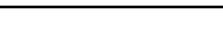
The screenshot shows the 'Create Security Group' dialog box. At the top, there are fields for 'Security group name' (set to 'admin-security-group'), 'Description' (set to 'SSH traffic'), and 'VPC' (set to 'vpc-0edd6d5450de65e8a | custom-vpc'). Below these, under 'Security group rules:', the 'Inbound' tab is selected. A table lists two rules:

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	My IP 187.207.87.71/32	SSH from Admin
Custom ICMP	Echo Request	N/A	Anywhere 0.0.0.0/0, ::/0	Ping requests

At the bottom right of the dialog are 'Cancel' and 'Create' buttons.

Chaining security groups

Inbound communications can be allowed from a CIDR range or another security group. This way, network traffic can be compartmentalized if we use multiple security groups for every functional layer in the topology. For example, suppose that we have a three-tier web application with the rules shown in the following diagram:

Inbound Rule			
Security Group Name	Allow	Source	Comment
WebTierELB	TCP 443	0.0.0.0/0	
WebTier	TCP 80	WebTierELB	
AppTierELB	TCP 8080	WebTier	
AppTier	TCP 8080	AppTierELB	
DBTier	TCP 3306	AppTier	

Bastion host

We will now test the connectivity of the NAT gateway by connecting to the internet from an EC2 instance running in a private subnet, as follows:

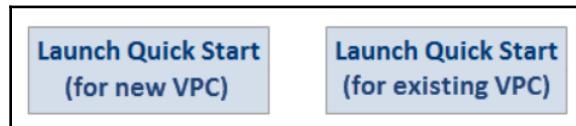
1. Navigate to the EC2 dashboard, then create a new instance, choosing the private subnet **1a** (where our NAT gateway is actually running).

As you can see, this instance has no public IP assigned; this is correct, because it is a private resource. Now, we will introduce a security best practice called **bastion host**.

The bastion host is an instance that performs a security function, like SSH server for Linux, or RDP gateway for Microsoft. This instance is always running in the private subnet, and it controls connectivity from administrators to private instances. Exceptional security must be put in place in this server, like OS hardening, monitoring, and port forwarding.

The AWS team has crafted a solution in CloudFormation so that you can deploy the bastion host solution that follows the best practices; for the purposes of this example, we will use <https://amzn.to/2wGA1U8>.

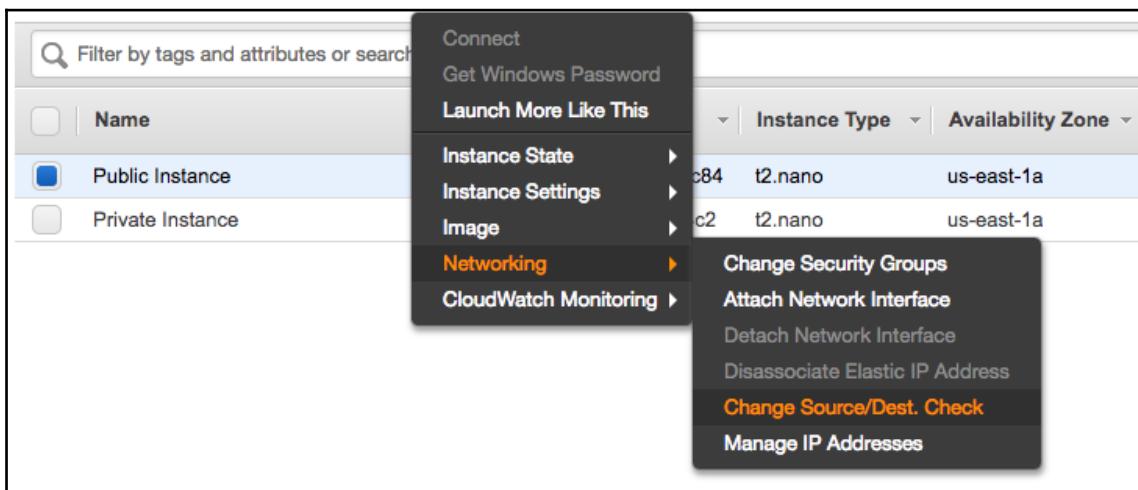
2. Click on the **Launch Quick Start (for existing VPC)** option. Make sure that it is placed in the same region as your VPC:



The bastion host must be placed in the public subnets across AZs for resiliency.

3. Choose one of the **web-app-tier** public subnets, fill in all of the parameters, and set the value of **Enable TCP Forwarding** to **true**.
4. Choose **Next**, and check the option, **I acknowledge that AWS CloudFormation might create IAM resources**, in the **Capabilities** section.

Every EC2 instance has a restriction if the traffic it receives is not intended to be the destination. Proxies, or NAT instances, are good examples of this; to allow the free passing of network packets, you have to change **Change Source/Dest. Check** attribute, as follows:



For Linux instances, you can use the following command to enable port forwarding and connect to the bastion host via SSH:

```
ssh -i path/to/private.pem -A ec2-user@bastion-host-dns
```

The `-A` option configures the session to enable agent forwarding. Now, just issue an `ssh ec2-user@private-ip` command to reach the private server in the private subnet.

Hybrid deployment

In this scenario, companies maintain applications and services running locally on their premises, whether they are physical servers or private clouds. Customers may decide to migrate one simple application, or a complete data center, comprised of hundreds of applications, with a global footprint. To achieve this, customers can employ hybrid solutions to use the cloud for new projects, to migrate existing ones, or as a DR site, running on a standby replica in AWS.

Hybrid networks can be configured as an extension of the on-premises data center, using hardware or software VPNs, with a preference on the former. Hardware VPNs are preferred because they are more robust and perform better. They use hardware accelerated encryption mechanisms and are more stable than their software counterparts.

To interconnect your premises network with a VPC, you need a **virtual private gateway**—a resource that acts as the network concentrator for the AWS side. On the customer side, you would configure **customer premises equipment (CPE)**, which could be a network router; this component is known as a **customer gateway**.

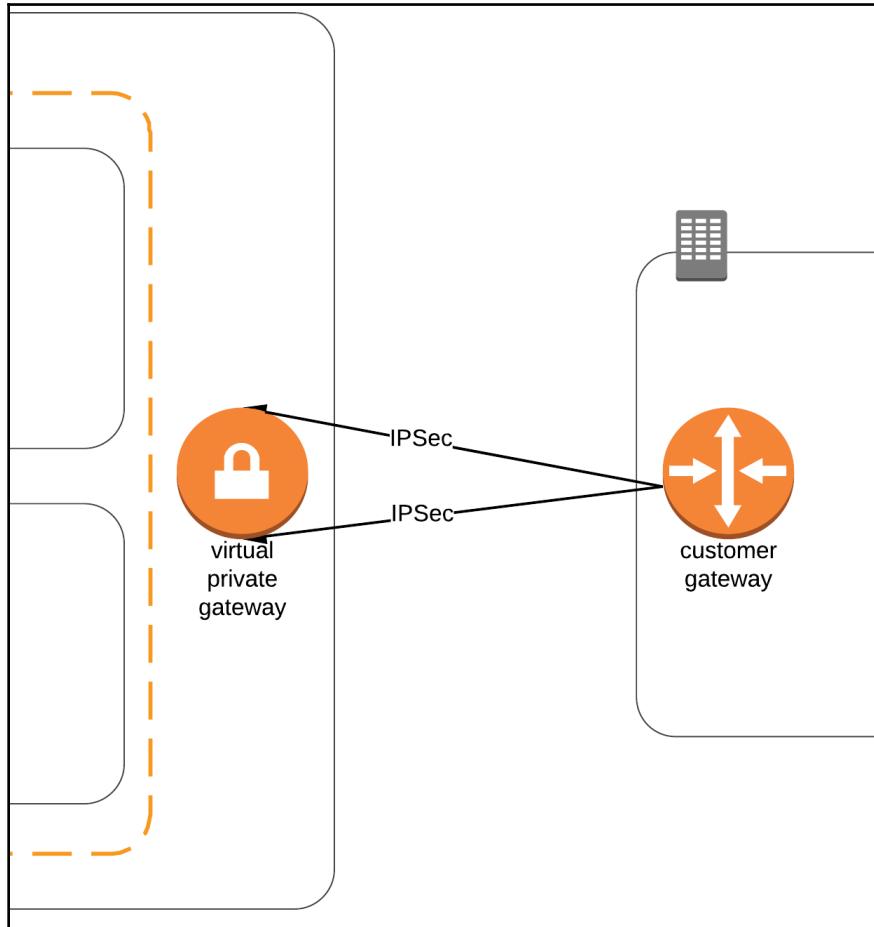
Software VPNs

This is the least preferred option for private connectivity, as it doesn't support the IPSec protocol; instead, this solution can use a variety of security transmission mechanisms such as **Point-to-Point Tunneling Protocol (PPTP)**, or SSLv3/TLSv1. Solutions like OpenVPN can be found on the AWS Marketplace. This is a straightforward solution, but it is not a resilient or robust one. It can be used for different scenarios, where throughput is low and cost is a restriction.

Software VPNs can solve problems like cross-region VPC connectivity. If your current regions do not support VPC peering, you can use a software appliance that is running on both regions to bring connectivity with a high degree of compatibility.

Static hardware VPNs

This kind of connectivity is a managed service from AWS, and it is implemented via a resilient object (a virtual private gateway), by allocating two VPN endpoints. If the customer's router does not support the BGP protocol, a static VPN must be configured:



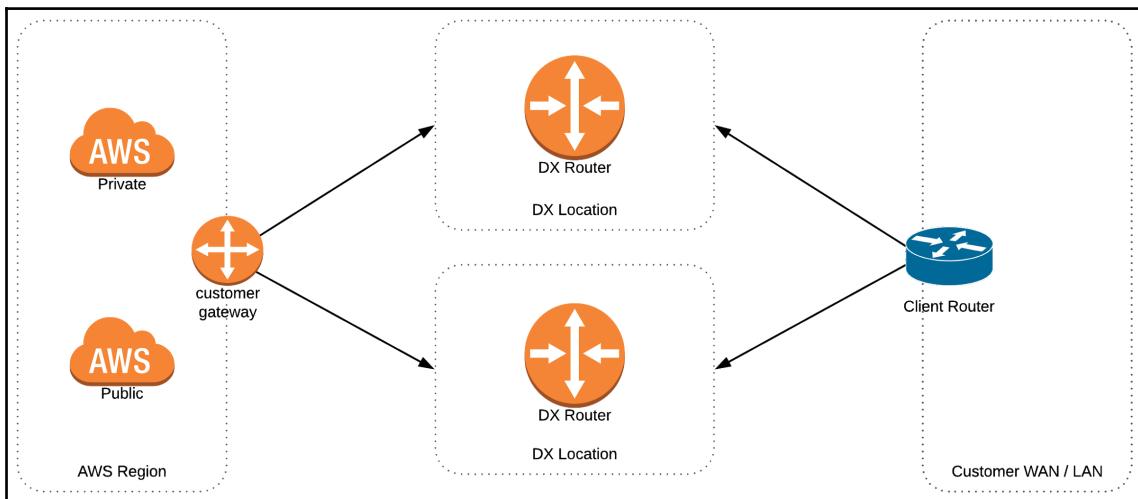
Dynamic hardware VPNs

The customer router, known as the **customer gateway**, must support the **Border Gateway Protocol (BGP)**, and a static public IP must be in place. From the AWS side (your VPC), a virtual private gateway represents the object on which all connections will be managed by the VPC. Additionally, the VPN connection will associate one VGW with a CGW for IPSec tunnels.

The route tables must be updated with the subnet information, and you must enable route propagation so that neighbors learn about other routes, and new on-premises networks can be added dynamically.

Direct Connect (DX)

Direct Connect provides customers with the ability to interconnect their data centers to AWS privately. This connection is done via optic fiber, and improves the consistency and throughput of remote connections. The service can be requested with ports up to 10 Gbps (via a Direct Connect partner) or 1 Gbps and below (directly from the marketplace):



The preceding diagram shows, at a high level, how the DX provider connects both sides in its facilities. Once the connection has been made, you can manage connectivity via virtual interfaces. Public VIFs are designed to carry traffic from your location to public AWS service endpoints like S3 and DynamoDB. Private VIFs are used to connect your on-premises networks to several AWS VPCs.

Consider using Direct Connect if you plan to transfer big quantities of data on an ongoing basis, and performance, stability, and throughput are critical for the business.

Storage gateway use cases

When a migration or disaster recovery project must be used, hybrid storage solutions must be evaluated, and that is where the **storage gateways** come to the rescue. With virtualization methods, you can drop in a storage solution that is compatible with your existing products. It is imperative that the cloud solution can be integrated with on-premises drivers and standard storage protocols, like NFS v3, v4.1, and iSCSI. You can use local caching capabilities and security with SSL and AES 256.

You can use a storage gateway as a disaster recovery strategy for continuous incremental backups failing over to the cloud.

Network filesystems with file gateways

The NFS protocol is widely supported by many operating systems at the application level; for distributed filesystems, SMB is also supported. The storage service on the AWS side is Amazon S3, which is highly durable and resilient by design; so, every file on the premises side becomes an object in the cloud. The file gateway interface has an API called **CacheRefresh**, which allows for refreshing the client caches programmatically so that they are notified when new objects have been uploaded or changed. You can run this gateway on premises or in an EC2 instance.

Block storage iSCSI with volume gateway – stored

The iSCSI protocol provides a transparent interface, and is a good alternative for SANs. Operating systems are agnostic of the storage medium, and they initiate communications to target devices for block operations. The volume gateway is backed by Amazon S3, and data is stored as blocks that can be restored on EBS volumes in the cloud; this is a perfect choice to restore the data layer with EC2 instances.

Block storage iSCSI with volume gateway – cached

Your data is stored on premises and is then to the cloud. For low latency access in the local network, a cache disk is used to improve the user experience by accessing recent data locally while a second disk is used for buffering transfer jobs into S3. This gateway supports up to 32 volumes with 32 TB each, for a total of 1 PB per gateway.

Virtual tape library iSCSI with a tape gateway

The **virtual tape library (VTL)** interface allows you to store virtual tapes in S3 and Glacier. This appliance includes a tape drive and a media changer, which is used to emulate tape robots.

Every volume gateway can run in a virtual machine on the on-premises side. Be aware of the hardware specifications and use cases before provisioning the appliance and choosing a gateway solution. Storage gateways can be used for continuous data synchronization, disaster recovery, and as a drop-in solution for hybrid storage architectures.

The Database Migration Service

Database migration is always a complex task, and it must be evaluated, validated, and executed carefully. The AWS Database Migration Service deals with all of the complexity of executing a replication, multi-engine job via a migration instance, and keeps the origin and destination databases in sync. DMS can be used to migrate existing OLTP and OLAP workloads into the AWS cloud.

DMS can be used for one-time and ongoing live migrations with zero downtime. Multiple databases (shards) can be consolidated into a single, master database. This results in a lot of flexibility for service-oriented architectures, like microservices and database federation patterns.

If your migration can be done with downtime, the simplest strategy is to use vendor tooling to perform the activity, like `.bak` files, `pg_dump`, or textual logic dump files (`.sql`). You can select which tables, schemas, or databases to replicate on a per-job basis.

If the migration cannot tolerate service downtime, you must consider the use of DMS. You must also do the same for other use cases like transformations between engines and cross-region replication in regions where this service feature is not available.

Homogeneous migration

Same engine migrations, such as Oracle to Oracle, are supported with minimal impact to existing applications. You can opt to upgrade the database versions and modernize your data layer.

Homogeneous migrations may not be the best option for DMS; it is always important to establish a migration objective and use the best tool for the job.

The AWS Schema Conversion tool

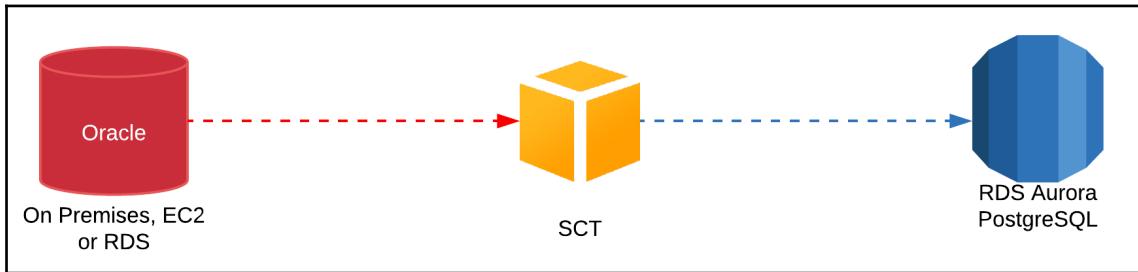
The AWS Schema Conversion tool can be used to convert from different database engines and create an application assessment report. This report summarizes database objects and shows the compatibility between the source and target, as shown in the following screenshot:



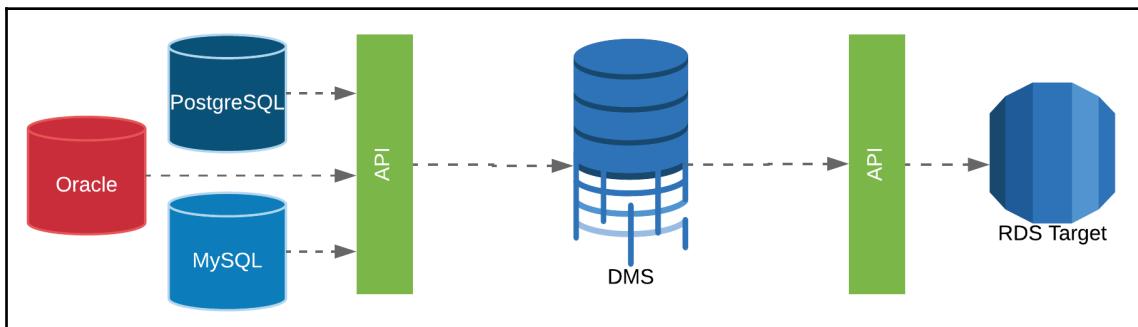
You can even use data extraction agents to convert the data warehouses to Amazon Redshift. The SCT tool can perform SQL statement conversions between engines.

Heterogeneous migrations

In the planning phase, SCT can be used to move the schema first by converting objects automatically. Several non-convertible objects will be manually rewritten by a DBA, with the detailed recommendations provided by the migration assessment report:



In the execution phase, the retrieval and copy of continuous data can be done directly by DMS; this copy can be paused and resumed at any time, and it is also fault-tolerant, as shown in the following diagram:



The heterogeneous target can also be configured to the following destinations, and you can leverage different patterns for storing and accessing your data:

- S3
- Aurora
- DynamoDB
- Redshift

Summary

In this chapter, you learned about one of the most important factors of operating services in the cloud: network design. We covered the various integration options available to us. We also discussed hybrid architectures, including the most popular deployment modes in the cloud or hybrid. In the hybrid deployment mode, multiple connectivity options were discussed, like VPNs and Direct Connect. The AWS Storage Gateway was also discussed, along with the use cases for each type of gateway. Finally, the Database Migration Service came into play, with several use cases for one-time and continuous database migrations.

Further reading

- **Hybrid Cloud Architectures with AWS:** <https://aws.amazon.com/enterprise/hybrid/>
- **AWS Migration Whitepaper:** <https://d1.awsstatic.com/whitepapers/Migration/aws-migration-whitepaper.pdf>
- **Amazon VPC Limits:** <https://docs.aws.amazon.com/vpc/latest/userguide/amazon-vpc-limits.html>
- **Frequently asked questions about Amazon VPC:** <https://aws.amazon.com/es/vpc/faqs/>
- **AWS Direct Connect FAQs:** <https://aws.amazon.com/directconnect/faqs>
- **AWS Storage Gateway FAQs:** <https://aws.amazon.com/storagegateway/faqs/>

5 Resilient Patterns

Application and service availability can be extended to the regional level, AWS provides simple and yet powerful mechanisms to achieve global audiences and the option to use different construction blocks to build resiliency such as disaster recovery options and super available applications. This chapter presents a pattern to use Route 53 as the de facto solution to achieve higher levels of service by introducing DNS failover.

The following topics will be covered in this chapter:

- Route 53
 - Health checks
 - Record types

Technical requirements

Register a domain name with Route 53, for more information use the following docs: <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/registrar.html>.

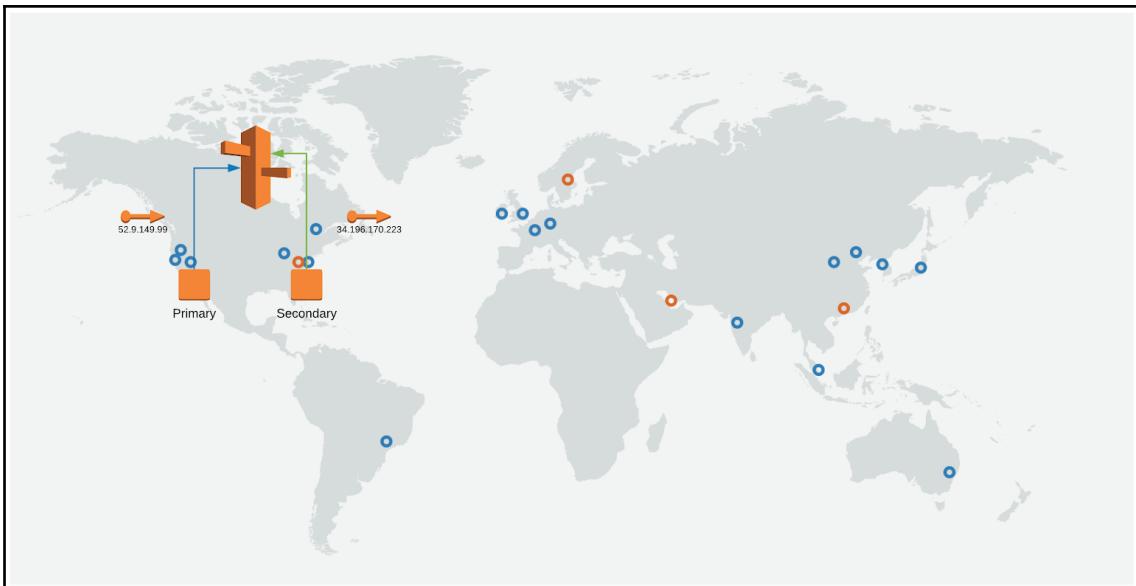
Route 53

Our deployment will consist of two EC2 instances configured with regional availability. Route 53 provides the following routing types:

- **Simple routing:** Use this routing for simple servers, this is used for simple resolution
- **Geolocation routing policy:** Route traffic with the minimal latency
- **Geoproximity routing policy:** This is configured via traffic flow for complex routing

- **Multivalue answer routing policy:** Randomize up to 8 records
- **Weighted routing policy:** Assign specific weights to records (for example, 80% - 20% split)
- **Failover routing policy:** For active-passive scenarios

We will use the failover routing policy to create the following architecture:



Primary	N. California
Secondary	N. Virginia

To provision your environment launch an EC2 instance on each region (**us-east-1** and **us-west-1**). You can use the following script for the user data from `user-data.txt` present on GitHub:

```
#!/bin/bash -ex
yum -y update
yum -y install httpd php
chkconfig httpd on
wget
https://s3.amazonaws.com/aws-certified-solution-architect-associate-guide/reliability/index.php -O /var/www/html/
service httpd start
```

It is a good idea for this kind of scenarios to use Elastic IPs so provision one for each region and associate them with the instance.

Health checks

Go through the following steps to perform health checks:

1. Navigate to Route 53 and click on **Create health check**:

The screenshot shows the AWS Route 53 Health Checks console. On the left, there's a navigation sidebar with links like Dashboard, Hosted zones, Health checks (which is selected and highlighted in orange), Traffic flow, Traffic policies, Policy records, Domains, Registered domains, and Pending requests. The main content area has a title "Welcome to Route 53 health checks". It includes a "Create health check" button and sections for "Health check concepts". There are two main cards: "Availability and performance monitoring" (with an icon of a computer monitor showing a checkmark) and "DNS failover" (with an icon of a shield containing a stethoscope and a plus sign). Both cards have descriptive text and "Learn more" links.

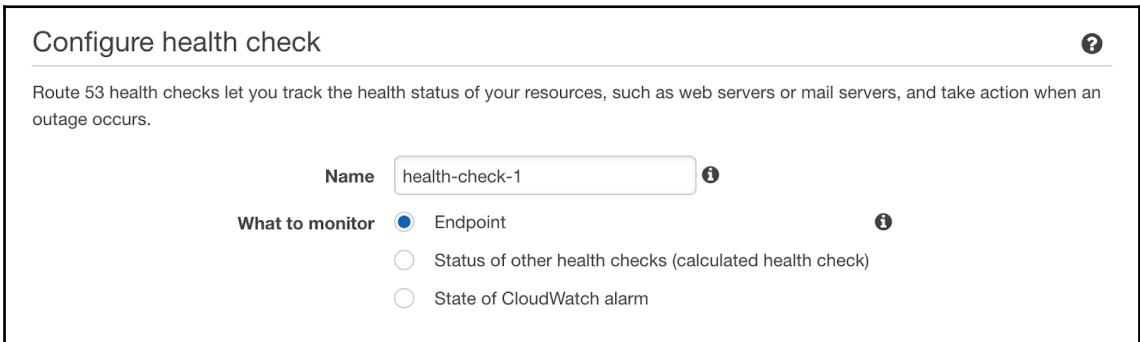
2. You must specify a name, use `health-check-1` or something that makes sense to your monitoring and use the **Endpoint** option on **What to monitor**.

Configure health check

Route 53 health checks let you track the health status of your resources, such as web servers or mail servers, and take action when an outage occurs.

Name ⓘ

What to monitor Endpoint ⓘ
 Status of other health checks (calculated health check)
 State of CloudWatch alarm



3. You must configure the endpoint, this can be an IP address or a domain name. Choose the IP address and paste the Elastic IP of the primary server.

Monitor an endpoint

Multiple Route 53 health checkers will try to establish a TCP connection with the following resource to determine whether it's healthy.
[Learn more](#)

Specify endpoint by IP address Domain name

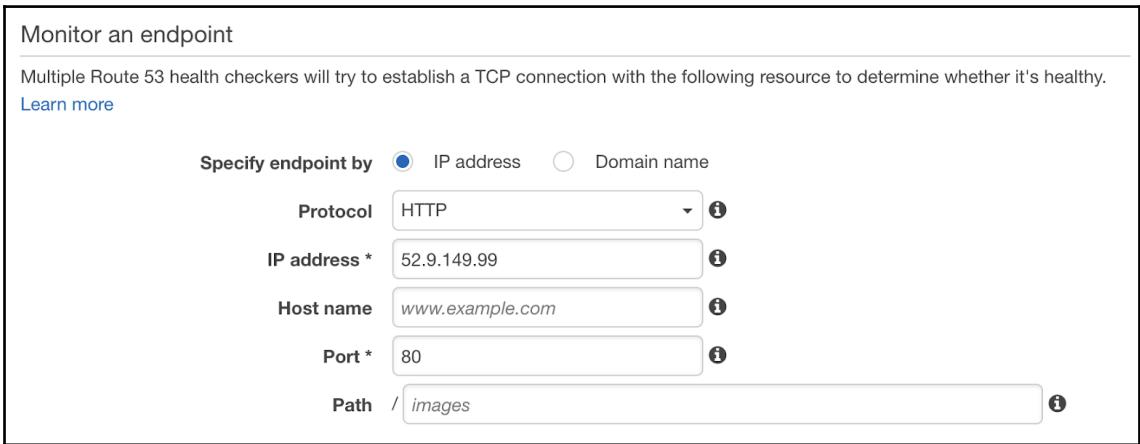
Protocol ⓘ

IP address * ⓘ

Host name ⓘ

Port * ⓘ

Path / ⓘ



4. In the advanced configuration section set the **Request interval** to **Fast (10 seconds)** and the **Failure threshold** as **2** to be able to fail fast. Leave all other options as default and click **Next**:

▼ Advanced configuration

Request interval Standard (30 seconds) Fast (10 seconds) [i](#)

Failure threshold * [i](#)

String matching No Yes [i](#)

Latency graphs [i](#)

Invert health check status [i](#)

Disable health check By default, disabled health checks are considered healthy. [Learn more](#) [i](#)

Health checker regions Customize Use recommended [i](#)

US East (N. Virginia)
US West (N. California)
US West (Oregon)
EU (Ireland)
Asia Pacific (Singapore)
Asia Pacific (Sydney)
Asia Pacific (Tokyo)
South America (São Paulo)

URL <http://52.9.149.99:80/> [i](#)

Health check type Basic + additional options: Fast Interval ([View Pricing](#))

* Required [Cancel](#) [Next](#)

5. You can configure a CloudWatch alarm to get notified when changes in the health status occur, for simplicity choose **No** and click on **Create health check**.

Get notified when health check fails

If you want CloudWatch to send you an Amazon SNS notification, such as an email, when the status of the health check changes to unhealthy, create an alarm and specify where to send notifications.

Create alarm Yes No [?](#)

* Required

Cancel Previous Create health check

6. In the Route 53 navigate to the **Hosted Zones** section and select your domain name hosted zone. In my case is the name of my website.

Domain Name	Type	Record Set Count	Comment
hipermediasoftware.com	Public	2	

7. You can see that after the registration of a domain name NS and SOA records are present.

Record Set Name		Type	Value	Evaluate Target Health	Health Check ID	TTL	Region	Weight
	ns-2031.awsdns-61.co.uk.			-	-	172800		
hipermediasoftware.com	NS	ns-466.awsdns-58.com. ns-1000.awsdns-61.net. ns-1118.awsdns-11.org.		-	-			
	SOA	ns-2031.awsdns-61.co.uk.	awsdns-hostmaster.amazon.com	-	-	900		

We will create a record set for this hosted zone.

Record types

Route 53 supports multiple record types like the following:

Record type	Use case	Example
A - AAAA	IP addressing	52.9.149.99
CNAME	Aliasing	elb.example.com or aws.com
MX	Email servers	10 mail.example.com

1. In our case, we will be configuring Elastic IPs so use the type A and use the primary IP address for the value. Change the **TTL (Seconds)** to **60** seconds.

The screenshot shows the 'Create Record Set' wizard in the AWS Route 53 console. The configuration is as follows:

- Name:** www.hipermediisoft.com
- Type:** A – IPv4 address
- Alias:** No
- TTL (Seconds):** 60
- Value:** 52.9.149.99
- Routing Policy:** Failover
- Failover Record Type:** Primary
- Set ID:** www-Primary
- Associate with Health Check:** Yes
- Health Check to Associate:** health-check-1

A large blue 'Create' button is at the bottom right.

2. The **Routing Policy** must be set to **Failover** and for record type this IP represents the primary server so choose **Primary**. It is necessary to associate this record with a health check, choose **Yes** and filter the **health-check-1**.
3. Do the same procedure for the secondary server, this time specifying the failover secondary resource with the secondary IP value.

Create Record Set

Name: www hipermediisoft.com

Type: A – IPv4 address

Alias: Yes No

TTL (Seconds): 60 1m 5m 1h 1d

Value: 34.196.170.223

IPv4 address. Enter multiple addresses
on separate lines.
Example:
192.0.2.235
198.51.100.234

Routing Policy: Failover

Route 53 responds to queries using primary record sets if any are healthy, or using secondary record sets otherwise. [Learn More](#)

Failover Record Type: Primary Secondary

Set ID: www-Secondary

Associate with Health Check: Yes No

When responding to queries, Route 53 can omit resources that fail health checks. [Learn More](#)

Health Check to Associate: health-check-1

Create

4. Now that we have created the primary and secondary records, let's visualize the behavior of the health check. As you can observe, the health checking process is made from several different sources (regions) to validate sanity and isolate global networking failures. We have a healthy target.

Name	Status	Description	Alarms	ID
health-check-1	18 minutes ago now Healthy	http://52.9.149.99:80/	No alarms configured.	573c019b-1393-4c31-ab8d-06d5983892aa

Health checker region	Health checker IP	Last checked	Status
Asia Pacific (Tokyo)	54.248.220.3	Oct 31, 2018 3:37:32 AM UTC	Success: HTTP Status Code 200, OK
Asia Pacific (Tokyo)	54.250.253.195	Oct 31, 2018 3:37:33 AM UTC	Success: HTTP Status Code 200, OK
Asia Pacific (Singapore)	54.255.254.195	Oct 31, 2018 3:37:29 AM UTC	Success: HTTP Status Code 200, OK
Asia Pacific (Singapore)	54.251.31.163	Oct 31, 2018 3:37:29 AM UTC	Success: HTTP Status Code 200, OK
Asia Pacific (Sydney)	54.252.79.131	Oct 31, 2018 3:37:37 AM UTC	Success: HTTP Status Code 200, OK
Asia Pacific (Sydney)	54.252.254.227	Oct 31, 2018 3:37:31 AM UTC	Success: HTTP Status Code 200, OK
EU (Ireland)	176.34.159.195	Oct 31, 2018 3:37:27 AM UTC	Success: HTTP Status Code 200, OK
South America (São Paulo)	177.71.207.131	Oct 31, 2018 3:37:32 AM UTC	Success: HTTP Status Code 200, OK
South America (São Paulo)	54.232.40.99	Oct 31, 2018 3:37:33 AM UTC	Success: HTTP Status Code 200, OK
US East (N. Virginia)	107.23.255.35	Oct 31, 2018 3:37:37 AM UTC	Success: HTTP Status Code 200, OK
US East (N. Virginia)	54.243.31.195	Oct 31, 2018 3:37:37 AM UTC	Success: HTTP Status Code 200, OK
US West (N. California)	54.183.255.163	Oct 31, 2018 3:37:30 AM UTC	Success: HTTP Status Code 200, OK
US West (N. California)	54.241.32.67	Oct 31, 2018 3:37:30 AM UTC	Success: HTTP Status Code 200, OK
US West (Oregon)	54.245.168.3	Oct 31, 2018 3:37:33 AM UTC	Success: HTTP Status Code 200, OK

5. We will simulate a downtime of our primary web server, to do that stop the primary server as shown up in the following screenshot:

6. Go back to the health check status page, and see the erratic behavior happening now.

The screenshot shows the AWS CloudWatch Metrics Health Checks console. At the top, there are buttons for 'Create health check', 'Delete health check', and 'Edit health check'. Below this is a search bar labeled 'Filter by keyword'. A table lists a single health check entry:

Name	Status	Description	Alarms	ID
health-check-1	Unhealthy	http://52.9.149.99:80/	No alarms configured.	573c019b-1393-4c31-ab8d-06d5983692aa

Below the table are tabs for 'Info', 'Monitoring', 'Alarms', 'Tags', 'Health checkers' (which is selected), and 'Latency'. Under 'Health checkers', there is a section to 'View current status' or 'View last failed check' with a 'Refresh' button. A table lists health checker regions and their details:

Health checker region	Health checker IP	Last checked	Status
Asia Pacific (Tokyo)	54.248.220.3	Oct 31, 2018 3:46:39 AM UTC	Failure: The health checker could not establish a connection within t...
Asia Pacific (Tokyo)	54.250.253.195	Oct 31, 2018 3:46:40 AM UTC	Failure: The health checker could not establish a connection within t...
Asia Pacific (Singapore)	54.255.254.195	Oct 31, 2018 3:46:42 AM UTC	Failure: The health checker could not establish a connection within t...
Asia Pacific (Singapore)	54.251.31.163	Oct 31, 2018 3:46:41 AM UTC	Failure: The health checker could not establish a connection within t...
Asia Pacific (Sydney)	54.252.79.131	Oct 31, 2018 3:46:36 AM UTC	Failure: The health checker could not establish a connection within t...
Asia Pacific (Sydney)	54.252.254.227	Oct 31, 2018 3:46:35 AM UTC	Failure: The health checker could not establish a connection within t...

7. Use your APEX to validate the service availability, in my case the website **hipermediisoft.com**, and perform a record test under **Hosted Zones** use the button **Test record set**. When the failover has been done Route 53 will update the underlying IP for the secondary failover website.

The screenshot shows the 'Check response from Route 53' tool. It has two main sections: 'Check response from Route 53' on the left and 'Response returned by Route 53' on the right.

Check response from Route 53

This tool returns values based on the settings in a Route 53 hosted zone. You can use it to see how Route 53 works — what value is returned to a DNS request given the values that you specified in your resource record sets. For geolocation and latency resource record sets, you can also simulate requests from a particular DNS resolver and/or client IP address to find out what response a client with that resolver and/or IP address would receive from Route 53. [Learn More](#)

Hosted zone: hipermediisoft.com.

Record name: www

Type*: A

Simulate sending DNS request from specific IP address (optional)

For geolocation and latency resource record sets, type the IP address of a DNS resolver. The **Response returned by Route 53** section displays the response that Route 53 returns to the specified IP address.

Resolver IP address: 192.0.2.25

More options

*Required

Response returned by Route 53

Response from Route 53 based on the following options.

DNS request sent to	Route 53
EDNS0 client subnet IP	24
DNS response code	NOERROR
Protocol	UDP
Response returned by	34.196.170.223
Route 53	(highlighted with an orange circle)

8. Return things to normal, start the primary server again and the health check must be changed to healthy again.

Check response from Route 53

This tool returns values based on the settings in a Route 53 hosted zone. You can use it to see how Route 53 works — what value is returned to a DNS request given the values that you specified in your resource record sets. For geolocation and latency resource record sets, you can also simulate requests from a particular DNS resolver and/or client IP address to find out what response a client with that resolver and/or IP address would receive from Route 53. [Learn More](#)

Hosted zone hipermediisoft.com.

Record name www

Type* A

Simulate sending DNS request from specific IP address (optional)

For geolocation and latency resource record sets, type the IP address of a DNS resolver. The **Response returned by Route 53** section displays the response that Route 53 returns to the specified IP address.

Resolver IP address 192.0.2.25

More options

*Required Get response

Response returned by Route 53

Response from Route 53 based on the following options.

DNS request sent to
Route 53

EDNS0 client subnet 24
IP

DNS response code NOERROR

Protocol UDP

Response returned by
Route 53 52.9.149.99

Test again the record set and in a short amount of time the primary IP will be returned by the DNS query.

This simple and powerful demo shows the strong capacities of the AWS Global Infrastructure and how to gain resilience at the multi-region level.

Summary

This chapter showed the service capabilities of Route 53 with an overview of the principal routing types available and performing an active-passive configuration for a multi-region deployed web server. We used health checks to perform broad testing and configured a public hosted zone, simulated a service failure and validated the automatic failover performed via Route 53.

Further reading

- **Supported DNS Record Types:** <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/ResourceRecordTypes.html>
- **Amazon Route 53 FAQs:** <https://aws.amazon.com/route53/faqs/>

6

Event Driven and Stateless Architectures

Technologies evolve and new patterns replace the old way of doing things. In the early days of the internet, there was no concept of state, every request to a web server responded with a resource like HTML and other resources. The user experience of the website was improved by designing persistence mechanisms such as sessions via cookies, disk, or memory. The web has evolved a lot since then. We have modernized web apps and we have open standards and a big ecosystem to create applications. Newer applications make use of the **Representational State Transfer** architectural style to create Restful web applications that use the HTTP protocol to transfer requests for web resources. In the following sections, we will understand the difference between stateful and stateless with several implementation reference architectures.

In this chapter, we will cover the following topic:

- Web application hosting
- Serverless application architecture
- Streaming data architecture

Technical requirements

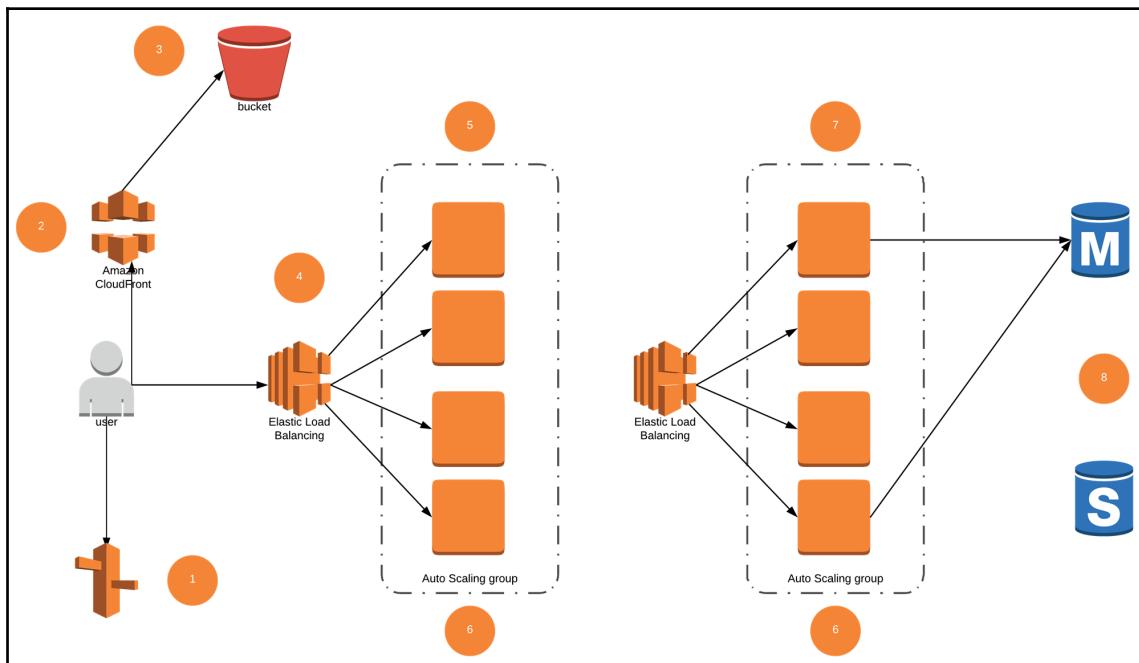
There are no technical requirements for this chapter.

Web application hosting

The three tier is a common architecture for web applications. This architecture is designed to be highly scalable by providing the web tier, which is responsible for the incoming requests from the Elastic Load Balancer. They act as a proxy layer, then these requests are propagated to the middleware tier.

Route 53

DNS requests are solved by **Route 53** (1) answering with a **CloudFront** distribution (2) with origin on S3 (3) for static assets and an ELB for dynamic data (4). The best practice is to offload all static data to S3 buckets by removing servers the responsibility to manage data. This decouples other layers (5) and provides maximum flexibility when resources are created or terminated by **Auto Scaling** (6).



The data layer is designed to use transactional data stored directly on RDS (8), this is a best practice to avoid having EC2 persistent servers that won't scale and will become a single point of failure. This way the middleware layer (7) instances use RDS to read and store data out of the instance.

Serverless application architecture

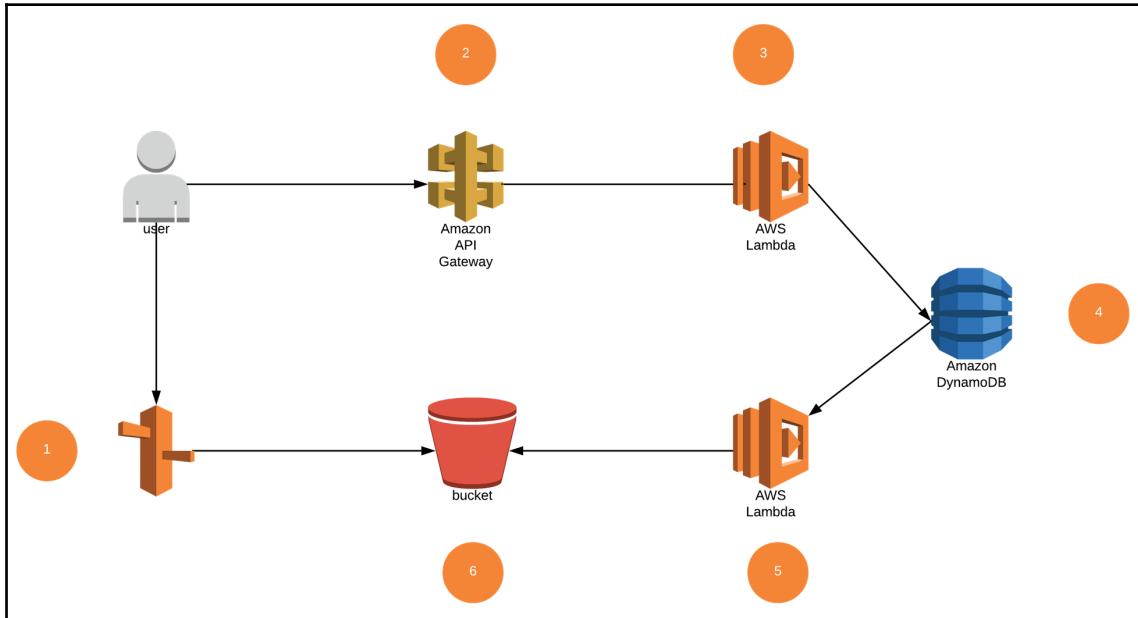
Serverless models are designed to be very efficient and cost-effective. They provide a high level of abstraction over the computing layer and they are 100% designed to be stateless.

The stateless design principle consists of offloading all persistence out of the compute service, in the following diagram several persistence services are used, such as Amazon S3, Amazon ElasticSearch, and DynamoDB.

This architecture represents a voting platform with a real-time dashboard that presents voting results in real time. Every vote is done via a mobile app. When the user votes, this event triggers an API gateway service endpoint invoking a custom resource like `POST /votes/candidate`.

When this vote is received by the API gateway resource an integration is executed, for this case is a Lambda function that provides the logic to increment the `valid_votes` table for the specific candidate.

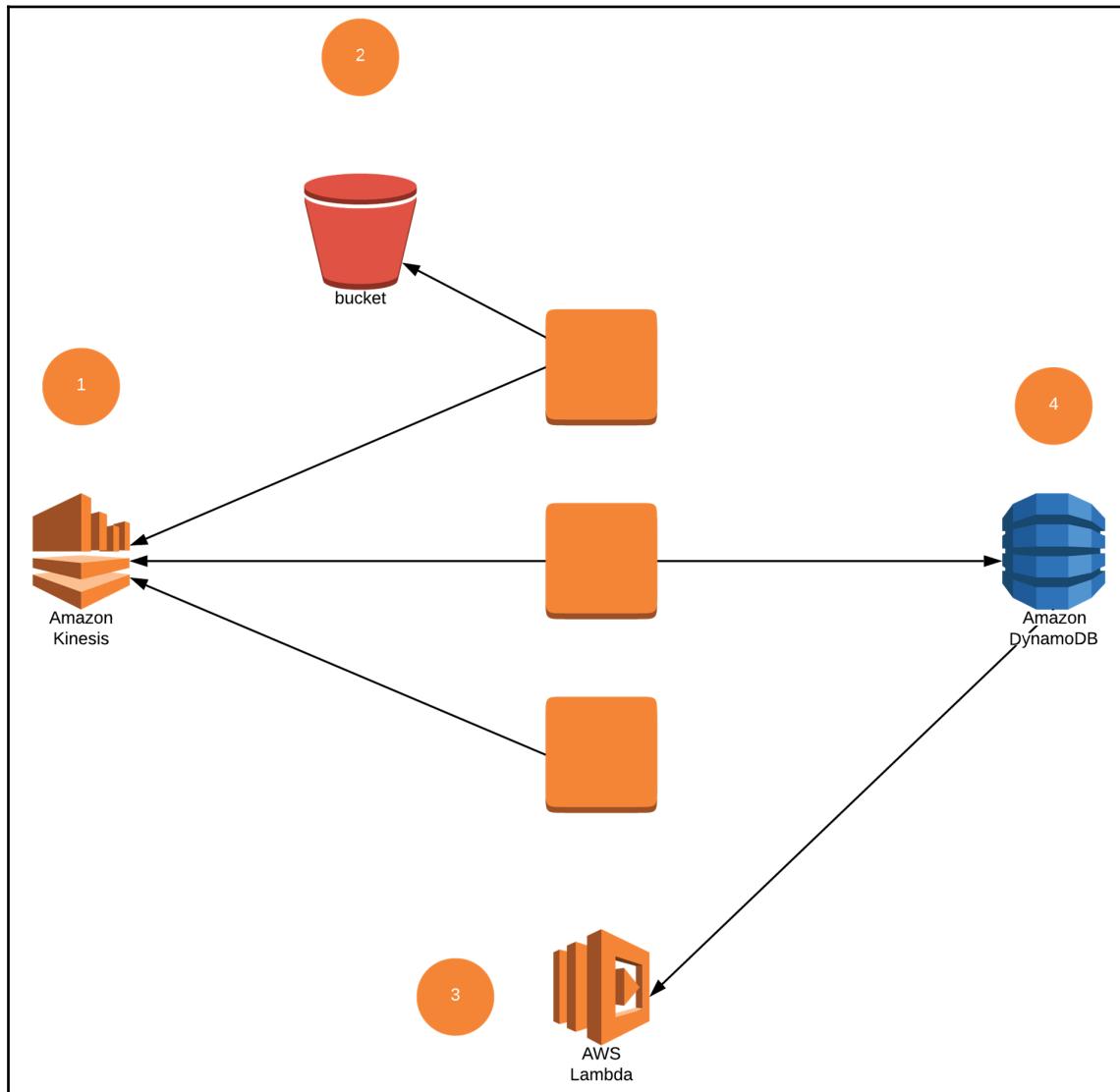
DynamoDB can be enabled to publish modifications to DynamoDB Streams, this streams can be processed with every new event by a second Lambda function that will increment a fragment of the static web page shown hosted on an S3 bucket.



Here every request is performed to **Route 53** (1), public hosted zones respond with different answers like **ElasticSearch** (3); this service can be used to index documents from **DynamoDB** (6) streams acting as a search domain for applications. Mobile apps can interact with this search cluster to achieve fast retrieval of data. All static information like media or images are stored directly on the S3 bucket for the application and dynamic backend logic is handled by the **API Gateway** (5) service. This API Gateway is integrating requests with AWS Lambda functions that are limited on execution time and do not provide local storage; here the **DynamoDB** (6) service is used to store key-value pairs and JSON documents and it is an excellent data store for temporary data needs as shopping carts or application data.

Streaming data architecture

AWS Kinesis is a scalable streaming service that can be used to ingest data and is fully managed. The ingestion can be done via the Kinesis Client Library or the Kinesis Streams API (1), in both cases data is durably stored up to 24 hours to different workflows can be executed like batch processing.



Kinesis can ingest data directly to S3 which makes a good integration to have the data out of the processing workers represented by the Kinesis apps. **AWS Lambda** (3) is used to process real-time data and the results of this processing are durably stored in **DynamoDB** (4).

This design promotes scalability on the worker tier (Kinesis Apps) by decoupling the storage to managed services for different purposes.

Summary

This chapter presented several reference architectures that showed how to offload state to external services by decoupling systems architecture and improving on several service domains. We analyzed the full paths for scalable web applications, serverless applications, and streaming data architectures.

Further reading

Following documentation will help you gain more information on the topic:

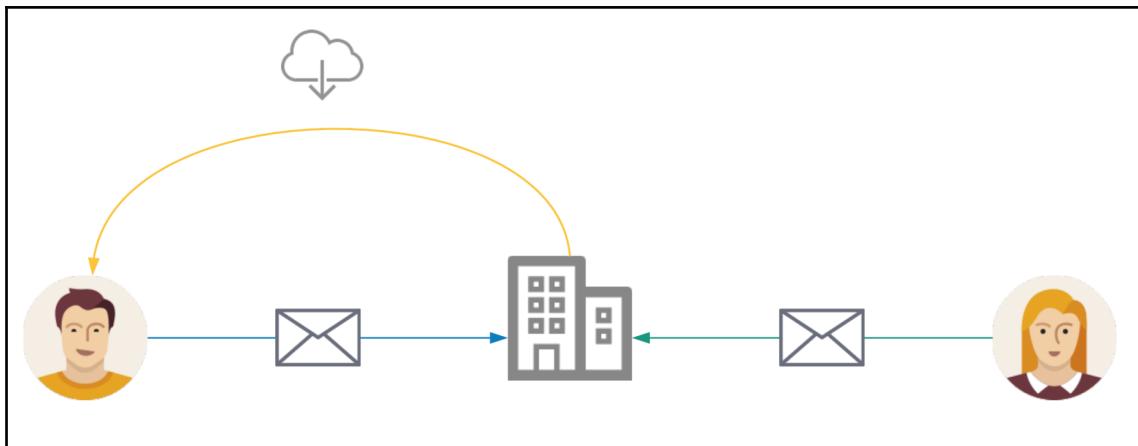
- <https://aws.amazon.com/architecture/>
- <https://aws.amazon.com/kinesis/data-streams/faqs/>
- <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html>
- <https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Expiration.html>
- <https://aws.amazon.com/elasticsearch-service/>

7

Integrating Application Services

Information systems are not reliable by nature, since any component can fail; however, it is possible to improve a system's reliability and availability by distributing them among redundant components, and these components can represent individual service layers, each one with a higher level of cohesion.

Imagine the following scenario: I wish to deliver a letter and receive a confirmation when the letter has arrived. I only know the sender address and the post office that the letter must be picked up from:



This letter represents the content of the message I want to deliver, but in the process, some problems may arise. If I deliver the letter personally, I may run the risk of not finding the recipient in time, so I will need to return again, and if the delivery must be done across many areas of the world, additional considerations such as the local language must be taken, so I can find the destination address. I could also lose the letter in the process. This model doesn't scale and requires a lot of knowledge from both parties to communicate reliably.

Messaging services address all of these problems, and both parties communicate through a service proxy—an intermediary that has the main responsibility of delivering the message and guaranteeing the delivery. This solves the problems associated with timing because the sender could store the letter for as long as necessary and retry the delivery until they are successful. The mail service has all the information relating to postcodes, offices, and the best delivery routes for every city. The service has a presence in multiple locations with local employees who speak the language, so the message delivery can be fulfilled and a receipt signature can be obtained from the recipient.

The following topics will be covered in this chapter:

- SQS as a reliable broker
- Managing N:N communications with SNS
- WebSockets in AWS
- Authenticating your web and mobile apps with **Cognito**
- Web app demo

Technical requirements

In this chapter, we will create an IAM user that will allow us to interact from an administrative point of view with the message queue. To avoid friction with this process, let's use the following script: <http://bit.ly/2P3Rj6y>.

Make sure to download the trust policy file, `trust-policy-for-sqs.json`, in the same path where you executed the previous shell from: <http://bit.ly/2nDrUnM>.

It is assumed that the key pair has been configured for the AWS CLI and that you have administrator access to create IAM users. Execute the following command:

```
chmod +x && ./create-sqs-user-and-role.sh
```

The script will perform the following tasks:

- Create the `sqs-user`
- Create a key pair for the `sqs-user`
- Configure the `[sqs-user]` profile in `~/.aws/credentials`
- Create a service role for EC2
- Create an instance profile for EC2

SQS as a reliable broker

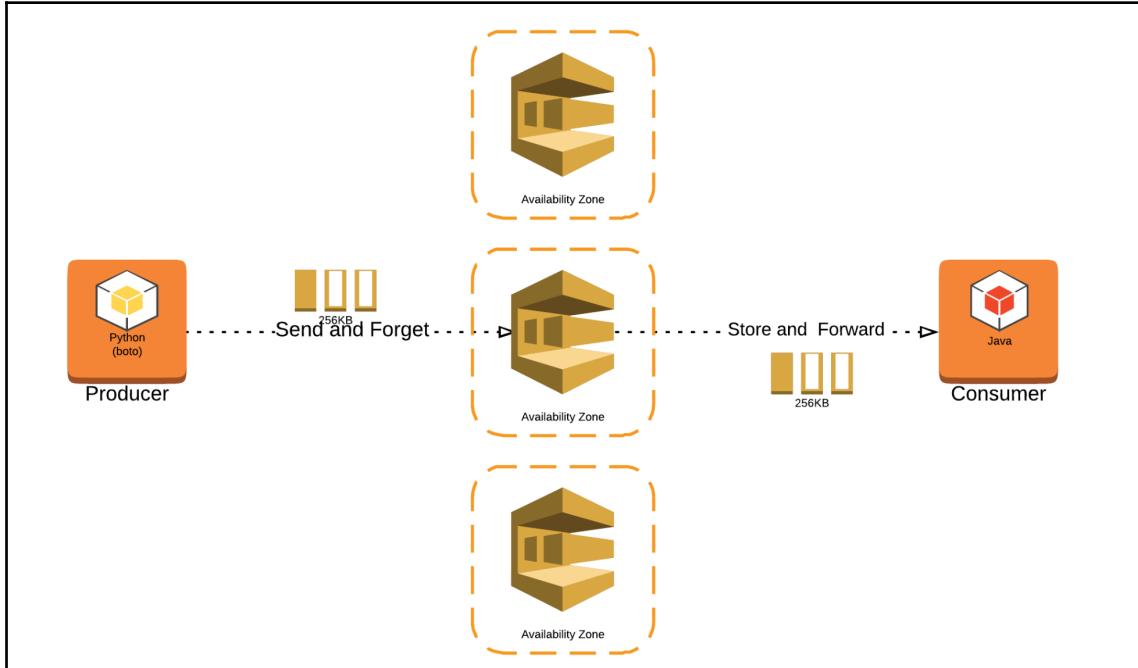
Messaging systems use a broker to handle communications. These systems are designed to provide a communication buffer between distributed components, and they exchange information through messaging channels known as queues, encapsulating data in messages.

Messaging promotes the integration of many heterogeneous technologies through a communication bus that's completely agnostic to operating systems and programming languages. Let's imagine two applications: one written in Python and the other in Java. If both applications use **Simple Queue Service (SQS)**, we can gain interoperability between platforms and work with string objects that can contain any kind of message structure; for example, XML or JSON.

Asynchrony

SQS implements asynchronous communications with a **send-and-forget** approach. This means that the message producer does not need to wait for an answer from the consumers regarding improving the producer throughput and decoupling response times; even if the consumer system is not available at the time, SQS will store the message redundantly so that it can be consumed when the producer is available again. SQS can also be used for batch processing and coordinating message processing in different schedules.

SQS is designed with high scalability, availability, and reliability in mind; this service is vendor neutral. This service availability is achieved by storing every received message in multiple availability zones. SQS is an elastic service that provides unlimited throughput:



This message broker promotes low coupling between the producer and the consumer, and heterogeneous technologies and disparate processing times can be used on both sides. SQS allows multiple producers on the same queue and multiple consumers on the same queue.

Creating a queue

In our example, we don't want to lose any user payments, and we want to record every transaction in the message queue. With the following command, we will create the PaymentsQueue queue:

```
aws sqs create-queue --queue-name PaymentsQueue --profile sqs-user
```

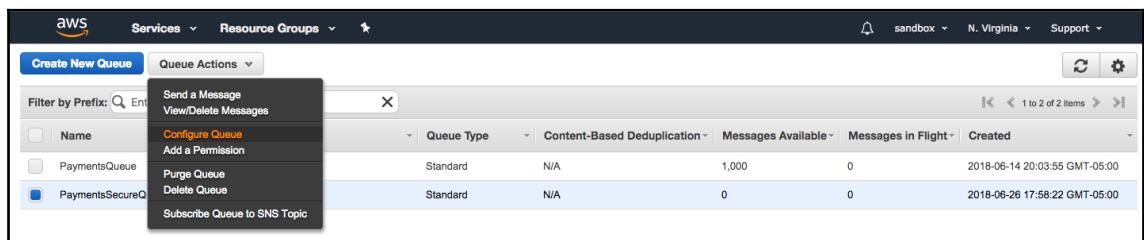
```
Gabanox-MBP:~ gabanox$ aws sqs create-queue --queue-name PaymentsQueue --profile sqs-user --region us-east-1
{
    "QueueUrl": "https://queue.amazonaws.com/ACCOUNT_NUMBER/PaymentsQueue"
}
Gabanox-MBP:~ gabanox$ |
```



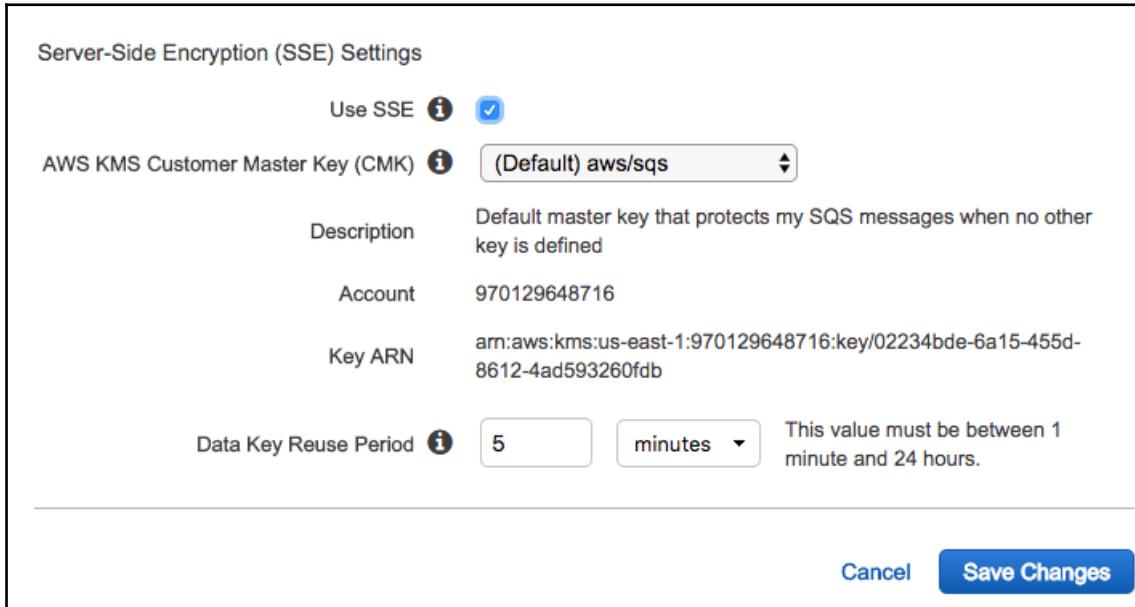
This command uses the bootstrapped keys for the `sqs-user` profile, and this queue URL will be created under the creator account; if this queue is shared, the owner is the one responsible for paying the service. Having a tagging strategy can help with budget showbacks.

Security

SQS allows secure transport communications via secure protocols such as **Transport Level Security (TLS)**, and it also has security at-rest because every message can be encrypted with a unique cryptographic key. IAM can also be used for the management plane and to protect access to resources. To enable server-side encryption (AWS takes care of this process), choose the **Queue Actions** queue, followed by **Configure Queue**, as shown in the following screenshot:



The simplest and most portable way to use encryption at the queue level is to use a **Customer Master Key (CMK)** under the **Server-Side Encryption (SSE) Settings**, as shown in the following screenshot:



With this attribute in place, every message sent from now on will be encrypted transparently when they are stored by the service. When the message gets consumed by the client, its content will be decrypted on demand.

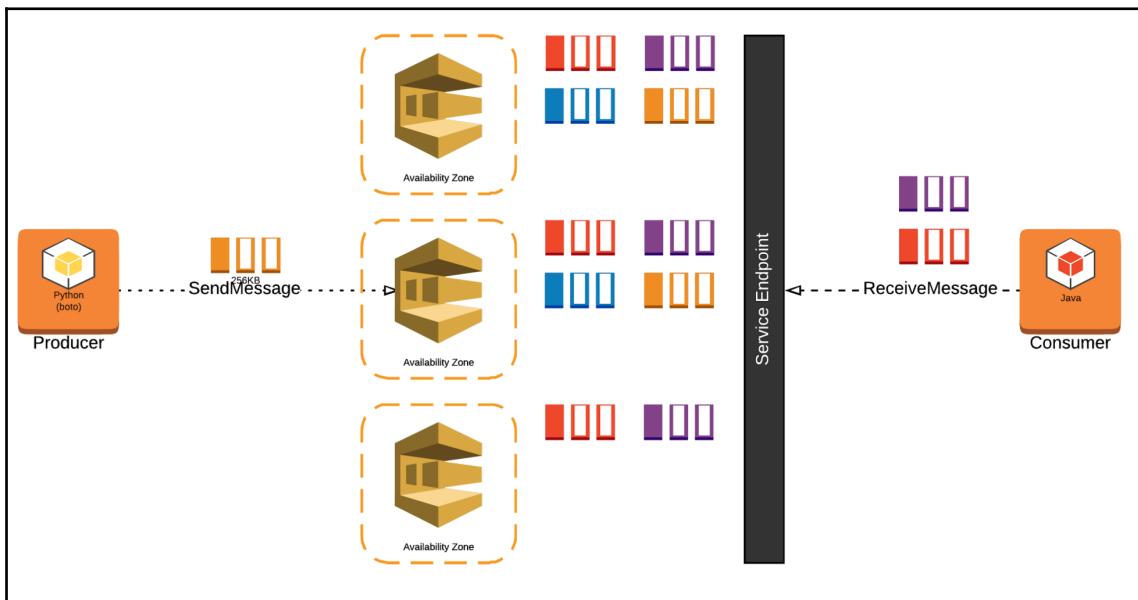
Durability

Messages get stored on SQS for 4 days by default; however, you can change the message retention from 1 minute to 14 days, as shown in the following screenshot:

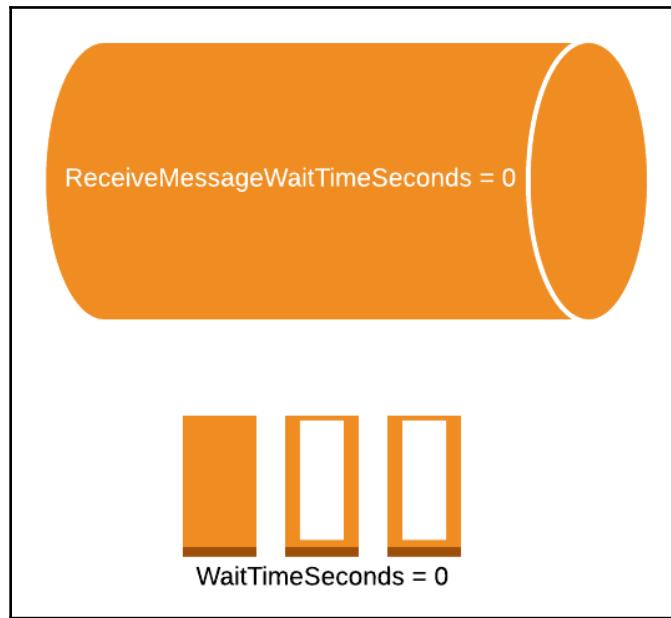
Queue Settings

Default Visibility Timeout	<input type="text" value="30"/> seconds	Value must be between 0 seconds and 12 hours.
Message Retention Period	<input type="text" value="14"/> days	Value must be between 1 minute and 14 days.
Maximum Message Size	<input type="text" value="256"/> KB	Value must be between 1 and 256 KB.
Delivery Delay	<input type="text" value="0"/> seconds	Value must be between 0 seconds and 15 minutes.
Receive Message Wait Time	<input type="text" value="0"/> seconds	Value must be between 0 and 20 seconds.

Every received message is stored redundantly in multiple servers distributed on multiple AZs. SQS will perform a weighted random distribution to query the most probable servers that contain the universe of messages:



SQS provides a quality of service that enables high throughput and high durability (storing the message in three AZs); however, this high throughput comes with a price, as in SQS standard queues, the message order is not guaranteed, so it is up to the application to order the messages with a message sequencer pattern. The way SQS retrieves messages depends on factors such as the message volume (greater than 1,000) and the pooling configuration specified at the message or the queue level:



When a value of *zero* is configured, the pooling mode is configured automatically for short pooling and less probable to retrieve 100% of the sample. This can be configured at the message or the queue level; if done via the SQS queue, all messages will inherit this configuration. A value greater than *zero* (as shown in the previous diagram) enforces long pooling, and this mode will retrieve the full sample of messages, minimizing CPU client cycles and diminishing costs, avoiding unnecessary requests to the service API:

	Attribute	Short	Long
Message	WaitTimeSeconds	0	≤ 20
Queue	ReceiveMessageWaitTimeSeconds	0	≤ 20

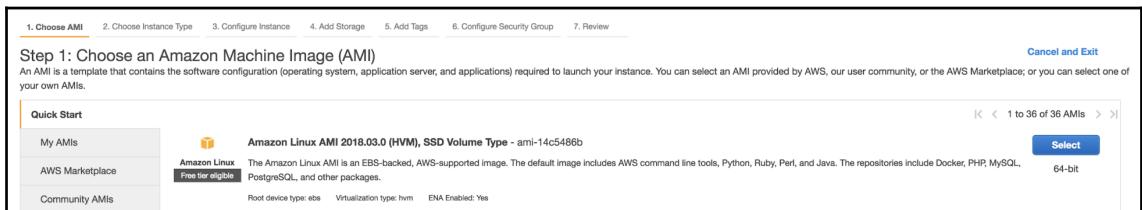
This table summarizes the two attributes available for long polling for messages and queues. They are not mutually exclusive.

Message delivery

In the following example, we will create two instances, an `SQSProducer`, which will be responsible for sending messages via a Python script, and a series of `SQSConsumers` that will be consuming messages from the queue simulating a batch process by decoupling the producer from the consumer.

Our message producer instance will make use of a bootstrapped script that will read the EC2 instance associated tags using `/opt/aws/bin/ec2-describe-tags` and then configure the destination queue, the number of messages, and the region.

1. Create a new EC2 instance and choose the most recent AWS AMI. It is recommended to use a `t2.nano`:



2. In the **Configure instance** step, make sure to specify the instance role **SQSMessagingRole-Profile** (this role was configured by the `create-sqs-user-and-role.sh` script); leave all other options at their default settings:



3. Pass the user data in the **Advanced Details** section using the following bootstrap script: <http://bit.ly/2w0XxLM>:



The screenshot shows the 'Advanced Details' section of a CloudFormation template. It contains a 'User data' field with a bootstrap script. The script is a shell script that performs several tasks:

```

#!/bin/bash

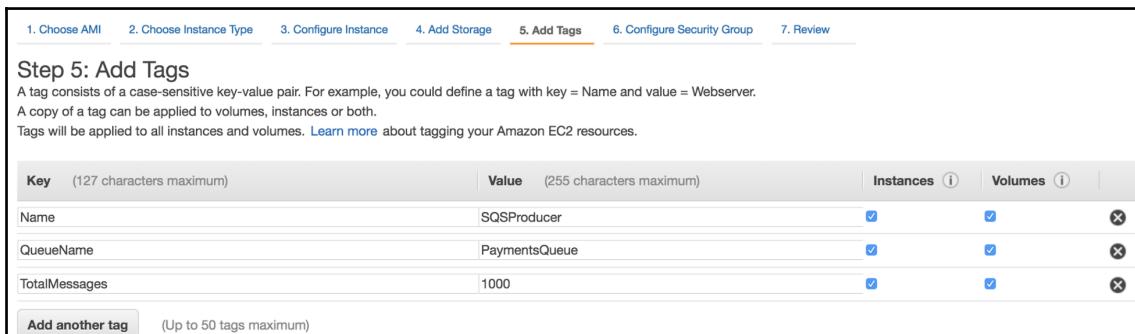
cat <<EOT >> ~/.bash_profile
export EC2_HOME=~opt/aws/apitools/ec2
export PATH=$PATH:$EC2_HOME/bin
export JAVA_HOME=/usr/
EOT
source ~/.bash_profile

idDocument=http://169.254.169.254/latest/dynamic/instance-identity/document
export REGION=$(curl -s $idDocument | grep region | cut -d\` -f4)
export RESOURCE_ID=$(curl -s $idDocument | grep instanceId | cut -d\` -f4)
export QUEUE_NAME=$(opt/aws/bin/ec2-describe-tags --region $REGION --filter "resource-type=instance" --filter "resource-id=$RESOURCE_ID" --filter "key=QueueName" | cut -f5)
export TOTAL_MESSAGES=$(opt/aws/bin/ec2-describe-tags --region $REGION --filter "resource-type=instance" --filter "resource-id=$RESOURCE_ID" --filter "key=TotalMessages" | cut -f5)

curl http://d1d49d4wvn4f.cloudfront.net/chapter08/sqs-simple-producer.py -o sqs-simple-producer.py
pip install boto3
python sqs-simple-producer.py $QUEUE_NAME $REGION $TOTAL_MESSAGES
  
```

Below the code, there are four buttons: 'Cancel', 'Previous', 'Review and Launch' (which is highlighted in blue), and 'Next: Add Storage'.

4. In the **Add Tags** section, use the following information (replacing `TotalMessages` at discretion). Also, note that the queue name must be the same as the one that we created in [Creating a queue](#):



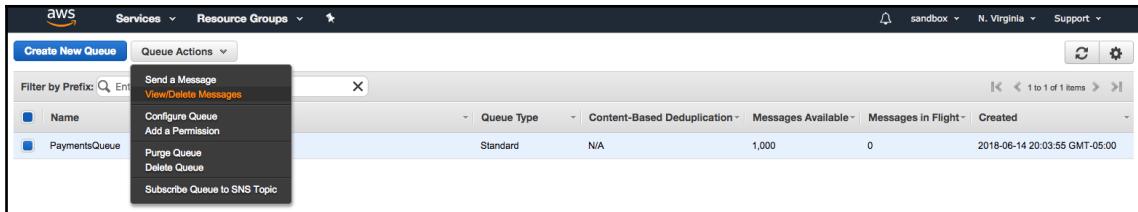
The screenshot shows the 'Step 5: Add Tags' section of the CloudFormation wizard. It lists three tags:

Key	Value	Instances	Volumes
Name	SQSProducer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
QueueName	PaymentsQueue	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
TotalMessages	1000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

At the bottom, there is a button labeled 'Add another tag'.

Continue with the remaining steps, leaving the default values as they are.

This will download the Python script, which will simulate the creation of a number of purchases into the queue as defined with the instance tag `TotalMessages`. You can visualize the messages by navigating to SQS, selecting the queue, and using the **View/Delete Messages** from the **Queue Actions**; also note the number of messages available in the queue in the queue summary:



In the following view, you can poll for a message (1) and open its contents by clicking the **More Details** (2) column:

View up to: 200 messages Poll queue for: 30 seconds

Polling for new messages once every 2 seconds.

1 

2 

	Size	Sent	Receive Count	
CARD CARD_NUMBER=42424242424050 CARD_TYPE=VISA EXPIRATION_DATE=12/2021 C	215 bytes	2018-06-25 14:07:29 GMT-05:00	1	More Details
		2018-06-25 14:07:29 GMT-05:00	1	More Details
		2018-06-25 14:07:29 GMT-05:00	1	More Details
		2018-06-25 14:07:29 GMT-05:00	1	More Details
		2018-06-25 14:07:29 GMT-05:00	1	More Details
		2018-06-25 14:07:29 GMT-05:00	1	More Details
		2018-06-25 14:07:29 GMT-05:00	1	More Details
		2018-06-25 14:07:29 GMT-05:00	1	More Details
		2018-06-25 14:07:29 GMT-05:00	1	More Details
		2018-06-25 14:07:29 GMT-05:00	1	More Details
		2018-06-25 14:07:29 GMT-05:00	1	More Details
		2018-06-25 14:07:30 GMT-05:00	1	More Details
		2018-06-25 14:07:30 GMT-05:00	1	More Details
		2018-06-25 14:07:30 GMT-05:00	1	More Details
		2018-06-25 14:07:30 GMT-05:00	1	More Details
		2018-06-25 14:07:30 GMT-05:00	1	More Details
		2018-06-25 14:07:30 GMT-05:00	1	More Details
		2018-06-25 14:07:31 GMT-05:00	1	More Details

Message Details 

Message Body **Message Attributes**

```
ACTION=PAYOUT  
ITEM_CODE=050  
PAYMENT_TYPE=CREDIT_CARD  
CARD_NUMBER=42424242424050  
CARD_TYPE=VISA  
EXPIRATION_DATE=12/2021  
CVV=123  
COUNTRY=MEX
```

Message ID: ffc2c8d-f86a-4fa4-86e4-a1af3cd48386
Size: 215 bytes
MD5 of Body: d567dd2448616f8fa837f473a5729791
Sender Account ID: AROAJUC3QPV46NDTEXUIC:i-0dc48e52d8c8c8699
Sent: 2018-06-25 14:07:29.298 GMT-05:00
First Received: 2018-06-26 17:11:49.838 GMT-05:00
Receive Count: 1
Message Attribute Count: 0

Close

100%

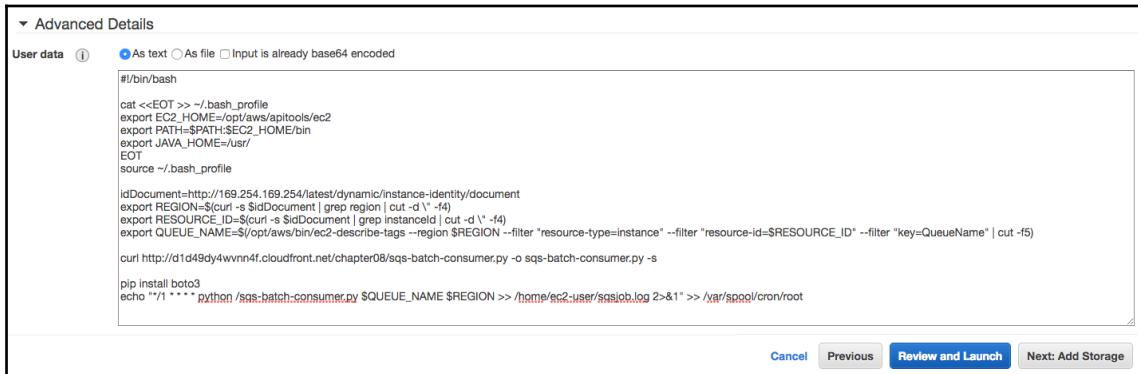
Stopped after polling the queue at 6.1 receives/second for 3.3 seconds. Messages shown above are now available to other consumers.

Close **Delete Messages**

Also, note from the previous screenshot that you can specify **Message Attributes** in the form of key-value pairs; these attributes are not encrypted when the server-side encryption is enabled, and they can be of string, number, or binary type. **Message Attributes** allow for content filtering or the passing of contextual information about the message for applications.

Message reception

To implement the producer, follow the same principle by creating an EC2 instance in the **Advanced Details** section under **User Data** with the following script: <http://bit.ly/2nCfT1X>:



```

#! /bin/bash

cat <<EOT >> ~/.bash_profile
export EC2_HOME=/opt/aws/apitools/ec2
export PATH=$PATH:$EC2_HOME/bin
export JAVA_HOME=/usr/
EOT
source ~/.bash_profile

idDocument=$(curl -s http://169.254.169.254/latest/dynamic/instance-identity/document)
export REGION=$(curl -s $idDocument | grep region | cut -d ' ' -f4)
export RESOURCE_ID=$(curl -s $idDocument | grep instanceId | cut -d ' ' -f4)
export QUEUE_NAME=$(/opt/aws/bin/ec2-describe-tags --region $REGION --filter "resource-type=instance" --filter "resource-id=$RESOURCE_ID" --filter "key=QueueName" | cut -f5)

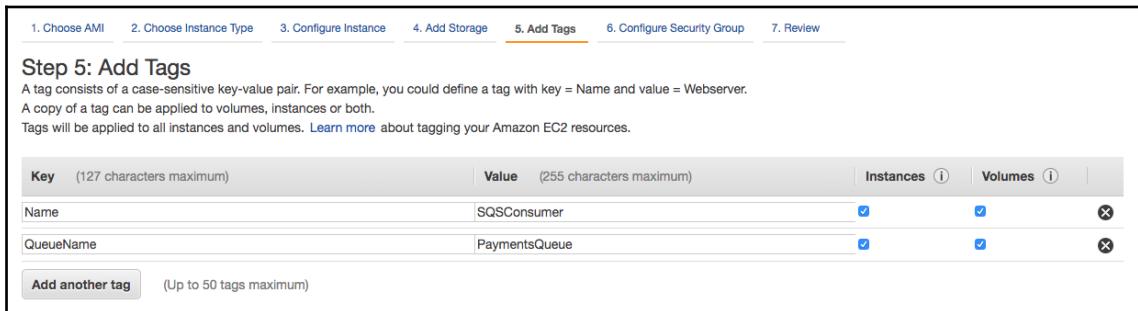
curl http://d1d49dy4wnnn4.cloudfront.net/chapter08/sqs-batch-consumer.py -o sqs-batch-consumer.py

pip install boto3
echo "*/1 * * * * python /sqs-batch-consumer.py SQUEUE_NAME $REGION >> /home/ec2-user/sqsjob.log 2>&1" >> /var/spool/cron/root

```

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Storage](#)

This script will download the `sqs-batch-consumer` script and configure a cron job that runs every minute to poll for 10 messages from each receive messages operation. The script will use the following tags:



Key	Value	Instances	Volumes
Name	SQSConsumer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
QueueName	PaymentsQueue	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

[Add another tag](#) (Up to 50 tags maximum)

If you SSH to the instance, you can visualize the cron output by issuing the following command:

```
tail -f /home/ec2-user/sqsjob.log
```

You are encouraged to create more than one worker to see that the queue gets drained more quickly, and you can also run these scripts locally from your computer with the proper access keys configured:

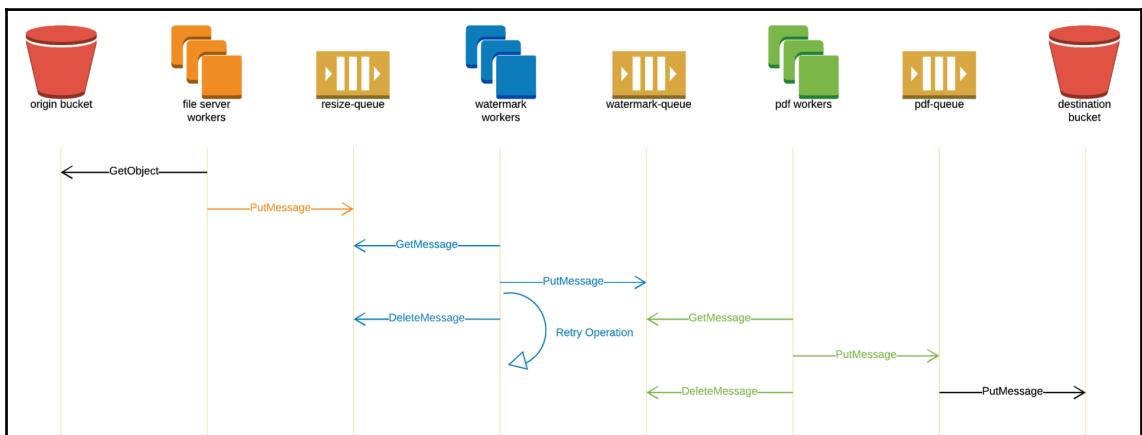
SQSproducer	http://bit.ly/2w0XxLM
SQSConsumer	http://bit.ly/2MNCVh2

Download the preceding files if you want to test the end-to-end solution locally.

Messaging patterns

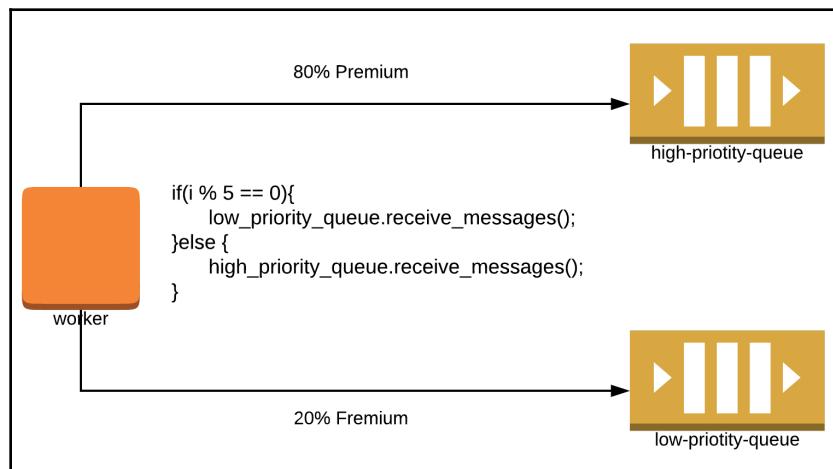
Queues can be thought as buffers to offload long processing tasks and critical operations. In the following business scenario, you have two kinds of customers: the first one is a freemium user, and the second one is a premium customer. Both users need to be able to convert images to PDFs, and this operation also resizes every image to fit 500 x 500 pixels; if the customer is on a freemium model, the image must be watermarked before it's rendered on to a **Portable Document Format (PDF)**.

This multiple processing pipeline can be designed with SQS and multiple worker layers, and also we can run into the problem of having a high demand on certain times of the day, throughout the week. To be able to process more messages, we will make use of AutoScaling. The following cloud pattern is known as a **queuing chain pattern**:



One thing to know is that the application code is responsible for the deletion of the message, and to implement this pattern correctly, every operation must be **idempotent**, which means that no matter how many times the operation is performed, the result must be the same. If an instance from the watermark fails, AutoScaling will create one automatically and the operation will be retried. Also, with the use of CloudWatch you can monitor the `ApproximateNumberOfMessages` metric, the average processing latency, and the fleets size to dynamically scale out the number of workers needed for every step; this pattern is known as the **job observer**.

When a new requirement arises, we need to establish high priority for premium customers. To achieve this, we can have two queues and poll first from the high priority queue and then from the low priority queue. Even your application logic could use a modulo arithmetic operation to ponderate poll operations for both queues:



This is a pseudocode implementation in which every fifth message is processed from the low priority queue; more sophisticated schemes can be achieved, but it is up to the application to define this logic. This is the **priority queue pattern**.

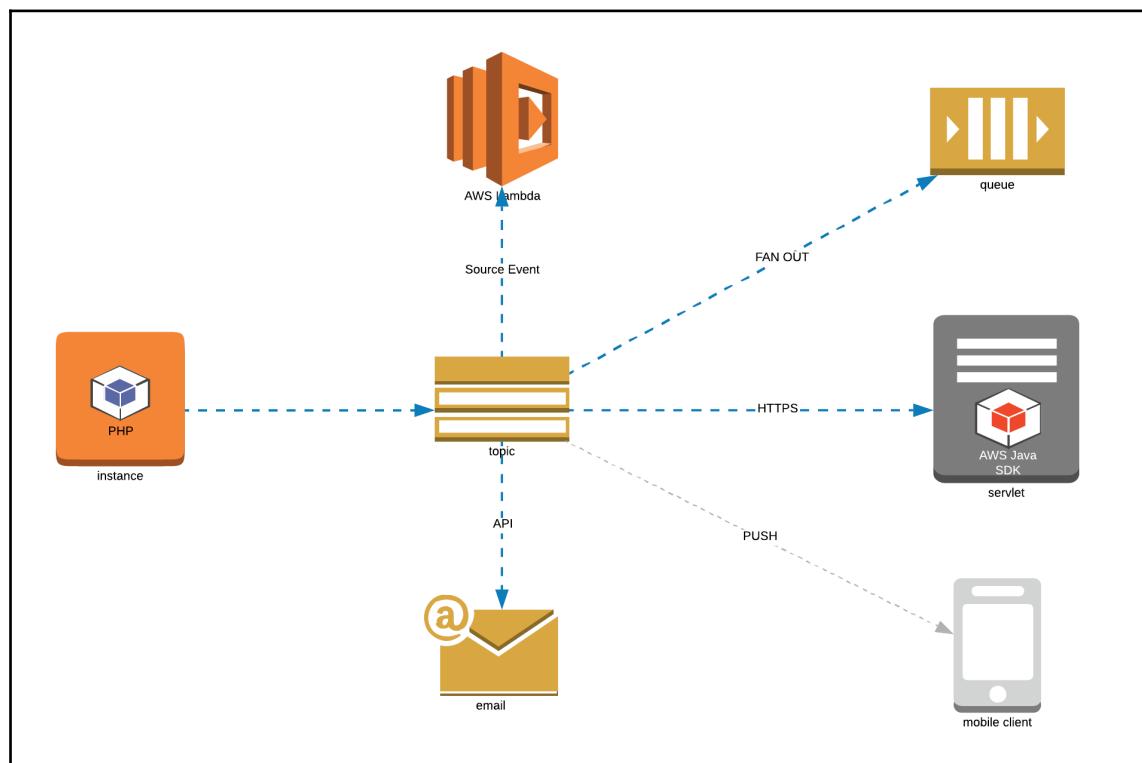
Managing 1:N communications with SNS

The **Simple Notification Service (SNS)** is a messaging service designed to work with publisher/subscriber semantics by decoupling distributed system components using an event model based on asynchronous push messaging. This way, every topic subscriber receives a copy of the message that is broadcast.

The message is a **first-class citizen**, with the flexibility to re-routed between components and applications with a high level of interoperability, since messages are composed of text with a maximum limit of 256 KB. The message can contain payload and attributes, and these attributes can be used to filter messages and provide contextual information about the message. The message payload can be encrypted to provide additional security aspects for communication.

Subscriber

The communication model of SNS is one-to-many *1:N*; each time a message gets published to the topic, the service is responsible for the delivery of the message to every one of the subscribers available:



In the preceding diagram, we have subscribed a Lambda function, an SQS queue, and an external server via an HTTPS endpoint, and a mobile application via vendor push messaging and email.

Messages will be delivered to all the channel's subscribers, and subscribers who arrive late will lose their sent messages. This kind of messaging is excellent for orchestrating communications between microservices, and every listening party chooses to process the message or drop it.

Once a message has been sent, it cannot be sent again, and this is because SQS is not persistent, as the service only acts as a message relay with the ability to scale up to trillions of messages per second.

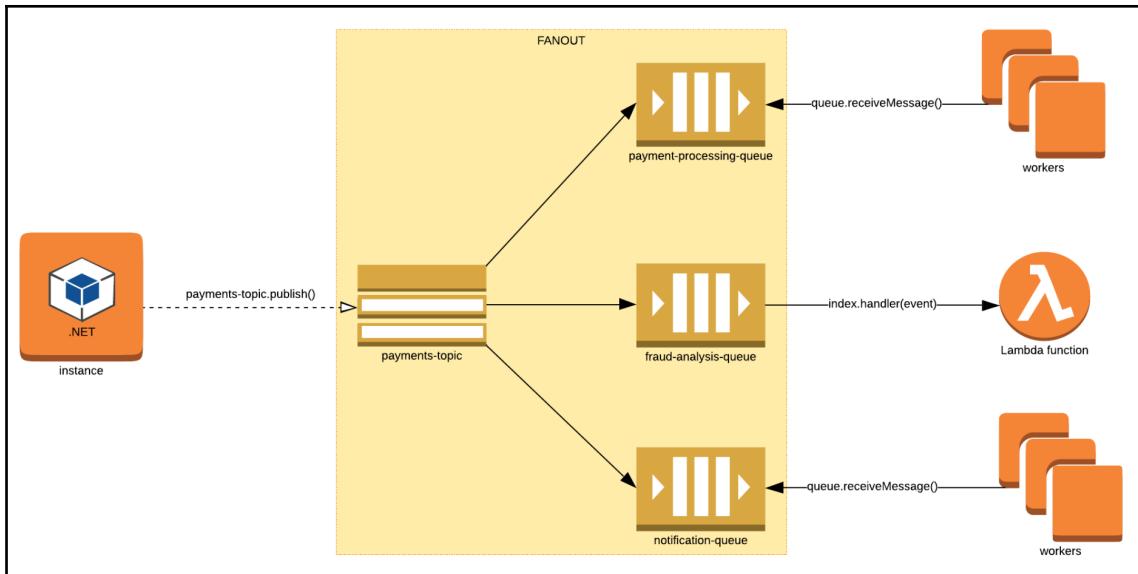
The available subscribers are as follows:

Subscriber	Use case
Amazon SQS	Fanout
Application	Push messaging for mobile devices
Email	Human-intended communication
Email/JSON	Application processing via email
HTTP/HTTPS	Webhooks
Lambda function	Serverless processing and real-time
SMS	Mobile phones without push capabilities, 2FA authentication

This summary can help you decide the best use case for every kind of subscription available for SNS.

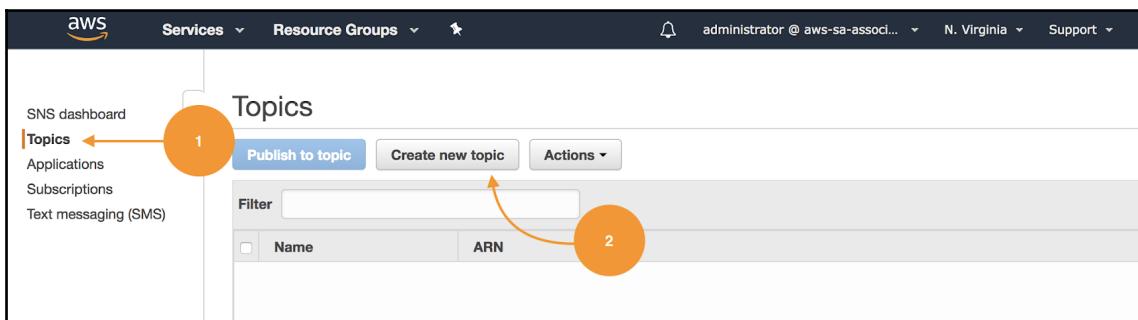
Fanout

Fanout (the number of inputs that can be connected to a specified output). This way, a single published message can be copied on to multiple destinations, such as SQS queues; these kinds of designs promote decoupling and combine two messaging patterns, one-to-many communications, and durable reliable message delivery, as shown in the following diagram:



In this case, the **payments-topic** publishes the payment information of every SQS queue subscribed.

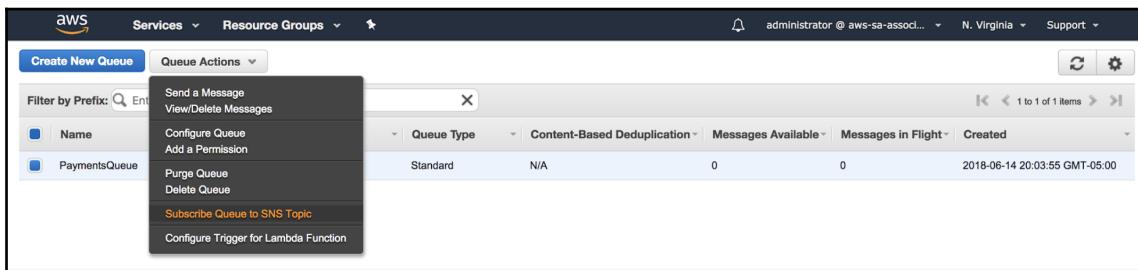
1. To achieve this, we will create an SNS topic called **payments-topic**. Navigate to the SNS service, choose **Topics** | **Create new topic**:



2. We need to specify a topic name, and to be able to subscribe, the ARN will be needed:

The screenshot shows the 'Create new topic' dialog box. It has a title bar 'Create new topic'. Below it, a note says 'A topic name will be used to create a permanent unique identifier called an Amazon Resource Name (ARN)'. There are two input fields: 'Topic name' containing 'payments-topic' and 'Display name' containing 'Payments'. At the bottom right are 'Cancel' and 'Create topic' buttons.

3. Now update our PaymentsQueue by subscribing it to the SNS topic, as follows:



4. In the subscription window, we have the option to choose from which topic our queue will be listening to. Choose the **payments-topic**, as shown in the following screenshot:

Subscribe to a Topic

Select an SNS Topic from the *Choose a Topic* drop-down or enter a topic's ARN in the *Topic ARN* text box and then press *Subscribe* to allow your queue(s) to receive SNS notifications from the topic and to subscribe your queue(s) to the topic.

Topic Region

Choose a Topic

Topic ARN

5. Now it's time to send a message from the topic, so let's go back to the SNS console, choose the **payments-topic**, and **Publish to topic**:

Publish a message

Amazon SNS enables you to publish notifications to all subscriptions associated with a topic as well as to an individual endpoint associated with a platform application.

Topic ARN

Subject

Message format Raw JSON

Message

```
ACTION=PAYOUT  
ITEM_CODE=00  
PAYMENT_TYPE=CREDIT_CARD  
CARD_NUMBER=424242424242400  
CARD_TYPE=VISA  
EXPIRATION_DATE=12/2021  
CVV=123  
COUNTRY=MEX
```

Time to live (TTL)

Message Attributes Attribute type

The fanout pattern is a sophisticated way to achieve low latency reliable communications with the lowest effort. SNS will keep trying until it succeeds to deliver the messages to the SQS queue and HTTPS messages can be retried with retry policies and back-off functions until the subscriber receives the message.

Functionality such as webhooks can be achieved with SNS and mobile push capabilities to create real-time applications, chats, marketing campaigns, and IoT.

Authenticating your web and mobile apps with Cognito

Authentication capabilities in applications should be simple and safe by default. Designing customized authentication systems takes a long time and is a specialized endeavor. The users of these applications want to explore an app's functionality without spending too much time creating user accounts. It is our main goal to minimize the friction associated with this processes as much as possible.

Newer users feel comfortable using their existing credentials and use them in multiple authentication facades. This practice is called **federation** (the word *federation* originates from the latin *foederatio*, which means *union*). These credentials that are external to our systems can be used to produce security assertions in the form of tokens. The representations of the original credentials can be used to assume an identity or to gain temporary access to a private resource.

Cognito user pools

Cognito user pools is a serverless managed database that you can use to authenticate your web and mobile apps. It is also a user management system that can scale and also integrate existing identity providers. User pools free the developer from the implementation of common user flows such as the following:

- Signing up and signing in
- Updating the user profile data via standard and custom attributes
- Forgot password by providing challenges
- Token-based authentication for **JSON Web Tokens (JWTs)**
- Email and SMS two-factor verification

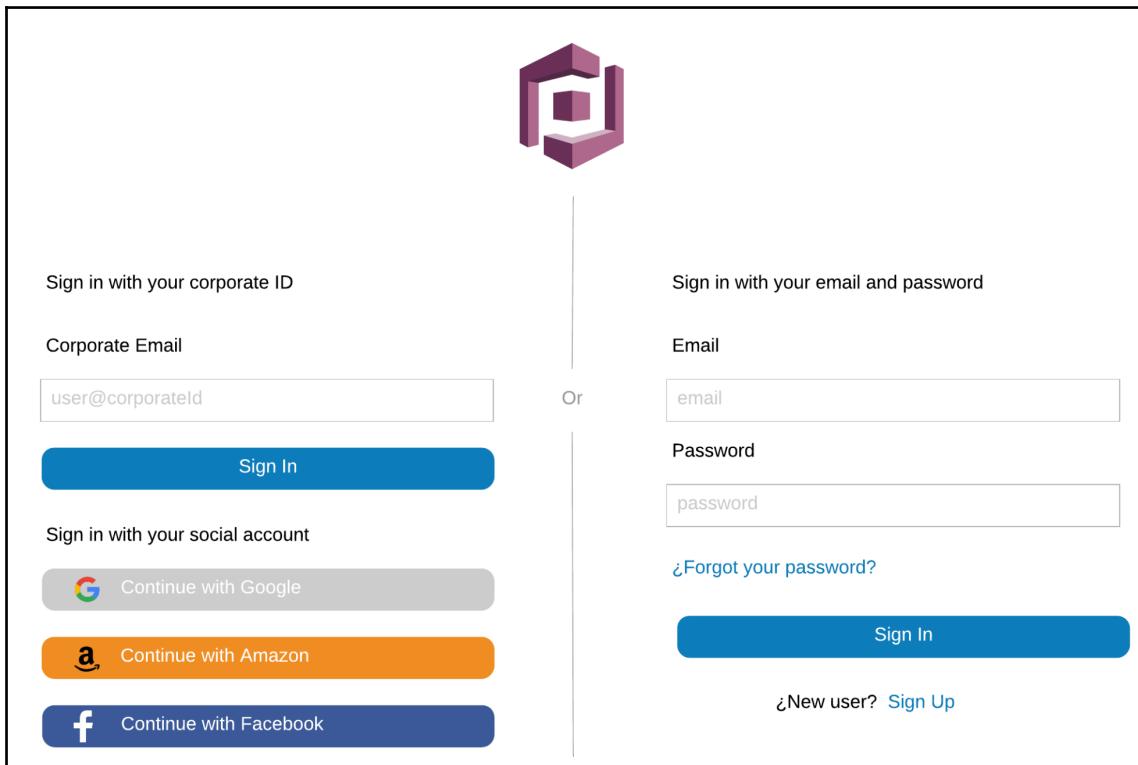
These flows can be customized via AWS Lambda to integrate your existing authentication and authorization schemes. The Cognito API gives you the ability to integrate user administration tasks into your own back-office applications:

Username	Enabled	Status	Updated	Created
gabriel	Enabled	FORCE_CHANGE_PASSWORD	Aug 7, 2018 10:27:36 PM	Aug 7, 2018 10:27:36 PM
john	Enabled	FORCE_CHANGE_PASSWORD	Aug 7, 2018 10:28:04 PM	Aug 7, 2018 10:28:04 PM
johndoe	Enabled	FORCE_CHANGE_PASSWORD	Aug 7, 2018 10:28:36 PM	Aug 7, 2018 10:28:36 PM

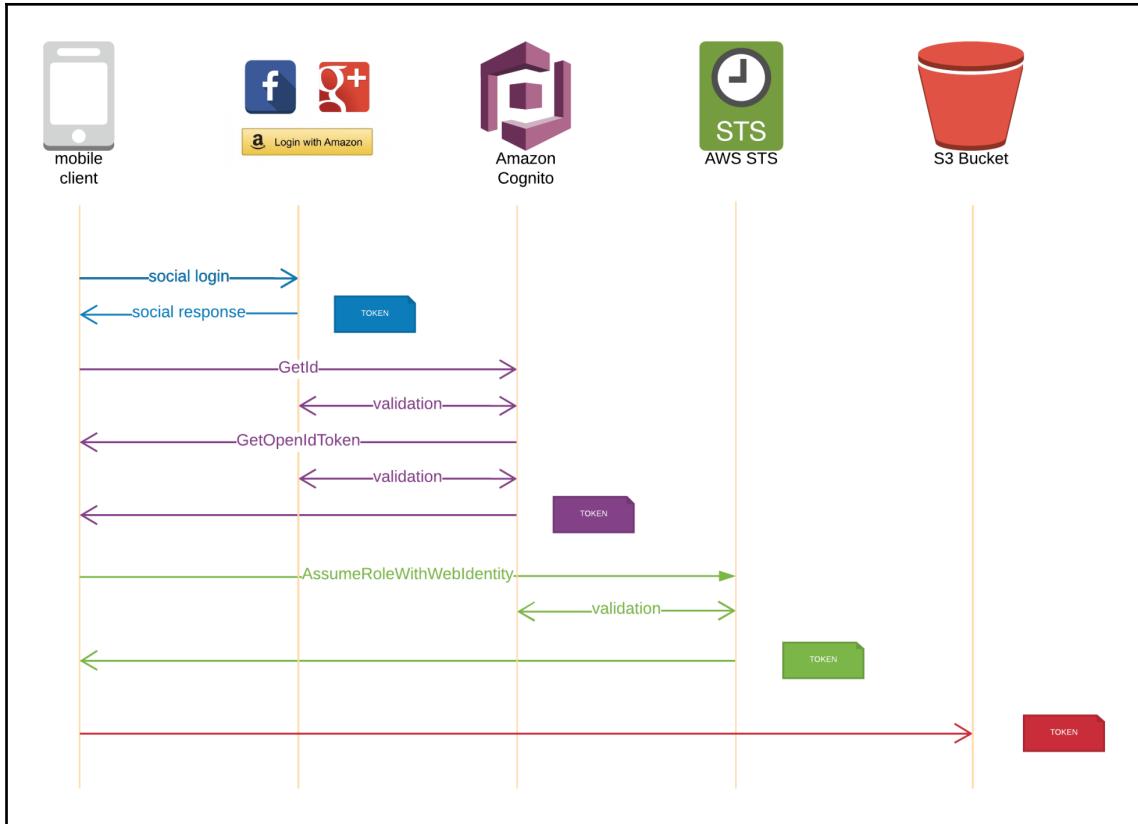
The preceding screenshot shows how you can manage user pool identities from the AWS console. This gives you the possibility to use the Cognito admin API to manage these identities in your own back-office systems.

Federated identities

You can use Cognito federated identities for corporate or social credentials to exchange them for temporary tokens that are used to access AWS service APIs. Developers can build custom user interfaces and work with authenticated and unauthenticated roles, so users can use your app once they have authenticated parts of your app anonymously:



Cognito user pools have support for OAuth 2.0, which is an open standard protocol for authentication, and the user pool can interact with multiple **Identity Providers (IdP)** to exchange credentials and gain temporary access tokens to invoke services such as S3:



In this example, the mobile client logs in with federated credentials, and this assertion is used to invoke the Cognito authentication flow, and the mobile app SDK will assume the web identity with STS and will return temporary user credentials to invoke S3 APIs.

Cognito can also synchronize data between devices providing the same experience when the user switches from a web experience to mobile, and with AppSync, you can have offline capabilities and events listening in the background. Let's assume that the customer has upgraded the subscription plan via web and they must immediately gain access to premium features in the mobile app. With the Cognito and AppSync SDKs, this is possible in a smooth way.

API Gateway integration

The **API Gateway** service is a service facade for RESTful applications, the REST architectural approach that integrates via web services. This lightweight implementation is the preferred protocol for communication on the internet because it uses the minimum common denominator of the HTTP protocol providing interoperability.

The API exposes a series of resources in a stateless manner, and applications interact with verbs, indicating the server operation to be done. API Gateway works with the OpenAPI specification; in our example, we will create a demo PetStore API to demonstrate the service functionality.

In the AWS console, under **Networking and content delivery**, find the **API Gateway service**, and then create a new API with the **Example API** option:

Create new API

In Amazon API Gateway, an API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

New API Clone from existing API Import from Swagger Example API

Example API

Learn about the service by importing an example API and turning on hints throughout the console.

```
1 {
2     "swagger": "2.0",
3     "info": {
4         "description": "Your first API with Amazon API Gateway. This is a sample API that integrates via HTTP with our demo Pet Store endpoints",
5         "title": "PetStore"
6     },
7     "schemes": [
8         "https"
9     ],
10    "paths": {
11        "/": {
12            "get": {
13                "tags": [
14                    "pets"
15                ],
16                "description": "PetStore HTML web page containing API usage information",
17                "consumes": [
18                    "application/json"
19                ]
1. 
```

Settings

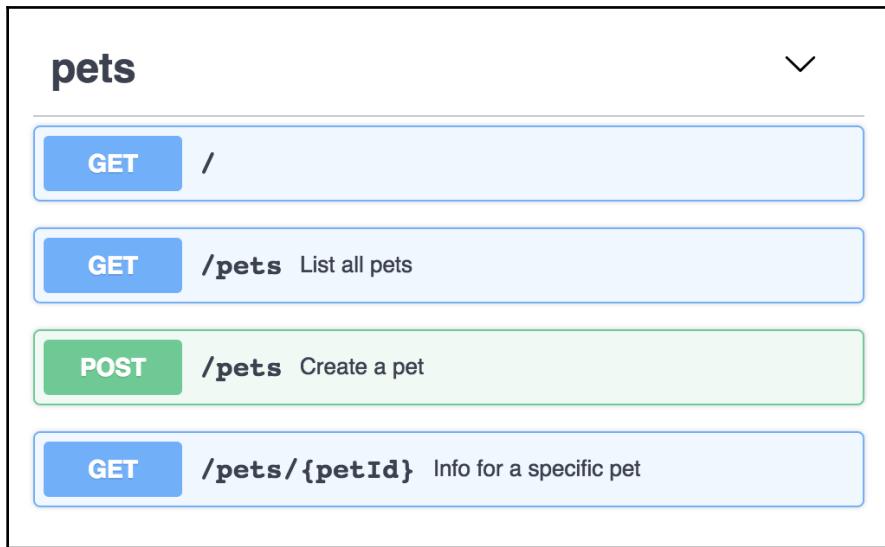
Endpoint Type: Regional ⓘ **1**

* Required Fail on warnings Ignore warnings **Import** **2**

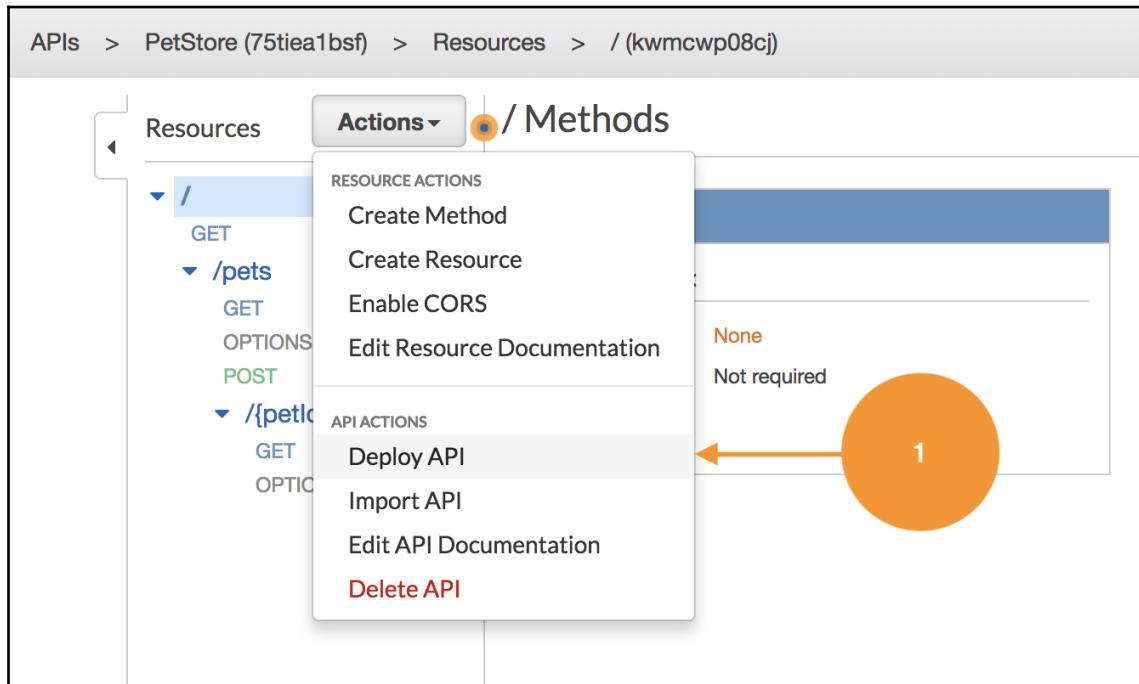
We are importing the API definition from a Swagger 2.0 JSON file, and this bottom-up approach will create the necessary artifacts to deploy the API. The endpoint can be configured with the following scopes:

Scope	Use case
Private	Only accessible from the VPC, useful to microservices architectures using SOA.
Regional	Low latency endpoint available only to regional resources, helps with compliance.
Edge optimized	Deployed via CloudFront using edge locations providing the best latency for globally dispersed users.

1. Choose **Regional | Import** and the following resources will be created:



2. The next step is to deploy the API to a stage. Under API resources choose **Deploy API**:



3. You will be required to create a stage, in this case, create a new one called prod:

Deploy API

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage [New Stage] ▾
Stage name* prod
Stage description Production Environment
Deployment description First deployment

Cancel Deploy

This will take a short amount of time, and you can have as many stages depending of your software pipeline.

4. Copy the API URL; you can use the terminal with `curl` on Linux systems or the Postman client to make requests:

Stages Create prod Stage Editor Delete Stage

dev prod uat

Invoke URL: <https://75t1ea1bsf.execute-api.us-east-1.amazonaws.com/prod>

Settings Logs Stage Variables SDK Generation Export Deployment History Documentation History Canary

5. The API includes a landing page where you can check verbose information about the web service; this text can be arbitrary:

The screenshot shows a browser window with the URL <https://75tiealbsf.execute-api.us-east-1.amazonaws.com/prod>. The page title is "Welcome to your Pet Store API". The content includes a message about successful deployment, information about the /pets resource, and instructions for testing with Postman. It also shows a sample JSON body for creating a new pet.

```
{  
    "type" : "cat",  
    "price" : 123.11  
}
```

The preceding operation represents GET /. Now try invoking the service methods as follows:

1. List the pets collection (GET /pets):

The terminal output shows the results of a curl command to list all pets from the API. The response is a JSON array containing three objects, each representing a pet with fields id, type, and price. An orange arrow points from the text "List all pets" to the JSON data.

```
Gabanox-MBP:~ gabanox$ curl -X GET https://75tiealbsf.execute-api.us-east-1.amazonaws.com/prod/pets  
[  
  {  
    "id": 1,  
    "type": "dog",  
    "price": 249.99  
  },  
  {  
    "id": 2,  
    "type": "cat",  
    "price": 124.99  
  },  
  {  
    "id": 3,  
    "type": "fish",  
    "price": 0.99  
  }]  
Gabanox-MBP:~ gabanox$
```

List all pets

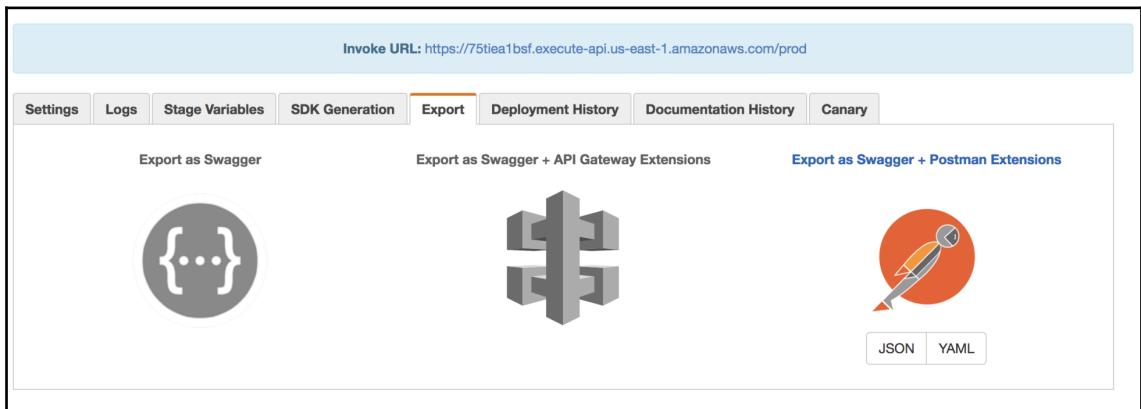
2. Create a pet object (POST /pets):

```
Gabanox-MBP:~ gabanox$ curl -H "Accept: application/json" -H "Content-type: application/json" -X POST \
> -d '{"type": "cat", "price":123.11}' \
> https://75tiea1bsf.execute-api.us-east-1.amazonaws.com/prod/pets/
{
    "pet": {
        "type": "cat",
        "price": 123.11
    },
    "message": "success"
}Gabanox-MBP:~ gabanox$
```

3. Get a pet object (GET /pets/1):

```
]Gabanox-MBP:~ gabanox$ curl -X GET https://75tiea1bsf.execute-api.us-east-1.amazonaws.com/prod/pets/1
{
    "id": 1,
    "type": "dog",
    "price": 249.99
}Gabanox-MBP:~ gabanox$
```

A great advantage of using open specifications is that the API is portable, so you have the ability to export your API definition with Swagger and additional definitions:



Under **SDK Generation**, you can download specific programming language clients, for example, Java or iOS. This will give developers a more programmatic experience for calling service methods.

This Swagger definition makes use of JSON schema (<http://json-schema.org/>), an extensive vocabulary for annotating and validating entities. Let's analyze the CreatePet method:



Navigate to **Models**, and you will discover multiple entities definitions used for the API to mock and validate objects.

This mocking feature is awesome because it promotes agility for development teams working on frontend and backend independently. This feature lets you decouple by designing a contract that both parties must implement (the API) and faster integration cycles:

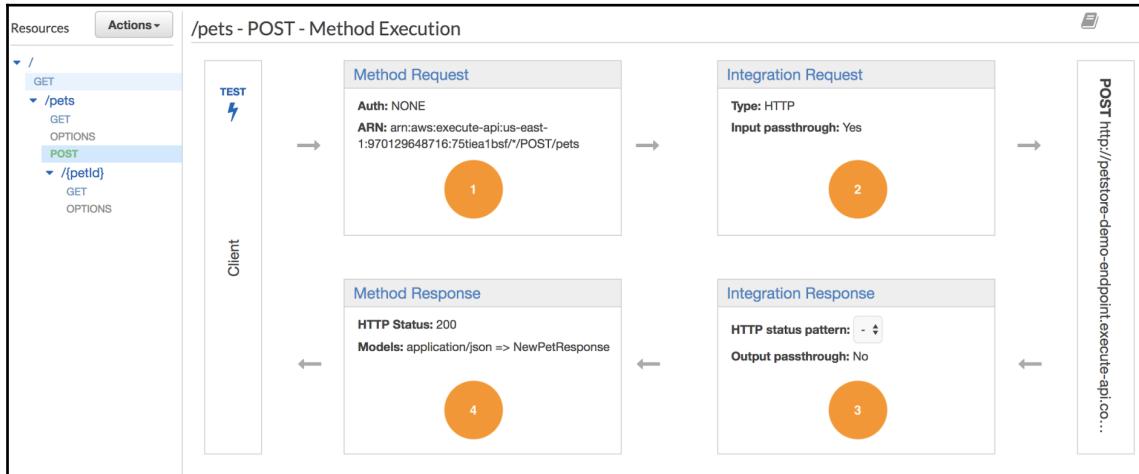
The screenshot shows the AWS API Gateway console with the following details:

- APIs** sidebar: PetStore, Resources, Stages, Authorizers, Gateway Responses, Models (selected).
- Models** sub-menu: Empty, NewPet (highlighted), NewPetResponse, Pet, Pets, PetType.
- Create** button.
- Update Model** section:
 - Model name: NewPet
 - Content type: application/json
 - Model description: (empty)
- Model schema***:

```
1  {
2   "type" : "object",
3   "properties" : {
4     "type" : {
5       "$ref": "https://apigateway.amazonaws.com/restapis/75tie1bsf/models/PetType"
6     },
7     "price" : {
8       "type" : "number"
9     }
10  }
11 }
```

Request flow

For this exercise, take a closer look at the CreatePet method:



1. The request is validated in **Method Request**, and in this case, we can restrict this to accept only well-formed entities according to the JSON schema definition. This works for URL query parameters, headers, and the request body. Here, it is also possible to request authorization via IAM and to use an API key to sign requests:

A screenshot of the Request Body configuration for the CreatePet method. The configuration table has two columns: Content type and Model name.

Content type	Model name
application/json	NewPet

2. The integration request can be made with one of the following:

- **Lambda function:** This is one of the most used integrations in AWS because it allows you to design microservices architectures by using small services provided by AWS Lambda.
- **HTTP:** Any kind of HTTP endpoint that can be reached by API Gateway can be invoked as part of the integration; this promotes hybrid environments or third-party service providers, for example, a payment external API.
- **Mock:** Fast integration cycles can be achieved by mocking resources, and you can hardcode the responses associated with different HTTP status codes, for example, a 200 successful request and a 500 error.
- **AWS service:** Fully fledged backends by mapping the HTTP verb to an action of another service, for example, receiving a `PUT` operation to a resource such as `/payments` and producing a `PutMessage` action in an SQS queue or a `POST /user` for a `PutItem` action in a DynamoDB table. This allows to abstract the code required to take a managed service action.
- **VPC Link:** This is a private integration that will only work inside the VPC. The requests will be encapsulated and routed to a network load balancer.
 1. You can choose an integration response that responds with custom predefined objects or custom responses based on Apache Velocity templates; in this case, the response is returning a custom header to enable **Cross-Origin Resource Sharing (CORS)**:

[Method Execution](#) /pets - POST - Integration Response

First, declare response types using [Method Response](#). Then, map the possible responses from the backend to this method's response types.

HTTP status regex	Method response status	Output model	Default mapping
▼ -	200	Yes	X

Map the output from your HTTP endpoint to the headers and output model of the 200 method response.

HTTP status regex: default [i](#)

Content handling: Passthrough [i](#)

[Cancel](#) [Save](#)

▼ Header Mappings

Response header	Mapping value i
Access-Control-Allow-Origin	i edit

▶ Mapping Templates

2. The method response is the last step on the request lifecycle, where we can provide an HTTP status independent from the integration type; in this case, the response uses a mocked NewPetResponse model for the HTTP 200 status:

[Method Execution](#) /pets - POST - Method Response

Provide information about this method's response types, their headers and content types.

HTTP Status
▼ 200 X

Response Headers for 200

Name
Access-Control-Allow-Origin edit

Response Body for 200 [i](#)

Content type	Models
application/json	NewPetResponse edit

[+ Add Header](#) [+ Add Response Model](#)

We can add behavior for 400 and 500 HTTP codes with different models or JSON structures.

API Gateway is an excellent choice to expose functionality in a managed way, releasing from the controller logic, middleware, and authentication proxies. If custom authentication to the API is needed, such as JWT or a simple data source, you can use **Authorizers**:

Authorizers

Authorizers enable you to control access to your APIs using Amazon Cognito User Pools or a Lambda function.

+ Create New Authorizer

Create Authorizer

Name *

Type * i

Lambda Cognito

Lambda Function * i

us-east-1

Lambda Invoke Role i

Lambda Event Payload * i

Token Request

Token Source* i **Token Validation** i

Authorization Caching i

Enabled **TTL (seconds)**

300

Create **Cancel**

This authorizer will intercept every call to perform custom validation for each request, and you can also use Cognito identity pools and Cognito federated identities.

WebSockets in AWS

Real-time applications are a real challenge, and this is because additional infrastructure and protocols must be met. WebSockets come to the rescue by providing bi-directional communication between the client and the server. In the old days, when the Classic Load Balancer was the only option the way to balance web sockets communication was to work in TCP mode, delegating the WebSocket duplex function to a service proxy such as NGINX to upgrade the connection or using some framework such as `socket.io`.

AWS IoT

AWS IoT is a service designed to connect objects to the IoT via the SDK; you can use the WebSockets via **MQTT**, a lightweight protocol designed for unreliable networks, and this way, you can push messages back to the client. The device gateway component is a messaging broker that can scale up to trillion messages per second and has a RESTful API that can be integrated with multiple clients such as mobile and enterprise applications.

AWS AppSync

AWS AppSync also supports the WebSockets protocol, and it can push messages published to subscribers listening to specific topics. This is the preferred way to interact because it has support for many data sources and makes it easy to integrate with web and mobile clients. This is the recommended practice in most cases.

Web app demo

To perform this demo, it is recommended to follow the AWS IoT chat application; you can find further instructions on the GitHub repo: <https://github.com/aws-samples/aws-iot-chat-example>.

This example walks you through a web app real-time application using the following services:

- AWS IoT, to exchange messages between web clients
- Transmit messages using the MQTT over the WebSocket protocol to reduce network bandwidth requirements
- Authenticate clients with Amazon Cognito and attach IoT policies to allow clients to do the following:
 - Connect to the AWS IoT Device Gateway
 - Publish messages to specific topics
 - Subscribe and receive messages from specific topics
- Authentication with Amazon Cognito user pools and Cognito federated identities
- Serverless computing with AWS Lambda
- API access control provided by Amazon API Gateway
- Room persistence via Amazon DynamoDB
- Static site hosting on Amazon S3
- Amazon CloudFront as a proxy and CDN

The application uses the Serverless Framework (<https://serverless.com/framework/>), CloudFormation, and Node.js as the local runtime environment. You can try the end functionality in the demo website (<https://d2ab9cgiumppbb1.cloudfront.net/login>)

Summary

In this chapter, we covered the uses of messaging systems, the publisher and subscriber perspectives, and how to integrate them into applications. We also discussed multiple messaging patterns such as the queuing chain pattern, job observer, and priority queue.

We also used the simple notification service to produce fan-out to integrate into SQS and manage one-to-many communications.

In the third section, we introduced Amazon Cognito user pools and federated identities while explaining the use cases for each one. API Gateway is used to deploy a RESTful API with mock implementations.

Lastly, we covered WebSockets and which technologies are available in AWS to work with WS and best practices, and, finally, we looked at a web app showing how all the integration services discussed in this chapter can be used in a web application.

Further reading

- <https://www.enterpriseintegrationpatterns.com/>
- <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-long-polling.html>
- <https://aws.amazon.com/es/sqs/faqs/>
- http://en.clouddesignpattern.org/index.php/Main_Page

8

Disaster Recovery Strategies

A key motivation for adopting public clouds is the capabilities that they provide, like geographic redundancy and compute bursting. Compute bursting is a functionality that on-premises applications can use when they run out of resources or capacity; it enables them to offload to the cloud when demand increases or a failure arises.

AWS provides several building blocks for disaster recovery operations by leveraging low costs and operations. In this chapter, we will discuss the three main models that users can adapt to increase their business continuity.

The following topics will be covered in this chapter:

- Availability metrics
- Backup and restore
- Pilot light
- Warm standby
- Multi-site active-active

Technical requirements

There is no technical requirement for this chapter.

Availability metrics

High availability (HA) is a complex variable, because a deep understanding of how every component interacts with each other must be in place. Metrics come to the rescue; they provide a quantitative perspective of the system operations discipline, but in some cases, not all of the factors are correlated, and a qualitative perspective can help. This database of indicators can be used to build a model to represent the availability qualities of a system.

A system's availability can be calculated by the number of nines in the digits, representing the percentage of time a system is operating, without downtime. This is shown in the following table:

Number of nines	Uptime %	Downtime per day
1	90%	2.4 hours
2	99%	14 minutes
3	99.9%	86 seconds
4	99.99%	8.6 seconds
5	99.999%	.86 seconds

The business perspective

A risk management framework must evaluate the impact when a business function gets interrupted and companies need to invest time in business continuity planning; the impact must be categorized as critical (urgent) or non-critical (non-urgent), in order to architect its availability. Several disruptions can be tolerated for a certain timespan, and others, considered to be critical, need to be restored immediately after they are detected. Some activities are even considered critical by law.

A business function that is categorized as critical has to be monitored by the control plane and have robust recovery procedures in place.

Business impact analysis

To understand these business impacts, we have to assign a risk factor to major service disruptions. Amazon's one hour of downtime on Prime Day may have cost it up to \$100 million in lost sales.

To design for resiliency, two terms come into play, as follows:

- **Recovery Time Objective (RTO)**
- **Recovery Point Objective (RPO)**

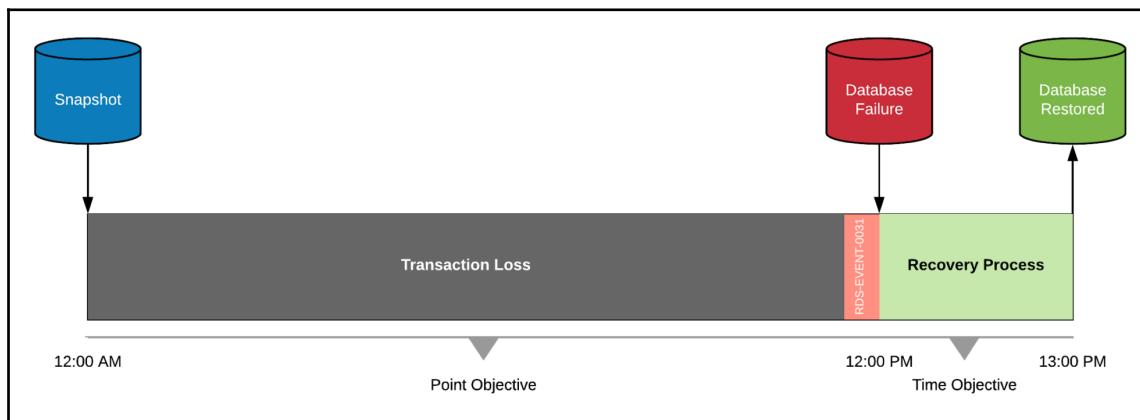
Recovery Time Objective (RTO)

Once the service interruption is detected, the RTO is the time from that point until the service is restored to its regular service level. For example, an RDS database is marked as **failed (DB Instance Status)** as it has a hardware failure and cannot be recovered as a part of the service. The system admin is subscribed to the RDS event SNS topic, and he/she receives a failure event (**RDS-EVENT-0031**) via email.

Suppose that the business has stated that the time objective for this database is one hour. From that point, the system admin has one hour to restore the service operation, according to the **Organization Level Agreement (OLA)**.

Recovery Point Objective (RPO)

The Recovery Point Objective is the acceptable amount of data that can be lost in the event of a downtime. For example, the failed RDS database can be restored in another instance, using the last snapshot available; in this case, the 12:00 AM snapshot. This recovery mode will drop eight hours of database transactions; from the business perspective, the risk associated with the loss of the master database node and the volume of transactions is low, and the cost to maintain a standby replica is high:



To lower the RPO and RTO, several methods can be put in place, like continuous snapshots, active replication, and redundancy.

The AWS Disaster Recovery options will be discussed next, ordered from the highest to the lowest recovery objectives.

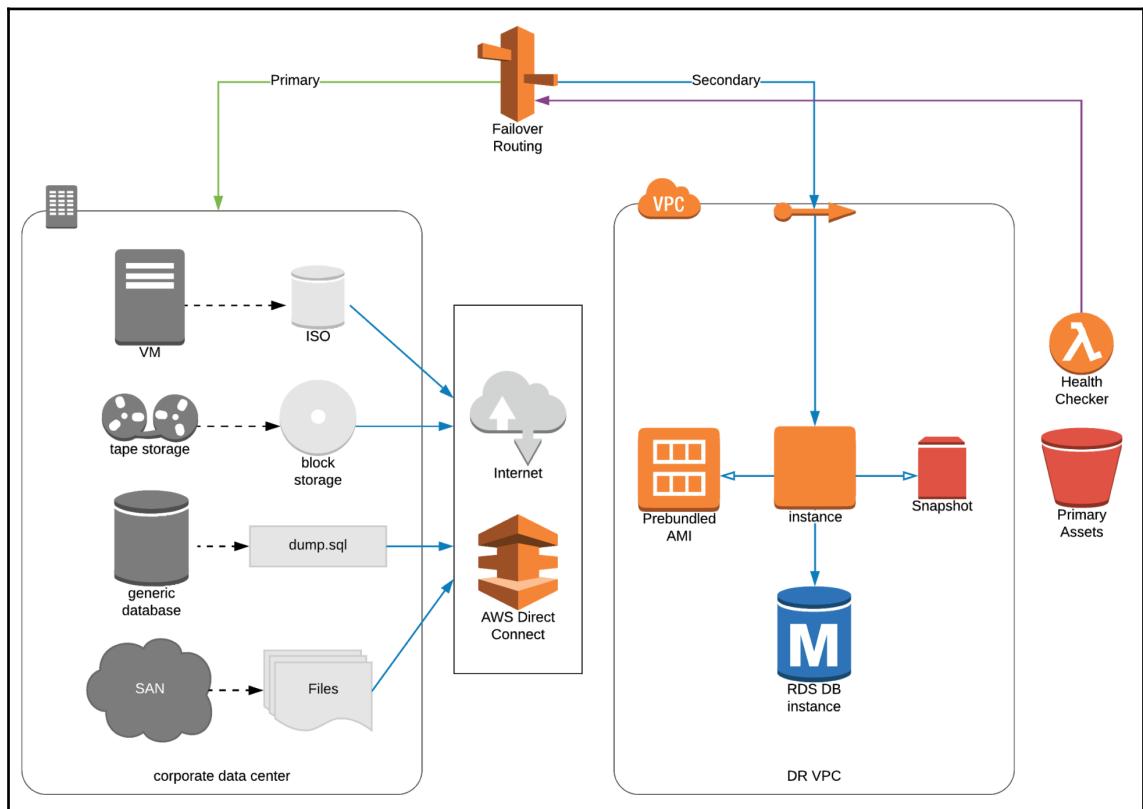
Availability monitoring

Monitoring the availability of the primary service can be done in the form of health checks. When a downtime condition happens, it will trigger failover; in this case, a DNS failover is used via Route 53 HTTP health checks, and a Failover Routing Policy is configuring the on-premises side as **Primary** and the cloud environment as **Secondary** to fail over the AWS cloud.

Backup and restore

This strategy consists of the active replication of all of the primary assets that will be needed to recover service on the cloud. This can include files, databases, disks, and virtual images, to name a few.

The following diagram shows the overall solution for a backup and restore strategy:



Preparation phase

In the preparation phase, a golden AMI needs to be created, with all of the required dependencies to run the application. You also have to allocate an Elastic IP that will be used to route production traffic to the disaster instance. This instance will be created when the failure is notified.

EBS volumes can be created on demand, with the latest snapshot; services like AWS Storage Gateway replicate disk blocks directly on S3 with the EBS format, so data can be restored on premises, or can be used in new EBS volumes, which will be attached to the recovery EC2 instance.

In an Elastic IP, the static resource will be used to switch traffic to AWS. Create the relevant security groups, IAM roles, and networking.

In the case of a disaster

In a disaster scenario, manual intervention can be used to restore the services on the cloud or on premises, but automation is always best. To automate the failure response, you can use AWS Lambda, by performing a `GetHealthCheckStatus` request on the health check object of Route 53, which represents your primary web application. Lambda scheduled functions are designed to provide a minimum of one-minute granularity for execution, plus the health check request interval (the standard is 30 seconds, and the fast version is 10 seconds).

The Lambda function can programmatically trigger the following actions:

- Creating the RDS instance
- Creating an EBS Volume
- Creating an EC2 instance with the DR AMI
- Associating the Elastic IP to the running instance

The EC2 instance will perform the following activities:

- Attaching the EBS volume to the instance, via bootstrapping
- Downloading the database backup file from S3 in the instance store, and execute it remotely, using the RDS endpoint

Some of the relevant services are as follows:

- IAM
- Route 53
- Lambda
- EBS
- EC2
- RDS
- CloudWatch
- Direct Connect
- Storage Gateway
- Import Export Snowball

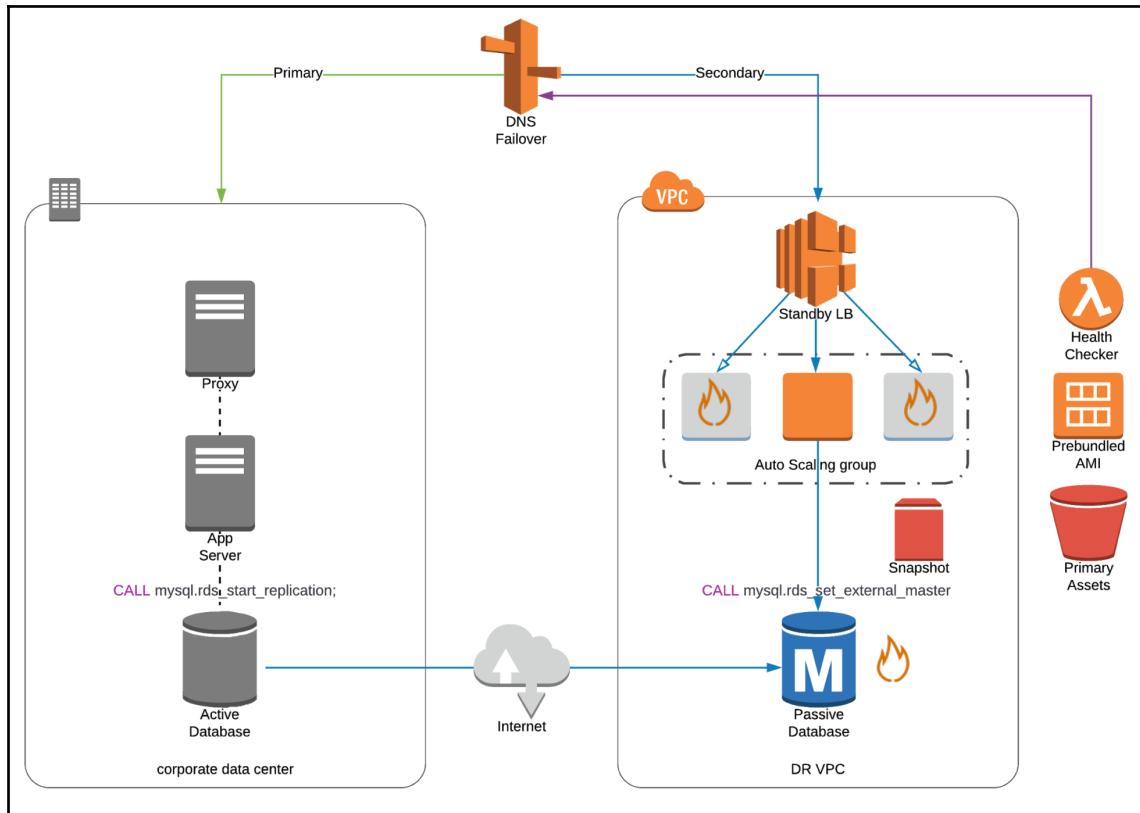
Trade-offs

Objective	Measure	Advantages	Constraints
RPO	High	<ul style="list-style-type: none">• Low cost• Simple approach• Can be done manually	<ul style="list-style-type: none">• Backing up Windows depends on the network throughput• Hybrid storage will require different approaches to data recovery
RTO	High	<ul style="list-style-type: none">• Services without SLAs will benefit from this model	<ul style="list-style-type: none">• Data loss is high

Pilot light

Pilot Light uses the analogy of a gas heater, in which the standby system is always on, ready to ignite the complete system when the failure occurs. Primary data must be frequently replicated to the cloud. The premises database layer needs to be in sync with a standby replica in the RDS.

The following diagram shows the overall solution for a backup and restore strategy:



The preparation phase

Database synchronization must be configured from the source (production) to the pilot light environment (warm standby). To achieve this, you can use the vendor standard replication scheme to replicate the data layer transactions asynchronously. The Database Migration Service can also be used, leveraging the high throughput of multiple databases into a single, consolidated, federated database.

You must also replicate other dependencies, less frequently than the primary data source, to reduce the mismatched dependencies across environments (software packages, libraries, system images, and so on).

In the case of a disaster

In the case of a disaster, we can leverage sophisticated monitoring techniques. Route 53 can be used to monitor the primary environment endpoint; if the primary web environment consists of multiple health checks, a calculated health check can be used to gain a wider understanding of the health of your services.

The failover routing policy provides records that can be configured as ALIAS (for CNAMES DNS) or as type A (for IP addresses). The health check string matching feature allows you to find a specific string or pattern in the check response. An example is as follows:

```
{
    "STATUS"      : "OK",
    "DB"          : "UP",
    "MQ"          : "UP",
    "API"         : "UP"
}
```

Route 53 will automatically fail over to the warm standby, and, with the use of Lambda (as in the backup and restore strategy), the pilot can be triggered to initiate the furnace, by starting the web server, adjusting the AutoScaling limits so that the pilot light environment can accommodate the current load of the production environment; this could take several minutes to be fully usable and horizontally scale to match the current demand.

Some of the relevant services are as follows:

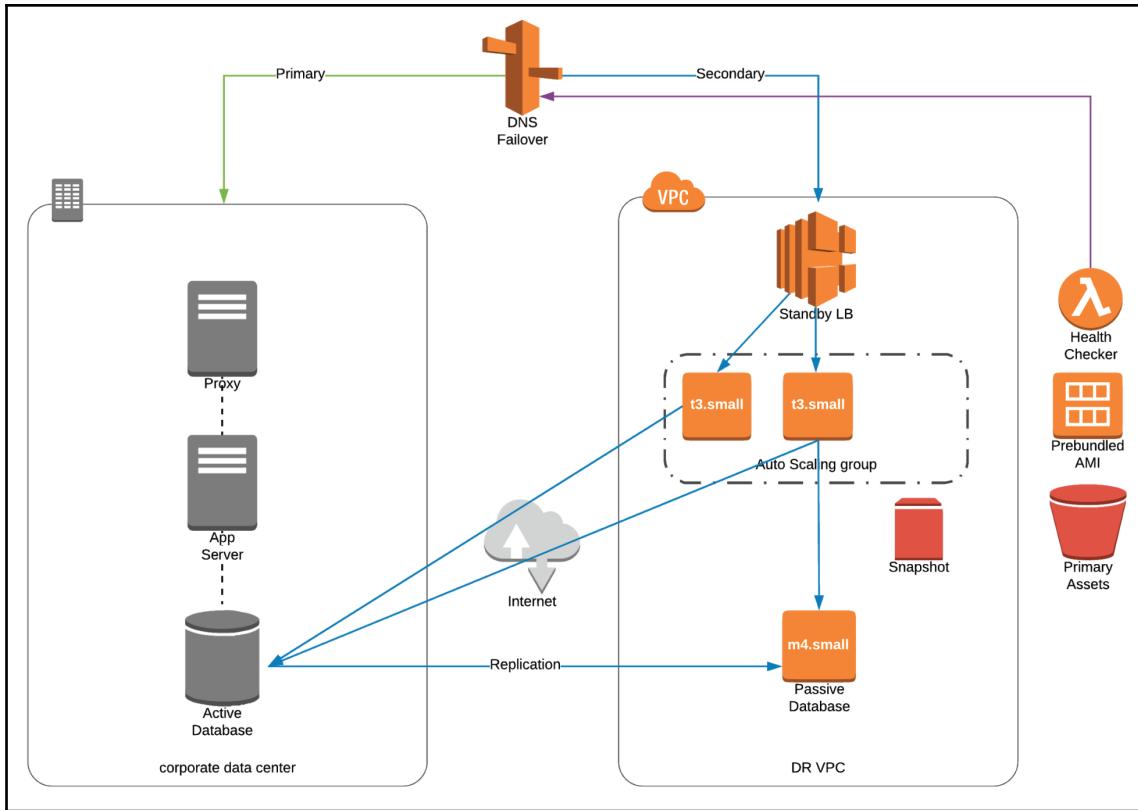
- Relational Database Service (RDS)
- Database Migration Service (DMS)

Trade-offs

Objective	Measure	Advantages	Constraints
RPO	Medium	• Transactions are fully replicated	<ul style="list-style-type: none"> • Database replication tools • Replication monitoring • Licensing • Compliance standards
RTO	Medium	• Less downtime than backup and restore	<ul style="list-style-type: none"> • Cost increase • Connectivity and throughput

Warm standby

In this scenario, the warm standby is always running and fully functional, but with the minimal amount of resources, to keep the costs at a minimum. This environment can be used for testing and quality assurance. It is illustrated in the following diagram:



The preparation phase

To prepare, create the required golden AMIs, with all of the dependencies and code built in, so the service can fail over at any time, as a result of a disruption in the active site (on premises). Prepare a replication instance that is constantly updated with the latest primary site transactions.

In the case of a disaster

In the case of a disaster, use Route 53 with the failover routing policy, since these servers are always using the primary active database that they will require to implement a fail-fast code strategy, to switch traffic to the new master database (warm passive standby). This can be done by catching connector exceptions in the code and using sensible timeouts. Scale horizontally, to accommodate for the current production traffic. In order to build resiliency, use multiple AZs.

Some of the relevant services are as follows:

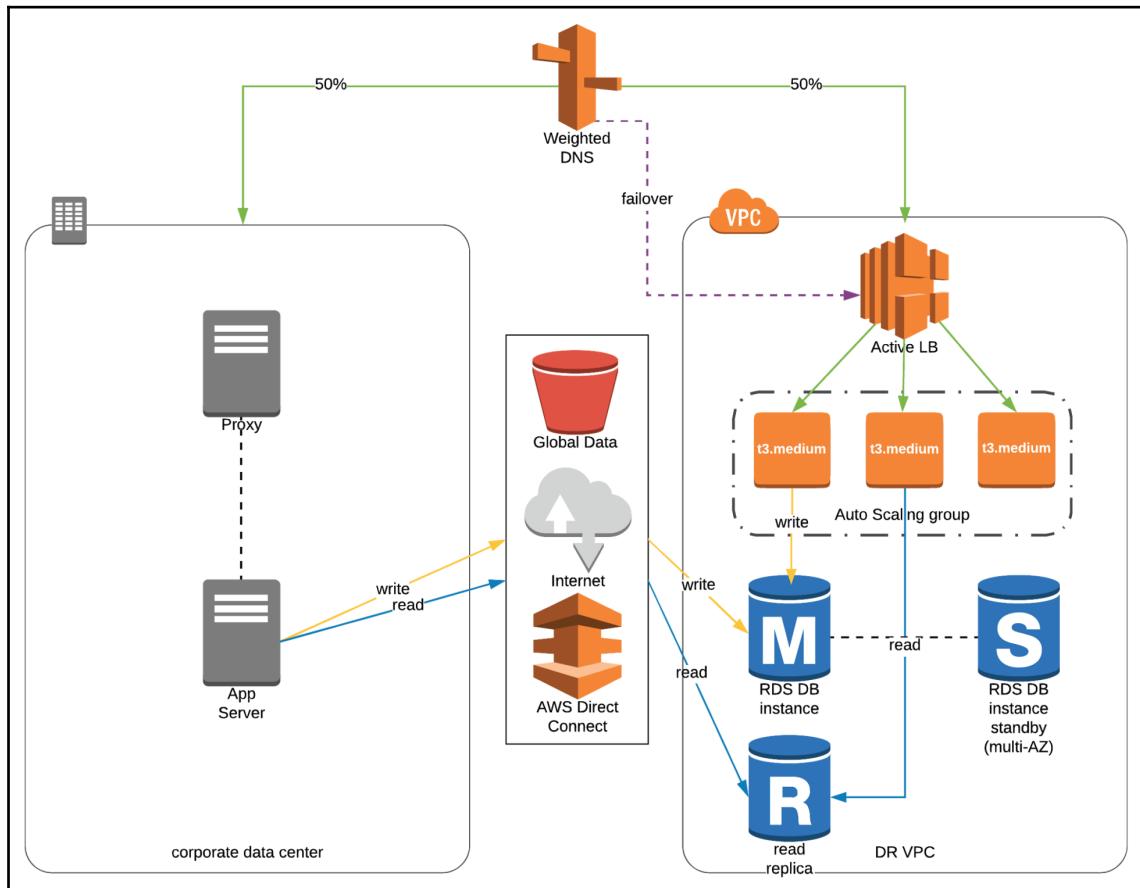
- Elastic load balancing, with cross-zone load balancing
- EC2, with burstable capacity (t3 family)

Trade-offs

Objective	Measure	Advantages	Constraints
RPO	Low	<ul style="list-style-type: none">• Secondary environment always available	<ul style="list-style-type: none">• Low capacity servers• Replication monitoring• AMI upgrading
RTO	Low	<ul style="list-style-type: none">• Transactional data is durable and in sync	<ul style="list-style-type: none">• Replication lag• Network stability• Synchronization of data

Multi-site active-active

The multi-site active-active strategy provides strong advantages, as compared to other DR scenarios, because a redundant service architecture is running in the cloud. In this scenario, the primary service (on-premises) can failover at any time, and can cope with the production traffic. A strong point of multi-site active-active is that traffic can be split for AB testing, game day exercises, and custom traffic distribution. Multiple regions can be used to achieve golden availability metrics with five nines:



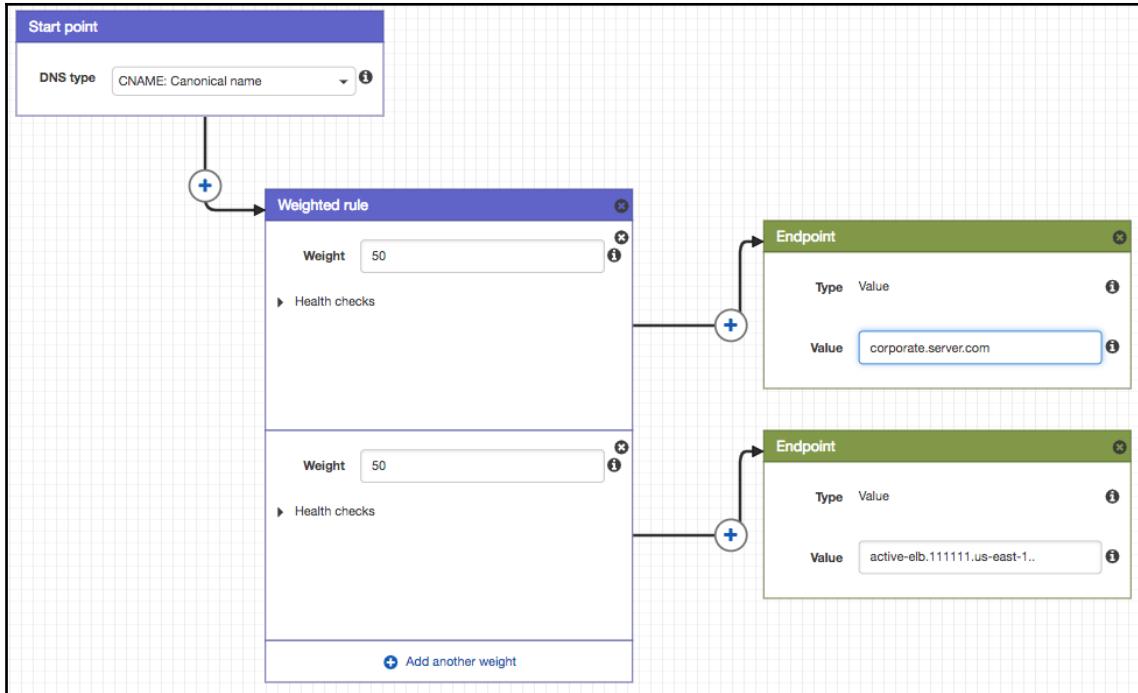
The preparation phase

To prepare, migrate the business-essential applications, and perform database synchronization. This replication can be done synchronously or asynchronously, depending on the quality of the service to be delivered.

Create a Route 53 health check for each environment, and a failover policy in which on-premises is the primary target and AWS is the secondary target. Also, configure a DNS traffic split of 50%, as follows:

Location	Routing Policy	
Failover	Weighted	
On-premises	<p>Routing Policy: Failover</p> <p>Route 53 responds to queries using primary record sets if any are healthy, or using secondary record sets otherwise. Learn More</p> <p>Failover Record Type: <input checked="" type="radio"/> Primary <input type="radio"/> Secondary</p> <p>Set ID: Primary</p>	<p>Routing Policy: Weighted</p> <p>Route 53 responds to queries based on weighting that you specify in this and other record sets that have the same name and type. Learn More</p> <p>Weight: 50</p> <p>Set ID: Virginia Load Balancer</p> <p>Description of this record set that is unique within the group of weighted sets. Example: My Seattle Data Center</p>
AWS	<p>Routing Policy: Failover</p> <p>Route 53 responds to queries using primary record sets if any are healthy, or using secondary record sets otherwise. Learn More</p> <p>Failover Record Type: <input type="radio"/> Primary <input checked="" type="radio"/> Secondary</p> <p>Set ID: Secondary</p>	<p>Routing Policy: Weighted</p> <p>Route 53 responds to queries based on weighting that you specify in this and other record sets that have the same name and type. Learn More</p> <p>Weight: 50</p> <p>Set ID: San Francisco Data Center</p> <p>Description of this record set that is unique within the group of weighted sets. Example: My Seattle Data Center</p>

The Route53 traffic flow can be used to graphically describe these routing rules, precedence, or even more complex flows:



In the case of a disaster

In the case of a disaster, Route53 takes the full responsibility of failing over the secondary environment. Third-party DNS providers can do the job, if they provide fast replication over anycast networks.

Since this type of architecture is active-active, a different architectural approach can be taken, like **Command Query Responsibility Segregation (CQRS)**, in which two independent models are used: one for writing, and one for reading. An Amazon Aurora cluster with read replicas can be deployed, and DynamoDB (with global tables) can be very helpful in this scenario.

Some of the relevant services are as follows:

- DynamoDB global tables
- RDS Aurora, with multi-master
- Route53 traffic flow

Trade-offs

Objective	Measure	Advantages	Constraints
RPO	Low	<ul style="list-style-type: none">• Database is always replicated and consistent	<ul style="list-style-type: none">• Network throughput and availability• Eventual consistency
RTO	Low	<ul style="list-style-type: none">• Low latency, using dedicated network connectivity and an AWS backbone	<ul style="list-style-type: none">• Synchronous or asynchronous writes• Quorum and last write wins configurations

Best practices

- If the data volume is considerably large, Import/Export Snowball can be used to transfer large quantities of data, without using the internet
- Optimize data transfer with a private fiber service, like Direct Connect, to improve the connection stability
- Use reserved instances to guarantee the availability of computing resources
- Use resource pooling (Elastic IP)
- Create snapshot lifecycle policies
- Monitor for resource exhaustion and service limits (CloudWatch, Trusted Advisor)
- Use private hosted zones in Route 53, to create alias records for the RDS endpoint, or any dynamic resource naming
- Use multiple availability zones
- Automate and monitor the recovery phase

Summary

In this chapter, we learned about the main patterns used in DR strategies by using the cloud. We learned to implement successful backup and restore, use a pilot light, and multi-site active-active scenarios.

We also learned how to implement a full DR exercise in a hybrid environment.

Further reading

The implementation details in this chapter represent an opinionated view of how to recover using managed services, and do not represent the standard perspective of AWS. Use the implementation details as guidelines only. For more information, read the white paper, *Using Amazon Web Services for Disaster Recovery*, located at <http://bit.ly/s2QC6P>.

9

Storage Options

Enterprises produce high volumes of data every day and they must be managed in a secure and durable way. Storage services enable companies to manage data easily. Every day, huge quantities of data is generated at a high velocity with high variety, and the ability to get information out of it becomes a real challenge.

Data-driven companies like Formula One are great examples of how much profit a billion dollar company can make by the adoption of big data and data processing technologies.

The following is quoted from <https://aws.amazon.com/solutions/case-studies/formula-one/>:

"During each race, 120 sensors on each car generate 3 GB of data, and 1,500 data points are generated each second".

To handle enterprise data, governance models must be adopted like data classification and its business lifecycle. Several factors must be taken into account when handling data like the speed at which is generated, the kind like structured, (relational databases), semi-structured (JSON documents) or unstructured (images), the access patterns (read oriented, write-oriented), the cost structure, and other relevant perspectives.

AWS provides a full portfolio of storage solutions that make a perfect fit for different use cases. Services designed to be fully compatible with your existing applications are ready to work at a massive scale. The AWS managed services do all the heavy lifting for you so that you can focus on the business values rather than the storage maintenance.

- Understand the main storage options in AWS
- Work with S3 and change the storage type for different scenarios
- Configure a lifecycle policy for an S3 bucket
- Archive data in Glacier and retrieve the data
- Design a data structure in DynamoDB

Technical requirements

There are no technical requirements for this lesson.

Relational databases

Relational databases have a wide spectrum of use cases and excel in several properties related to persistence such as atomicity, consistency, independency, and durability. Relational models have been widely used in many industries and have proven to solve almost any kind of problem.

Normalization is an important aspect of relational models, since they provide strong consistency and control mechanisms as foreign keys and constraints that sustain business rules.

RDS

The **Relational Database Service (RDS)** is compatible with the following engines and acts as a drop in replacement for the following engines:

- PostgreSQL
- MariaDB
- MySQL
- Oracle Database
- Microsoft SQL Server

RDS provides its own database engine called Amazon Aurora, which is an AWS product designed to be compatible with engines like MySQL and PostgreSQL.

RDS is a great choice to migrate and integrate existing applications in the cloud. RDS is designed to scale vertically and horizontally to support any kind of relational workload. RDS is engineered to provide an SLA of 99.95%.

Managed capabilities

A managed service like RDS provides administration tasks like automated backups, failover, and host replacement in the case of failures and software patches.

RDS is a container service, which means that the database engine, infrastructure, platform, and operations are fully managed. RDS also provides backups and high availability for your database when deployed with the multi-AZ attribute.

The service is responsible for patching your database in scheduled maintenance windows. Keep it secure with the ability to encrypt data and communications and the ability to recover your data in the case of a disaster with database snapshots.

Navigate to the RDS console to analyze the main options we have to create a DB instance:

The screenshot shows the Amazon RDS dashboard for the US West (Oregon) region. On the left sidebar, under the 'Instances' section, the 'Create database' button is highlighted. The main content area displays various RDS resources: DB Instances (0/40), Allocated storage (0 bytes/100.00 TB), Reserved Instances (0/40), Snapshots (67), and Event Subscriptions (0/20). To the right, there's an 'Additional information' panel with links like 'Getting started with RDS', 'Overview and features', and 'Documentation'. Below that is a 'Database Preview Environment' section with a note about early access to new DB engine versions. At the bottom, a 'Service health' section indicates that the Amazon Relational Database Service (Oregon) is operating normally.

In the left side, we have selected the RDS dashboard and the options right below.

Instances

Every database you create can be created on db optimized instance types such as db.t2, db.m4, db.r4, and so on, depending of the database engine chosen. The storage type can be general purpose SSD and provisioned IOPS, which is the recommended option.

The allocated storage starts with a minimum of 100 GiB and a maximum of 16 TiB. Customers have the option to increase the storage or, change the underlying storage type.

Database objects are created on database instances. Think about security and stability for all RDS customers, having the ability to update and upgrade every single database instance in a consistent way. To achieve that kind of control, AWS has to take full control of every layer to run the database engine, this is why privileged access to the operating system is limited. Nevertheless, customers have the ability to fine-tune and configure their engines through parameter groups.

Parameter groups

Parameter groups are containers for configuration parameters specific to every database engine and version. Via parameter groups the database runtime can be modified, several changes can be influenced or specific behaviors can be set. Examples of these parameters are the `autocommit` mode and the `auto_increment_increment` values for MySQL.

Option groups

DB engines have the option to run plugins or extensions that enhance the engine behavior. For example, Oracle Database can enable the Oracle Spatial package to work with spatial indexes, or **Transparent Data Encryption (TDE)** to improve security and manageability. This is why the options groups come handy and is a way RDS can promote extensibility and support for additional database features.

Snapshots

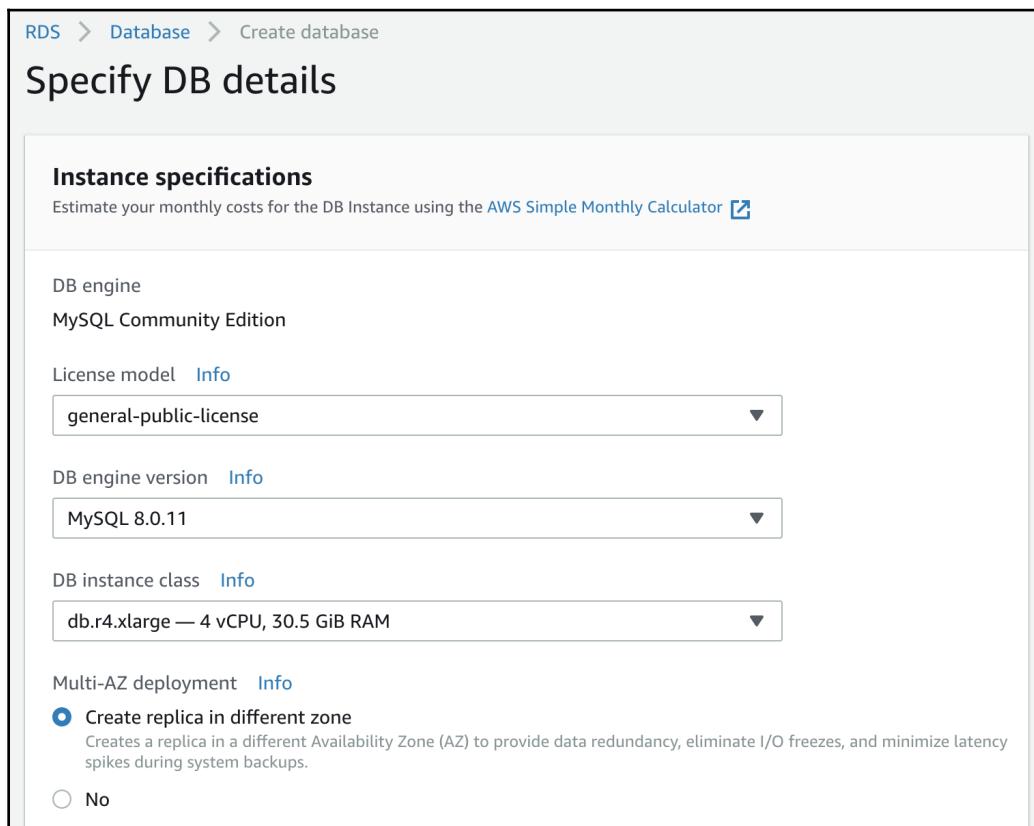
Snapshots are a way to restore in a point in time the database by performing a volume snapshot. This is a managed activity that uses S3 managed buckets. It is pretty common to think that these snapshots can be managed from the user's end because they are already in S3 and apply lifecycle policies to archive them in Glacier; but this is a misconception. RDS snapshots are only managed through the API with the capability to retain up to 7 days of snapshots. These snapshots can be copied across regions to synchronize data and recover from a regional disaster.

Events

RDS can publish all the database activity happening at the instance and API level via SNS topics pushed via email subscription. There are several event categories that can be subscribed, for example, DBAs get notifications from configuration changes and automatic backups, and operations receive availability and failover events.

Multi-AZ

This is maybe one of the best-managed capabilities of RDS. Single-AZ databases (default option) represents a single fault domain that fully comprises the overall availability of persistent systems that make use of relational databases. Multi-AZ gives customers the ability to deploy a secondary instance running in another Availability Zone independent from the master database. In the following screenshot, you have the option to deploy the instance in single AZ or multi-AZ:



This standby instance is synchronously replicated. This means that every commit in the master is written in the secondary replica before the acknowledgment of the operation. Primary, and secondary are always up to date and automatic failover is taken by the service in the following scenarios:

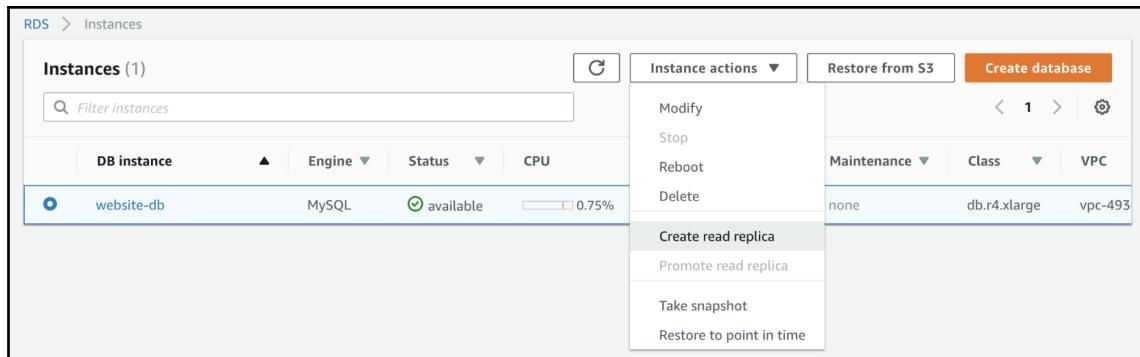
- Loss of availability in primary Availability Zone
- Loss of network connectivity to primary
- Compute unit failure on primary
- Storage failure on primary
- Maintenance Windows

RDS isolates instance IPs by providing a DNS resource that applications use to connect via the protocol and the connector provided by every engine provider.

Read replicas

RDS can be scaled horizontally by creating up to 5 read replicas (15 for Amazon Aurora) to offload read workloads. These replicas can be used to increase the read throughput, for example, reporting and business intelligence. Read replicas are replicated asynchronously which means that certain replication lag is present and are eventually consistent.

To create a read replica select the master database and choose **Instance Action | Create read replica**, as shown in the following screenshot:



The read replica creation will take some minutes when it becomes available you can now connect using the replica endpoint as shown in the following screenshot:

The screenshot shows the AWS RDS Instances page. The instance name is "website-read-replica-1". The "Summary" section shows the engine is MySQL 8.0.11, DB instance class is db.r4.xlarge, DB instance status is available, and Pending maintenance is none. The "Connect" section displays the endpoint "website-read-replica-1.xxxxxxx.us-east-1.rds.amazonaws.com" (circled in orange), port 3306, and publicly accessible status Yes. Below this, the "Security group rules" section lists two rules: one for rds-launch-wizard-1 (sg-016992dd1d2126248) allowing CIDR/IP - Inbound from 200.91.236.86/32, and another for rds-launch-wizard-1 allowing CIDR/IP - Outbound from 0.0.0.0/0.

Advanced scenarios can be implemented to load balance read traffic using multiple RDS read replicas with HAProxy for example.

Virtual sharding can also be implemented to optimize and extend the limits of a single writable database. To implement this mechanism multiple instances need to be provisioned and application logic must be in place to condition read and write requests.

Caching

Several patterns can be found in your databases, for example, a higher rate of ReadThroughput than WriteThroughput reported from CloudWatch. Amazon ElastiCache is an in-memory datastore that can help to alleviate the frequent use of views, complicated queries or read-only data as catalogs or the top selling products.

Object storage

Object storage is an abstraction which deals with all the complexities of managing physical storage infrastructure, read and write operations are optimized for large scale and performance. The data plane is 100% managed and distributed, the control plane takes control of every request authenticating and authorizing every object operation with a resilient design. Customers are always in control of their data with open protocols avoiding lock-in.

How object storage is different from other persistence mechanisms?

In object storage there is no provisioning of infrastructure and no upfront costs, customers pay only for what they use on a pay-as-you-go basis by leveraging the economies of scale.

This technology sits in the middle of everything, any kind of application or business strategy can benefit from this storage strategy from small servers to large enterprises on big data.

Simple storage service

S3 provides a key-value store for objects; think of it as a really big HashMap in which you provide a key and store a blob of data as is. Data in S3 is managed through buckets which can contain an unlimited number of objects. Buckets are created with the regional scope and bucket names need to be unique because they are global and DNS compliant.

The naming rules for buckets which are as follows and can be found at <https://docs.aws.amazon.com/AmazonS3/latest/dev/BucketRestrictions.html>.



Data organization

Data is organized logically via object keys, these keys need to be unique at the bucket level. To list keys in a bucket, you need to provide the bucket name and the region, the list operation by default will retrieve 1000 keys where you have the option to paginate and retrieve a subset of the keys.

There is no underlying filesystem in S3, the data plane is flattened nevertheless hierarchical structures can be simulated because you can search for subsets of objects by listing them hierarchically using a prefix and a delimiter.

```
CommonPrefixes = photos
Delimiter = '/'
photos/2019/January/sample.jpg
photos/2019/February/sample2.jpg
photos/2019/February/sample3.jpg
photos/2019/February/sample4.jpg
```

Keys can be filtered virtually by a prefix that resembles hierarchical structures.

```
aws s3 ls s3://mybucket/notReallyAFolder/object.txt
```

S3 is not a relational database, there is no SQL semantics or materialized views that can be queried frequently, but with Amazon Athena customers can solve an important number of scenarios. With Athena, you can use SQL and perform interactive queries and analysis of all your data set.

S3 was designed with security in mind and provides three different ways of encryption, with the full control of where encryption is made, plus secure transmission protocols.

The ability to store data and treat them in the most abstract way; as an object provides a lot of flexibility and simplifies its management. From the S3 perspective objects are just sequences of bytes and every object is an opaque data type. S3 is an abstract service thus all access is through an API. S3 was designed to be operated via the Internet with unlimited capacity and is highly available (99.99%) and highly durable (99,99999999%) in nature over a given year. Cloud storage is designed to be more reliable than physical storage.

"If you store 10,000 objects with us, on average we may lose one of them every 10 million years or so. This storage is designed in such a way that we can sustain the concurrent loss of data in two separate storage facilities."

- Jeff Bar, Chief Evangelist for AWS

Integrity

To improve reliability S3 calculates an MD5 hash of the object, this process can be validated from both ends; the customer and AWS. This mechanism will give customers the guarantee that the object hasn't been compromised on transit when it is created, and that the object remains the same when reading it.

Customer	AWS
Content-MD5 HTTP Header	ETag

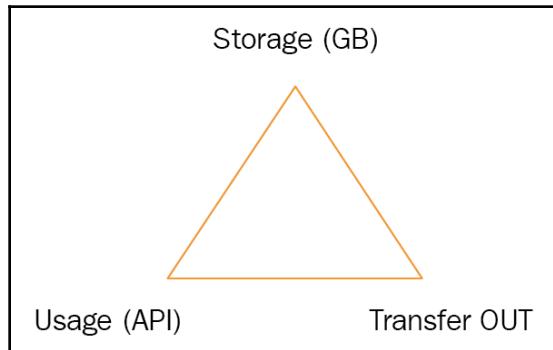
Internally AWS constantly compares objects with the MD5 database to validate consistency. When inconsistencies are detected the object is repaired automatically. Buckets can also be versioned, providing data integrity by keeping multiple versions of every object as they are changed through time.

Availability

S3 can tolerate the concurrent loss of two facilities and remain available, so it is tolerant to network partitioning in favor of eventual consistency. Availability can vary depending of the type of storage being used with different service-level agreements.

Cost dimensions

The **Simple Storage Service** doesn't charge you for the data ingress. You pay for the storage volume per month, for the use of your data calculated per request (PUT, COPY, POST, LIST, and GET requests) and for the transfer out of cross region requests.



Reducing cost

To find opportunities for costs you can use the **Analytics** feature under **Management** in the console to find usage patterns for candidate objects to transition to **Infrequent Access (IA)**.

A screenshot of the AWS Storage Analytics interface. The top navigation bar has tabs for Overview, Properties, Permissions, Management, Analytics, Metrics, and Inventory. The Analytics tab is active, indicated by a blue background and a circled number '1'. Below the tabs, there's a section titled 'Storage class analysis' with a progress bar showing 'Analyzing your data' and a message 'We are analyzing your storage usage and will share the observed infrequent access patterns.' A large orange circle with the number '2' contains an arrow pointing to the 'Add filter' button in a sidebar. The sidebar also includes fields for 'Filter name' (set to 'allfiles') and 'Prefix / tags to monitor (optional)', both of which have arrows pointing to them from the orange circle. At the bottom of the sidebar are 'Save' and 'Cancel' buttons. To the right of the sidebar, there's a small chart showing three bars of increasing height, with the text 'No data available yet.' above it.

This analysis feature groups objects by age and creates a CSV file with detailed information about every object access pattern.

Durability

AWS S3 provides four kinds of storage and one option in order to optimize for costs. For maximum durability, you can use STANDARD, STANDARD_IA, or GLACIER. If you can afford to lose objects and cost is the primary objective then you should use STANDARD_IA, ONEZONE_IA, and the RRS option.

Maximum durability

Every object by default is stored with STANDARD storage class with 119s of durability. Use this storage layer for objects that require real-time access and cannot be lost under any circumstance.

The STANDARD_IA layer provides a balance between cost optimization, availability, and durability. It provides the same durability of STANDARD, but with lower costs because data is classified for infrequent access. Nevertheless, when data is read, the cost is a little bit higher than the STANDARD tier.

Limited durability

The ONEZONE_IA offers the best alternative for low cost and real-time access but at the cost of a higher probability to lose your data. This storage layer will only replicate objects at the AZ level with 119s but it is not resilient to the complete loss of the AZ.

The RSS (REDUCED_REDUNDANCY) layer provides low cost and low replication ratio, RRS objects have an average annual expected loss of 0.01% of objects. If an RRS object is lost, when requests are made to that object, S3 returns a 405 error.

Let's assume you are storing 800 MB files in size and you have 1 PB of data. With RSS you can expect to lose .01% of your files every year.

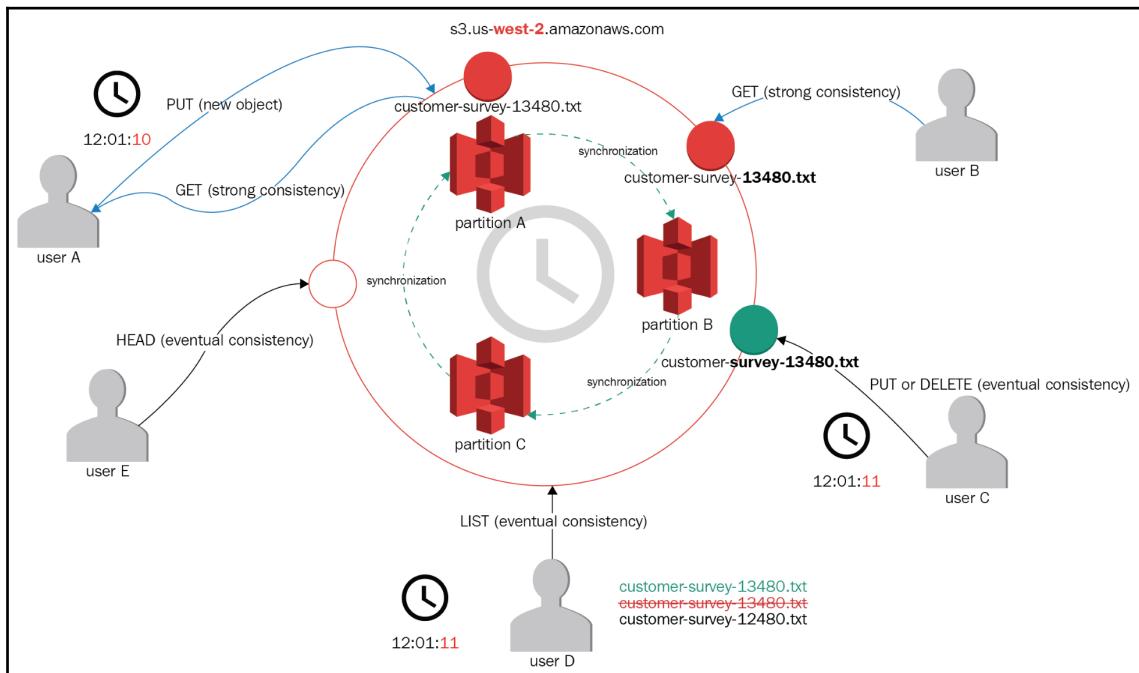
$$.0001 \times 1.2 \text{ Billion} = 12 \text{ Million files lost per year}$$

Use cases

Use ONEZONE_IA and REDUCED_REDUNDANCY for derived data that can be reproduced from the original (classified as STANDARD); for example financial reports (RSS) generated from bank statements (STANDARD) or image thumbnails (ONEZONE_IA) from original bitmaps (STANDARD_IA).

Consistency

To achieve high availability S3 uses multiple distributed partitions (servers) to service requests. The eventual consistency model of S3 is the way of how changes committed to the system become visible to every actor and is an important trade-off in distributed storage systems. The following scenarios can materialize as a result of replication latency.



1. User A PUTs a new object at 12:01:10, S3 responds with a 200 OK status; now every user has a strong consistent view of the system with the ability to read after the object was written.
2. User B performs a GET operation and this results in a consistent read.

3. User C DELETES or PUTS a change on the object at 12:01:11, which is an eventually consistent operation.
4. User D requests a LIST operation at the same time as User C
5. Users C and D can be in one of three different states:
 1. The key does not exist as a result of the deletion.
 2. The object is stale and shows the previous version of the object.
 3. The object is updated and shows the newest version of the object.
6. User E requests a HEAD operation to ask if the object exists (before the object is created); this is always eventually consistent read.

Storage optimization

There are three forms to change the storage class in S3:

- When the object is created
- Performing a copy of an existent object with a new storage type
- Using lifecycle policies

Creating objects from the CLI

The following command will perform the copy operation with the default storage type (STANDARD):

```
aws s3 cp storage-types.txt s3://object-types-solutions-architect
```

The s3api command provides detailed information about the objects in the specified bucket as follows:

```
Gabanox-MBP:~ gabanox$ aws s3api list-objects --bucket object-types-solutions-architect
{
    "Contents": [
        {
            "Key": "storage-types.txt",
            "LastModified": "2018-10-13T20:43:36.000Z",
            "ETag": "\"eff5bc1ef8ec9d0e640fc4370f5eacd\"",
            "Size": 3,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "sandbox",
                "ID": "99108d3ed498e2a918f54529f7a119f7c6e6223840a761164df1ef215b36702f"
            }
        }
    ]
}
```

Try the same command but this time with --storage-class REDUCED_REDUNDANCY:

```
aws s3 cp storage-types.txt s3://object-types-solutions-architect --  
storage-class REDUCED_REDUNDANCY
```

You can validate in the console as well, now the object has been created with RSS.

 storage-types.txt	Oct 13, 2018 3:46:33 PM GMT-0500	3.0 B	Reduced redundancy
---	-------------------------------------	-------	--------------------

Try the same but this time with Infrequent Access and OneZone_IA.

```
aws s3 cp storage-types.txt s3://object-types-solutions-architect --  
storage-class STANDARD_IA
```

```
aws s3 cp storage-types.txt s3://object-types-solutions-architect --  
storage-class ONEZONE_IA
```

We have now successfully created new objects specifying standard and one zone storage classes.

Copy an existing object

In order to change the object type you need to copy the original object, set the storage class to one of the following STANDARD_IA, ONEZONE_IA, or REDUCED_REDUNDANCY, delete the original object or overwriting the existing object.

```
aws s3 cp s3://object-types-solutions-architect/storage-types.txt  
s3://object-types-solutions-architect/storage-types.txt --storage-class  
STANDARD_IA
```

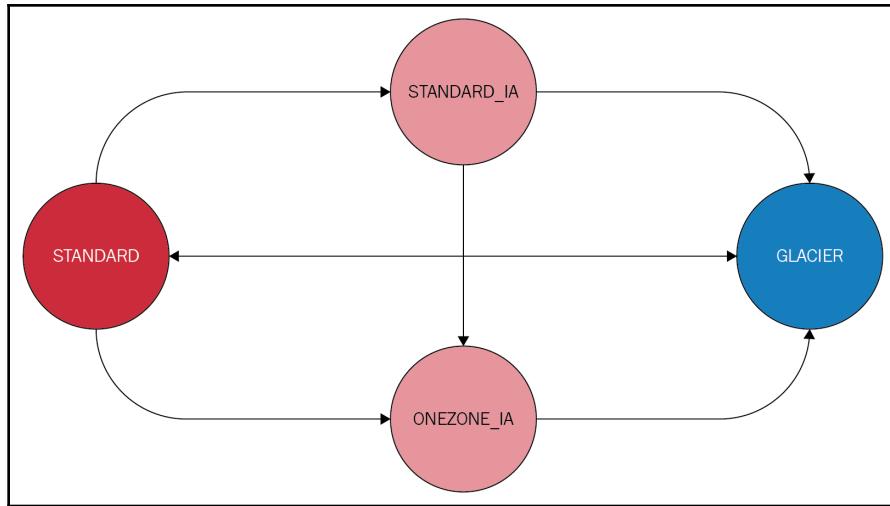
We have changed the storage type by copying the object with a new storage class.

Using a lifecycle policy

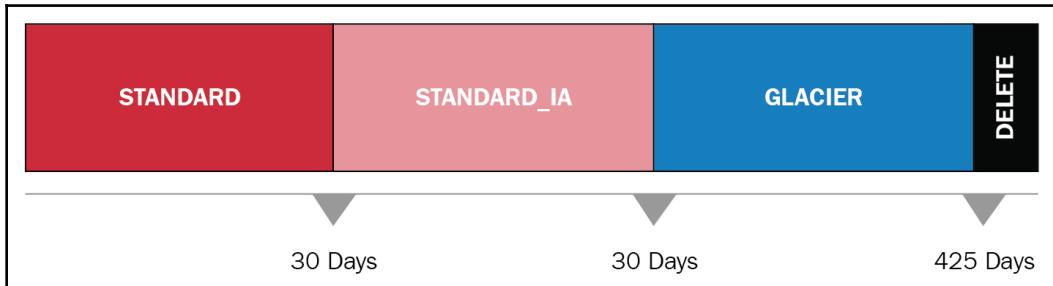
Multiple custom policies can be assigned at the bucket level to automatically transition objects between storage classes. It is possible to define the rules on which every object is transitioned to different layers until the deletion if required.

Lifecycle policies

Lifecycle policies are a great way to manage all the object lifecycles and achieve compliance. These rules are defined using XML and you can use the APIs to specify them. The following transitions are valid:



In our example we deal with hot objects accessed in real time, optimize them for Infrequent Access and then moved to a colder storage; in this case Amazon Glacier. After a year has passed the objects need to be deleted permanently.



To add a lifecycle rule, execute the following steps:

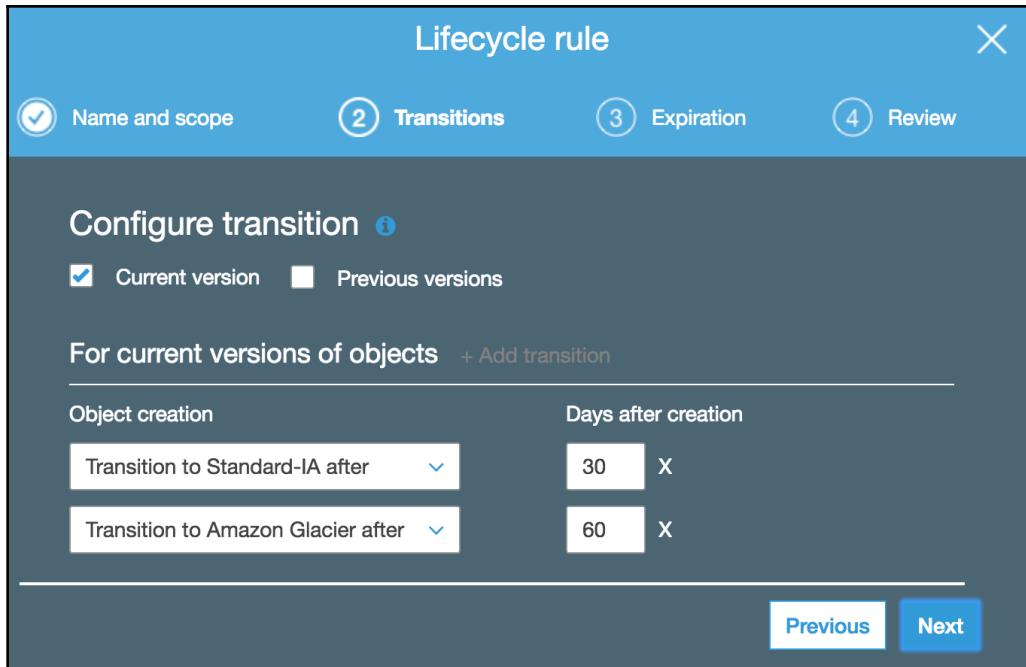
1. Navigate to the bucket to be managed, choose the **Management** tab, and under **Lifecycle** choose **+ Add lifecycle rule**:

The screenshot shows the AWS S3 console under the 'Lifecycle' tab. At the top, there are tabs for Overview, Properties, Permissions, and Management. Below these are sub-tabs for Lifecycle, Replication, Analytics, Metrics, and Inventory. A button '+ Add lifecycle rule' is highlighted with a yellow circle labeled '1'. To its right are Edit, Delete, and Actions dropdown buttons. A message box states: 'There is no lifecycle rule applied to this bucket. Here is how to get started.' An orange arrow points from the '1' on the '+ Add lifecycle rule' button to the message box.

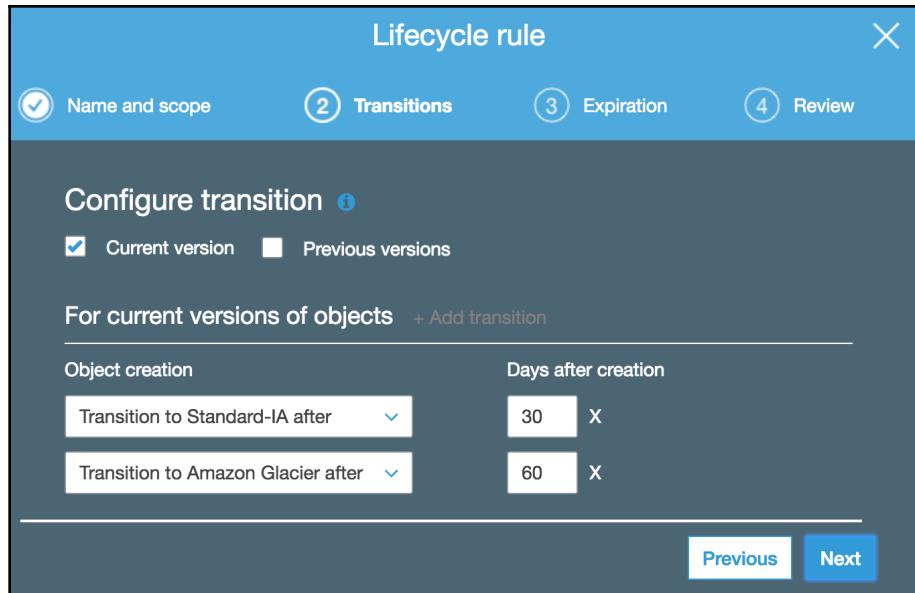
2. You will be prompted to choose a **Name and scope** for the rule, here you can use multiple object tags to filter and a prefix.

The dialog is titled 'Lifecycle rule' and has four tabs: 1. Name and scope (selected), 2. Transitions, 3. Expiration, and 4. Review. The 'Name and scope' tab contains a field 'Enter a rule name' with the value 'STANDARD_IA-STANDARD_IA-GLACIER-REMOVE'. Below it is a section 'Add filter to limit scope to prefix/tags' with a 'tag bank-statements | X' button. A text input field 'Type to add prefix/tag filter' is also present. At the bottom are 'Cancel' and 'Next' buttons.

3. In the step 2 **Transitions**, you can configure transitions for current versions and previous versions of the objects. Note that in order to transition the object to STANDARD_IA or ONEZONE_IA, the object needs to be stored in the STANDARD tier for 30 days, from STANDARD_IA to GLACIER 60 days is the minimum amount of time to optimize for archiving.



4. In our scenario we'd like to remove all the objects after a year, so let's configure a rule to purge the objects which are the result of 60 days + 1 year after.



5. Review the rules and transitions in step 4 on the **Review** screen, and you have created a fully managed object lifecycle policy.
6. Let's create another rule to transition objects from STANDARD storage to GLACIER after 15 days and delete them past 380 days, in this case, we will make use of the s3api CLI.



7. Create the lifecycle configuration file (`lifecycle.json`):

```
{  
    "Rules": [  
        {  
            "Expiration": {  
                "Days": 380  
            },  
            "ID": "STANDARD-GLACIER-DELETE",  
            "Filter": {  
                "Prefix": ""  
            },  
            "Status": "Enabled",  
            "Transitions": [  
                {  
                    "Days": 15,  
                    "StorageClass": "GLACIER"  
                }  
            ]  
        }  
    ]  
}
```

And put the bucket lifecycle policy substituting the bucket name:

```
aws s3api put-bucket-lifecycle-configuration --bucket bucket_name --lifecycle-configuration file://lifecycle.json
```

8. You can verify these changes in the console or by using the `s3cli` command:

```
aws s3api get-bucket-lifecycle-configuration --bucket bucket_name
```

You can download the JSON file here <http://bit.ly/2yh3Ug0>.

Archiving with Glacier

Amazon Glacier is a service optimized to store data with low cost and for long-term retention, it provides the same durability as S3, that is, 119s and security at rest with AES256 for all objects.

Glacier is a cold storage solution meaning that access is not on real time. If data availability is required, consider the use of the Simple Storage Service. Amazon Glacier is a great option to achieve the lowest cost for storing cold data.

Data in Glacier is aggregated via archives. Archives can be anything like a document, video, a ZIP file, and so on. This archives can hold up to a maximum of 40 TB. Archives are stored in vaults and every account has a limit of 1000 vaults.

Information can be retrieved through retrieval jobs. Data archiving can be reinforced with **Vault Lock Policies** that can be used to meet storage retention and compliance, and to achieve **write once read many (WORM)** storage capabilities.

When data is restored, Glacier will copy the object to S3 so you can have on-demand access to the data. If you configured the notification topic you will receive an SNS notification when the data is retrieved. If data stored on Glacier was managed by S3 this data needs to be restored using the S3 APIs, on either case you'll have to retrieve it using the Glacier APIs.

Retrieval options

Amazon Glacier provides three different types of retrieval options that range from a few minutes to hours. The retrieval options available are:

- **Expedited:** This is the fastest retrieval job, customers need to wait 1-5 minutes to have data available and this can be done OnDemand and Provisioned
- **Standard:** This is the default option where the job completes in 3 to 5 hours
- **Bulk:** This job will retrieve any volume of information with the lowest cost, but you will need to wait 5-12 hours

You have the option to only retrieve a subset of a large object using SQL. This feature is called **Glacier Select** where you can read the bytes you need.

Workflow

The following steps depict a typical workflow with Amazon Glacier, from creating the Vault to the job retrieval.

1. The first step is to create a Vault, so navigate to the Glacier console and create a Vault.

Create Vault Step 1: Vault Name Step 2: Event Notifications Step 3: Event Notification Details Step 4: Review	<p>Welcome to Amazon Glacier</p> <p>Data is stored in Amazon Glacier in "archives." An archive can be any data such as a photo, video, or document. You can upload a single file as an archive or aggregate multiple files into a TAR or ZIP file and upload as one archive.</p> <p>A single archive can be as large as 40 terabytes. You can store an unlimited number of archives and an unlimited amount of data in Amazon Glacier. Each archive is assigned a unique archive ID at the time of creation, and the content of the archive is immutable, meaning that after an archive is created it cannot be updated.</p> <p>Vaults allow you to organize your archives and set access policies and notification policies. Get started by giving your vault a name. You can then create your vault now or click Next Step to set up your vault's properties.</p> <p>Region: US East (N. Virginia) </p> <p>Vault Name*: long-term-retention-vault </p> <p>Cancel Next Step</p>
--	--

2. Create a topic or choose an existing one to receive notifications from jobs. Use the CLI to describe the vault.

```
Gabanox-MBP:Desktop gabanox$ aws glacier describe-vault --account-id - --vault-name long-term-retention-vault
{
    "VaultARN": "arn:aws:glacier:us-east-1:11111111111111:vaults/long-term-retention-vault",
    "VaultName": "long-term-retention-vault",
    "CreationDate": "2018-10-14T00:49:09.259Z",
    "NumberOfArchives": 0,
    "SizeInBytes": 0
}
```

3. Alternatively, you can create the vault using the CLI.

```
aws glacier create-vault --account-id - --vault-name long-term-retention-vault
```

4. Let's add a file (in this case I have created `archive.zip`) to our vault with the following command:

```
aws glacier upload-archive --account-id - --vault-name long-term-retention-vault --body archive.zip
```

Note that files added will not be reflected in the vault immediately, inventories get updated once a day so you can build an independent search index to query up to date information.

5. Jobs are the interface from which applications can retrieve data, for example, the vault inventory.

```
aws glacier initiate-job --account-id - --vault-name long-term-retention-vault --job-parameters '{"Type": "inventory-retrieval"}'
```

This operation will return the `InProgress` status code, you can poll until you have the `Succeeded` status. You will receive an SNS email notification too.

6. Now it's time to get the job output:

```
aws glacier get-job-output \
--account-id - \
--vault-name long-term-retention-vault \
--job-id aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa \
outfile job-result.txt
```

The output file contains the archives found in the Vault.

7. Now it's time to retrieve the archive using the archive ID. Another job must be initiated, but this time using the `archive-retrieval` type. The job parameters file can be found here (<http://bit.ly/2ykqtAp>):

```
aws glacier initiate-job \
--account-id - \
--vault-name long-term-retention-vault \
--job-parameters file://archive-retrieval.json
```

This command will return the `job-id` and Vault location, now it's time to retrieve the object.

```
aws glacier get-job-output \
--account-id - \
--vault-name long-term-retention-vault \
--job-id "WWVGEX0hqGbG-W3FHFs4SBjEpQsA0tKuenoEywu0qIcd5441NHUi06R6rTB1jWj1SKV914hE6-FV06aaaaaaaaaaaaaaaa" \
file.tar
```

NoSQL

Applications deal with change constantly and new challenges need to be faced when dealing with social networks, Internet of Things and big data. Relational databases were not designed for massive scalability; they have their own domain where relational databases excel.

The term NoSQL, groups multiple technologies and algorithms focused on distributed computing. What happens when applications need to deal with constant schema changes or high concurrency? NoSQL databases have their strong suite to solve this kind of problems.

DynamoDB

DynamoDB is a fully managed NoSQL database and is part of the AWS offering to non-relational products. This service gives end users all the flexibility required to store unlimited amounts of data, provides strong consistency, low cost, predictable performance with a single digit millisecond latency, and high availability. DynamoDB achieves these capabilities by sharding and scaling the storage and compute capacity. The capacity is provisioned through the API at the **table** and **index** level providing customers with the ability to fine-tune every workload. Reads and writes can have different capacity units assigned via the API.

Control plane

The control plane provides interaction with high-level operations to manage objects as tables and indexes.

In this example, we will create a new table to hold musical artists information that will be queried by artist.

Create DynamoDB table

Tutorial 

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* 

Primary key* Partition key
 
 Add sort key

The AWS CLI could also be used to control the tables management.

- Create table:

```
aws dynamodb create-table --table-name Music --attribute-definitions AttributeName=Artist,AttributeType=S --key-schema AttributeName=Artist,KeyType=HASH --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5
```

- List tables:

```
aws dynamodb list-tables
```

- Describe table:

```
aws dynamodb describe-table --table-name Music
```

```
Gabanox-MBP:~ gabanox$ aws dynamodb describe-table --table-name Music
{
    "Table": {
        "AttributeDefinitions": [
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            }
        ],
        "TableName": "Music",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            }
        ],
        "TableStatus": "ACTIVE",
        "CreationDateTime": 1539926463.103,
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 5
        },
    }
},
```

- Update table:

The update table action is used to add indexes, modify the table throughput and enable trigger style events called **DynamoDB Streams**.

Let's increase 100% of the table capacity.

```
aws dynamodb update-table --table-name Music --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=10
```

- Delete table:

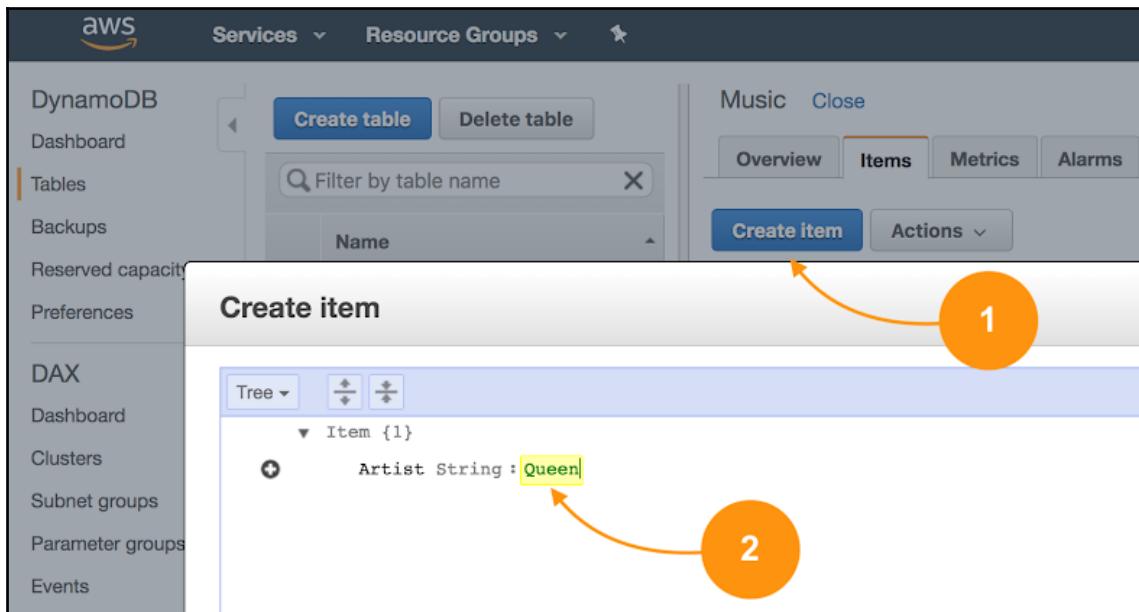
```
aws dynamodb delete-table --table-name Music
```

- Data plane:

One of the most important topics about table design is the key architecture. Think about DynamoDB as a series of servers storing and retrieving data from multiple partitions to increase throughput. Design your tables with a key attribute with the highest cardinality, to promote uniform data distribution across all partitions.

- Simple Primary Key:

The minimum attributes to define a table in DynamoDB is to provide a partition key, this attribute is known as the hash attribute. In order to perform queries, you need to provide the table name and the primary key value to be able to get one particular item.



Once you have created the `Artist` table, choose **Create item**. This will show a popup window where you can introduce the item attributes, in our example, we are choosing the simplest model specifying only the partition key. Add one item with your favorite artist name; add a second one but this time use more attributes like in the following example:

	Artist	BestSoldAlbum	Website 
<input type="checkbox"/>	Led Zeppelin		
<input type="checkbox"/>	Queen		
<input type="checkbox"/>	Pink Floyd	The Dark Side of the Moon	http://www.pinkfloyd.com/

The artist name is a good partition key because every artist name is different and there are many different artists. A bad partition key would be to use the genre where there are few music categories (low ordinality).

Non relational databases are designed to be schemaless which results in huge gains for models that change frequently promoting flexibility in the future. Maybe applications will evolve and a new set of attributes will need to be persisted.

There is no need to alter the physical structure of the tables, you simply put items with the required attributes. Application **data access objects (DAOs)** need to be aware of this evolution and be able to query different types of entities.

- **Scan:** The **Scan** operation will read sequentially all the items in the table up to 1 MB of data. This is the less efficient and costliest operation in DynamoDB. It is a good idea to use it to retrieve catalog data and then cache it somewhere.

- **Query:** Our Music table can be queried in the following way:

The screenshot shows a query interface for a Music table. At the top, it says "Query: [Table] Music: Artist" and "Viewing 1 to 1 items". The main area has sections for "Partition key" (Artist), "String" (set to "="), and a value input field containing "Pink Floyd". Below this is a "Sort" section with radio buttons for "Ascending" (selected) and "Descending". Under "Attributes", there are radio buttons for "All" (selected) and "Projected". A "Start search" button is present. The results table has columns: Artist, BestSoldAlbum, and Website. One row is shown: Pink Floyd, The Dark Side of the Moon, and http://www.pinkfloyd.com/.

In order to perform a query to the Music table, I need to make use of the Artist index, there is no other strategy to query this table. What happens when we have composite keys? Now we have another strategy we can use the Artist name and the Song name; this unique combination will retrieve the specific item in the table. Delete the table and create a new one with the following table specification for the primary key.

- **Composite Primary Key:**

A sorting key let queries to be performed on a subgroup of data that share the same primary key with a unique component which is the range (sort key). This is a 1:M relationship, for example, multiple cities in the same country or multiple songs by the same artist. This data is stored lexicographically. The combination of a **partition key (PK)** and **range key (RK)** must be unique, but queries that only specify the PK will retrieve all the information related to that key ordered. The range key is an internal sorted index of the table.

Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* Music i

Primary key* Partition key

Artist String i

Add sort key

SongTitle String i

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".
- On-Demand Backup and Restore Enabled NEW!

In our new design the `Music` table now has a range key that will be used to find items related with an explicit partition key, in our example, the PK is the British band **Queen**:

Query: [Table] Music: Artist, SongTitle ^				Viewing 1 to 10 items	
Query	[Table] Music: Artist, SongTitle				
Partition key	Artist	String	=	Queen	
Sort key	SongTitle	String	=	Enter value	
+ Add filter					
Sort	<input checked="" type="radio"/> Ascending <input type="radio"/> Descending				
Attributes	<input checked="" type="radio"/> All <input type="radio"/> Projected				
Start search					
	Artist	SongTitle <small>i</small>			
	Queen	Another One Bites the Dust			
	Queen	Bicycle Race			
	Queen	Bohemian Rhapsody			
	Queen	Crazy Little Thing Called Love			
	Queen	Don't Stop Me Now			
	Queen	Fat Bottomed Girls			
	Queen	Killer Queen			
	Queen	Save Me			
	Queen	Somebody to Love			
	Queen	You're My Best Friend			

The **Query** operation specifies the **Queen** string attribute as the partition key, DynamoDB will perform a hash operation to determine the partition where this data is stored $f(x) = P$. The read operation will use the range key to find the associated data with the PK and retrieve the ordered items.

Managed capabilities

Unlike other NoSQL products in the market DynamoDB is fully managed, you never have to deal with provisioning, clustering, or partitioning the tablespace. Indexes in DynamoDB are automatically updated and provisioned throughput is managed at the index and table levels.

DynamoDB can tolerate the concurrent loss of two facilities and maintain its operation, with newest features as DynamoDB Streams and Global tables DynamoDB is an excellent choice for Web, Mobile, IoT, and Big Data applications.

Consistency

DynamoDB guarantees durability and reliability. When a write operation has been made, it has already been replicated in three AZs in less than a second. DynamoDB provides customers with the full control of the consistency model when reading data.

If your application can tolerate inconsistent state you can perform queries with eventually consistent reads, which can return stale data under high concurrency scenarios.

For applications that need to be consistent at all times, you can configure queries to be strongly consistent. As a result, every user will have the same view of data under any circumstance.

This applies for **GetItem**, **Query**, and **Scan** operations, take into account that eventually consistent reads cost half of a strong consistent read.

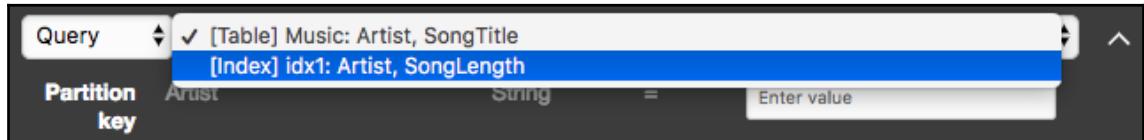
Local secondary index

What happens when our table evolves? Now we have a feature to our uses that provides the ability to choose songs by their length and favorite artist. Local secondary indexes are a way where we can use the partition key and a table attribute to query items. You can have up to 5 LSI per table and they must be defined when creating the table via the `local-secondary-indexes` parameter.

Let's recreate the Music table now with the following definition:

```
aws dynamodb create-table --table-name Music --attribute-definitions
AttributeType=S
AttributeName=SongTitle, AttributeType=S
AttributeName=SongLength, AttributeType=S --key-schema
AttributeName=Artist, KeyType=HASH AttributeName=SongTitle, KeyType=RANGE --
provisioned-throughput ReadCapacityUnits=5, WriteCapacityUnits=5 --local-
secondary-indexes
'IndexName=idx1, KeySchema=[{AttributeName=Artist, KeyType=HASH}, {AttributeNa
me=SongLength, KeyType=RANGE}], Projection={ProjectionType=ALL}'
```

The new table definition now has an index to query by Artist and SongLength. Let's try a simple query. Note how the console now has a new option to query items using the Index:



The end user wants to listen to a song for his bus commute and wants to listen to songs with a length less than five minutes. Note the songs sorted by the `SongLength` property in ascending order:

Query: [Index] idx1: Artist, SongLength ^ Viewing 1 to 9 items

Query [Index] idx1: Artist, SongLength

Partition key Artist String = Queen

Sort key SongLength String < 5:00

Add filter

Sort Ascending

Attributes All

Start search

<input type="checkbox"/>	Artist <small>i</small>	SongTitle	SongLength
<input type="checkbox"/>	Queen	Crazy Little Thing Called Love	2:44
<input type="checkbox"/>	Queen	You're My Best Friend	2:52
<input type="checkbox"/>	Queen	Killer Queen	3:01
<input type="checkbox"/>	Queen	Bicycle Race	3:02
<input type="checkbox"/>	Queen	Fat Bottomed Girls	3:23
<input type="checkbox"/>	Queen	Don't Stop Me Now	3:30
<input type="checkbox"/>	Queen	Another One Bites the Dust	3:35
<input type="checkbox"/>	Queen	Save Me	3:49
<input type="checkbox"/>	Queen	Somebody to Love	4:56

Try the inverse sorting operation with songs longer than five minutes.

The screenshot shows the AWS DynamoDB Query interface. The query is defined as follows:

- Query:** [Index] idx1: Artist, SongLength
- Partition key:** Artist = Queen
- Sort key:** SongLength > 5:00
- Sort:** Descending
- Attributes:** Projected

The results table displays one item:

	Artist	SongTitle	SongLength
<input type="checkbox"/>	Queen	Bohemian Rhapsody	5:51

As information is being added, these items are being projected to the index, this protection can be configured to project only specific attributes enabling data compression and good performance. New data will be added in the future like the `SongPrice`, the genre or the number of times a song has been listened like `ListenCount`.

DynamoDB provides powerful capabilities like atomic counters to implement a `ListenCount`, conditional writes, and collections.

What happens when the limit of 5 LSI has been reached for a table and search strategies where the partition key is not present in the queries?

Global secondary index

The global secondary indexes provide new ways to interact with our tables. GSIs can use alternate keys different from the PK. For example, a GSI can be used to listen to the most popular songs by genre. Navigate to the `Indexes` tab of the selected `Music` table and create an index as follows:

Create index

Primary key* Partition key
Genre String 

Add sort key
ListenCount String 

Index name* Genre-ListenCount-index 

Projected attributes All 

Read capacity units Write capacity units
5 5

Estimated cost \$2.91 / month ([Capacity calculator](#))

Approximate creation time is **5 minutes**. Additional write capacity may decrease creation time. A notification will be sent to the SNS topic dynamodb once the index creation is complete. Basic Alarms with 80% upper threshold using SNS topic 'dynamodb' will be automatically created. Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced configuration for alarms can be done in the alarms tab.

[Cancel](#) **Create index**

Note that the GSI has its own **read capacity units (RCU)** and **write capacity units (WCUs)**, these indexes act as a dependent table with the specified attributes projected automatically by the index. Once the index is in the **Active** state perform a **Scan** operation to see its contents. As you may note there are no items to show, this is because once the index is created Dynamo will start projecting new and updated data maintaining consistency between the **Music** table and the GSI.

Try adding some new records to the **Music** table and perform a query to the **Genre-ListenCount-index**:

The screenshot shows the AWS DynamoDB console interface. At the top, it says "Query: [Index] Genre-ListenCount-index: Genre, ListenCo..." and "Viewing 1 to 3 items". Below this is a search bar with the query "[Index] Genre-ListenCount-index: Genre, ListenCount". The search parameters are set to "Partition key" as "Genre" (String) = "Classic rock" and "Sort key" as "ListenCount" (String) > "100000". There is also an "Add filter" button and a "Sort" section with "Ascending" selected. A "Start search" button is at the bottom of the search area. The main part of the screen is a table with columns: Artist, SongTitle, Genre, ListenCount, and SongLength. The data rows are:

Artist	SongTitle	Genre	ListenCount	SongLength
Queen	Under Pressure	Classic rock	23000000	225
Queen	Radio Ga Ga	Classic rock	23000000	180
Queen	Love of My Life	Classic rock	30000000	216

DynamoDB Streams

Our application needs to send a confirmation email whenever a new user is put in the **Users** table, we want to avoid having **glue code** to make this happen. DynamoDB streams are an event-based notification system that can resemble traditional database triggers. Lambda functions can poll for changes in the DynamoDB stream which once activated for a table provides a service endpoint where all the ordered table changes are recorded and persisted up to 24 hours.

Global tables

Let's think about the global scale, our streaming application now adds real-time support and must be consistent globe wide accepting writes from everywhere at any time. Global tables are a managed solution to have multi-master architectures across multiple regions. DynamoDB streams are used to replicate changes and a control table to keep the latest write records to solve write conflicts.

Summary

In this chapter, we presented an overview of the core storage services like RDS, multi-AZ deployments, and read replicas. We took a deep dive on object storage with Amazon S3 and Glacier. You worked on a full workflow with glacier and used different storage types in S3 with the CLI and with lifecycle policies.

To end this module we designed a data model using a DynamoDB table and understood the differences between **Scan** and **Query** operations and local and secondary indexes.

Further reading

- **Object Lifecycle Management:** <https://docs.aws.amazon.com/AmazonS3/latest/dev/object-lifecycle-mgmt.html#lifecycle-transition-general-considerations>
- **Working with Storage:** https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PIOPS.StorageTypes.html
- **Supported Data Types:** <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DynamoDBMapper.DataTypes.html>
- **Best Practices for DynamoDB:** <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-practices.html>

10

Matching Supply and Demand

Reliability is the characteristic of a system to acquire compute resources when patterns in traffic change and the ability to acquire resources dynamically to match the demand. Services like Auto Scaling and Elastic Load Balancing are designed to provide such capabilities to compensate capacity as needed and scaling into lower costs.

Every company and application is different so proactive and reactive mechanisms that can adapt to organic traffic must be used or even schedule anticipated increases on demand.

The following topics will be covered in this chapter:

- Elastic Load Balancing and types
- ELB attributes
- AWS Auto Scaling

Technical requirements

There are no technical requirements for this chapter.

Elastic Load Balancing

Elastic Load Balancing provides applications with a reverse proxy service that hides and encapsulates direct access to EC2 instances. Load balancers are a great way to decouple applications by introducing indirection levels with managed services that provide capabilities like multi-AZ request distribution, security, and SSL offloading from EC2 fleets.

The Elastic Load Balancer is a fully managed service designed to scale horizontally with high availability. Applications can rely on load balancers to receive proxied traffic through TCP, HTTP/HTTPS, and SSL protocols.

Working with the service consists of provisioning an ELB, configuring its attributes, and registering targets. The ELB performs frequent health checks to validate the availability of the targets to handle traffic. When instances pass HTTP health checks successfully the ELB performs a weighted distribution algorithm called **least outstanding request** favoring the most performant instances, when working over TCP the ELB does a round robin distribution.

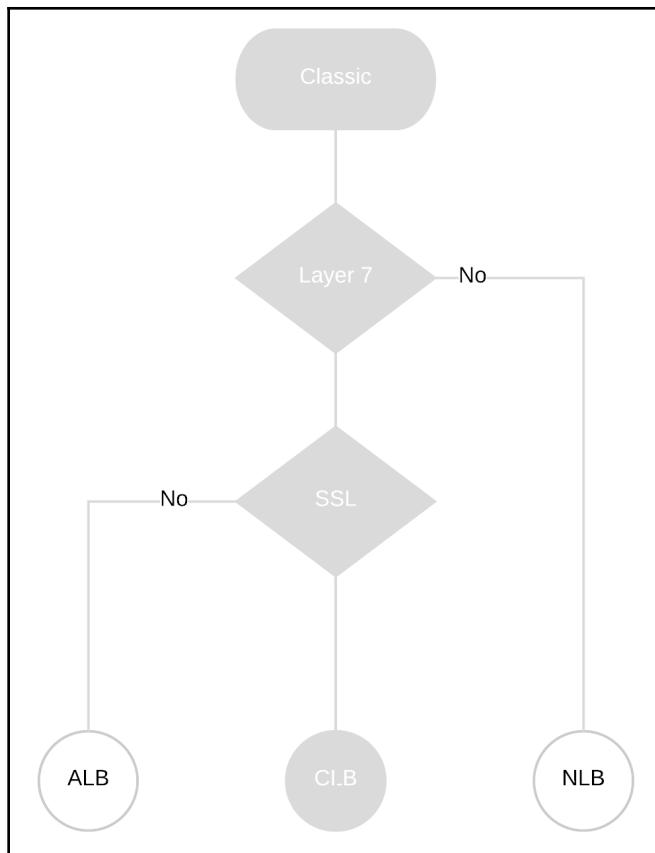
Designing application's fronted with load balancers take away the necessity to expose public IPs externally and the removal of single points of failure form the service layers promoting robustness, scalability, and reliability.

The Elastic Load Balancing service provides with three different load balancers for the general and specific purpose.

Classic Load Balancer – CLB

This has been the de facto solution for load balancing and is backward compatible with the EC2—classic networking mode. This is an **Open Systems Interconnection model (OSI model)** level 7 load balancer capable to handle communications with HTTP/HTTPS, TCP, and SSL. The CLB can map different ports for the listener and the destination target. For example an HTTPS listener on port 443 and a target with HTTP on port 80.

The following flow diagram will help you to decide when to use **Classic Load Balancer (CLB)**:

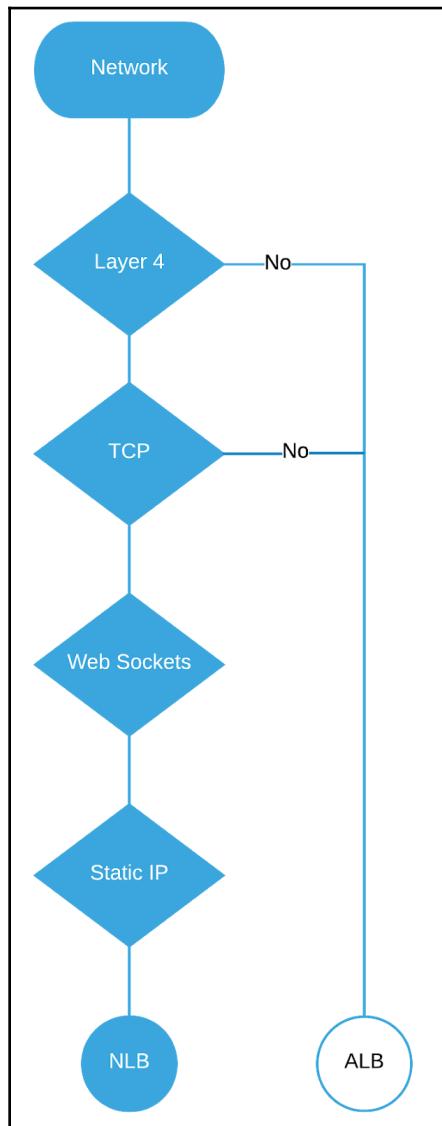


This chart is not comprehensive and there are characteristics that are shared between load balancers.

Network Load Balancer – NLB

Network Load Balancers work with TCP traffic only and they support dynamic host configuration which makes them a good choice for containers by having multiple tasks running in the same container instance. NLB works at layer 4 with the ability to scale up to millions of requests per second and they can be compliant with the WebSockets protocol.

You have the option to associate an Elastic IP for AZ with the Network Load Balancer which provides a good alternative for firewalls whitelisting.





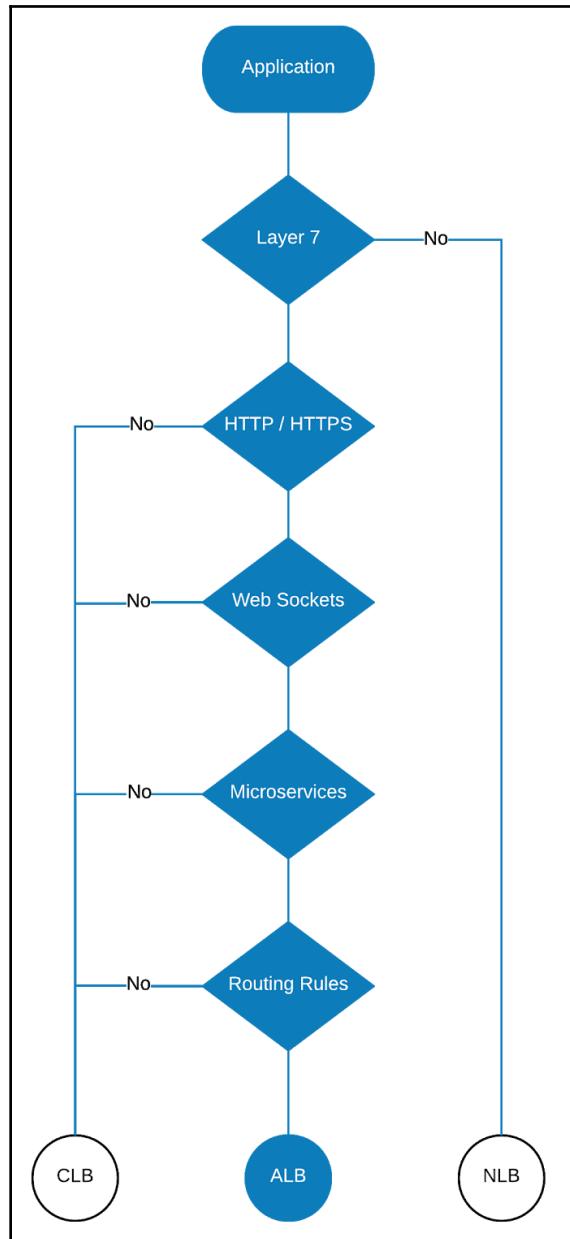
This chart is not comprehensive and there are characteristics that are shared between load balancers.

Application Load Balancer – ALB

Application Load Balancing is the developers friendly balancing service because it understands requests and performs routing logic. It is possible to decompose monolithic applications and define independent target groups based on:

- Protocol (for example, one target group for HTTPS and other for HTTP)
- Path (for example `admin.crm-app.net`, `payments.crm-app.net`, or `/img/*`)
- Port (for example 8080 for frontend and 8081 for backend)

Application Load Balancers highly compatible with microservices architectures.





This chart is not comprehensive and there are characteristics that are shared between load balancers.

Creating an Application Load Balancer

In the following guide we will provision an Application Load Balancer that will be integrated with the Auto Scaling service:

1. Navigate to the EC2 console, select **Load Balancers** and **Create Load Balancer** (1) as shown in the following screenshot:

A screenshot of the AWS EC2 console under the 'LOAD BALANCING' section. The 'Load Balancers' option is selected. At the top, there is a 'Create Load Balancer' button with a red circle around it. Below the button, a search bar and several filter options are visible. The main area displays a message: 'You do not have any load balancers in this region.' There is also a 'Select a load balancer' dropdown menu.

Here, you will be prompted to choose for a load balancer type, use the **Application Load Balancer**:

 A screenshot of a modal dialog titled 'Select load balancer type'. It explains that Elastic Load Balancing supports three types of load balancers: Application Load Balancers, Network Load Balancers (new), and Classic Load Balancers. It prompts the user to choose the type that meets their needs. Three options are shown:

- Application Load Balancer**: Handles HTTP and HTTPS traffic. It includes a 'Create' button and a detailed description: 'Choose an Application Load Balancer when you need a flexible feature set for your web applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.' A 'Learn more >' link is also present.
- Network Load Balancer**: Handles TCP traffic. It includes a 'Create' button and a detailed description: 'Choose a Network Load Balancer when you need ultra-high performance and static IP addresses for your application. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second while maintaining ultra-low latencies.' A 'Learn more >' link is also present.
- Classic Load Balancer**: A 'PREVIOUS GENERATION' option for HTTP, HTTPS, and TCP. It includes a 'Create' button and a detailed description: 'Choose a Classic Load Balancer when you have an existing application running in the EC2-Classic network.' A 'Learn more >' link is also present.

2. Now we will proceed to configure our load balancer by specifying the listeners, for this example use the HTTP listener. Secure (HTTPS) listeners will be required to create or import an SSL certificate to be used in the load balancer.
3. To provide full high availability and resiliency the load balancer needs to be associated with at least two AZs which can be public or private with a preference for the later.

The screenshot shows the AWS Step 1: Configure Load Balancer interface. At the top, there are tabs: 1. Configure Load Balancer, 2. Configure Security Settings, 3. Configure Security Groups, 4. Configure Routing, 5. Register Targets, and 6. Review. The first tab is selected. Below the tabs, the title "Step 1: Configure Load Balancer" is displayed, followed by "Listeners". A note states: "A listener is a process that checks for connection requests, using the protocol and port that you configured." A "Load Balancer Protocol" dropdown is set to "HTTP" and a "Load Balancer Port" input field contains "80". A "Add listener" button is visible. The next section is "Availability Zones", with a note: "Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. You can specify only one subnet per Availability Zone. You must specify subnets from at least two Availability Zones to increase the availability of your load balancer." A "VPC" dropdown is set to "vpc-026b3e6 (10.200.0.0/20)" and a "Select a subnet..." dropdown is set to "VPC". A table lists subnets across two Availability Zones:

Availability Zone	Subnet ID	Subnet IPv4 CIDR	Name
us-west-1a	subnet-0f137c	10.200.2.0/23	Private Subnet 1
us-west-1a	subnet-002560	10.200.0.0/24	Public Subnet 1
us-west-1b	subnet-01d9ae9	10.200.4.0/23	Private Subnet 2
us-west-1b	subnet-041a907	10.200.1.0/24	Public Subnet 2

Below the table, a "Tags" section is shown with a plus sign icon. At the bottom right, there are "Cancel" and "Next: Configure Security Settings" buttons.

4. Ignore the following warning about not using a secure listener for now:

Step 2: Configure Security Settings

⚠ Improve your load balancer's security. Your load balancer is not using any secure listener.
If your traffic to the load balancer needs to be secure, use the HTTPS protocol for your front-end connection. You can go back to the first step to add/configure secure listeners under [Basic Configuration](#) section. You can also continue with current settings.

Cancel Previous Next: Configure Security Groups

5. The load balancer needs to be associated with a security group, and it makes sense if we use the security groups chaining pattern to compartmentalize access to EC2 instances. As this is an internet facing load balancer open the HTTP 80 port; you can create the security group here or use an existing one:

Step 3: Configure Security Groups

A security group is a set of firewall rules that control the traffic to your load balancer. On this page, you can add rules to allow specific traffic to reach your load balancer. First, decide whether to create a new security group or select an existing one.

Assign a security group: Create a new security group
 Select an existing security group

Security group name: Web Server Security Group

Description: Web Server Security Group

Type	Protocol	Port Range	Source
HTTP	TCP	80	Custom 0.0.0.0/0, ::/0

Add Rule Cancel Previous Next: Configure Routing

6. It is time to create the routing scheme, create a new target group for this load balancer and name it `WebTargetGroup`, this target group will listen on the 80 port and send traffic over the port number 80 to instances.

7. We want to fail fast on nonresponsive instances to update the unhealthy threshold with a value of 2 and a 5 seconds timeout that will perform health checking every 10 seconds expecting an HTTP 200 code:

The screenshot shows the AWS CloudFormation console with the 'Step 4: Configure Routing' page for a new target group. The target group is named 'WebTargetGroup' and uses the 'HTTP' protocol on port 80, targeting 'instance' type targets. Under 'Health checks', the 'Protocol' is set to 'HTTP' and the 'Path' is '/'. Advanced health check settings are configured with a healthy threshold of 2, an unhealthy threshold of 2, a timeout of 5 seconds, an interval of 10 seconds, and success codes of 200. The status bar at the bottom right shows 'Cancel', 'Previous', and 'Next: Register Targets'.

8. If we had previously running instances, in the **Next:Register Targets** screen you can manually add instances to be associated with the load balancer; this is not the case right now so skip to the next screen and create the load balancer.

The screenshot shows the 'Load Balancer Creation Status' screen. It displays a green message box indicating that the load balancer 'app-elb' was successfully created. A note states that it might take a few minutes for the load balancer to be fully set up. A 'Close' button is located in the bottom right corner.

9. When the state of the load balancer becomes **active** you will have access to the **DNS name** provided, copy the DNS value because you will use it later.

Name	DNS name	State	VPC ID	Availability Zones	Type	Created At
app-elb	app-elb-	active	vpc-026b3	us-west-1a, us-west-1b	application	October 25, 2018 at 8:45:52 ...

We have now provisioned an Application Load Balancer and created the routing scheme for the backend web applications. Furthermore, you can change some of the ELB attributes and observe different behaviors.

ELB attributes

We will go through the ELB attributes in the following sections.

Stateless versus stateful

Robust scalable and fault-tolerant architectures are designed using stateless servers: servers which do not handle any user state like shopping carts, state between pages and authentication tokens. Services like DynamoDB and ElastiCache are great options to offload temporary session data to an external service, so when applications fail they do it gracefully.

Stateful applications, on the other hand, do manage conversational state with the end user and are not fault tolerant, for these legacy applications you can enable sticky sessions which are a good way to provide session affinity with the persistent server; once this attribute is enabled the load balancer will send a session cookie named AWSELB to route all future requests to the same server.



For more information about cookies use the following links to the ELB documentation:

<https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-sticky-sessions.html>

<https://aws.amazon.com/es/blogs/aws/new-elastic-load-balancing-feature-sticky-sessions/>.

Internet-facing versus internal-facing

An internet-facing load balancer will listen for requests coming from the internet and an internal-facing load balancer will only route requests from private IP addresses across the VPC.

TCP passthrough

This connection mode will take the request and pass it through as is without adding any additional header information, this is a good fit for SSL certificates that are out of the scope of the ELB, for example, CloudFront incoming requests.

Cross-zone load balancing

The cross-zone load balancing is an attribute that when enabled will balance the number of instances evenly across AZs.

Connection draining

Is a graceful method to remove in-service instances that have in-flight requests so end users have a good experience when rolling out new versions of code or scale out activities are performed.

AWS Auto Scaling

Create an EC2 instance, this instance will be used as a baseline to create a web server AMI, you can use the default VPC to make this process straightforward:

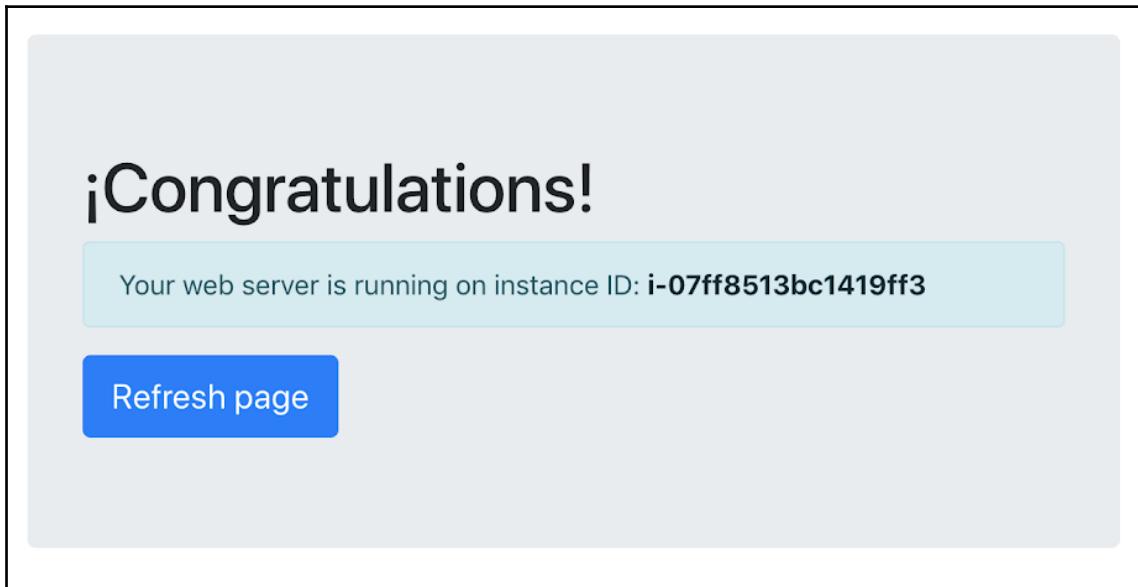
1. Navigate to EC2 and create an instance choosing the Amazon Linux 2 AMI, you can name it `Web Server Base`; make sure to create an SSH key and open SSH traffic via the port 22 to be able to log into the server when it becomes available. Copy the public IP as you will need it in the next steps.

Name	Instance ID	Instance Type	Availability Zone	Instance State
Web Server Base	i-07ff8513bc1419ff3	t2.micro	us-west-2b	running

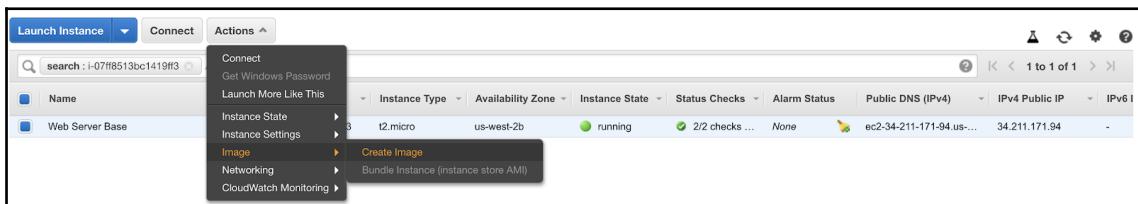
2. Using the public IP, `ssh` into the server. If you have questions about the process, check the first chapter or read the instructions for *Connecting to Your Linux Instance Using SSH* using the following link: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html>
3. Now execute the following commands in order to install the Apache web server and download the sample code for our demo web app:

```
sudo yum -y update
sudo yum -y install httpd php
sudo chkconfig httpd on
sudo wget
https://s3.amazonaws.com/aws-certified-solution-architect-associate-guide/index.php -O /var/www/html/index.php
sudo service httpd start
```

4. Open a web browser and paste the public IP copied from the Web Server Base instance, You should see the following web page:



5. Now it's time to create an AMI with this server, select the Web Server Base instance and use the **Create Image** sub-option as shown. From now on we will be using this new image to configure Auto Scaling.



6. On the **Create Image** screen, use a friendly name for the new image like **Web Server AMI** and click on **Create Image**:

The screenshot shows the 'Create Image' dialog box. At the top, there are fields for 'Instance ID' (i-07ff8513bc1419ff3), 'Image name' (Web Server AMI), 'Image description' (Web Server AMI), and a 'No reboot' checkbox which is unchecked. Below this is a section titled 'Instance Volumes' with a table. The table has columns: Volume Type, Device, Snapshot, Size (GiB), Volume Type, IOPS, Throughput (MB/s), Delete on Termination, and Encrypted. A single row is shown for the 'Root' volume: /dev/xvda, snap-0c7860dff2c9748dd, 8 GiB, General Purpose SSD (gp2), 100 / 3000 MB/s, N/A, checked, and Not Encrypted. There is a 'Cancel' button and a prominent blue 'Create Image' button at the bottom right.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted
Root	/dev/xvda	snap-0c7860dff2c9748dd	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

7. When the image status changes to **available**, select the new **Web Server AMI** image and use the **Launch** button. Now we will create instances from this image. Use a **t2.micro** or similar instance for the web server.

The screenshot shows the 'AMIs' page in the AWS Management Console. The 'Actions' dropdown is set to 'Launch'. A search bar shows 'search : ami-0bc5396efccafdd4a'. The table lists one item: 'Web Server AMI' with 'Name' 'Web Server AMI', 'AMI ID' 'ami-0bc5396efccafdd4a', 'Source' '...', 'Owner' 'Private', 'Visibility' 'Private', and 'Status' 'available'.

Name	AMI Name	AMI ID	Source	Owner	Visibility	Status
Web Server AMI	Web Server AMI	ami-0bc5396efccafdd4a	...	Private	Private	available

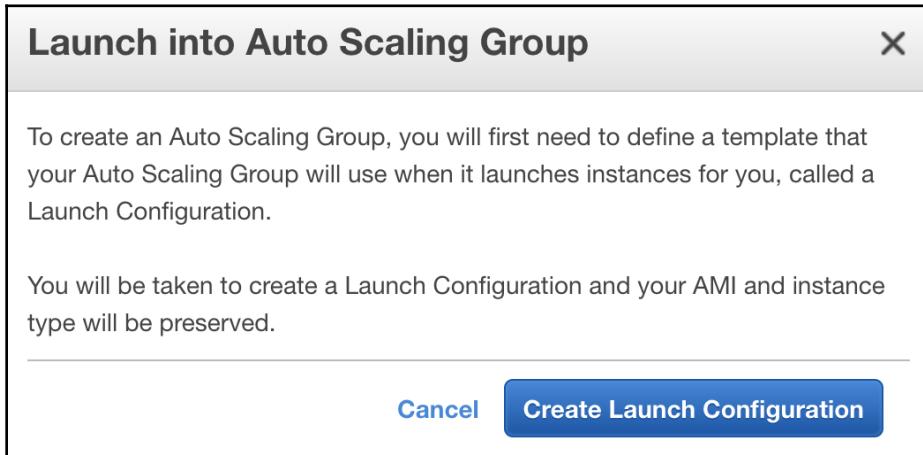
8. In the following **Configuring Instance** tab, use the VPC of your choice (default VPC can be used for this exercise). This step provides the option to launch the instance into an Auto Scaling group as shown in (1). Use this option.

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing.

Number of instances	1	Launch into Auto Scaling Group
Purchasing option	<input type="checkbox"/> Request Spot instances	
Network	vpc-d29adbab (default)	Create new VPC
Subnet	No preference (default subnet in any Availability Zone)	Create new subnet
Auto-assign Public IP	Use subnet setting (Enable)	

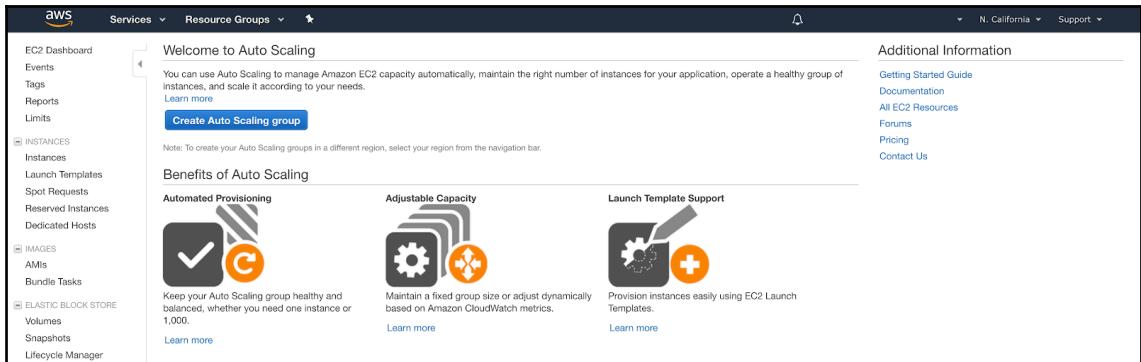
9. You will be presented with the wizard to create the Auto Scaling configuration.



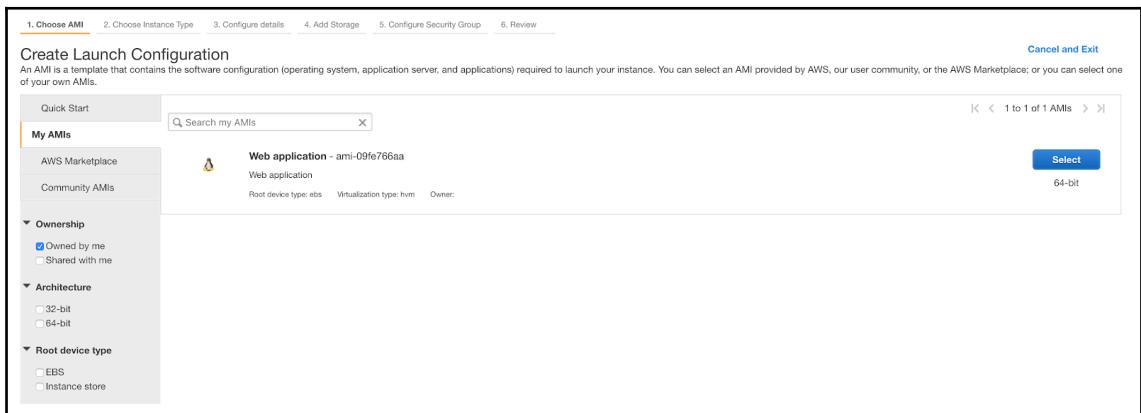
10. Choose **Create Launch Configuration** and you will be redirected to the Auto Scaling wizard with the AMI pre-selected.

Alternate flow

An alternate flow is to navigate to EC2 | Auto Scaling. Here you will be presented with the following welcome screen:



Choose **Create Auto Scaling group** and choose the **Web Server AMI** image under **My AMIs**:



Create a launch configuration

Both flows will require to create a launch configuration. Launch configurations are the way to let Auto Scaling know what to launch specifying details like:

- AMI
- Instance type
- IAM role
- User data
- Security groups

We won't add any additional storage, hence, click **Next** until **Configure Security Group** tab:

1. Choose AMI 2. Choose Instance Type 3. Configure details 4. Add Storage 5. Configure Security Group 6. Review

Create Launch Configuration

Name (i)

Purchasing option (i) Request Spot Instances

IAM role (i)

Monitoring (i) Enable CloudWatch detailed monitoring
[Learn more](#)

▶ Advanced Details

💬 Later, if you want to use a different launch configuration, you can create a new one and apply it to any Auto Scaling group. Existing launch configurations cannot be edited.

Cancel Previous Skip to review Next: Add Storage

Instances managed via Auto Scaling will require a security group in place to allow relevant traffic. You can create a new security group or choose an existing one, make sure to allow HTTP and SSH ingress traffic.

The screenshot shows the 'Configure Security Group' step of the AWS Create Launch Configuration wizard. At the top, there are tabs: 1. Choose AMI, 2. Choose Instance Type, 3. Configure details, 4. Add Storage, 5. Configure Security Group (which is selected), and 6. Review.

Create Launch Configuration

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:

- Create a new security group
- Select an existing security group

Security Group ID	Name	VPC ID	Description	Actions
sg-0b74abd2	default	vpc-026b3e	default VPC security group	Copy to new
sg-07ee1477	Web Server Security Group	vpc-026b3e	Web Server Security Group	Copy to new

Inbound rules for sg-0da44197 Selected security groups: sg-07ee1477, sg-0da44197

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0
SSH	TCP	22	0.0.0.0/0

[Cancel](#) [Previous](#) [Review](#)

Click to review, in this final step you will be prompted to download a key pair to use with this managed instances. Use or create a new one and finish with **Create launch configuration**.

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Key pair name
AutoScaling

Tip: You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

[Cancel](#) [Create launch configuration](#)

Up to this point, we have created the launch configuration for the Auto Scaling group.

Right after creating the launch configuration you will be directed to create the Auto Scaling group. If you leave the wizard you can navigate to EC2 | **Auto Scaling groups** and choose **Create Auto Scaling group** and select the previous launch configuration created.

Auto Scaling groups

The Auto Scaling group serves as a logical grouping for EC2 instances. Auto Scaling takes the responsibility to maintain the minimum capacity specified in the group and to provide elasticity to EC2 by executing policies driven by CloudWatch alarms or from scheduled actions.

When traffic is organic, customers can specify comprehensive metrics and alarms based on usage patterns. In the following example, we will define the capacity limits of this EC2 compute layer to guarantee to have two instances running at all time.

Continuing with the Auto Scaling wizard use a friendly name, adjust the minimum capacity to be two instances and use the available subnets; they can be public or private. Private subnets are preferred because they isolate the web tier to end users minimizing unnecessary exposure to resources externally.

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review [Cancel and Exit](#)

Create Auto Scaling Group

Launch Configuration [i](#) WebApp Launch Configuration

Group name [i](#) Web application Auto Scaling Group

Group size [i](#) Start with instances

Network [i](#) [C](#) Create new VPC

Subnet [i](#) [X](#)
Subnet 1 | us-west-1a
 [X](#)
Subnet 2 | us-west-1b

[Create new subnet](#)

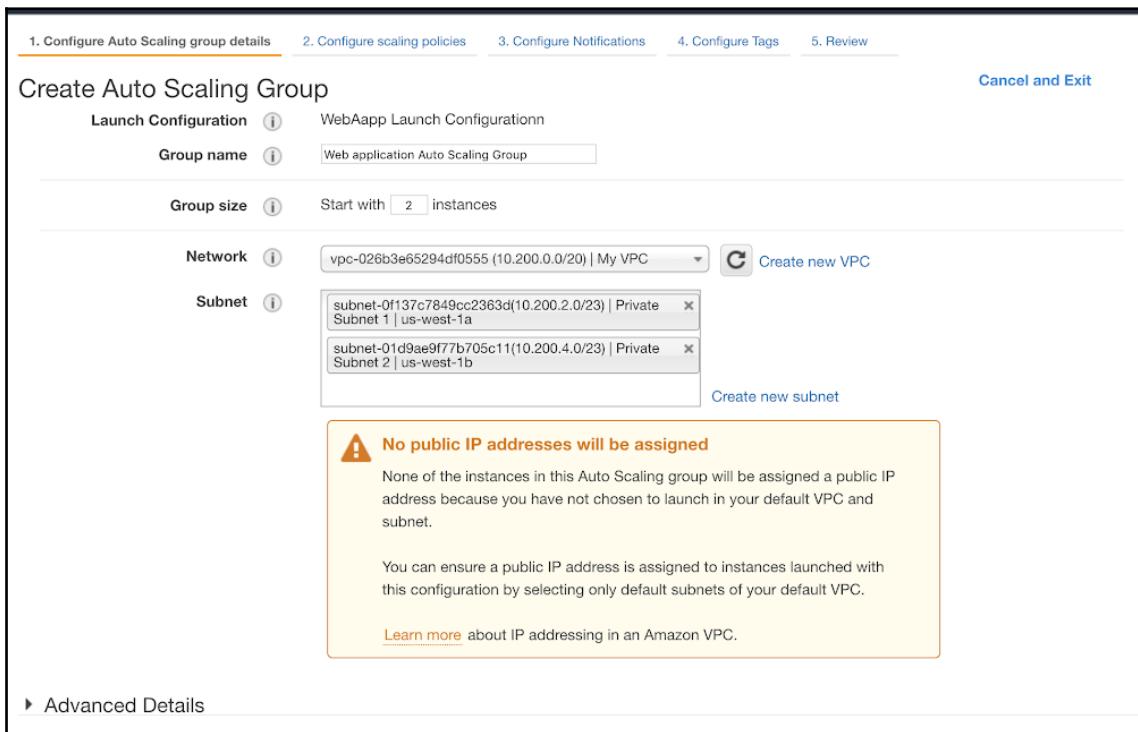
⚠ **No public IP addresses will be assigned**

None of the instances in this Auto Scaling group will be assigned a public IP address because you have not chosen to launch in your default VPC and subnet.

You can ensure a public IP address is assigned to instances launched with this configuration by selecting only default subnets of your default VPC.

[Learn more](#) about IP addressing in an Amazon VPC.

► Advanced Details



Expand the **Advanced Details** section and specify the ALB Target Groups created for this application; in this case the **WebTierTargetGroup**. Proceed with the next step.

▼ Advanced Details

Load Balancing	<input checked="" type="checkbox"/> Receive traffic from one or more load balancers	Learn about Elastic Load Balancing
Classic Load Balancers	<input type="text"/>	
Target Groups	<input type="text"/> WebTierTargetGroup X	
Health Check Type	<input checked="" type="radio"/> ELB <input type="radio"/> EC2	
Health Check Grace Period	<input type="text"/> 300	seconds
Monitoring	Amazon EC2 Detailed Monitoring metrics, which are provided at 1 minute frequency, are not enabled for the launch configuration WebApp Launch Configuration. Instances launched from it will use Basic Monitoring metrics, provided at 5 minute frequency. Learn more	
Instance Protection	<input type="text"/>	
Service-Linked Role	<input type="text"/> AWSServiceRoleForAutoScaling	C
The default role does not exist. It will be automatically created on your behalf.		

In the **Configure scaling policies** section, we can specify CloudWatch alarms that initiate scaling policies based on monitoring and to keep things simple use the **Keep this group at its initial size** option.

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

Create Auto Scaling Group

You can specify any scaling policies if you want to adjust the size (number of instances) of your group automatically. A scaling policy is a set of instructions for making such adjustments in response to an Amazon CloudWatch alarm that you assign to it. In each policy, you can choose to add or remove a specific number of instances or a percentage of the existing group size, or you can set the group to an exact size. When the alarm triggers, it will execute the policy and adjust the size of your group accordingly. [Learn more about scaling policies.](#)

Keep this group at its initial size
 Use scaling policies to adjust the capacity of this group

You have the option to tag these resources and it is a good practice. On the review screen confirm the provided values and click **Create Auto Scaling group**.

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

Create Auto Scaling Group

Please review your Auto Scaling group details. You can go back to edit changes for each section. Click **Create Auto Scaling group** to complete the creation of an Auto Scaling group.

Auto Scaling Group Details

Group name	Web application Auto Scaling Group
Group size	2
Minimum Group Size	2
Maximum Group Size	2
Subnet(s)	subnet-01 , subnet-0f1
Load Balancers	
Target Groups	WebTargetGroup
Health Check Type	ELB
Health Check Grace Period	300
Detailed Monitoring	No
Instance Protection	None
Service-Linked Role	AWSServiceRoleForAutoScaling

Scaling Policies

Notifications

Tags

[Edit details](#) [Edit scaling policies](#) [Edit notifications](#) [Edit tags](#)

[Cancel](#) [Previous](#) [Create Auto Scaling group](#)

When you create this group it will be automatically selected, inspect the **Instances** tab. Here the ALB associated instances are auto referenced and managed from this Auto Scaling group. Observe the **Lifecycle** column in the following screenshot, instances are in the **Pending** state, before they get attached to the Auto Scaling group:

Details	Activity History	Scaling Policies	Instances	Monitoring	Notifications	Tags	Scheduled Actions	Lifecycle Hooks
Actions								
Filter: Any Health Status ▾	Any Lifecycle State ▾	Filter instances...						
i-0abbe3f2	Pending	WebApp Launch Configuration	us-west-1b	Healthy				
i-0af734b3	Pending	WebApp Launch Configuration	us-west-1a	Healthy				

Perform a refresh until instances transition to the **InService** lifecycle state.

The screenshot shows the AWS Auto Scaling Groups console. At the top, there's a search bar and a navigation bar with tabs: Details, Activity History, Scaling Policies, Instances, Monitoring, Notifications, Tags, Scheduled Actions, and Lifecycle Hooks. The 'Instances' tab is selected. Below the tabs, there's a table with columns: Instance ID, Lifecycle, Launch Configuration Name, Availability Zone, Health Status, and Protected From. Two instances are listed: i-0abbe3f21b (InService, WebApp Launch Configuration, us-west-1b, Healthy, Not Protected) and i-0af734b307 (InService, WebApp Launch Configuration, us-west-1a, Healthy, Not Protected).

Now instances have become managed from this Auto Scaling group. Up next we will simulate an instance failure by terminating one managed instance.

Resiliency

Systems must be designed to degrade gracefully and provide recoverability automatically. To test out this assumption we want to inject failure to this layer.

1. Choose an instance of your choice and terminate it.

The screenshot shows the AWS Instances console. A context menu is open over an instance named 'WebServer'. The menu has sections: 'Connect' (disabled), 'Get Windows Password' (disabled), 'Launch More Like This', and 'Actions'. Under 'Actions', there are options: 'Start', 'Stop', 'Reboot', and 'Terminate'. The 'Terminate' option is highlighted. The main table lists two instances: 'WebServer' (selected) and another 'WebServer' entry. Columns include Name, owner, Instance ID, Instance Type, Availability Zone, Instance State, Launch Time, and Security Groups.

While the **Instance State** is **shutting-down**, the instance is no longer managed by the Auto Scaling group.

The screenshot shows the AWS Instances console after the instance has been terminated. The table now shows two rows: one for the terminated instance (Instance State: shutting-down) and one for the surviving instance (Instance State: running). The terminated instance's row is faded, indicating it is no longer active.

2. The ELB performs a graceful degradation of the service also by draining all in-flight connections. Soon Auto Scaling will detect a change in capacity and compensate this difference by launching a new one reading the associated launch configuration. Historical information for this Auto Scaling group is provided in the **Activity History** tab as shown in the following screenshot:

The screenshot shows the AWS EC2 Dashboard with the 'Auto Scaling Groups' section selected. In the main content area, the 'Activity History' tab is active. The table displays the following data:

Status	Description	Start Time	End Time
Waiting for ELB connection draining	Terminating EC2 instance: i-0af734b30	2018 October 25 09:25:22 UTC-5	
Successful	Launching a new EC2 instance: i-0af73	2018 October 25 09:05:01 UTC-5	2018 October 25 09:05:34 UTC-5
Successful	Launching a new EC2 instance: i-0abb	2018 October 25 09:05:01 UTC-5	2018 October 25 09:05:33 UTC-5

3. We can confirm the recent actions by looking at the **Instances** tab; the previous instance is **Terminating** state and the new one is in **Pending** state, as shown in the following screenshot:

The screenshot shows the AWS EC2 Dashboard with the 'Instances' tab selected. The table displays the following data:

Instance ID	Lifecycle	Launch Configuration Name	Availability Zone	Health Status	Protected from
i-07d76e0d	Pending	WebApp Launch Configurationn	us-west-1a	Healthy	
i-0abbe3f2	InService	WebApp Launch Configurationn	us-west-1b	Healthy	
i-0af734b3	Terminating	WebApp Launch Configurationn	us-west-1a	Unhealthy	

The previous information is consistent with the EC2 instances list.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Launch Time	Security Groups
WebServer	i-07d	t2.micro	us-west-1a	running	October 25, 2018 at 9:25:55 ...	ConfigServerSG...
WebServer	i-0ab	t2.micro	us-west-1b	running	October 25, 2018 at 9:05:00 ...	ConfigServerSG...
WebServer	i-0af7	t2.micro	us-west-1a	terminated	October 25, 2018 at 9:05:00 ...	

To enhance your knowledge about Auto Scaling and improve the reliability of the overall system, try to configure CloudWatch to keep proactive monitoring for your instances.



For more information read the monitoring features in the Auto Scaling user guide: <https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-monitoring-features.html>.

Play around a little bit with the sticky sessions attribute and observe the affinity to only one server after each refresh of the web page; disable the attribute and again you will be permuted with both web servers.

Summary

In this chapter, you have implemented a full end to end solution that provides resiliency and scalability to the web server tier using AMIs Auto Scaling groups and Elastic Load Balancing by provisioning and configuring an application load balancer.

We discussed different types of load balancers available and the benefits and tradeoffs of each one. A mention of some of the most relevant and day to day use attributes available like cross-zone load balancing and connection draining.

With Auto Scaling we created a logical grouping of instances that is designed to provide a minimum capacity at all times and tested by the injection of failure to observe how the system performs auto-healing.

Further reading

- **Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch:** <https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-instance-monitoring.html>
- **Dynamic Scaling for Amazon EC2 Auto Scaling:** <https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-scale-based-on-demand.html>
- **Amazon EC2 Auto Scaling Lifecycle Hooks:** <https://docs.aws.amazon.com/autoscaling/ec2/userguide/lifecycle-hooks.html>

11

Introducing Amazon Elastic MapReduce

The volume of data created by mankind is increasing massively. In the last two years, we have created more data than in the previous history of the human race—unstructured data grows every second. This is why new paradigms must be used to properly manage it.

The term **big data** is used more and more frequently, but what exactly is big data? How big is big data? It all depends on the perspective. Imagine a small company that works with spreadsheets accumulating data every year to the point where this tool is no longer useful. The company needs a new strategy such as relational databases and ERP software.

This same analogy works for big companies. Big data is using non-traditional methods to analyze vast amounts of information from different sources and types. Latency plays an important role in the big-data pipeline, because, depending on the velocity at which data is ingested, transformed, enriched, and processed, the tools employed will be different.

This chapter provides an introduction to Amazon Elastic MapReduce and some complementary tools and concepts about clustering and network performance.

The following topics will be covered in this chapter:

- Clustering in AWS
- Placement groups
- Elastic MapReduce

Technical requirements

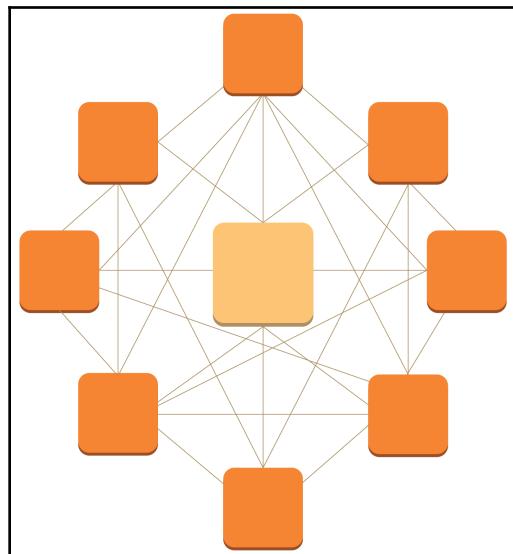
You will need access to the CLI, Python 2.6.5 or higher, an IAM user with sufficient permissions to create roles, EC2 instances, and related resources. An AdministratorAccess policy can be used.

Clustering in AWS

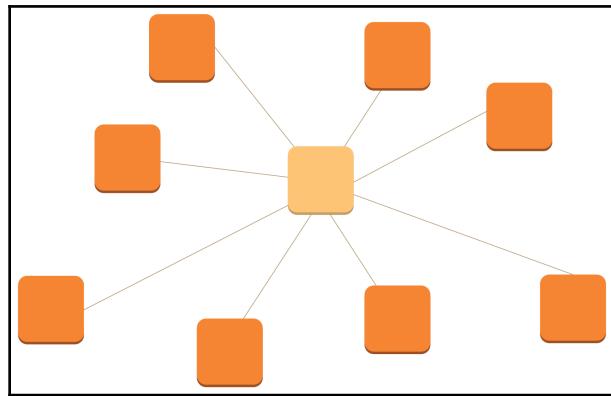
Clustering is a way to group the compute resources physically. The nearest the better improving the communications performance and lowering jitter. Clusters can be tightly or loosely coupled and have a master node that performs all the orchestration activities of the compute nodes. Every cluster in AWS is a single **Availability Zones (AZ)** concept. To gain resilience, it can use specialized persistence services such as EFS, EBS, and Amazon S3.

There are two main groups of clusters in AWS, each one with a specific purpose:

- **Cluster HPC:** This cluster is tightly coupled, and the network performance is a major concern. In this model, we use higher throughput instances, placement groups, jumbo frames, and single AZ compute nodes, and they need strong orchestration mechanisms. Examples of these technologies are media transcoding services and fraud risk analysis:



- **Distributed grid:** The distributed grid uses a different approach. In some cases, each node can be a client and a server. Here, the geography is not the main concern, because they are loosely coupled. You can use multiple AZs and multiple regions to deploy distributed clusters. They require low orchestration mechanisms. Examples of this technology are **SETI@home** and **blockchain** projects:



Both these models use parallel processing to split big jobs into smaller ones, Auto Scaling can be used to gain elasticity and cost efficiency.

High performance computing

High performance computing (HPC) allows scientists and engineers to solve complex, compute-intensive problems.

CfnCluster

AWS has designed a framework to create and manage clustering environments on EC2 in an easy way. CFNCluster is an HPC-persistent cluster that uses the following AWS services:

- AWS CloudFormation
- **AWS Identity and Access Management (IAM)**
- Amazon SNS
- Amazon SQS
- Amazon EC2

- Auto Scaling
- Amazon EBS
- Amazon CloudWatch
- Amazon S3
- Amazon DynamoDB

You can make use of the CLI to get started with your clustering environment. For Linux machines, perform the following steps. You'll need to have the AWS CLI program installed and properly configured.

```
sudo pip install cfncluster
```

1. Once installed, configure the cluster environment parameters such as the access keys, region, and VPC, as follows:

```
[ec2-user@ip-172-31-0-44 ~]$ cfncluster configure
Cluster Template [default]:
AWS Access Key ID []:
AWS Secret Access Key ID []:
Acceptable Values for AWS Region ID:
  ap-south-1
  eu-west-3
  eu-west-2
  eu-west-1
  ap-northeast-2
  ap-northeast-1
  sa-east-1
  ca-central-1
  ap-southeast-1
  ap-southeast-2
  eu-central-1
  us-east-1
  us-east-2
  us-west-1
  us-west-2
AWS Region ID []: us-east-1
VPC Name [public]: cluster-vpc
```

This command will provision all the resources necessary to bootstrap the default cluster.

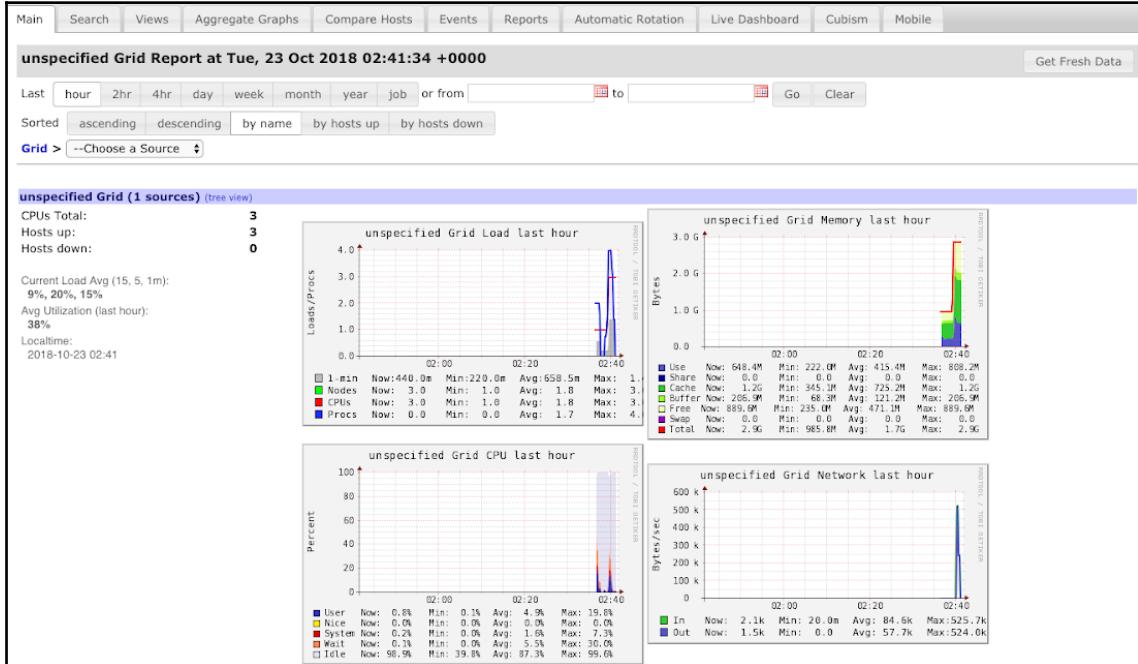
2. To override these values, edit the file `~/.cfncluster/config`:

Name	Environment	company	owner	Instance ID	Instance Type	Availability Zone	Instance State	Launch Time	Security Groups
CFNCluster Command CLI				i-08661892c16d0ca05	i2.micro	us-west-1b	running	October 22, 2018 at 6:50:04 ...	cfn-cluster-sg
Compute				i-05244a6ae43ebec2	i2.micro	us-west-1c	running	October 22, 2018 at 9:36:49 ...	cfncluster-mycluster-ComputeSecurityGroup-987...
Compute				i-0e7b08d0cal9cb576	i2.micro	us-west-1c	running	October 22, 2018 at 9:36:49 ...	cfncluster-mycluster-ComputeSecurityGroup-987...
Master				i-055eba50a11a46ed	i2.micro	us-west-1c	running	October 22, 2018 at 9:33:14 ...	cfncluster-mycluster-MasterSecurityGroup-WJQ...

3. When the cluster is ready, use `cfncluster status <name-of-the-cluster>`. You can ssh into the master node and submit jobs and monitor the jobs execution process in the Ganglia URLs as follows:

```
[ec2-user@ip-172-31-0-44 ~]$ cfncluster status mycluster
Status: CREATE_COMPLETE
MasterServer: RUNNING
MasterPublicIP: 13.52.14.117
ClusterUser: ec2-user
MasterPrivateIP: 172.31.24.39
GangliaPublicURL: http://13.52.14.117/ganglia/
GangliaPrivateURL: http://172.31.24.39/ganglia/
```

The Ganglia monitor shows the dashboard as follows:



The POSIX qsub utility is used to submit jobs to the scheduler.



For more information about the CfnCluster, please go to <https://aws.amazon.com/hpc/sc15/getting-started/>, and for a comprehensive guide on the CLI, visit <https://cfncluster.readthedocs.io/en/latest/>.

Enhanced networking

Enhanced networking is a virtualization method that can be enabled on a per-instance type basis, leveraging higher network throughput at the instance level, improving higher I/O performance and low latency. There are two ways to achieve this networking feature.

- **Intel 82599 VF: Virtual Function (VF)** is a virtualization method compatible with C3, C4, D2, I2, M4, and R3 instance types with velocities of up to 10 Gbps. You can verify your actual support for VF with the following command:

```
aws ec2 describe-instance-attribute --instance-id instance_id --attribute sriovNetSupport
```

- **Elastic Network Adapter**: The **Elastic Network Adapter (ENA)** is a built-in capability for EC2 instances that can be enabled at the OS kernel level. This feature is compatible with C5, C5d, F1, G3, H1, I3, m4.16xlarge, M5, M5d, P2, P3, R4, R5, R5d, X1, X1e, and z1d instance types. You can verify your actual support for ENA with the following command:

```
aws ec2 describe-instances --instance-ids instance_id --query "Reservations[] . Instances[] . EnaSupport"
```

Jumbo frames

Jumbo frames are a standard mechanism to alter the **Maximum Transmission Unit (MTU)** of the EC2 instance. The Ethernet protocol has a maximum transmission unit of 1,500 bytes, with jumbo frames enabled at the instance level, this communication limit can be extended to 9,000 MTU by increasing the message payload.

Only certain instance types are jumbo frames are compatible. To check this, you can use the tracepath command to validate the amount of transmission units between hosts. Also, you can use the ip link command to validate at the interface level:

```
[ec2-user ~]$ tracepath amazon.com
 1?: [LOCALHOST]          pmtu 9001
 1:  ip-172-31-16-1.us-west-1.compute.internal (172.31.16.1)    0.187ms pmtu
1500
 1:  no reply
 2:  no reply
 3:  no reply
 4:  100.64.16.241 (100.64.16.241)                      0.574ms
 5:  72.21.222.221 (72.21.222.221)                      84.447ms asymm
21
 6:  205.251.229.97 (205.251.229.97)                  79.970ms asymm
19
 7:  72.21.222.194 (72.21.222.194)                  96.546ms asymm
16
 8:  72.21.222.239 (72.21.222.239)                  79.244ms asymm
15
 9:  205.251.225.73 (205.251.225.73)                  91.867ms asymm
16
 ...
31:  no reply
      Too many hops: pmtu 1500
      Resume: pmtu 1500
```

In the previous example code, jumbo frames between the instance and the next hop are not enabled. In the following example code, jumbo frames are enabled for the eth0 interface:

```
[ec2-user ~]$ ip link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state
UP mode DEFAULT group default qlen 1000
      link/ether 02:90:c0:b7:9e:d1 brd ff:ff:ff:ff:ff:ff
```

All VPC communications can be configured to operate using jumbo frames, but the MTU must be downgraded to 1,500 before the communications leave the VPC, to avoid unexpected results.

Placement groups

Placement groups are a great way to improve the network performance (the highest packets per second between instances) and the lowest latency for intensive applications by co-locating instances physically in the same hardware.

The spread placement groups extends the single hardware limitations of a placement group by using different distributed hardware, eliminating single points of failure.

Creating a placement group

1. To create a placement group, navigate to EC2, and select **Placement groups** and **Create Placement Group**, as follows:

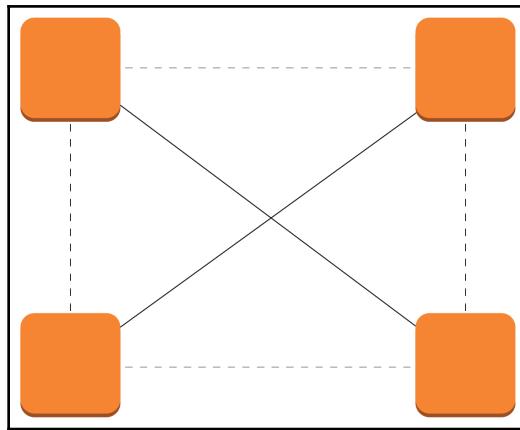
The screenshot shows the 'Create Placement Group' wizard. At the top, it says 'Placement groups > Create Placement Group'. Below that is the title 'Create Placement Group'. There are two main input fields: 'Name*' containing 'HPC Placement Group' and 'Strategy*' set to 'Cluster'. Below these fields is a link '▶ AWS Command Line Interface command'. At the bottom right are two buttons: 'Cancel' and a blue 'Create' button.

2. The allocation of instances inside the placement group is a *one-time-only* action. If you want to modify the placement group by adding instances—you'll need to relaunch the complete cluster again with run-instances or via the console—into the placement group:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
HPC Cluster	i-05565eacf248261b6	m5.large	us-east-1c	running	Initializing	None
HPC Cluster	i-0ae4bd0617e3639...	m5.large	us-east-1c	running	Initializing	None
HPC Cluster	i-0d809ab4b461f11dc	m5.large	us-east-1c	running	Initializing	None
HPC Cluster	i-0f69ef14e4f83ca49	m5.large	us-east-1c	running	Initializing	None

AMI ID	-	Source/dest. check	true
Key pair name	LabCodeDeploy	T2/T3 Unlimited	-
Owner	970129648716	EBS-optimized	True
Launch time	October 21, 2018 at 11:38:24 PM UTC-5 (less than one hour)	Root device type	ebs
Termination protection	False	Root device	/dev/xvda
Lifecycle	normal	Block devices	/dev/xvda
Monitoring	basic	Elastic GPU	-
Alarm status	None	Elastic GPU type	-
Kernel ID	-	Elastic GPU status	-
RAM disk ID	-		
Placement group	HPC Placement Group		
Virtualization	hvm		
Reservation	r-022a09cf30ecf6d65		
AMI launch index	3		
Tenancy	default		
Host ID	-		
Affinity	-		

As a result, we now have a 10 GB fully-meshed EC2 network in a single AZ:



At the moment of writing this book, only these instance types are allowed for placement grouping:

- **General purpose:** M4, M5, M5d
- **Compute optimized:** C3, C4, C5, C5d, cc2.8xlarge
- **Memory optimized:** cr1.8xlarge, R3, R4, R5, R5d, X1, X1e, z1d
- **Storage optimized:** D2, H1, hs1.8xlarge, I2, I3, i3.metal
- **Accelerated computing:** F1, G2, G3, P2, P3

Benchmarking

EC2 network performance depends on different factors such as the physical distance between instances, the maximum instance transmission units, the size of the EC2 instance, and the network optimizations, for example, the protocol type and the kind of workload running.

Let's create two instances (client and server) to measure the network performance and install the iperf3 software package, using the following command:

```
yum -y install  
https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm  
& yum -y install iperf3
```

On the server side, do the following (defaults to the 5201 TCP port):

```
sudo iperf3 -s
```

On the client side, use this command (10 parallel connections, 60 seconds interval):

```
sudo iperf3 -c 192.0.2.0 -P 10 -i 1 -t 60 -v

$ iperf3 -c 192.0.2.0 -t 2 -i 1 -P 2 -R
Connecting to host 192.0.2.0, port 5201
Reverse mode, remote host 192.0.2.0 is sending
[ 4] local 198.51.100.0 port 47122 connected to 192.0.2.0 port 5201
[ 6] local 198.51.100.0 port 47124 connected to 192.0.2.0 port 5201
[ ID] Interval      Transfer Bandwidth
[ 4] 0.00-1.00    sec 572 MBytes  4.80 Gbits/sec
[ 6] 0.00-1.00    sec 572 MBytes  4.80 Gbits/sec
[SUM] 0.00-1.00    sec 1.12 GBytes  9.60 Gbits/sec
-
[ 4] 1.00-2.00    sec 573 MBytes  4.80 Gbits/sec
[ 6] 1.00-2.00    sec 573 MBytes  4.80 Gbits/sec
[SUM] 1.00-2.00    sec 1.12 GBytes  9.61 Gbits/sec
-
[ ID] Interval      Transfer Bandwidth      Retr[ 4]
0.00-2.00    sec 1.12 GBytes  4.82 Gbits/sec 0          sender
[ 4] 0.00-2.00    sec 1.12 GBytes  4.81 Gbits/sec      receiver
[ 6] 0.00-2.00    sec 1.12 GBytes  4.81 Gbits/sec 0          sender
[ 6] 0.00-2.00    sec 1.12 GBytes  4.81 Gbits/sec      receiver
[SUM] 0.00-2.00    sec 2.24 GBytes  9.63 Gbits/sec 0          sender
[SUM] 0.00-2.00    sec 2.24 GBytes  9.63 Gbits/sec      receiver

iperf Done.
```

From the preceding output, we come to the conclusion that the total bandwidth available between the two instances is ~9.63 Gbits per second.

Elastic MapReduce

Elastic MapReduce (EMR) is a fully-managed cluster platform for running big-data and analytics frameworks such as Apache Hadoop, Spark, HBase, Presto, Impala, Cascading, and Flink. Running Hadoop clusters is a complex and time-consuming task. EMR provisions the cluster and installs frequently used frameworks for data scientists, analysts, and engineers.

EMR provides the flexibility to bootstrap your cluster, with a series of steps defined by the customer to install, configure, and prepare your data to be processed. EMR can use the Hadoop distributed file system on EBS volumes or EMRFS with Amazon S3 as the backing persistence service.

EMR clusters have a variety of use cases, from ETL and batch processing to real-time applications integrating Amazon Firehose or Apache Spark, and a wide number of connectors and integration architectures. Clusters on EMR can be transient for a one-time use case, or persistent, meaning they are constantly processing data without interruption.

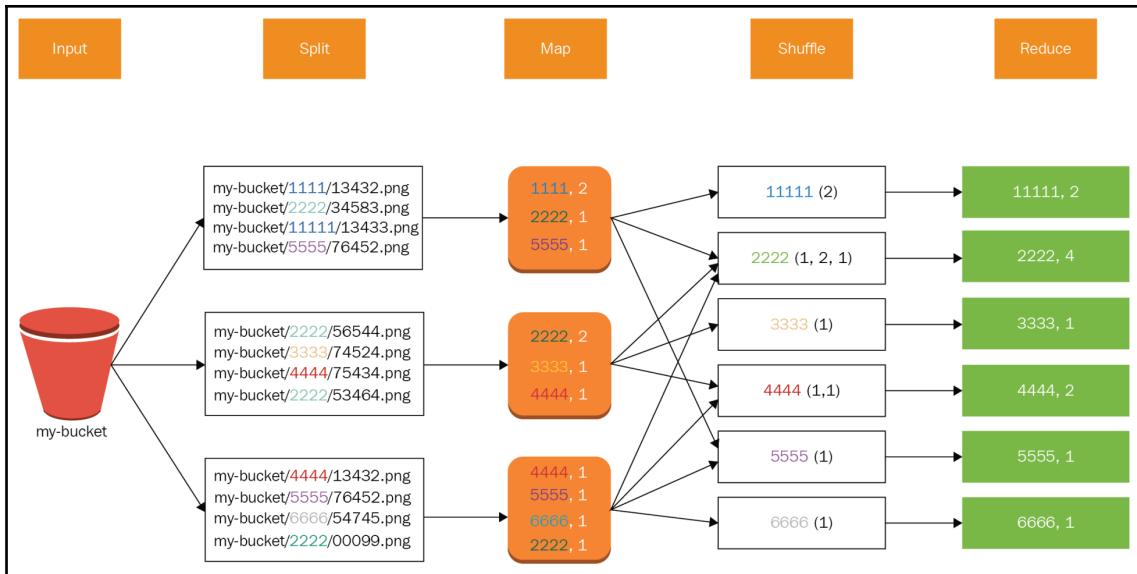
Spot instances can be used to lower the costs of massive processing of compute nodes, making EMR a cost-efficient solution. The EMR architecture is elastic, with the capacity to increase the number of processing nodes and unlimited resilient storage when using Amazon S3.

MapReduce

Consider the following problem: You have one billion objects in S3, and you want to list all the keys to make an inventory. Assuming you can list 1,000 keys per second and you perform this task sequentially, it would take 11 days to complete this task from a single program. Now consider the same problem, but with 100 programs running in parallel—the time drops considerably. This is the main concept about the MapReduce algorithm.

MapReduce is a programming framework that allows developers to use distributed computing to split jobs across hundreds or even thousands of machines to execute them in parallel, and it can be run on commodity hardware using **Hadoop Distributed File Systems (HDFS)**, with a resilient design.

- **Map:** The mappers are responsible for mapping the input data (in the preceding problem, the object keys); here, the filtering and sorting phases are made.
- **Reduce:** The reducers are responsible for the data aggregation and summarization (in our case, they perform the key count for the common key prefixes):



B10781_12_10

In the previous diagram, we can observe the full life cycle of a MapReduce job that is programmed to count and summarize the number of different prefixes in an S3 bucket with huge amounts of data.

Analyzing a public dataset

Public datasets are a great way to learn, by understanding others' problems and thinking out of the box.

Log analysis is a great use case due to its non-structured nature. Companies want to take advantage by mining server access logs, understanding their users' behavior, targeting ad campaigns in real time, and finding useful patterns in data. From an IT perspective, active monitoring requires us to understand failure origins and anomaly detection. In our exercise, we need to find the 404 HTTP errors and the corresponding source page count for this dataset.

Amazon hosts public datasets via the *AWS Public Dataset Program* (<https://aws.amazon.com/opendata/public-datasets/>); other sources are also pretty interesting as Kaggle datasets.

1. Navigate to <https://www.kaggle.com/datasets> and perform a search for NASA access logs. Then download the log files locally:

The screenshot shows the Kaggle website interface. At the top, there's a navigation bar with links for 'Search kaggle', 'Competitions', 'Datasets', 'Kernels', 'Discussion', 'Learn', and more. A user profile icon is on the far right. Below the navigation is a dark blue header for the dataset 'NASA_access_log_1995' by Vincent Zhang, updated 7 months ago (Version 1). The header includes a 'share' button with '1 voter'. The main content area shows the dataset details: 'Data (23 KB)', 'API', a download command 'kaggle datasets download -d vinzzhang/nasa-access_log_1995', and a 'Download All' button. Below this, there are sections for 'Data Sources' (listing 'NASA_access_log_95') and 'About this file' (describing it as 'Web log files from NASA in 1995'). A green status bar at the bottom says '✓ Registration complete. You are now logged in.' The bottom part of the screenshot displays the raw log file content.

```
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0" 200 6245
unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET /shuttle/countdown/ HTTP/1.0" 200 3985
199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET /shuttle/missions/sts-73/mission-sts-73.html HTTP/1.0" 20
burger.letters.com - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/countdown/liftoff.html HTTP/1.0" 304 0
199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/missions/sts-73/sts-73-patch-small.gif HTTP/1.0"
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 304 0
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/video/livevideo.gif HTTP/1.0" 200
205.212.115.106 - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/countdown.html HTTP/1.0" 200 3985
d104.aa.net - - [01/Jul/1995:00:00:13 -0400] "GET /shuttle/countdown/ HTTP/1.0" 200 3985
129.94.144.152 - - [01/Jul/1995:00:00:13 -0400] "GET / HTTP/1.0" 200 7074
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET /shuttle/countdown/count.gif HTTP/1.0" 200 40310
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 200 786
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET /images/KSC-logosmall.gif HTTP/1.0" 200 1204
```

2. Create an S3 bucket and upload the log data in an /input folder:

The screenshot shows the AWS S3 console with the path 'Amazon S3 > emr-input-data-associate-architect'. The 'Management' tab is selected. A search bar contains the placeholder 'Type a prefix and press Enter to search. Press ESC to clear.' Below the search bar are buttons for 'Upload', '+ Create folder', and 'Actions'. The location 'US East (N. Virginia)' is shown with a refresh icon. The results table has columns: Name, Last modified, Size, and Storage class. The table shows the following data:

Name	Last modified	Size	Storage class
input	--	--	--
results	--	--	--
hive-query.q	Oct 22, 2018 7:05:40 PM GMT-0500	547.0 B	Standard
results_\$.folder\$	Oct 22, 2018 7:07:54 PM GMT-0500	0 B	Standard

At the bottom right of the table area, it says 'Viewing 1 to 4'.

3. Navigate to EMR in the AWS console and click on `Create cluster`, as follows:

Welcome to Amazon Elastic MapReduce

Amazon Elastic MapReduce (Amazon EMR) is a web service that enables businesses, researchers, data analysts, and developers to easily and cost-effectively process vast amounts of data.

You do not appear to have any clusters. Create one now:

[Create cluster](#)

How Elastic MapReduce Works

Upload

Upload your data and processing application to S3.

[Learn more](#)

Create

Configure and create your cluster by specifying data inputs, outputs, cluster size, security settings, etc.

[Learn more](#)

Monitor

Monitor the health and progress of your cluster. Retrieve the output in S3.

[Learn more](#)

4. Show the advanced options before creating the cluster so we can make sure that the Hive framework will be installed. Find the **Add steps** sections, choose the add a **Hive program** step, and then click **Configure**, as shown here:

Create Cluster - Advanced Options [Go to quick options](#)

Software Configuration

Release emr-5.17.0

<input checked="" type="checkbox"/> Hadoop 2.8.4	<input type="checkbox"/> Zeppelin 0.7.3	<input type="checkbox"/> Livy 0.5.0
<input type="checkbox"/> JupyterHub 0.8.1	<input type="checkbox"/> Tez 0.8.4	<input type="checkbox"/> Flink 1.5.2
<input type="checkbox"/> Ganglia 3.7.2	<input type="checkbox"/> HBase 1.4.6	<input checked="" type="checkbox"/> Pig 0.17.0
<input checked="" type="checkbox"/> Hive 2.3.3	<input type="checkbox"/> Presto 0.206	<input type="checkbox"/> ZooKeeper 3.4.12
<input type="checkbox"/> MXNet 1.2.0	<input type="checkbox"/> Sqoop 1.4.7	<input type="checkbox"/> Mahout 0.13.0
<input checked="" type="checkbox"/> Hue 4.2.0	<input type="checkbox"/> Phoenix 4.14.0	<input type="checkbox"/> Oozie 5.0.0
<input type="checkbox"/> Spark 2.3.1	<input type="checkbox"/> HCatalog 2.3.3	<input type="checkbox"/> TensorFlow 1.9.0

AWS Glue Data Catalog settings (optional)

Use for Hive table metadata [i](#)

Edit software settings [i](#)

Enter configuration Load JSON from S3

```
classification=config-file-name,properties=[myKey1=myValue1,myKey2=myValue2]
```

Add steps (optional) [i](#)

Step type [Configure](#) 

Auto-terminate cluster after the last step is completed

[Cancel](#) [Next](#)

Apache Hive is a data warehouse project that runs on top of Apache Hadoop, and we will be using Hive to interact via SQL to aggregate the data.

5. You can download the Hive query from <https://github.com/gabanox/Certified-Solution-Architect-Associate-Guide/blob/master/chapter11/hive-query.q>.
6. Make sure to have the Hive query uploaded in the same S3 bucket for simplicity and create a folder to be used as the output destination for the EMR job:

Add step

Step type Hive program

Name

Script S3 location*  S3 location of your Hive script.
s3://<bucket-name>/<path-to-file>

Input S3 location  S3 location of your Hive input files.
s3://<bucket-name>/<folder>/

Output S3 location  S3 location of your Hive output files.
s3://<bucket-name>/<folder>/

Arguments
Specify optional arguments for your script.

Action on failure  What to do if the step fails.

After adding the job, we need to configure the hardware for our cluster; click **Next**.

7. In the cluster configuration screen, for ease of configuration, make sure to leave the default VPC selected, and change the instance count for the **Core** node type to 2, and for less expensive instances, use **c1.medium** in all node types. Do the same for the **Task** node and change it to 1, and click **Next**:

Node type	Instance type	Instance count	Purchasing option	Auto Scaling
Master Master - 1	m4.large	1 Instances	<input checked="" type="radio"/> On-demand	Not available for Master
Core Core - 2	c1.medium	2 Instances	<input checked="" type="radio"/> On-demand	Not enabled
Task Task - 3	c1.medium	1 Instances	<input checked="" type="radio"/> On-demand	Not enabled

+ Add task instance group

Cancel Previous Next

8. In the general cluster settings, you can specify the output for the cluster logs; in this case, leave everything as it is and click **Next**:

Create Cluster - Advanced Options [Go to quick options](#)

General Options

Cluster name

Logging [i](#)
S3 folder [i](#)

Debugging [i](#)

Termination protection [i](#)

Tags [i](#)

Key	Value (optional)
<input type="text" value="Add a key to create a tag"/>	<input type="text"/>

Additional Options

EMRFS consistent view [i](#)

Custom AMI ID [i](#)

▶ Bootstrap Actions

[Cancel](#) [Previous](#) [Next](#)

9. The security screen shows the options available to log into the cluster using SSH; this time, we don't require this option enabled because the output logs of the job can be downloaded via the console.

10. In the **Permissions** section, choose the **Default** option to let EMR create the required access role for this cluster:

Create Cluster - Advanced Options [Go to quick options](#)

Security Options

EC2 key pair Proceed without an EC2 key pair [i](#)

Cluster visible to all IAM users in account [i](#)

Permissions [i](#)

Default Custom

Select custom roles to tailor permissions for your cluster.

EMR role EMR_DefaultRole [i](#)

EC2 instance profile EMR_EC2_DefaultRole [i](#)

Auto Scaling role EMR_AutoScaling_DefaultRole [i](#)

▶ Authentication and encryption

▶ EC2 security groups

i No EC2 key pair has been selected, so you will not be able to SSH to this cluster or connect to HUE (unless you are using a VPN). [Learn how to create an EC2 Key Pair.](#)

[Cancel](#) [Previous](#) [Create cluster](#)

11. Now the job is running, you can visualize the standard output and standard error logs under the **Log files** column:

ID	Name	Status	Start time (UTC-5)	Elapsed time	Log files	Actions
s-3KDCFYWZW35K9	Hive program	Pending		0 seconds	No logs created yet	View jobs
s-17ZG0GZ611SK	Hive program	Pending		0 seconds	No logs created yet	View jobs
s-14FE9GL3JATR	Setup hadoop debugging	Pending		0 seconds	No logs created yet	View jobs

When the job is completed successfully, go to the S3 bucket and grab the results in the /output folder to check the 404 aggregation results. Make sure to terminate the cluster to avoid unnecessary charges.

Summary

In this chapter, you have learned about some of the options available for clustering in AWS. We remarked on the differences between Cluster HPC and Distributed Grids, and we created a cluster with the CfnCluster framework.

We also discussed some of the networking optimizations available at the hypervisor and interface level, and we learned how to inspect for jumbo frames capabilities and performed a TCP benchmark between instances and created a compute placement group.

We introduced EMR and learned how the MapReduce programming model works, creating an EMR cluster that performs aggregation from logs from a public dataset.

Further reading

The *Mastering Hadoop 3* and the *Big Data Architect's Handbook* books are recommended. To deep dive into **Extract Transform and Load (ETL)** workflows, read about **AWS Glue** (<https://aws.amazon.com/glue>).

EMR works perfectly with Amazon S3 to build data lakes; for more information, go to the following link: <https://aws.amazon.com/big-data/datalakes-and-analytics/>, and for a deep understanding of Hadoop architecture and HDFS, use the following links:

- *HDFS Architecture*: <https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html>.
- *Hadoop Architecture Overview*: <http://ercoppa.github.io/HadoopInternals/HadoopArchitectureOverview.html>.

12

Web Scale Applications

Web scale applications need different approaches to design and architecture, and a new strategy can be taken. Servers are hard to scale to adjust the right capacity when needed and we have the operation overhead to manage the operating system and the runtime for applications. Serverless solutions, such as AWS Lambda, are a very good choice to migrate applications that can scale to thousands of requests concurrently without managing servers.

AWS Lambda is a fully managed compute service that handles all the runtime and operations so customers can focus on their applications.

The topic on AWS Lambda is covered very much in detail in this chapter.

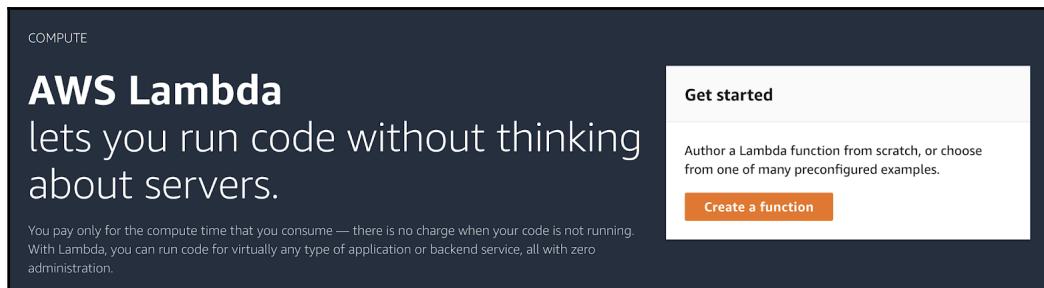
Technical requirements

There are no technical requirements for this chapter.

AWS Lambda

Lambda provides the container to run several runtimes like Python, Java, .NET Core, Go, and Node.js. With AWS Lambda, you can specify the resources that your application will need and it will be executed as a response to events or schedules.

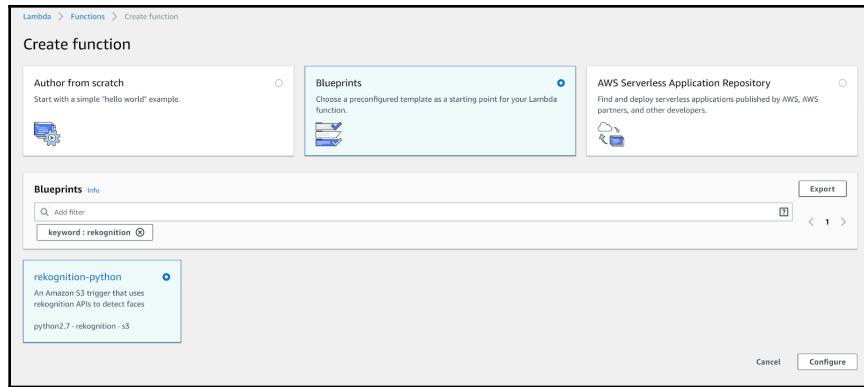
1. To get started with your first Lambda function, navigate to the Lambda service and choose **Create a function**, as shown in the following screenshot:



2. We will make use of the function **Blueprints** to create a faces detection function that uses **Amazon Rekognition** to extract metadata out of images uploaded to an S3 bucket. The final result will be a print to the standard output of the Lambda container that will be streamed to CloudWatch logs. The code performs an analysis trying to find image landmarks using machine learning.



For more information about detecting images in faces using Amazon Rekognition, use the following link: <https://docs.aws.amazon.com/rekognition/latest/dg/faces-detect-images.html>. Use the search bar and put the word `rekognition`. This is a function authored in Python. Click on **Configure**, as shown in the following screenshot:



3. A **Name** and a **Role name** must be specified, as shown in the following screenshot:

The screenshot shows the 'Create function' wizard in the AWS Lambda console, specifically the 'Basic information' and 'Policy templates' sections.

Basic information

- Name:** FaceRecognition
- Role:** A dropdown menu is open, showing the option "Create a new role from one or more templates". A note below explains that Lambda automatically creates roles with permissions from selected policy templates, including basic Lambda permissions and CloudWatch logging.
- Role name:** FaceRecognitionRole

Policy templates: A note states that a role will be generated with permissions from selected policy templates. The available templates listed are:

- Amazon S3 object read-only permissions
- Amazon Rekognition no data permissions
- Amazon Rekognition write-only permissions

4. If you keep scrolling, you will find the source event. Lambda uses the pull and push model to listen to events, when the create object in S3 is emitted. Our Lambda function will be executed up to 15 minutes. Make sure to specify an existing bucket in the same region as the function.

S3 trigger Remove

Bucket
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.
 ▾

Event type
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
 ▾

Prefix
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

Suffix
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

Lambda will add the necessary permissions for Amazon S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Enable trigger
Enable the trigger now, or create it in a disabled state for testing (recommended).

5. Make sure to mark the **Enable trigger** checkbox. If you are curious analyze the Python code and click **Create function**, as shown in the following screenshot:

The screenshot shows the AWS Lambda function creation interface. At the top, it says "Lambda function code" and "Code is preconfigured by the chosen blueprint. You can configure it after you create the function." Below this, the "Runtime" is set to "Python 2.7". The code editor contains the following Python code:

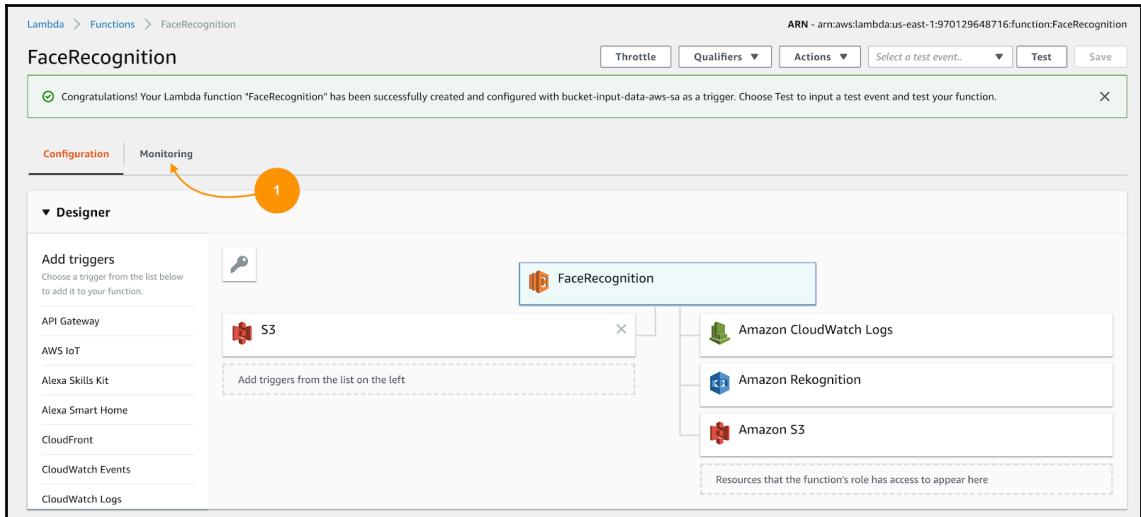
```
1 from __future__ import print_function
2
3 import boto3
4 from decimal import Decimal
5 import json
6 import urllib
7
8 print('Loading function')
9
10 rekognition = boto3.client('rekognition')
11
12
13 # ----- Helper Functions to call Rekognition APIs -----
14
15
16 def detect_faces(bucket, key):
17     response = rekognition.detect_faces(Image={"S3Object": {"Bucket": bucket, "Name": key}})
18     return response
19
20
21 def detect_labels(bucket, key):
22     response = rekognition.detect_labels(Image={"S3Object": {"Bucket": bucket, "Name": key}})
23
24     # Sample code to write response to DynamoDB table 'MyTable' with 'PK' as Primary Key.
25     # Note: role used for executing this Lambda function should have write access to the table.
26     #table = boto3.resource('dynamodb').Table('MyTable')
27     #table.put_item(Item=response)
```

* These fields are required.

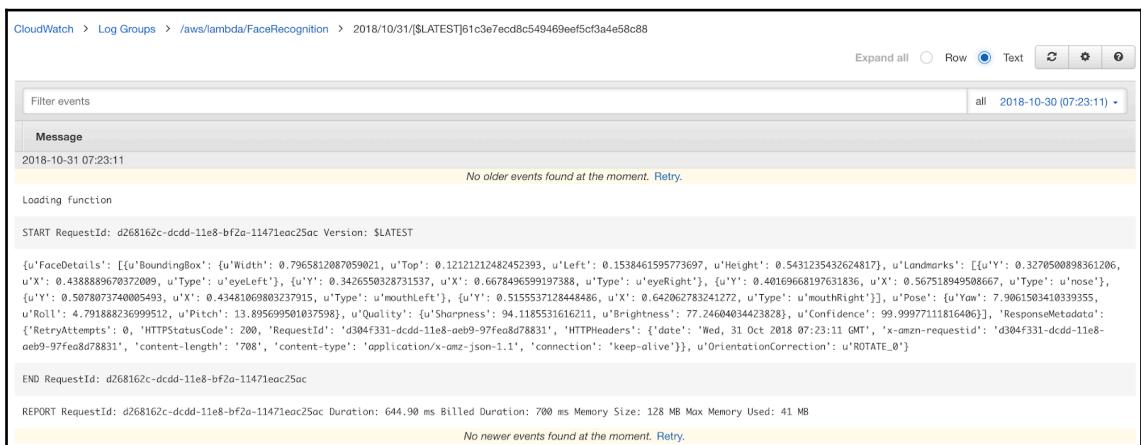
Cancel Previous Create function

6. Now we have created a Lambda function and we are ready to test it out. Take a moment to look at line 10, here we are making use of the Rekognition SDK, which means that you can automate any kind of processing task from lambda by using the service APIs and providing the right IAM permissions.

7. Proceed to upload any kind of photo to the origin S3 bucket. Once the function is triggered by the create object event we can see the execution results and other metrics in the **Monitoring** tab (1):



8. Find the button **View logs** in **CloudWatch | Log Groups** to be redirected to the log stream for this function.



Here we can see the results of the `print()` Python function and the time it took to execute the function and the memory used for the operation (41 MB).

You have created a serverless compute function that works only when events are generated and the capacity to execute 1000 functions in parallel and the advantage of never paying for IDE infrastructure. AWS offers 1 Million requests per month at no cost.

Summary

In this chapter, we created a Lambda function by using the blueprints with an Amazon Rekognition service that performs analysis on images to detect faces. We configured an S3 bucket as the origin from Lambda that triggers the processing event.

Further reading

- **AWS Lambda Limits:** <https://docs.aws.amazon.com/lambda/latest/dg/limits.html>
- **Using the AWS Serverless Application Model (AWS SAM):** https://docs.aws.amazon.com/lambda/latest/dg/serverless_app.html
- **Mastering AWS Lambda:** <https://www.packtpub.com/virtualization-and-cloud/mastering-aws-lambda>

13

Understanding Access Control

Having a solid understanding of how AWS manages and implements access control for your AWS resources is essential if you want to pass the exam. Security features heavily in the exam; it's expected that you will be able to determine the best methods of access, and will also have the ability to define and recommend different access control methods, depending on a specific set of scenario requirements.

In this chapter, we will look at a range of different access control methods, with a special focus on the **Identity and Access Management (IAM)** service. We will also explain the differences between authorization and authentication, which can sometimes be misconstrued as the same principle, even though the two are very different mechanisms.

In this chapter, we will cover the following topics:

- Authentication, authorization, and access control
- Authentication via access control methods
- IAM authorization

Technical requirements

In this chapter, it will be helpful if you have basic knowledge and awareness of the IAM service.

Authentication, authorization, and access control

Before I focus on the different methods of access control and how access control is managed, I want to step back and explain how access is actually granted to a resource—how you, a service, or an application (essentially, any identity), gain the access and permission. I will also explain the access control methods used to carry out the actions that are required.

Authentication

At a high level, when an identity has to gain access to a resource or an environment (for example, a user logging in to an AWS account), the user has to identify itself in the form of a username, and then verify that they are who they say they are, which is normally confirmed in the form of a password. If the verification process is successful, the identity is then authenticated. **Authentication** simply means that the identity has correctly identified itself as an identity that is permitted to gain access.

When you are authenticating to a system, your identity has to be a unique identity to ensure that there is no confusion with other identities. You have probably seen this yourself if you have tried to sign up to a newsletter on a website or have created a new email address. Your first attempt of an ID may have been rejected because it was already in use. In such a situation, you have to come up with another unique login ID. The password, however, does not have to be unique; it's likely that multiple identities for the same system are using the same password, without even knowing it.

Without realizing it, many of us perform actions relating to the principles of authentication on a daily basis; it's not just a process related to the cloud or computers. For example, we all use credit/debit cards, and when we pay for something with our card, we are essentially providing an identity to the card reader, defining who we are and ensuring that we can pay for the item we are buying. The verification of that identity comes in the form of a pin number; this verification completes the authentication to the bank, and the money is deducted from our account.

So, authentication is a two-step process, comprised of an identity and the verification of that identity.

Authorization

Authorization is very different from authentication, and sometimes, people confuse the definitions of these two terms. Authorization dictates what actions the authenticated identity can perform. These actions are usually in the form of permissions sets, which either explicitly allow or explicitly deny access to resources. Within AWS, these permissions are largely defined within the IAM service, and I will get to how IAM manages the permissions for identities later in this chapter.

So, bear in mind that it's not possible to gain access to a resource without first passing an authentication barrier, which identifies and verifies who you are. Only when it has been authenticated can an identity be authorized to perform actions, using the list of permissions that have been assigned.

Access control

Now, you should have a clearer understanding of authentication and authorization. How do these mechanisms play a part in access control? The term **access control** is an umbrella term for how we can manage, restrict, and allow access to resources, and there are different mechanisms for achieving this within AWS.

I have already mentioned one method: using a username and password for your user accounts. Another method of access control is to implement **Multi-Factor Authentication (MFA)**, which uses a randomized, six-digit number that changes on a regular basis and has to be entered, along with the password. This provides a second level of verification in the authentication process, making it harder to compromise a specific account.

There are several other methods of access control used within AWS; an example is **federated access**. Federated access allows users outside of your AWS IAM environment to connect and access your AWS resources, using existing accounts, such as their Google or Facebook account, or even their on-premise corporate MS Active Directory accounts. We will discuss federation later.

IAM roles are another method of access control. By using roles, you are able to assume temporary credentials, allowing you to gain access to AWS resources.

Authenticating via access control methods

AWS offers the ability to access resources through a wide range of access control mechanisms, with an array of authentication methods. In this section, I want to cover some of them (that you will need to know about for the exam) in greater detail.

Usernames and passwords

I already mentioned IAM usernames and passwords, which are used to log in to your AWS account. These usernames are created by an administrator of IAM; at the same time, they will issue you with a password. Depending on how the administrator configured your account, you may be required to change your password upon the first successful authentication to AWS.

If no password policy has been configured, passwords for accounts can be inherently insecure. This is because people are likely to use easy to remember passwords, such as 123456 or Password1; these passwords are more common than you think, in spite of end users becoming more aware of privacy and security. This can be a major issue if the users that are using insecure passwords have a privileged access level to your resources; for example, other administrators might have weak passwords, while also having the authorization to create or delete a wide range of AWS resources.

Multi-factor authentication

To help combat the potential security risk mentioned in the preceding section, IAM has a feature known as MFA. This is recommended for any user that has an elevated set of permissions within your AWS account, and it is a security best practice to use MFA in your AWS root account.

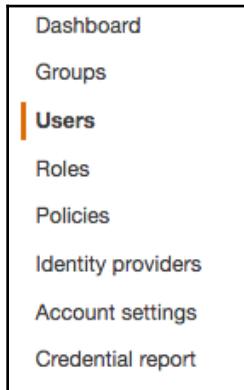
MFA provides a second layer of authentication, following a user logging into your AWS account with a password. MFA will ask the user to enter a six-digit, randomized number, which will change very frequently; if the correct response is entered, then the user will be fully authenticated. This ensures that, should a password be compromised, there will be a second factor of authentication which is far harder to breach since the code is frequently changing.

There are many supported MFA devices that can be used with AWS, as you can see in the following table:

MFA Form Factors					
	Virtual MFA Device	Hardware Key Fob MFA Device	Hardware Display Card MFA Device	SMS MFA Device (Preview)	Hardware Key Fob MFA Device for AWS GovCloud (US)
Device	See table below.	Purchase.	Purchase.	Use your mobile device.	Purchase.
Physical Form Factor	Use your existing smartphone or tablet running any application that supports the open TOTP standard.	Tamper-evident hardware key fob device provided by Gemalto, a third-party provider.	Tamper-evident hardware display card device provided by Gemalto, a third-party provider.	Any mobile device that can receive Short Message Service (SMS) messages.	Tamper-evident hardware key fob device provided by SurePassID, a third-party provider.
Price	Free	\$12.99	\$19.99	SMS or data charges may apply.	\$15.95
Features	Support for multiple tokens on a single device.	The same type of device used by many financial services and enterprise IT organizations.	Similar to key fob devices, but in a convenient form factor that fits in your wallet like a credit card.	Familiar option with low setup costs.	A key fob device exclusively for use with AWS GovCloud (US) accounts.
Compatibility with AWS GovCloud (US)	✓				✓
Compatibility with Root Account	✓	✓	✓		
Compatibility with IAM User	✓	✓	✓	✓	✓

Setting MFA for a user is a simple process, and it is completed from within the IAM service, using the following steps:

1. Log in to your AWS account.
2. Select the IAM service from the service page.
3. Select **Users** from the menu, as follows:



4. Select the user that you would like to configure MFA for.

5. Select the **Security credentials** tab, as shown in the following screenshot:

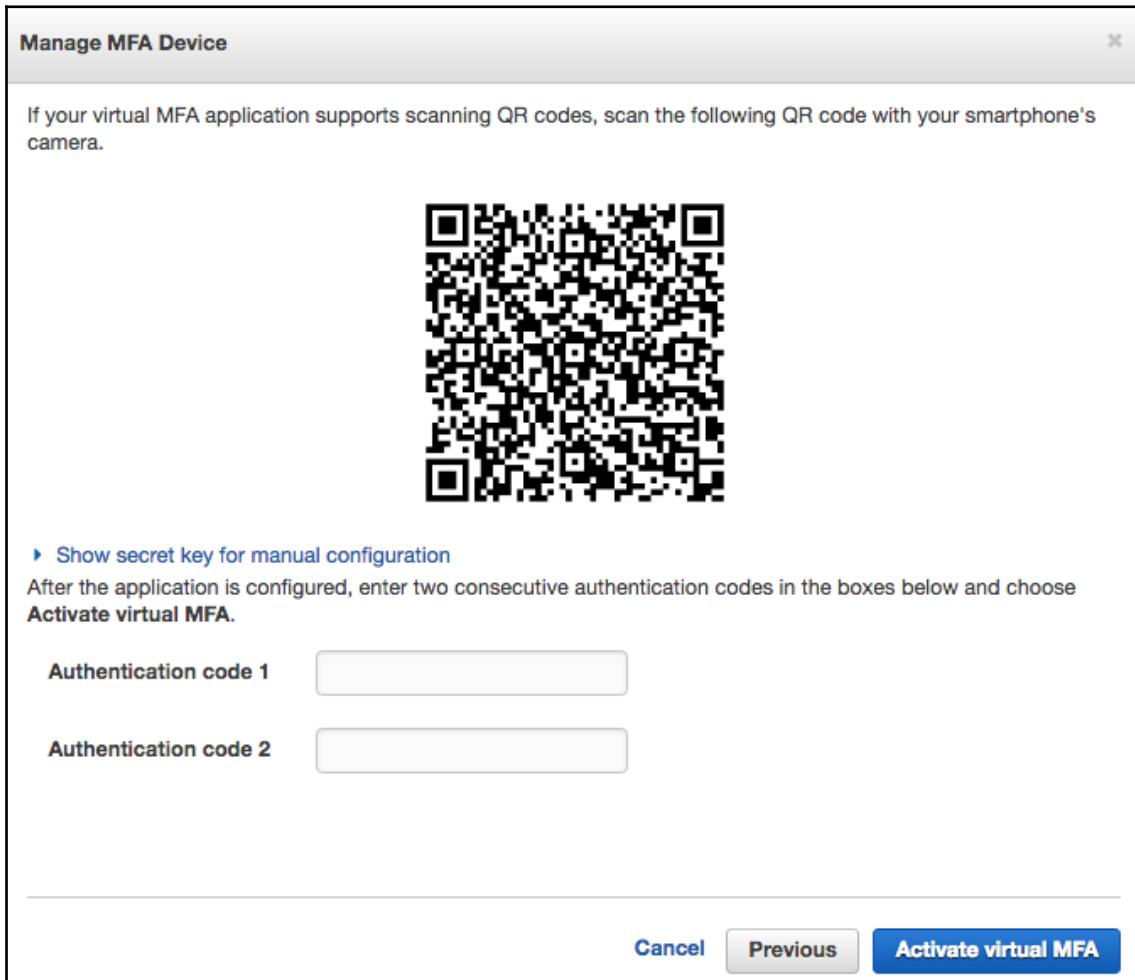
The screenshot shows the AWS IAM 'Users' section with 'PackT' selected. Under 'PackT', the 'Summary' tab is active. At the top, it displays the User ARN (arn:aws:iam::730739171055:user/PackT), Path (/), and Creation time (2018-09-14 10:50 UTC+0100). Below this, there are four tabs: 'Permissions', 'Groups', 'Security credentials' (which is highlighted in orange), and 'Access Advisor'. Under the 'Sign-in credentials' section, it shows the following details:

Console password	Enabled		Manage password
Console login link	https://stuscott.siginin.aws.amazon.com/console		
Last login	Never		
Assigned MFA device	No		
Signing certificates	None		

6. You will notice that the **Assigned MFA device** reads **No**. Click on the pencil to the right of **No** to configure the setting.
7. Depending on your chosen MFA device, you will be asked to select either a virtual device or a physical device; afterwards, click on **Next Step**:

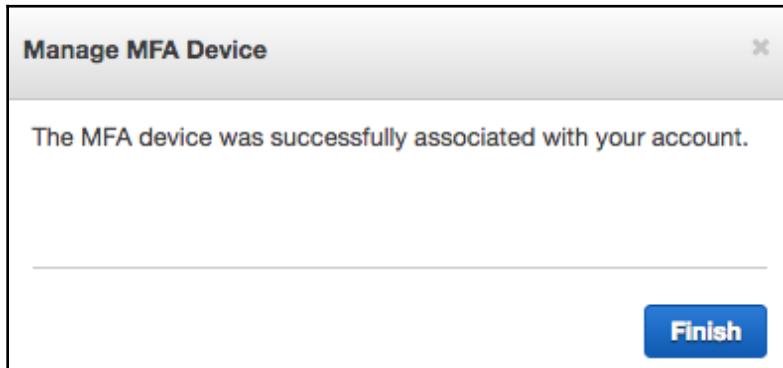
The dialog box is titled 'Manage MFA Device'. It contains the instruction 'Select the type of MFA device to activate:' followed by two radio button options: 'A virtual MFA device' (selected) and 'A hardware MFA device'. Below this, a note states 'For more information about supported MFA devices, see [AWS Multi-Factor Authentication](#)'. At the bottom right are 'Cancel' and 'Next Step' buttons.

8. For the rest of this process, I am going to select the virtual MFA device and use the Google Authenticator app on my phone to complete the process.
9. Next, you will be presented with the following screen, showing a QR code that you can scan with the Google Authenticator app:



10. After scanning the image, the app will display a six-digit code, which I must then enter within the **Authentication code 1** space. I must then wait for the six-digit number to change within the application, and then I must enter that new number within the **Authentication code 2** space.

11. Click on **Activate virtual MFA**.
12. Upon successful configuration, you will be shown the following message, stating that MFA has been successfully associated with your account:



The next time the user logs in to the AWS account with their username, they will be asked to enter their password as normal, and then they will be asked to enter the MFA six-digit number, which can be accessed using the MFA device that is used to configure the user:

A screenshot of a 'Multi-factor Authentication' form. The title is 'Multi-factor Authentication'. Below it, a message says 'Please enter an MFA code to complete sign-in.' There is a label 'MFA Code:' followed by a blue input field. At the bottom are two buttons: a blue 'Submit' button and a blue 'Cancel' link.

Programmatic access

When you are using AWS, there will be times when you will not actually need to log in to the AWS Management Console to access resources and perform specific tasks. AWS offers programmatic access to your resources through a range of SDKs and the AWS CLI. This allows users to access, modify, and control your AWS environment and resources through a Terminal window by issuing commands (instead of through the Management Console, with a point-and-click approach).

However, to do this, you have to be authenticated as a user that has the necessary authorization to perform actions; this is achieved through the use of access keys. These keys come as a pair, and can be generated for any user within IAM. They are then associated to only that user.

One key is known as the **access key**, and the other is the **secret access key**; they are constructed differently. The access key is 20 characters in length and is comprised of uppercase alphanumeric characters: for example, DWKYR45XM92EKLHP61DP. The secret access key is 40 characters in length and is made up of both uppercase and lowercase alphanumeric characters, in addition to some non-alphanumeric characters. An example of a secret access key is as follows:

```
SPv9S3/GWkC95Km/S24Vohr57Xs/awo07F6HEck9
```

When the access keys are created, they are created as a pair and are linked through a mathematical algorithm. They can be created from within IAM, or through the AWS CLI or SDK. Now, I will explain how you can use these keys to create the verification of authentication when accessing resources programmatically.

In this example, I shall explain how you can use these access keys to authenticate and verify your identity using the AWS CLI. This assumes that you have already installed the AWS CLI software and have downloaded the access keys from IAM. For more information on how to do this, please visit <https://docs.aws.amazon.com/cli/latest/userguide/installing.html>:

1. Open a Terminal window on your local computer.
2. Enter `aws configure`.
3. Enter the AWS Access Key ID.
4. Enter the AWS Secret Access Key.
5. It will then ask you to enter the Default region name and Default output format.

Upon completion, when you make a request to access a resource within your AWS account, the AWS CLI is configured with your access keys, which both identify and verify who you are. This allows you to be authenticated and authorized, as per the permissions associated with your user account within IAM.

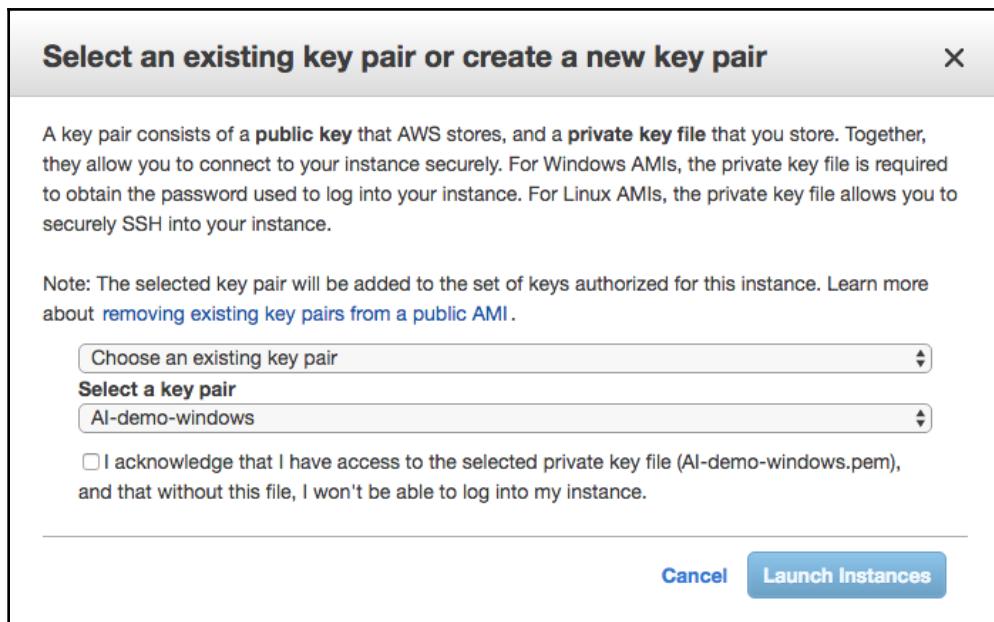
One point to mention regarding the access keys is that you are prompted to download them once they are created, and you only have one chance to do this. If you lose your secret access key, it's not possible to retrieve it, and you will have to create a new set of keys.

Key pairs

Many people confuse access keys with key pairs; however, they are two very different techniques of authentication and access control.

As you just learned, access keys are used to verify and authenticate to AWS when you are accessing resources programmatically. Key pairs are used by EC2 instances to allow you to connect and authenticate to an instance once it has been created. Again, key pairs consist of two keys: a public key and a private key, linked by a cryptographic algorithm.

At the last stage of launching a new instance, whether it be a Linux or Windows-based AMI, you will be asked to download a key pair, as shown in the following screenshot:

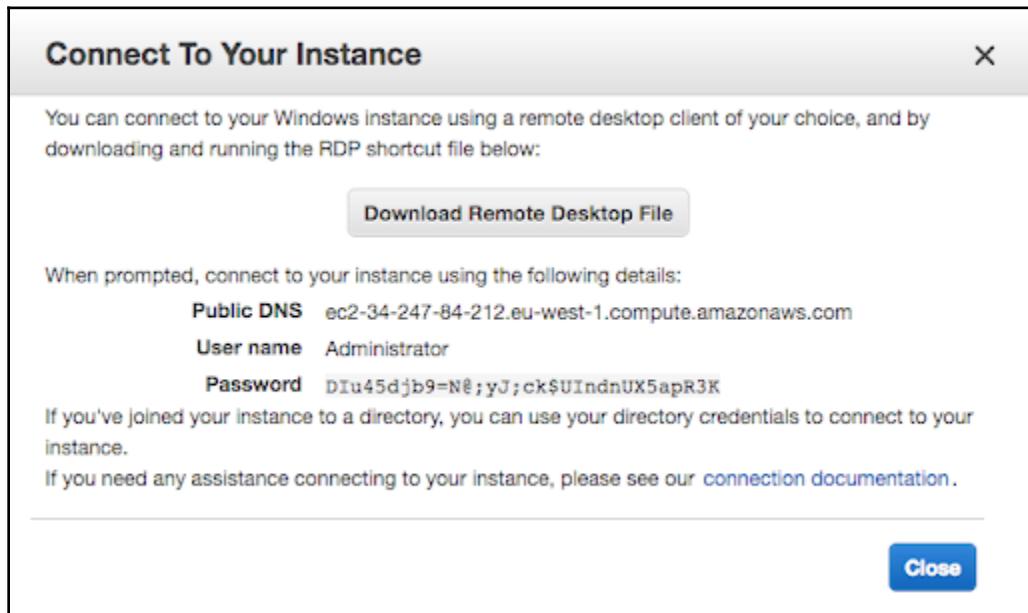


This is the last stage of your EC2 configuration before it is launched. At this stage, you have three options in the drop-down list, as follows:

1. Select an existing key pair
2. Create a new key pair
3. Proceed without a key pair

If you select an existing key pair, you have to ensure that you have access to that key pair, as you will need to provide the private key to that pair when connecting to the instance. If you decide to create a new key pair, AWS will generate a new set of keys for you, and you will be prompted to download the keys. Lastly, you can always choose to proceed without a key pair; be warned that if you select this option, you will have no way of connecting to that instance locally, to administer it in any way.

If you selected to create a new key pair, you must give it a name before you can download it. Once downloaded, you can launch your new instance. The method to connect to the instance using the key pairs differs between Linux and Windows. When connecting to a Windows device, the private key pair is used to decrypt the Windows administrator password. Once you select the appropriate key pair, when trying to connect to your Windows instance, it will provide you with the decrypted Windows password, allowing you to log in:

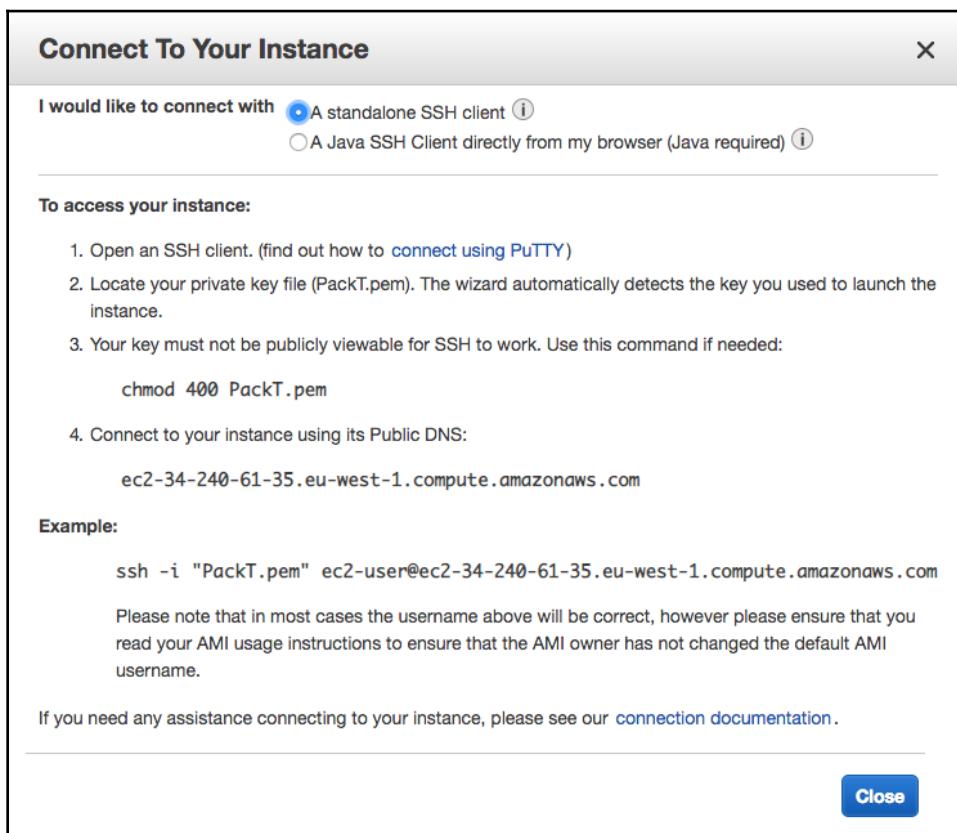


When connecting to a Linux host, you can use your normal SSH client; a common client that many people use is PuTTY. Before you can use your private key, you have to ensure that it is not publicly visible. To do this, run `chmod 400` against your key name; an example is as follows:

```
chmod 400 PackT.pem
```

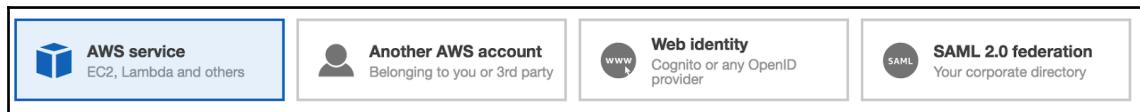
In the preceding command, `PackT.pem` is the downloaded private key name.

AWS will provide you with a connection string to the instance, to make it easier for you when connecting via the AWS Management Console, as you can see in the following screenshot:



IAM roles

IAM roles are a great way to assign permissions to users, applications, and other services, such as EC2, to allow them to perform actions that require authentication and authorization. Roles are assumed, and they can be used to apply temporary credentials to an identity or service:



First, I will focus on using IAM roles with a service, such as EC2. Suppose that you have an application running on an EC2 instance, performing image processing. One process of your application requires read-only access to Amazon S3. One way that developers could architect the application to authenticate to the service would be to store credentials locally, on the instance itself. However, this goes against security best practices, especially within the cloud. You should never store credentials on an instance locally; instead, you should use an IAM role.

An IAM role is configured with permissions, just like a normal IAM user would have. When an EC2 instance is associated with an IAM role, the instance has the permissions that are given to that role.

In our example, we could create a role called `Role_S3_Access`, assign the following permissions, and associate it with the EC2 instance:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3>List*"
      ],
      "Resource": "*"
    }
  ]
}
```

This will allow that instance to access S3 objects within your account, without having to store any local credentials on the instance.

Cross-account roles

Previously, I mentioned that roles can also be associated with users—even users within another AWS account. If you have users that need to access resources within your AWS account from another AWS account, access can be granted through a role configured for **cross-account** access, which that user can then assume. This would allow you to control exactly what that role is authorized to do, and would remove the need to create a separate IAM account for the user in your own corporate account.

When creating a cross-account role, a trust relationship also has to be configured between the two AWS accounts. Let's look at the following example.

Suppose that your organization runs two AWS accounts, prod and dev, and a user in dev needs to temporarily access an RDS database in prod. The IAM administrator in prod can create a cross-account role, `CrossAccountRoleRDS`, to allow for the relevant access.

During the role creation, the administrator will be asked to enter the AWS account ID of dev; dev will be known as the **trusted** account. Meaning prod, the **trusting account**, will be trusting dev to access its resources:

The screenshot shows a form titled "Specify accounts that can use this role". It has an "Account ID*" input field with a placeholder "Account ID" and an "i" icon for help. Below it is an "Options" section with two checkboxes: "Require external ID (Best practice when a third party will assume this role)" and "Require MFA" with an "i" icon.

When the appropriate permissions have been assigned to the role and the role has been created, then an established trust between prod and dev will be created for the use of that role. This trust can be seen within the Management Console, when you look at the details of the role and select the **Trust relationships** tab:

The screenshot shows the "Trust relationships" tab in the AWS IAM Role details page. It has tabs for "Permissions", "Trust relationships" (which is selected), "Access Advisor", and "Revoke sessions". A note says "You can view the trusted entities that can assume the role and the access conditions for the role. Show policy document". The "Edit trust relationship" button is highlighted. The "Trusted entities" section shows "The following trusted entities can assume this role." and lists "The account 582636008125". The "Conditions" section says "The following conditions define how and when trusted entities can assume the role." and notes "There are no conditions associated with this role."

At this stage, the dev account has access to the role prod, but the administrator has to modify the permissions of the user(s) in dev that need to assume that role. To do this, the administrator has to add the following permissions to the user or group, which allows them to assume the role created in prod (with an AWS account ID of 123456789012):

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": ["sts:AssumeRole"],  
        "Resource": "arn:aws:iam::123456789012:role/CrossAccountRoleRDS"  
    }]  
}
```

Web identity and SAML federation

It is also possible to create roles relating to federated access, both or web identity providers and those using SAML federation. This would allow for entities to access your AWS resources temporarily, through the use of roles. We will discuss federated access in the next section.

Federation of access

Federated access to AWS resources allows you to authenticate an identity that doesn't exist within IAM in your AWS account as a user. Sometimes, it's infeasible to have to set up hundreds, or even thousands, of accounts, to allow users to gain permissions to your resources, especially when user accounts may already exist outside of AWS for those users. Wouldn't it be better to allow these users to authenticate to AWS using their existing corporate Microsoft Active Directory accounts, or even their Facebook accounts? Doing so would remove the need for heavy administration within IAM, maintaining and creating all of these additional users.

Web identity federation

Suppose that you have just designed a mobile gaming app, whereby you require the users to create an account, to allow them to post their high scores and achievements, which might be stored within a DynamoDB database within your AWS account. As a part of this process, access to AWS resources will be required, and, as such, authentication has to take place.

It is not possible to create all of these users within IAM; instead, you can use **web identity federation** to supply temporary credentials, which are based on a role with specific permissions.

Web **Identity Providers (IdPs)** are simply other websites that maintain a database of identities that can be used to provide authentication services to other systems that have been configured for federated access. Some common examples of Web IdPs include the following:

- Facebook
- Google
- Amazon

Any site that is **OpenID Connect (OIDC)** compatible can be used as a web IdP.



For more information on OIDC, please refer to <https://openid.net/connect/>.

Amazon Cognito is often used as a link between the web IdP and your AWS resources, to control the authentication process; it can scale to millions of users.



For further information on Amazon Cognito, please refer to <https://aws.amazon.com/cognito/>.

SAML 2.0 federation

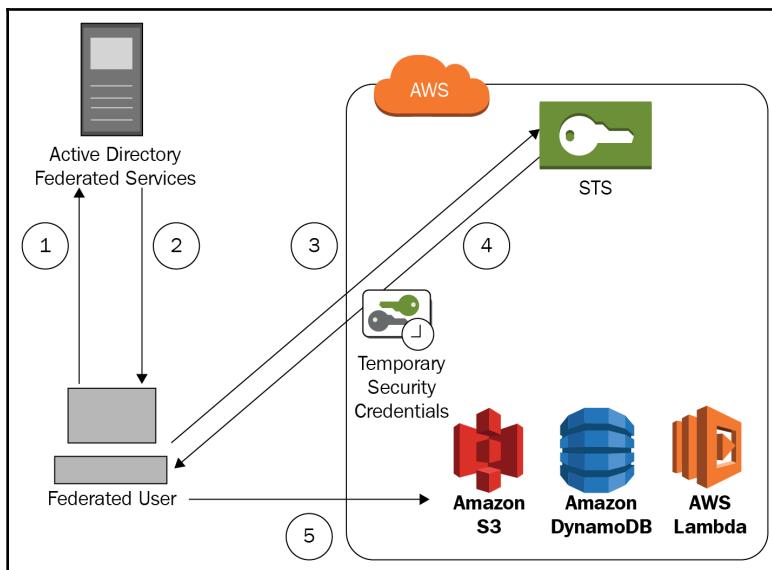
Instead of federating user access through web identity, which is common for mobile apps, you might want to authenticate your internal users to AWS using your own Microsoft Active Directory accounts with **SAML 2.0**.

Security Assertion Markup Language (SAML 2.0) federation allows you to enforce a **single sign-on (SSO)** approach for users, to prevent your IAM administrator(s) from creating hundreds of IAM accounts. It enables the exchange of authentication tokens between two providers: one being the IdP, and the other being the service provider. In this example, MS AD would be the IdP, and AWS would be the service provider. This SSO approach allows your internal corporate users to authenticate and access your AWS resources, including the management console.

I will explain how this approach would work in the following scenario.

Suppose that you have 100 MS Active Directory users internally, and they have to access resources within your AWS account, which include S3, DynamoDB, and Lambda. Instead of setting up 100 IAM user accounts, you want to implement SAML 2.0 federation, leveraging your existing MS AD accounts.

As illustrated in the following diagram, this mechanism can be summarized with five steps:



1. First, the federated user (one of the 100 internal users) authenticates to their internal **AD federated services (ADFS)** server, via an SSO URL.
2. Once they have authenticated with the required Active Directory username and password, SAML sends an assertion back to the client, indicating that authentication was successful and that federated access is required.
3. This successful authentication assertion is then sent to the **AWS Security Token Service (STS)**, using the `AssumeRoleWithSAML` API. This allows it to assume a predefined role within IAM, which would have the permissions required to use S3 DynamoDB and Lambda.
4. A response is then sent back to the authenticated user by the STS service, with temporary credentials and the permissions associated with the role assumed.
5. The user is then authenticated and authorized to access the AWS resources, as required, without ever having to be set up as a user within IAM.

Now that we have covered authentication, I will dive deeper into authorization, and some of the methods used to control this aspect of access control. There are two key services that you need to know about, in order to understand how the authorization process works. These services are IAM and S3.

IAM authorization

To understand authorization within IAM, you will have to look at the different components within the service, as follows:

- Users
- Groups
- Roles
- Identity-based policies

Users

Users are simply IAM objects that reflect the unique identity of someone that requires access to your AWS resources, and they are used as a part of the authentication process, as discussed earlier. The user object can have an associated password, which can be used in conjunction with MFA. Permission policies can be assigned to a specific user, authorizing them to gain access to services and resources; however, this is not considered a best practice. Instead, policies should be assigned to groups.

Groups

Groups are also objects. However, they do not identify with a single identity, like users do. Instead, users are associated to groups, and these groups have associated permissions policies. This means that any user within the group inherits the permissions that are assigned to that group. This makes it far easier, from a management perspective, to control access.

Suppose that you have an operations team of 20 people. If you assigned permissions on a per-user basis, then you would have to assign the relevant policies 20 times. It would be far easier to create an IAM group called `Operations`, assign a single policy with the required permissions to that single group, and then add the 20 users to the group. Should the operations team gain additional responsibilities, and, in turn, require an additional authorization to perform specific tasks, the administrator will only need to update the policy on the group, and not the 20 individual users.

Roles

Roles were covered in the previous section of this chapter, but to recap at a high level, IAM roles allow an identity to assume a set of permissions that are associated to that specific role. These roles can be assumed by another user, which may already have a permission policy assigned to them, either individually or via a group. However, what happens when they temporarily assume the permissions of a role? Are those permissions added to that user's existing policy, or do they replace their permissions? The answer is that they replace their current set of permissions with the policy of the role while they have assumed that role.

Identity-based policies

The policies within IAM are **JavaScript Object Notation (JSON)** based documents, which are used to define permissions that are then associated to different entities, authorizing them to perform specific actions and preventing them from doing other actions within your AWS account. These policies can be associated to users, groups, and roles, and are known as identity-based policies. Establishing a solid understanding of these policies is critical to maintaining secure access to your resources.

The JSON policy itself follows a very specific syntax, and is comprised of a number of parameters; for every policy, there will be a minimum of one statement. The syntax of a typical IAM identity-based policy would contain the following:

- **Version:** This is stipulated by AWS as the version of the policy language. AWS recommends that, as a best practice, you always use the latest version; at the time of writing, it is version 2012-10-17.
- **Statement:** Each statement is comprised of a number of **Statement IDs (Sids)**, which define the permissions set out for the policy; think of this as a container for all of the Sids within that policy.

- **Sid (Statement ID):** Each statement must have at least one Sid. You can name your Sids, to help you identify specific permissions within the policy. Each Sid is comprised of a number of attributes and parameters, which define the permissions set out for the statement of the policy. The following attributes are present within each Sid in a statement:
 - **Effect:** Here, you can select to either Allow or Deny access. This determines the effect of the permissions outlined within the rest of the statement. It's important to remember that any Deny effect will always overrule an Allow effect.
 - **Action:** This attribute dictates what actions are to be allowed or denied (depending on the Effect entry), and it relates to API calls. If more than one action is required, then multiple actions can be listed. For example, the following EC2 actions could be listed under one statement:

```
"Action": [  
    "ec2:AttachVolume",  
    "ec2:DeleteVolume"  
]
```

Within this section, you can also include wildcards to cover a wide range of actions. If you wanted to include all of the actions for EC2 (effectively, all EC2 APIs), you could simply add the Action of the following:

```
"Action": [  
    "ec2:*"  
]
```

- **Resource:** This defines the resource that the Effect and Action apply to. All entries are supplied using the **Amazon Resource Name (ARN)**.



For more information on the AWS ARN syntax and format, please refer to https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-arns.

Again, resources can be used with wildcards; an example is as follows:

```
"Resource": [  
    "arn:aws:ec2:*:*:instance/*",  
    "arn:aws:ec2:*:*:volume/*"  
]
```

- **Condition:** This is an optional attribute that, as the name implies, allows you to apply the permissions specified within the statement, based on a specific condition. Again, as an example (sticking with EC2), you could add the following condition:

```
"Condition": {  
    "IpAddress": {  
        "aws:SourceIp": "10.1.0.0/24"  
    }  
}
```

This condition would ensure that the permissions within the statement would only be applied if the source IP address of the identity was within the network range of 10.1.0.0/24. If the request came from a source IP within this range, the permissions would be applied; if it didn't, then the permissions would not be applied.

So, using the examples covered in the preceding section, the IAM policy would look as follows:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "ec2Permissions",  
            "Effect": "Allow",  
            "Action": [  
                "ec2:AttachVolume",  
                "ec2:DeleteVolume"  
            ],  
            "Resource": [  
                "arn:aws:ec2:*:*:instance/*",  
                "arn:aws:ec2:*:*:volume/*"  
            ],  
            "Condition": {  
                "IpAddress": {  
                    "aws:SourceIp": "10.1.0.0/24"  
                }  
            }  
        }  
    ]  
}
```

This policy can be attached to a user, group, or role, and the identities that are associated to this policy will be able to attach and delete volumes for any EC2 instances and volumes within your AWS account, on the condition that their source IP, when making the request, was in the 10.1.0.0/24 subnet range.

Managed policies versus inline policies

Within IAM, you will see that there is a large amount of predefined policies that have already been created by AWS; at the time of writing this book, there are over 400! These are AWS managed policies that have been built to save you time, as they cover some of the most commonly used permission sets. They are listed alphabetically within IAM, under the **policies** section, and can be searched and filtered. For example, if I were to search for **RDS**, the following results would be displayed:

Filter policies ▾ Q RDS			
	Policy name ▾	Type	Used as
<input type="radio"/>	▶ AmazonRDSBetaServiceRolePolicy	AWS managed	None
<input type="radio"/>	▶ AmazonRDSDirectoryServiceAccess	AWS managed	None
<input type="radio"/>	▶ AmazonRDEnhancedMonitoringRole	AWS managed	None
<input type="radio"/>	▶ AmazonRDSFullAccess	AWS managed	Permissions policy (2)
<input type="radio"/>	▶ AmazonRDSPreviewServiceRolePolicy	AWS managed	None
<input type="radio"/>	▶ AmazonRDSReadOnlyAccess	AWS managed	None
<input type="radio"/>	▶ AmazonRDSServiceRolePolicy	AWS managed	None
<input type="radio"/>	▶ AWSApplicationAutoScalingRDSClusterPolicy	AWS managed	None
<input type="radio"/>	▶ AWSQuickSightDescribeRDS	AWS managed	None
<input type="radio"/>	▶ RDSCloudHsmAuthorizationRole	AWS managed	None

However, when you are creating your policies, you may find that the AWS managed policies don't meet your requirements, and you may need a policy that is more defined, to ensure that you are not using overly permissive policies. As a result, you also have the option to create your own managed policies; these are referred to as customer managed policies. Both AWS managed and customer managed policies can be attached to any user, group, or role.

There are a number of ways to create your customer managed policies, including the following:

- Writing them from scratch with a JSON policy editor
- Using the visual editor within IAM
- Copying an existing managed policy, and customizing it as required

All of these can be carried out from within IAM, by following the steps detailed in the next sections.

Writing policies from scratch by using a JSON policy editor

1. From within the AWS Management Console, select **IAM**.
2. Select **Policies** from the menu.
3. Select **Create policy**.
4. Select the **JSON** tab.
5. Start writing your policy directly into the JSON document, as follows:

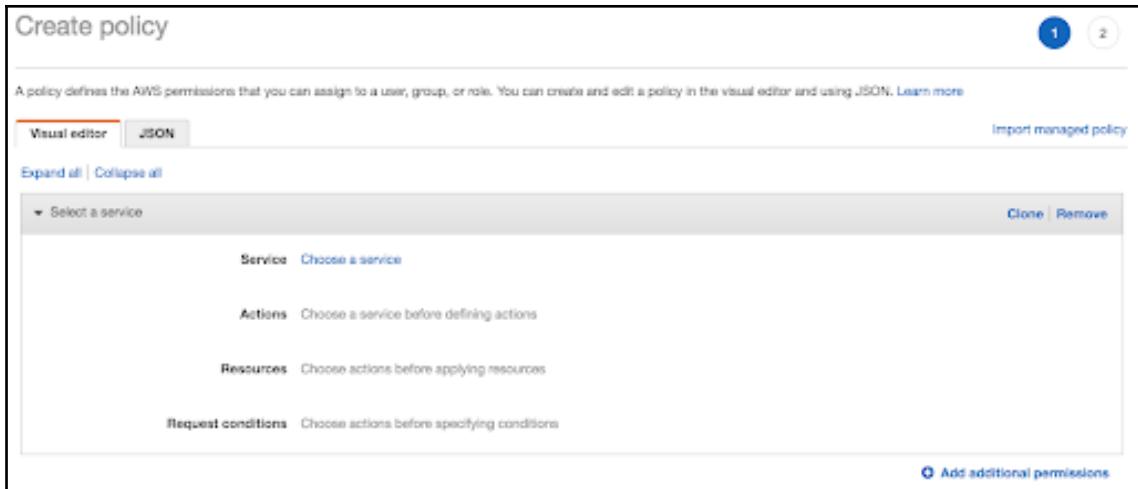
The screenshot shows the 'Create policy' interface in the AWS Management Console. At the top, there are tabs for 'Visual editor' (disabled) and 'JSON' (selected). Below the tabs, a note says: 'A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON.' A 'Learn more' link is provided. To the right of the note are two numbered circles: '1' (blue) and '2' (white). On the far right, there's a 'Import managed policy' button. The main area contains a code editor with the following JSON code:

```
1 - {
2   "Version": "2012-10-17",
3   "Statement": []
4 }
```

6. Select **Review policy**.
7. Add a policy name and description.
8. Select **Create policy**.
9. The policy will then appear as a custom policy, within the policy listings.

Using the visual editor within IAM

1. From within the AWS Management Console, select **IAM**.
2. Select **Policies** from the menu.
3. Select **Create policy**.
4. Select the **Visual editor** tab.
5. Using the corresponding fields and drop-down lists, select the relevant **Services**, **Actions**, **Resources**, and **Request conditions**, as follows:



6. Select **Review policy**.
7. Add a policy name and description.
8. Select **Create policy**.
9. The policy will then appear as a custom policy within the policy listings. If you want to view the JSON document of the policy that you created, you can select it and view the JSON tab of the permissions page.

Copying an existing managed policy

1. From within the AWS Management Console, select **IAM**.
2. Select **Policies** from the menu.
3. Select **Create policy**.
4. Select **Import managed policy**, in the top-right of the screen.
5. Select the managed policy that you would like to import.
6. Select **Import**.
7. You can then add additional permissions by using the visual editor, or switch to the **JSON** tab and edit the policy directly from there.
8. When you are happy with the modifications, select **Review policy**.
9. Add a policy name and description.
10. Select **Create policy**.
11. The policy will then appear as a custom policy within the policy listings.

Inline policies

Inline policies differ from managed policies, in that the policy itself is embedded directly into the object that is going to be using it (the user, group, or role). This means that you can edit any user, group, or role, and create a policy directly into that IAM object. Inline policies do not appear within the list of managed policies, as they are specific to the object that they are embedded within.

To add an inline policy, select the user, group, or role from within the AWS Management Console, and select the **Permissions** tab of that object. From there, select **Add in-line policy**.

Summary

This chapter has focused on some of the key authentication, authorization, and access control mechanisms that are used when controlling access to your resources.

Understanding the access control process and how to implement specific controls is a must for all security architects and engineers. AWS provides solid security mechanisms, with the ability to enforce stringent access security controls, but it's down to us, the users of AWS, to implement and architect those policies.

Before implementing an access control strategy, you must have an awareness of all of the available options, to determine the best course of action for your organization and environment.

Further reading

- **Security Pillar AWS Well-Architected Framework:** <https://d1.awsstatic.com/whitepapers/architecture/AWS-Security-Pillar.pdf>
- **AWS IAM FAQs:** <https://aws.amazon.com/iam/faqs/>
- **IAM JSON Policy Conditions:** https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition.html

14

Encryption and Key Management

A key element of AWS security is the ability to be able to encrypt sensitive and confidential data across different services, helping to ensure it's protected from those who should not have access to the plaintext data. Understanding different encryption methods within these services allows you to maintain the confidentiality of the data.

This chapter will focus on a number of key services and the encryption options that are available to use. As a part of this, the **Key Management Service**, known as **KMS**, will also be discussed in detail. This service allows you to control and manage encryption keys which can either be imported from your own KMS system or those that are generated by AWS itself. The KMS service is also integrated with many other AWS services.

In this chapter, we will cover the following topics:

- An overview of encryption
- EBS encryption
- S3 encryption
- RDS encryption
- Key Management Service

Technical requirements

To gain the most from this chapter, you should have a basic understanding of the following services. Since this chapter covers encryption, the following AWS services will be mentioned throughout:

- **Elastic Block Store (EBS)**
- **Amazon Simple Storage Service (S3)**
- **Relational Database Service (RDS)**
- IAM policies

An overview of encryption

Before we get deeper into the different techniques and methods used by AWS to encrypt your data, an overview and an understanding of the different key cryptography mechanisms is needed.

Any data that has not been encrypted is known as **plaintext**, which simply means that the data is in a readable format without the need for any mathematical intervention to alter the data before it can be read. When data is in a state of plaintext, anyone who has read access to the data can access it and view the information contained within it. As long as this data is not sensitive or contains confidential information, then this unencrypted data can remain unencrypted. However, if the data IS sensitive, such as containing customer details and information, then there will be a requirement to protect and secure this data as a priority.

Sensitive data must be encrypted, since this is a basic level of security best practice; but what exactly is encryption and how does it affect and alter the data?

When data is encrypted, it is no longer in a state of plaintext. Instead, the data is known as **ciphertext**. The encryption process itself uses a mathematical algorithm to change the plaintext into ciphertext. This essentially becomes a scrambled string of characters, making the data unreadable by both people and systems using the data who do not have the appropriate *key* to revert the ciphertext back into plaintext.

Key encryption is based on keys that are used to both encrypt and decrypt your data. However, one method uses a single key to both encrypt and decrypt that same data (symmetric cryptography) while another method uses two separate keys, one to encrypt and another to decrypt (asymmetric cryptography).

Symmetric key cryptography

Symmetric key cryptography uses only a single key within the encryption and decryption process. As a result, if you were to encrypt your data using symmetric encryption, you would use a specified key to encrypt your data, turning it from a readable plaintext format into an unreadable ciphertext format. When you then need to gain access to the information with the data, you would then have to use that very same key that you used to encrypt it in order to decrypt it. You can think of it as the key to your front door. When you leave the house, you use your key to lock the door; when you return home to gain access to your house again, you use the **same key** to gain entry.

There is a small security concern with this method, however. If you encrypted the data and someone else then needed to gain access to the same data, they would need to be in receipt of the same key. There are many different methods that you could use to share this key with someone else, and here lies the issue: if this key was intercepted by a malicious user during this key transfer, then that malicious user could then access your data.

Following on from the previous analogy, let's say that you left your house and you locked the door. You then left this key under a plant pot outside the front door for your partner who didn't have a key, allowing them to gain access prior to you coming back home. If someone saw you leaving the key under the plant pot, that someone could then gain access to your house.

There are a number of common symmetric algorithms that are used in this method, but the one used most often within AWS is called the **Advanced Encryption Standard (AES)**.

Asymmetric key cryptography

The difference with asymmetric key cryptography is that there are **two keys** used in the encryption and decryption process. One key is used for the encryption process, and the second key is used to decrypt the data. This removes the security flaw mentioned in the *Symmetric key cryptography* section, as you no longer have to send the same key to someone else to decrypt the data.

Both of these keys are generated at the same time and are linked mathematically through an encryption algorithm, effectively joining them as a pair. One key within this pair is considered a public key and the other key is a private key. This private key should be kept by a single party and never shared with anyone else. The public key can be shared with anyone and everyone; it doesn't matter who has access to that key—it's public for a reason. It's important to note that to decrypt data, both of the keys are required.

At a high level, the process for asymmetric encryption would work as follows. Suppose that a third party wanted to send you an encrypted message or data of some sort. They would use your public key to encrypt the message. The message would then be sent to you in an encrypted format, preventing it from being easily viewable while in transmission. When you receive the message, you would then use your private key. Only you have access to the private key in conjunction with the public key in order to decrypt the message from ciphertext back into plaintext using the shared mathematical algorithm.

EBS encryption

The **Elastic Block Store (EBS)** service is AWS's answer to block-level storage and is used as persistent, reliable, and highly available storage that is attached to your EC2 instances. EBS offers built-in encryption options that allow you to easily implement encryption for data stored on these volumes. Due to the importance and criticality of data protection, EBS supports encryption across all EBS volume types and ensures that there is no negative performance impact relating to your IOPS of the volume. Do be aware, however, that although encryption is possible across all volume types, it is not available across all instance types within those volume types.



For an up-to-date list of available instances, please go to the following link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html#EBSEncryption_supported_instances.

One key point in understanding the encryption process of EBS volumes is that any operations relating to encryption are actually carried out on the EC2 instance that the volume is attached to rather than the volume itself. This measure was put in place to ensure that the data remains encrypted in transit (between the EC2 instance and EBS volume) and while at rest.

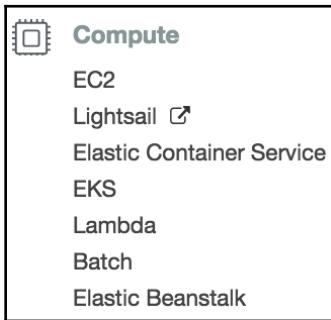
The encryption mechanism on the EBS volume is fully managed and controlled by the service; there is minimal effort required to configure it, and when the volume is encrypted, the sending and receiving of data between the volume and the attached EC2 instances is totally transparent.

There are a couple of different ways to encrypt an EBS volume, and it will depend on how and when you are creating/encrypting your volume.

Encrypting a new EBS volume

If you are encrypting a new EBS volume from scratch, you can implement the encryption of that volume by performing the following steps:

1. Select the **EC2** service under the **Compute** category within the AWS Management Console, as shown in the following screenshot:



2. Select **Volumes** from under the **Elastic Block Store** menu:



3. Click **Create Volume**. You will see something like the following:

The screenshot shows the EC2 Dashboard with the 'Create Volume' button highlighted. Below it is a table listing three existing EBS volumes:

Name	Volume ID	Size	Type	IOPS	Snapshot	Created	Availability Zone	State
Private Wind...	vol-0835051...	30 GiB	gp2	100 / 3000	snap-05a6ca8...	May 15, 2018 at 1:4...	eu-west-1b	in-use
Logging Server	vol-0bf4bc63...	8 GiB	gp2	100 / 3000	snap-03cb10ef...	May 4, 2018 at 5:13...	eu-west-1b	in-use
	vol-09c4d62...	10 GiB	gp2	100 / 3000		March 16, 2018 at 1...	eu-west-1a	available

4. Input your configuration details for the EBS volume.
5. Select the **Encrypt this volume** checkbox.
6. From the dropdown list next to **Master Key**, select the **(default) aws/ebs** option:

The screenshot shows the 'Encryption' configuration form. It includes a checkbox labeled 'Encrypt this volume' and a dropdown menu for 'Master Key'. The dropdown menu has a 'Filter by attributes' search bar and a list containing '(default) aws/ebs', which is highlighted.

You will notice that this adds additional details to the form:

- A **KMS Description**, providing a description of the key selected. This default key is the key used by KMS to encrypt EBS volumes.
- The **KMS Account**, which will show the AWS account from which the KMS key was used when you selected the **Master Key**.
- The **KMS Key ID**, which is a unique identifier of that particular key.
- The **KMS Key ARN**.

Upon completion of the newly created volume, any data then saved to this volume will be encrypted using the **Advanced Encryption Standard (AES-256)** algorithm. As we learned in the previous section, AES is an asymmetric encryption algorithm.

Encrypting a new EBS volume during the launch of a new EC2 instance

1. Select the EC2 service under the **Compute** category from within the AWS Management Console and click **Launch Instance**.
2. Select your desired AMI type.
3. Select your instance type.
4. Click **Next: Configure Instance Details**.
5. Enter the required configuration options for your instance.
6. Click **Next: Add Storage | Add New Volume | EBS**.
7. In the **Encrypted** column, select the dropdown box of the newly created volume and select the **(default) aws/ebs** KMS Key:

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted
Root	/dev/xvda	snap-06311f1d47660fd11	8	General Purpose SSD (GP2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted
EBS	/dev/sdb	Search (case-insensit)	8	General Purpose SSD (GP2)	100 / 3000	N/A	<input type="checkbox"/>	Not Encrypted

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

KMS Key Aliases KMS Key ID

KMS Key Aliases	KMS Key ID
Not Encrypted	
(default) aws/ebs	68a0a823-a7d5-4c17-8b42-fbd83739c45e

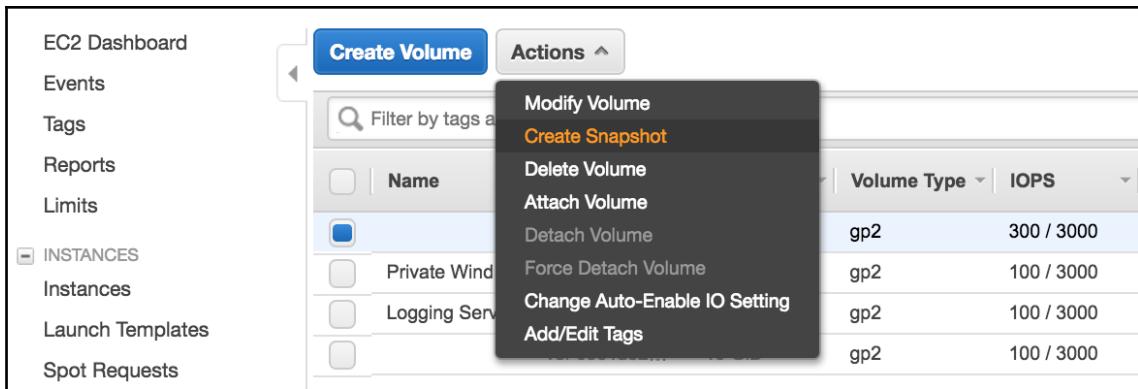
Once the encryption options have been set for the new volume, continue with the creation of your instance as normal.

Encrypting an existing EBS volume

If you have sensitive data being stored on an EBS volume that isn't currently encrypted, then you may need to encrypt the volume. To do so, you can follow these steps:

1. Choose your EBS volume from within the EC2 dashboard of the Management Console.

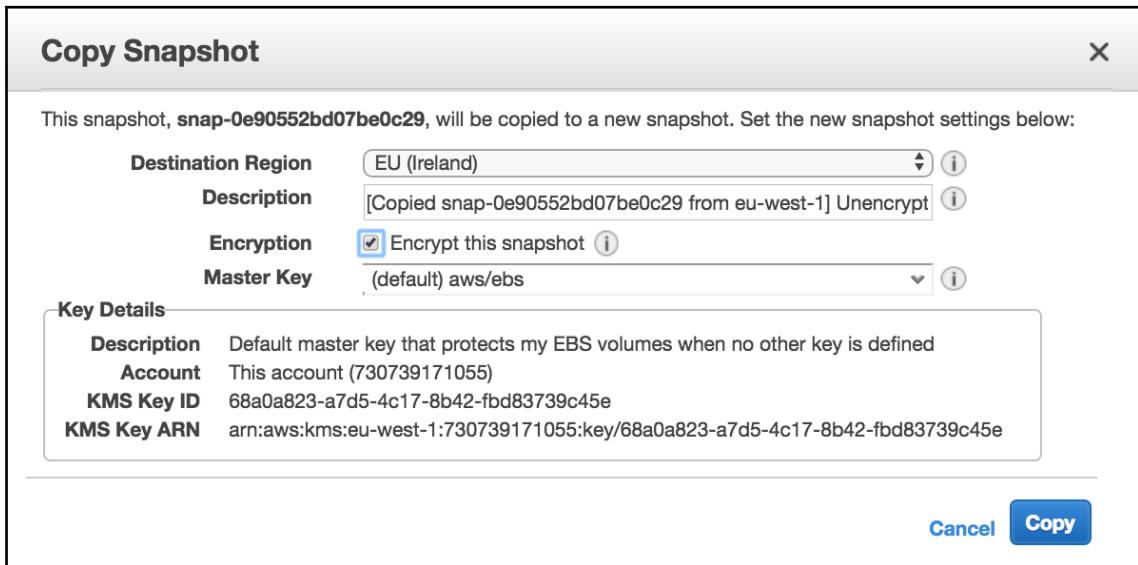
2. Select Actions | Create Snapshot:



3. Add a description for your snapshot and click on Create Snapshot:

The screenshot shows the 'Create Snapshot' dialog box. It has fields for 'Volume' (set to vol-04ad5ab560a2b40d5), 'Description' (set to 'Unencrypted'), and 'Encrypted' (set to 'Not Encrypted'). Below these are 'Key' and 'Value' fields for tags. A note says 'This resource currently has no tags' and 'Choose the Add tag button or click to add a Name tag'. At the bottom, there's an 'Add Tag' button, a note about 50 remaining tags, and 'Cancel' and 'Create Snapshot' buttons.

4. When the snapshot has been created, select **Snapshots** under the **Elastic Block Store** menu on the left-hand side of the console.
5. Find your snapshot and select it.
6. Click **Actions** | **Copy** and check the box for **Encryption**.
7. Select the **(default) aws/ebs Master Key** and click **Copy**:



At this point, you will now have an encrypted version of your snapshot, which will be of the original unencrypted volume. Next, you need to create a new volume from this newly created encrypted snapshot:

1. Select the newly encrypted snapshot.
2. Select **Actions | Create Volume**:

The screenshot shows the 'Create Volume' page in the AWS Snapshots interface. At the top, it displays the Snapshot ID: snap-077216f02dddec6a1. Below this, the 'Volume Type' is set to 'General Purpose SSD (GP2)'. The 'Size (GiB)' is set to 100. Under 'IOPS', it shows 300 / 3000 with a note about baseline and burstable IOPS. The 'Availability Zone*' is set to eu-west-1a. The 'Throughput (MB/s)' is listed as 'Not applicable'. The 'Encryption' option is set to 'Encrypted'. In the 'Tags' section, there is a note that the resource currently has no tags, and a link to 'Add tag' or 'click to add a Name tag'. There is also a note that up to 50 tags can be added. At the bottom right, there are 'Cancel' and 'Create Volume' buttons.

You can see that you no longer have the option to change the encryption setting. This is automatically set to **Encrypted** under the **Encryption** option. This is because when you are creating a new volume from an encrypted snapshot, that new volume will also be encrypted.

Amazon S3 encryption

Amazon S3 provides an object-level storage solution, allowing you to save objects up to 5 terabytes in size. Being a storage solution, and one of the most commonly used storage services within AWS, S3 provides a variety of encryption mechanisms to suit different requirements and compliance concerns.

There are five different encryption options available to encrypt your S3 objects, as follows:

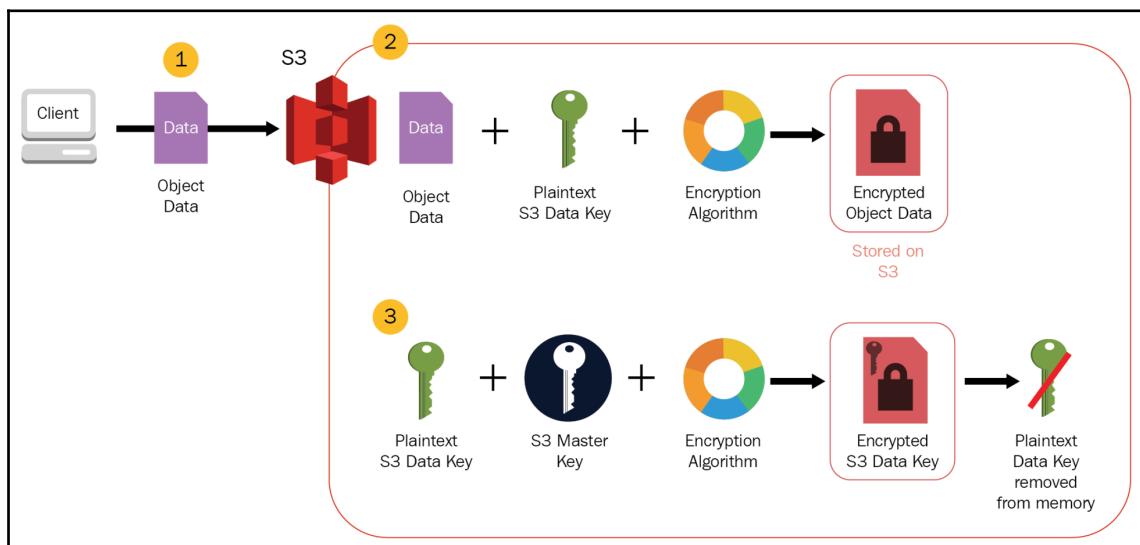
- Server-side encryption with S3 managed keys (SSE-S3)
- Server-side encryption with KMS managed keys (SSE-KMS)
- Server-side encryption with customer-managed keys (SSE-C)
- Client-side encryption with KMS managed keys (CSE-KMS)
- Client-side encryption with customer-managed keys (CSE-C)

The difference between server-side and client-side encryption is fairly simple. With server-side encryption, the encryption algorithm and process is run from the server-side—in this instance, within S3. Client-side encryption means that the encryption process is executed on the client first before the data is sent to S3 for storage.

I now want to take a closer look at each of these methods and the exact process that takes place for both the encryption and decryption actions.

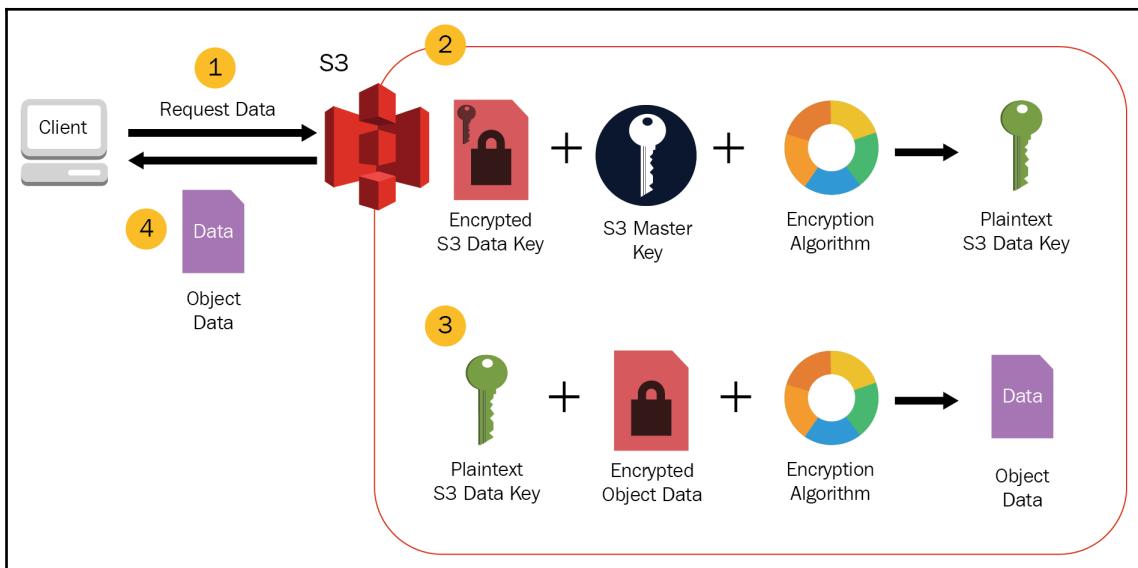
Server-side encryption with S3 managed keys (SSE-S3)

- Encryption:



1. The client selects their object(s) to upload to S3 and indicates the encryption mechanism of SSE-S3 during this process.
2. S3 then takes control of the object and encrypts it with a plaintext data key that's generated by S3. The result is an encrypted version of the object, which is then stored within your chosen S3 bucket.
3. The plaintext data key that used to encrypt the object is then encrypted with an S3 master key, resulting in an encrypted version of the key. This now encrypted key is also stored on S3 and has an association to the encrypted data object. Finally, the plaintext data key is removed from memory in S3.

- **Decryption:**

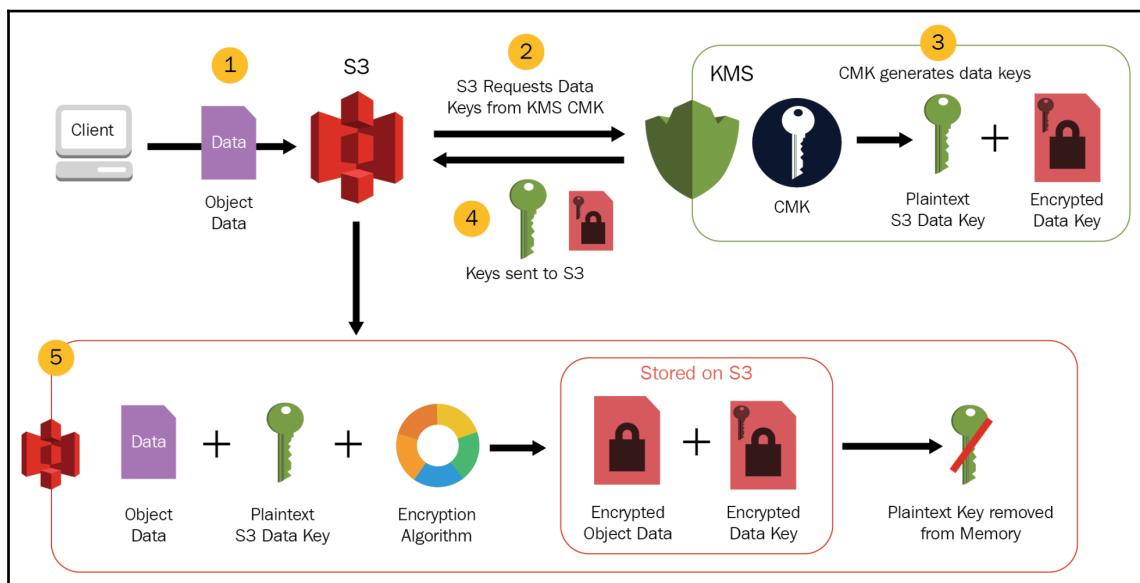


1. A user requests access to the encrypted object via a client
2. S3 is aware that the requested object is encrypted and so takes the associated encrypted data key of the object and uses the S3 master key to decrypt the data back into a plaintext data key
3. This plaintext data key is then used to decrypt the encrypted data object to produce a plaintext version of the object
4. When the object is decrypted, S3 then returns the data object to the client

As you can see, the encryption process is completely transparent to the user and they are not required to interact with S3 in a different way. The same access method is used and all encryption processes are handled by S3, as long as the user requested access has the required permissions to the data object in an encrypted form.

Server-side encryption with KMS managed keys (SSE-KMS)

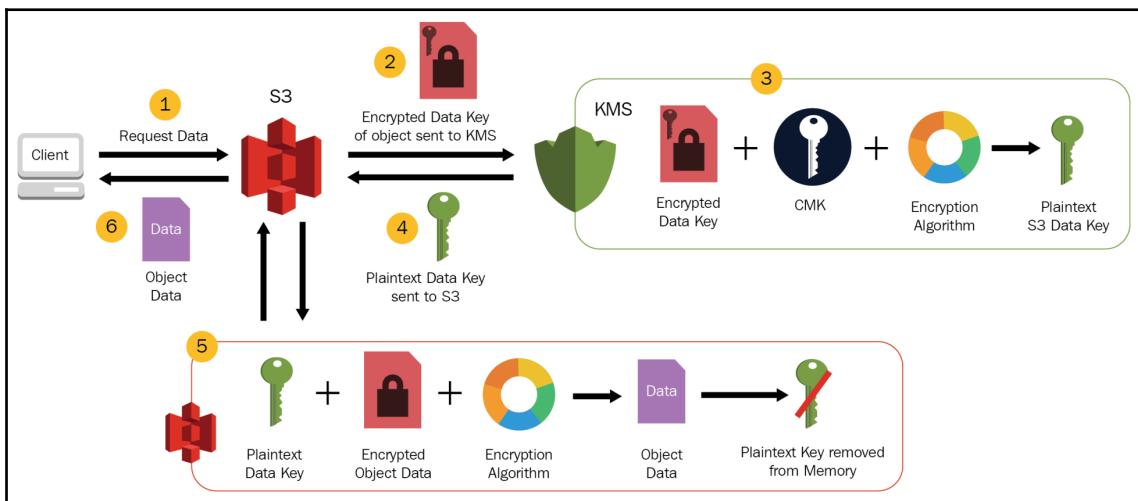
- Encryption:



1. The client selects their object(s) to upload to S3 and indicates the encryption mechanism of SSE-KMS during this process with an associated **customer master key (CMK)**.
2. S3 responds by requesting data keys from KMS to allow S3 to encrypt the data submitted by the client.
3. Using the CMK selected during step 1, The **Key Management Service (KMS)** will then generate two data keys—a plaintext data key and an encrypted version of that same data key.

4. KMS will then send both of these keys back to S3 to allow S3 to perform the encryption.
5. At this stage, S3 then encrypts the object data with the plaintext version of the data key and stores the resulting encrypted object alongside the encrypted version of the data key that was also received by KMS. The plaintext data key is then deleted from memory.

- **Decryption:**

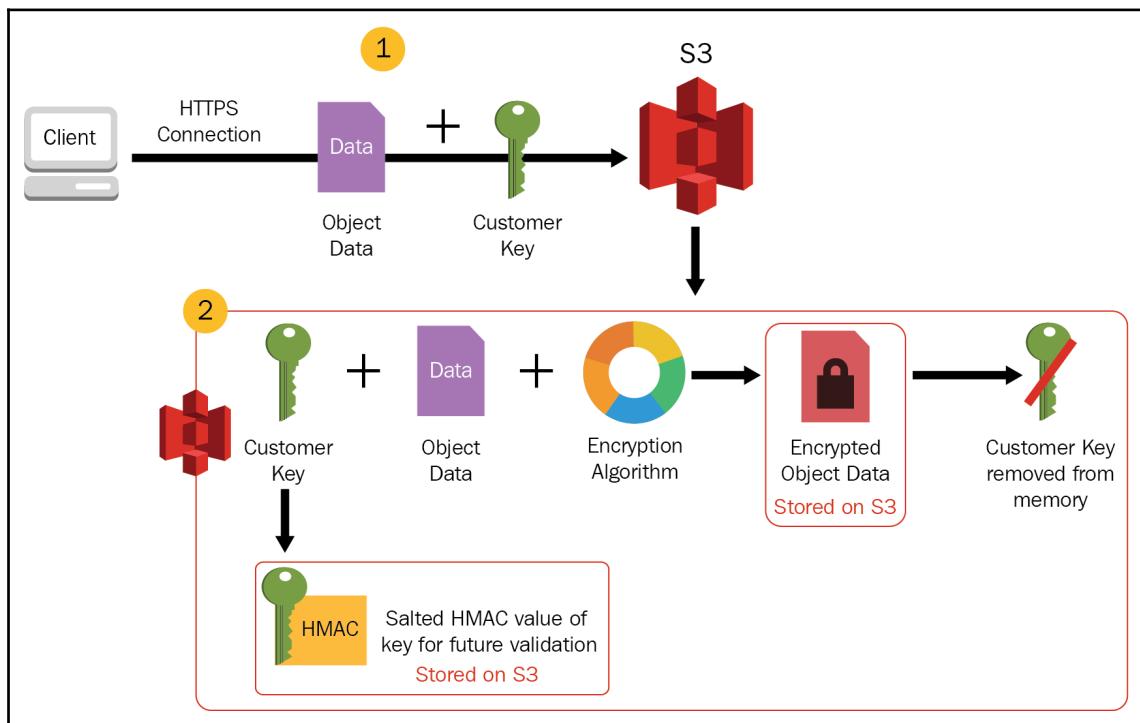


1. A user requests access to the encrypted object on S3 via a client.
2. S3 is aware that the object is encrypted and has the encrypted data key that is associated with the object. S3 sends this encrypted data key back to KMS.
3. On receipt of this encrypted data key, KMS uses the original CMK to decrypt the data key, thus generating a plaintext version of the data key.
4. The plaintext data key is then returned to S3.
5. Using the plaintext data key, the encrypted object data can then be decrypted, returning a plaintext version of the object data. Then, the plaintext data key is removed from memory with S3 one more.
6. Finally, the plaintext object can then be sent back to the client who requested the object.

Similarly to SSE-S3, this process is also transparent to the end client, but again, they can only access the object if the required permissions are in place.

Server-side encryption with customer managed keys (SSE-C)

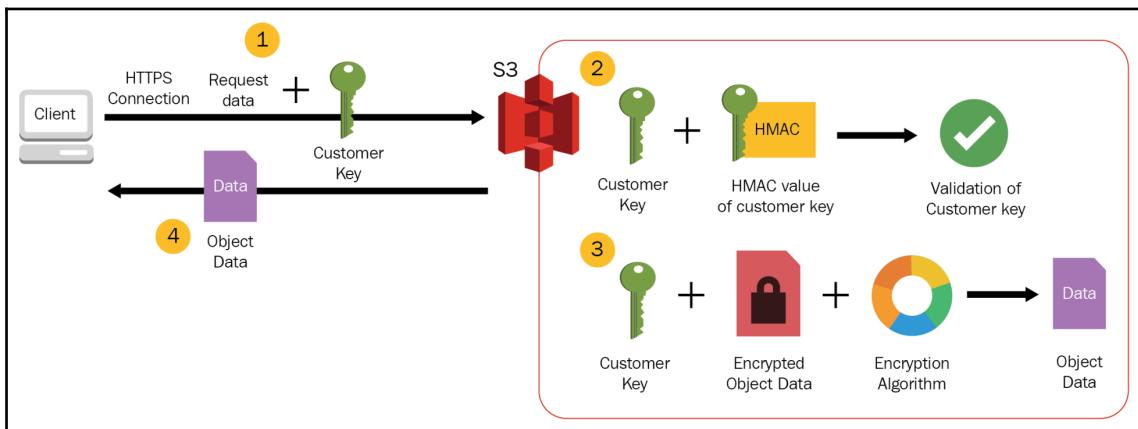
- Encryption:



1. The client uploads the object(s) to S3, along with the customer provided key across a HTTPS connection. If SSE-C is being used and an HTTPS connection is not used during the uploads, then it will fail and reject the communication. The channel needs to be encrypted as the key is being sent along with the object.

2. S3 will then take the customer provided key and the object and perform the encryption of the object. In addition to this, S3 will generate a salted HMAC value of the customer key to enable the validation of future access requests. This HMAC and the encrypted object are then stored on S3 with an association to each other. Again, the plaintext customer provided key is then removed from memory.

- **Decryption:**



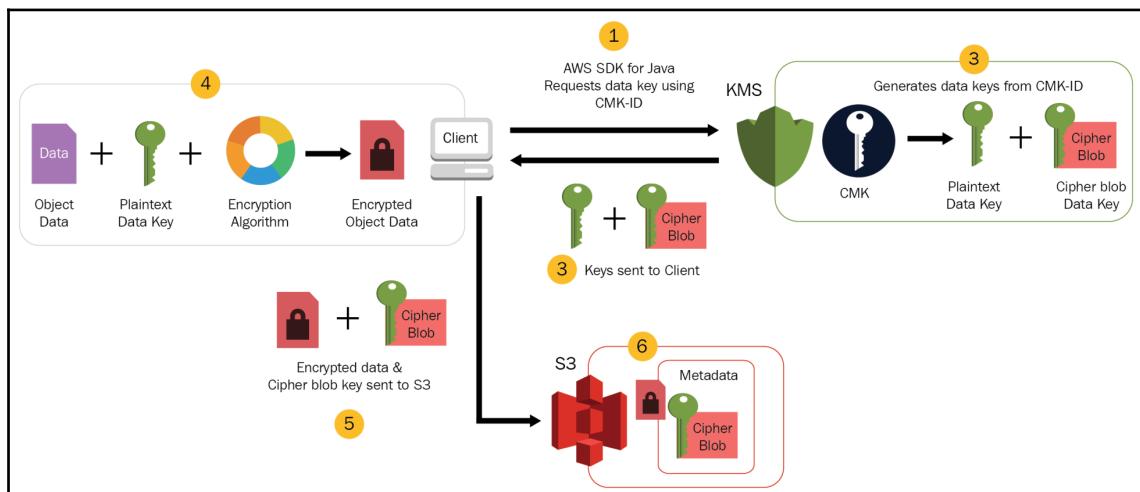
1. A user requests access to the encrypted object on S3 via a HTTPS connection. With this same request, the customer key is also sent to S3.
2. S3 uses the stored HMAC value of the key to validate and confirm that the key that has been sent is the correct key.
3. Upon successful validation, the customer key is then used to decrypt the object data.
4. The plaintext version of the object data is then sent back to the client.

Here, the process was slightly different to SSE-S3 and SSE-KMS, since the key is sourced from the client rather than by S3. As a result, you need to include encryption with your request by using a number of different request headers in the AWS SDK. These request headers are as follows:

- **Header:** `x-amz-server-side-encryption-customer-algorithm`: Used to specify the encryption algorithm. The header value here must be `AES256`.
- **Header:** `x-amz-server-side-encryption-customer-key`: This header is used to provide a 256-bit, base64-encoded encryption key for S3 during both the encryption and decryption operations of your objects.
- **Header:** `x-amz-server-side-encryption-customer-key-MD5`: This header provides a base64-encoded, 128-bit MD5 digest of your encryption key. This is used by S3 as a form of integrity check to make sure that the encryption key wasn't tampered with or experienced any errors during transmission.

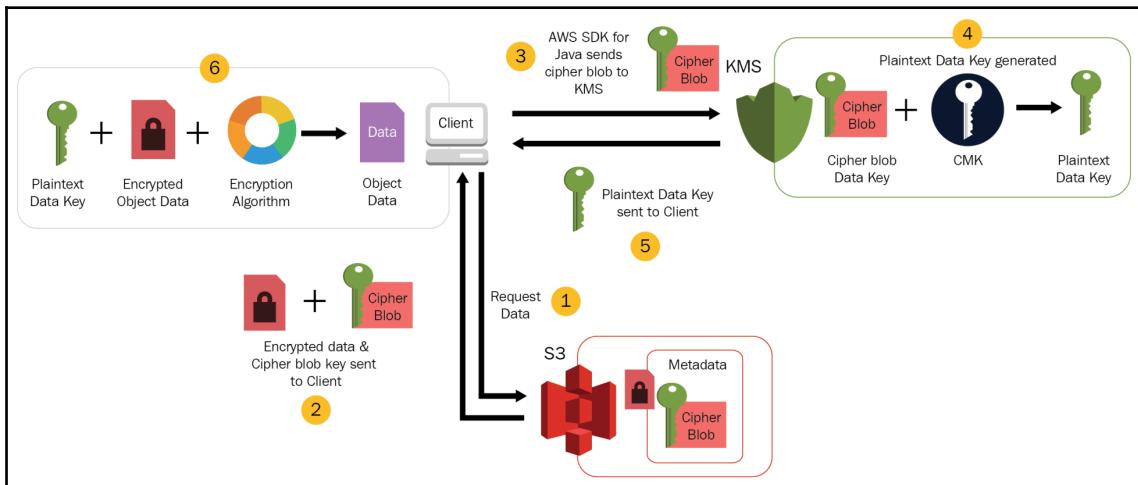
Client-side encryption with KMS managed keys (CSE-KMS)

- **Encryption:**



1. The client will use an AWS SDK, and in this example, the Java client, which will request data keys from KMS using a specified CMK,
2. Using the CMK selected during *step 1*, the **Key Management Service (KMS)** will then generate two data keys—a plaintext data key, and another key that will be a cipher blob of the first key.
3. KMS will then send these keys back to the requesting client.
4. The client will perform the encryption against the object data with the plaintext version of the data key and store the resulting encrypted object.
5. The client then uploads the encrypted object data and the cipher blob version of the key created by KMS to S3.
6. The final stage involves the cipher blob key being stored as metadata against the encrypted object that's maintaining a linked association.

- **Decryption:**



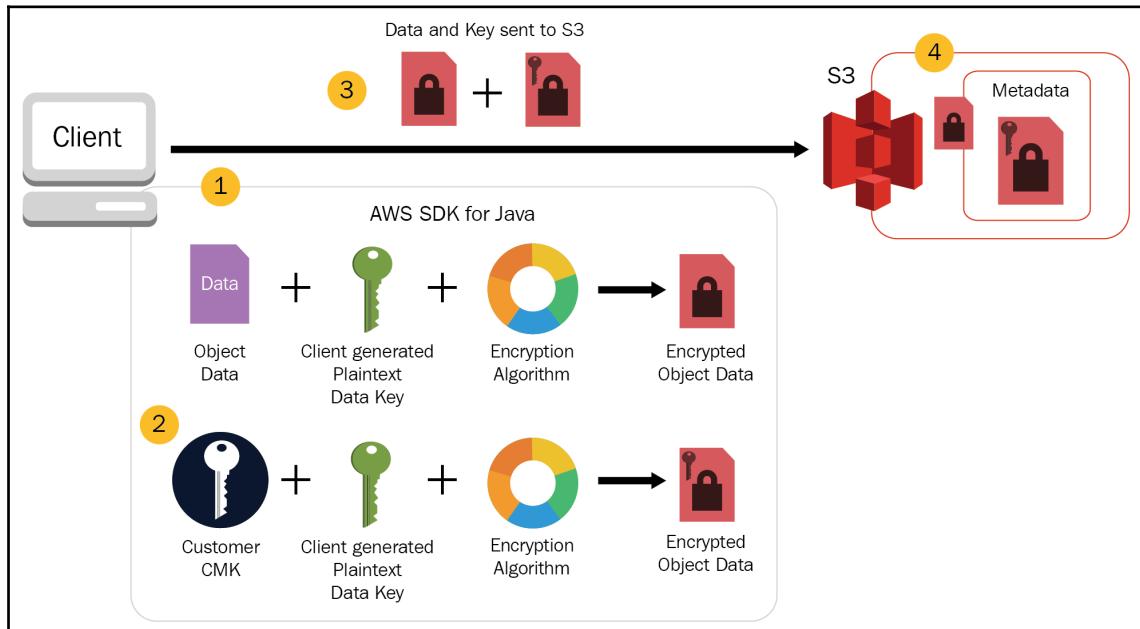
1. A user requests access to the encrypted object on S3.
2. The encrypted object is retrieved from the S3 bucket and sent back to the client, along with the associated cipher blob key.
3. The Java client will then send the cipher blob back to KMS to generate a plaintext data key.

4. KMS uses the original CMK that was used during the encryption process, along with the cipher blob, to create and generate a plaintext version of the data key.
5. KMS then sends this plaintext data key back to the requesting Java client.
6. Once the Java client has received the plaintext key, it can then use this to decrypt the object that it has already received from S3.

You will have noticed that all of the encryption operations were conducted on the client, using the AWS SDK. At no point did S3 perform any encryption operations. Even when the object was requested to be accessed again, S3 simply sent the encrypted data object back to the client to allow the client to perform the necessary steps.

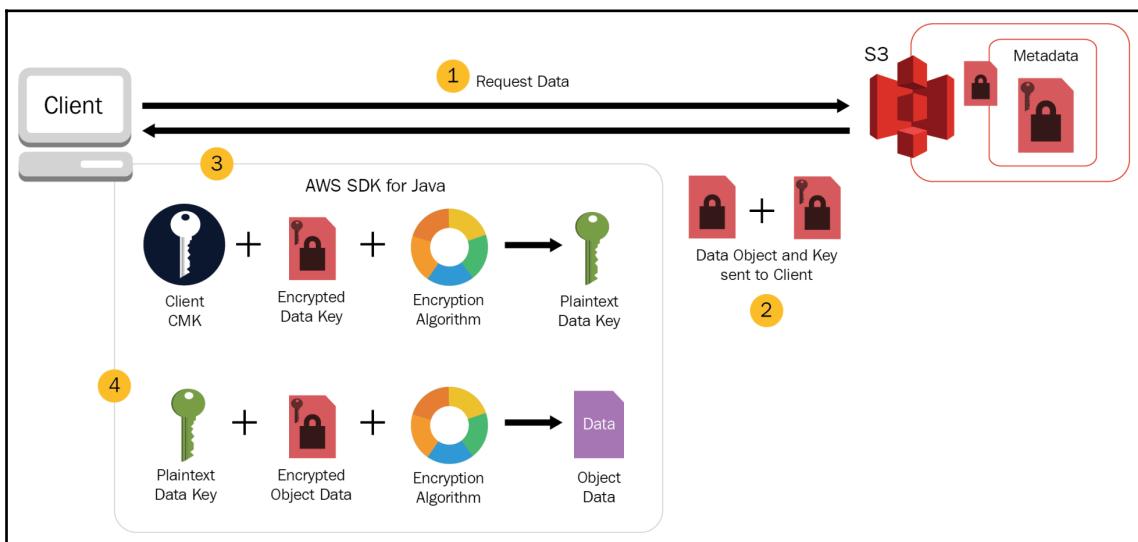
Client-side encryption with KMS managed keys (CSE-C)

- Encryption:



1. The client will use an AWS SDK, and in this example, the Java client, which will create a randomly generated plaintext data key, which is then used to encrypt the object data
2. A CMK created by the customer then encrypts this plaintext data key
3. At this point, the encrypted data key and the encrypted object data is sent from the client to S3 for storage
4. S3 then takes the encrypted data key and associates it with the encrypted object and stores both on S3

- **Decryption:**



1. A user requests access to the encrypted object on S3.
2. S3 responds by sending the requested object data, along with the associated encrypted data key, back to the client
3. Using the AWS SDK, the customer's CMK is then used with the encrypted data key to generate a plaintext version of that same data key
4. The encrypted object can then be decrypted using the plaintext data key, and its contents can be accessed

As with CSE-KMS, you can see here that all encryption/decryption operations are handled by the client itself. The server (S3) is not involved with this process other than storing the encrypted data which is sent to it.

RDS encryption

When using Amazon RDS, there may be times when the data held within your database needs to be encrypted due to its sensitivity. When RDS encryption is enabled, which uses the AES-256 algorithm, it ensures that all underlying storage that's used is encrypted, along with all associated read-replicas, automated backups, and snapshots, following the enablement without any further configuration needed.

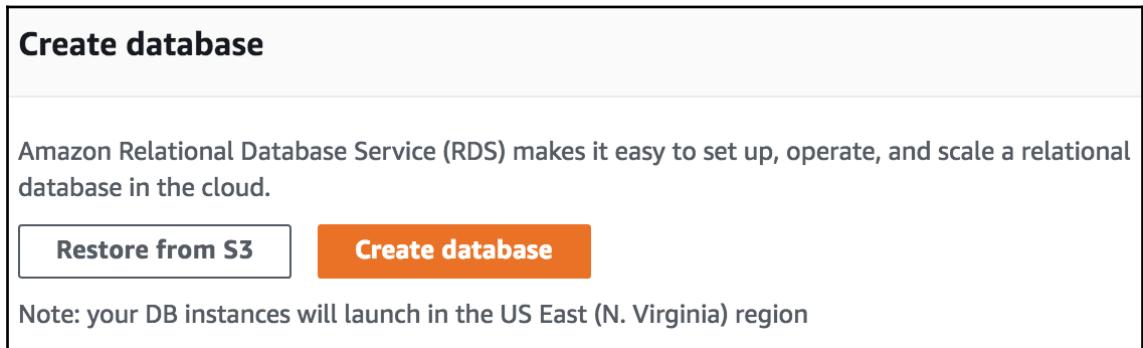
RDS encryption is offered at rest and is easily enabled by using the following database engines:

- Amazon Aurora
- MySQL
- MariaDB
- PostgreSQL
- Oracle
- Microsoft SQL Server

How to enable encryption

As an example, to configure encryption for Amazon Aurora, you can perform the following steps:

1. Select Amazon RDS from the AWS Management Console under the Database category.
2. Select **Create database** from the central window:



3. Select **Amazon Aurora** from the available engine types and the relevant **Edition** and click **Next**:

Amazon Aurora

Amazon
Aurora

Preview - Parallel Query

Amazon
Aurora

MySQL


MariaDB


PostgreSQL


Oracle


Microsoft SQL Server


Amazon Aurora

Amazon Aurora is a MySQL- and PostgreSQL-compatible enterprise-class database, starting at <\$1/day.

- Up to 5 times the throughput of MySQL and 3 times the throughput of PostgreSQL
- Up to 64TiB of auto-scaling SSD storage
- 6-way replication across three Availability Zones
- Up to 15 Read Replicas with sub-10ms replica lag
- Automatic monitoring and failover in less than 30 seconds

Edition

MySQL 5.6-compatible

MySQL 5.7-compatible

PostgreSQL-compatible

4. Select your required DB instance class under **Instance specifications**:

Instance specifications

Estimate your monthly costs for the DB Instance using the [AWS Simple Monthly Calculator](#).

DB engine

Aurora - compatible with MySQL 5.6.10a

DB instance class [Info](#)

db.t2.small — 1 vCPU, 2 GiB RAM

▼

Multi-AZ deployment [Info](#)

Create Replica in Different Zone

No

5. Enter your **DB Instance Identifier**, **Master username**, and **Master password**, and then click **Next**:

Settings

DB instance identifier [Info](#)
Specify a name that is unique for all DB instances owned by your AWS account in the current region.

DB instance identifier is case insensitive, but stored as all lower-case, as in "mydbinstance". Must contain from 1 to 63 alphanumeric characters or hyphens (1 to 15 for SQL Server). First character must be a letter. Cannot end with a hyphen or contain two consecutive hyphens.

Master username [Info](#)
Specify an alphanumeric string that defines the login ID for the master user.

Master Username must start with a letter. Must contain 1 to 16 alphanumeric characters.

Master password [Info](#) **Confirm password** [Info](#)

Master Password must be at least eight characters long, as in "mypassword". Can be any printable ASCII character except "/", "", or "@".

[Cancel](#) [Previous](#) [Next](#)

6. On this screen, you will be able to configure your security settings, database options, encryption, failover, backup, backtrack, monitoring, logging, and maintenance. This is the page where your encryption details are defined. Find the **Encryption** options section, as shown in the following screenshot, and select **Enable encryption**:

Encryption

Encryption

Enable encryption [Learn more](#)

Select to encrypt the given instance. Master key ids and aliases appear in the list after they have been created using the Key Management Service(KMS) console.

Disable encryption

Master key [Info](#)

(default) aws/rds ▾

Description	Account	KMS key ID
Default master key that protects my RDS database volumes when no other key is defined	730739171055	3629c93d-08e3-43da-9d8b-4b5b14c810cc

7. Using the dropdown box under **Master key**, you can select to use an alternate KMS key instead of the **(default) aws/rds** key. This default key will automatically be generated by AWS KMS when it is first required.
8. Continue to configure the database as required and your Aurora database will then have all of its storage encrypted.

When encryption is enabled on a database, it is completely transparent to the end user and only has a very minimal performance impact.

When creating your RDS database, you need to understand what data you will be storing within it to ascertain if you will need the data encrypted. Unfortunately, you are not able to enable encryption of a database after it has been created; you can only enable encryption during its creation. However, there is a method to encrypt an existing unencrypted database through the use of snapshots.

Steps to encrypt an existing database

Perform the following steps to encrypt an existing database:

1. You must first take a snapshot of your existing unencrypted database
2. Using this snapshot, you can then create a copy of it and enable encryption on the copy
3. Once you have the encrypted copy of the snapshot, you can create a new database from it
4. The new database generated from the encrypted snapshot will now also be encrypted

When using encryption within RDS, it is recommended that you also enable backups, and back up frequently. The reason for this is that your database could be changed into a terminal state if the KMS CMK selected during the encryption process is disabled. When in a terminal state, the database is no longer recoverable. AWS RDS will not be able to read or write to the database without having access to the KMS Key. If your RDS instance did enter a terminal state, then the only solution to gain access to the database again would be to restore from a backup. However, you would still need access to the KMS key, so if it was disabled, you could simply re-enable it. If it was deleted altogether, then all data would be lost and you would no longer be able to access the database.

When using RDS encryption, you should be aware of some of the limitations that exists, other than the point of only being able to enable encryption during the RDS's creation.

Once you have enabled encryption on a database, the process cannot be reversed, so once it is encrypted, it will always remain encrypted, meaning that you can't disable encryption at any point. As mentioned previously, all backups and snapshots that are taken of an encrypted database are also encrypted.

One final point to make on RDS encryption is that although it is available on all of the database engines that RDS offers, and across all regions, other than China (Beijing), it is not available across all instance types and classes. The following tables show the currently supported options of these:

Instance Type	Instance Class
General Purpose (M4)	db.m4.large db.m4.xlarge db.m4.2xlarge db.m4.4xlarge db.m4.10xlarge db.m4.16xlarge
Memory Optimized (R3)	db.r3.large db.r3.xlarge db.r3.2xlarge db.r3.4xlarge db.r3.8xlarge
Memory Optimized (R4)	db.r4.large db.r4.xlarge db.r4.2xlarge db.r4.4xlarge db.r4.8xlarge db.r4.16xlarge
Burst Capable (T2)	db.t2.small db.t2.medium db.t2.large db.t2.xlarge db.t2.2xlarge
General Purpose (M3)	db.m3.medium db.m3.large db.m3.xlarge db.m3.2xlarge



More information on the latest available supported types can be found here: <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html#Overview.Encryption.Availability>.

Key Management Service (KMS)

Throughout this chapter, I have mentioned the Key Management Service a number of times, mainly in relation to other services using it such as S3 and RDS, as well as many other AWS services that require encryption to be performed.

We already know that KMS uses symmetric cryptography, and this was evident when we looked at the S3 encryption mechanisms, since we saw that the very same key was used to decrypt the data that was used to encrypt the data. This is essentially symmetric cryptography. In this section, I will dive a little deeper into KMS to allow you to gain more of an understanding about the service itself.

So, what is KMS?

KMS is a central repository for storing encryption keys (**customer master keys (CMKs)**) which can be accessed by services and users to generate data keys, which can then be used to encrypt information while at rest. There is a distinct difference between these two types of keys—CMKs and Data Keys, also known as **data encryption keys (DEKs)**—which I shall come to shortly.

KMS is not a multi-region service like AWS Identity and Access Management is, for example, since KMS keys are region-specific. This is an important point when it comes to architecting your infrastructure and your encryption requirements. You need to ensure that you are able to access the CMKs that you need to and that there is a sufficient key infrastructure in the regions that you are looking to use for encryption, especially when looking at multi-region deployment of services.

Although KMS is a managed service offered by AWS, it does not mean that AWS has access to your keys, or that it is able to restore an accidentally deleted key.

KMS, by its very nature of use and design, is a highly sensitive component within your AWS infrastructure—it holds the keys to all of your encrypted data, and as a result access is highly restricted. AWS does NOT have access to ANY of your keys within your account, and it is not able to restore any keys that are deleted. AWS is only able to perform maintenance on the underlying infrastructure that KMS runs on; there is no visibility of key infrastructure above that from AWS. As a result, you must maintain a secure level of administration and control over the keys that you create within KMS. As a part of this, controlling who has access to these keys is essential.

Whenever encryption keys are used, compliance to regulations is usually a requirement within an organization. As such, AWS CloudTrail has integration with KMS, which tracks and records API requests made against your keys to understand who accessed them and when. This provides a great audit trail and a method of security tracking to ensure that KMS keys are only being used when they should be and by the correct identities. As an example, CloudTrail records an event every time an `encrypt`, `decrypt`, `GenerateDataKey`, or `GetKeyPolicy` API is used.

There are four main elements to be aware of that help to understand KMS, these being the following:

- CMK
- DEK
- Key policies
- Grants

Customer master keys

The CMK is fundamental to KMS, and this key will never leave the confines of the KMS service. As we have already seen from earlier in this chapter, CMKs are the keys which are selected within other AWS services as the master key to use for the encryption process. CMKs are used to generate data encryption keys, both plaintext, and encrypted versions, again, just like we saw during the S3 encryption section. It's these data encryption keys that are sent beyond KMS to other AWS services, but again, to reiterate, the CMK does not leave KMS—it remains secure inside the service.

Two types of CMKs exist. Firstly, CMKs that are created and managed by AWS, such as the **(default) aws/rds** key we saw earlier when applying encryption to an RDS database. These CMKs are generally created the first time that they are called upon and are used by other AWS services that call upon KMS on their behalf.

Secondly, there are CMKs which are created by us as the customers of AWS, and these CMKs can be created within KMS or imported into KMS by using existing key material that customers might already be using on premises. Customer-generated CMKs provide additional advantages such as being able to implement greater flexibility on the actual key, including key rotation, access control through key policies, disabling, and even deleting the key.

Data encryption keys (DEK)

Data encryption keys are the keys that are used to perform the encryption operation against data. To ensure that an object can be encrypted and decrypted again, the CMK generates both a plaintext version of a DEK and also an encrypted version of the same key. During encryption, a plaintext DEK is used with an encryption algorithm against data to encrypt the data. Generally, the plaintext version of the DEK is then deleted and the then encrypted data is stored with the encrypted version of the DEK. If a malicious user gains access to the encrypted data, they would not be able to decrypt it unless they had the plaintext DEK. The only way of getting a plaintext version of the DEK is via KMS. The encrypted DEK needs to be combined with the CMK to generate it, which can then be used to decrypt the data.

The process of encrypting one key with another key is known as envelope encryption. In this instance, the DEK is encrypted by the CMK.

Key policies

Key policies are critical to your customer created CMKs. Without a key policy, access to the CMK will not be possible, and without access to your CMK, it cannot be used to perform any encryption operations. Key policies are effectively accessed policies that are tied directly to your CMK, which dictate who can use and administer the CMK. This direct association to the CMK makes key policies resource-based policies.

Key policies are very similar to IAM policies in the fact that they are JSON-based and also follow the same syntax. During the creation of a CMK, AWS will automatically create a default key policy, which you can edit as required. By default, the AWS key policy will ensure that your AWS root account has full access to the CMK. Without this entry, you will not be able to perform access control with normal IAM policies against your CMK for other users. This entry will look similar to the following:

```
{  
  "Sid": "Enable IAM User Permissions",  
  "Effect": "Allow",  
  "Principal": {"AWS": "arn:aws:iam::111122223333:root"},  
  "Action": "kms:*",  
}
```

```
    "Resource": "*"  
}
```

There is also a section within the key policy that allows you to configure administrators for that key who can perform functions such as the following:

- "kms:Create*
- "kms:Revoke"
- "kms:Disable*", and many more

However, these administrators are not able to use the CMK to perform any encryption operations. To be able to use the key for encryption purposes, you need to be listed as a user for the key within the policy. As a listed principal to use the key, those identities will be able to perform encryption operations, which can include the following:

- "kms:Encrypt"
- "kms:Decrypt"
- "kms:ReEncrypt*"
- "kms:GenerateDataKey*"
- "kms:DescribeKey"

Any users that are listed as principals to use the key are automatically added to a **Grant** section of the policy, which is identified by the Sid of **Allow attachment of persistent resources**.

Grants

KMS grants simply allows users of the CMK—and by users, I mean those who can perform encryption/decryption operations—to delegate their own permissions, or at least a subset of their permissions to another principal. Similarly to key policies, grants are also a resource-based access control method to the CMK.

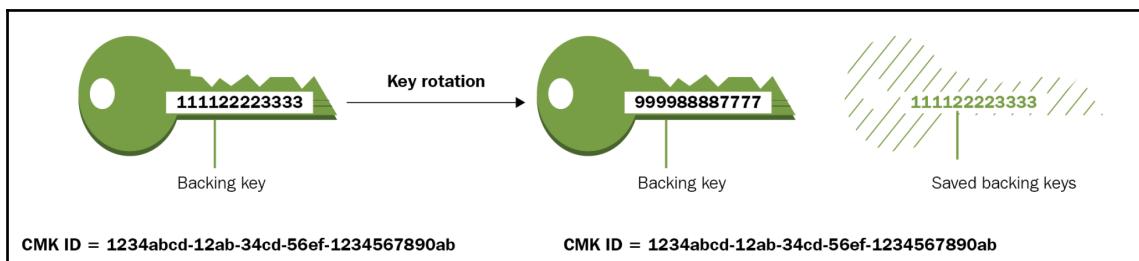
Grants can't be created through the AWS Management Console, since they have to be created programmatically through the KMS APIs via the CLI, for example. When assigning the grant to a grantee using the CreateGrant API, different parameters are defined within the command that details the CMK identifier and the grantee, along with the permissions that are being assigned by the principal issuing the command.

Key rotation

A fundamental element of key infrastructure security is that of key rotation. Rotating keys reduces the risk of a security threat by reducing the shelf life of the key. The longer a single key is active, the more time a malicious user could try and breach that key. Therefore, rotating keys regularly is a security best practice when looking at any KMS system.

KMS offers two solutions to rotating your keys—an automatic method and a manual method. Key rotation takes place on the CMKs and not the DEKs. By allowing KMS to manage the rotation for you (automatically), KMS will ensure that your key is rotated every 365 days. This time frame can't be altered. But what is actually happening when your key is automatically rotated, and what does it mean?

When key rotation takes place, it's the backing key that changes; all other details remain the same with the key, such as the **Amazon Resource Name (ARN)** and the CMK-ID. The backing key is the actual component that is used in the cryptographic process when generating, encrypting, and decrypting the DEKs. So, during the rotation, a new backing is generated, but all existing and older keys are retained to allow data that was encrypted with older backing keys of the CMK to still perform encryption/decryption operations:



If you wanted to perform a manual key rotation, which would provide you with greater control over the frequency of rotation.

One point to bear in mind with the automatic key rotation of your CMKs is that should you have a key in a **Disabled** state, which will mean that the key will not be rotated. However, if the key is then re-enabled, AWS will assess the age of the last key rotation, and if that key was rotated more than 365 days ago, it will perform an immediate and automatic key rotation.

Manual key rotation

Performing a manual key rotation takes a different approach to the automatic key rotation. During a manual key rotation, a completely new CMK is created, whereas, with the automatic approach, only the backing key is changed—the original CMK-ID is retained.

This then requires an additional administrator for any applications or services that may have been using the existing key, as you would need to update their configuration to point to the newly created CMK-ID. An alias name for the key could be used to simplify this process. This way, you would only need to update the alias-target to redirect to the new CMK.

Summary

Encryption is a necessity when storing data within the cloud—there is a level of trust given to AWS any time a customer stores data on their infrastructure. However, should this data be sensitive and confidential, additional measures should be put in place by us as the customer to ensure that the data is protected. This is often a requirement of many compliance regulations and governance controls that organizations are required to meet. AWS is aware of the importance of this factor and so has provided numerous methods and mechanisms of encryption to allow you to do just that.

Many AWS services come with some form of encryption, and in this chapter, we covered some of the most common services which are referenced within the certification. These services interact with the KMS, and so gaining a good understanding of this service and the different services and components is essential when understanding your encryption strategy.

My final point is that encryption must be applied as a security best practice for all sensitive data stored on the cloud.

Further reading

More information regarding the topics covered within this chapter can be found at the following links:

- **AWS Whitepaper – Encrypting Data at Rest:** https://d1.awsstatic.com/whitepapers/AWS_Securing_Data_at_Rest_with_Encryption.pdf
- **AWS S3 Encryption/Decryption mechanisms Infographic:** https://awsinfographics.s3.amazonaws.com/S3_Encryption_Infographic.png

15

An Overview of Security and Compliance Services

Understanding what each of the AWS security services does as a function is essential if you want to be able to be able to protect your infrastructure and applications as set out by the requirements within your security strategy.

Security is AWS's number one priority, and so, as a result, each security service has been designed and crafted with a specific objective. This chapter will look at a number of different services that focus on security and compliance at a high level, to enable you to select and architect the appropriate solution.

The services that will be discussed within this chapter include the following:

- AWS CloudTrail
- Amazon Inspector
- AWS Trusted Advisor
- AWS Systems Manager
- AWS Config

There are a number of other security-focused services, and detailed information on these can be found within the following chapters:

- **AWS Identity and Access Management (IAM)** is discussed within Chapter 14, *Understanding Access Control*
- **AWS Web Application Firewall (WAF)** is discussed within Chapter 18, *Web Application Security*
- **AWS Key Management Service (KMS)** is discussed within Chapter 15, *Encryption and Key Management*

Technical requirements

To gain the most from this chapter, you should have an understanding of the following AWS services:

- EC2, including AMI configuration
- **Simple Notification Service (SNS)**
- An awareness of AWS Lambda

AWS CloudTrail

AWS CloudTrail is a service that is used heavily when it comes to compliance and auditing. This is because its core function is to track and record **application programming interface (API)** calls made within your AWS account. These API requests can be initiated from users within the management console, when using an SDK or via the AWS CLI, to those initiated by other AWS services that may be responding to events, for example, an alert being sent by SNS when a CloudWatch metric threshold has been met.



CloudTrail is a global service covering all regions and most AWS services: <https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-aws-service-specific-topics.html>.

With so much coverage, not only does this make an effective compliance tool, it also makes a great security analysis tool. This is because the information captured during an API call also contains useful metadata that includes information such as the source IP address of the API request initiator, the identity of that initiator, and the timestamp information.

From a security analysis point of view, CloudTrail also integrates well with Amazon CloudWatch, so it's possible to configure CloudWatch metric filters to alert you with the help of SNS when a specific API call is requested that may be restricted, for example, an API request to create a new security group. If this API call was considered restricted through your own standards and it was then detected, your security team could be notified and be armed with the relevant information as to who requested and carried out the API call.

When an API call is made, AWS CloudTrail records that request as an event, and these events are then aggregated together within a log file. As with other logs files generated by other AWS services, this log file is then stored in a dedicated Amazon S3 bucket. The location of this S3 bucket is selected during the configuration and the setup of AWS CloudTrail. Retaining these log files allows you to have a recorded history of API requests (events) that have taken place within your AWS account. As I mentioned previously, CloudTrail can also interact with CloudWatch by sending a copy of the CloudTrail Logs to a CloudWatch Log group, as well as S3.

By using both S3 and CloudWatch, it allows you to keep a copy of all the logs on highly available durable storage (S3), which can be reviewed at any time and will be required to meet specific compliance and regulations that you may adhere to.



AWS has created a specific whitepaper on this very need; it is entitled *Security at Scale: Logging in AWS: How AWS CloudTrail can help you achieve compliance by logging API calls and changes to resources* (https://d0.awsstatic.com/whitepapers/compliance/AWS_Security_at_Scale_Logging_in_AWS_Whitepaper.pdf).

Also, thanks to logs being sent to CloudWatch, it allows you to filter and alert (SNS integration) on data captured within the log files, as they are received as a security tool.

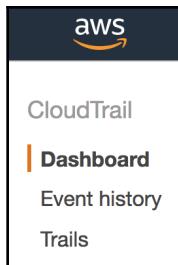
CloudTrail is also used to help resolve operational issues that may have occurred. Using the CloudTrail dashboard within the management console, you can filter event data to ascertain what may have caused the issue, who caused it, and when it was caused. This results in a quicker resolution to operational issues.

Let's now take a look at how you can configure AWS CloudTrail for your account.

Go to **AWS Management Console**, and follow these steps:

1. Select the **CloudTrail** service, which can be found under the **Management Tools** category.

2. Select **Trails** from the left menu:



3. Then, select **Create trail**:

Trails

Deliver logs to an Amazon S3 bucket. CloudTrail events can be processed by one trail for free. There is a charge for processing events with additional trails. For more information, see [AWS CloudTrail Pricing](#).

Create trail

This is where you are configuring the actual trail that will be used to capture your API calls. You can create different trails to capture different information.

4. Enter a name for your trail; it's best to use something descriptive and useful to help you identify different trails after you have created them.
5. The Trail itself can apply to a single region or to *all* regions; depending on the trail use case, you can select the most appropriate options:

Create Trail

Trail name*

Apply trail to all regions Yes No [?](#)

6. The next two sections relate to the type of events that you want CloudTrail to capture:

The screenshot shows the AWS CloudTrail configuration interface. It is divided into two main sections: "Management events" and "Data events".

Management events: This section provides insights into management operations. It includes a filter for "Read/Write events" with options: All (selected), Read-only, Write-only, and None. A link to "Learn more" is also present.

Data events: This section provides insights into resource operations. It includes a filter for "S3" (selected) and "Lambda". A link to "Learn more" is also present.

Table View: A table view shows resources for S3 buckets. The columns are: Bucket name, Prefix, Read, Write. A checkbox at the top left is checked, labeled "Select all S3 buckets in your account". Below the table, it says "Showing 0 of 0 resources" and "No resources found".

7. These sections are split between **Management events** and **Data events**.

Management events focus on different management operations that are carried out by your AWS services, such as the IAM API of **AttachRolePolicy** or the EC2 API **CreateSubnet**. Within this section of the configuration, you have the option to record a variety of API operations:

- **All:** This will record *all* management events across your account within the region(s) specified within this Trail
- **Read-only:** This will record APIs that only operate as read request, for example, the API of **DescribeSubnets**
- **Write-only:** This option will record API requests that have a modifying effect on your resources, such as **TerminateInstances**
- **None:** This option will not record any management events

8. **Data events** record resource operations on specific resources or services. As you can see, the options for **Data events** currently reads **S3** and **Lambda**. The **S3** option allows you to capture object-level API calls at either a read or a write level. If you select **Lambda**, you are able to record invoke API operations for your functions.
9. Once you have selected your events to record, you are then prompted to enter all information relating to the storage of your log files:

Storage location

Create a new S3 bucket Yes No

S3 bucket* 

Log file prefix 
Location: PackT/AWSLogs/730739171055/CloudTrail/us-east-1

Encrypt log files with SSE-KMS Yes No 

Create a new KMS key Yes No

KMS key* 
KMS key and S3 bucket must be in the same region.

Enable log file validation Yes No 

Send SNS notification for every log file delivery Yes No 

10. You can select an existing S3 bucket to store your CloudTrail Log files, or you can ask CloudTrail to create a new bucket for you based on the name of the bucket you enter. If you let CloudTrail create the bucket, then relevant permissions will be applied to the bucket, allowing the CloudTrail service to deliver logs to it. If you decide to use an existing bucket, you will need to add permissions to the bucket policy, to allow the CloudTrail service principle to deliver these log files. AWS has thankfully created a template to allow you to do this, which can be found here: <http://docs.aws.amazon.com/awscloudtrail/latest/userguide/create-s3-bucket-policy-for-cloudtrail.html#using-an-existing-bucket-for-ct-logs>.
11. Within the **Advanced** section, you can also add a prefix, to allow you to easily identify the logs for this particular Trail. You may have multiple Trails all being saved to the same bucket, by adding a prefix, this allows you to differentiate between the log files.
12. Encryption is also an option, using the server-side encryption with KMS. By selecting **Yes**, you will be prompted to either select an existing KMS key or ask CloudTrail to create a new one. In this example, I have asked for a new one to be created called PackT-CMK.
13. The log file validation is used as a security and an audit feature that allows you to determine whether the log file has changed or has been tampered with in any way after it has been delivered to your nominated S3 bucket.
14. Finally, you can set up an SNS notification for a successful delivery of your log file to your S3 bucket. Next, click **Create**:

The screenshot shows a configuration step for CloudWatch Logs. It includes a heading 'CloudWatch Logs', a descriptive text about enabling SNS notifications, and a prominent blue 'Configure' button.

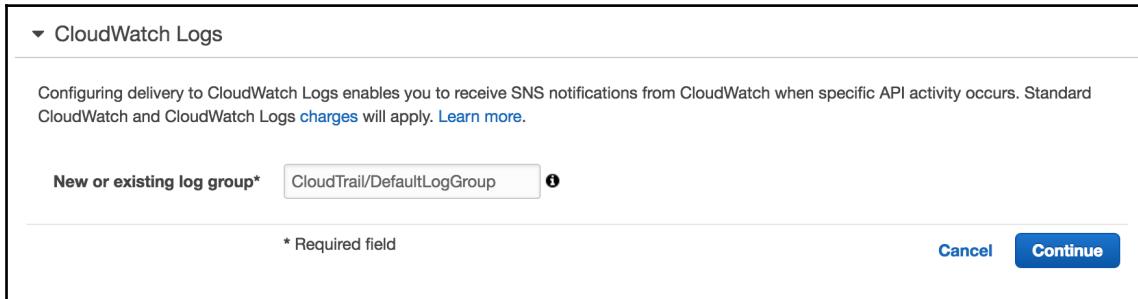
▼ CloudWatch Logs

Configuring delivery to CloudWatch Logs enables you to receive SNS notifications from CloudWatch when specific API activity occurs. Standard CloudWatch and CloudWatch Logs charges will apply. [Learn more](#).

Configure

Your Trail has now been created; however, there is an additional configuration that you can implement at this point, such as the **CloudWatch** integration.

15. To do so, you need to select your newly created Trail and scroll down to the **CloudWatch** configuration section and click **Configure**. This will then allow you to select either a **New or existing log group**:



Now we have our Trail configured, let's look at the entire process of when an API request is initiated:

1. An API is called either by a user or a service within your AWS account
2. AWS CloudTrail identifies whether this API is to be recorded within any active and configured Trails
3. Following a successful match with a Trail, CloudTrail records the API as an event within the active log file, along with all associated metadata
4. The log file will then be delivered to the nominated S3 bucket within the corresponding Trail and to a CloudWatch Logs group (if this component has been configured)
5. If encryption has been enabled for the Trail, the log file will be stored and encrypted with the specified KMS Key using server-side encryption with KMS (SSE-KMS)

Amazon Inspector

Amazon Inspector is a security service designed to help secure your EC2 instances against known vulnerabilities and threats. It is used to identify and notify you if any of these vulnerabilities exist within your EC2 fleet, in addition to identifying those that could also affect your applications running on your EC2 instances. Like Amazon CloudTrail, Inspector is also a managed service that only operates through an Amazon Inspector agent that is installed locally on the instances that you want to protect.

Once the agent is installed, it is possible to automatically run scheduled assessments that look for security weaknesses and flaws, using a series of predefined rules packages. These rules packages can consist of one or more of the following four categories:

- **Center for Internet Security (CIS) Benchmarks:** These are global standards that are used across the industry, reflecting best practices for protecting data. These standards are refined and updated on a regular basis. AWS is a CIS Benchmarks member company, and Amazon Inspectors associated certifications can be found here: <https://www.cisecurity.org/partner/amazon-web-services/>. Additionally, if you want to learn more about the CIS benchmarks, then visit <https://www.cisecurity.org/>.
- **Common Vulnerabilities and Exposures (CVE):** This is a database of known threats that is publicly viewable, allowing anyone to look up and research found vulnerabilities. The vulnerabilities that are defined on this list have been confirmed and verified, and have documentation surrounding the threat in detail.



To learn more on the CVE list and what it contains you can visit <https://cve.mitre.org/>.

- **Runtime behavior analysis:** This effectively looks at the behavior of how your EC2 instances are running during an assessment run that is being performed on the instance. For example, it would check for any unused listening TCP ports or insecure server protocols, such as Telnet or rlogin.
- **Security best practices:** This rules package is only available to run on Linux instances and will examine them for weak points using different security best practices. A couple of examples of the types of checks that this package performs are as follows:
 - **Disable root login over SSH:** This checks whether the SSH daemon allows you root to log into your instance
 - **Disable password authentication over SSH:** This checks to see whether password authentication over SSH is configured

I mentioned previously that Amazon Inspector is an agent-based security tool; therefore, for any instances that you want to assess, you will need to ensure the agent is installed and running. The agent collects data based on the rules packages selected during the assessment and records the information as telemetry data, which is sent back to the Amazon Inspector service over an encrypted channel using **Transport Layer Security (TLS)**.

As the service is managed by AWS, once the agent is installed, all future updates to that agent are automatically carried out by AWS, removing the need for you to maintain agent software updates.

Installing the agent

Installing the Amazon Inspector agent is a simple process, it first requires you to download the agent on to the instance and then install it.

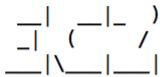
The following shows how to download and install the agent for a Linux-based EC2 instance:



For further information on how to perform an installation of a windows instance, please see the following AWS documentation: https://docs.aws.amazon.com/inspector/latest/userguide/inspector_installing-uninstalling-agents.html#install-windows.

1. Connect to your Linux instance.
2. Download the agent by entering one of the two following commands:

```
wget https://d1wk0tztpsntt1.cloudfront.net/linux/latest/install  
curl -O https://d1wk0tztpsntt1.cloudfront.net/linux/latest/install
```



Amazon Linux 2 AMI

```
https://aws.amazon.com/amazon-linux-2/  
[ec2-user@ip-10-0-1-132 ~]$  
[ec2-user@ip-10-0-1-132 ~]$  
[ec2-user@ip-10-0-1-132 ~]$  
[ec2-user@ip-10-0-1-132 ~]$  
[ec2-user@ip-10-0-1-132 ~]$ wget https://d1wk0tztpsntt1.cloudfront.net/linux/latest/install
```

- Once it is downloaded, you will get the following message, showing that the download has been successful:

```
[ec2-user@ip-10-0-1-132 ~]$ wget https://d1wk0tztzpsntt1.cloudfront.net/linux/latest/install
--2018-08-10 14:50:04-- https://d1wk0tztzpsntt1.cloudfront.net/linux/latest/install
Resolving d1wk0tztzpsntt1.cloudfront.net (d1wk0tztzpsntt1.cloudfront.net)... 52.85.200.218, 52.85.200.14, 52.85.200.57, ...
Connecting to d1wk0tztzpsntt1.cloudfront.net (d1wk0tztzpsntt1.cloudfront.net)|52.85.200.218|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 34671 (34K)
Saving to: 'install'

100%[=====] 34,671      --.-K/s   in 0.001s

2018-08-10 14:50:04 (62.5 MB/s) - 'install' saved [34671/34671]
[ec2-user@ip-10-0-1-132 ~]$
```

- At this point, you then need to install the agent by running the command `sudo bash install`.
- When the installation has completed, you will get a message:

```
Installation script completed successfully.

Notice:
By installing the Amazon Inspector Agent, you agree that your use is subject to the terms of your existing AWS Customer Agreement or other agreement with Amazon Web Services, Inc. or its affiliates governing your use of AWS services. You may not install and use the Amazon Inspector Agent unless you have an account in good standing with AWS.

* * *
Current running agent reports to arsenal endpoint:
Current running agent reports version as: 1.1.767.0
This install script was created to install agent version:1.1.767.0
In most cases, these version numbers should be the same.

[ec2-user@ip-10-0-1-132 ~]$
```

The agent is now installed and is ready to start collecting information based on the assessment and the rule packages selected.

Assessment templates, runs, and findings

Once your agent is installed, you are able to set up and configure the assessment template and run and view the output through the assessment results.

The assessment templates outline the configuration of data as to how an assessment will be run on your EC2 instances. For example, you can define which rules packages you wish to run and how long the assessment should run. AWS has a recommendation of one hour, but you can select a shorter period of 15 minutes, or up to 24 hours, if required. The template also allows you to select an SNS topic, which will notify you when an assessment run starts and stops.

When your assessment template is created, your agent installed and an Amazon Inspector role created (which is created the first time you use Amazon Inspector), and then you are able to run an assessment. This role is simply used by Amazon Inspector to have read-only access to your EC2 instances. The policy of this role is as follows:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "ec2:DescribeInstances"  
            ],  
            "Resource": [  
                "*"  
            ]  
        }  
    ]  
}
```

During an assessment run, the agent is used to filter and gather information as telemetry data, which is then fed back to Amazon Inspector. At this point, the information is assessed based on the rules packages outlines within the assessment template.

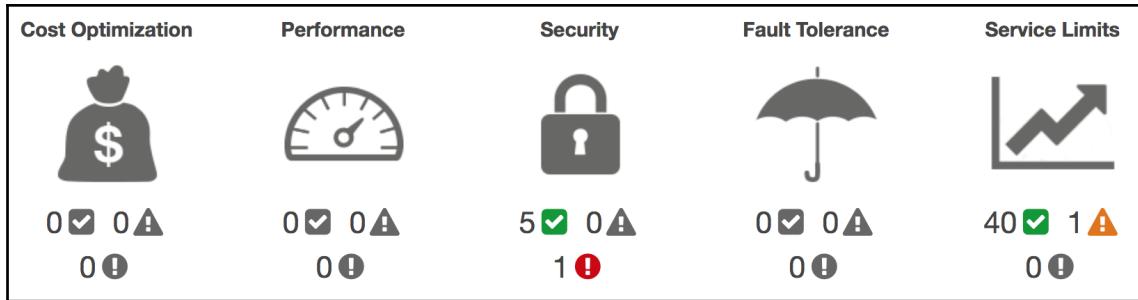
Any findings can then be presented within an assessment report, which can be distributed to auditors or your security teams to rectify any issues or concerns highlighted. Depending on how much detail you want within the report, you can select either of the following two reports:

- A findings report
- A full report

Within the findings report, you can expect to see a summary of the assessment run, the EC2 instances included within the run, the rules packages assessed, and a list of findings. The full report contains all of this information, plus information containing items that successfully passed the assessment.

AWS Trusted Advisor

The AWS Trusted Advisor service is extremely useful and can help you optimize your AWS infrastructure within your environment based on best practices. It provides recommendations based on the following five core categories:



- **Cost Optimization:** This category will provide suggestion and improvements on how you could reduce your costs based on your current AWS environment and resources
- **Performance:** Trusted Advisor will scan and assess your resources to see whether there are changes that could be made to rectify any performance concerns
- **Security:** This will analyze your environment to look for security vulnerabilities and threats, based on best practice
- **Fault Tolerance:** This category will provide suggestions and improvements that could enhance your resilience within your deployed resources
- **Service Limits:** This will show you how close you are to particular service limitations, allowing you to request an increase or to reduce your resource consumption

Using these categories, Trusted Advisor will scan, assess, analyze, and review your architecture to help you optimize and follow best practices that can easily be implemented, which serves only to benefit you and your solutions. This is a service that should be used by all organizations, as it helps to have an automatic and helping hand when it comes to reviewing and checking your deployed solutions, ensuring that they are economical, efficient, secure, resilient, and within resource capacity.

As you can see from the preceding diagram, there are a number of related icons to each category: a green tick, a yellow exclamation mark, and a red exclamation mark, and each of these relates to a specific static. These icons symbolize how many checks within those categories fall within each status level:

- The green tick means *no action is necessary* and shows the number of checks within the category that passed successfully.
- The yellow exclamation mark means *investigation required*, as it could lead to further issues and problems.
- The red exclamation mark means it *requires immediate attention*. If any check comes back as red, it should be looked at right away, especially if this is within the security category.

Being such a powerful and useful tool, it unfortunately doesn't come for free; however, there are a number of checks included that anyone can use. Nevertheless, to harness the full power and the benefits of Trusted Advisor, you will need to ensure that you have a business or an enterprise support plan for AWS.

If you do not have one of these support plans, then there are six free checks within the security category as well as having the entire service limits category also included.

The six **free** checks included under **Security** are these:

- **Security Groups - Specific Ports Unrestricted:** This checks across your security groups to see whether there are any rules with ports associated to unrestricted access (0.0.0.0/0).
- **Amazon EBS Public Snapshots:** This simply identifies whether any EBS snapshots have been marked as public, allowing other accounts to be able to see the snapshot.

- **Amazon RDS Public Snapshots:** Similar to the second point mentioned, these check to see whether any RDS snapshots have been marked as public.
- **IAM Use:** This looks at your use of IAM and ensures that you have at least one other user other than the root account set up. This is because it is not recommended that you perform any actions with the root account due to its high privilege level within your account.
- **MFA on Root Account:** This checks to ensure that you have configured the use of multi-factor authentication when logging in with your root account. Again, this is highly recommended due to its elevated permission level.
- **Amazon S3 Bucket Permissions:** This is used to assess the permissions against buckets that might be overly permissive.

So, to reiterate, if you do not have the business or enterprise support plan, then Trusted Advisor will not check *any* cost optimization, performance or fault tolerant checks for you against your infrastructure, but it checks a very small portion of the checks within the security category.

Now, if we go back to our preceding screenshot, we can see that we have one yellow warning and one red warning, so let's take a look to see what caused these issues.

Yellow warning under service limits

From within the main page of the Trusted Advisor dashboard, any alerts and warning will be displayed. From here, I can select the yellow warning which is listed as shown:

▶  **VPC Network Interfaces**

Checks for usage that is more than 80% of the VPC Network Interfaces Limit.

1 of 15 items have usage that is more than 80% of the service limit.

We can see that the issue relates to the **VPC Network Interfaces** check and that one of the 15 items that this check performs a review against is breaching the 80% usage of the service limit within my account. To gain more information, I can click on the expand arrow or the title of the check. This then presents the following:

Service Limits Checks

VPC Network Interfaces Refreshed: a day ago

Checks for usage that is more than 80% of the VPC Network Interfaces Limit. Values are based on a snapshot, so your current usage might differ. Limit and usage data can take up to 24 hours to reflect any changes. In cases where limits have been recently increased, you may temporarily see utilization that exceeds the limit.

Alert Criteria

Yellow: 80% of limit reached.
Red: 100% of limit reached.
Blue: Trusted Advisor was unable to retrieve utilization or limits in one or more regions.

Recommended Action

If you anticipate exceeding a service limit, open a case in Support Center to [request a limit increase](#).

Additional Resources

[VPC Network Interface Limits](#)

1 of 15 items have usage that is more than 80% of the service limit.

◀◀ ◀◀ 1 to 15 of 15 ▶▶ ▶▶ View 20 ▲				
	Service	Region	Limit Amount	Current Usage
<input type="checkbox"/>	vpc	us-east-2	5	1
<input type="checkbox"/>	vpc	ca-central-1	5	1
<input type="checkbox"/>	vpc	us-west-1	5	1
<input type="checkbox"/>	⚠ vpc	eu-west-1	5	4
<input type="checkbox"/>	vpc	eu-west-2	5	1
<input type="checkbox"/>	vpc	eu-central-1	5	1
<input type="checkbox"/>	vpc	eu-west-3	5	1
<input type="checkbox"/>	vpc	us-east-1	5	3
<input type="checkbox"/>	vpc	us-west-2	5	1
<input type="checkbox"/>	vpc	ap-northeast-1	5	1
<input type="checkbox"/>	vpc	ap-northeast-2	5	1
<input type="checkbox"/>	vpc	ap-southeast-2	5	1
<input type="checkbox"/>	vpc	sa-east-1	5	1
<input type="checkbox"/>	vpc	ap-southeast-1	5	1
<input type="checkbox"/>	vpc	ap-south-1	5	1

Now I can clearly see where the issue is. Within the **eu-west-1** region, I currently have four VPCs configured and the service limit on a VPC in any region is five, hence the 80% alert. Now that I am aware of this, I can plan accordingly if I need many more VPCs within that region. If I another two or more, then I would need to request an increase via AWS, as I would only be allowed one more VPC in that region until my service limit was reached.

Red warning under service limits

If I go back to the dashboard of Trusted Advisor, I can select the red alert, as shown:

Recommended Actions

- ! Security Groups - Specific Ports Unrestricted**

Checks security groups for rules that allow unrestricted access (0.0.0.0/0) to specific ports.
6 of 16 security group rules allow unrestricted access to a specific port.

I can see that this relates to six different weaknesses in my security groups, so if I click on the warning, I can review it in more detail, as follows:

Recommended Actions

► ! Security Groups - Specific Ports Unrestricted

Refreshed: a day ago Download CSV

Checks security groups for rules that allow unrestricted access (0.0.0.0/0) to specific ports. Unrestricted access increases opportunities for malicious activity (hacking, denial-of-service attacks, loss of data). The ports with highest risk are flagged red, and those with less risk are flagged yellow. Ports flagged green are typically used by applications that require unrestricted access, such as HTTP and SMTP.

If you have intentionally configured your security groups in this manner, we recommend using additional security measures to secure your infrastructure (such as IP tables).

Alert Criteria

Green: Access to port 80, 25, 443, or 465 is unrestricted.
Red: Access to port 20, 21, 1433, 1434, 3306, 3389, 4333, 5432, or 5500 is unrestricted.
Yellow: Access to any other port is unrestricted.

Recommended Action

Restrict access to only those IP addresses that require it. To restrict access to a specific IP address, set the suffix to /32 (for example, 192.0.2.10/32). Be sure to delete overly permissive rules after creating rules that are more restrictive.

Additional Resources

[Amazon EC2 Security Groups](#)
[List of TCP and UDP port numbers](#) (Wikipedia)
[Classless Inter-Domain Routing](#) (Wikipedia)

6 of 16 security group rules allow unrestricted access to a specific port.

		Region	Security Group Name	Security Group ID	Protocol	From Port	To Port
<input type="checkbox"/>	●	eu-west-1	launch-wizard-1	sg-20c2835b	tcp	3389	3389
<input type="checkbox"/>	△	eu-west-1	launch-wizard-2	sg-275b3a5d	tcp	22	22
<input type="checkbox"/>	△	eu-west-1	Logging Private	sg-bcff58c1	tcp	22	22
<input type="checkbox"/>	△	eu-west-1	launch-wizard-3	sg-d9a3c2a3	tcp	22	22
<input type="checkbox"/>	△	eu-west-2	launch-wizard-1	sg-453ebd2d	tcp	22	22
<input type="checkbox"/>	△	eu-west-2	launch-wizard-2	sg-d3a19fb	tcp	22	22

I can see from the table that I have five security groups marked as **yellow** and one marked as **red**. I can then use the **Alert Criteria** to understand how this check has categorized its findings between these two colors:

Alert Criteria

Green: Access to port 80, 25, 443, or 465 is unrestricted.

Red: Access to port 20, 21, 1433, 1434, 3306, 3389, 4333, 5432, or 5500 is unrestricted.

Yellow: Access to any other port is unrestricted.

I need to act on these warnings immediately to reduce the chance of a breach occurring within my environment.

As you can see just using these few freely available checks is invaluable, imagine the findings that could be generated if you had full access to *all* the checks available and the benefits that it could bring to your organization.

AWS Systems Manager

The **AWS Systems Manager** (known as **SSM**) service has been derived from previous services that you may have heard of that are no longer in service, these being AWS EC2 Systems Manager and Amazon Simple Systems Manager.

The service itself is designed to help you manage and give visibility to your infrastructure, particularly when it comes to your EC2 fleet. It also has the ability to configure and manage your virtual compute resources on-premises, as well as within your AWS environment, ensuring they meet specific compliance needs. Using AWS SDKs, the AWS CLI, AWS Toolkit for Windows PowerShell, or even the AWS Management Console, you are able to centrally manage, review, and automate tasks for your resources. This makes it a great tool for helping you view your infrastructure through a single pane of glass, making it highly scalable.

Much like Amazon Inspector, SSM also uses agents that are installed locally to help with the configuring and deploying of updates to your EC2 and on-premises virtual servers. When requests are sent by Systems Manager, the agent receives the request and carries out the required task.

From AMIs with Windows Server 2003-2012 R2 published after November 2016 already have the agent installed by default. If you have EC2 instances without the agent, then you can download and install as shown here: <https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-install-ssm-win.html>.



If you are running Linux AMIs, then by default, the agent is also installed on Amazon Linux, Amazon Linux 2, Ubuntu Server 16.04, and Ubuntu Server 18.04 LTS base AMIs. If you are running any other Linux AMIs, then you will need to install the agent manually following this process: <https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-install-ssm-agent.html>.

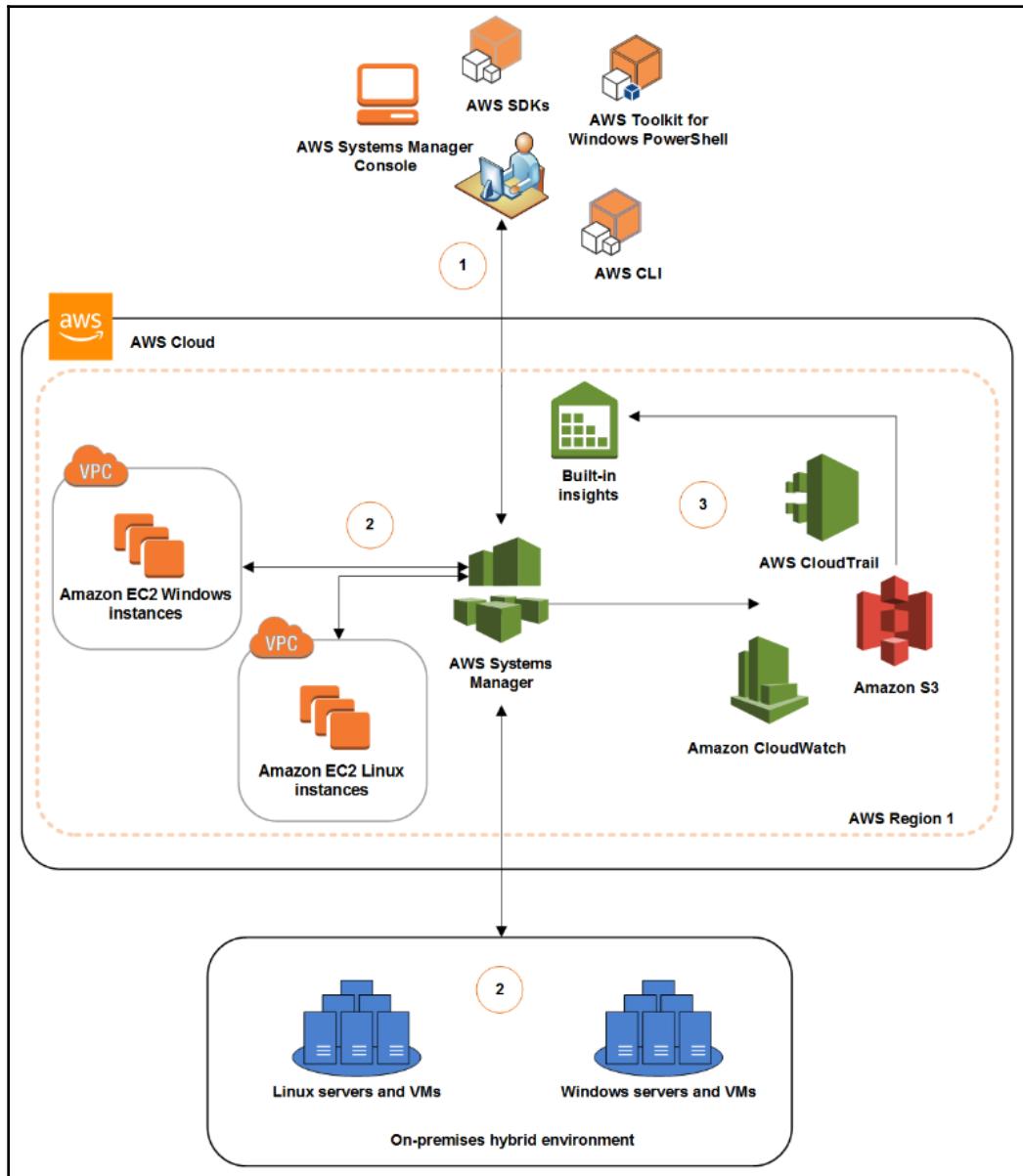
If you want to control and automate configurations on your on-premise virtual servers then you will need to perform a manual installation: <https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-install-managed-win.html>.

So, once the agent is installed on the required EC2/virtual server fleet, you are able to control and manage your instances. But how does this work? At a very high level, it's very simple.

Once you have verified the installation of SSM agents on your resources, you can then use Systems Manager to configure and deploy configuration changes to resource groups (your EC2/Virtual Servers).

The agent will then receive the request of configuration changes sent by SSM to be performed and verifies the packages to be run. The agent will then carry out and process the tasks required.

The output and results of these configuration changes can be sent to other AWS services for reporting and monitoring, such as AWS CloudTrail, Amazon S3, and Amazon CloudWatch:



Let's now look deeper at some of the elements that make up this service, these being the following:

- Resource groups
- Actions
- Insights
- Shared resources

Resource groups

This is simply a group of resources that are located within a single region that are identified and matched through the results of a query. These queries can be created within the Systems Manager console, which can then be saved and used as a resource group.

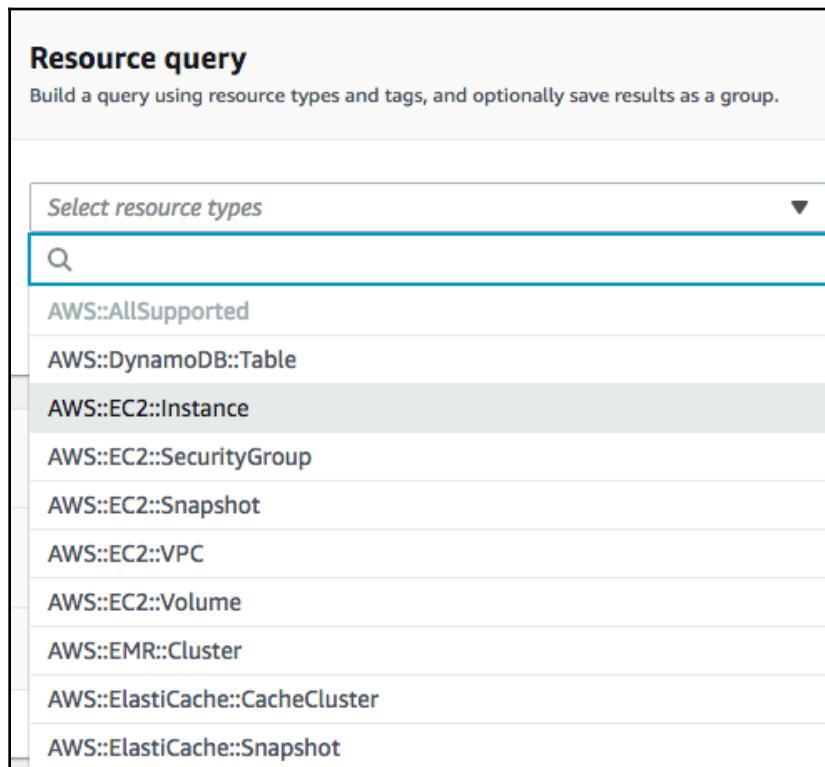
Creating a resource group

In this example, I will create a resource group based on EC2 tag names of PackT:

1. From within **AWS Systems Manager** console, I select **Find Resources** from the left menu:

The screenshot shows the AWS Systems Manager console with the 'Find resources' page selected. The left sidebar contains navigation links for Resource Groups, Insights, Actions, and Shared Resources. The main area displays a 'Resource query' section with fields for 'Select resource types' (set to 'All supported resource types'), 'Tag key' (empty), and 'Optional tag values' (empty). A note says 'Type the tag key and optional values shared by the resources you want to find, and then press Enter.' Below this is a 'Query results' section with a search bar and a table header for 'Name', 'Service', 'Type', and 'ID'. A message 'No resources.' is displayed. At the bottom right is a 'Save query as group' button.

2. Within the **Resource query** section, I will select the **AWS::EC2::Instance** from the **Select resource type** drop-down box:



3. I can now enter the tag key of Name with an optional tag value of PackT:

The screenshot shows the AWS Systems Manager 'Find resources' interface. At the top, it says 'AWS Systems Manager > Find resources'. Below that is a section titled 'Find resources' with a 'Resource query' sub-section. It includes fields for 'Select resource types' (set to 'AWS::EC2::Instance'), 'Tag key' (set to 'Name'), 'Optional tag values' (set to 'PackT'), and a 'View query results' button. A note below says 'Type the tag key and optional values shared by the resources you want to find, and then press Enter.' Underneath is a 'Query results (2)' section with a table showing two EC2 instances. The table has columns: Name, Service, Type, and ID. The first instance is 'EC2 Instance i-02cdee2bc75464dd3' (Service: EC2, Type: Instance, ID: i-02cdee2bc75464dd3). The second instance is 'EC2 Instance i-0ba7aa7ccbc805b45' (Service: EC2, Type: Instance, ID: i-0ba7aa7ccbc805b45). There is also a 'Save query as group' button at the bottom right of the results table.

Name	Service	Type	ID
EC2 Instance i-02cdee2bc75464dd3	EC2	Instance	i-02cdee2bc75464dd3
EC2 Instance i-0ba7aa7ccbc805b45	EC2	Instance	i-0ba7aa7ccbc805b45

4. Once this information is entered, I select **View query results**, and the following is displayed showing two instances that match the key of Name and the key value of PackT.
5. When I am happy with the returned results, I can select **Save query as group** using the orange button on the bottom right.

This will then ask me for additional details, such as a **Group name**, a **Group description**, and, optionally, **Group tags**:

The screenshot shows the 'Group details' configuration page. It has two main sections: 'Group details' and 'Group tags - optional'. In the 'Group details' section, the 'Group name' field contains 'PackT_Instances'. Below it, a note says 'Maximum 128 characters. Has to start with a letter, and can only contain letters, numbers, and hyphens.' In the 'Group description - optional' section, the 'Instances tagged with PackT' field contains 'Instances tagged with PackT'. A note below says 'Maximum 512 characters. It can only contain letters, numbers, hyphens, underscores, dots, and spaces.' The 'Group tags - optional' section is expanded, showing a table with one row. The row has a 'Name' column containing 'PackT_Instances' and a 'Remove' button. An 'Add another row' link is below the table. At the bottom right are 'Cancel' and 'Create group' buttons.

Group details

Group name
PackT_Instances

Maximum 128 characters. Has to start with a letter, and can only contain letters, numbers, and hyphens.

Group description - *optional*
Instances tagged with PackT

Maximum 512 characters. It can only contain letters, numbers, hyphens, underscores, dots, and spaces.

▼ **Group tags - optional**
The tags specified here will not be applied to group resources, but only the resource group itself.

Name	Remove
PackT_Instances	Remove

[Add another row](#)

[Cancel](#) [Create group](#)

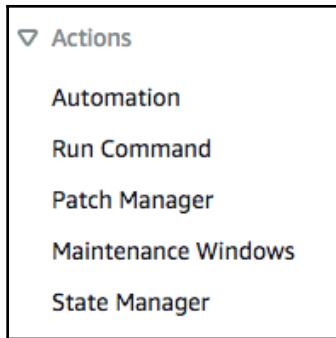
- When I have added the required information, I then need to select **Create group**.
- This will then display a confirmation message that the group is created:

The resource group "PackT_Instances" has been successfully created in the current region (eu-west-1).

One of the benefits of having these resource groups configured is that it allows you to perform configuration changes on multiple instances at once through a single command within Systems Manager.

Actions

Actions stipulate what functions SSM can perform on your resources, and this can be broken down into the following sections:

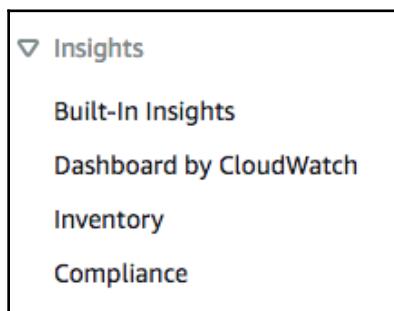


- **Automation:** This is a feature that allows you to perform automated maintenance tasks that may otherwise require some form of scheduled script or manual effort. As a few examples of the capabilities of automation actions you can do the following:
 - Update AMIs with applications installs and agents allowing you to ensure your AMIs are always up to date
 - Install the latest updates for drivers
 - Update the OS with the latest updates ensuring known security risks are removed
 - Perform password/SSH key resets for your Windows/Linux instances
- **Run Command:** This is a very useful tool that allows you to perform updates to multiple instances at a time by remotely executing scripted commands. For an example of how this works and if you want to test this **Run Command** yourself, you can view this tutorial here, which will guide you through the relevant step: <https://aws.amazon.com/getting-started/tutorials/remotely-run-commands-ec2-instance-systems-manager/>.
- **Patch Manager:** This is very self-explanatory: it allows you to automate the installation of patching your EC2/virtual servers. It also allows you to review the current status of your resources by identifying any missing patches. Through the use of trusted sources via configured baselines, you are able to determine where the update or security patch is installed from to ensure it is an approved patch.

- **Maintenance Windows:** As expected, this action allows you to define a window for scheduled maintenance against your resources, for example performing some of the actions defined in the automation section mentioned previously. The windows itself are defined by a time frame, along with the resources associated with that window and the specific actions or tasks that can take place.
- **State Manager:** This actions allows you to specify and define a particular state for your resources through an automated configuration management process, for example, allowing you to run specific bootstrap data that would install certain software during the start-up of an EC2 instance, or perhaps you could use state management to set specific network configuration settings across your instances.

Insights

This element provides an overview and management element for centrally viewing your resources. Similar to actions, there are a number of different insight methods available:



- **Built-In Insights:** This integrates with other AWS services, such as AWS Config, AWS CloudTrail, and Amazon Trusted Advisor, as well as the **Personal Health Dashboard**:

Built-In Insights

The built-in insights of your AWS resources. You can see what is happening in general or resource group specifically.

The screenshot shows the AWS CloudTrail Insights interface. At the top, there are tabs: Config, CloudTrail (which is selected and highlighted in red), Personal Health Dashboard, and Trusted Advisor. Below the tabs, it says "Resource Groups" and shows a dropdown menu with "PackT_Instances". A table below lists two events:

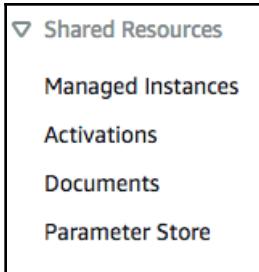
Resource ID	Resource type	Event time	Event name
i-02cdee2bc75464dd3	Instance	8/17/2018, 10:41:14 AM	CreateTags
i-0ba7aa7ccbc805b45	Instance	8/17/2018, 10:41:27 AM	CreateTags

You can filter on specific resource groups as this diagram shows (PackT_Instances) or general information across your environment.

- **Dashboard by CloudWatch:** This allows you to view custom CloudWatch dashboards that you have created to view metric and threshold information for a range of resources.
- **Inventory:** This is a very useful feature of SSM, as it allows you to gain an insight into what OS your resources are running and what software they have installed on them, as well as key metadata about the instance itself. This allows you to identify whether your fleet of resources is operating within the defined policies set out by your organization from a software perspective.
- **Compliance:** This enables you to search your resources for noncompliance when it comes to patch and configuration. As a result, this integrates with patch manager and State Manager to gather the information required. By collecting data across your resources, you are able to drill down into non-compliant resources to identify what needs to be changed.

Shared resource

This element identifies shared resources to help manage and configure your instances and on-premise virtual servers made up of the following four resources:



- **Managed Instances:** This displays all of the configured instances that are ready for use with SSM, including both EC2 and virtual servers from your on-premises hybrid solution. A part of this configuration requires that the instances have the agent installed and that the necessary IAM permissions are in place.
- **Activations:** This element is entirely for your on-premise virtual servers in your hybrid environment. To enable connectivity of these servers and the management of these servers by SSM, you need to create a managed-instance activation, which essentially functions as a secret access key authorizing permissions to the SSM service. This is where you can create those manage-instance activations.
- **Documents:** Within the documents section, you are able to choose different types of documents that are used to specify which actions and scripts are carried out on your selected instances. You are able to use these documents in conjunction with your own configured parameters, allowing you to implement a level of customization. The documents are formatted in either a JSON or a YAML format. The different document types that are available are as follows:
 - **Command:** Command documents are used by both the State Manager and Run Command functions to apply specific configurations and to carry out commands on your instances. There are a wide scope of pre-configured command documents within SSM, for example, **AWS-ConfigureCloudWatch**. This is used to export metrics and log files from your instances to Amazon CloudWatch.

- **Policy:** Policy documents effectively apply policies to your instances ensuring specific tasks or actions are being out. If the policy is removed, then that function ceases to continue. An example of a policy document is **AWS-GatherSoftwareInventory**. This is used to perform a software inventory of your instances.
- **Automation:** Automation documents allow to you perform automated maintenance tasks that are referenced from within the Action section mentioned previously. A good example of an automation document would be **AWS-CreateDynamoDbBackup**. This performs an automatic backup of a DynamoDB table.
- **Parameter Store:** This element of SSM relates to security and storage. It is used to retain configuration data management and secrets management. This information could be credentials, passwords, database strings, and so on. So effectively information that needs to be kept secure that can be called upon by SSM through the Run Command, Automation, and State Manager. Parameter Store can also be used with other AWS services as well, for example, EC2, AWS Lambda, CloudBuild, which can then retrieve key data from this central repository.

The storage of your parameters within the store is modeled on a hierarchical model. This allows you to create a tiered structure of your parameters that provides better management and ease of use, especially when trying to locate the correct parameter. Due to the sensitivity of the data, the parameter values themselves can be encrypted. However, this hierarchical string to your parameter is not encrypted.

AWS Config

AWS Config is another key service when it comes to compliance, and it has a close relationship with AWS CloudTrail.

The service focuses on the importance of resource management within your organization. In a typical on-premises environment, you are often required to have a full understanding of your assets within the data centers, as well as knowing their configuration status and the current version of the software. Much of this is required for audits that are normally reviewed a number of times a year. This would often include the requirement to have an awareness of resource dependencies and communication paths. Trying to operate your own manual method of gaining this data within your AWS account, with its continuous fluctuation and scaling of resources, could prove both time consuming and a never-ending task, and this is why AWS Config was introduced. AWS Config helps with the overall resource management and compliance of your resources within your account.

The service itself is regional, which means if you are operating a solution across multiple regions, then you need to configure AWS Config within each of these regions if you want to use the different functions and features of the service. For services that are not region specific, such as IAM, there is an option within AWS config to monitor global services.

The AWS Config functions and features that I just referred to allow the service to do the following:

1. Perform a complete inventory of all supported resources within your AWS account.
2. Capture a snapshot in time, within a single region of all supported resources. This snapshot will also include configuration details and metadata information against the resource.
3. Record and capture the changes made to a support resource; this information is then retained within a **configuration item (CI)** that includes useful metadata about the change that occurred.
4. Retain the history of all configurations against individual resources; this can offer significant advantages from a compliance and auditing perspective.
5. Through the use of AWS Config rules, it's possible to check specific configuration compliance requirements against supported resources. Again, from a compliance perspective, this is invaluable.
6. Provide details of who, what, and when made a change to a resource through interacting and communicating with AWS CloudTrail, which, as we know, monitors API calls.
7. Notify you when a resource has undergone a change, allowing you to investigate, if required.

8. Provide a logical mapping of relationships between your supported resources. For example, which VPC a subnet belongs to and the resources that within that subnet.



For a full list of supported resources offered by AWS Config, please visit <https://docs.aws.amazon.com/config/latest/developerguide/resource-config-reference.html#supported-resources>.

As you can see, this service is capable of carrying out a lot of useful tasks, but how does it do this? Well, there are a number of components that all have their place within AWS Config, so let's take a look at some of these.

Configuration item

This is the core element of AWS Config, the **Configuration Item** (CI) is a JSON file that holds all of the information relating to a resource change. This would include configuration information, relationship data as well as metadata surrounding the resource related to the CI. The information that AWS config records is stored within CIs, and a CI is created every time a change occurs on a supported resource. In addition to this, AWS Config also creates CIs for resources with a direct relationship with the source resource that changed to ensure that the change didn't affect these resources too. For example, if you disconnected an EBS volume from an instance, a CI would be created for the volume as well as a CI for the instance that it was attached to as well.

Taking a closer look at the CI, it's comprised of the following five elements:

- **Metadata:** This gathers metadata relating to the resource itself, such as a time stamp, an MD5Hash used for comparing CIs to prevent duplication of data.
- **Attributes:** This shows common information across resources, such as ARN information, resource types, and tags.
- **Relationships:** This defines the directly affected relationships between resources.
- **Current configuration:** Interestingly, this captures the output that is received by running a **describe** or **list** API call, using the **AWS command-line interface (CLI)**.

- **Related events:** This is where the direct interaction with AWS CloudTrail comes into play. It will show the AWS CloudTrail ID of the API that was used which triggered the change that resulted in AWS Config creating the CI.

As explained earlier, all information used by AWS Config is captured within these CIs, meaning that the data can be used by other components of AWS Config, such as configuration streams, configuration history, and configuration snapshots, so let me explain what function each of these components does.

Configuration streams

Once a change has been captured within your environment and recorded within a CI, the CI is then sent to a configuration stream. This stream is simply an SNS topic, where it also receives data relating to other AWS Config notifications such as when the state of compliance changes for a resource that's being monitored by an AWS Config rule. So, as well as receiving these CIs, the configuration stream is used as a notification of alerts and tasks within AWS Config.

The configuration stream itself, the SNS topic, allows you to monitor events and changes in real time. Being an SNS topic, you can configure different endpoints, by setting the endpoint as an SQS queue, you could then programmatically digest and monitor this data as it's being recorded. Do bear in mind that if you chose email as an endpoint for this SNS, then your mailbox could get overwhelmed with data.



To help with this, AWS provides information on how to filter the incoming data; more information on this method can be found here:
<https://docs.aws.amazon.com/config/latest/developerguide/monitoring-resource-changes-by-email.html>.

Configuration history

The configuration history uses CIs to retain a complete history of the events of each resource that AWS Config monitors over a given period of time. This allows you to check the status and configuration of any resource from any point. This configuration history can be queried programmatically or by using the AWS Management Console where it displays the history within a visual timeline of events as shown with this VPC:



To use the AWS CLI, you would use the following API:

```
aws configservice get-resource-config-history
```

This API could be used with the additional parameter to drill down to specific resources and resources types, such as an EC2 instance:

```
aws configservice get-resource-config-history --resource-type
AWS::EC@::Instance --resource-id i-0ba7aa7ccbc805b45
```

All of the configuration history files are stored in a predefined S3 bucket selected during the configuration, for high availability of the data. Each configuration history file only contains the CIs for a particular resource type over a given period of time; for example, a configuration history file might show all changes to EBS volumes over a period of six hours.

Configuration snapshot

Again, like most other elements of AWS Config, snapshots also use the data provided by CIs. A configuration snapshot creates an entire point-in-time reference of all of your supported resources being monitored by AWS Config. This snapshot containing data from the CIs at the time in question will then be sent to S3 for storage.

Configuration recorder

The configuration recorder allows you to turn the recording capabilities of AWS Config on and off and is responsible for the generation of the CIs. When you first configure AWS Config, the configuration recorder is automatically started, but at any time, you can stop AWS Config from recording data. If you do stop the configuration recorder, then all recorded CIs up to that point will remain, along with the configuration history, but any new changes will not be recorded until it is turned back on.

Config rules

The **Config rules** are an essential component when looking to enforce specific compliance needs within your organization. These rules can be custom created or you can use a number of predefined rules that AWS has configured against some of the most common requirements:

ec2-volume-inuse-check	db-instance-backup-enabled	desired-instance-type
Checks whether EBS volumes are attached to EC2 instances. Optionally checks if EBS volumes are marked for deletion when an instance is terminated. EC2	Checks whether RDS DB instances have backups enabled. Optionally, the rule checks the backup retention period and the backup window. RDS	Checks whether your EC2 instances are of the specified instance types. EC2

The rules themselves are just a Lambda function that performs evaluation logic to determine the result and whether the resource in question is compliant or noncompliant. An important point to make is that any resource identified as non-compliant will still remain operational and in service, as they are simply highlighted as non-compliant, allowing you to determine the best course of action to rectify the issue. If the compliance status of a resource changes, a message is sent to the configuration stream.

The rules are evaluated every time there is a change against a resource, or they can be configured to run on a scheduled basis.

Resource relationship

The resource relationship provides a mapping from one resources connectivity and dependency to another. This logical map is stored within the CI. As you can see from this image, this VPC has relationships with other resources such as network ACLs, security groups, internet gateways, route tables, and so on:

EC2 VPC vpc-c3af7ca6

on August 22, 2018 11:45:42 AM British Summer Time (UTC+01:00)

Manage resource ?

15th March 2017 11:25:32 AM 1 Change

20th March 2017 11:03:05 PM 6 Changes

30th March 2017 2:58:08 PM 10 Changes

04th April 2017 4:11:00 PM 2 Changes

Now Calendar

View Details

Configuration Details

Amazon Resource Name	arn:aws:ec2:eu-west-1:730739171055:vpc/vpc-c3af7ca6	VPC ID	vpc-c3af7ca6
Resource type	AWS::EC2::VPC	State	available
Resource ID	vpc-c3af7ca6	VPC CIDR	172.31.0.0/16
Resource name	null	DHCP Options Set	dopt-612c3503
Availability zone	Multiple Availability Zones	Default VPC	true
Created on	Not available	Instance tenancy	default
Tags (0)			

Relationships 10

EC2 NetworkAcl	EC2 NetworkInterface	EC2 InternetGateway	EC2 RouteTable	EC2 SecurityGroup
acl-16ec4873	eni-cb3a4598	igw-1220de77	rtb-22c36547	sg-2fe54256
acl-e291ab86	eni-e0c7d49f			sg-39ac6a40
acl-ead2e08e				sg-4ce44335
				sg-59af7220
				sg-63d0b406

High-level process overview

1. During the initial configuration of the AWS Config, you specify which data (resource types) to record, which allows the configuration recorder to define which resources to capture and record data for.
2. A discovery of supported resources is then carried out by AWS Config, using the details from the configuration recorder.
3. If a change occurs against one of these resources, a CI is generated, which captures all the relevant data. The CI is also sent to the configuration stream to allow real time monitoring of events.
4. AWS Config performs a compliance check against the change to evaluate whether there is a change of state. If the state of compliance changes, a notification is sent to the configuration stream.

5. If there is a requirement to initiate a configuration snapshot, then AWS Config generates new CIs to create a point-in-time reference of the resources, which is then saved to S3.
6. After set periods of time, configuration history files for different resource types will be saved and sent to S3 for reporting and analysis as and when needed.

Summary

AWS offers a wide range of security and compliance services available to us as customers, and it's down to us to use them within our own environments to help safeguard and protect our resources and data. It's very easy to implement and add data and services within AWS, but understanding how to protect them effectively is a different matter.

You need to understand what data you have within your environment, become aware of how to monitor that data, and whether any changes are happening to not only the data, but the resources that are behind that data as well.

Maintaining compliance within a cloud environment can be a tricky task to implement effectively; due to the fluidity of the cloud, resources are changing all of the time, but you need to know when a change is a standard operational change, or a restricted unauthorized change. The services covered within this chapter can help you maintain a level of security, compliance, and reporting, to help you govern, protect, and control your environment through best practices.

Further reading

More information regarding some of the topics covered in this chapter can be found here:

- **Trusted Advisor Blogs:** <https://aws.amazon.com/blogs/aws/category/aws-trusted-advisor/>
- **AWS Config Best Practices:** <https://aws.amazon.com/blogs/mt/aws-config-best-practices/>
- **AWS Webinar: Automated Compliance and Governance: AWS Config and AWS CloudTrail – 2017 AWS Online Tech Talks:** https://www.youtube.com/watch?v=9g0u_05WBIG

16

AWS Security Best Practices

To help you configure, secure, and protect your AWS environment against both internal and external threats, there are a number of security best practices that you can adhere to that will help you maintain a high level of control across your environment. This chapter will focus on some of those best practices and how you can use them in your daily operations when running workloads across AWS.

I have always said that it is very easy to deploy resources within AWS, often with just a few simple clicks within the AWS Management Console, or a few commands run at the AWS CLI level. Either way, deploying the resources is the easy part, architecting and maintaining a strict level of access control, data protection, and availability is a different matter. Any data, service, or application running in AWS needs to be protected and secured if you want to minimize threats to both yourself and your customers.

Using the best practices recommended within this chapter, it will help you define additional security measures within your **information security management system (ISMS)**. By having these outlines defined, you can begin to build a set of policies and procedures that your organization must adhere to; this is also very helpful when it comes to audit and compliance controls.

We will cover the following topics in this chapter:

- Shared responsibility model
- Data protection
- **Virtual Private Cloud (VPC)**
- **Identity and Access Management (IAM)**
- EC2 security
- Security services

Technical requirements

To gain the most from this chapter, you should have an understanding of the following services:

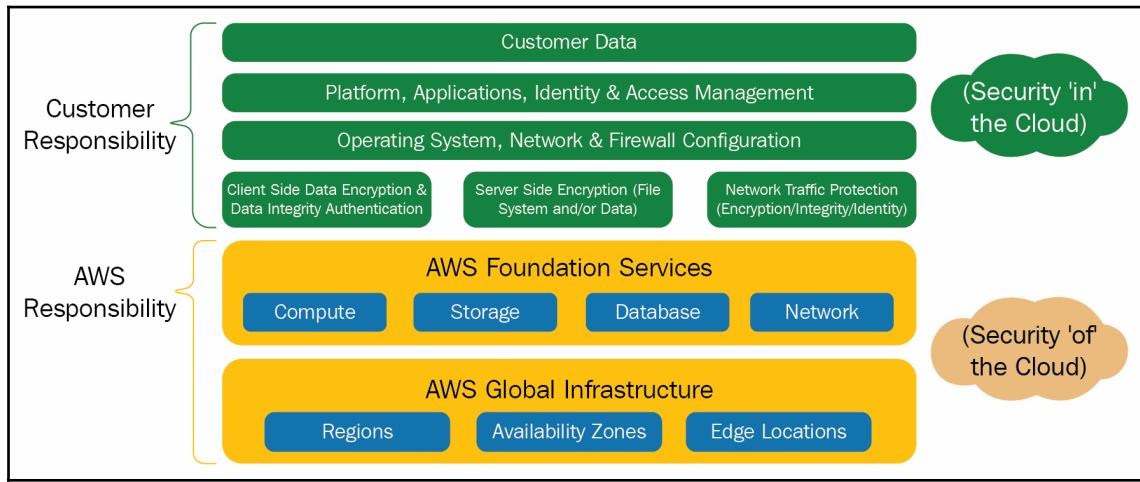
- Amazon Simple Storage Service (S3)
- Amazon Virtual Private Cloud (VPC)
- Amazon Elastic Compute Cloud (EC2)
- AWS Identity and Access Management (IAM)
- AWS Key Management Service (KMS)

You would have already come across some of these point already in this book; however, in this chapter, I want to focus on the key best practices and why you should be implementing them. This chapter will not focus on how to configure those services, it will explain the reasoning behind the importance of maintaining the best practice.

Shared responsibility model

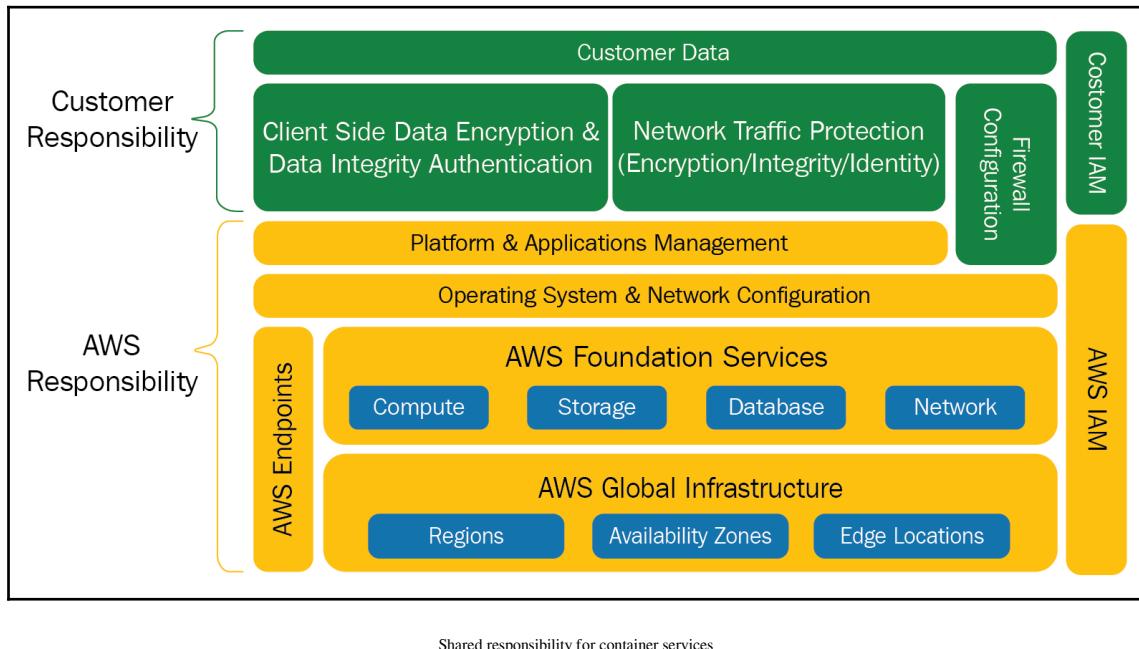
Understanding the AWS shared responsibility model is absolutely fundamental when it comes to using and deploying infrastructure within AWS. You need to be fully aware of where your responsibility starts and ends from a security perspective. How can you possibly architect your environment if you do not know where your boundary of responsibility ends? The simple answer is you can't. If you presume that another party, in this case, AWS, is maintaining a certain level of security of your infrastructure, you will almost inevitably leave a vulnerability within your infrastructure allowing a malicious user to take advantage of the weakness and gain unauthorized entry into your environment.

Many users of AWS are only aware of one shared responsibility model, which looks as follows and covers the infrastructure elements of AWS, such as EC2:



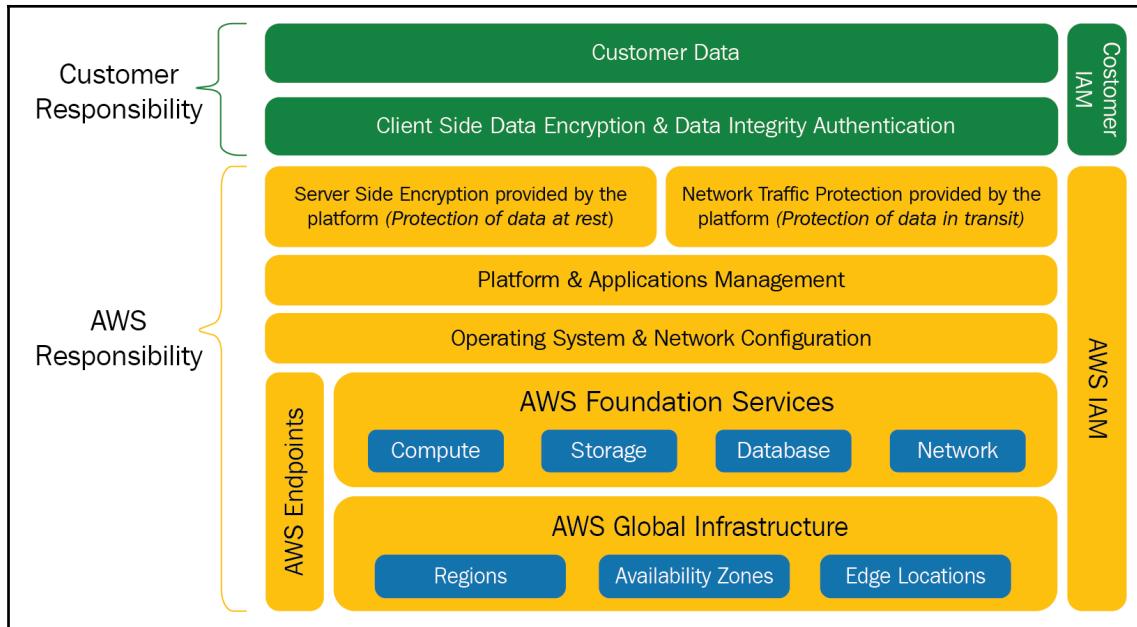
You would have likely seen this representation before where it clearly shows the responsibility boundaries between AWS and us and the customer. However, this is only applicable for those services that operate at the infrastructure level, where we have access to the underlying operating system (such as EC2). However, not all AWS services provide this level of access to us, for example RDS and EMR. For these services, we have no control over the underlying operating system, and so how does this affect the responsibilities?

For these services and others that fall into the category classed as **container** services the shared responsibility model then looks as follows:



As you can see, both the platform and application management and all OS and network configuration has become the responsibility of AWS and is no longer the customer's responsibility. Other components still remain the responsibility of the end user, such as the configuration of firewall management, and an example of this would be where RDS uses security groups, we as the customers would still be held responsible for configuring and implementing these.

It's this understanding between infrastructure-based and container-based services that is required as a best practice. However, there is one more responsibility model to include, and that is based around abstract services, such as S3, DynamoDB, and SQS, where, as a user of the service, we do not have access to the underlying OS or the platform at all. We simply have access to an endpoint and the underlying infrastructure and platform has been abstracted from us, the customers, completely. This model is as follows:



Shared responsibility for abstract services

Here, we can see that even more responsibility has moved to AWS, simply because as a user of these services we have less control over the management of the service itself.

So now we have three models that we need to be aware of to ensure that we have a solid understanding of the security strategy and methods we put in place to protect our data.

To reiterate, the three different methods are as follows:

- Shared responsibility model for infrastructure services
- Shared responsibility model for container services
- Shared responsibility model for abstract services

It is considered a best practice to be fully immersed with a clear definition of the roles and responsibilities between us the customers and AWS, the cloud service provider, across the range of different services that are offered by AWS. Ensure you have a solid grasp this concept, as it will help you to architect, implement, and reduce the overall risk level and exposure of your solutions. All other best practices stem from the foundations of this principle.

Data protection

From a data protection perspective, there are a number of elements to consider when looking at best practices.

Using encryption at rest for sensitive data

Encryption is such an important part of protecting your data from being seen and read by anyone that it is not intended for. AWS offers a range of services and methods for encryption, and here are some of the key points to be made aware of in preparation for your certification.

Encryption should be used for data when both at rest and in transit, especially for sensitive data. All too often, we have seen in the media vast amounts of data that have been mistakenly exposed and customer data leaked because it was all stored in plaintext.

KMS can be used to control your keys used for encryption, and a vast majority of AWS services are supported by KMS, allowing you to perform encryption at rest with relative ease. KMS is a managed service and provides you with the ability to generate and import customer master keys allowing you the ability to encrypt data and also generate data encryption keys, which are used by services such as S3 to perform data encryption. Use KMS where you can for your encryption requirements; being a managed service, it handles the security of your encryption keys without you having to secure them yourself.

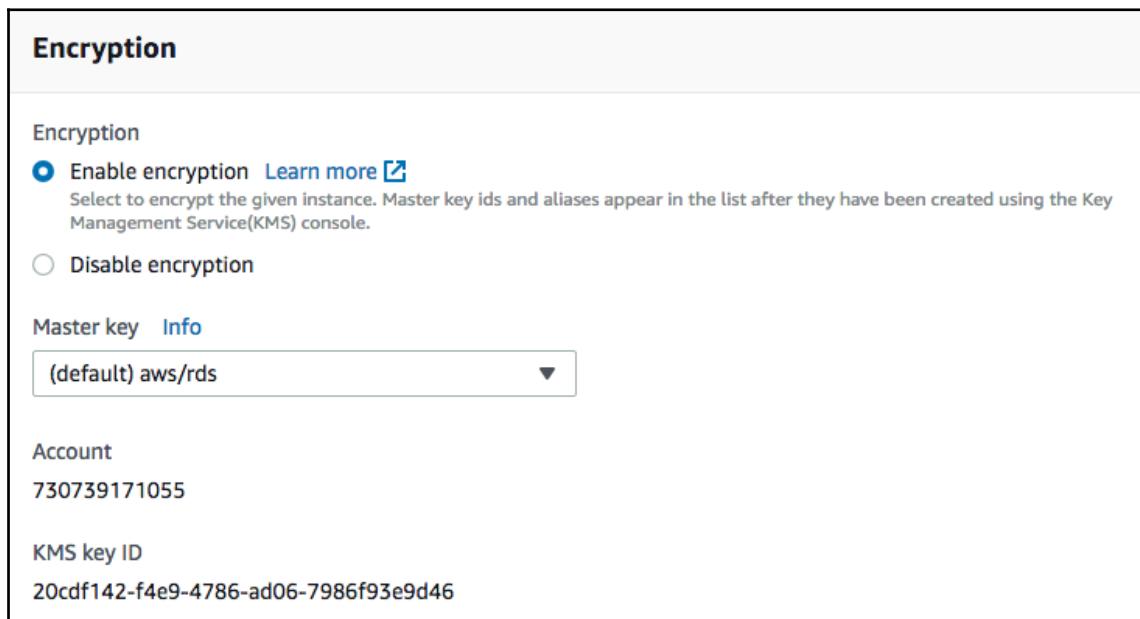
For any data that contains sensitive and customer information, it should be encrypted. In fact, this is a requirement of many compliance certifications, such as the **General Data Protection Regulation (GDPR)**. If you are responsible for maintaining security within your organization, ensure you are familiar with the requirements set out by specific compliance and governance controls that your business is bound by. Implementing security control to meet these standards from the start is easier to implement than post production deployment.

Taking advantage of encryption features built into AWS services

Many AWS services also offer their own level of encryption in addition to KMS. For example, S3 uses a myriad of encryption mechanisms including these:

- Server-side encryption options:
 - SSE-KMS (use of KMS)
 - SSE-S3 (encryption managed by AWS)
 - SSE-C (encryption managed by customer-managed keys)
- Client-side encryption options:
 - CSE-KMS (use of KMS)
 - CSE-C (encryption managed by customer-managed keys)

Another example is **Relational Database Service (RDS)**. When creating your database, you can select **Enable encryption** on the **Configure advanced settings** screen:



This encryption will by default encrypt all of your storage at rest, all of your DB snapshots, backups, and also any read replicas that you may have in place. KMS is being used to apply this encryption and the first time RDS encryption is enabled it will create the default master key of AWS/RDS. The encryption process itself is managed by RDS without the end user having to perform any other actions.

In addition to KMS encryption, the service also utilizes other methods of encryption at the platform level, Oracle and SQL Server **Transparent Data Encryption (TDE)**, which could be used in conjunction with the KMS keys mentioned previously, although this would add an additional minimal impact to the performance of the database. MySQL cryptographic functions and Microsoft SQL Transact-SQL cryptographic functions are also possible encryption mechanisms with RDS. Although these additional encryption mechanisms will not be mentioned on the exam, I am just trying to demonstrate that you do not only have to rely on KMS to perform encryption at rest for your data services, many of them come with their own features of encryption.

These encryption mechanisms should be explored when you are using services that offer them to understand how they could help you achieve the best level of protection of your data.

Having encryption applied to your data when at rest is one thing, but it's also just as important that your data does not get intercepted between services or while connecting to your services to transfer data, for example. Applying encryption in transit is also a best practice when it comes providing additional data protection.

Using encryption in transit for sensitive data

AWS services are often accessed across the internet, which is inherently insecure. As a result, AWS supports IPSec and SSL/TLS when protecting data in transit with all AWS Services.

When using HTTPS (HTTPS over SSL/TLS) with your web servers, it can cause additional processing and computing resources to terminate the session; as this scales to hundreds or tens of thousands of connections, it can cause a significant impact on your fleet. To help alleviate this issue, it's a good idea to implement an elastic load balancer that can handle the termination of HTTPS and scale accordingly, therefore reducing the impact upon the web servers when it comes to processing the request.

Protecting against unexpected data loss

In addition to encryption, which protects your data from being read by unknown and malicious users, data protection also covers the potential loss of data that may occur and so you need to implement methods and techniques to help you protect against this threat. This threat might come internally, or it might be from an external source trying to gain access to your data.

Data backups are the obvious method of ensuring you maintain your data, and there are numerous ways to do this. Thankfully, AWS has many supported and built-in features across a range of services that allow you to perform backups, from EBS snapshots of your block storage, to automatic backups of your databases. I am not going to run through each and every possible option of providing a backup to your data, whether it be object, block or file, but it is important to maintain a back-up strategy for your environment.

When doing so, think about the location of the backup data: are you backing it up to a different availability zone or region? The reason for doing so is all a part of business continuity. If you stored your back-up data in the same AZ as the source data and the AZ failed, you have lost access to your backup data. Similarly, depending on the criticality of that data to the business, should your back up data be stored within a different region than the source data in the event of a complete regional failure? These are all questions that need to be asked from a business perspective and need to be answered.

It is considered to be best practice store your backups in a different AZ at regular intervals. Again, this is dependent on the importance and criticality of the data that you are backing up.

As a part of your backup strategy, you should have a clear definition as to your DR strategy, and whether you are going to adopt a recommended method as defined by AWS, which includes the following:

- Backup and restore
- Pilot light
- Warm standby
- Multisite active-active

As well as backups of your data to help protect against data loss, there are additional mechanisms built into some services, such as S3.

S3 offers a number of different methods to help against data loss, either unintentional or intentional. These include the following:

- MFA delete
- Life cycle policies
- Versioning

Using S3 MFA delete to prevent accidental deletion

MFA delete is used to protect against the accidental deletion of an object. Using multi-factor authentication ensures that the request to delete the object was intentional by way confirming the deletion with a secondary authentication method requiring the user to enter a random six-digit number provided by their MFA device.

Using S3 lifecycle policies

Using lifecycle policies allows you to automatically manage the storage class of the S3 object. For governance and compliance reasons, you may need to retain the data for a number of years without the need for accessing it, this is common among financial institutions. Leaving this data within S3 for that period of time will not only lead to an increased cost, but it also makes it vulnerable to accidental exposure if your bucket or objects were to be made public accidentally. Using life cycle can automatically move the data into Glacier into protected vaults that can't be made public.

Implementing S3 versioning to protect against unintended actions

Enabling versioning on an S3 bucket is a very effective method of protecting objects against accidental deletion and also from unintended actions against the object itself. Every time the object is modified, a new version of the object is saved within the bucket. This allows you to revert back to a previous version of the object at any point. So, if an object did get deleted, the previous version would still be within the bucket.

One point to bear in mind with versioning is that it uses additional storage space for every version of the same object.

Virtual Private Cloud

The **Virtual Private Cloud (VPC)** is your own segment of the AWS cloud where you can deploy your resources and build solutions. Through the use of different subnets, route tables, and an internet gateway, you can configure your VPC to communicate with the internet, in addition to allowing traffic from the internet to access your resources, such as a web server.

The creation of your VPCs can be very simple, but understanding how traffic and boundaries are implemented is a security must. There are a variety of methods to control traffic and access to different network segments. To isolate and control network traffic, you should adopt as many of these options as possible

Using security groups to control access at an instance level

You should always use security groups to allow you to manage network traffic at an instance level, controlling what traffic is allowed to and from instances. They allow you to group resources together that have similar functionality. It's important to remember that security groups are stateful by design and you can apply multiple security groups to a single resource.

Using NACLs to control access at a subnet level

Similar to security groups, **Network Access Control Lists (NACLs)** also control network traffic, but this time at the network level. NACLs work at the subnet level and control the flow of network traffic in and out of the subnet. Unlike security groups, NACLs are stateless, meaning you have to add explicit response rules for network traffic. When using NACLs in conjunction with security groups, they can be an effective barrier against intrusion. As NACLs operate at the subnet level, you can block restricted traffic to a whole subnet with ease.

Implementing the rule of least privilege

For both security groups and NACLs, you should apply configure them based on **the rule of least privilege**. This essentially means that you should be restrictive as possible on the type of ports that are used and the source and destination used within the rules. Try to steer away from using **any** with the fields; for example, if your application only communicates across a single port, then just add that single port number. The more specific and restrictive, the greater the security of your infrastructure. The wider the access scope is, the greater the chance of malicious infiltration. The same applies when specifying the source and destination entries: narrow it down to the smallest range possible. Don't open up access to a whole subnet if only a single resource requires the access; instead add the single IP address.

Implementing layers in your VPC

When architecting your VPC, you should aim to layer your infrastructure between subnets; this allows you to layer your security, with each having its own defined set of restrictions. This also allows you to group similar infrastructures together, which in turn allows you to be more restrictive with each set of NACL and security group rules.

Creating Flow Logs to obtain deeper analysis of network traffic

VPC Flow Logs should also be created to allow you to gain a greater insight into your network traffic. VPC Flow Logs allow you to capture IP traffic information that flows between your network interfaces of your resources within your VPC and can be applied to your network interfaces on a single instance, a subnet, or your entire VPC. The data captured from your VPC flow logs is then sent to Amazon CloudWatch to allow you to analyze the data in greater detail.

Identity and Access Management

IAM is probably the most common security service of IAM, so many service integrate with IAM and the features that it provides, and so it makes perfect sense to ensure you are aware of some of the best practices from an access control perspective.

Avoid sharing identities

You should not share an IAM user identity among a shared group of users. This makes it very difficult from a security risk perspective, especially when an incident occurs as a result of an action that was carried out by that identity. It's easy to track what actions a user has carried out by analyzing the logs from CloudTrail. However, if this user is shared by multiple parties, it's not so easy to identify the individual who carried out the action. Everyone should have their own log-on credentials if they need access to AWS.

Using MFA for privileged users

You should always look to use multi-factor authentication for users that have privileged permissions, and it's a **must** for the root account, administrator, and any identify considered to have power user access.

If you are unsure if an identify should be using MFA based on their privileges, err on the side of caution and configure that identify for MFA. If the user has the ability to launch and create resources and delete resources, then MFA is recommended. If accounts with privileges such as this become compromised, then the effects of the intruder can cause considerable damage to your infrastructure. With the additional second authentication requirement, it significantly reduces the chances of compromise.

Using roles

If you have an application installed on an instance and that application required access to other AWS services, you might be tempted to store credentials locally on the instance that the application can use to authenticate with. However, this is considered poor practice; instead you should use IAM roles that can be associated with the instance. This way, credential information is removed from the local instance removing the chance of them being compromised. Instead the permissions associated to the role are adopted by the instance when access to another resource is required.

Password policy

You should always implement an effective password policy to meet the standards required set out by your internal security policies. AWS provides a wide variety of options as follows to allow you to be as specific as necessary:

Modify your existing password policy below.

Minimum password length:

Require at least one uppercase letter ⓘ
 Require at least one lowercase letter ⓘ
 Require at least one number ⓘ
 Require at least one non-alphanumeric character ⓘ
 Allow users to change their own password ⓘ
 Enable password expiration ⓘ
Password expiration period (in days):
 Prevent password reuse ⓘ
Number of passwords to remember:
 Password expiration requires administrator reset ⓘ

Apply password policy **Delete password policy**

The longer your password length, the greater the strength of that password. Also, ensuring you specify that different characters are used also enhances the robustness of the password. Aim to use as many of these features as possible.

Assigning permissions to groups instead of to individual users

Use groups to manage more than one user with similar access requirements. By assigning permissions to groups and then assigning users to groups, it reduces the amount of administrative effort required to grant/deny access to resources. This also reduces the chances of mistakes, as you are only modifying a single policy instead of a number of policies assigned to individual users.

Rotating your access keys

Your access keys are used for programmatic access to your resources; for example, when accessing your environment via the AWS CLI instead of the Management Console, it's best practice to rotate your keys regularly, think of it as the same process of changing your password on a regular basis; it's for the same reasons. The longer the keys are left in operation without rotation, the longer a malicious user has to try to compromise them.

Assigning permissions according to the rule of least privilege

When creating your access policies, you should ensure that the access you are assigning and the minimal permissions required to perform the task that the identity is required to perform. For example, you would not assign the following policy if the identities associated with that policy only needed PUT access to a specific bucket:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "s3:*",  
            "Resource": "*"  
        }  
    ]  
}
```

With this policy, it gives the identities associated full access to all permissions of S3 across all buckets within your AWS account. Giving the identity far greater access than they need allows for actions that can perform any S3 action, which would mean objects could get deleted or even made public. If their role does not require them to carry out any of these functions, then the policy should reflect the requirements.

Re-evaluating permissions and deleting accounts

Over time, roles and responsibilities change, and with these changes, the required permissions are also changed. Ensure that the users within the groups that you have configured are still applicable to that group. If they are not, remove the user. Also, ensure you have a policy of how to remove credential information for when a user no longer requires any access at all.

An additional risk with loose policy permissions is that if the credentials were to be compromised then the malicious user would have a far greater potential to cause disruption and gain access to data that otherwise would have not been possible had the correct permissions sets on the policy.

Do not use the root account as an operational user

When you first create your AWS account you create a root user, this account has access to everything within AWS. As a result, it is highly recommended that you do not use this account to perform everyday tasks and operations; instead this account should be used to create your first IAM user with privileged access. You should then use this new IAM user to continue administering your account and setting up other identities. Once you have created this power user, you should disable the root accounts access and secret access keys. MFA is a must for your root account to provide an additional level of authentication.

EC2 security

EC2 is one the services that is covered the most within the certification, and you need to have a solid understanding of all things EC2, which includes the best practices, when it comes to securing them.

Implementing a patching strategy

As we know from the shared responsibility model, when it comes to managing the security of our instances, we, the customers, are responsible for securing the OS itself, and this includes downloading and installing the latest patches that are released by the OS vendors. New patches often have fixes for previously identified security weaknesses. Failure to install the latest patches could cause an unexpected security exposure that could be exploited.

You could leverage the capabilities of the EC2 Systems manager to help you manage and control the patches required for your EC2 fleet. In addition to the Systems Manager, you could also bootstrap your instances to ensure that upon booting, all the latest patches are downloaded and installed.

Controlling access with security groups

I covered security groups in the VPC section of this chapter, but restricting access with security groups is essential for maintaining a restriction of access at the protocol and port level. Again, implement a rule of least privilege when it comes to designing and implementing your rules in your security groups, and only allow access to the ports and protocols that the instance needs to communicate with to perform its tasks. To help maintain a level of encryption in transit, you can ensure that the protocols associated with your security group are only secure and encrypted protocols, such as HTTPS or SSH. In addition to this, restrict the source and destination to only those resources that you need the resource to connect to.

Encrypting sensitive data on persistent storage

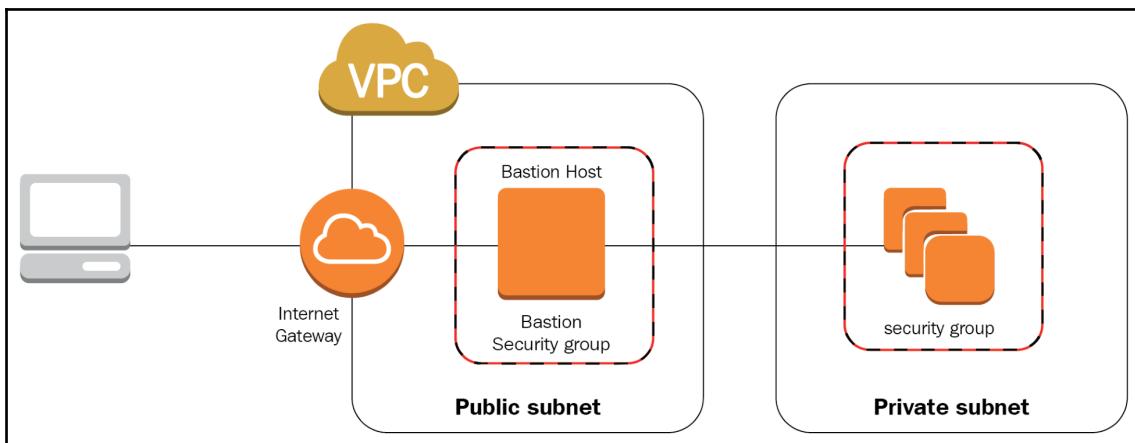
If you are storing critical and sensitive data, then you should be using persistent storage, such as Amazon EBS, which can be attached and detached from EC2 instances. Using EBS also provides an encryption option for your data stored at rest, which will also encrypt all your snapshots of the same volume. This encryption is managed by the **Key Management Service (KMS)**.

Harden the operating system

Although AWS provides security mechanisms to control the communication among resources with security groups and NACLs, it's a good idea to implement your own method and measures of security on your EC2 instances. There are a wide variety of resources that explain the techniques and actions to take when you need to implement an additional level of hardening with your operating system, whether it be Linux or Windows.

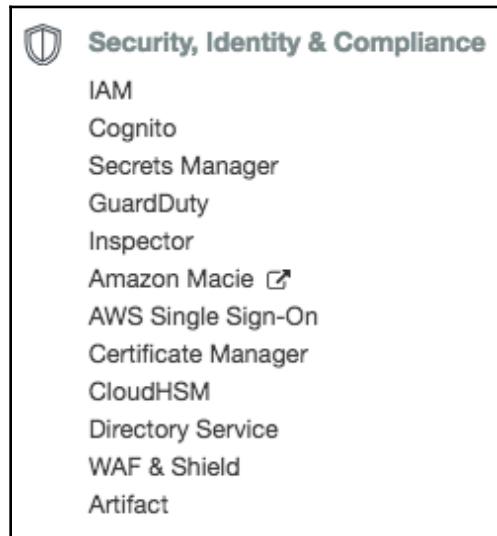
Using Bastion hosts to connect to your EC2 instances

Bastion hosts are instances that sit in your public subnet of your VPC and are used to control and allow external connectivity to your private instances within your private subnets. External connections can connect often via SSH or RDP, where the bastion host will act as a "jump" server to your private instances. By configuring security groups and NACLs, the bastion essentially acts as a bridge to your private instances via the internet. It is essential that you harden your OS of your bastion host as this provides a connection from the public internet to your private instances:



Security services

AWS has a dedicated category for their AWS security centric services within the AWS Management Console, and I am sure over time this list will continue to grow as new demands and threats are identified within the ever-growing field of cloud computing. At the time of writing this book, the category listing looks as follows:



To help you in the ever continuing task of securing your infrastructure its best practice to use these services where required, some of these services have been covered within this book and are accessed within the certification; others do not play as much of an active part for the *Solution Architect - Associate*. However, I feel it's good to have an awareness of what each of these services does at a high level to help with preparation and to be aware of the purpose of each security service:

- **IAM:** The Identity and Access Management service is used to help you manage access to your resources by creating identities with specific permissions allowing or denying access to your resources.
- **Cognito:** Cognito is used to help you manage access control to your web and mobile applications. It can scale quickly and easily to millions of users and supports federated access through web identity providers in addition to enterprise identity using SAML 2.0.

- **Secrets Manager:** This service allows you to safeguard secrets to access your services, applications, and resources. For example, you can easily manage database credentials and API keys. Access to the secrets are governed by API keys with fine-grained permissions.
- **GuardDuty:** Amazon GuardDuty is a regionally-based intelligent threat-detection service that allows users to monitor their AWS account for unusual and unexpected behavior. This is achieved by analyzing CloudTrail event logs, DNS logs, and VPC flow logs.
- **Inspector:** Amazon Inspector is a managed service used to help find security vulnerabilities within your EC2 instances and any applications running on them. This is automatically achieved via a series of assessments against a set of defined resources, based on hundreds of best practices and known security weaknesses.
- **Amazon Macie:** Amazon Macie provides an automatic method of detecting, identifying, and also classifying data that you are storing within your AWS account. Backed by machine learning, your data is actively reviewed as different actions are taken, spotting access patterns and user behavior by analyzing CloudTrail event data.
- **AWS Single Sign-on:** AWS SSO simplifies the process of managing SSO access to multiple AWS accounts within your AWS Organizations by presenting a portal that users can sign in to using existing credentials.
- **Certificate Manager:** This is to manage and deploy both private and public SSL/TLS certificates, enabling you to secure network communications. Certificates can be deployed quickly and easily across Amazon CloudFront distributions, ELBs, and APIs on the API gateway.
- **CloudHSM: Cloud Hardware Security Module (HSM)** is a hardware device for securing cryptographic key storage. It is a fully managed service and allows you to generate and use your own encryption keys within AWS using FIPS 140-2 Level 3 validated HSMs.
- **Directory service:** This service simplifies the task of configuring and implementing Amazon Cognito, Amazon Cloud Directory, and Microsoft AD within your AWS account.

- **WAF and Shield:** The AWS Web Application Firewall (WAF) helps to prevent websites and web applications from being maliciously attacked by common web attack patterns, such as SQL injection and cross-site scripting. AWS WAF also works closely with AWS CloudFront distributions to filter between legitimate and harmful inbound requests. AWS Shield is used to protect your applications from a **Distributed Denial of Service (DDoS)** attack using two tiers of protection: standard and advanced.
- **Artifact:** This service is used as a central repository for all of AWS's compliance-related reports and agreements. For example, reports relating to **Service Organization Control (SOC)** reports, **Payment Card Industry (PCI)**, and certifications from accreditation bodies. You can also find other agreements such as the **Nondisclosure Agreement (NDA)**.

There are also two more services that I would like to mention, although they are not within the security, identity, and compliance category. They are AWS CloudTrail and AWS Config. When used together, these services form a great security-auditing tool to track, record, and assess changes within your account.

- **CloudTrail:** CloudTrail tracks and records AWS API requests made within your account. These API calls can be programmatic, from an SDK, the AWS CLI or from within the AWS management console, or even from a request made by another AWS service.
- **Config:** AWS Config can capture resource changes, act as resource inventory, discover supported resources running within your environment, store configuration history, provide a snapshot in time of current resource configurations, notify you when a change has occurred on a resource, integrate with AWS CloudTrail, and enforce rules that check the compliance of your resources.

Summary

Implementing security best practices requires an understanding of the shared responsibility model; once you understand your responsibilities, you can begin to architect and implement additional levels of security throughout your environment. Adhering to best practices from the outset will significantly help to protect your data and resources from a wide variety of threats, risks, and exposures, both internally and externally.

This chapter has focused on some of the common security best practices, some of which may be referenced within the certification. Understanding the reasoning behind the best practice helps you to protect against the threat in a structured way.

New threats are being defined and exposed all the time, and so it's important to keep revisiting your security strategy, to ensure that it is still meeting all the requirements stipulated within your **Information Security Management System (ISMS)**.

Further reading

- **AWS Security Best Practices Whitepaper:** https://d1.awsstatic.com/whitepapers/Security/AWS_Security_Best_Practices.pdf
- **How to encrypt an EBS Volume:** <https://cloudacademy.com/blog/how-to-encrypt-an-ebs-volume-the-new-amazon-ebs-encryption/>

17

Web Application Security

Applying security within AWS requires a layered approach, one of those layers can be centered around your web application infrastructure. Ensuring you implement controls and safeguards against your web applications is essential. By their very nature, they are external facing to the open public and with that comes additional threats and risks. As soon as your services are made available to the public, it will not be long before someone, somewhere will be trying to access your data and application in a malicious and harmful way. This chapter will focus on some of the methods and techniques that can be used to help mitigate such threats and exposures.

The following topics will be covered in this chapter:

- AWS **web application firewall (WAF)**
- AWS Shield
- AWS Firewall Manager

Technical requirements

To gain the most from this chapter you should have some knowledge or experience with using the following services:

- Amazon CloudFront
- Application Load Balancer
- Familiarity of the OSI model

Within this chapter I will be covering the following services and explaining what they are and what they are used for:

- AWS WAF
- AWS Shield
- AWS Firewall Manager
- Amazon CloudFront security features

AWS web application firewall

AWS WAF works closely with Amazon CloudFront and **Application Load Balancers (ALBs)** and its primary function is to prevent your web applications from being subjected to intrusion by common attack patterns. By working in conjunction with CloudFront distributions and ALBs, AWS WAF can dictate how these services respond to web requests based on preconfigured conditions. This allows all HTTP and HTTPS requests to be filtered and identified as genuine or damaging inbound requests which are then either allowed or blocked as required.

There are three main component of the AWS WAF service in how it configured to help protect your web applications, these components are comprised of the following:

- Conditions
- Rules
- Web access control lists (ACLs)

Each of these is configured in order, you must first start with configuring conditions, which are then added to your rules, which are then in turn added to your web ACLs.

Conditions

The conditions section within WAF defines the first part of configuration and customization helping you to select the components of the web request you want to monitor against.

At the time of writing, there are six configurable conditions which are as follows:

- **Cross-site scripting (XSS):** This threat aims to access client-side data such as a stored cookie information, via embedded scripts within web pages that are normally trusted by the client. As you can see from the following screenshot, you are able to create the condition based on particular parts of the request, such as the header or the query string:

Create cross-site scripting match condition

A cross-site scripting match condition lets you allow or block web requests that contain scripts that can exploit vulnerabilities in web applications. [Learn more](#)

Name*

Region*

Use global to create WAF resources that you would use with CloudFront distributions and other regions for WAF resources that you would use with ALBs in that region.

Filter settings

Specify the settings that you want to use to allow or block web requests. If you add more than one filter to a cross-site scripting match condition, a web request needs to match only one of the filters for the request to match the condition. (The filters are ORed together.)

Part of the request to filter on 0

Transformation

Header
HTTP method
Query string
Single query parameter (value only)
All query parameters (values only)

Filters in this cross-site scripting match condition

Part of the request to filter on
Body

This condition has no filters.

* Required

Creation of cross-site scripting match condition

- **Geo match:** This provides a method of restricting requests based on a geographic location. Do bear in mind that if you want to use this function in WAF you first have to disable any geo restriction settings that you might be using in your associated CloudFront distribution. Using the **Location** field you can scroll down to find the country of origin that you are interested in:

Create a geo match condition

A geo match condition lets you allow, block, or count web requests based on the geographic origin of the request. [Learn more](#)

Name* PackT_Geo

Region* Global (CloudFront)
Choose Global (CloudFront) to create AWS WAF resources to use with CloudFront distributions in all AWS Regions. Choose a specific AWS Region to create AWS WAF resources to use with an Application Load Balancer in that region.

Filter settings

Add one or more locations.

Location type* Country

Location* Please select

Filter

- Afghanistan - AF
- Aland Islands - AX
- Albania - AL
- Algeria - DZ
- American Samoa - AS
- Andorra - AD
- Angola - AO

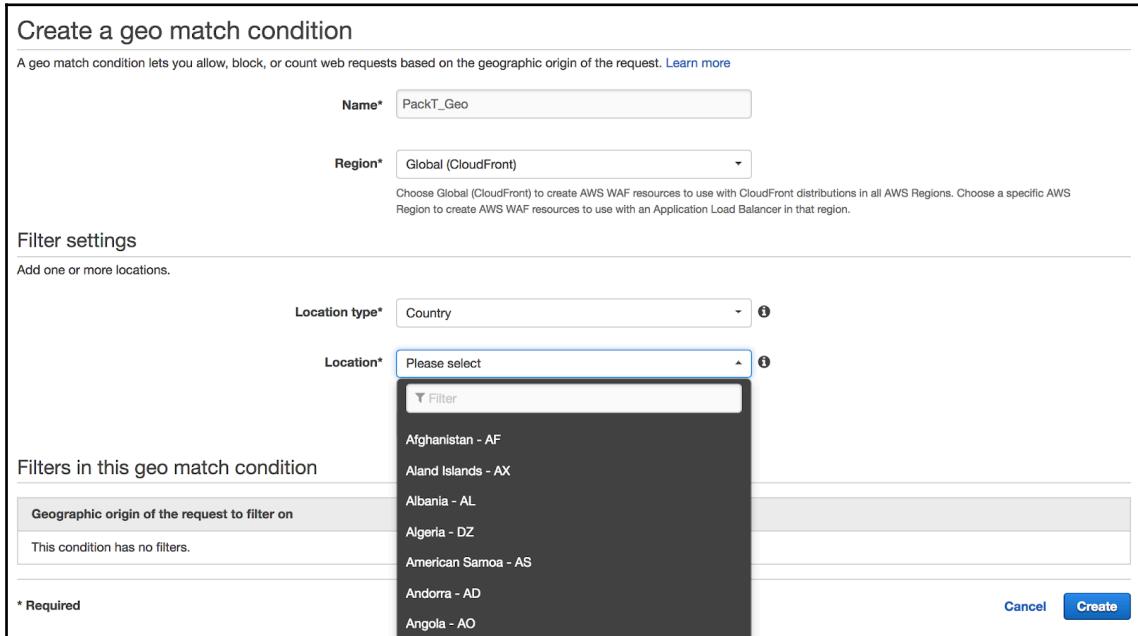
Filters in this geo match condition

Geographic origin of the request to filter on

This condition has no filters.

* Required

Cancel Create



- **IP addresses:** This condition will allow you to restrict web requests based on a specific source IP address or a range of IP addresses:

Create IP match condition

An IP match condition contains a list of IP addresses and/or IP address ranges. These IPs are the source of the requests that you want to allow or block. [Learn more](#)

Name*

Region* [Global](#)

Use global to create WAF resources that you would use with CloudFront distributions and other regions for WAF resources that you would use with ALBs in that region.

IP addresses

Add one or more IP addresses or IP address ranges by using CIDR notation.

IP Version* IPv4 IPv6 [?](#)

Address* [?](#)

AWS WAF supports /8 or any range from /16 to /32 CIDR blocks for IPv4
Examples:
For a single IP address, please specify like 192.0.2.44/32
For an IP range from 192.0.2.0 to 192.0.2.255, please use 192.0.2.0/24

[Add IP address or range](#)

Filters in IP match condition

IP address of the request to filter on

This condition has no filters.

* Required [Cancel](#) [Create](#)

- **Size constraints:** Using mathematical operators, such as $>$, $<$, or $=$, you are able to set conditions of the request to filter against a specific size of different parts of the request, such as the header, URI, or body.

Create size constraint condition

A size constraint condition lets you allow or block web requests based on the lengths of specified parts of the request. [Learn more](#)

Name*

Region* Use global to create WAF resources that you would use with CloudFront distributions and other regions for WAF resources that you would use with ALBs in that region.

Filter settings

Specify the settings that you want to use to allow or block web requests. If you add more than one filter to a size constraint condition, a web request needs to match only one of the filters for the request to match the condition. (The filters are ORed together.)

Part of the request to filter on

Comparison operator

Size (Bytes)

Transformation

Add filter

Filters in this size constraint condition

Part of the request to filter on

This condition has no filters.

* Required Cancel

- **SQL injection attacks:** SQLi attacks performed against a database can access, modify, and cause significant damage by executing administrative level functions. This is carried out via a malicious SQL query entered within an entry field on the remote database where it is executed. Much like the XSS condition, you can filter on a specific part of the request to look for these attacks:

Create SQL injection match condition

A SQL injection match condition lets you allow or block web requests that can exploit vulnerabilities in applications that use SQL databases. [Learn more](#)

Name*

Region*

Use global to create WAF resources that you would use with CloudFront distributions and other regions for WAF resources that you would use with ALBs in that region.

Filter settings

Specify the settings that you want to use to allow or block web requests. If you add more than one filter to a SQL injection match condition, a web request needs to match only one of the filters for the request to match the condition. (The filters are ORed together.)

Part of the request to filter on ⓘ

Transformation

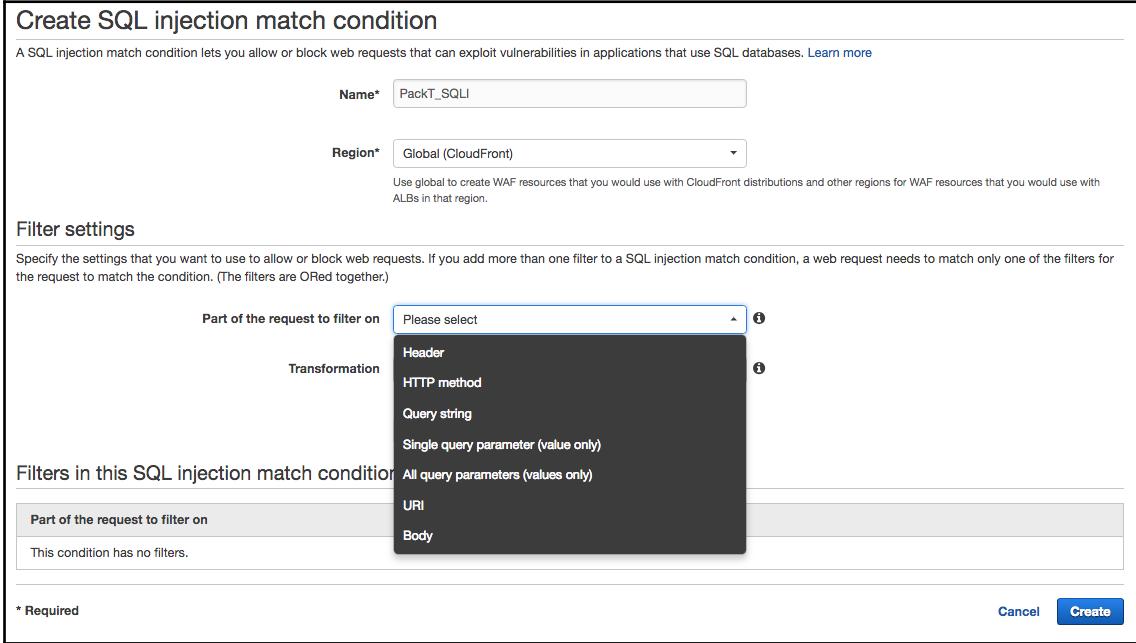
Header
HTTP method
Query string
Single query parameter (value only)
All query parameters (values only)

Part of the request to filter on ⓘ

Body

This condition has no filters.

* Required Cancel Create



- **String and regex matching:** These conditions are based upon strings within the request, again, you are able to specify different parts of the request to filter the string against:

Create string match condition

A string match condition contains a list of the strings that appear in web requests that you want to allow or block. [Learn more](#)

Name* PackT_String

Region* Global (CloudFront)

Type* String match

String match

Regex match

Filter settings

Specify the settings that you want to use to allow or block web requests. If you add more than one filter to a string match condition, a web request needs to match only one of the filters for the request to match the string match condition. (The filters are ORed together.)

Part of the request to filter on Please select

Match type Please select

Transformation Please select

Value is base64-encoded

Value to match* Type a plain text string

Add filter

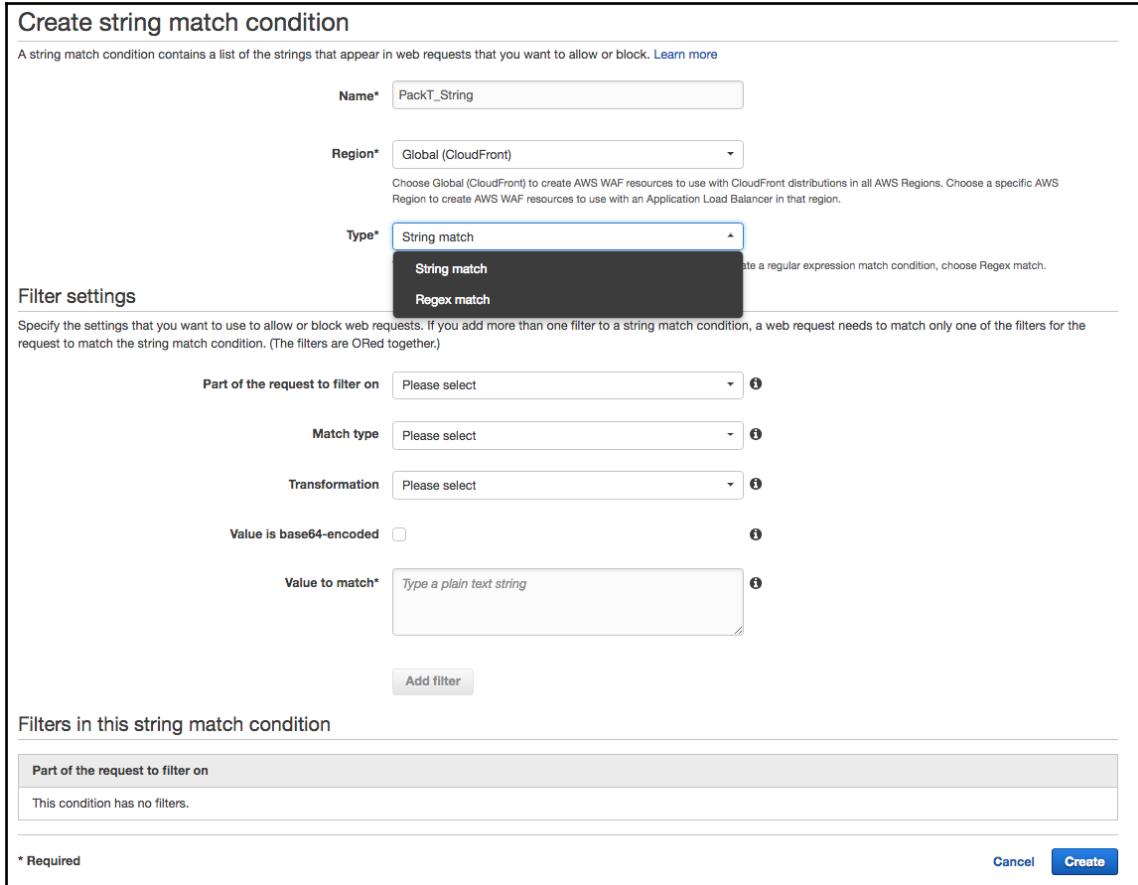
Filters in this string match condition

Part of the request to filter on

This condition has no filters.

* Required

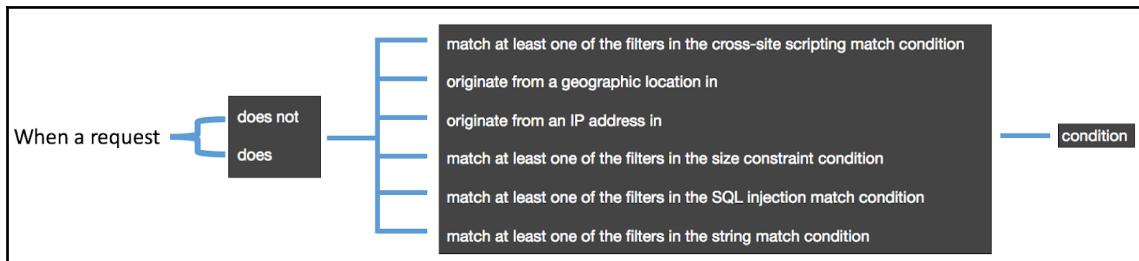
Cancel Create



Rules

Once conditions are defined within WAF, it's then possible to move onto the next stage of the configuration process, which involves WAF rules. These rules enable you to package the different conditions together. If you have more than one condition in a rule then each condition identified is ANDed with the previous. A web request only matches a rule when it meets **all** conditions within that rule. For example, you might have a rule with 2 conditions, an IP address condition and a geo match condition. In this case, a match for the rule would only be found when the web request met both the IP address and geo match conditions that you specified. If it only met one of these conditions, the rule would not be used as a match. When a match is found, the corresponding action of the rule will be taken.

During the compilation of your rules, different options can be defined on each condition, depending on what the condition is. For every condition added to the rule, you will need to specify the following for the requests that match the rule:



There are two types of rules available:

- Regular rule
- Rate-based rule

Rate-based rules are exactly the same as regular rules, however, they count the requests received from the same IP address across a 5 minute time frame. You can specify the maximum number of requests received which has to be above 2000. Anything below 2000 is considered a regular rule. Once the limit is reached, the requests are blocked, as shown in the following screenshot:



The actions of **Allow**, **Block**, and **Count** are still not defined at this stage, this decision is selected within the web ACL itself which is the final configuration stage allowing it to be configured for your CloudFront distribution or ALB.

Web ACL

The web ACL is used to create rule sets that are then associated to either your CloudFront distributions or your ALB. A Web ACL is comprised of one or more Rules, which are read in order, and each rule has an action associated, **Allow**, **Block**, or **Count**.

If a rule has an **Allow** action the web request is considered safe and legitimate and is allowed through to the web application infrastructure to be processed. If the action is set to **Block** the request is dropped immediately and no further processing is carried out. **Count** actions simply count the number of requests that met that rule.

Create rules

Rules contain the conditions that you want to use to filter web requests. You add rules to a web ACL, and then specify whether you want to allow or block requests based on each rule. [Learn more](#)

Add rules to a web ACL

Rules

If a request matches all of the conditions in a rule, take the corresponding action				
Order	Rule	Action		
▼ 1	PackT	<input checked="" type="radio"/> Allow <input type="radio"/> Block <input type="radio"/> Count <input type="button" value="X"/>		
▲ 2	Stuart_Scott	<input checked="" type="radio"/> Allow <input type="radio"/> Block <input type="radio"/> Count <input type="button" value="X"/>		

If a request doesn't match any rules, take the default action	
Default action*	<input type="radio"/> Allow all requests that don't match any rules <input checked="" type="radio"/> Block all requests that don't match any rules

* Required

In this example we can see that this web ACL has two rules, both configured to **Allow** traffic should the request match the conditions within those rules. You will also notice that there is a default action at the bottom of the ACL. This ensures that AWS WAF can direct any web request that doesn't meet any rule within the web ACL. In this example, any web request that doesn't meet either of the two **Allow** rules, it will be blocked.

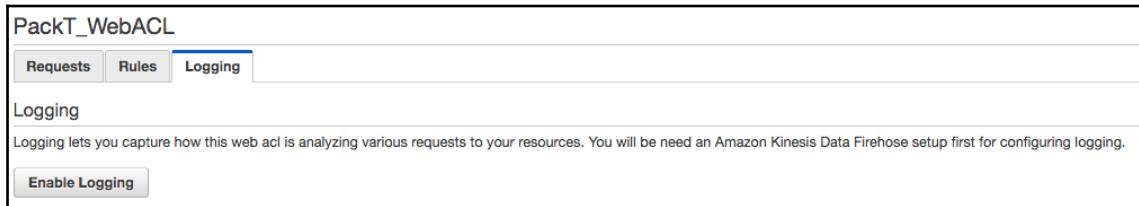
When a request is blocked the user will receive a 403 error to their browser indicating that access to the resource is forbidden.

As the rules are read in order, as soon as a match is found with a rule the request will take the action of that rule, even if there is another rule within the web ACL that would also be a match, so care needs to be taken when organizing your rule positioning within a Web ACL.

Monitoring

Having the ability to monitor the status and effectiveness of your web ACL can offer some useful data to help you protect your web application infrastructure, as well as help in incident resolution.

Once your web ACL is configured you have the ability to enable logging which will allow you to store detailed information about all requests managed and handled by your web ACL.



As a prerequisite to configuring this logging feature, you must have Amazon Kinesis Firehose configured that starts with `aws-waf-logs-`. Kinesis Firehose is a service which is used to control and deliver real-time streaming data to storage services. As a part of enabling and configuring logging for your Web ACL you can select a storage service to store your data, for example, Amazon S3.

In addition to using the logging feature which will capture metadata about all requests processed by your Web ACL, it also integrates well with Amazon CloudWatch.

The AWS WAF metrics currently supported by CloudWatch include:

- AllowedRequests
- BlockedRequests
- CountedRequests
- PassedRequests

They are all self-explanatory, the first three simply count the number of requests that have either been allowed, blocked, or counted (relating to the action of the rule within the web ACL). The final metric is another sum of the number of requests that didn't match any rule within the web ACL and therefore took the default action of the web ACL.

AWS Shield

AWS Shield is accessed from within the same dashboard as AWS WAF and again is used to protect your web application infrastructure. AWS Shield is used to protect your infrastructure from **Distributed Denial of Service (DDoS)** attacks. There are different types of DDoS attacks, but the end goal is always the same, to significantly impact the performance and ability of the targeted web servers. This strain on resources prevents any genuine web request from being processed by your infrastructure due to the effort required to process the bad requests. This performance hit can be so severe that your website or application can appear to be offline.

DDoS

There are different types of DDoS attacks which AWS Shield can mitigate against (all are covered by AWS Shield Advanced).

- **User datagram protocol reflection attacks:** This causes the attacked web server to send an increased response traffic using UDP to a spoofed source IP address, with the aim of rendering significant poor performance of the targeted server.
- **SYN flood:** When a TCP connection is made to a server, the server responds with an **acknowledgment (ACK)**, the client then responds again to complete a 3-way handshake. With a SYN flood, the client never sends the final response. As a result, there are connections on the targeted server that remain open and this then drains the available resource to serve legitimate traffic.
- **DNS query flood:** These attacks are caused by users sending numerous DNS queries against a DNS server to deplete it's resources, so within AWS, this would be Route 53 (AWS DNS service).
- **HTTP flood/cache-busting attacks:** HTTP floods are multiple HTTP requests sent to a server which include GET and POST commands. Cache-busting is similar to that of HTTP floods, however, they force the requested content to come from an origin server instead of a cached edge location. Again, as with all other types of request this is within the intention of draining and causing a performance impact on the source resources.

Shield plans

By default when using AWS WAF you will automatically get a basic level of protection (AWS Shield Standard) against these DDoS attacks at layer 3 and 4 of the OSI model, there is, however, an additional level of protection offered for an additional price, this is known as **AWS Shield Advanced**, which also protects at layer 7. The following table taken from the AWS Management Console when viewing Shield shows the differences between the two levels of AWS Shield plans offered:

Features	AWS Shield Standard	AWS Shield Advanced
Active monitoring		
Network flow monitoring	✓	✓
Automated application (layer 7) traffic monitoring	-	✓
DDoS mitigations		
Helps protect from common DDoS attacks, such as SYN floods and UDP reflection attacks	✓	✓
Access to additional DDoS mitigation capacity	-	✓
Visibility and reporting		
Layer 3/4 attack notification and attack forensic reports	-	✓
Layer 3/4/7 attack historical report	-	✓
DDoS response team support		
Incident management during high severity events	-	✓
Custom mitigations during attacks	-	✓
Post-attack analysis	-	✓
Cost protection		
Reimburse related Route 53, CloudFront, and ELB DDoS charges	-	✓
Status	Activated	Not activated
Price	No additional cost for all AWS customers	\$3,000/month plus additional data transfer fees AWS WAF included at no additional cost Learn more
Activate AWS Shield Advanced		

If your organization has a heavy emphasis on web infrastructure, then it could be well worth the additional cost of \$3000 a month to help protect yours against such attacks which could devastate your environment if a DDoS attack was successful.

AWS Firewall Manager

The AWS Firewall Manager is an extremely useful tool if you are using AWS Organizations and multiple AWS accounts. In fact, without configuring AWS Organizations you are unable to use the feature at all.

AWS Firewall Manager has been designed to help you manage, control and implement your AWS WAF rules across the whole AWS Organization with ease and simplicity. Once you have set up your AWS WAF rules using the conditions that you define, they can then be deployed and used within other AWS accounts within your organization without having to recreate them in each account. Any new resources that are created that need to be protected will automatically be safeguarded via the AWS Firewall Manager. For example, you might create additional CloudFront distributions in your existing accounts, or even if a new account is created within your organization with a new CF distribution, AWS Firewall Manager will use policies to ensure the resources are protected.

From a high-level perspective of how the rules are deployed across accounts, your rules are created within AWS WAF, these are then placed into a group. These groups are then referenced within a policy using Firewall Manager. This allows the AWS Firewall Manager to apply these policies to different resource types, for example, ALB or CloudFront distributions.

Before using AWS Firewall Manager

Before using the AWS Firewall Manager you need to fulfill a number of steps. As previously mentioned you need to be using AWS Organizations. The account must either join an existing AWS Organization or create a new one. Do bear in mind that the AWS Organization must have all features enabled, not just consolidated billing.

Next, you must designate one of your AWS account to be the Firewall Manager administrator account.

Set administrator

To use AWS Firewall Manager, you need to set an account in your AWS Organization as the AWS Firewall Manager administrator account. The administrator account will be able to create and manage AWS WAF rules across all accounts within the organization. This account can be either the AWS Organizations master account, or a member account in the organization.

Current administrator for AWS Firewall Manager Not set

AWS Account ID for administrator

This account will be able to set security policy across your organization

[Cancel](#) [Set administrator](#)

Finally, each member account within your AWS Organization that wants to use the features created by the AWS Firewall Manager administrator needs to enable AWS Config within each region that contains resources that need to be protected.

Once you have configured all prerequisite requirements of AWS Firewall Manager, you can then begin to create your groups of rules and policies to enable you to deploy web ACLs across multiple resources in multiple accounts from a single location.

Amazon CloudFront security features

When discussing AWS WAF earlier in this chapter I explained that web ACLs can be associated to CloudFront distributions to help protect your web applications from malicious activity. In addition to AWS WAF, Amazon CloudFront has a host of other security features that could also be used to help prevent harmful activity across your infrastructure. Here are some of the common features used:

- **Geo-restriction:** When using geo restriction within CloudFront it enables you to restrict or permit traffic based on the source IP address. For compliance reasons you may need to restrict requests from specific countries, using the geo restriction features you can control who can and can't access your resources on a per CloudFront distribution basis.
- **Origin access identity:** If you are serving your content via an S3 bucket then it will typically have read access to the public to allow anyone to retrieve the content. This means that if someone knows the URL of object(s) then they can access that content directly, bypassing CloudFront and any restrictions that you may have in place. As a result, you can restrict access to the entire bucket by limiting access to a single user, an origin access identity. This identity is a particular CloudFront user which is associated with your distribution as the sole identity who can retrieve content. This ensures that content can ONLY be services via CloudFront.
- **HTTPS connectivity:** Should your web applications require requests to be encrypted then CloudFront can be configured to use HTTPS for requests. If content is requested by an end user then CloudFront at the Edge location will encrypt the content using SSL/TLS before sending it to the end user. If the content is not in cache, then the requests will be encrypted and sent to the origin. The content will then be encrypted at the origin and sent back to the edge location, where it is then encrypted and stored, before being encrypted again and sent back to the requester.

- **Signed URLs and cookies:** This allows you to restrict access to your content and resources to very selective users, perhaps only to those who have subscribed to your services. Signed URLs can be generated manually or by your web application, and will generally include an expiry time frame to which that URL can be used to access content. Signed cookies restrict access based on multiple files, such as all pages available to your subscribed users. Both Signed URLs and signed cookies involve a level of encryption using public/private key pairs.

Summary

Security threats to web application and infrastructure have and always will be a concern for the organization who provide public-facing services. Unfortunately, you have to prepare for those who are determined to breach your security defenses and take advantage of any weaknesses that may exist within your infrastructure. AWS has a number of Security services that help you to control and manage these inevitable attacks, in this chapter we discussed AWS WAF, AWS Shield, AWS Firewall Manager, and some of the CloudFront security features. Although some of these services come at an additional cost, this cost is negligible when it comes to comparing it against the cost of compromised data and resources within your business. Not just a financial cost, but reputational cost which can unintentionally adversely affect your business going forward.

Further reading

- **Secure Content Delivery with Amazon CloudFront:** https://d1.awsstatic.com/whitepapers/Security/Secure_content_delivery_with_CloudFront_whitepaper.pdf
- **Using WAF to mitigate OWASP's to 10 Web Application Vulnerabilities:** <https://d1.awsstatic.com/whitepapers/Security/aws-waf-owasp.pdf>

18

Cost Effective Resources

Cost efficiency is one important aspect when designing solutions in the cloud, a non-cost effective architecture can compromise full projects by making them unviable. It is really easy to lose control of expenses when running multiple environments with numerous teams. There are many strategies to keep costs at the minimum and identify non-cost compliant resources; AWS provides several services that working together can make the best to keep budgets under control. We will discuss the well-architected principles and best practices for costs optimization.

Pooled resources and serverless services reduce your costs significantly by removing the necessity to maintain and operate compute infrastructure. Managed services are cost-effective because they improve the **Return on Investment (ROI)** and minimize the **total cost of ownership (CTO)** for any kind of technology.

Cloud computing is a great environment for the economies of scale by which customers pay less for what they use because of resource sharing and pooling. Cloud economics also apply for volume discounts and future price drops by service.

The following topics will be covered in this chapter:

- Reserved Instances
- Billing and cost management
- AWS Organizations

Technical requirements

IAM users will require special permissions in order to interact with the billing and cost management console, for additional information to grant users access please navigate to the following link <https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/grantaccess.html>.

Reserved Instances

Reserved Instances (RI) have two objectives: the ability to reserve capacity for when you need it. Imagine a stakeholder event where you want to make a demo with a specific instance type, and to create savings in your account.

Reserved Instances work by purchasing instances of specific types. When those reserved instances match attributes for your currently running on-demand instances, they are billed with the RI model.

Standard Reserved Instances

It is possible to specify the term of how long you will reserve the instance. This can be chosen for 1 or 3 years, the latter being the most economic term. Besides the term, other factors like instance type, size, and availability zone are also used to price the discount. Customers have the option to pay using three options:

- **All Upfront** (best price)
- **Partial Upfront** (the discount is proportional to the partial payments)
- **No Upfront** (pay on a per month basis)

To buy RIs navigate to EC2 | **Reserved Instances** | **Purchase Reserved Instances** as shown in the following screenshot:

Purchase Reserved Instances X

Only show offerings that reserve capacity

Platform	Linux/UNIX	Tenancy	Default	Offering Class	Any	Payment Option	All Upfront	Search
Instance Type	m5.xlarge	Term	Any	Offering Class	Any	Quantity Available	Desired Quantity	Normalized units per hour
AWS	36 months	\$0.088	\$2,322.00	\$0.000	All Upfront	convertible	Unlimited	1 <input type="text"/> 8 Add to Cart
AWS	36 months	\$0.074	\$1,937.00	\$0.000	All Upfront	standard	Unlimited	1 <input type="text"/> 8 Add to Cart
AWS	12 months	\$0.132	\$1,153.00	\$0.000	All Upfront	convertible	Unlimited	1 <input type="text"/> 8 Add to Cart
AWS	12 months	\$0.114	\$1,003.00	\$0.000	All Upfront	standard	Unlimited	1 <input type="text"/> 8 Add to Cart

You currently have no items in your cart. Cancel View Cart

In the previous reservation, an m5.xlarge is selected with the **All Upfront** payment option. This search has found four instance options in the N. Virginia region with terms from 12 to 36 months. As you can see in the **Offering Class** some instances are offered as convertible.

Convertible Reserved Instances

Once you reserve an instance for a fixed term, you may want to stop using it or to exchange your instance for bigger or smaller ones. Assuming the conversion is made with the **same instance type** and this is done **in the same AZ**, a conversion factor can be applied using a normalization factor, take the following example:

One m1.xlarge can be changed for two m1.large, four m1.medium, and eight m1.small.



For more information about normalization factors and how RI's are applied use the following link: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/apply_ri.html.

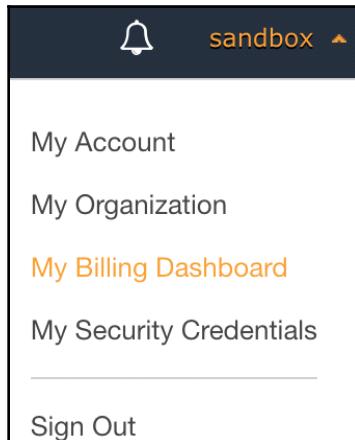
Reserved Instances can also be resold in the AWS Marketplace so current reservations can be transferred to new owners and customers can find instances with shorter terms.

Billing and cost management

The **Billing & Cost Management Dashboard** provides a place to manage the economics of your account. Here you can visualize your costs grouped by several dimensions as service, region, or a specific tag. You can find trends and patterns in usage, you have the ability to configure budgets to keep your expenses under control when the forecasts or limits have been exceeded and receive Reserved Instances recommendations based on use.

One of the key advantages of cloud computing is the trade of capita expense for the variable expense by paying only what you use and when you use it with **No Upfront** investments. However in some cases making upfront investments can lower your costs up to 75%.

To navigate to the billing dashboard you must have explicit permissions or be logged as root. Navigate to your account data, as shown in the following screenshot:



The first step is to update the billing preferences and configure billing alarms. Under the **Preferences** section (**Receive Billing Alerts**) use the following link to navigate to CloudWatch alarms easily.

The screenshot shows the AWS Billing Preferences page. On the left, there's a sidebar with links like Dashboard, Bills, Cost Explorer, Budgets, Reports, Cost Allocation Tags, Payment Methods, Payment History, Consolidated Billing, Preferences (which is selected), Credits, and Tax Settings. The main content area is titled 'Preferences' and has sections for 'Billing Preferences' and 'Cost Management Preferences'. Under 'Billing Preferences', there are three checked checkboxes: 'Receive PDF Invoice By Email', 'Receive Free Tier Usage Alerts', and 'Receive Billing Alerts'. The 'Receive Billing Alerts' section includes a note about monitoring AWS usage charges and recurring fees, and a note about receiving ongoing reports daily. It also shows a dropdown for 'Save to S3 Bucket' set to 'billing-bucket-gabanox', a 'Verify' button, and a green checkmark indicating a valid bucket. Below this is a note about applying permissions to the S3 bucket. A table allows configuring report granularity for different types of reports. At the bottom, there's a note about needing EC2 usage tags for certain reports, a 'Manage report tags' link, and a 'Save preferences' button.

CloudWatch aggregates data at the service level and cost management is not exception, here you can visualize trends in usage and create billing alarms that will notify you before you go out of budget.

Billing alarms

Billing alarms are a great way to receive email notifications when your spending threshold has been crossed, by doing so you won't exceed your spending and have the option to take appropriate actions.

The screenshot shows the AWS CloudWatch Metrics Summary interface. On the left sidebar, under the 'CloudWatch' section, 'Metrics' is selected. In the main content area, there's a 'Metric Summary' section with a message about monitoring operational and performance metrics. Below it is an 'Alarm Summary' section which states 'You do not have any alarms created in the US East (N. Virginia) region.' A callout highlights the 'Create a billing alarm' link, which is described as receiving email alerts when AWS charges exceed a chosen threshold. To the right, there's an 'Additional Info' sidebar with links like 'Getting Started Guide' and 'Create Alarm' at the bottom. At the bottom of the main content, there's a 'Service Health' section showing the 'Amazon CloudWatch Service' is operating normally.

This alarm will aggregate charges at the account level for this specific region. In the following example, I have configured a limit of 100 USD and an existing topic to receive emails. You could also use the new list to specify an email automatically if no previous topic was created. Next click on **Create Alarm** and you're set.

Create Alarm

Billing alarm

You can create a billing alarm to receive e-mail alerts when your AWS charges exceed a threshold you choose. Simply:

1. Enter a spending threshold
2. Provide an email address
3. Check your inbox for a confirmation email and click the link provided

When my total AWS charges for the month

exceed: USD

send a notification to: New list
(gabanox@gmail.com)

Reminder: for each address you add, you will receive an email from AWS with the subject "AWS Notification - Subscription Confirmation". Click the link provided in the message to confirm that AWS may deliver alerts to that address.

Alarm Preview

This alarm will trigger when the blue line goes above the red line

EstimatedCharges > 100 for 1 datapoints within 6 ho...

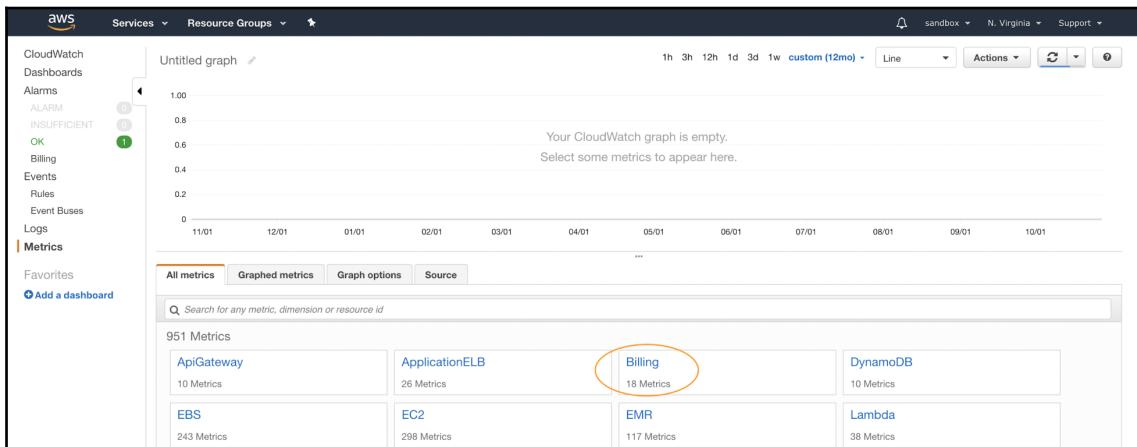
125
100
75
50
25
0
10/21 10/23 10/25
19:00 19:00 19:00

More resources

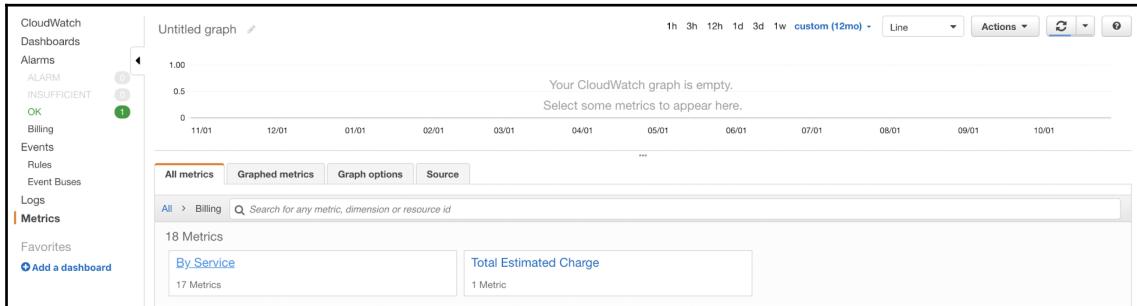
[AWS Billing console](#)
[Getting started with billing alarms](#)
[More help with billing alarms](#)
[AWS Billing FAQs](#)

Service level alarms

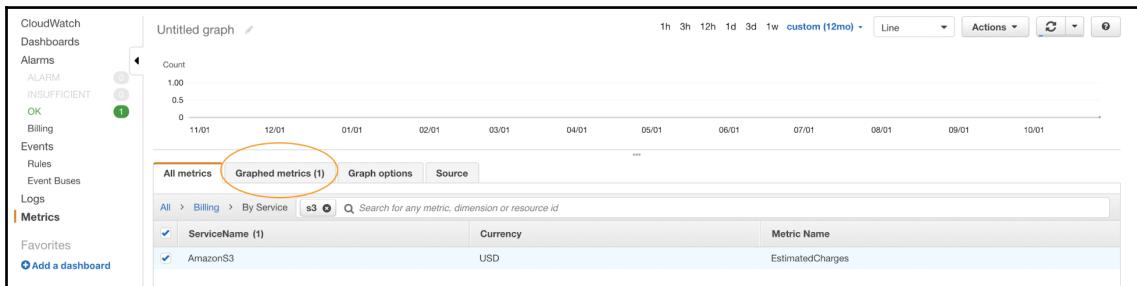
Using the CloudWatch console, find the **Metrics** sub-menu. Here we have the **Billing** category we will be using it to specify billing alarms per service:



And then use the **By Service** metrics, these metrics are automatically aggregated under service namespaces by CloudWatch.



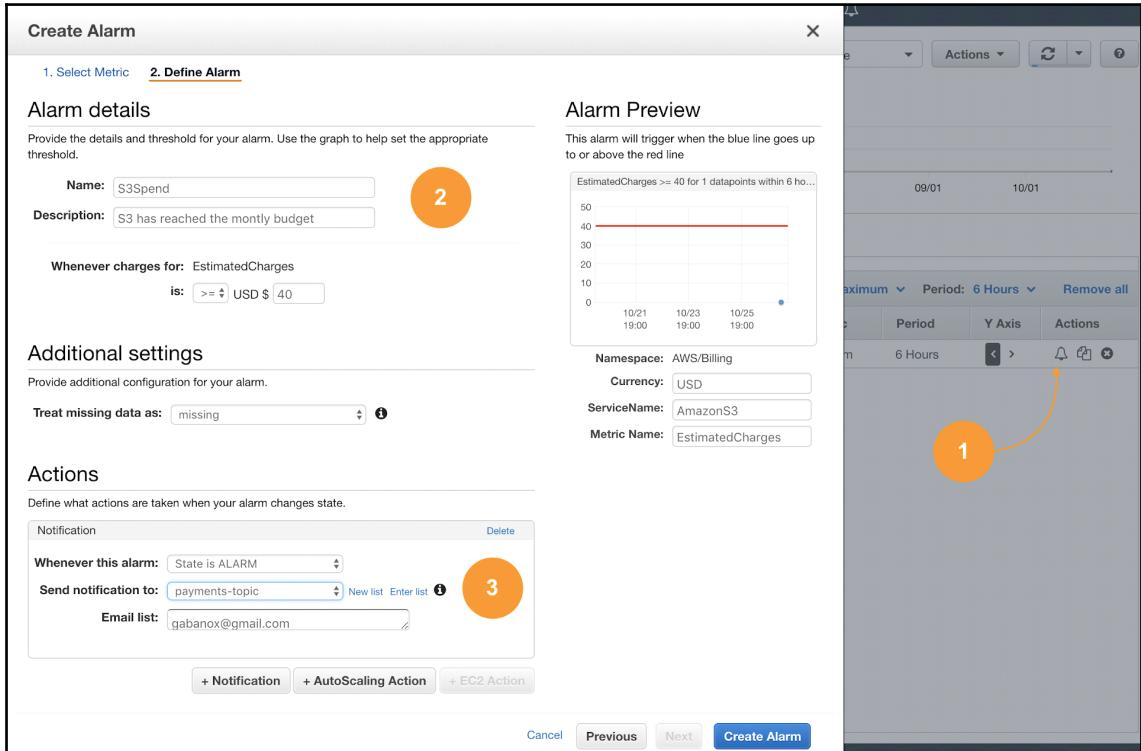
Here you will see a list of every metered service, use the filter to search for s3.



The **Graphed metrics** tab provides with configuration parameters like statistics and actions like the creation of alarms. Use the bell icon to perform the alarm configuration.



In the **Create Alarm** window, specify the name and description for the alarm (2). And the SNS topic to be used for notifications, update the EstimatedCharges value for any number that makes sense to you. Now you are ready to create the alarm for S3 spending.



All Cloudwatch alarms can be in one of the following statuses:

- **OK:** The metric is within the defined threshold and has not exceeded the limit expression.
- **ALARM:** The metric is outside of the defined threshold. Actions must be taken.
- **INSUFFICIENT_DATA:** The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state.

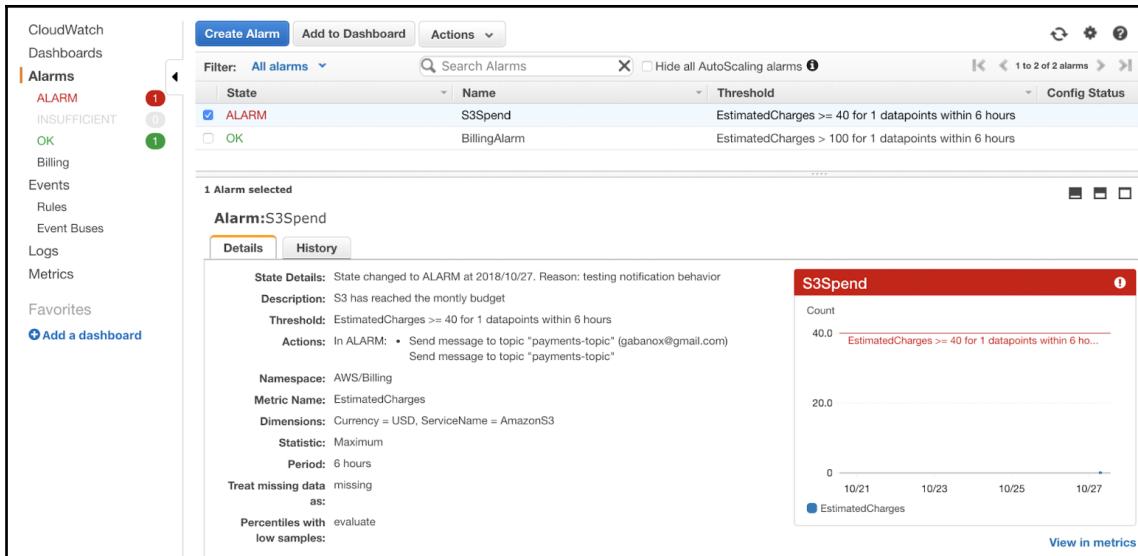
Navigate to the **Alarms** sub-menu and you can see the current alarms created for this region, both alarms are in the **OK** state, so we are within budget.

The screenshot shows the AWS CloudWatch Alarms interface. On the left, there's a sidebar with options like CloudWatch Dashboards, Alarms (selected), ALARM, INSUFFICIENT, OK, Billing, Events, Rules, Event Buses, Logs, Metrics, Favorites, and Add a dashboard. The main area has tabs for Create Alarm, Add to Dashboard, and Actions. A search bar and a filter dropdown set to 'All alarms' are at the top. Below is a table with columns for State, Name, Threshold, and Config Status. Two rows are listed: 'OK' for S3Spend (Threshold: EstimatedCharges >= 40 for 1 datapoints within 6 hours) and 'OK' for BillingAlarm (Threshold: EstimatedCharges > 100 for 1 datapoints within 6 hours). A modal window titled '1 Alarm selected' is open for the 'S3Spend' alarm, showing the Details tab and a History section with two entries: a State update on 2018-10-27 12:51 UTC-5 and a Configuration update on 2018-10-27 12:49 UTC-5.

An interesting feature of CloudWatch alarms is the ability to change their state programmatically. The `set-alarm-state` API action is used for this purpose so let's try it out by alarming the `S3Spend` alarm.

```
aws cloudwatch set-alarm-state --alarm-name S3Spend --state-value ALARM --state-reason "testing notification behavior"
```

The preceding command will trigger the notification in your email inbox and change the alarm status in the console.



This way you can test how alarms are being executed and design robust notification systems maintaining all stakeholders accountable.

Billing reports

Billing reports are recurrent generated usage reports for your account that are delivered once or on a daily basis into an S3 bucket. In the preferences menu of the billing dashboard (**Account menu | My Billing Dashboard**), you can enable the option to receive billing reports as shown in the following screenshot:

1. In order to verify the S3 destination for this reports, you need to create or specify the bucket of your choice and apply the "sample policy" (1).

Receive Billing Reports

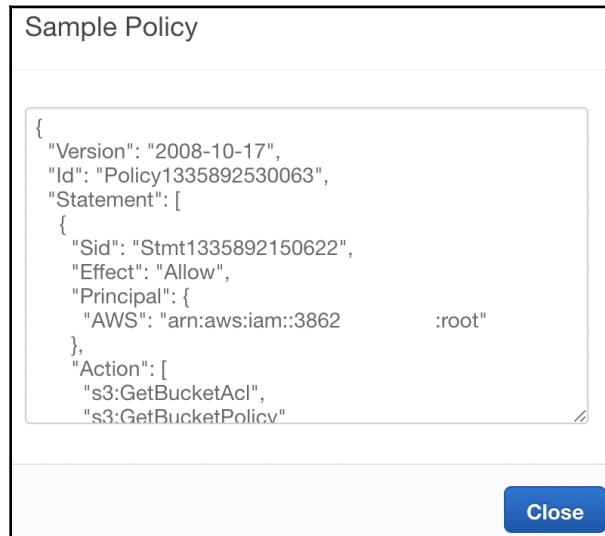
Turn on this feature to receive ongoing reports of your AWS charges once or more daily. AWS delivers these reports to the Amazon S3 bucket that you specify where indicated below. For consolidated billing customers, AWS generates reports only for paying accounts. Linked accounts cannot sign up for billing reports.

Save to S3 Bucket: **Verify** ✓ Valid Bucket **2**

Note: You must apply appropriate permissions to your S3 bucket *sample policy* **1**

You can also configure the granularity of these reports to display your AWS usage. In the table below, select whether you want the reports to display data by the month or hour. Your reports can also display usage by custom tags that you create, or by AWS resource.

2. Click the **sample policy** link to copy and paste the policy in the destination bucket.



3. Navigate to the bucket you want to enable to receive the billing reports, under **Permissions | Bucket Policy** paste the previously copied IAM policy like the following example:

The screenshot shows the AWS S3 Bucket Policy editor for the bucket 'billing-bucket-gabanox'. The policy is defined as follows:

```

1  {
2      "Version": "2008-10-17",
3      "Id": "Policy1335892530063",
4      "Statement": [
5          {
6              "Sid": "Stmt1335892150622",
7              "Effect": "Allow",
8              "Principal": {
9                  "AWS": "arn:aws:iam::00000000:root"
10             },
11             "Action": [
12                 "s3:GetBucketAcl",
13                 "s3:GetBucketPolicy"
14             ],
15             "Resource": "arn:aws:s3:::billing-bucket-gabanox"
16         },
17         {
18             "Sid": "Stmt1335892526596",
19             "Effect": "Allow",
20             "Principal": {
21                 "AWS": "arn:aws:iam::00000000:root"
22             },
23             "Action": "s3:PutObject",
24             "Resource": "arn:aws:s3:::billing-bucket-gabanox/*"
25         }
26     ]
27 }

```

Buttons at the top include Overview, Properties, Permissions, Management, Access Control List, Bucket Policy (which is selected), and CORS configuration. Buttons at the bottom right are Delete, Cancel, and Save.

4. You need to specify the delivery granularity for these reports. In my case I have enabled all the options, you will need to wait at least one hour to receive the first billing report. When the first CSV report appears, download it.

The screenshot shows the AWS S3 bucket contents for 'billing-bucket-gabanox'. The objects listed are:

Name	Last modified	Size	Storage class
8716-aws-billing-csv-2018-10.csv	Oct 27, 2018 3:48:56 PM GMT-0500	60.8 KB	Standard
aws-programmatic-access-test-object	Oct 27, 2018 4:31:15 PM GMT-0500	4.0 B	Standard

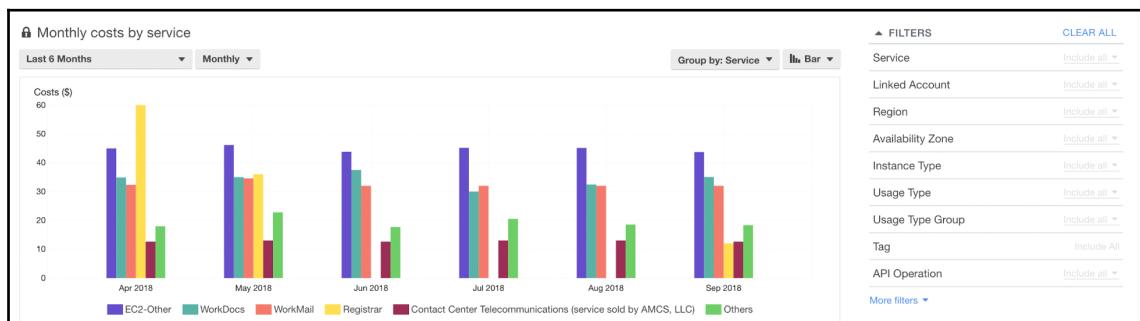
Buttons at the top include Overview, Properties, Permissions, Management, Upload, Create folder, Actions, and US East (N. Virginia). A search bar at the top says 'Type a prefix and press Enter to search. Press ESC to clear.' A message at the bottom right says 'Viewing 1 to 2'.

You can use AWS Lambda with S3 as the origin event to parse this reports as they are generated or access them via the AWS billing API as well.

Cost Explorer

The AWS Cost Explorer is a visual interface that makes it simple to understand usage patterns in your account. You have the ability to filter and group data using predefined filters and tags to analyze spend trends with high granularity.

Cost Explorer provides the ability to create your own custom reports for ease of use and keep up to date with billing usage. AWS provides the cost explorer at no cost and has predefined reports like monthly costs by service.



The right pane provides several filters that can be applied to visualize data grouped by a specific service, region, usage type, API operation, or custom tag providing a complete visibility for customers.

Reserved Instances recommendations

The billing APIs help us learn about the account behavior identifying opportunities to save money. The billing dashboard provides this functionality directly in the cost explorer. Use the hamburger icon at the top left to learn about automatic recommendations based on historical usage data.

Cost Explorer > Reserved Instance Recommendations

Reserved Instance Recommendations

\$793	52%	3
Estimated Annual Savings*	Savings vs. On-Demand	Purchase Recommendations

Based on your past 30 days of EC2 usage, we've identified **3 three-year, all-upfront, standard RI purchase recommendations** to save an estimated **\$793 annually**, representing a savings of **52% versus on-demand costs**. You can take action on these recommendations in the [EC2 RI Purchase Console](#).

Sort by: [Monthly Estimated Savings](#) ▾ [Download CSV](#)

Purchase Recommendations (3)	Details
Buy 20 t2.nano reserved instances <small>Size flexible**</small> US East (N. Virginia) Linux/UNIX Shared Based on your past 30 days of on-demand usage, we recommend purchasing 20 t2.nano reserved instances to cover 5 normalized units per hour of t2 family usage to maximize savings. View Associated EC2 Usage	\$52.83 monthly savings Upfront Cost: \$1,140.00 Recurring Monthly Cost: \$0.00
Buy 5 t2.nano reserved instances <small>Size flexible**</small> US West (N. California) Linux/UNIX Shared Based on your past 30 days of on-demand usage, we recommend purchasing 5 t2.nano reserved instances to cover 1.25 normalized units per hour of t2 family usage to maximize savings. View Associated EC2 Usage	\$9.90 monthly savings Upfront Cost: \$395.00 Recurring Monthly Cost: \$0.00
Buy 1 t2.micro reserved instance US West (N. California) Windows (Amazon VPC) Shared Based on your past 30 days of on-demand usage, we recommend purchasing 1 t2.micro reserved instance. View Associated EC2 Usage	\$3.34 monthly savings Upfront Cost: \$278.00 Recurring Monthly Cost: \$0.00

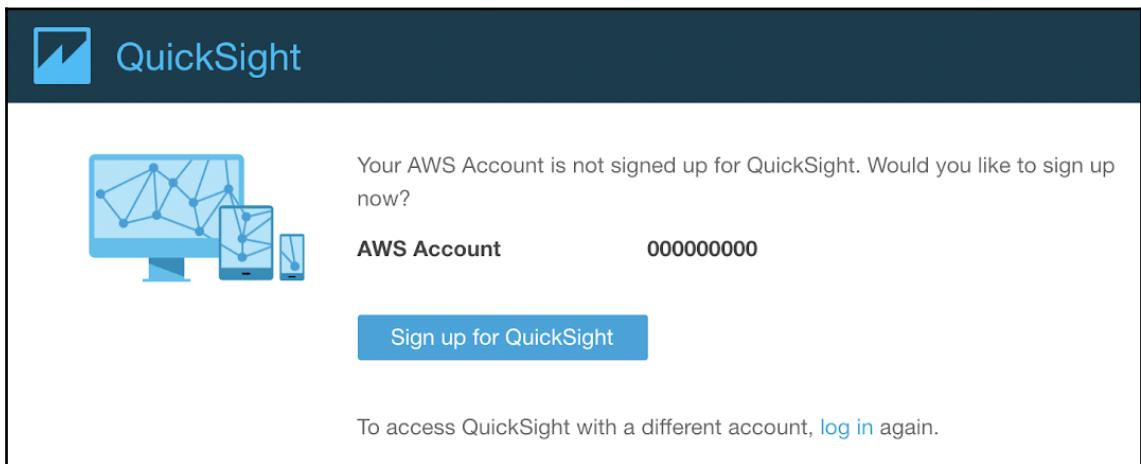
Viewing 1-3 of 3 recommendations

The previous report recommends to reserve EC2 instances for the current region. Choosing the 3 years term and **All Upfront** payment option; savings are calculated up to 793 USD which is a considerable price drop of 52% for the next three years.

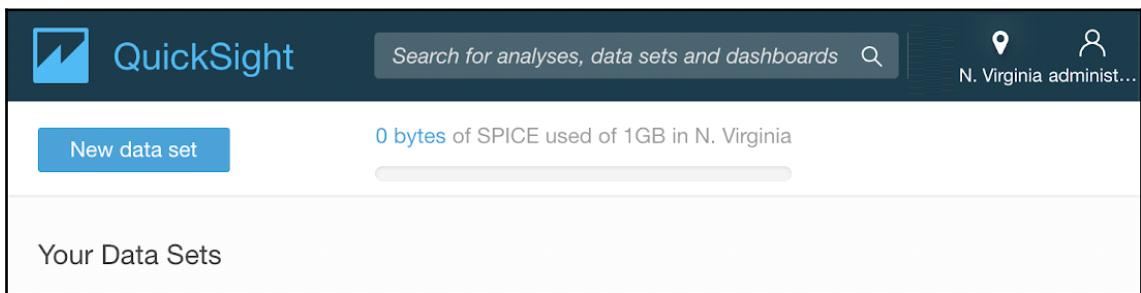
QuickSight visualization

QuickSight is a serverless business intelligence service with a parallel in-memory processing engine (SPICE) for data visualization. We will use it to create a pivot table to understand usage and costs via the billing report generated.

1. Use the console to find QuikSight, if this is the first time you use it you will need to sign up for the service:



2. The next step is to create a new data set:



3. For simplicity, use the option **Upload a file** and load the billing CSV. Confirm the changes for the file schema proposed. Click **Next**:

Confirm file upload settings ×

Settings
csv file, Sheet 0

InvoicelD	PayerAcco...	LinkedAcc...	RecordType	RecordID	Billin
Estimated	8716		PayerLineItem	1000000002...	2018-
Estimated	8716		PayerLineItem	1000000002...	2018-
Estimated	8716		PayerLineItem	1000000002...	2018-
Estimated	8716		PayerLineItem	1000000002...	2018-
Estimated	8716		PayerLineItem	1000000002...	2018-

[Edit settings and prepare data](#) **Next**

4. Click **Visualize** for the data to be visualized:

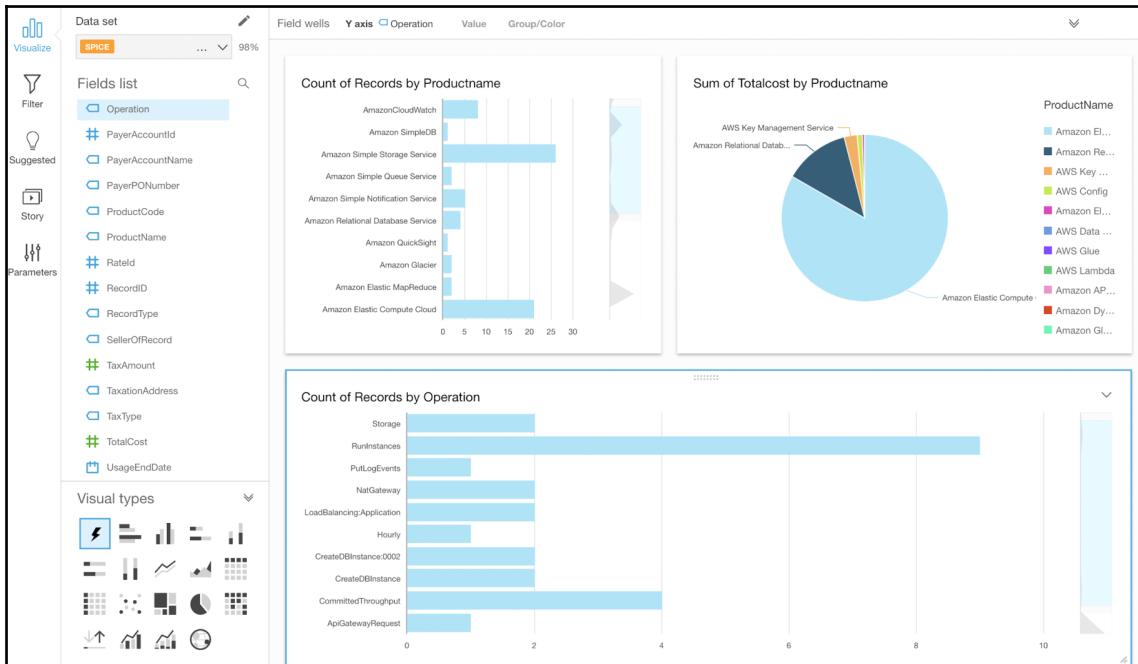
Data source details ×

Table: Sheet 0
Estimated table size: 121.6KB SPICE
Data source: 970129648716-aws-billing-csv-2018-10.csv

Import to SPICE ✓ 11GB available SPICE

[Edit/Preview data](#) **Visualize**

5. Use the **Add (+)** button on the top left part of the screen to incorporate new tiles and play around with the available fields. An example of a custom dashboard is shown in the following screenshot:



Quicksight gives finance users to understand spending with the ability to drill down into data and use different dimensions combinations to present this information to executives and create business reports.

Cost Allocation Tags

Tags are a great tool for classifying resources, in fact, every organization should have policies for the proper classification of assets to enforce data governance. The Cost Allocation Tags are a way to reference resources expenses with custom tags.

The screenshot shows the AWS Cost Allocation Tags interface. On the left, a sidebar lists various cost management options like Dashboard, Bills, Cost Explorer, Budgets, Reports, and Cost Allocation Tags (which is selected and highlighted in orange). The main content area has two sections: 'AWS-Generated Cost Allocation Tags' (disabled) and 'User-Defined Cost Allocation Tags' (enabled). Below this is a note about activating tags for cost allocation. At the bottom, there's a table listing three tags:

Tag key*	Status
Environment	Active
Project	Active

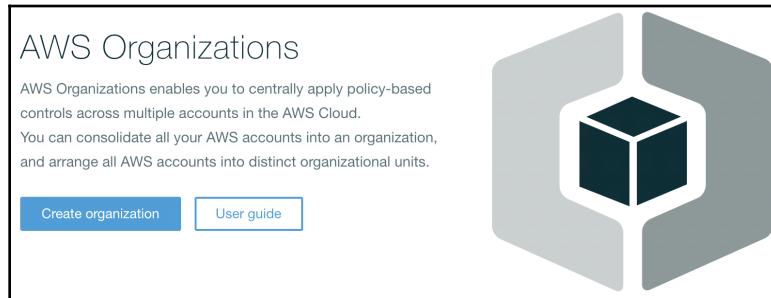
In the previous image, the **Environment** and **Project** tags have been activated to keep record of the expenses generated across all resources tagged with this keys making easier to perform a cost analysis when visualizing the billing report CSV files.

AWS Organizations

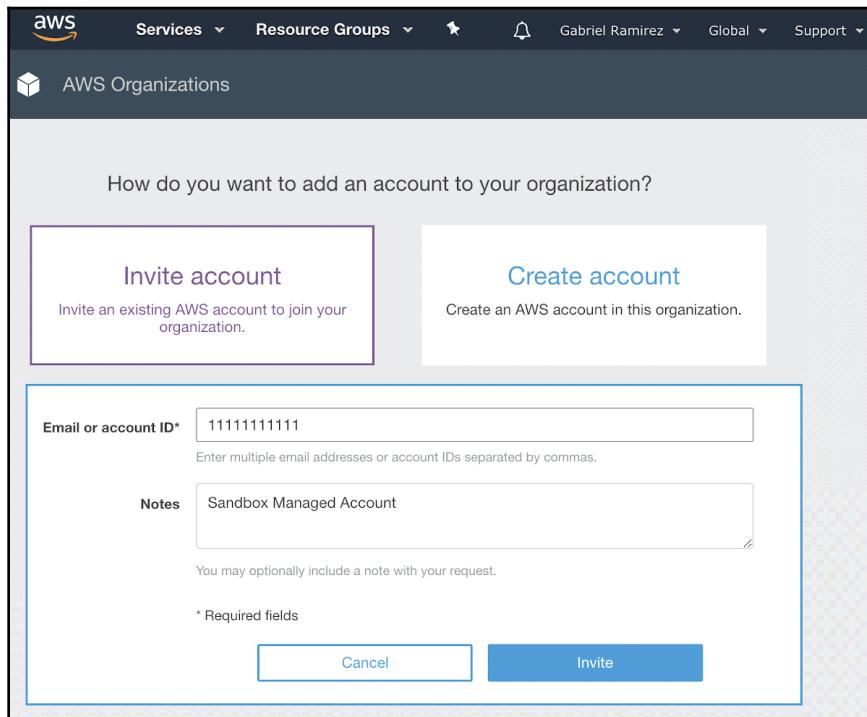
AWS Organizations are a way to centrally manage account hierarchies that resemble organizational structures that make sense from the management, security, and billing perspectives. When you create an organization you have the ability to create **organizational units (OUs)** and apply black or whitelisting IAM policies that override IAM user policies favoring the former.

To centrally manage billing an organization is needed and for this purpose, a master account must be chosen to manage the entire organization of linked accounts.

1. To get started with organizations navigate to the AWS console and search for AWS Organizations.



2. You will be prompted to create a new organization, use this option to get started. The next step is to send invites to linked accounts; you will need the managed account ID.



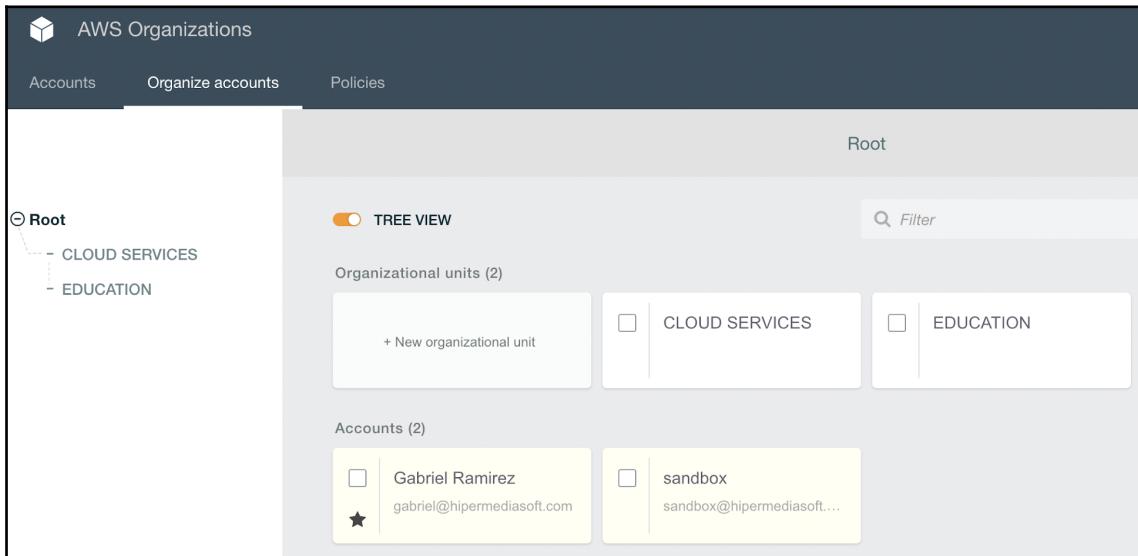
3. The linked account will receive the invitation in the same organization's welcome screen. Click on the invitation link and accept the invitation.

The screenshot shows the AWS Organizations 'Invitations' page. At the top, there is a header bar with the AWS Organizations logo and the title 'AWS Organizations'. Below the header, the main content area has a title 'Invitations' and a sub-instruction: 'You have invitations to join other organizations. Review the details to respond to the invitations. You can only join one organization at a time.' A detailed card displays the invitation information:

Organization ID	o-5h
Master account name	Gabriel Ramirez
Master account email	gabriel@hipermediasoft.com
Requested controls	Enable only consolidated billing The master account pays the charges accrued by all member accounts. Policy-based controls are not available.
Notes	Sandbox Managed Account

At the bottom of the card are two buttons: 'Accept' and 'Decline'.

4. This account now is managed centrally and stops billing at this account level and starts billing at the master account.



You can organize the nodes in a tree structure by using the checkboxes in the accounts section and choosing the move link to associate accounts with organizational units.

Summary

In this chapter we took a deep dive into cost management tools, we started by understanding the benefits and characteristics of Reserved Instances. Then we moved to the AWS console Billing Dashboard to understand the most important capabilities of the service and provided several examples to understand usage like billing alarms, billing reports, the Cost Explorer, and the Cost Allocation Tags.

We ended this module with a practical example of QuickSight for analyzing billing data creating a custom dashboard, and consolidated billing using two different accounts with the AWS Organizations service.

Further reading

- **Consolidated billing process:** https://docs.aws.amazon.com/es_es/awsaccountbilling/latest/aboutv2/useconsolidatedbilling-procedure.html
- **Other Reports:** <https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/billing-reports-other.html>

19

Working with Infrastructure as Code

Infrastructure as Code (IaC) has become a widely adopted practice in the IT industry because it provides systems with a consistent way to describe desired states for different purposes. IaC solves many of the problems encountered with manual operations like low consistency between deployments, stale documentation, and error operations. This practice is done based on two important pillars: **automation** and **APIs**. This is why cloud computing makes a total sense for this activity. AWS provides every service as an API and automates operations by providing several services like CloudFormation.

Treating infrastructure as code makes a total sense from the software engineering perspective. Software is reusable and modular. It can be designed to be loosely coupled and with high cohesion. This chapter focuses on the IaC practice from the CloudFormation service perspective.

Technical requirements

There are no technical requirements for this chapter.

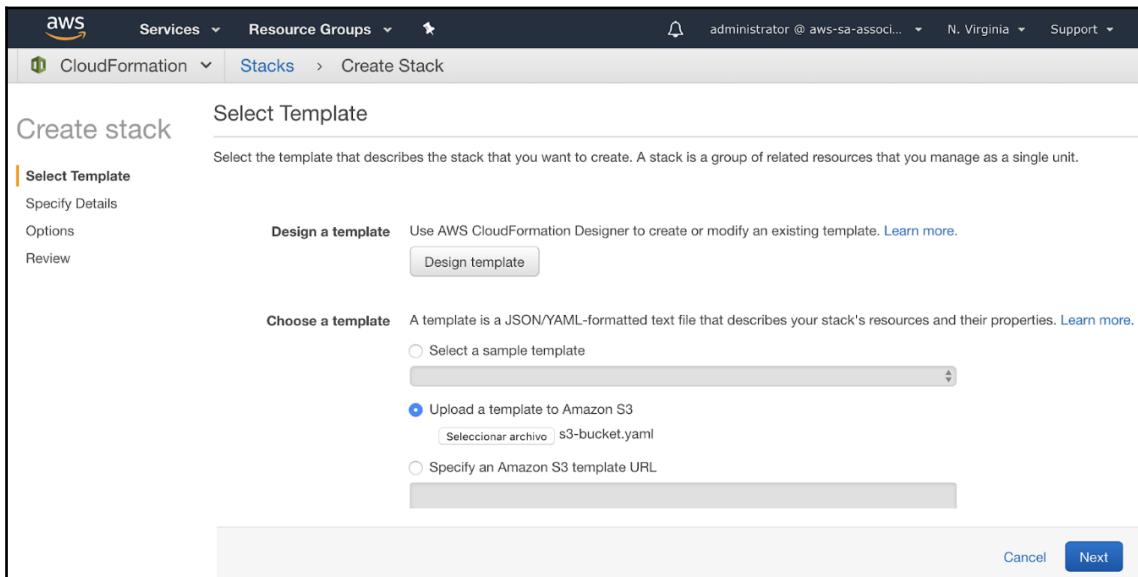
AWS CloudFormation

CloudFormation supports two standard formats, JSON and YAML. Personally I find YAML cleaner. YAML is a full serialization language with robust features and it also supports comments. We will use the YAML format for the following examples. For a quick reference to YAML format use this repo. It is a good idea to use a text editor; let's learn the hard way and use a simple text editor to learn pragmatically by doing lots of templates. **Atom** and **Sublime Text** are good editors. For YAML, the spacing convention is important and these editors will help you to do it easily.

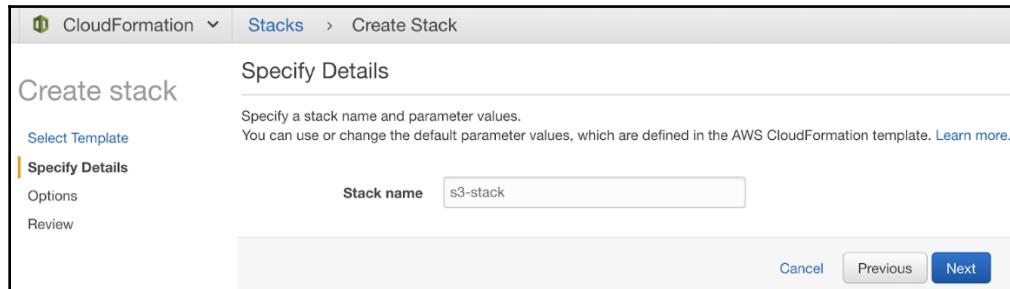
To get started let's create the simplest template possible, `s3-bucket.yaml` (<https://github.com/gabanox/Certified-Solution-Architect-Associate-Guide/blob/master/chapter20/s3-bucket.yaml>):

```
Resources:  
MyS3Bucket:  
  Type: AWS::S3::Bucket
```

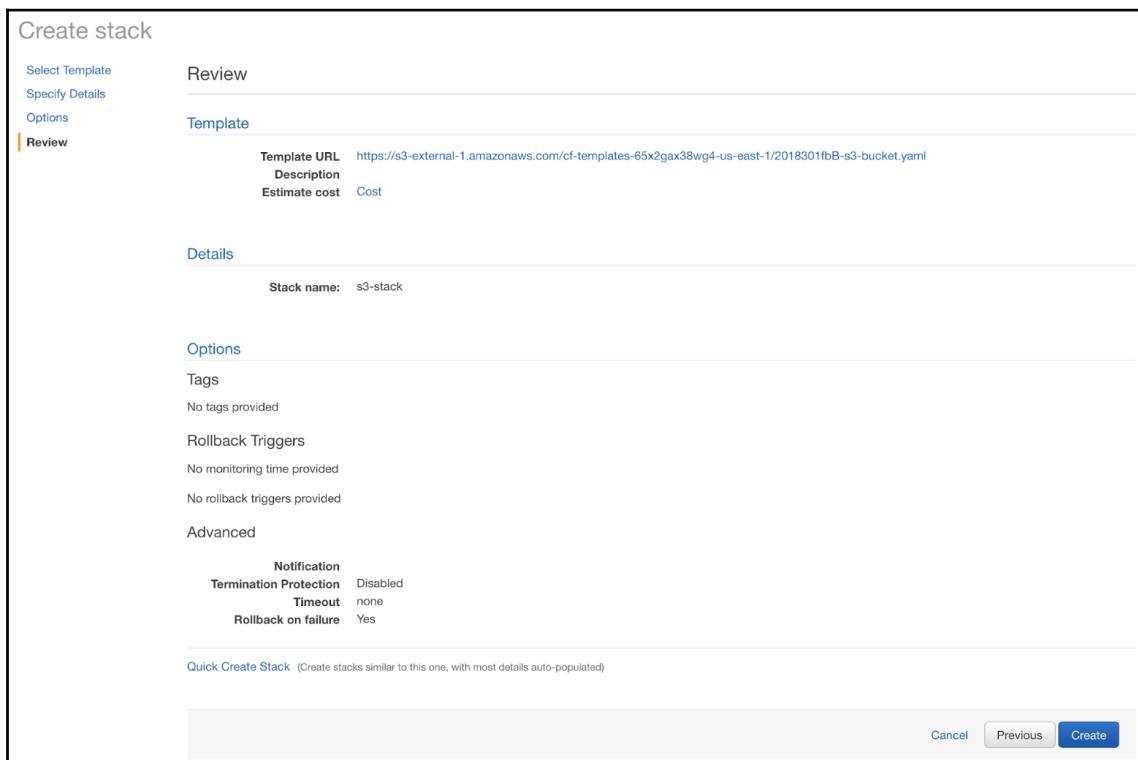
Navigate to **CloudFormation** | **Create Stack** and choose the option to upload template to S3. This will use S3 as a staging area to upload and then create the stack. If templates were previously uploaded to S3, you can reference those templates using their URLs. Click **Next**, as shown in the following screenshot:



In the **Specify Details** screen you can specify input data for the template. The required input data is the **Stack name**, which is a logical identifier that will be used to manage the lifecycle of the provisioned objects, in fact, created objects will have this name concatenated to keep track of object instances. Click **Next**, as shown in the following screenshot:



In the **Options** screen, you can specify additional configurations like triggers for rollbacks, SNS notification topics, and IAM permissions. Leave the **Options** screen as is and click **Next**, you will be presented with a Review screen, as shown in the following screenshot:



Click on **Create** and the stack will be provisioned automatically. **Stacks** are the logical grouping of resources provisioned or updated by a single template and they are managed as a unit.

The template falls in the **CREATE_IN_PROGRESS** status, poll until the template changes the status to **CREATE_COMPLETE**. Use the **Events** tab, shown in the following screenshot, to understand the steps taken to provision the S3 bucket:

Created Time	Status	Type	Logical ID	Status Reason
2018-10-28 11:42:26 UTC-0500	CREATE_COMPLETE	AWS::CloudFormation::Stack	s3-bucket-stack	
2018-10-28 11:42:51 UTC-0500	CREATE_COMPLETE	AWS::S3::Bucket	MyS3Bucket	
2018-10-28 11:42:30 UTC-0500	CREATE_IN_PROGRESS	AWS::S3::Bucket	MyS3Bucket	Resource creation initiated
2018-10-28 11:42:30 UTC-0500	CREATE_IN_PROGRESS	AWS::S3::Bucket	MyS3Bucket	
2018-10-28 11:42:26 UTC-0500	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	s3-bucket-stack	User Initiated

Now that the resource has been provisioned without errors, validate the resource created in the S3 console. Pretty fun huh?

Bucket name	Access	Region	Date created
s3-bucket-stack-mys3bucket-163gba2begj9e	Not public *	US East (N. Virginia)	Oct 28, 2018 11:42:31 AM GMT-0500

Because this resource is managed via the stack the proper way to delete it is directly from CloudFormation to we don't leave orphaned resources. This auto assigned resource ID is a good way to work using infrastructure as code by applying the concept to *cattle not pet* (<http://cloudscaling.com/blog/cloud-computing/the-history-of-pets-vs-cattle/>). If you assign names to resources you get attached to them, but if you treat them like cattle you assign random numbers like r-01, r-02, and you manage them collectively. With the template checked use the actions button and choose delete. After this operation has completed you shouldn't see the S3 bucket anymore. This operation will not cascade the deletion if the bucket contains objects.

Template anatomy

The only required section of the template is the Resources section:

```
---
AWSTemplateFormatVersion: "version date"

Description:
  String

Metadata:
  template metadata

Parameters:
  set of parameters

Mappings:
  set of mappings

Conditions:
  set of conditions

Transform:
  set of transforms

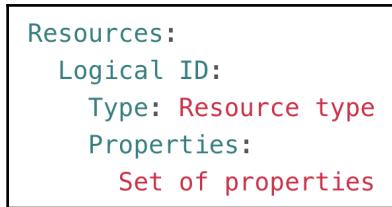
Resources:
  set of resources

Outputs:
  set of outputs
```

Resources

The logical ID serves as the logical name that can be referenced from within the template or from external templates. The type is one of the canonical resources defined by AWS or they can be custom resources; CloudFormation is designed to be an extensible language.

For more information about the resources available, visit <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html>:



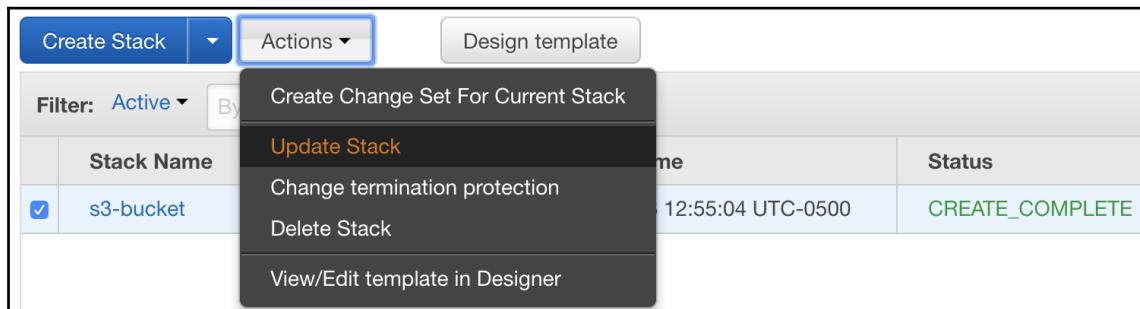
The Properties are values that can be used to write specific behavior to infrastructure objects. For example, the VersioningConfiguration.

Stack updates

Stack updates are a way to analyze the impact of changing infrastructure attributes and parameters. Updates can result in one of the following behaviors:

- Update: modify
- Update: replace
- Update: delete

Create the s3-bucket stack again. Here, the naming convention is fundamental. To execute the stack update, create another YAML file called s3-bucket-versioned.yaml (<https://github.com/gabanox/Certified-Solution-Architect-Associate-Guide/blob/master/chapter20/s3-bucket-versioned.yaml>). Select the s3-bucket stack and under the **Actions** drop-down, use the **Update Stack** menu option. You will be taken to the wizard. Go through every step with the defaults:



The last screen shows a preview of the changes we made and impact that this update will trigger:

Preview your changes				
Based on your input, CloudFormation will change the following resources. For more information, choose View change set details .				
Action	Logical ID	Physical ID	Resource type	Replacement
Modify	MyS3Bucket	s3-bucket-mys3bucket-mah5frsudrv	AWS::S3::Bucket	False

Here, the replacement column states that it will keep the actual resource with a modification of its properties. After the change has been executed, validate the changes in the s3-bucket:

Amazon S3 > s3-bucket-mys3bucket-mah5frsudrv

Overview Properties Permissions Management

Versioning
Keep multiple versions of an object in the same bucket.
[Learn more](#)
 Enabled

Server access logging
Set up access log records that provide details about access requests.
[Learn more](#)
 Disabled

Static website hosting
Host a static website, which does not require server-side technologies.
[Learn more](#)
 Disabled

Object-level logging
Record object-level API activity using the CloudTrail data events feature (additional cost).
[Learn more](#)
 Enabled

Stack updates is a great analysis and change management tool to keep momentum with your stack resources. Be aware that some resources will cause to replace the current object with a fresh one. In some cases you will need to override this behavior by specifying a deletion policy.

Deletion policy

A **deletion policy** is a directive that overrides the behavior when deleting stacks. For example, to create a DB snapshot after terminating the RDS instance or retaining an S3 bucket as shown in the following example.

```
Resources:  
  MyS3Bucket:  
    Type: AWS::S3::Bucket  
    Properties:  
      VersioningConfiguration:  
        Status: Enabled  
      DeletionPolicy: Retain
```

The available values are:

- Delete: This is the default behavior for every object unless other policy is specified
- Retain: This ignores the deletion request of the stack, is possible to orphan this resource by detaching from the main stack
- Snapshot: Several compute resources provide this capability to perform an EBS snapshot before the tier down, examples of this are AWS::EC2::Volume and AWS::ElasticCache::CacheCluster

Outputs

Several resources provide read only functionality to retrieve value properties to be used in another context. For example, consider the following AWS::EC2::EIPAssociation allocation:

```
Resources:  
  MyEIP:  
    Type: AWS::EC2::EIP  
  
Outputs:  
  MyEIPIPv4Address:  
    Description: Elastic IP Address  
    Value : !Ref MyEIP
```

The Outputs section specifies a logical output resource called `MyEIPIPv4Address` that is used to read the result value of the `MyEIP` resource. The resulting output is as follows:

Key	Value	Description	Export Name
MyEIPIPv4Address	52.205.88.117	Elastic IP Adress	

Outputs are a great way to maintain high cohesion with your templates by referencing values internally in the template and cross template. They are also used to overcome template size limits when working with complex templates.

Template reusability

Software is flexible, meaning that it can be adapted to changes and evolve. For this example, we will use a network-tier template and a compute-tier template by breaking these infrastructure responsibilities to be managed independently.

Analyze the following template `ec2-instance.yaml`:

```
Resources:  
  Ec2Instance:  
    Type: AWS::EC2::Instance  
    Properties:  
      ImageId: ami-0922553b7b0369273  
      KeyName: MyEC2KeyPair  
      NetworkInterfaces:  
        - AssociatePublicIpAddress: true  
          DeviceIndex: 0  
          GroupSet:  
            - sg-0abe65d366868b774  
          SubnetId: subnet-91a8b9f5
```

This code is brittle, what will happen when using a different subnet or the EC2 key pair is wrong spelled? The code will break. In software engineering, we use the open/closed design principle which means that software entities should be *open for extension, but closed for modification*. Bringing this concept to this template will require it to be opened in the text editor and modified to update the EC2 key pair, the Subnet ID, and security group ID. Also, AMIs are regional resources meaning that this template is not portable but we can improve on this. CloudFormation works with sensible defaults, for example, we didn't specified an instance type so the stack provisioned an `m1.small`.

Parameters

Parameters are a way to provide dynamic variables that can vary from deployment to deployment, this way applications and end users can specify input data that will be replaced in the template at runtime.

Specify Details

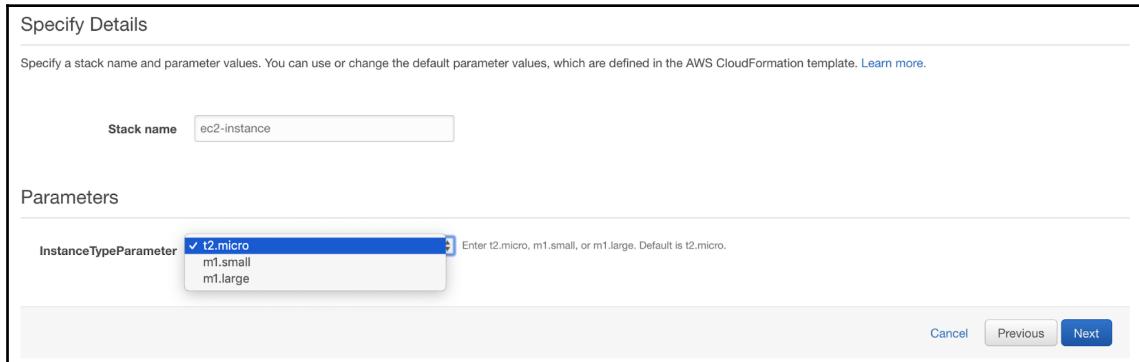
Specify a stack name and parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. [Learn more](#).

Stack name

Parameters

InstanceTypeParameter t2.micro m1.small m1.large Enter t2.micro, m1.small, or m1.large. Default is t2.micro.

Cancel Previous Next



The use of parameters will promote maintainability and reusability for your stacks. We make use of the `Ref` intrinsic function to reference the named declaration:

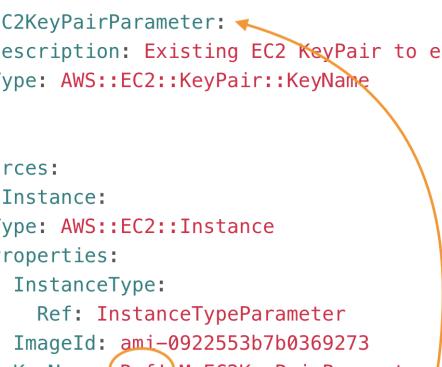
```
Parameters:  
  InstanceTypeParameter:  
    Type: String  
    Default: t2.micro  
    AllowedValues:  
      - t2.micro  
      - m1.small  
      - m1.large  
    Description: 'Enter t2.micro, m1.small, or m1.large. Default is t2.micro.'  
  
Resources:  
  Ec2Instance:  
    Type: AWS::EC2::Instance  
    Properties:  
      InstanceType:  
        Ref: InstanceTypeParameter  
      ImageId: ami-0922553b7b0369273  
      KeyName: MyEC2KeyPair  
      NetworkInterfaces:  
        - AssociatePublicIpAddress: true  
        DeviceIndex: 0  
        GroupSet:  
          - sg-0abe65d366868b774  
        SubnetId: subnet-91a8b9f5
```

Let's do the same for the EC2 KeyName:

```
Parameters:
  InstanceTypeParameter:
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - m1.small
      - m1.large
    Description: Enter t2.micro, m1.small, or m1.large. Default is t2.micro.

  MyEC2KeyPairParameter:
    Description: Existing EC2 KeyPair to enable SSH access to the instance
    Type: AWS::EC2::KeyPair::KeyName

Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType:
        Ref: InstanceTypeParameter
      ImageId: ami-0922553b7b0369273
      KeyName: Ref! MyEC2KeyPairParameter
      NetworkInterfaces:
        - AssociatePublicIpAddress: true
        DeviceIndex: 0
        GroupSet:
          - sg-0abe65d366868b774
        SubnetId: subnet-91a8b9f5
```



Note the usage of the inline intrinsic function `Ref!` function on `KeyName`.

What happens when multi-valued parameters need to be used? Lists are a way to reference 1:M values and managed through indexes. For our example, we want to reference multiple security groups:

```
MySecurityGroupParameter:
  Type: List<AWS::EC2::SecurityGroup::GroupName>
```

This way, CloudFormation will populate a list of the available security groups by name with multiple selection for this case:

```
Parameters:  
  InstanceTypeParameter:  
    Type: String  
    Default: t2.micro  
    AllowedValues:  
      - t2.micro  
      - m1.small  
      - m1.large  
    Description: Enter t2.micro, m1.small, or m1.large. Default is t2.micro.  
  
  MyEC2KeyPairParameter:  
    Description: Existing EC2 KeyPair to enable SSH access to the instance  
    Type: AWS::EC2::KeyPair::KeyName  
  
  MySecurityGroupParameter:  
    Type: List<AWS::EC2::SecurityGroup::GroupName>  
  
Resources:  
  Ec2Instance:  
    Type: AWS::EC2::Instance  
    Properties:  
      InstanceType:  
        Ref: InstanceTypeParameter  
      ImageId: ami-0922553b7b0369273  
      KeyName: Ref! MyEC2KeyPairParameter  
      SecurityGroups:  
        - Ref: MySecurityGroupParameter
```



Lists in YAML format are declared using the hyphen sign (-). In the UI, end users will be presented with a multi selection list for the existing security groups in this region and account.

The screenshot shows the 'Specify Details' step of a CloudFormation stack creation. It includes fields for the stack name (set to 'ec2-instance'), parameters (InstanceTypeParameter set to 'm1.small', MyEC2KeyPairParameter set to 'Docker', and MySecurityGroupParameter set to 'docker-admin x, http x, ssh x'), and a bottom navigation bar with 'Cancel', 'Previous', and 'Next' buttons.

We left the AMI selection to the end of this example. AMIs are examples of multi-selection logic because AMIs are dependent of the AWS region, but AMIs also have other complex search attributes like the architecture (32 or 64 bits) and virtualization method, and they are assigned to specific instance types for different workloads.

Mappings

Mappings are hash tables, they can be referenced to perform searches by key name and retrieve specific values. We want our `ec2-instance.yaml` template to be portable across geographic regions but also to stay in compliance for the authorized corporate AMIs.

```
Parameters:
  InstanceTypeParameter:
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - m1.small
      - m1.large
    Description: Enter t2.micro, m1.small, or m1.large. Default is t2.micro.

  MyEC2KeyPairParameter:
    Description: Existing EC2 KeyPair to enable SSH access to the instance
    Type: AWS::EC2::KeyPair::KeyName

  MySecurityGroupParameter:
    Type: List<AWS::EC2::SecurityGroup::GroupName>

Mappings:
  RegionMap:
    us-east-1:
      HVM64: ami-0ff8a91507f77f867
      HVMG2: ami-0a584ac55a7631c0c
    us-west-1:
      HVM64: ami-0bdb828fd58c52235
      HVMG2: ami-066ee5fd4a9ef77f1

Resources:
  Ec2Instance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType:
        Ref: InstanceTypeParameter
      ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", HVM64]
      KeyName: Ref! MyEC2KeyPairParameter
      SecurityGroups:
        - Ref: MySecurityGroupParameter
```



The full list of regions and AMIs with architectures is concise for brevity.

The pseudo function `AWS::Region` is used to replace the current region code into the function `FindInMap` call, this is specially useful to retrieve values, for example, account number.

The `Fn::FindInMap` function receives three parameters, the `MapName`, the `TopLevelKey`, and the `SecondLevelKey`. In this case, the function will be replaced with this values:

```
ImageId: !FindInMap [RegionMap, us-east-1, HVM64]
```

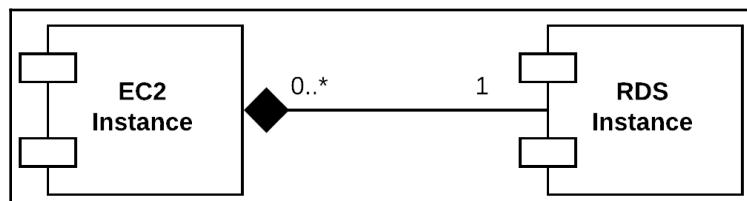
And passed to the resource as:

```
ImageId: ami-0ff8a91507f77f867
```

A completely refactored example is provided in the following link: <https://github.com/gabanox/Certified-Solution-Architect-Associate-Guide/blob/master/chapter20/ec2-instance-full.yaml>. Take your time to analyze it fully.

Depends on

AWS CloudFormation is designed to work in parallel. The engine has the intelligence to solve complex dependency graphs, for example, creating an internet gateway must be done first than a public subnet for a public facing VPC and so on. But sometimes these dependencies are solved at higher levels of abstraction and knowledge, for example, the RDS database must be created first than the EC2 instance.



To solve this kind of hard dependencies, we can use the attribute `DependsOn` as shown in the following snippet:

```
Resources:  
  MyEC2Instance:  
    Type: AWS::EC2::Instance  
    Properties:  
      ImageId:  
        Fn::FindInMap:  
          - RegionMap  
          - Ref: AWS::Region  
          - AMI  
      DependsOn: MyDB  
  MyDB:  
    Type: AWS::RDS::DBInstance  
    Properties:  
      AllocatedStorage: '5'  
      DBInstanceClass: db.m1.small  
      Engine: MySQL  
      EngineVersion: '5.5'  
      MasterUsername: MyName  
      MasterUserPassword: MyPassword
```

Once the resource creation has started, you can test out this by listing the EC2 instances and the RDS instances:

The image contains two side-by-side screenshots of the AWS console.

Top Screenshot (EC2 Instances): The URL is <https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:sort=instanceId>. The left sidebar shows 'Instances' selected. The main area displays a message: "You do not have any running instances in this region." It also includes a note: "First time using EC2? Check out the [Getting Started Guide](#)." A button at the bottom says "Click the Launch Instance button to start your own server."

Bottom Screenshot (RDS Instances): The URL is <https://console.aws.amazon.com/rds/home?region=us-east-1#dbinstances:sort=dbinstanceName>. The left sidebar shows 'Instances' selected. The main area shows one instance named "dmkeyoqrjnia8y" with the status "creating". Other columns include DB instance, Engine, Status, CPU, Current activity, Maintenance, and Class.

As you can see, the RDS is in the creating state and the EC2 provisioning process is suspended.

Helper scripts

CloudFormation is an API based interface for managing resources and every resource is an object with properties and behavior. But what happens with resources like EC2 instances where CloudFormation doesn't have any control over the operating system and internal processes? The instance status changes to **Running** and CloudFormation understands that the final state of the object has been reached successfully.

Amazon Linux images incorporate scripts to perform the following activities:

cfn-init

Installs software packages, parses metadata, creates files into disk, and configures operating system services declaratively.

```
Mappings:
RegionMap:
  us-east-1:
    HVM64: ami-0ff8a91507f77f867
    HVMG2: ami-0a584ac55a7631c0c
  us-west-1:
    HVM64: ami-0bdb828fd58c52235
    HVMG2: ami-066ee5fd4a9ef77f1

Resources:
Ec2Instance:
  Type: AWS::EC2::Instance
  Metadata:
    AWS::CloudFormation::Init:
      config:
        packages:
          yum:
            httpd: []
        files:
          "/var/www/html/index.html":
            content: "<html><body>Hello World!</body></html>"
            mode: '000644'
            owner: root
            group: root
        services:
          sysvinit:
            httpd:
              enabled: 'true'
              ensureRunning: 'true'
  Properties:
    ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", HVM64]
```

In the previous example, the `AWS::CloudFormation::Init` type will do the following activities:

- Install the `httpd` server using the latest stable version using the `yum` package manager
- Create the file `/var/www/html/index.html` with the content string and assign the proper Linux permissions
- Run the `httpd` process and configure it as an operating system daemon

cfn-signal

This script is used to send completion signals to the CloudFormation stack to notify certain activities. For example when specific scripts have been executed, software packages have been installed or external dependencies have been reached (like a payment gateway):

```
Properties:  
  ImageId: ami-a4c7edb2  
  InstanceType: t2.micro  
  UserData: !Base64  
    'Fn::Join':  
      - ''  
      - - |  
          #!/bin/bash -x  
          ./install.sh  
          # Signal the status from cfn-init  
          - '/opt/aws/bin/cfn-signal -e $? '  
          - '           --stack '  
          - !Ref 'AWS::StackName'  
          - '           --resource MyInstance '  
          - '           --region '  
          - !Ref 'AWS::Region'  
          - |+  
  
CreationPolicy:  
  ResourceSignal:  
    Timeout: PT5M
```

When the `install.sh` file has been executed the instance signals the stack which is build dynamically at runtime using the `AWS::StackName` pseudo parameter. It is possible to specify creation policies as depicted that will timeout after 5 minutes triggering a rollback.

cfn-hup

This script is a very interesting one that will poll for changes to the stack metadata. When new changes upstream are generated, this script will execute specific actions. Think of this feature as an management tool to apply changes continuously in your EC2 instances by propagating them via CloudFormation stacks. For a complete example reference, use the following link: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/deploying.applications.html>.

Multi-tier web app

Our last example consists of a three tier web app that incorporates multiple concepts that were addressed in this chapter. To demonstrate this functionality, we will use the OpsWorks **VPCELB** template. Several templates grouped by category can be viewed from the following documentation page: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/sample-templates-services-us-west-2.html#w2ab1c23c48c13c25>.

AWS OpsWorks is a configuration management service that works with Chef and Ansible. Choose the basic AWS OpsWorks stack to create the EC2 instances layer and deploy a PHP application.

AWS OpsWorks					
Template Name	Description	View	View in Designer	Launch	
Basic AWS OpsWorks stack	Creates an AWS OpsWorks stack, layer, instances and associated resources to run a PHP application.	View	View in Designer	Launch Stack	
Load-balanced AWS OpsWorks stack in a Amazon VPC	Creates an AWS OpsWorks stack with a load-balanced application that runs inside a VPC.	View	View in Designer	Launch Stack	Launch Stack

When the application is fully deployed, navigate to the OpsWorks menu in the AWS console and click on **Go to my stacks**. Here we can see CloudFormation has created the OpsWorks stack and deployed the application.

Use the ELB link to navigate to your web app, as shown in the following screenshot:

Simple PHP App

Congratulations!

Your PHP application is now running on the host “php-app1” in your own dedicated environment in the AWS Cloud.

This host is running PHP version 5.3.29.

Let's repeat the same procedure to provision the data layer of our app by using the RDS DB instance with provisioned IOPs template.

Amazon Relational Database Service						
Template Name	Description	View	View in Designer	Launch		
Amazon RDS DB instance with provisioned IOPs	Creates an Amazon RDS database instance with provisioned IOPs.	View	View in Designer	Launch Stack		
Amazon RDS DB instance with a read replica	Creates an Amazon RDS database instance with a read replica.	View	View in Designer	Launch Stack		
Amazon RDS DB instance with a deletion policy	Creates an Amazon RDS database instance with a deletion policy that specifies Amazon RDS to take a snapshot of the database before deleting it.	View	View in Designer	Launch Stack		
Amazon RDS DB instance with a database parameter group	Creates an Amazon RDS database instance with a database parameter group.	View	View in Designer	Launch Stack		

Finally, we have created several complex components using CloudFormation to build a three tier web app with Elastic Load Balancing, EC2 instances, and an RDS database.

CloudFormation Stacks						
Stack Name	Stack ID	Created Time	Updated Time	Deleted Time	Status	Status Reason
<input checked="" type="checkbox"/> RDSMySQLBasicDBSample	arn:aws:cloudformation...	2018-10-29 18:47:21 UTC-0500			CREATE_COMPLETE	
<input type="checkbox"/> OpsWorksLoadBalancedSample	arn:aws:cloudformation...	2018-10-29 18:31:38 UTC-0500			CREATE_COMPLETE	

From this point, instances and applications can be managed via the OpsWorks stack.

Best practices

- Create reusable templates and share them across the organization.
- They represent code so put them under versioning systems.
- Avoid having a single template and create multiple templates grouped by tier, owner, function, or lifecycle. Think of loosely coupled components that are service providers.
- Design complex stacks using nested stacks to promote reusability.
- CloudFormation is no silver bullet, take advantage of the strongest points of the service, and complement with OpsWorks, Terraform, Ansible, and so on.

Summary

In this chapter, you learned the basic principles of the Infrastructure as Code practice by automating ops with AWS CloudFormation. We started with the simplest infrastructure as code that can be used to create AWS resources and manage changes to streamline the change management process.

We took a deep dive to the CloudFormation templates and usage of the most used elements and created a full three tier web application.

Further reading

For more information, please visit the following AWS documentations:

- <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/conditions-section-structure.html>
- <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference.html>
- <https://aws.amazon.com/es/premiumsupport/knowledge-center/cloudformation-reference-resource/>
- <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/transform-aws-serverless.html>

20

Automation with AWS

Automation is an industry best practice that must be embraced on cloud environments, this can be achieved from simple scripts to fully managed services. With AWS automation, it is really powerful and simple. Services provide integrated functionality that can be used to initiate remediation plans, disaster recovery strategies or configuration management.

The following topics will be covered in this chapter:

- Incident Response
 - CloudWatch Logs Agent
 - CloudWatch Metric Filters

Technical requirements

There are no requirements for this chapter.

Incident Response

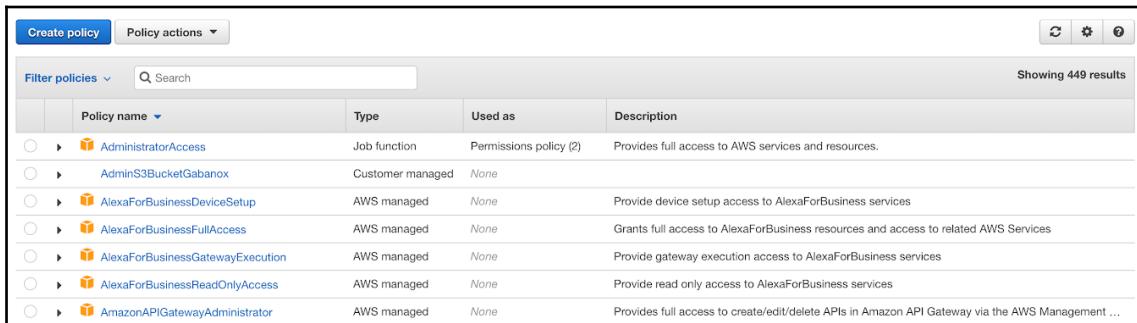
AWS has several services that provide visibility on change management, security and behavior. CloudWatch provides a monitoring system that generates service metrics via the collection of data points and the ability to create alarms and notification publishing via SNS.

CloudWatch also provides logging functionality; CloudWatch Logs manages log streams from multiple sources so system administrators can understand application behavior by collecting and analyzing application logs.

CloudWatch Logs Agent

CloudWatch provides an agent that can be configured with different filesystem sources to collect, aggregate and stream log files to CloudWatch Logs. The first step is to create an IAM policy that will provide the access required to write log streams into CloudWatch Log groups.

1. Navigate to IAM and choose **Policies | Create policy**:



The screenshot shows the AWS IAM Policies list. At the top left is a blue 'Create policy' button. To its right are 'Policy actions' dropdown and three small icons. Below the header is a search bar with 'Search' placeholder text and a 'Showing 449 results' message. A 'Filter policies' dropdown is on the left. The main area is a table with columns: 'Policy name', 'Type', 'Used as', and 'Description'. The table lists several pre-existing policies:

Policy name	Type	Used as	Description
AdministratorAccess	Job function	Permissions policy (2)	Provides full access to AWS services and resources.
AdminS3BucketGabanox	Customer managed	None	
AlexaForBusinessDeviceSetup	AWS managed	None	Provide device setup access to AlexaForBusiness services
AlexaForBusinessFullAccess	AWS managed	None	Grants full access to AlexaForBusiness resources and access to related AWS Services
AlexaForBusinessGatewayExecution	AWS managed	None	Provide gateway execution access to AlexaForBusiness services
AlexaForBusinessReadOnlyAccess	AWS managed	None	Provide read only access to AlexaForBusiness services
AmazonAPIGatewayAdministrator	AWS managed	None	Provides full access to create/edit/delete APIs in Amazon API Gateway via the AWS Management ...

2. In the **Create policy** menu use the JSON editor and paste the following IAM Policy: <https://github.com/gabanox/Certified-Solution-Architect-Associate-Guide/blob/master/chapter21/CloudWatchAgentPolicy.txt>.
3. Review and name the policy `CloudWatchAgentPolicy` and click on **CreatePolicy**.
4. Our example will use a file to configure the CWL agent. Create an S3 bucket, and upload this configuration file. Make sure to replace `my-bucket` with your own bucket name and make it readable (you can use IAM roles).

5. Now navigate to **IAM Roles** and choose **Create Role**. Choose **EC2 Role** and click **Next: Permissions**.

Create role

1 2 3

Select type of trusted entity

AWS service EC2, Lambda and others **Another AWS account** Belonging to you or 3rd party **Web identity** Cognito or any OpenID provider **SAML 2.0 federation** Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose the service that will use this role

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

API Gateway	CodeDeploy	EKS	IoT	Rekognition
AWS Support	Config	EMR	Kinesis	S3
AppSync	Connect	ElastiCache	Lambda	SMS
Application Auto Scaling	DMS	Elastic Beanstalk	Lex	SNS
Application Discovery Service	Data Lifecycle Manager	Elastic Container Service	Machine Learning	SWF
	Data Pipeline	Elastic Transcoder	Macie	SageMaker
Auto Scaling	DeepLens	Elastic Load Balancing	MediaConvert	Service Catalog
Batch	Directory Service	Glue	OpsWorks	Step Functions
CloudFormation	DynamoDB	Greengrass	RDS	Storage Gateway
CloudHSM	EC2	GuardDuty	Redshift	Trusted Advisor
CloudWatch Events	EC2 - Fleet	Inspector		
CodeBuild				

* Required [Cancel](#) [Next: Permissions](#)

6. Filter the IAM policy using the name CloudWatchAgentPolicy. Use the check button and proceed to the next step.

Create role

1 2 3

▼ Attach permissions policies

Choose one or more policies to attach to your new role.

Create policy

Filter policies ▾ CloudWatchAgentPolicy Showing 1 result

	Policy name ▾	Used as	Description
<input type="checkbox"/>	CloudWatchAgentPolicy	None	IAM Policy to write events into CloudWat...

7. You will be prompted to give the role a name, use CloudWatchAgentRole and make sure the IAM Policy is attached.

Create role

1 2 3

Review

Provide the required information below and review this role before you create it.

Role name* CloudWatchAgentRole
Use alphanumeric and '+,=,@-_ characters. Maximum 64 characters.

Role description Allows EC2 instances to call AWS services on your behalf.
Maximum 1000 characters. Use alphanumeric and '+,=,@-_ characters.

Trusted entities AWS service: ec2.amazonaws.com

Policies CloudWatchAgentPolicy

Permissions boundary Permissions boundary is not set

* Required

8. Now launch an EC2 instance using the IAM Role created (CloudWatchAgentRole).



9. Provide the following user data as initialization for your EC2 Instance. This data can be downloaded from <https://github.com/gabanox/Certified-Solution-Architect-Associate-Guide/blob/master/chapter21/bootstrap.txt>.

```
#!/bin/bash
curl
https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-setup.py -O
chmod +x ./awslogs-agent-setup.py
./awslogs-agent-setup.py -n -r us-east-1 -c s3://cloudwatch-bucket-config-file/my-config-file
```

The agent will be downloaded locally, and the Python script will be configured to use the `my-config-file` provided. When the service becomes active it will start streaming syslog data to CloudWatch as shown in the following screenshot:

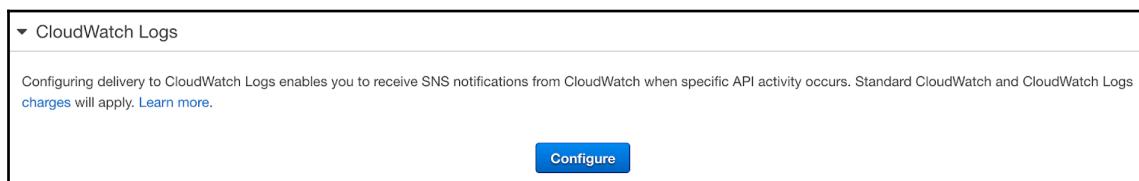
Filter events		all 2018-10-29 (19:53:10)
Time (UTC +00:00)	Message	
2018-10-30		
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] Booting paravirtualized kernel on Xen HVM	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] clocksource: refined-jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 7645519600211568 ns	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] setup_percpu: NR_CPUS=8192 nr_cpumask_bits=15 nr_cpu_ids=15 nr_node_ids=1	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] percpu: Embedded 44 pages/cpu atfffb8003e200000 s140184 r8192 d31848 u262144	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] PV spinlock hash table entries: 256 (order: 0, 4096 bytes)	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] Built 1 zonelets, mobility grouping on. Total pages: 257928	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] Policy zone: DMA32	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] Kernel command line: root=LABEL=/ console=tty1 console=ttyS0 selinux=0 nvme_core.io_timeout=4294967295	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] PID hash table entries: 4096 (order: 3, 32768 bytes)	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] Memory: 988296K/1048180K available (8204K kernel code, 1914K rdata, 2764K rodata, 2024K init, 3892K bss, 59884K reserved, 0K cma-reser	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] SLUB: HWalign=64, Order=0-3, Order=0-3, MinObjects=0, CPUs=15, Nodes=1	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] Kernel/User page tables isolation: enabled	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] trace: allocating 25743 entries in 101 pages	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] Hierarchical RCU implementation	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] RCU: restricting CPUs from NR_CPUS=8192 to nr_cpu_ids=15.	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] RCU: Adjusting geometry for rcu_fanout_leaf=16, nr_cpu_ids=15	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] NR_IRQS: 52454, nr_irqs: 952, preallocated irqs: 16	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] xen:events: Using 2-level ABI	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] xen:events: Xen HVM callback vector for event delivery is enabled	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] Console: colour VGA+ 80x25	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] console[tty1] enabled	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] console[ttyS0] enabled	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] clocksource: hpet: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 30580167144 ns	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.000000] tsc: Fast TSC calibration using PIT	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.0200003] tsc: Detected 239.953 MHz processor	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.028009] Calibrating delay loop (skipped), value calculated using timer frequency.. 4800.09 BogoMIPS (lpj=9600184)	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.036004] pid_max: default: 32768 minimum: 301	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.040020] ACPI: Core revision 20170728	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.051901] ACPI: 3 ACPI AML tables successfully acquired and loaded	
▶ 19:52:36	Oct 30 19:52:36 ip-172-31-30-4 kernel: [0.060031] Security Framework initialized	

Congratulations you have created log data ingestion solution that stores activity information from instances in a centralized way. You'll want to create an AMI with this configuration and use it as your baseline.

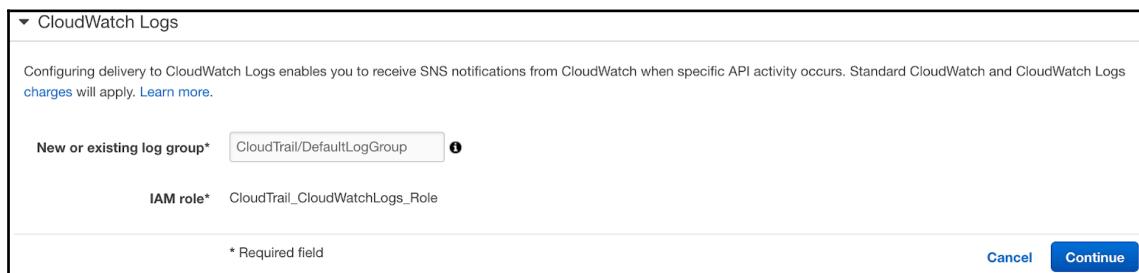
CloudWatch Metric Filters

How could we act on several events happening in our servers, infrastructure or even account activity? CloudWatch Logs solves part of the problem, now we need to have an notification system that lets us know when some events of interest occur.

1. Make sure CloudTrail is active and configured, select your trail and navigate to the **CloudWatch Logs** section.



2. Now proceed to configure the streaming of events to CloudWatch Logs. You will be required to provide a name for this log group.



3. Click on **Continue**, the last step is to verify the role that will be created for us.
You can specify custom roles for this operation.

AWS CloudTrail will deliver CloudTrail events associated with API activity in your account to your CloudWatch Logs log group

In order to successfully deliver CloudTrail events to your CloudWatch Logs log group, CloudTrail will assume the role you are creating or specifying. Assuming the role grants CloudTrail permissions to two CloudWatch Logs API calls:

1. CreateLogStream: Create a CloudWatch Logs log stream in the CloudWatch Logs log group you specify
2. PutLogEvents: Deliver CloudTrail events to the CloudWatch Logs log stream

▼ Hide Details

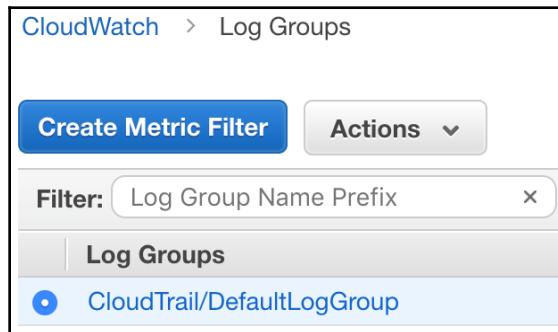
Role Summary ?

Role Description	AWS CloudTrail will assume the role you create or specify to deliver CloudTrail events to your CloudWatch Logs log group
IAM Role	<input type="button" value="Create a new IAM Role"/>
Role Name	CloudTrail_CloudWatchLogs_Role

[View Policy Document](#)

Cancel Allow

4. Click on **Allow** and you are done. The next step is to create a Metric Filter in the CloudWatch Log group created; in our case **CloudTrail/DefaultLogGroup**.



This filter has the ability to search for specific events and aggregate information about them. In our example we want to get notified when someone has three failed events when trying to access the account using wrong IAM credentials in a timespan of 5 minutes.

5. The **Create Metric Filter** can be configured to use a filter pattern that represents an expression that is applied to every single event.

Step 1: Define Pattern

Step 2: Assign Metric

Define Logs Metric Filter

Filter for Log Group: CloudTrail/DefaultLogGroup

You can use metric filters to monitor events in a log group as they are sent to CloudWatch Logs. You can monitor and count specific terms or extract values from log events and associate the results with a metric. [Learn more about pattern syntax](#).

Filter Pattern

Failure

[Show examples](#)

Select Log Data to Test

- Custom Log Data - Test Pattern

[Clear](#)

```
"sourceIPAddress": "200.91.236.86",
"userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3656.6 Safari/537.36",
"errorMessage": "Failed authentication",
"requestParameters": null,
"responseElements": {
    "ConsoleLogin": "Failure"
},
```

Results

Found 1 matches out of 29 event(s) in the sample log.

Line Number	Line Content
19	"ConsoleLogin": "Failure"

[Cancel](#) Assign Metric

6. Use the **Filter Name** as `ConsoleSignInFailures` and paste the following pattern:

```
{ ($.eventName = ConsoleLogin) && ($.errorMessage = "Failed authentication") }
```

7. We need to specify a **Metric Namespace** and a **Metric Name** as shown in the following screenshot:

Create Metric Filter and Assign a Metric

Filter for Log Group: CloudTrail/DefaultLogGroup

Log events that match the pattern you define are recorded to the metric that you specify. You can graph the metric and set alarms to notify you.

Filter Name: `ConsoleSignInFailures` i

Filter Pattern: `{ ($.eventName = ConsoleLogin) && ($.errorMessage = "Failed authentication") }`

Metric Details

Metric Namespace: `CloudTrailMetrics` i Select existing namespace

Metric Name: `ConsoleSigninFailureCount` i

[Show advanced metric settings](#)

[Cancel](#) [Previous](#) **Create Filter**

8. Click on **Create Filter**, the next step is to add an alarm when the expression matches 3 events in a period of 5 minutes.

Create Alarm

[1. Select Metric](#) **2. Define Alarm**

Alarm details

Provide the details and threshold for your alarm. Use the graph to help set the appropriate threshold.

Name:

Description:

Whenever: **ConsoleSigninFailureCount**

is:

for: datapoints i

Alarm Preview

This alarm will trigger when the blue line goes up to or above the red line for 1 datapoints within 5 minutes

ConsoleSigninFailureCount >= 3 for 1 datapoints with 5 minutes

Time	Value
10/30 01:00	3
10/30 02:00	3
10/30 03:00	3

Additional settings

Provide additional configuration for your alarm.

Treat missing data as: i

Namespace: CloudTrailMetrics

Metric Name:

Actions

Define what actions are taken when your alarm changes state.

Notification Delete

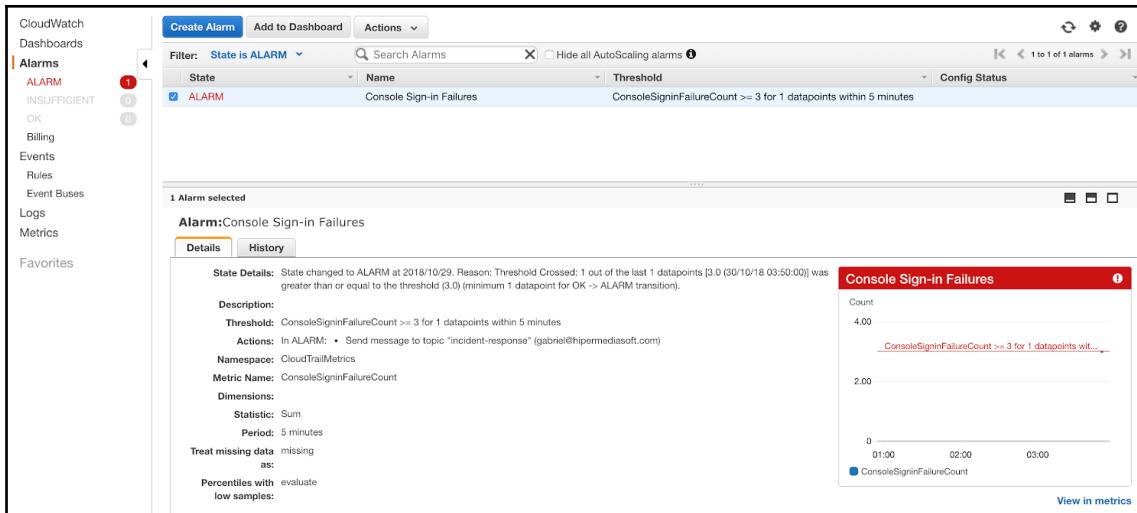
Whenever this alarm: i

Send notification to: Select list i

Email list:

[Cancel](#) [Previous](#) [Next](#) **Create Alarm**

9. To test out this behavior, try to login to the console using the same user with wrong passwords three times. The CloudWatch alarm must be in the ALARM state and an email notification must be received immediately.



10. To implement comprehensive alarms like this AWS has provided a CloudFormation template with several alarms using this model. You can deploy the following template in your accounts to keep accountable and secure.

```
https://s3-us-west-2.amazonaws.com/awscloudtrail/cloudwatch-alarms-for-cloudtrail-api-activity/CloudWatch\_Alarms\_for\_CloudTrail\_API\_Activity.json
```

Summary

In this chapter we learned about some of the monitoring and alerting capabilities of AWS with respect to managed services. Custom alarms and filters were demonstrated with the use of CloudWatch to implement an intrusion system using log analysis. We installed the CloudWatch agent for EC2 instances to stream logs directly to CloudWatch Logs.

Further reading

- **Quick start: Install and configure the CloudWatch Logs agent in a Linux EC2 instance at the time of launch:** https://docs.aws.amazon.com/es_es/AWSCloudWatch/latest/logs/EC2NewInstanceCWL.html
- **AWS Config Rules – Dynamic Compliance Checking for Cloud Resources:** <https://aws.amazon.com/es/blogs/aws/aws-config-rules-dynamic-compliance-checking-for-cloud-resources/>

21

Introduction to the DevOps practice in AWS

DevOps has become a widely adopted practice for every kind of company across industries, but, its implementation is a real challenge that requires a set of best practices, the use of specific purpose tools, mature processes, and a collaborative team. In 2011, Jon Jenkins in a velocity video shared a median time between deployments inside AWS of 11.6 seconds in a work week; this is a really impressive metric and speaks about the maturity AWS has in the DevOps industry. AWS have created many of the services used today in the DevOps landscape, such as Code Commit, CodeDeploy, CodeBuild, and CodePipeline, to name a few. They have been battle tested at global scale and are available for every AWS customer.

Technical requirements

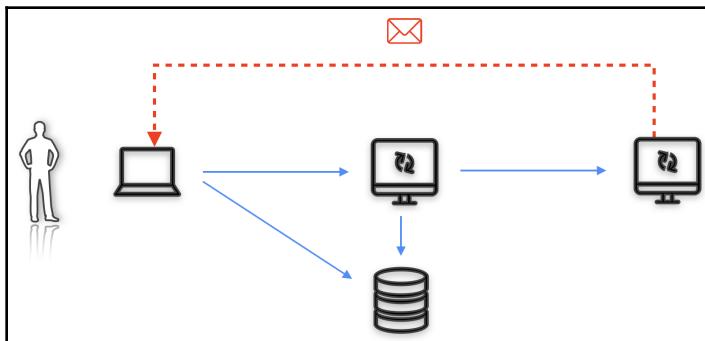
There are no requirements for this chapter.

CI / CD pipeline

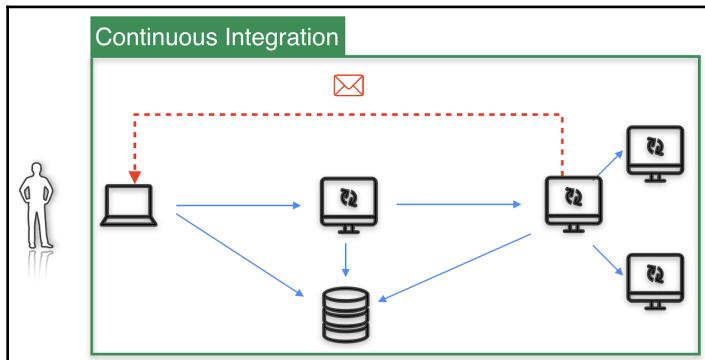
Jez Humble in his book, *Continuous Delivery*, speaks about this and he says, "*if it hurts, do it more often, and bring the pain forward.*". So, are you in pain when you deploy new versions to production, or do the development team suffers when they integrate their code every week? If you answered yes to any of these two questions, maybe you need to adopt the practice of **Continuous Integration (CI)** and **Continuous Deployment (CD)**.

Continuous Integration consist of the frequent and early integration process between code branches. This process must be made automatically every time a developer pushes code to development branches. The exhaustive code testing must be performed in parallel and provide early results about test failures, incompatible interfaces, or regressions in code. All of this has the purpose to improve the quality of software, we want to fail fast, we want to fail early, and we want to fail safe, this is why we build several isolated environments pushing forward code base from Dev, QA, and Production.

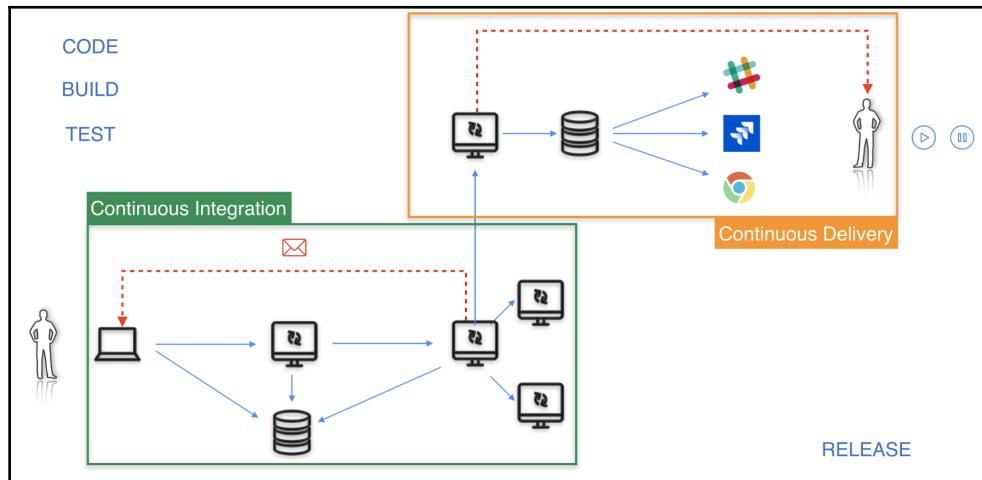
The workflow can be designed as a pipeline and it starts by cloning the source code and working locally. Once the features have been developed they are pushed back to code repositories like GitHub or CodeCommit. From here, several testing, such as code quality, can be performed with git hooks or servers like Jenkins. If all tests pass and software quality is acceptable, code is moved forward to the next step. If code fails, it is not submitted and a notification is sent to the development team and the failure is recorded.



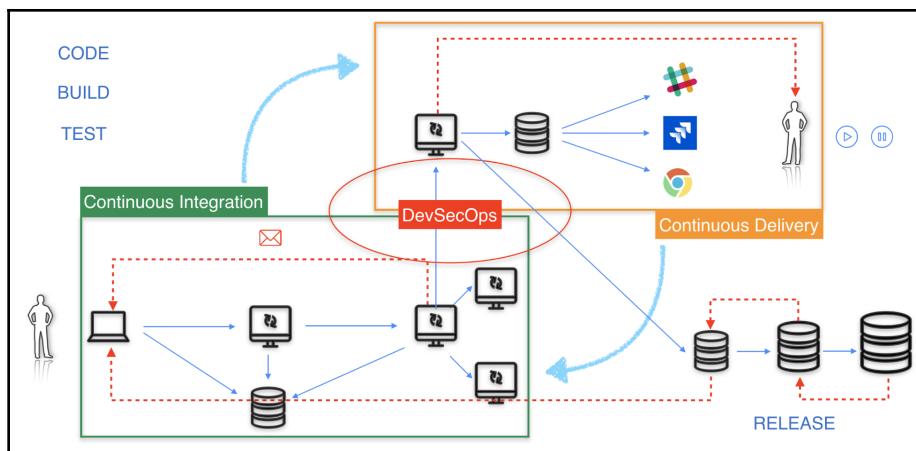
Once the fix has been done, the process is repeated until all tests pass. Other developers branches and code is integrated into the tests. This is why it is called Continuous Integration.



Now we can move to the second phase, the quality environment for **user acceptance testing (UAT)**.



Here other kind of tests are performed, such as functional and regression testing. Here a human can be involved in the quality process; a product owner that tests for usability and major features. He is the one that takes business decisions and he/she can decide to rollback, maintain, or release a new incremental product. This is the phase of Continuous Delivery and Continuous Deployment with the difference that for former deploys software artifacts in nonproductive environments and the later does it on production.



At any time the process can be rolled back and the main idea of the software pipeline is to move forward improving the quality at every stage with stable small frequent releases. This is done iteratively and uses one important dimension that traverses every process; the security DevSecOps is the practice of performing secure ops at scale on every step.

AWS provides services for every step and are depicted in the following table:

Code	AWS CodeCommit	AWS Code Pipeline AWS CodeStar
Build	AWS CodeBuild	
Test	AWS CodeBuild	
Deploy	AWS CodeDeploy	
	AWS CloudFormation	
	AWS Elastic Beanstalk	
Provision	AWS CloudFormation	
Monitor	Amazon CloudWatch	
	AWS Config	

This list is not exhaustive and several services can be used interchangeable.

AWS CodeDeploy

AWS CodeDeploy is a managed service that can be used to streamline your delivery process. CodeDeploy uses an agent installed on EC2 instances and on-premises servers to poll for software revisions available. This agent takes the control of the delivery process by pulling software artifacts from GitHub and Amazon S3 as requested from deployments.

CodeDeploy is also AWS Lambda compatible and is platform agnostic. It provides a series of lifecycle events called **hooks** when deploying new software. Examples of these hooks are the `BeforeInstall` and `ValidateService`. The `BeforeInstall` can take care of installing all required dependencies before the cut over of the previous version and the `ValidateService` hook can perform some smoke testing after deployment.

To demonstrate the use of AWS CodeDeploy, we will create a web artifact that will consist of an `index.html` web page. You can download the `index.html` sample code from <https://github.com/gabano/Certified-Solution-Architect-Associate-Guide/blob/master/chapter22/index.html> or create one by your own.

CodeDeploy will require the CodeDeploy service role to perform deployments. Navigate to **IAM, Create role**, and choose the CodeDeploy service using the **CodeDeploy** use case in the lower section of the screen:

Create role

Select type of trusted entity

1 2 3

AWS service EC2, Lambda and others **Another AWS account** Belonging to you or 3rd party **Web identity** Cognito or any OpenID provider **SAML 2.0 federation** Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose the service that will use this role

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

API Gateway	CodeDeploy	EKS	IoT	Rekognition
AWS Support	Config	EMR	Kinesis	S3
AppSync	Connect	ElastiCache	Lambda	SMS
Application Auto Scaling	DMS	Elastic Beanstalk	Lex	SNS
Application Discovery Service	Data Lifecycle Manager	Elastic Container Service	Machine Learning	SWF
Auto Scaling	Data Pipeline	Elastic Transcoder	Macie	SageMaker
Batch	DeepLens	Elastic Load Balancing	MediaConvert	Service Catalog
CloudFormation	Directory Service	Glue	OpsWorks	Step Functions
CloudHSM	DynamoDB	Greengrass	RDS	Storage Gateway
CloudWatch Events	EC2	GuardDuty	Redshift	Trusted Advisor
CodeBuild	EC2 - Fleet	Inspector		

Select your use case

CodeDeploy
Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.

CodeDeploy for Lambda
Allows CodeDeploy to route traffic to a new version of an AWS Lambda function version on your behalf.

* Required

Cancel Next: Permissions

The IAM Policy will be automatically added to the service role. Assign it a name, for example, **CodeDeployRole** and click on **Create role**, as shown in the following screenshot:

The screenshot shows the 'Create role' review step in the AWS IAM console. It's the third step in a three-step process, indicated by the number '3' in a blue circle at the top right. The page title is 'Create role' and the section is 'Review'. A note says 'Provide the required information below and review this role before you create it.' The 'Role name*' field contains 'CodeDeployRole'. Below it, a note says 'Use alphanumeric and '+=, @-_ characters. Maximum 64 characters.' The 'Role description' field contains 'Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.' Below it, a note says 'Maximum 1000 characters. Use alphanumeric and '+=, @-_ characters.' The 'Trusted entities' field shows 'AWS service: codedeploy.amazonaws.com'. Under 'Policies', there is a single policy named 'AWSCodeDeployRole'. At the bottom, there are buttons for '* Required', 'Cancel', 'Previous', and a blue 'Create role' button.

Provision a couple of EC2 instances and use the user data to bootstrap the CodeDeploy agent installation process by copy pasting the following script:

```
#!/bin/bash -x
REGION=$(curl 169.254.169.254/latest/meta-data/placement/availability-zone/
| sed 's/[a-z]$//')
yum update -y
yum install ruby wget -y
cd /home/ec2-user
wget https://aws-codedeploy-$REGION.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
```



For more details about the process, please visit: <https://aws.amazon.com/es/premiumsupport/knowledge-center/codedeploy-agent-launch-configuration/>.

We will deploy the code using a subset of EC2 instances. For this demo, we are using EC2 tags to deploy the software revision. Use the following tags for the new instances. Also make sure to specify an S3 role to gain access to the deployment S3 bucket that will be used as the software repository for the artifact.

EC2 Tags	
Name	WebApp
Environment	Production

Now we are ready to create our web app.

AppSpec file

AWS CodeDeploy uses a configuration file called `appspec.yml` that defines the deployment process, hooks, and how is the software artifact managed. In your local file system create a directory named `myApp` using the following command and `cd` into the directory:

```
mkdir myApp && cd myApp
```

Create the `appspec.yml` on the root of the `myApp` directory. For you reference, you can find the complete file at <https://github.com/gabanox/Certified-Solution-Architect-Associate-Guide/blob/master/chapter22/appspec.yml>:

```
version: 0.0
os: linux
files:
  - source: /index.html
    destination: /var/www/html/

hooks:
  BeforeInstall:
    - location: deploy/before_install
      timeout: 300
      runas: root
  AfterInstall:
    - location: deploy/restart_server
      timeout: 300
      runas: root
```

Create a directory inside `myApp` called `deploy/` here we will create two files; `before_install` and `restart_server` (for non Unix systems, specify a file extension like `.txt`).

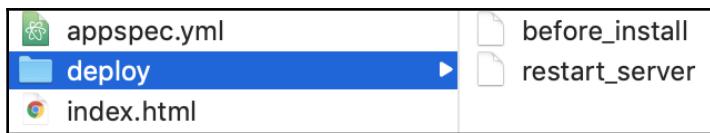
The `before_install` file can be found at https://github.com/gabanox/Certified-Solution-Architect-Associate-Guide/blob/master/chapter22/before_install:

```
#!/bin/sh
yum update
yum install -y httpd
service httpd start
```

The `restart_server` file can be found at https://github.com/gabanox/Certified-Solution-Architect-Associate-Guide/blob/master/chapter22/restart_server:

```
#!/bin/sh
service httpd restart
```

Following should be the structure of `myApp` directory:



Now it's time to compress it's contents. On Unix based systems we can make use of the command line with the following command inside the `myApp` directory:

```
zip -r ../myApp.zip *
```

Following screenshot shows the output of the preceding command:

```
Gabanox-MBP:Desktop gabanox$ mkdir myApp && cd myApp
Gabanox-MBP:myApp gabanox$ zip -r ../myApp.zip *
    adding: appspec.yml (deflated 45%)
    adding: deploy/ (stored 0%)
    adding: deploy/restart_server (stored 0%)
    adding: deploy/before_install (deflated 13%)
    adding: index.html (deflated 24%)
Gabanox-MBP:myApp gabanox$
```

On Windows machines just make sure the ZIP file contains all the files. Now we can upload this artifact to S3 via the console of the CLI.

```
aws s3 cp myApp.zip s3://<your-bucket-name>
```

The uploaded `myApp.zip` file can be seen in the following screenshot:

The screenshot shows the Amazon S3 console interface. At the top, there's a breadcrumb navigation: "Amazon S3 > code-deploy-gabanox". Below it is a navigation bar with tabs: "Overview" (selected), "Properties", "Permissions", and "Management". A search bar below the navigation bar contains the placeholder text "Type a prefix and press Enter to search. Press ESC to clear." Underneath the search bar are three buttons: "Upload", "Create folder", and "Actions". To the right of these buttons, the region "US East (N. Virginia)" is shown with a dropdown arrow. Below this, a table lists the uploaded file "myApp.zip". The table has columns for Name, Last modified, Size, and Storage class. The file details are: Name: myApp.zip, Last modified: Oct 30, 2018 5:16:35 PM GMT-0500, Size: 1.1 KB, Storage class: Standard. At the bottom of the table, it says "Viewing 1 to 1".

Now we are ready to deploy our web application. Navigate to CodeDeploy and choose **Create application**.

The screenshot shows the AWS CodeDeploy console under the "Developer Tools" section. On the left, there's a sidebar with navigation links: "Source", "Build", "Deploy", "Applications" (which is highlighted in orange), "Deployment configurations", "On-premises instances", and "Pipeline". The main content area is titled "Applications" and shows a table with columns "Application name", "Compute platform", and "Created". A search bar is at the top of the table. Below the table, it says "No results" and "There are no results to display." At the top right of the main content area, there are buttons for "View details", "Deploy application", and "Create application" (which is highlighted in orange).

You must provide a name for the application; we will use `my-app` as the name for this example. Click on **Create application**, as shown in the following screenshot:

The screenshot shows the 'Create application' configuration page in the AWS CodeDeploy console. The navigation path is: Developer Tools > CodeDeploy > Applications > Create application. The main section is titled 'Application configuration'. It contains two fields: 'Application name' with the value 'my-app' and 'Compute platform' set to 'EC2/On-premises'. At the bottom are 'Cancel' and 'Create application' buttons.

Developer Tools > CodeDeploy > Applications > Create application

Create application

Application configuration

Application name
Enter an application name
 100 character limit

Compute platform
Choose a compute platform

[Cancel](#) [Create application](#)

In the next screen you will Create a deployment group. Deployment groups are a logically group of compute resources where the application will be deployed. You will need to specify a group name, for this example, we will use Production.

Deployment group name

Enter a deployment group name

100 character limit

Specify the service role created (CodeDeployRole). And the deployment type strategy. Here we are using in-place. A link to blue green deployments is provided at the end of this chapter. Blue green is a great way to deploy software with minimum downtime and a wide use practice for delivering reliable software.

Deployment type

Choose how to deploy your application

In-place
Updates the instances in the deployment group with the latest application revisions. During a deployment, each instance will be briefly taken offline for its update

Blue/green
Replaces the instances in the deployment group with new instances and deploys the latest application revision to them. After instances in the replacement environment are registered with a load balancer, instances from the original environment are deregistered and can be terminated.

Now we need to configure our environment. Since we are deploying in a subset of EC2 instances, select **Amazon EC2 instances**, as shown in the following screenshot:

Environment configuration

Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add to this deployment

Amazon EC2 Auto Scaling groups

Amazon EC2 instances

You can add up to three groups of tags for EC2 instances to this deployment group.

One tag group: Any instance identified by the tag group will be deployed to.

Multiple tag groups: Only instances identified by all the tag groups will be deployed to.

Tag group 1

Key	Value - optional	
<input type="text" value="Name"/> X	<input type="text" value="WebApp"/> X	Remove tag
<input type="text" value="Environment"/> X	<input type="text" value="Production"/> X	Remove tag

Add tag

+ Add tag group

On-premises instances

Fill in the information related to EC2 instances keys for **Name** and **Environment**:

Deployment settings

Deployment configuration
Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application will be deployed and the success or failure conditions for a deployment.

CodeDeployDefault.OneAtATime ▾ or [Create deployment configuration](#)

Load balancer

Select a load balancer to manage incoming traffic during the deployment process. The load balancer blocks traffic from each instance while it's being deployed to and allows traffic to it again after the deployment succeeds.

Choose a load balancer

► Advanced - optional

[Cancel](#) [Create deployment group](#)

We have three deployment strategies, use the **OneAtATime**, uncheck the **Choose a load balancer** and click **Create deployment group**.

The revision (software version) can be hosted on an S3 bucket or GitHub repository. Use the S3 option and specify the URI from the previously created S3 bucket:

Create deployment

Deployment settings

Deployment group
Select a deployment group
 Production ▾

Application
my-app

Compute platform
EC2/On-premises

Deployment type
In-place

Revision type
 My application is stored in Amazon S3 My application is stored in GitHub

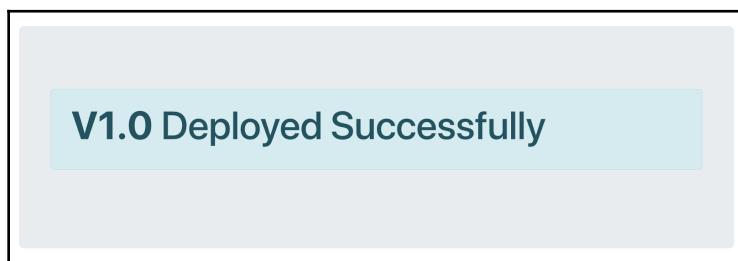
Revision location
Copy and paste the Amazon S3 bucket where your revision is stored
 s3://code-deploy-gabanox/myApp.zip X
s3://bucket-name/folder/object.[zip|tar|tgz]

Revision file type
 .zip ▾

CodeDeploy automatically starts the deployment of the revision created using the deployment group targets. Because we choose one at a time, the behavior is one deployment after another:

The screenshot shows the AWS CodeDeploy console interface. On the left, there's a navigation sidebar with links like 'Source > CodeCommit', 'Build > CodeBuild', 'Deploy > CodeDeploy' (which is expanded), 'Getting started', 'Deployments', 'Deployment' (which is selected and highlighted in orange), 'Applications', 'Deployment configurations', 'On-premises instances', and 'Pipeline > CodePipeline'. Below that are 'Feedback' and 'Return to the old experience' links. The main content area has several sections: 'Deployment status' showing 'Installing application on your instances' with a progress bar at 1 of 2 instances updated, 'Deployment details' showing Application: my-app, Deployment ID: d-DHYXKV59W, Status: In progress, Deployment configuration: CodeDeployDefault.OneAtATime, Deployment group: Production, Initiated by: user, 'Revision details' showing Revision location: s3://code-deploy-gabanox/myApp.zip, Revision created: Just now, Description: Application revision registered by Deployment ID: d-DHYXKV59W, and 'Deployment lifecycle events' which lists two instances: i-05f0aebbf13ab3f49 and i-010a0742db57f42f3. The first instance is in progress with the most recent event being 'AfterInstall' and the second instance has succeeded with the most recent event being 'ValidateService'.

After the deployment process, the bar will be green and it is great. You have created a fully automated process to deliver software frequently using AWS managed services; feel free to play around with other software version and other configurations. Use the instances public IPs to validate the deployment.



Congratulations!

Summary

We presented an overview of the Continuous Integration and Continuous Deployment practices from the DevOps perspective by defining the pipeline workflow. We discussed about several services that AWS provides for DevOps engineers running workloads in AWS and created a Continuous Delivery process by deploying a software version on EC2 instances using AWS CodeDeploy.

Further reading

- **Development operations and AWS:** <https://aws.amazon.com/es/devops/>
- **Blue/Green Deployments on AWS:** https://d1.awsstatic.com/whitepapers/AWS_Blue_Green_Deployments.pdf

22

Mock Test 1

1. An enterprise customer is migrating his Oracle database and the DBA has stated that on AWS we need to have 10,000 IOPS in order to satisfy the performance demands. Replication is done via Oracle Data Guard and this must work in the cloud too. What instance type would you choose for this requirement?
 - A. Dense Storage Instances
 - B. High I/O instances
 - C. Memory Optimized Instances
 - D. RDS db.t2.xlarge
2. A distributed application is using SQS to send 350 messages per second. Client applications receive empty responses and experience high CPU processing from workers. How could you solve this issue in the simplest way?
 - A. Change the `WaitTimeSeconds` attribute to a value greater than zero in the `ReceiveMessage` API call
 - B. Set the `ReceiveMessageWaitTimeSeconds` queue attribute to something different to zero
 - C. Configure short polling in the queue and change to the `.fifo` suffix
 - D. Extend the visibility timeout of the queue

3. You have been designated to design a highly available, highly durable storage solution for customer statements for a big bank company. The solution must be economically efficient. **Financial statements (FS)** are generated monthly and must survive the concurrent loss of two Availability Zones, a **credit score (CS)** is calculated by using the original financial statement and stored in XML. **Risk scores (RS)** are stored as text files and require as input the customer statements (FS) and **credit scores (CS)** to be calculated in the next 2 months. These documents must be retained for 2 years before deletion. Choose the S3 storage model appropriate for this use case.
- A. Store FS with S3 Standard-IA, CS in S3 One Zone-IA, RS in Glacier with Lifecycle management to delete RS after 2 years.
 - B. Store FS with S3 Standard, CS in S3 Standard with RRS option, RS in Glacier with Lifecycle management policy to delete RS after 2 years.
 - C. Store FS with S3 Standard, CS in S3 Standard with RRS option, RS with S3 Standard and Lifecycle management policy to transition files to Glacier after 60 days and delete them after 2 years
 - D. Store FS with S3 Standard, RS in Glacier and CS in S3 Standard with RRS option and Lifecycle management policy to transition files to Glacier after 60 days and delete them after 2 years
4. **AWSGeek** is a company that sells training videos on their website. Our CIO has told the business it is losing revenue because S3 links to videos are leaked on the internet. Which approach would you take in order to protect the video assets in a scalable fashion, while optimizing for performance?
- A. Submit a change request to the frontend development team to retrieve the user's current IP and change the S3 bucket policy to allow access only from known clients
 - B. Use cookies on the web server to identify the logged-in user and create a CloudFront distribution to accelerate video downloads
 - C. Use S3 pre-signed URLs and CloudFront signed cookies
 - D. Use CloudFront by creating web distributions. This distribution unique ID (<http://d111111abcdef8.cloudfront.net/videos/video1.mp4>) permits confidentiality

5. You, as a solutions architect, are required to fix an issue in a three-layer application. Network segregation is done via public and private subnets; web servers are placed in the public subnet with t3.medium instances and app servers with m5.large instances in the private subnet. The company recently increased the web and app server layers horizontally due to a growth in sales. High latency is reported when downloading data from an external service provider. Which could be the solution for this scenario?
 - A. Add an internal load balancer to increase throughput in the app layer
 - B. Add two Elastic Network Interfaces per instance and increase the instance size
 - C. Migrate the NAT instance to a NAT Gateway
 - D. Compress app layer outbound traffic
6. A retail company is considering using the AWS Cloud to design a disaster recovery strategy. The company runs on-premises infrastructure. Which of the following options is not a good DR solution?
 - A. Storage Gateway, Route53, and S3 with S3 One Zone-IA
 - B. Import/Export Snowball, EBS, Glacier
 - C. Direct Connect, VPC, Glacier
 - D. S3 Standard, EBS Snapshot, Storage Gateway stored volumes
7. AWSome Products, an e-commerce company, is designing an application that must be compliant with the **Payment Card Industry Data Security Standard (PCI DSS)** in order to store sensitive user information in S3. The company wants to leverage the capabilities of the cloud with a cryptographic solution that provides unique keys for each object and manage all the security operations. Also, the InfoSec department requires that every uploaded object is always encrypted to prevent object loss. Which one is the best option for this scenario?
 - A. SSE - KMS with a bucket ACL with READ-only permissions and MFA delete
 - B. SSE - S3 with a bucket policy using a string condition and versioning
 - C. SSE - C with a bucket policy using a string condition and object lifecycle
 - D. AWS KMS-Managed **Customer Master Key (CMK)** with envelope encryption

8. A government agency wants to transfer 1 PT of data over the internet, and for this transfer to be successful it must be done in 1 week. The ISP connection can only handle 100 Mbps and the network is at 80% use throughout the week. This job needs to be cost efficient and provide end-to-end security. Which solution describes the best alternative for the agency with the current restrictions?
- A. Install the AWS CLI and use the s3 cp command and work in parallel
 - B. Use VM Import/Export and Storage Gateway
 - C. Use Direct Connect with a 1 Gbps port to perform the transfer
 - D. Use AWS Snowball with several appliances
9. Which statement about RDS is incorrect?
- A. RDS read replicas work in asynchronous mode
 - B. RDS Multi-AZ instances are resilient to hardware failure
 - C. RDS can be provisioned with 95 GB SSD and 1,000 provisioned IOPS
 - D. RDS provides TDE for SQL Server and Oracle Databases
10. A coworker asks you to create a RAID 1 configuration on a file server to improve performance and fault tolerance. What recommendation would you give to him?
- A. Create a RAID 5 configuration to provide low redundancy cost
 - B. Create a RAID 1 to mirror every block to a redundant disk
 - C. Create EBS snapshots scheduled with a snapshot lifecycle policy
 - D. Create a RAID 0 to distribute I/O between the available disks in the array
11. An infrastructure engineer is migrating their CRM application to Docker containers; the application is stateful and works with HTTP. He is asking for advice on which kind of load balancer must be used for this scenario. What kind of load balancer is needed and which configuration attributes are relevant?
- A. Classic Load Balancer with sticky sessions and cross-zone load balancing
 - B. Network Load Balancer with an Elastic IP and cross-zone load balancing
 - C. Application load balancer with sticky sessions and cross-zone load balancing
 - D. Route 53 with a CNAME and a Classic Load Balancer with sticky sessions

12. Your company is developing a software as a Service solution for an ERP software hosted in the AWS marketplace. The servers created by end customers need to provide confidentiality to provide users with usernames and passwords for the first run. How would you implement such a feature in a secure and simple way?
- A. Store all the usernames and passwords in an S3 bucket and include a fetch script in the application code
 - B. Use an RDS database with a table storing the usernames and passwords unencrypted
 - C. Use the instance metadata and ask the user to log in with the private key certificate so he can read the username and password
 - D. Send the username and password over simple email
13. The CIO has contracted you to implement a solution that provides full visibility and an automatic response to audit every system change in the corporate AWS account. The solution must be scalable and cost-efficient. What will you do?
- A. Use CloudTrail and AWS Config with custom Lambda functions
 - B. Use CloudWatch with metrics and filters that alert changes to SNS topics
 - C. Use a marketplace instance that records every infrastructure change in the local filesystem
 - D. Use CloudTrail and API access to system events
14. In which service are you not allowed to use SSH to manage the operating system?
- A. RDS
 - B. EMR
 - C. EC2
 - D. Elastic Beanstalk
15. Which responsibilities does the customer take, according to the shared responsibility model in AWS? Mark all that apply.
- A. Decommissioning storage devices
 - B. Encryption of EBS volumes
 - C. Management of IAM credentials
 - D. Controlling physical access to data centers

16. A SysOps admin wants to be notified about important events when working with Amazon Glacier. What kind of notifications are available to achieve this?
 - A. SNS topics when files are copied to S3
 - B. Archival complete and Glacier add files
 - C. Vault Inventory and Retrieval Job Complete
 - D. Retrieval Job Complete only
17. Which of the following statements are true about VPC subnets? Choose all that apply.
 - A. /24 CIDR block for subnets can be chosen
 - B. /17 CIDR block for VPCs can be chosen
 - C. Each subnet spans only one AZ
 - D. Each subnet spans at least two AZs
18. To provide the highest network throughput you can do all the following, except ___. Choose the wrong option.
 - A. Use Placement Groups
 - B. Enable Jumbo frames
 - C. Use Spot Instances
 - D. Use enhanced networking
19. Gamma Games is designing a real-time mobile app that uses SQS queues to send playback history to other users in a reliable way; the application is Java serverless and uses DynamoDB to track all the game records. As the game becomes more popular, history is exceeding 256 KB per message. Which could be a good solution for this?
 - A. Change the database layer from DynamoDB to RDS, using blobs to store the game history
 - B. Use compression to store the game history in a smaller SQS message
 - C. Use the DynamoDB enhanced client to store the history blob in S3
 - D. Store blob information in S3 and send the reference in the SQS message

20. A high traffic news website is low performing at peak times. After investigating, you notice that hot news queries are slowing the database. Hot news must be refreshed every 15 minutes. Which could be a non-intrusive scalable solution for this problem?
- A. Use ElastiCache for the hot news queries and modify the application DAO
 - B. Use CloudFront with a web distribution and a TTL for hot news of 900 seconds
 - C. Change the database instance type to a bigger one with more memory available
 - D. Use read replicas of the primary database and modify the application DAO

23

Mock Test 2

1. When designing a storage strategy for Amazon S3, which feature should you highlight to your cost center manager due to the additional cost it brings.
 - A. MFA Delete
 - B. Versioning
 - C. Lifecycle Rules
 - D. S3 Bucket Policies
2. You are working as a Solutions Architect for a large retail company. You are required to create a virtual private cloud which connects to your on-premise environment to create a hybrid cloud. Not all applications from your on-premise datacenter can be migrated to AWS. Which option is best suited to create the most reliable connection with the highest throughput for your hybrid environment?
 - A. Amazon CloudFront
 - B. VPN Connectivity
 - C. Direct Connect
 - D. VPC Endpoints
 - E. Internet Gateway

3. You have configured a VPC with 2 subnets, one is to be used as a public subnet and another as a private subnet. You have already created an Internet Gateway, but what 2 additional steps do you need to configure to ensure this is a public subnet?
 - A. Add a route from your public subnet that points to 0.0.0.0/0 via the IGW
 - B. Configure a Bastion Host
 - C. Attach your IGW to your VPC
 - D. Add a route from your private subnet that points to 0.0.0.0/0 via the IGW
4. From a disaster recovery perspective, what is the recovery time objective (RTO)?
 - A. RTO is defined as the maximum amount of time for which a data could be lost for a service.
 - B. RTO is defined as the amount of time it takes for your core services to meet restart after a failure
 - C. RTO is defined as the amount of time left between backups in your backup strategy
 - D. RTO is defined as the maximum amount of time in which a service can remain unavailable before it's classed as damaging to the business
5. You have been asked to ensure that your organization's data is encrypted when stored on S3. The requirements specify require that encryption must happen before the object is uploaded using keys managed by AWS. Which S3 encryption options is best suited?
 - A. SSE-KMS
 - B. CSE-KMS
 - C. SSE-S3
 - D. CSE-C
 - E. SSE-C

6. You have been asked to design and implement a new disaster recovery strategy for your organization. The directory has indicated the importance of the service and that it must be available with the lowest RTO. There are no other requirements for this strategy. Which DR option would meet the needs of the director?
 - A. Backup
 - B. Pilot Method
 - C. Warm Standby
 - D. Multi-Site
7. When IAM policies are being evaluated for their logic of access, which 2 of the following statements are incorrect?
 - A. Explicit denies are always overruled by an explicit allow
 - B. The order in which the policies are evaluated does not matter to the end result
 - C. Explicit allows are always overruled by an explicit deny
 - D. Access to all resources are denied by default until access is granted
 - E. Access to all resources are allowed by default until access is denied
8. You are investigating an ongoing incident which was caused by an EC2 instance being manually shut down. You have been asked by your manager to identify who requested the shutdown to ascertain the reasoning behind it. Which service would you use to identify the identity of the API call which results in the shutdown of the resource?
 - A. Amazon CloudWatch
 - B. AWS CloudTrail
 - C. Amazon Inspector
 - D. AWS Redshift
9. If your **Elastic Load Balancer (ELB)** finds an unhealthy instance while performing health checks for your EC2 fleet, what default action does your ELB take?
 - A. Your ELB will launch new EC2 instances to replace the unhealthy ones
 - B. Your ELB will notify you via SNS to allow you to add an additional healthy instance

- C. Your ELB will stop sending requests to the unhealthy instances
D. Your ELB will stop sending requests to the fleet until the healthy instance is removed or replaced
10. Your company needs to understand the security provided by AWS Glacier before using it for storage to help enforce regulatory and compliance requirements. Which of the below statements is true with regards to AWS Glacier data security?
- A. You can set up MFA delete against your vaults
B. You can use Vault Lock policies to lock vault access policies
C. AES-128 is used to encrypt all data
D. You can use bucket policies to control who has access to your vaults
11. You have been using AWS Elastic Beanstalk to provision and build your environment, deploying applications quickly and easily. The finance department has asked how much Elastic Beanstalk is costing per deployment. Which statement is correct relating to AWS's billing policy for AWS Elastic Beanstalk?
- A. You only pay for the cost of resources deployed providing that the resources are located within the same availability zone within a region
B. You are charged \$0.10 per resource deployment across all regions in your AWS account
C. You pay a one-time monthly use fee of Elastic Beanstalk allowing you to deploy as many resources as required for the same flat fee
D. You only pay for the cost of resources deployed, there is no additional charge for using Elastic Beanstalk
12. The development team has contacted you explaining that their web instance in the public subnet is no longer able to communicate with the application instance in the private subnet. Select 2 options that could be the cause of this communication issue between the 2 instances.
- A. The security groups associated for each instance could be misconfigured
B. There is no bastion host configured within the public subnet to allow the connection to the private subnet

- C. The internet gateway has been created for the VPC but it has NOT yet been attached to the VPC
- D. The Network Access Control Lists might be blocking the traffic at a network level
- E. The Security group rule set associated to the application instance has been set to allow ANY protocol from ANY source
13. As the administrator of your VPC configuration you are refining the access of your private instances access to the internet. Looking at the difference between NAT instances and NAT gateway you see there are additional benefits with a Gateway. What additional benefits do Gateways have over NAT instances? (Choose 3).
- A. You can launch more than one NAT Gateway for scalability
- B. Increased level of performance optimized for NAT traffic
- C. Uniform offering of instance size and type
- D. Increased Bandwidth over NAT instances
- E. Can be used as a Bastion Server
14. Working as an architect you highlight the importance of high availability with their AWS RDS database and suggest that they implement Multi-AZ. The customer ask you what benefit they gain from implementing this feature. How does Multi-AZ help to maintain a level of high availability?
- A. Multi-AZ allows a secondary RDS instance in a different availability zone to that of the primary to take the role of the primary DB in the event of it failing
- B. Multi-AZ allows a secondary read replica in a different availability zone to that of the primary to take the role of the primary DB in the event of it failing
- C. Multi-AZ allows a secondary RDS instance in a different region to that of the primary to take the role of the primary DB in the event of it failing
- D. Multi-AZ allows another primary RDS instance in a different AWS account to that of the primary to take the role of the primary DB in the event of it failing

15. You have been hired to help implement a static website for a pharmaceutical company which is projected to expand at a high rate over the next 2 years. The customer base of the company has a global reach and the customer has explained it must remain highly available. After reviewing the operations of business you suggest that Amazon S3 is used to host the website. What advantages does S3 have to help you implement this solution and the demands upon it over the next 2 years?
- A. S3 has integration with AWS Config to help with worldwide distribution of content traffic
 - B. S3 is highly scalable and highly available due to its design
 - C. S3 can scale to handle almost any load automatically
 - D. S3 has integration with Amazon CloudFront to help with worldwide distribution of content traffic
 - E. S3 can be configured through an AWS support ticket to have increased throughput to handle unexpected loads
16. After configuring your EC2 instance you are required to select a _____ to enable you to connect to your instance securely.
- A. Secret Access Key
 - B. Key Pair
 - C. IAM Role
 - D. Security Group
17. You are discussing EC2 options with a customer who has concerns over security when using shared tenancy. Which other tenancy options might be preferable to this customer? (Choose 2)
- A. VMware on AWS instances
 - B. Dedicated Hosts
 - C. Dedicated Servers
 - D. Isolated Instances
 - E. Dedicated Instances
18. You are designing storage requirements for a client and you suggest EBS as a storage solution. During conversations with your client you explain your reasoning but they have queries and concerns over EBS scalability and resiliency. What features of EBS can you explain to your client to reassure them? (Choose 3)
- A. EBS volumes are automatically replicated across other hosts within the same availability zone for resiliency

- B. EBS volumes are easily provisioned to offer additional storage as and when required
- C. By creating a snapshot of a volume you can launch a larger volume using the same snapshot.
- D. EBS replicates all volumes writes to S3 for additional resiliency
- E. EBS volume are automatically backed up by default to at least 2 other regions
19. You are explaining the differences between Security Groups and Network Access Control Lists to a customer. What 2 key points are important to understand when understanding how these 2 security controls differ from each other? (Choose 3)
- A. Security groups are stateful by design and NACLs are not
- B. NACLs are stateful by design and security groups are not
- C. Security groups allow you to add a 'Deny' action within the rule set
- D. NACLs allow you to add a 'Deny' action within the rule set
- E. Security groups control access at the instance level
- F. NACLs control access as the instance level
20. You notice that one of your EC2 instances has a failed system status check. Which action should you take to resolve the issue as quickly as possible?
- A. Reboot the instance to perform the System Status check again
- B. Increase the instance size for better performance
- C. Stop and start the instance causing it to launch on a different host
- D. Troubleshoot the network configuration of the instance to find the cause
21. As a part of a distributed and secure document repository, you are using Amazon S3 with Amazon CloudFront. What is the best method of ensuring all web requests to the documents traverse your CloudFront infrastructure rather than the S3 public URL?
- A. Configure S3 Bucket permissions to ensure access is granted via the CloudFront distribution
- B. Configure an **Origin Access Identity (OAI)** and associate that to the relevant CloudFront Distribution

- C. Set the principal of the S3 bucket policy as the CloudFront distribution
D. Create an IAM user with the relevant permissions to access the bucket and share this identity for permitted users only.
22. By default which metric of EC2 is not monitored and measured by Amazon CloudWatch?
- A. CPU Utilization
B. Disk Read Operations
C. Network In
D. Memory Utilization
23. As an administrator, you have configured an AWS Topic to notify engineers when a threshold is met from a metric within CloudWatch to allow them to proactively respond. After adding the engineers to the topic using the Email endpoint, one of the engineers explains that they are not receiving topic notifications. What is the likely cause of the problem?
- A. The engineer does not have the `sns:notify` permission against his IAM user account
B. The engineer has not accepted the subscription confirmation
C. The engineer has exceeded his inbound quota for SNS notifications
D. The total number of subscribers has been exceeded resulting in some engineers not receiving notifications
24. As the CTO of a fast-paced gaming company you are looking to migrate your infrastructure to AWS. You have a requirement to use a database that can deliver reliable performance at scale which provides consistent single-digit millisecond latency and in-memory caching. You aim to use this to store high scores, dynamic content and query game data. Which of the following services would be best suited for this environment?
- A. Amazon Redshift
B. Amazon RDS
C. Amazon DynamoDB
D. Amazon EMR

25. Your manager has explained that the cost of your fleet of instances is too high and that resources are being wasted on each instance. As a result, you need to implement a more effective and efficient method of deploying resources. What are the best methods to help minimize the costs of your instances? (Choose 2)
- A. Increase the size of your EC2 instances but reduce the quantity
 - B. Implement Auto Scaling to manage the size of your fleet with demand
 - C. Use small instances to reduce resource wastage
 - D. Manually shut down your instances at the end of the day
 - E. Reduce the size of the ephemeral storage on each instance
26. As a user of a KMS CMK you need to delegate a subset of your own permissions to another principal within the security team. What element of KMS allows you to do this as a form of resource-based access control?
- A. KMS Grants
 - B. KMS Key Policies
 - C. KMS Data Encryption Keys
 - D. KMS CMK Principal delegation
27. You are a cloud support engineer for a large retail organization, and your security and compliance team has expressed the requirement of a configuration management solution in your AWS environment. You have suggested that AWS Config would be the best solution to meet all requirements. What two features does AWS Config provide as a benefit to configuration management solution? (Choose 2)
- A. Automatic deployment of patch fixes
 - B. IAM permission summaries
 - C. Identify cost savings through best practices
 - D. History of configurational changes
 - E. Compliance checks

28. As a solutions architect you are looking for a service that can act as a lightning fast in-memory data store, you settle on AWS Elasticache but are unsure about which engine to use, Memcached or Redis. You decide to go with Redis as it offers a richer set of features. Which features below does Redis offer that Memcached does not? (Choose 3)
- A. Data partitioning
 - B. Replication
 - C. Snapshots
 - D. Sub-millisecond latency
 - E. Advanced data structures
29. Select the features which best describe the Simple Queue Service FIFO queues over standard queues. (Choose 3)
- A. Exactly-once processing
 - B. At-least-once delivery
 - C. Best-effort ordering
 - D. First In First Out delivery
 - E. 300 **transaction per second (TPS)**
30. Which of the following services does CloudFormation not support?
- A. AWS CloudTrail
 - B. Amazon CloudWatch
 - C. Amazon Glacier
 - D. Amazon EBS
31. As a part of designing your new application uses best practices of a decoupled environment you have implemented SQS to manage messages between multiple AWS services. In addition to your standard and FIFO queues, you decide that implementing dead letter queues are a good idea. What benefits do dead letter queues provide? (Choose 2)
32. One of your S3 bucket policies has the following entry:

```
{  
    "Version": "2012-10-17",  
    "Id": "S3BucketPolicy",  
    "Statement": [  
        {  
            "Sid": "IPAddress",  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": "s3:*",  
            "Resource": "arn:aws:s3:::mybucket/*"  
        }  
    ]  
}
```

```
        "Resource": "arn:aws:s3:::packtbucket/*",
        "Condition": {
            "IpAddress": {"aws:SourceIp": "54.240.143.0/24"},
            "NotIpAddress": {"aws:SourceIp": "54.240.143.188/32"}
        }
    }
]
```

```
}
```

Which statement is true about the bucket policy?

- A. Access is allowed to all S3 buckets in the account with a source IP address of 54.240.143.188/32
 - B. Access is allowed to the packtbucket as long as the source IP address is 54.240.143.188/32
 - C. Access is allowed to all buckets for all IP addresses within the subnet of 54.240.143.0/24 except 54.240.143.188/32
 - D. Access is allowed to the packtbucket for all IP addresses within the subnet of 54.240.143.0/24 except 54.240.143.188/32
 - E. Access is allowed for the IAM principal account to the packtbucket if its source IP address is 54.240.143.0/24
33. Which of the following statements relating to security groups is incorrect?
- A. They are stateful by design
 - B. You are able to explicitly deny traffic that meets specific rules
 - C. They offer security at the instance level
 - D. All rules are read before an action is taken
34. You have designed a multi VPC configuration for your organization to operate different services. However, you need some EC2 instances to communicate with each other between the VPCs. VPC Peering will enable this functionality, however, what important factors do you need to consider when configuring the connectivity. (Choose 2)
- A. The EC2 instance types need to be within the same family for EC2 peering
 - B. The VPC CIDR Blocks of each VPC need to be the same size
 - C. The VPC CIDR Blocks of each VPC can't overlap

- D. Both VPCs need to have public subnets to allow the connectivity
E. Security groups of the EC2 instances need to be updated to allow access
35. Working as the security administrator of your AWS Account for an AI and robotics organization, you want to grant access for the development to be able to administer and manage AWS resources programmatically through SDKs, REST, and Query API operations. What should you configure within IAM to allow these user to carry out this task?
- A. Issues access keys to each IAM user
 - B. Issue Key Pairs to each IAM user
 - C. Configure the IAM users for MFA access
 - D. Add an inline policy for each IAM user granting programmatic access
36. Your team need to have programmatic access to AWS through the AWS CLI. You have contacted your IAM administrator and they have issued access keys to all the users. What command needs to be run on their client to set up the AWS CLI with the corresponding access keys?
- A. AWS Setup
 - B. AWS CLI Configure
 - C. AWS CLI Setup
 - D. AWS Configure
 - E. AWS Configure Setup
37. An incident occurs against one of your applications, immediately you use CloudTrail to help you identify who or what was happening at the time of the incident. However, you can't find any log information at the time of the incident. Which statement is correct regarding AWS CloudTrails log files?
- A. Log files are delivered by CloudTrail to S3 within 24 hours of an API call
 - B. Log files are delivered by CloudTrail to S3 within 1 hour of an API call
 - C. Log files are delivered by CloudTrail to S3 within 30 minutes of an API call
 - D. Log files are delivered by CloudTrail to S3 within 15 minutes of an API call

38. You are looking to implement an element of automation within your AWS environment for your web infrastructure to ensure specific standards and compliance are met from both internal and external governance controls. You want to ensure that all EBS volumes deployed are encrypted and that you are only using specific instance types. If EBS volumes are deployed without encryption or the wrong instance type is used, you want to be automatically notified. What service would you use to implement the compliance requirements?
- A. AWS CloudTrail
 - B. AWS CloudFormation
 - C. AWS Config
 - D. AWS Systems Manager
 - E. AWS Trusted Advisor
39. You have decided to use an AWS VPN to connect your AWS environment to your corporate network via a customer gateway with two tunnels. For DR purposes and resiliency you have configured a second customer gateway in case the first customer gateway fails. What statement is false when configuring customer gateways?
- A. You must update your routing tables for your subnets
 - B. Specify the routing type required, either static or dynamic
 - C. The IP address associated to the customer gateway needs to be private
 - D. Enable route propagation
40. Working for a large financial organization, encryption of data is key. KMS is heavily within your environment and you have set up access controls to allow certain users to access specific CMKs. You want to use IAM to control these permissions rather than key policies in KMS. What statement needs to be added to your CMK key policy to enable you to perform these actions?
- A.
- ```
{
 "Sid": "Enable IAM User Permissions",
 "Effect": "Allow",
 "Principal": {"AWS": "arn:aws:iam::123456789123:root"},
 "Action": "kms:*",
 "Resource": "*"
}
```

B.

```
{
 "Sid": "Enable IAM User Permissions",
 "Effect": "Deny",
 "Principal": {"AWS": "arn:aws:iam::123456789123:root"},
 "Action": "kms:*",
 "Resource": "*"
}
```

C.

```
{
 "Sid": "Enable IAM User Permissions",
 "Effect": "Allow",
 "Principal": {"AWS": "arn:aws:iam::123456789123:IAM"},
 "Action": "kms:*",
 "Resource": "*"
}
```

D.

```
{
 "Sid": "Enable IAM User Permissions",
 "Effect": "Allow",
 "Principal": {"AWS": "arn:aws:iam::123456789123:root"},
 "Action": "kms:decrypt",
 "Resource": "*"
}
```

41. To help manage resource with demand for your web application you have implemented an internet facing elastic load balancer and Auto Scaling across your EC2 fleet within your VPC. The VPC configuration consists of 1 public subnet and 2 private subnets. Your EC2 fleet resides within the public subnet. Which statement is false relating to the ELB in this scenario?

- A. The ELB must reside in the public subnet
- B. The ELB and all the EC2 fleet should be in the same subnet
- C. The ELB should have a route to the internet gateway
- D. The ELB can detect unhealthy targets

42. Your client is experiencing issues with his resources not being able to quickly meet the demand of traffic. You have explained that configuring Auto Scaling and ELB will help to resolve their issues. What elements of Auto Scaling need to be configured? (Choose 3)
- A. Scaling Policy
  - B. Launch Configuration
  - C. Auto Scaling Group
  - D. Scaling Permissions
  - E. Launch Policy
43. You are using EBS volumes as block storage for your EC2 fleets across your VPC which spans multiple AZs. To resolve an ongoing incident you need to detach an EBS volume from an instance in AZ-1 and reattach it to an instance in AZ-2. What is the best method to achieve this outcome?
- A. Stop both instances in AZ-1 and AZ-2. Detach the EBS volume from the instance in AZ-1. Reattach the EBS volume to the instance in AZ-2
  - B. Stop the instance in AZ-1. Create a snapshot of the EBS volume. Create a new instance from the snapshot in AZ-2
  - C. Stop the instance in AZ-1. Create a snapshot of the EBS volume. Create a new volume from the snapshot in AZ-2. Attach to the instance in AZ-2
  - D. Create a new EBS volume in AZ-2 pointing to the target EBS volume in AZ-1. Attach the new EBS volume to the instance in AZ-2
44. Which features do the Application load balancer and Network load balancer have that the Class load balancer does not? (Select 3)
- A. Health Checks
  - B. Load Balancing to multiple ports on the same instance
  - C. CloudWatch Metrics
  - D. IP address as targets
  - E. Load balancer deletion protection

45. What is the default limit of reserved Elasticache nodes that you can have without having to contact AWS Support to request an increase?
- A. 10
  - B. 20
  - C. 30
  - D. 40
46. You work for a large enterprise that requires hundreds of users to gain access to the AWS Management console. As the solution architect, you propose the quickest and best approach to allow authentication for that many users would be to use SSO with SAML 2.0 based federation using your MS Active Directory server. Before federated access can be configured, what must be in place first?
- A. A trust between your organisation Idp and AWS
  - B. The configuration of the Security Token Service (STS)
  - C. A role with associated permissions
  - D. Imported user accounts from the MS AD server to IAM
  - E. MFA activation on all IAM accounts
47. When IAM evaluates policies it follows specific logic to ascertain the permissions allowed. Which of the following statements is false?
- A. The order in which the policies are evaluated is not important
  - B. All IAM users have a default access to deny except the root account
  - C. An explicit allow overrides any default denies
  - D. An explicit deny will be overridden by an explicit allow
48. To increase the high availability of your RDS databases you decide to implement Multi-AZ for your MySQL databases. What prerequisites need to be in place before you can configure and setup Multi-AZ for these databases? (Choose 2)
- A. All I/O operations on the MySQL DB need to be stopped
  - B. The source DB must be running MySQL 5.6 or later
  - C. The retention value of the automatic backups needs to set to a value of 1 or more
  - D. You must be running the MyISAM storage engine
  - E. You must assign an IAM role to the standby DB instance

49. You are running a DynamoDB database when you start to receive 400 error codes, what could be the cause of the issue?
- A. The request has failed due to a temporary failure of the server
  - B. The request processing has failed because of an unknown error, exception or failure.
  - C. The query string contains a syntax error.
  - D. Read or write requests have exceeded the provisioned throughput for the table, therefore throttling the requests
50. Within your AWS environment, you are using IAM roles attached to EC2 instances for best practice security. A new administrator who manages your IAM controls deleted a number of IAM roles believed to be not in use. However, one of these roles was associated to some of your EC2 instances. What effect does this have on the EC2 instances that have associations with that role?
- A. The EC2 instance continues to operate as normal as the role is attached to the EC2 instance itself
  - B. The EC2 instance no longer has access to any resources that the role permitted
  - C. The EC2 instance automatically restarts in an attempt to reassociate the IAM role
  - D. The EC2 instance adopts the role of another EC2 instance in the same security group
51. Your organization has over 500 EC2 instances and in an effort to reduce the time it takes to coordinate and manage patch and maintenance updates for your instances you decide to use the Patch Manager feature from which AWS service?
- A. AWS Systems Manager
  - B. Amazon EC2
  - C. AWS Service Catalog
  - D. AWS OpsWorks

52. Can 2 or more EC2 instances be attached to the same EBS volume in the same availability zone?
- A. Only when the EC2 instances are of the same type
  - B. No, only 1 EC2 instance can attach to a single EBS volume
  - C. Only if the EC2 instances reside on the same dedicated host
  - D. No, the instances have to be in a different AZs to connect to the same EBS volume
53. You have configured an Amazon S3 bucket for website hosting to store a static website and you want to use Route53 to route traffic to that website, however, the name of the bucket does not appear in the alias target list. What could be the likely cause of this issue?
- A. You have configured the bucket in a region not supported by Route 53
  - B. The Route 53 health check has failed for the S3 bucket
  - C. The bucket name must exactly match the name of the record in Route 53
  - D. You do not have the relevant permissions to modify Route 53 recordsets
54. When using Route 53, what do you need to configure to route end users' requests to one of your application's endpoints?
- A. A Traffic Policy
  - B. A Policy Record
  - C. A Hosted Zone
  - D. Latency Based Routing (LBR)
55. You have a fleet of EC2 instances across 3 AZ's within a single region. You decide to use Auto Scaling to help balance the traffic based on a single scaling adjustment. Which scaling policy types best suits this requirement?
- A. Target Tracking Scaling
  - B. Step Scaling
  - C. Simple Scaling
  - D. Metric-based scaling

56. You have been asked by a customer to use a specific AMI when configuring their Auto Scaling solution for their EC2 fleet. In which element of Auto Scaling do you set the AMI selection?
- A. Scaling Policy
  - B. Launch Configuration
  - C. Launch Policy
  - D. Scaling Permissions
  - E. Auto Scaling Group
57. Which EC2 instance family is best suited to deliver a balance of compute, memory, and networking resources, and can be used for a variety of workloads?
- A. General Purpose
  - B. Storage Optimized
  - C. Compute Optimized
  - D. Memory Optimized
58. What component of Amazon CloudWatch allows you to monitor and troubleshoot your applications using custom log files?
- A. CloudWatch Alarms
  - B. CloudWatch Metrics
  - C. CloudWatch Graphed Metrics
  - D. CloudWatch Logs
59. What are the three different states of a CloudWatch Alarm? (Choose 3)
- A. Off
  - B. OK
  - C. On
  - D. Alarm
  - E. Insufficient\_Data
  - F. Error

60. You have just finished setting up a number of alarms within Amazon CloudWatch, however, when reviewing them you notice that the last alarm you created has a status of **INSUFFICIENT\_DATA**. What is the most likely cause of this state?
- A. The alarm was configured incorrectly and is unable to determine the result
  - B. You do not have access to view the status of the Alarm
  - C. The Alarm has just started so it doesn't have enough information to determine the state of the alarm
  - D. The metric is outside of the defined threshold
61. You have designed a new mobile app that sends notification to end users regarding leaderboard information and when new releases are becoming available. Which service can help you implement this solution?
- A. S3
  - B. EMR
  - C. Redshift
  - D. SNS
  - E. Storage Gateway
62. Your operations support team wants to be notified every time an EC2 instance is started or terminated within your Auto Scaling solution. Is this feature available within the configuration of Auto Scaling?
- A. No, Auto Scaling does not provide any means of notification
  - B. Yes, if you configure SNS within the Auto Scaling group
  - C. No, you will need to configure SNS outside of the Auto Scaling group based on AS metrics
  - D. Yes, this is configured by default for all Auto Scaling groups
63. As a part of your organization's disaster recovery and business continuity you are looking for a solution that will backup your on-premise data to AWS while ensuring the same data remains on your local storage volumes within your data center for low latency access. You look at the Storage Gateway service as a solution, which option would be best suited for your requirements?
- A. Cached volume gateways
  - B. Gateway VTL
  - C. Stored Volume Gateways
  - D. File Gateway

64. By default, CloudTrail logs that are stored on S3 are encrypted. Which method of encryption is selected by default?
- A. SSE-KMS
  - B. CSE-KMS
  - C. SSE-S3
  - D. SSE-C
  - E. CSE-C
65. Which of the following is NOT a regional component of the AWS global infrastructure?
- A. Edge Location
  - B. Availability Zone
  - C. Region
  - D. Landing Zone
  - E. Regional Edge Caches

# Assessment

## Mock Test 1

1. B

High I/O instances are storage optimized. Examples of this family are H1 (high disk throughput), I3 (high random I/O performance), and D2 (dense storage). EBS-optimized instances have exclusive capacity for I/O operations.

RDS is not compatible with Oracle Enterprise High Availability options because it uses Multi-AZ and access to the OS is limited. Also, note that burstable performance instances are not compatible with EBS optimizations.

This question is focused on the portability of current features and performance only. Costs or managed services are not mentioned in the question.



### References:

- **Amazon EC2 Instance Types:** <https://aws.amazon.com/ec2/instance-types/>
- **Amazon RDS Instance Types:** <https://aws.amazon.com/rds/instance-types/>
- **Amazon RDS for Oracle Database FAQs:** <https://aws.amazon.com/rds/oracle/faqs/>
- **Instances optimized for Amazon EBS:** [https://docs.aws.amazon.com/es\\_es/AWSEC2/latest/UserGuide/EBSOptimized.html](https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/EBSOptimized.html)
- **Determining the IOPS needs for Oracle Database on AWS:** <https://d1.awsstatic.com/whitepapers/determining-iops-needs-for-oracle-database-on-aws.pdf>

2. B

Short polling is the default behavior of an SQS queue (`ReceiveMessageWaitTimeSeconds=0`). Any value greater than zero will enable long polling and less CPU overhead. This can also be overridden at `ReceiveMessage` by increasing the `WaitTimeSeconds` attribute. Extending the visibility timeout will only avoid double processing of messages by multiple workers.

SQS FIFO queues can only handle a throughput of 300 messages per second (soft limit) and do not accept duplicates; duplicate messages are allowed in this design because there is no restriction in the question, standard `.fifo` queue migration is not possible. Since the message volume per second is higher, we are talking about standard SQS queues.



A and B are correct answers, but the question is focused on the simplest way of achieving long polling; option A will require the reconfiguration of every worker client, while option B only requires a change at the queue level so all clients will use this configuration. B is the simplest way.

**References:**

- **Amazon SQS FAQs:** <https://aws.amazon.com/sqs/faqs/>
- **Amazon SQS FIFO (First-In-First-Out) queues:** <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/FIFO-queues.html>
- **Amazon SQS Long Polling:** <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-long-polling.html>
- **Amazon SQS Visibility Timeout:** <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-visibility-timeout.html>

3. C

S3 Standard provides the highest durability by default and it replicates objects in three AZs; FS is the most critical information. CS is the derived data and in the case of data loss, it can be recreated again, which is a low-cost option.

RS requires FS and CS files to Glacier is not an option for RS because it doesn't provide frequent access to data. Objects in Glacier Standard must reside for at least 60 days in S3 Standard.

This question is focused on durability and economics; trade-offs must be made to achieve low costs, durability, and resiliency of data.



### References:

- **Amazon S3 Storage Classes:** <https://aws.amazon.com/s3/storage-classes/>
- **Object Lifecycle Management:** <https://docs.aws.amazon.com/AmazonS3/latest/dev/object-lifecycle-mgmt.html>
- **Amazon S3 FAQs:** <https://aws.amazon.com/s3/faqs/>
- **Amazon S3 Reduced Redundancy Storage:** <https://aws.amazon.com/s3/reduced-redundancy/>

#### 4. C

Option A could work, but it is not scalable, since bucket policies have a size limit of 20 KB. CloudFront URLs will improve the performance of the web app but does not guarantee confidentiality because the resource is still public and static.

Signed URLs will allow an expiration time that can match the web session of the user, and accelerating through CloudFront will solve the performance improvements required but no confidentiality. CloudFront signed cookies will enable authentication and authorization for the platform users.

### References:

- **Uploading Objects Using Presigned URLs:** <https://docs.aws.amazon.com/AmazonS3/latest/dev/PresignedUrlUploadObject.html>
- **Choosing Between Signed URLs and Signed Cookies:** <https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-choosing-signed-urls-cookies.html>
- **IAM Policies, Bucket Policies, and ACLs!:** <https://aws.amazon.com/blogs/security/iam-policies-and-bucket-policies-and-acls-oh-my-controlling-access-to-s3-resources/>

5. C

Option A is not the best fit because the load balancer only acts as a proxy service for balancing and resilience capabilities; this is a horizontally scaled service, so doesn't represent a bottleneck.

Option B will have the opposite effect: dual-homed servers will result in bandwidth bisection.

Option C is the right answer, since a NAT Gateway is a horizontally scaled service and the NAT instance is limited by the instance networking capabilities; multiple NAT instances, balancing, and scripted failover can be architected but this is not the best option because a managed service can be used offloading all the ops.

Option D is a good option but solves only part of the problems and works in the short term.

**References:**

- **NAT Instances:** [https://docs.aws.amazon.com/vpc/latest/userguide/VPC\\_NAT\\_Instance.html](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_NAT_Instance.html)
- **Comparison of NAT Instances and NAT Gateways:** <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-comparison.html>
- **Elastic Network Interfaces:** <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html>
- **Enhanced Networking on Linux:** <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking.html>

6. D

Regarding option A, S3 One Zone Infrequent Access is not a good option because the redundancy level is low and it could lead to a total loss of data because the AZ comprehends a single fault domain. In option B, Glacier by definition is not a good choice for DR because it doesn't give you real-time access to your data. This has to be considered from an RTO perspective.

Option C is the same as Option B. Option D is the best option because several backup services are mentioned as EBS Snapshots, Storage Gateway and S3 with 99.99999999% durability and 99.99% availability.

### References:

- **AWS Storage Gateway FAQs:** [https://aws.amazon.com/storagegateway/faqs/?nc1=h\\_ls](https://aws.amazon.com/storagegateway/faqs/?nc1=h_ls)
- **Configuring DNS Failover:** <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/dns-failover-configuring.html>
- **AWS Direct Connect Resiliency Recommendations:** <https://aws.amazon.com/directconnect/resiliency-recommendation/>

## 7. C

Option A: SSE- KMS gives you the flexibility to choose your encryption keys generated via KMS, but bucket ACLs don't give the option to use conditions.

Option B: SSE - S3 is the right answer because it offloads all the key management to KMS via S3. The bucket policy will reinforce to use a StringNotEquals condition of "s3:x-amz-server-side-encryption": "AES256" denying Put operations and versioning to prevent accidental deletion.

Option C: Object lifecycle does not enforce security or security management.

Option D: This is a way to only provide confidentiality and integrity to the data from the customer side taking full responsibility of the process

### References:

- **PCI DSS:** <https://aws.amazon.com/compliance/pci-dss-level-1-faqs/>
- **How to Prevent Uploads of Unencrypted Objects to Amazon S3:** <https://aws.amazon.com/blogs/security/how-to-prevent-uploads-of-unencrypted-objects-to-amazon-s3/>
- **Using Versioning:** <https://docs.aws.amazon.com/AmazonS3/latest/dev/Versioning.html>
- **Protecting Data Using Encryption:** <https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingEncryption.html>

## 8. D

Option A: The AWS CLI can solve the problem but it doesn't scale to petabytes to achieve the transfer in one week. Option B: The VM Import/Export is designed to work with virtual images and storage gateway and has a different use case; it is not for one-time-only data transfer.

Option C: Direct Connect can do the job, but the 1 Gbps has a low bandwidth to finish in one week. Option D: This is the correct choice, as several Snowballs can be used in parallel to copy the petabyte and achieve the full job in one week.

### References:

- **Multipart Upload Overview:** <https://docs.aws.amazon.com/AmazonS3/latest/dev/mpuoverview.html>
- **FAQs about AWS Direct Connect:** <https://aws.amazon.com/es/directconnect/faqs/>
- **How to Transfer Petabytes of Data Efficiently:** <https://docs.aws.amazon.com/snowball/latest/ug/transfer-petabytes.html>

## 9. C

Option A: Read replicas perform asynchronous replication, so eventual consistency and replication lag are common scenarios.

Option B: The Multi-AZ deployment is designed to provide 99.95% availability. RDS can detect common hardware failures and perform failovers to provide business continuity.

Option C: RDS cannot be provisioned with less than 100 GB of SSD.

Option D: Transparent Data Encryption can be enabled only for SQL Server and Oracle Databases.

### References:

- **Amazon RDS Multi-AZ Deployments:** [https://aws.amazon.com/rds/details/multi-az/?nc1=h\\_ls](https://aws.amazon.com/rds/details/multi-az/?nc1=h_ls)
- **Amazon RDS Read Replicas:** <https://aws.amazon.com/rds/details/read-relicas/>
- **Limits for Amazon RDS:** [https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP\\_Limits.html](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Limits.html)

- **Microsoft SQL Server Transparent Data Encryption Support:** <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.SQLServer.Options.TDE.html>
- **Oracle Transparent Data Encryption:** <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.Oracle.Options.AdvSecurity.html>

## 10. C

Option A: RAID 5 cannot be used in EC2 because it uses parity and it consumes IOPS available.

Option B: RAID 1 is a good option but the array will be limited to the worst performing volume.

Option C: EBS snapshots are the best option to provide fault tolerance and performance. EBS automatically replicates every block operation at the AZ level.

Option D: RAID 0 is designed to improve performance at the cost of fault tolerance.

### References:

- **RAID Configuration on Linux:** <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/raid-config.html>
- **Automating the Creation of Consistent Amazon EBS Snapshots with Amazon EC2 Systems Manager (Part 1):** <https://aws.amazon.com/blogs/compute/automating-the-creation-of-consistent-amazon-ebs-snapshots-with-amazon-ec2-systems-manager-part-1/>
- **Amazon EBS FAQs:** [https://aws.amazon.com/ebs/faqs/?nc1=h\\_ls](https://aws.amazon.com/ebs/faqs/?nc1=h_ls)

## 11. C

Option A: The classic load balancer can work but it doesn't provide routing flexibility.

Option B: The network load balancer works on the Layer 4 of the OSI model, this is not compatible with HTTP.

Option C: The application load balancer is the perfect fit for this scenario because it provides control to route requests to containers using ports and can be enabled for sticky sessions to maintain affinity between the client and the server.

Option D: The requirement doesn't specify routing at the global level so Route 53 is not necessary.

**References:**

- **Load Balancer Types:** <https://docs.aws.amazon.com/AmazonECS/latest/developerguide/load-balancer-types.html>
- **Configure Sticky Sessions for Your Classic Load Balancer:** <https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-sticky-sessions.html>
- **Service Load Balancing:** <https://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-load-balancing.html>

12. C

Option A: The application script can be modified and used to access other users' data.

Option B: The RDS table does not store the data in an encrypted format and can be vulnerable.

Option C: The instance metadata server is a great choice because it provides confidentiality and automation in the process. The metadata server can only be reached from inside the instance.

Option D: Email can be compromised, exposing sensitive data.

**References:**

- **Instance Metadata and User Data:** <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html>
- **Encrypting Amazon RDS Resources:** <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html>

13. A

Option A: This is the best choice because CloudTrail maintains auditability and AWS Config Rules allow you to use Lambda functions to automate events.

Option B: CloudWatch events with filters are not the most scalable and effortless option.

Option C: A Marketplace instance is not scalable because it stores audit trails and system changes in the local disk.

Option D: CloudTrail is only half of the job and CloudTrail does not provide an API.

**References:**

- **Best Practices for Monitoring:** [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/monitoring\\_best\\_practices.html](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/monitoring_best_practices.html)
- **AWS Config Rules – Dynamic Compliance Checking for Cloud Resources:** <https://aws.amazon.com/blogs/aws/aws-config-rules-dynamic-compliance-checking-for-cloud-resources/>
- **AWS Config vs. CloudTrail:** <https://www.sumologic.com/blog/amazon-web-services/aws-config-vs-cloudtrail/>

**14. A**

Option A: RDS is a container service and privileged access to the operating system is limited.

Option B: EMR allows users to SSH into the master instance to manage the core and task nodes in the cluster.

Option C: EC2 is an Infrastructure as a Service resource and SSH access is permitted.

Option D: Elastic Beanstalk is a container service but SSH access is permitted.

**References:**

- **Amazon RDS FAQs:** <https://aws.amazon.com/rds/faqs/>
- **Working with Option Groups:** [https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER\\_WorkingWithOptionGroups.html](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_WorkingWithOptionGroups.html)
- **Working with DB Parameter Groups:** [https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER\\_WorkingWithParamGroups.html](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_WorkingWithParamGroups.html)
- **Connect to the Master Node Using SSH:** <https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-connect-master-node-ssh.html>
- **eb ssh:** <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb3-ssh.html>

**15. A, B, and C**

**References:**

- **Overview of Security Processes:** <https://aws.amazon.com/whitepapers/overview-of-security-processes/>
- **Shared Responsibility Model:** <https://aws.amazon.com/compliance/shared-responsibility-model/>

16. C

Option A: SNS is used to deliver notifications but S3 file copy is not enabled for Glacier.

Option B: When data is aggregated to vaults in Glacier, notification is not available.

Option C: Vault Inventory and Retrieval Job Complete are the only available SNS notifications from Glacier.

**References:**

- **Configuring Vault Notifications in Amazon Glacier:** <https://docs.aws.amazon.com/amazonglacier/latest/dev/configuring-notifications.html>
- **Amazon Glacier FAQs:** <https://aws.amazon.com/glacier/faqs/>

17. A, B, and C

Option D: The VPC is not highly available in nature; it is the customer's responsibility to architect for availability using multiple AZs.

**References:**

- **VPCs and Subnets:** [https://docs.aws.amazon.com/vpc/latest/userguide/VPC\\_Subnets.html#vpc-sizing-ipv4](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Subnets.html#vpc-sizing-ipv4)
- **Amazon VPC FAQs:** <https://aws.amazon.com/vpc/faqs/>
- **IP Addressing in Your VPC:** <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-ip-addressing.html>

18. C

Option A: Placement Groups are a great way to achieve maximum packets per second between instances.

Option B: Jumbo frames increase the payload of every message.

Option C: Spot instances have more compute power but do not increase network throughput.

Option D: Enhanced networking will improve the instance capabilities by using Virtual Function or **Elastic Network Interfaces (ENA)**.

**References:**

- **Network Maximum Transmission Unit (MTU) for Your EC2 Instance:** [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/network\\_mtu.html](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/network_mtu.html)
- **Placement Groups:** <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html>
- **Spot Instances:** <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html>
- **Enabling Enhanced Networking with the Elastic Network Adapter (ENA) on Linux Instances:** <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking-ena.html>

**19. D**

Option A: Using RDS will solve the problem temporarily and couple the application.

Option B: Compression solves only part of the problem now; in the future, a new design must be proposed.

Option C: The problem is not on DynamoDB so this option is useless.

Option D: SQS provides functionality to store the blob message in S3; this is a great solution to store even bigger messages in the future.

**References:**

- **Managing Large Amazon SQS Messages Using Amazon S3:** <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-s3-messages.html>
- **Limits in DynamoDB:** <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Limits.html>
- **AWS Storage Options:** <https://aws.amazon.com/whitepapers/storage-options-aws-cloud/>

**20. B**

Option A: ElastiCache requires additional complexity in order to fine-tune the hotkeys and requires application re-engineering.

Option B: CloudFront is the less intrusive solution, creating a distribution with cache behaviors and TTL will do the job.

Option C: At some point, the database size will be a constraint and it won't scale to meet demand.

Option D: Read replicas will require application re-engineering.

### References:

- **Amazon RDS Read Replicas:** <https://aws.amazon.com/rds/details/read-replicas/>
- **Overview of Distributions:** <https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/distribution-overview.html>
- **Amazon ElastiCache FAQs:** <https://aws.amazon.com/elasticache/faqs/>

## Mock Test 2

1. B

Versioning stores a separate copy of every object on S3 whenever a change is made to the object. As a result, additional storage is required for multiple versions of the same object which will, in turn, increase S3 costs

2. C

3. A and C

To create a public subnet you must create an IGW, attach it to your Public subnet and configure your route tables accordingly to point to the internet via your IGW.

4. D

RTO is defined as the maximum amount of time in which a service can remain unavailable before it's classed as damaging to the business. The smaller the RTO the more it generally costs in preparation to achieve by having additional resources running ready to take the load in the event of a failure.

5. B

CSE-KMS—client-side encryption with KMS managed keys. This option uses keys managed by AWS using the KMS service, but also ensures that the encryption of the object takes place on the client prior to upload to S3.

6. D

Out of the available options, multi-site has the lowest RTO as it has a replication of production resources readily available and running in the event of an incident. Options A, B, and C will all take a longer time to gain a fully resumed service.

7. A and E

An explicit deny will always overrule an allow, access or leave privilege will be granted during evaluation logic. Due to the associated security risks, access to all resources are denied by default until the administrator grants access to services.

8. B

AWS CloudTrail is used to record and track API calls made within your AWS account which are sent to a CloudTrail Log. This log data contains metadata about the API call which includes the identity who initiated the API call as well timestamp information.

9. C

Your ELB does not perform any actions that result in the starting/stopping of EC2 instances. If Auto Scaling is configured, then Auto Scaling will be responsible for managing the fleet size and adding new instances if needed. Your ELB will simply stop sending any requests to the unhealthy instance.

10. B

Vault Lock Policies are similar in configuration to Vault Access Policies, the difference being is that once they are set, they cannot be changed. This helps to maintain compliance to specific governance controls.

11. D

12. A and D

Communication between resources within a VPC is usually resolved by reviewing the security groups which control access at an instance level, and NACLs which control access at the subnet level.

13. B, C, and D

Using the following URL you can see a table from AWS that compares the differences between AWS Instances and AWS Gateways: <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-comparison.html>.

14. A

When Multi-AZ is configured a secondary RDS instance, known as a replica is deployed within a different availability zone within the same Region as the primary instance. That it's single and only purpose, to provide a failover option for a primary RDS instance.

15. B, C, and D

16. B

Key pairs consist of a public key and private key, the public key is maintained by AWS and the Private key must be stored and kept by you to allow you to remotely and securely connect to the newly created instance.

17. B and D

Restricted tenancy options for AWS include both dedicated hosts and dedicated instances. Dedicated hosts offer additional administrative functionality than dedicated instances, however, both ensure that no other customer will be running on the same underlying host.

18. A, B, and C

19. A, D, and E

| Security Group                                                                | Network ACL                                                             |
|-------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| Operates at the instance level                                                | Operates at the subnet level                                            |
| Supports allow rules only                                                     | Supports allow rules and deny rules                                     |
| Is stateful: Return traffic is automatically allowed, regardless of any rules | Is stateless: Return traffic must be explicitly allowed by rules        |
| We evaluate all rules before deciding whether to allow traffic                | We process rules in number order when deciding whether to allow traffic |

20. C

System status checks are usually caused by an underlying host issue. Typically by a loss of network connectivity, power, software, or hardware errors on the underlying host. By stopping and restarting the instance, the instance will likely launch on a different host, thereby eliminating the failed underlying host.

21. B

If you are serving your content via an S3 bucket then it will typically have read access to the public to allow anyone to retrieve the content. This means that if someone knows the URL of object(s) then they can access that content directly, bypassing CloudFront and any restrictions that you may have in place. As a result, you can restrict access to the entire bucket by limiting access to a single user, an Origin Access Identity. This identity is a particular CloudFront user which is associated with your distribution as the sole identity who can retrieve content. This ensures that content can ONLY be served via CloudFront.

22. D

By default memory utilization and swap usage is not tracked by CloudWatch. To monitor these additional metrics you need to run specific scripts on your instances to push this data to CloudWatch.

23. B

When someone subscribes to an SNS topic via the email endpoint, the recipient has to accept the subscription from their email address before they will start to receive notifications. This prevents administrators from simply adding subscribers without their consent.

24. C

DynamoDB is designed to deliver high performance at scale, single digit latency, in-memory caching and security all as a managed service. The other services wouldn't be able to offer the requirements as well as DynamoDB. Redshift is a data warehouse, RDS is a relational database and EMR is used for Big Data processing, so again, not a good fit for purpose.

25. B and C

When using Auto Scaling with the right sizes instances to minimize resource wastage you can significantly reduce your EC2 costs to match the demand of traffic to your fleet size. This helps to ensure that you only pay for what you use.

26. A

KMS Grants simply allows users of the CMK, and by users I mean those who can perform encryption/decryption operations, to delegate their own permissions, or at least a subset of their permissions to another principal. Similarly to Key Policies, Grants are also a resource-based access control method to the CMK.

27. D and E

28. B, C, and E

|                                                  | Memcached | Redis |
|--------------------------------------------------|-----------|-------|
| Sub-millisecond latency                          | Yes       | Yes   |
| Developer ease of use                            | Yes       | Yes   |
| Data partitioning                                | Yes       | Yes   |
| Support for a broad set of programming languages | Yes       | Yes   |
| Advanced data structures                         | -         | Yes   |
| Multithreaded architecture                       | Yes       | -     |
| Snapshots                                        | -         | Yes   |
| Replication                                      | -         | Yes   |
| Transactions                                     | -         | Yes   |
| Pub/Sub                                          | -         | Yes   |
| Lua scripting                                    | -         | Yes   |
| Geospatial support                               | -         | Yes   |

29. A, D, and E

If message ordering is essential you should use FIFO, First In First Out queues. They ensure the order of messages is maintained and that there is no duplication of messages ensuring only-once processing. FIFO queues have a limited number TPS set at 300 whereas Standard queues do not.

30. C

CloudFormation Supported AWS Services: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-supported-resources.html>.

31. A and C

A dead-letter queue is used by the source queue to send messages that fail to be processed. This could be the result of code within your application, corruption within the message or simply missing information within a database that the message data relates to. Either way, if the message can't be processed by a consumer after a maximum number of tries specified, the queue will send the message to a dead-letter queue. This allows engineers to assess why the message failed to identify where the issue is to help prevent further messages falling into the DLQ.

32. D

The S3 bucket has a conditional element which means access specified within the effect, principle, effect, and resource are only applicable if the conditions are met. This condition allows all IP addresses within the 54.240.143.0/24 range but NOT 54.240.143.188/32.

33. B

Only **Network Access Control Lists (NACLs)** allows you to explicitly deny traffic. Within a security group, traffic is allowed based on the rules that are within it. If traffic is assessed by a security group and there is no rule matching it then the traffic is dropped without a **Deny** action required.

34. C and E

If the IP address range of 2 VPCs overlaps then it is not possible to have them configured as peers. This is something to bear in mind when designing your VPCs with CIDR blocks. Once you have peered your VPCs together, you must update any security groups associated to resources that you want to communicate within the peered VPC (<https://docs.aws.amazon.com/vpc/latest/peering/vpc-peering-security-groups.html>).

35. A

Access keys are used to allow programmatic access to IAM users. This allows them to manage infrastructure and run commands from the command line instead of using the management console. The access keys are used as the authentication method when communicating with AWS.

36. D

To authenticate you as a user when using the AWS CLI you need to start by configuring the AWS CLI using the AWS Configure command. This allows you to set information such as your access keys for authentication, default region and output format of the CLI.

37. D

AWS CloudTrail will typically deliver log files within 15 minutes of when an API is called, in addition to publishing new log files about every 5 minutes.

38. C  
AWS config can use managed rules to help evaluate your resources across a range of categories, including Compute, Storage, Database, Security and more. These rules evaluate your resources to ensure they are meeting compliance as defined by the rule type. To fulfill the needs of this scenario you would use the managed rules of: ENCRYPTED\_VOLUMES and DESIRED\_INSTANCE\_TYPE.
39. C  
When configuring your customer gateway, it's important you use a publicly accessible IP address. Private IP addresses will not work.
40. A  
To allow IAM administration of access control to CMKs the root account needs full KMS access to the CMK.
41. B  
EC2 instances associated with an ELB can reside in different availability zones and therefore different subnets. In fact, this is the best practice to allow for resiliency in the event of an AZ failure.
42. A, B, and C  
When setting up your auto scaling configuration you need to define the parameters of your scaling operations with the scaling policy along with the instance types to be used within the Launch configuration and the logical grouping of your instance with the scaling group.
43. C  
EBS Volumes can't be moved between availability zones. As a result, you need to create a snapshot of the volume first which you can then use to create a new EBS volume in a different AZ. This allows you to copy data between AZs from EBS volumes.
44. B, D, and E  
[https://aws.amazon.com/elasticloadbalancing/features/#Details\\_for\\_Elastic\\_Load\\_Balancing\\_Products](https://aws.amazon.com/elasticloadbalancing/features/#Details_for_Elastic_Load_Balancing_Products)
45. B  
By default, you can have up to 20 reserved Elasticache nodes.
46. A  
When using SAML 2.0 based federation you need to ensure that a trust exists between your organization's IdP and your AWS account. Without this trust in place, federated access will not be allowed.
47. D  
An explicit deny will ALWAYS override any allow.

48. B and C

You must be running at least MySQL 5.6 and have your backup value set to 1 or more for the creation of a standby instance in a Multi-AZ configuration. You must also be running InnoDB as a storage engine rather than MyISAM.

49. D

DynamoDB common errors: <https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/CommonErrors.html>.

50. B

When an IAM role is deleted, any identities associated with the role no longer have the permissions that were assigned to the role. In this case, all permissions that were associated with the role are removed from the EC2 instance.

51. A

AWS Systems Manager Patch Manager automates the process of patching managed instances with security-related updates.

52. B

Only a single EC2 instance can ever connect to a single EBS volume, however, multiple EBS volumes can be attached to a single EC2 instance.

53. C

<https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/troubleshooting-s3-bucket-website-hosting.html>

54. A

A traffic policy is the set of rules that you define to route end users' requests to one of your application's endpoints. Traffic policies can be JSON text files and uploaded using the Route 53 API, the AWS CLI, or the various AWS SDKs.

55. C

<https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-scale-based-on-demand.html#as-scaling-types>

56. B

The launch configuration is used to set the option for the EC2 instances that are added to an auto scaling group, this includes the AMI and instance type and so on.

57. A

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html>

58. D

Amazon CloudWatch logs allow you to import log data from other systems, services and application into CloudWatch allowing you to monitor against its data and set alarms.

59. B, D, and E

Amazon CloudWatch has 3 different alarm states to let you know the status of each one configured, these being Alarm, OK and insufficient.

60. C

CloudWatch Alarms have 3 different states:

- **OK:** The metric is within the defined threshold
- **ALARM:** The metric is outside of the defined threshold
- **INSUFFICIENT\_DATA:** The metric is not available or there is not enough data is available for the metric to determine the alarm state

61. D

The Simple Notification Service would be best to help implement this solution.

62. B

You can now instruct Auto Scaling to send a notification when it launches or terminates an EC2 instance. There are actually four separate notifications:

EC2\_INSTANCE\_LAUNCH, EC2\_INSTANCE\_LAUNCH\_ERROR,  
EC2\_INSTANCE\_TERMINATE, and EC2\_INSTANCE\_TERMINATE\_ERROR. More information can be found here: <https://aws.amazon.com/blogs/aws/auto-scaling-notifications-recurrence-and-more-control/>.

63. C

Stored volume gateways are often used as a way to backup your local storage volumes to Amazon S3 whilst ensuring your entire data library also remains locally on-premise for very low latency data access. Volumes created and configured within the storage gateway are backed by Amazon S3 and are mounted as iSCSI devices that your applications can then communicate with.

64. C

S3 uses its own built in encryption of SSE-S3 to encrypt CloudTrail log files. However, you can select to have the logs encrypted with KMS, using SSE-KMS.

65. D

<https://cloudacademy.com/blog/aws-global-infrastructure/>

# Another Book You May Enjoy

If you enjoyed this book, you may be interested in another book by Packt:



## AWS Certified Developer - Associate Guide

Vipul Tankariya, Bhavin Parmar

ISBN: 978-1-78712-562-9

- Create and manage users, groups, and permissions using AWS Identity and Access Management services
- Create a secured Virtual Private Cloud (VPC) with Public and Private Subnets, Network Access Control, and Security groups
- Get started with Elastic Compute Cloud (EC2), launching your first EC2 instance, and working with it
- Handle application traffic with Elastic Load Balancing (ELB) and monitor AWS resources with CloudWatch
- Work with AWS storage services such as Simple Storage Service (S3), Glacier, and CloudFront
- Get acquainted with AWS DynamoDB – a NoSQL database service
- Coordinate work across distributed application components using Simple Workflow Service (SWF)

## **Leave a review - let other readers know what you think**

Please share your thoughts on this book with others by leaving a review on the site that you bought it from. If you purchased the book from Amazon, please leave us an honest review on this book's Amazon page. This is vital so that other potential readers can see and use your unbiased opinion to make purchasing decisions, we can understand what our customers think about our products, and our authors can see your feedback on the title that they have worked with Packt to create. It will only take a few minutes of your time, but is valuable to other potential customers, our authors, and Packt. Thank you!

# Index

## A

- access control 331
- access key 337
- ACLs
  - replicating 81, 82, 83
- AD federated services (ADFS) 345
- administration group, IAM
  - business case 47, 48, 50, 51, 52, 53
- Advanced Encryption Standard (AES) 356
- Advanced Encryption Standard (AES-256)
  - algorithm 359
- Amazon CloudFront
  - about 69
  - security features 462, 463
- Amazon Glacier
  - about 257
  - retrieval options 257
  - workflow 258, 259
- Amazon GuardDuty 443
- Amazon Inspector agent
  - installing 397, 398
- Amazon Inspector
  - about 395, 443
  - assessment run 399
  - assessment templates 398
  - findings 399
- Amazon Machine Image (AMI) 114
- Amazon Macie 443
- Amazon Resource Name (ARN) 348, 385
- Amazon S3 encryption
  - about 363, 364
  - client-side encryption with KMS managed keys (CSE-C) 373
  - client-side encryption with KMS managed keys (CSE-KMS) 371, 372
  - server-side encryption with customer managed keys (SSE-C) 368, 369, 370
  - server-side encryption with KMS managed keys (SSE-KMS) 366, 367
  - server-side encryption with S3 managed keys (SSE-S3) 364, 365
- Amazon Web Services (AWS)
  - about 7, 65
  - cloud-based architectures, designing 10, 11
- Apache Hive 316
- API Gateway integration
  - about 208, 209, 210, 211, 213, 214
  - request flow 215, 216
- Application Load Balancer (ALB)
  - about 278, 447
  - creating 280, 281, 282, 283
- application programming interface (API) 389
- application servers 8
- AppSpec file 527, 529, 531, 535
- architecture 7
- Artifact 444
- asymmetric key cryptography 357
- Atom 488
- authentication 330
  - about 332
  - cross-account roles 342
  - federated access 343
  - IAM roles 341
  - key pairs 338, 340
  - multi-factor authentication 332, 334, 335, 336
  - passwords 332
  - programmatic access 337, 338
  - SAML 2.0 federation 344, 345
  - SAML federation 343
  - usernames 332
  - web identity 343
  - web identity federation 343, 344

authorization 331  
availability metrics  
  about 222  
  business impact analysis 223  
  business perspective 223  
availability monitoring 225  
Availability Zone (AZ) 66, 301  
AWS AppSync 219  
AWS Auto Scaling  
  about 286, 287  
  alternate flow 290  
  groups 293, 294, 295  
  launch configurations, creating 291  
  resiliency 296, 297  
AWS CloudFormation  
  about 488, 489, 490  
  deletion policy 494  
  dependencies 502  
  outputs 494  
  stack updates 492, 493  
  template anatomy 491  
  template reusability 495  
AWS CloudTrail  
  about 389, 390  
  configuring 390, 391, 392, 394, 395  
  reference 389  
AWS CodeDeploy  
  about 524, 526  
  AppSpec file 527, 529, 531, 535  
AWS Config  
  about 416, 444  
  Config rules 421  
  configuration history 419  
  Configuration Item (CI) 418  
  configuration recorder 420  
  configuration snapshot 420  
  configuration streams 419  
  high-level process overview 422  
  resource relationship 421  
AWS Direct Connect 68  
AWS Firewall Manager  
  about 460  
  pre usage 461  
AWS global infrastructure  
  10,000-feet view 66  
  about 64  
  data centers 65, 66  
  service company, becoming 64, 65  
AWS IoT 219  
AWS Kinesis 183  
AWS Lambda 323, 324, 326, 327  
AWS Organizations 482, 483, 485  
AWS Regions  
  100,000-feet view 67  
  about 67  
  compliance 67  
  connectivity 68  
  cost 68  
  endpoint access 68  
  latency 67  
  supported services 68  
AWS Schema Conversion tool 164  
AWS security best practices  
  about 424  
  data protection 429  
  EC2 security 439  
  Identity and Access Management 435  
  security services 442  
  shared responsibility model 425, 426, 427  
  technical requirement 425  
  Virtual Private Cloud (VPC) 434  
AWS Security Token Service (STS) 345  
AWS Shield  
  about 458  
  shield plans 459  
  used, for mitigating against DDoS attacks 459  
AWS Single Sign-on 443  
AWS Systems Manager  
  about 405  
  actions 411, 413  
  insights 413, 414  
  resource groups 408  
  shared resource 415  
AWS Trusted Advisor  
  about 400, 401, 402  
  red warning under service limits 404, 405  
  yellow warning under service limits 402, 404  
AWS Web Application Firewall (WAF)  
  about 444, 447  
  conditions 447

rules 454, 455  
AWS Well-Architected Framework 30

## B

backup and restore  
about 225  
disaster scenario 226, 227  
preparation phase 226  
trade-offs 227  
bastion host 157, 158  
**Billing & Cost Management Dashboard**  
about 467, 468  
billing alarms 469  
billing reports 474, 475, 477  
**Cost Allocation Tags** 482  
**Cost Explorer** 477  
QuickSight visualization 479, 480, 481  
**Reserved Instances recommendations** 478  
service level alarms 470, 471, 472, 473, 474  
bootstrapping 132, 134  
Border Gateway Protocol (BGP) 161  
bucket  
configuring 72, 75, 76, 77, 78  
business impact analysis  
availability monitoring 225  
Recovery Point Objective (RPO) 224  
Recovery Time Objective (RTO) 224

## C

**CAF business perspectives**  
business perspective 29  
governance perspective 29  
people perspective 29  
**CAF technical perspectives**  
operations perspective 30  
platform perspective 29  
security perspective 29  
cascading deletion 131, 132  
Center for Internet Security (CIS) Benchmarks 396  
Certificate Manager 443  
CfnCluster  
about 302, 303  
reference 305  
CI/CD pipeline 521, 523  
ciphertext 355

Classic Load Balancer (CLB) 275  
Cloud Adoption Framework  
about 29  
perspectives 29  
cloud computing 10  
cloud design patterns  
about 28  
reference 28  
cloud design principles 11, 12, 13, 14, 15, 16, 18, 20, 22, 26, 27, 28  
Cloud Hardware Security Module (HSM) 443  
CloudFront  
about 71  
static website, hosting 91, 92, 93, 94, 95, 96, 98, 99, 100, 101, 102, 103, 104, 105  
CloudTrail  
about 38, 444  
enabling 38  
CloudWatch Logs Agent 511, 513, 514  
CloudWatch Metric Filters 515, 516, 518, 519  
cluster HPC 301  
clustering 301  
clusters, in AWS  
cluster HPC 301  
distributed grid 302  
high performance computing (HPC) 302  
Cognito user pools 205  
Cognito  
about 442  
mobile apps, authenticating 204  
web, authenticating 204  
Command Query Responsibility Segregation (CQRS) 234  
Common Vulnerabilities and Exposures (CVE) 396  
communication model, SNS  
Fanout 200, 202, 203  
subscriber 199, 200  
communications  
managing, with SNS 198  
complexity  
minimizing 8  
composite primary key 265  
compute layer  
scaling 135, 136  
conditions, WAF

cross-site scripting (XSS) 448  
geo match 449  
IP addresses 450  
size constraints 451  
SQL injection attacks 452  
string and regex matching 453  
confidentiality, integrity, and availability (CIA) 22  
configuration management database (CMDB) 13  
container services 427  
Content Delivery Network (CDN) 69  
content delivery network (CDN) 63  
Continuous Deployment (CD) 521, 523  
Continuous Integration (CI) 521, 523  
control plane, DynamoDB 260, 261, 262, 263, 265, 266  
Convertible Reserved Instances 466  
Conway's law 9  
cookies, ELB documentation  
reference 285  
cross-account roles 342  
cross-region replication (CRR) 71  
CSE-C 372, 373  
CSE-KMS 371  
customer gateway 159  
customer master key (CMK) 366, 382  
customer premises equipment (CPE) 159

## D

data access objects (DAOs) 263  
data center  
extending 142  
data encryption keys (DEK) 383  
data protection  
about 429  
encryption features, using 430, 431  
encryption, using at rest for sensitive data 429  
encryption, using in transit for sensitive data 431  
protection, against unexpected data loss 432  
S3 lifecycle policies, using 433  
S3 MFA delete, using for preventing accidental deletion 433  
S3 versioning implementation, for protection against unintended actions 433  
data redundancy  
with managed services 71

data replication  
with managed services 71  
Data Tier 8  
database management systems (DBMS) 8  
Database Migration Service  
about 163  
AWS Schema Conversion tool 164  
heterogeneous migrations 165  
homogeneous migration 164  
database server  
scaling 136, 137, 138  
Direct Attached Storage (DAS) 125  
Direct Connect (DX) 161  
Direct Connect Gateway 68  
directory service 443  
disaster recovery 127, 129, 130  
Distributed Denial of Service (DDoS) attack 26, 444  
distributed memory-caching systems 8  
divide and conquer approach 110  
durability, AWS S3  
about 248  
limited durability 248  
maximum durability 248  
dynamic hardware VPNs 161  
DynamoDB Streams 261  
DynamoDB  
about 260  
control plane 260

## E

EBS encryption  
about 357, 358  
existing EBS volume, encrypting 360, 361, 362, 363  
new EBS volume, encrypting 358, 359  
new EBS volume, encrypting during launch of new EC2 360  
EC2 persistence model 125, 126  
EC2 security  
about 439  
Bastion hosts, using 441  
operating system, hardening 441  
patching strategy, implementing 440  
security groups access, controlling 440

sensitive data, encrypting on persistent storage 440

**E**

Edge Locations 69

Elastic Block Store (EBS) service 357

Elastic Compute Cloud (EC2) 113

Elastic Load Balancing

- about 275
- Application Load Balancer (ALB) 278
- Classic Load Balancer (CLB) 275
- Network Load Balancer (NLB) 276

Elastic MapReduce (EMR) 310

Elastic Network Adapter (ENA) 305

ELB attributes

- connection draining 285
- cross-zone load balancing 285
- Internet-facing, versus internal-facing 285
- stateless, versus stateful 284
- TCP passthrough 285

encryption replication 89

encryption

- Amazon S3 encryption 363
- asymmetric key cryptography 356
- EBS encryption 357
- overview 355
- RDS encryption 374
- symmetric key cryptography 356
- technical requisites 355

enhanced networking 305

extract, transform, and load (ETL) 20

## F

federated access 331

federated identities

- using 206

federation 204

first-class citizen 199

floating IP cloud pattern

- reference 135

functional decomposition 8

## G

General Data Protection Regulation (GDPR) 429

Glacier

- about 257
- archiving 257

global CDN

- about 69
- active-active 70
- active-passive 70
- Amazon CloudFront 69
- complexity 71
- network-partitioning tolerance 70
- rationale 69
- single region / multi-region patterns 69

grants 384

## H

Hadoop Distributed File Systems (HDFS) 311

helper scripts 504, 505, 506

High availability (HA) 222

high performance computing (HPC) 302

horizontal scalability 113

hybrid deployment

- about 159
- Direct Connect (DX) 161
- software VPNs 159
- static hardware VPNs 160
- dynamic hardware VPNs 161

## I

IAM authorization

- about 346
- groups 346, 347
- identity-based policies 347
- inline policies 353
- managed policies, versus inline policies 350
- roles 347
- users 346

IAM identity-based policy

- Sid (Statement ID) 348
- statement 347
- version 347

IAM roles 331, 341

Identity and Access Management (IAM)

- about 38, 329, 435, 442
- access keys, rotating 438
- access structure, designing 44
- accounts, deleting 439
- administration group, creating 45, 46, 47
- cross-account roles 57, 58, 59, 60, 61

identity sharing, avoiding 436  
inline policies 53, 54, 55, 56, 57  
MFA, using for privileged users 436  
password policy 437  
permissions, assigning to groups 438  
permissions, assigning, according to rule of least privilege 438  
permissions, re-evaluating 439  
roles, using 436  
root account access, disabling 439  
user, creating 40, 41, 42, 43  
**Identity Providers (IdP)** 206  
**images** detection in faces, Amazon Rekognition reference 323  
**Incident Response**  
about 510  
CloudWatch Logs Agent 511, 513, 514  
CloudWatch Metric Filters 515, 516, 518, 519  
**information security management system (ISMS)**  
424  
**Infrastructure as a Service (IaaS)** 64  
**Infrastructure as Code (IaC)** 16, 487  
inline policies 353  
Intel 82599 VF  
Virtual Function (VF) 305  
inter-region connectivity 68

## J

JavaScript Object Notation (JSON) 347  
job observer 198  
JSON 488  
JSON Web Tokens (JWTs) 205  
jumbo frames 305

## K

key encryption 355  
**Key Management Service (KMS)**  
about 35, 354, 381, 382  
customer master keys 381, 382  
data encryption keys (DEK) 381, 383  
grants 384  
key policies 383  
key rotation 385  
manual key rotation 386  
key management

technical requisites 355  
key pairs 338, 339  
key policies 383  
key rotation  
performing 385

## L

LAMP stack, Amazon Linux image reference 114  
**LAMP**  
installing 120, 121, 122  
least outstanding request 275  
lifecycle policies  
about 252  
archiving, with Glacier 257  
rule, adding 252, 254, 256  
**Linux Unified Key Setup (LUKS)** 35  
log analysis 312  
Logic Tier 8

## M

managed capabilities, NoSQL  
consistency 267  
DynamoDB Streams 272  
global secondary index 270, 272  
global tables 272  
local secondary index 268, 270  
managed policies  
about 350  
existing managed policy, copying 352  
visual editor, using within IAM 351  
writing, from scratch with JSON policy editor 351  
manual key rotation  
performing 386  
MapReduce 311  
Maximum Transmission Unit (MTU) 305  
messaging 187  
metadata replication 86, 87, 88  
migration strategy  
factors 139  
refactors 140  
mobile apps  
authenticating, with Cognito 204  
monitoring features, Auto Scaling user guide reference 298

## Multi-Factor Authentication (MFA)

about 331, 332, 334, 335  
reference 61

## multi-site active-active

about 232  
best practices 235  
disaster scenario 234  
preparation phase 233  
trade-offs 235

## multi-tier web app

about 507, 508  
best practices 509

## multilayered architecture

## N

### n-tier architecture

Network Access Control Lists (NACLs) 434  
Network Attached Storage (NAS) 127  
Network Load Balancer (NLB) 277  
Ninja of Three (NoT) team 28  
Nondisclosure Agreement (NDA) 444

### normalization factors

reference 466

## NoSQL

about 260  
DynamoDB 260  
managed capabilities 267

## O

### object storage

about 244  
Simple Storage Service (S3) 244  
Open Systems Interconnection model (OSI model)  
275  
OpenID Connect (OIDC) 344  
Organization Level Agreement (OLA) 224  
organizational units (OUs) 482

## P

parallel configuration 112  
paravirtualization (PV) 115  
partition key (PK) 265  
Payment Card Industry (PCI) 444  
pilot light  
about 227

## disaster scenario

229  
preparation phase 228  
trade-offs 229

## placement groups

about 307  
benchmarking 309  
creating 307, 309

## plaintext

355  
Point of Presence (PoP) 68  
Point-to-Point Tunneling Protocol (PPTP) 159  
Portable Document Format (PDF) 197  
Presentation Tier 8  
private key 115  
private traffic 152, 153, 154  
proactive scalability 113, 136  
programmatic access 337  
public dataset  
analyzing 312, 313, 315, 316, 317, 319, 320  
public key infrastructure (PKI) 115

## Q

Quality of Service (QoS) 7, 71  
queuing chain pattern 197  
QuickSight 479

## R

### range key (RK)

265  
RDS dashboard  
caching 243  
events 241  
instances 239, 240  
multi-AZ 241, 242  
option groups 240  
parameter groups 240  
read replicas 242, 243  
snapshots 240  
RDS encryption  
about 374  
enabling 374, 375, 376, 378  
existing database, encrypting 379  
reactive scalability 113  
read capacity units (RCU) 271  
read replicas 138  
Recovery Oriented Computing (ROC) 110  
Recovery Point Objective (RPO) 224

Recovery Time Objective (RTO) 224  
relational database management system (RDBMS) 20  
Relational Database Service (RDS)  
  about 238, 430  
  managed capabilities 238, 239  
relational databases  
  about 238  
  RDS 238  
Reserved Instances (RI)  
  about 465  
  Convertible Reserved Instances 466  
  Standard Reserved Instances 465, 466  
resiliency 123, 124  
resource group  
  creating 408, 409, 410  
retrieval options, Glacier  
  bulk 257  
  expedited 257  
  standard 257  
Return on Investment (ROI) 16  
Route 53  
  about 180, 229  
  failover routing policy 168  
  geolocation routing policy 167  
  geoproximity routing policy 167  
  health checks, performing 169, 170, 172  
  multivalue answer routing policy 168  
  record types 173, 174, 175, 176, 177  
  routing types 167  
  simple routing 167  
  weighted routing policy 168  
rule of least privilege 435

## S

S3 bucket  
  states 72  
S3  
  distributed nature 84, 85  
SAML 2.0 344  
SAML 2.0 federation 344  
scalability  
  about 112  
  horizontal scalability 113  
  proactive scalability 113, 136  
  reactive scalability 113  
  vertical scalability 113  
secret access key 337  
Secrets Manager 443  
Secure Shell (SSH) 115  
security groups  
  about 119, 155  
  chaining 156  
  creating 156  
send-and-forget approach 187  
serial configuration 111  
serverless application architecture 181, 182  
Service Level Agreements (SLAs) 20  
Service Organization Control (SOC) reports 444  
shared nothing approach 112  
shared responsibility model 425, 426, 427  
shared security model 31, 32, 33, 35, 36, 37, 38  
Sid (Statement ID), IAM identity-based policy  
  action attribute 348  
  condition 349  
  effect attribute 348  
  resource 348  
Simple Monthly Calculator  
  reference 141  
Simple Notification Service (SNS)  
  communications, managing 198  
Simple Queue Service (SQS)  
  asynchrony 187  
  durability 190, 192  
  message delivery 193, 194, 195  
  message reception 196, 197  
  messaging patterns 197  
  queue, creating 189  
  reliable broker 187  
  security 189  
Simple Storage Service (S3)  
  about 109  
  availability 246  
  consistency 249, 250  
  cost dimensions 247  
  cost reduction 247  
  data organization 245  
  durability 248  
  integrity 246  
  static website, hosting 91, 92, 93, 94, 95, 96,

97, 98, 99, 100, 101, 102, 103, 104, 105  
storage optimization 250  
use cases 249  
single points of failure (SPOF) 70, 113  
single sign-on (SSO) approach 344  
Software as a Service (SaaS) 140  
software VPNs 159  
SSE-C 368  
SSE-KMS 366  
SSE-S3 365  
Standard Reserved Instances 465, 466  
static hardware VPNs 160  
storage gateways  
    use cases 162  
storage optimization, S3  
    existing object, copying 251  
    lifecycle policy, using 251  
    objects, creating from CLI 250, 251  
storage options  
    relational databases 238  
    technical requisites 238  
StranglerApplication  
    reference 140  
streaming data architecture 183, 184  
Sublime Text 488  
symmetric key cryptography 356

## T

tags  
    replicating 80, 81  
TCO calculator  
    reference 140  
template reusability, AWS CloudFormation  
    mappings 500, 502  
    parameters 496, 498, 499, 500  
template, AWS CloudFormation  
    resources 491  
Total Cost of Ownership (TCO) 71  
Transparent Data Encryption (TDE) 240, 431  
Transport Layer Security (TLS) 397  
Transport Level Security (TLS) 189  
trusted account 342  
trusting account 342

## V

Vault Lock Policies 257  
vertical scalability 113  
virtual private cloud (VPC) 142  
Virtual Private Cloud (VPC)  
    about 34, 434  
virtual private cloud (VPC)  
    default VPC 144, 145, 146, 147, 148  
Virtual Private Cloud (VPC)  
    Flow Logs, creating 435  
    layers, implementing 435  
    NACLs, using for controlling access at subnet  
        level 434  
    rule of least privilege, implementing 435  
    security groups, for controlling access at instance  
        level 434  
virtual private cloud (VPC)  
    sizing 143  
    tenancy 143  
virtual private gateway 159  
virtual public internet gateway  
    provisioning 149, 150  
virtual tape library (VTL) 163  
virtualization  
    technologies 115, 116, 117, 118, 119

## W

warm standby  
    about 230  
disaster scenario 231  
preparation phase 231  
trade-offs 231  
web ACL  
    about 455, 457  
    monitoring 457, 458  
web app  
    demo 220  
Web Application Firewall (WAF) 444  
web application hosting  
    about 180  
    Route 53 180  
web identity federation 344  
Web Identity Providers (IdPs) 344  
web server

scaling 123  
web  
authenticating, with Cognito 204  
WebSockets, in AWS  
about 219  
AWS IoT 219

write capacity units (WCUs) 271  
write once read many (WORM) storage capabilities  
257

## Y

YAML 488