INTRODUCTION

Amazon Web Services (AWS) is an amazing collection of computing, storage, and networking services that enable you to spin up a complete IT environment very quickly, without a significant upfront investment. The cloud is where almost every organization wants to go, if it isn't there already. Of the major public cloud providers, AWS is the largest by a huge margin. Many organizations seek competent, experienced AWS system administrators. One of the best ways to demonstrate competence in managing an AWS environment and to set yourself apart from others is by certifying as an AWS SysOps Administrator.

This book is designed to help you prepare to take and pass the AWS Certified SysOps Administrator Associate exam offered by AWS. Ideally, you should have a year or so of hands-on experience administering AWS before you sit for the exam. The next best solution would be to sign up for the AWS Quick Labs and read the AWS documentation along with this book.

How to Use This Book

The best way to make use of this book is to go through each of the eleven chapters, starting from the beginning. Read the chapter a couple of times and take the review test at the end of the chapter.

End of Chapter Questions

At the end of each chapter module, you'll find questions that test your mastery of the content discussed in the chapter. The questions closely resemble the actual AWS certification exam questions. It's important that you attempt all the questions at the end of each chapter before proceeding to the next chapter. By reviewing the answers for both the questions that you've gotten right as well as those you've answered incorrectly, you can fill in the gaps in your learning.

Online Practice Exams

Follow the same strategy for the practice exams—take the exam, read the explanations, and look in the book for answers to the questions that you missed.

Preparing for the Real Certification Exam

I recommend the following strategy to pass the AWS certification exam.

Take and Retake the Chapter Review Tests and the Practice Exams

Make sure you look up the relevant material in the text for each question that you miss in a review test. Then take the test again. You should be able to answer *all* the questions correctly in each of the eleven chapters before you attempt the online practice exams. And make sure you can correctly answer *all* the questions in the practice exams before you take the actual exam.

The practice exams that come with this book draw their questions from a test bank of over 250 questions. Each practice exam has 65 questions which are randomly selected from the test bank. So, taking the practice exams multiple times means that eventually you'll encounter all 250 questions. I strongly suggest that you take the practice exams multiple times for another reason too. Many of the answers come with detailed explanations that will help you understand why the answer is correct. Reading these explanations will strengthen your understanding of the key concepts.

Read the Docs

Although the book is comprehensive, it can't go into the details of the many topics that it covers. This book is devoted to helping you pass the AWS certification exam. However, you need to know far more than the contents of this book to become a competent AWS SysOps? Administrator. Review the appropriate AWS documentation for each chapter to learn the topics in depth.

Do the Exercises at the End of the Chapters

Make sure you do every exercise at the end of each chapter. These exercises are designed to provide a hands-on experience with all the AWS services that will be on the exam.

Practice with the AWS CLI

The exercises at the end of each chapter expect you to perform tasks using the AWS console for the relevant service. Try to perform tasks from the command line as well, through the AWS CLI. I have sprinkled examples of AWI CLI usage throughout the book, and you can search online for other examples or find them in the AWS documentation.

Read the FAQS for All the AWS Services

This book reviews many AWS services such as Amazon EC2 and AWS CloudFormation. AWS publishes a very useful FAQ page for each of these services. Do yourself a favor and read the FAQs for each of the services covered in this book. The FAQs serve to help you understand each of the services in more depth and explain the ramifications of using various options with each of the services. You can view all the AWS service FAQs here: https://aws.amazon.com/faqs/. AWS FAQs for various services will help you answer many certification exam questions, so read all the FAQs for each service that is covered in this book, such as Amazon VC and Amazon VPC.

Think Strategically

Most of the exam questions are based on scenarios. Read these scenarios a couple of times and understand exactly what the question is asking. Try not to get lost in the specifics of the scenarios. Try to figure out the AWS services that are involved and how to use them in this scenario. Rule out options that are obviously wrong. From the remaining options, select the most reasonable answer.

Who Should Read This Exam Guide

This book is intended for individuals who have technical expertise in deployment, management, and operations on AWS. It validates an examinee's ability to do the following:

- Deploy, manage, and operate scalable, highly available, and fault-tolerant systems on AWS.
- Implement and control the flow of data to and from AWS.
- Select the appropriate AWS service based on compute, data, or security requirements.
- Identify the appropriate uses of AWS operational best practices.
- Estimate AWS usage costs and identify operational cost control mechanisms.
- Migrate on-premise workloads to AWS.

NOTE Most of the material in this section is from the AWS Certification site (https://aws.amazon.com/certification/).

Exam Prerequisites

There are no prerequisites for taking the SysOps Administrator Associate examination. AWS recommends, however, that you have the following AWS knowledge before you take the test:

- Minimum of one year of hands-on experience with AWS
- Experience managing operating systems on AWS
- Understanding of the AWS tenets—architecting for the cloud
- Hands-on experience with the AWS CLI and SDKs/API tools
- Understanding of network technologies as they relate to AWS
- Understanding of security concepts, with hands-on experience in implementing security controls and compliance requirements

AWS recommends that you possess the following general IT knowledge:

- One or two years' experience as a systems administrator in a systems operations role
- Understanding of virtualization technology
- Monitoring and auditing systems experience
- Knowledge of networking concepts (such as DNS, TCP/IP, and firewalls)
- Ability to translate architectural requirements

Exam Question Types

There are two types of questions on the examination:

- **Multiple choice** One correct response and three incorrect responses (distractors).
- **Multiple response** Two or more correct responses out of five or more options. Select one or more responses that best complete the statement or answer the question.

Distractors, or incorrect answers, are response options that an examinee with incomplete knowledge or skill may choose. However, they are generally plausible responses that fit in the content area defined by the test objective. Unanswered questions are scored as incorrect; there is no penalty for guessing.

Unscored Content

Your examination may include unscored items that are placed on the test to gather statistical information. These items are not identified on the form and do not affect your score.

Exam Results

The AWS Certified SysOps Administrator Associate (SOA-C01) exam is a pass or fail exam. The examination is scored against a minimum standard established by AWS professionals who are guided by certification industry best practices and guidelines.

Your results for the examination are reported as a score from 100 to 1000, with a minimum passing score of 720. Your score shows how you performed on the examination as a whole and whether or not you passed. Scaled scoring models are used to equate scores across multiple exam forms that may have slightly different difficulty levels.

Your score report contains a table of classifications of your performance at each section level. This information is designed to provide general feedback concerning your examination performance. The examination uses a compensatory scoring model, which means that you do not need to "pass" the individual sections, only the overall examination. Each section of the examination has a specific weighting, so some sections have more questions than others.

The following table lists the main content domains and their approximate weightings.

Domain	Percent of Examination
Domain 1: Monitoring and Reporting	22
Domain 2: High Availability	8
Domain 3: Deployment and Provisioning	14
Domain 4: Storage and Data Management	12
Domain 5: Security and Compliance	18
Domain 6: Networking	14
Domain 7: Automation and Optimization	12
TOTAL	100

Using the Objective Map

The objective map included in Appendix A has been constructed to help you cross-reference the official exam objectives from AWS with the relevant coverage in the book. References have been provided for the exam objectives exactly as AWS has presented them, including the section that covers that objective and the chapter reference.

Online Content

This book includes online content that features the TotalTester exam software that will enable you to generate a complete practice exam or to generate quizzes by chapter module or by exam domain. See Appendix B for more information.

CHAPTER 3

AWS Identity and Access Management and AWS Service Security

In this chapter, you will

- Learn about the AWS shared responsibility security model
- Use AWS account security features
- Learn more about AWS Identity and Access Management (IAM)
- Learn how to manage AWS component security
- Learn how to secure your network
- Understand how to secure storage services
- Learn how to secure your databases: Amazon DynamoDB, Amazon RDS, Amazon Redshift, and Amazon ElastiCache Security
- Learn how to secure application services
- Understand how to improve security with AWS monitoring tools and services

AWS security is a very broad topic because of the many ways in which you can secure your AWS resources and the flow of Internet traffic to your cloud-based servers and applications.

The AWS Shared Responsibility Security Model

In an AWS cloud, security and compliance are a shared responsibility between AWS and its customers. AWS manages the infrastructure components, ranging from the host operating system and the virtualization layer, down to the physical security of the facilities that host the services. The customers are responsible for managing the guest OS, including applying all updates and security patches and other application software. This shared responsibility between you and AWS reduces the burden on you to secure your infrastructure in the AWS cloud and provides a stronger security posture.

Figure 3-1 illustrates the AWS shared responsibility security model. This separation of security responsibilities is often referred to as the security "of" the cloud versus security "in" the cloud.

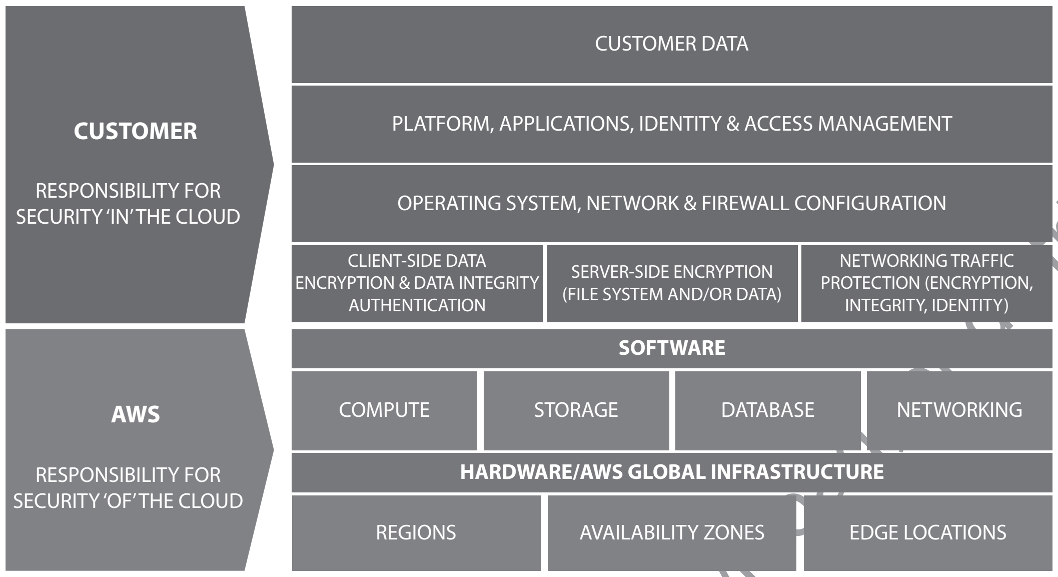
CUSTOMER
RESPONSIBILITY FOR SECURITY 'IN' THE CLOUD
AWS
RESPONSIBILITY FOR SECURITY 'OF' THE CLOUD
CUSTOMER DATA
PLATFORM, APPLICATIONS, IDENTITY & ACCESS MANAGEMENT
OPERATING SYSTEM, NETWORK & FIREWALL CONFIGURATION
CLIENT-SIDE DATA ENCRYPTION & DATA INTEGRITY AUTHENTICATION
SERVER-SIDE ENCRYPTION (FILE SYSTEM AND/OR DATA)
NETWORKING TRAFFIC PROTECTION (ENCRYPTION, INTEGRITY, IDENTITY)
SOFTWARE
COMPUTE
STORAGE
DATABASE
NETWORKING
HARDWARE/AWS GLOBAL INFRASTRUCTURE
REGIONS
AVAILABILITY ZONES
EDGE LOCATIONS

Figure 3-1 The AWS shared responsibility model in the cloud

AWS Responsibility: Security of the Cloud

AWS is responsible for the security of the entire underlying infrastructure on which its customers run the AWS cloud services. Indeed, AWS considers protection of the infrastructure its main priority. The infrastructure consists of all the hardware and software, as well as the networking, storage, and physical facilities that support AWS cloud services. The shared responsibility model means that AWS manages the security for the following assets:

- Physical facilities
- Physical hardware
- Network infrastructure
- Virtualization infrastructure

AWS offers services such as IAM that you can use to manage users and user permissions in AWS services.

EXAM TIP Remember the distinction between "security of the cloud" and "security in the cloud." This helps you identify which security measures fall under AWS's responsibility. For example, protecting against IP spoofing and packet sniffing falls under the "security *of* the cloud" category, so it falls under AWS's responsibility. Patching EC2 instances and databases belongs to the "security *in* the cloud" and is therefore your responsibility, not AWS's. Similarly, managing the security groups for your EC2 instances and the access key rotation policies for your IAM users also fall under your domain, not AWS's.

Auditing AWS Infrastructure Security

It's not possible for all customers to visit the AWS data centers to ensure that the promised protection is indeed there. However, you can rest assured, because AWS offers several types of third-party auditor reports that verify AWS compliance with several computer security standards and regulations, such as Sarbanes-Oxley and PCI DSS.

NOTE Under the shared responsibility security model, the customer is fully responsible for the security of the guest operating system.

AWS Global Infrastructure Security

AWS locates all its computing resources in a global infrastructure, which includes the physical data centers, network, hardware and host OS software, and virtualization software to support users of these resources.

Physical and Environment Security AWS houses its global data centers in nondescript physical buildings and strictly controls physical access at the perimeter using video surveillance, intrusion detection systems, and other physical security measures. AWS enforces a two-factor authentication for authorized staff to access the data center floors, and all physical access to data centers is logged and audited.

AWS protects its data centers from disasters and failures via the following means:

- **Decommissioning old storage devices** AWS uses a formal decommissioning process that destroys data as part of the decommissioning process. Decommissioned magnetic storage devices are degaussed and physically destroyed.
- **Fire detection and suppression** AWS installs automatic fire detection and suppression equipment in its data centers.
- **Redundant power supplies** Data center electric power systems are fully redundant, using uninterruptible power supplies (UPS) through the use of power generators for provision of backup power in the event of electrical failures.
- **Climate control** Data centers are conditioned to maintain temperatures at the optional levels to prevent overheating and reduce service outages.

Business Continuity Management Business continuity management involves providing high availability for the data centers and fast incident detection and response.

AWS builds its data centers as clusters in multiple geographical regions. If a data center fails, its automatic processes direct customer traffic to the unaffected data centers. Distributing your applications across multiple availability zones (AZs) and regions enhances resiliency against failures caused by natural disasters or system failures. The AZs inside each region are designed as independent failure zones by physically separating the zones and locating them in lower-risk flood plains. Data centers also use power from different grids run by different utilities to reduce the possibility of a single point of failure. The Amazon Incident Management team provides fast incident response by proactively detecting incidents and managing their resolution.

In addition, AWS has implemented various types of internal communications to teach employees about their individual roles and responsibilities, including orientation and training programs, video conferencing, and electronic messages. The customer support teams maintain a Service Health Dashboard to alert customers to issues having major impact. The AWS Security Center offers security and compliance details that pertain to AWS.

The AWS Compliance Program AWS follows strict security best practices and security compliance standards. When you set up your systems on top of the AWS infrastructure, you'll share compliance responsibilities with AWS. AWS ties together governance focus and audit-friendly service features with relevant compliance and audit standards to help you operate in an AWS security–controlled environment.

The AWS infrastructure is designed to align with a variety of IT security standards, including the following:

- Service Organization Controls (SOC 1)/Statement on Standards for Attestation Engagements (SSAE 16)/International Standard on Assurance Engagements (ISAE 3402) (formerly SAS 70)
- SOC 2
- SOC 3
- Federal Information Security Management Act (FISMA), DoD Information Assurance Certification and Accreditation Process (DIACAP), and Federal Risk and Authorization Management Program (FedRAMP)
- DoD CSM Levels 1–5
- PCI DSS Level 1
- ISO 9001/ISO 27001
- International Traffic in Arms Regulations (ITAR)
- Federal Information Processing Standard (FIPS 140-2)
- Multi-Tier Cloud Security Standard (MTCS) Level 3

EXAM TIP You're likely to see questions relating to compliance and auditing of your AWS systems. Best practices for preparing for audits include gathering evidence of your IT controls; requesting from AWS third-party audited compliance reports and certifications; and requesting approval from AWS to perform network scans and penetration testing of your AWS instances and endpoints.

In addition, the AWS platform complies with other industry-specific security standards such as these:

- Criminal Justice Information Services (CJIS)
- Cloud Security Alliance (CSA) standards
- Family Educational Rights and Privacy Act (FERPA)

- Health Insurance Portability and Accountability Act (HIPAA)
- Motion Picture Association of America (MPAA)

NOTE You can access the AWS security and compliance reports through the AWS Artifact service (https://aws.amazon.com/artifact). AWS Artifact provides compliance-related reports and select online agreements. You can access reports such as AWS Service Organization Control (SOC) reports, Payment Card Industry (PCI) reports, and certifications from accreditation bodies that validate the implementation and effectiveness of AWS security controls. Online agreements include the Business Associate Addendum (BAA) and the nondisclosure agreement (NDA).

Securing its global infrastructure isn't AWS's only responsibility. AWS is fully responsible for securing all its managed services offerings, such as Amazon RDS, Amazon DynamoDB, and Amazon Elastic MapReduce (EMR). When you use any of the managed services, AWS takes care of the overall security configuration, such as patching the guest operating system and databases, firewall configuration, and many other security aspects. Your responsibility would be to take care of the access controls to your servers and databases.

If a third-party auditing services is auditing your organization, and it requires details about your physical network and ritualization infrastructure, you can approach your AWS representative to help the third-party auditors get the information they need. The AWS representative will facilitate the audits by the third-party auditing services. For auditing purposes, you're responsible for the applications that you run on AWS EC2 as well as securing the OS, including managing the system administrators group.

If an external auditor requests a list of your users and their statuses for audit purposes (say, to determine whether you're using Multi-Factor Authentication) you can generate a credentials report by signing into the AWS Management Console, opening the IAM console, and downloading the report. The credentials report shows all your users and their credential statuses, such as passwords, access keys, and MFA devices. You can also generate the credentials report from the command line, IAM APIs, or through AWS SDKs.

TIP If you need to conduct penetration testing for EC2 instances in your AWS account, you can do the testing with prior authorization from AWS.

Customer's Responsibility: Security in the Cloud

While AWS is responsible for the infrastructure and its support, you, as the customer, are responsible for everything you place in the cloud (think data!) or connect to the AWS cloud, in addition to securing the OSs, platforms, and data. AWS customer responsibility varies according to the services that the customer chooses. In the case of services categorized as Infrastructure as a Service (IaaS), for example, such as Amazon EC2, Amazon VPC, and Amazon S3, the customer performs all the security configuration and management tasks.

If you deploy an EC2 instance, you're responsible for managing the guest OS as well as the application software and utilities that you install on the EC2 instance. In addition, you're responsible for configuring the security groups on each of the instances. As you'll recall, a security group acts as a virtual firewall.

The type and extent of security configuration you must perform depends on the specific AWS service and the importance of the data you store in the cloud. With EC2, for example, the customer is responsible for securing the following:

- Amazon Machine Images (AMIs)
- Guest operating systems (including updates and security patching)
- Applications
- Firewalls (security groups)
- Data (stored on disk and in transit)
- Credentials
- Policies and configuration

Regardless of the type of service, you must set up certain security elements such as IAM, Secure Sockets Layer/Transport Layer Security (SSL/TLS) for encrypting data in motion, and a strong logging framework (using AWS CloudTrail) to protect your cloud infrastructure and the data you store in it.

Sharing Security Responsibility for AWS Services

You can categorize security and shared responsibility for the AWS infrastructure and platform services into the following categories, each of which has a slightly different security ownership model:

- **Infrastructure services** These are the various compute services such as EC2, and associated services such as Amazon Elastic Block Storage (EBS), Auto Scaling, and Amazon VPC. You control the OS and configure and manage the identity management system that enables access to the user layers of the virtualization stack.
- **Container services** These services typically live in separate EC2 or other infrastructure instances, and for the most part, you don't manage the OS or the platform layer. You are responsible for setting up network controls such as firewall rules and managing the platform-level identity and access management separately from IAM.
- **Abstracted services** These services include high-level storage, database, and messaging services such as S3, Glacier, DynamoDB, Simple Queue Service (SQS), and Simple Notification Service (SNS). These are services in the platform layer on which you build cloud applications. You use AWS APIs to access the endpoints of these abstracted services. Abstracted services are offered on a multitenant platform that stores your data securely in an isolated fashion.

Responsibility for IT Controls and Compliance

The same shared responsibility model for securing the IT environment also applies to IT control. You follow a distributed control strategy in the AWS cloud for managing, operating, and verifying IT controls. AWS is responsible for managing the controls associated with the physical infrastructure.

There are three types of controls based on how they're managed by AWS, you, and/or both:

- **Inherited controls** These are controls that you fully inherit from AWS, such as the physical and environment controls managed by AWS.
- **Shared controls** AWS provides the infrastructure requirements, and you must provide your own control implementation within your use of the AWS services. Here are examples:
  - **Patch management** AWS is responsible for patching the infrastructure, but you are responsible for patching your guest OS and application software.
  - **Configuration management** AWS configures its infrastructure devices, but you configure your own guest OS, databases, and applications.
- **Customer-specific controls** These controls are solely your responsibility, depending on the applications you deploy within AWS services. For example, Zone Security may require you to zone data within specific security environments.

Security for the AWS-Managed Services

Throughout this book, you'll learn about various AWS-managed services, such as the Amazon Relational Database Service (Amazon RDS), where AWS fully manages the relevant service. AWS is responsible for the security configuration of all of its managed services. You need to configure access controls with AWS IAM and account credentials for database user accounts for the managed service, such as a MySQL database service (RDS) and similar services.

Network Security

AWS secures its network infrastructure using several strategies:

- **Secure network architecture** Network devices such as firewalls and other boundary services use rule sets and access controls lists (ACLs) to control network traffic flow to and from each managed network interface.
- **Secure access points** Strategically placed cluster customer access points called *API endpoints* enable secure HTTP and (HTTPS) access to your storage and compute instances.
- **Transmission protection** You can connect to AWS access points using SSL to protect against tampering and message forgery. If you need additional layers of network security, you can use Amazon Virtual Private Cloud (VPC), which provokes a secure subnet within the AWS cloud. VPCs offer the ability to use an IPsec virtual private network (VPN) to provide an encrypted tunnel for transmitting data between your data center and Amazon VPC.

Network Monitoring and Protection

AWS uses several automated monitoring systems to enhance service performance and availability. The monitoring is designed to catch unauthorized activities at incoming and outgoing communication points by monitoring server/network usage, port-scanning activities, application usage, and intrusion attempts.

AWS security monitoring tools help identify the following types of attacks.

Distributed Denial-of-Service (DDoS) Attacks AWS locates API endpoints on world-class infrastructure and uses proprietary DDoS mitigation techniques. It also multihomes its networks across providers to achieve diversified Internet access, which helps in situations such as a DDoS attack.

Man-in-the-Middle (MITM) Attacks AWS encourages its users to use SSL. All the AWS APIs are available via SSL-protected endpoints. EC2 AMIs generate new SSH certificates when you first boot an instance. You can use the AWS Certificate Manager (ACM) to call the console and get the host certificates before logging onto the new instance.

IP Spoofing EC2 instances can't send spoof network traffic. The host-based firewall won't permit an EC2 instance to send traffic with any source IP or MAC address other than its own IP/MAC address.

Port Scanning AWS stops and blocks all unauthorized port scanning. Since, by default, all inbound ports of EC2 instances are closed, port scanning isn't effective with an EC2 instance. By configuring appropriate security groups, you can further minimize the threat of port scans. As a customer of AWS, you can request permission from AWS to conduct vulnerability scans that you need, but you must limit the scans to your own instances and must not violate the AWS Acceptable Use Policy.

Packing Sniffing by Other AWS Tenants EC2 instances that you own and that are located on the same physical hosts cannot listen to one another's traffic. Even if you place a VM into promiscuous mode to receive or "sniff" traffic being sent to other VMs, the hypervisor won't deliver any traffic that isn't addressed to this instance. Well-known security attacks such as Address Resolution Protocol (ARP) cache poisoning aren't possible in EC2 and VPC.

In addition to constant monitoring, AWS also performs regular vulnerability scans on the OS, web applications, and databases.

AWS Account Security Features

You can use various tools and features to protect your AWS account and AWS resources, as summarized in the following sections.

AWS Credentials

AWS uses several types of credentials for authentication to ensure that only authorized users and processes can access your accounts and resources.

AWS recommends that you regularly change your access keys and certificates. You can rotate the access keys of your IAM account as well as your IAM user accounts with the AWS IAM API.

TIP You can download a credentials report for your account from the Security Credentials page. The report shows all your account users and the statuses of their credentials, such as whether they have enabled MFA.

Passwords

AWS requires passwords for accessing the AWS account, as well as the individual IAM user account and the AWS Support Center. You can change the passwords from the Security Credentials page. AWS recommends the creation of strong and hard-to-guess passwords.

You can set a password policy for your IAM user accounts to ensure the use of strong passwords and ensure that they're changed on a regular basis. Several password policy options are available to you, such as preventing password reuse by users.

You can choose to require a password reset by the administrator when user passwords expire or allow some users to manage their passwords. The password expiration requires an administrator reset option in the console and prevents IAM users from choosing a new password when their password expires. If you decide to enable this option, be sure to do one of the following, so you're not locked out of your own AWS account when your password expires:

- Make sure you have the access keys, which will enable you to use the AWS CLI (or the AWS API) to reset your own password, even if you cannot log into the AWS Management Console.
- Alternatively or in addition, ensure that multiple administrators have the necessary administrative permissions to reset IAM user passwords. This way, even if you as the administrator don't have the access keys, other administrators can reset your console password for you.

AWS Multi-Factor Authentication

With Multi-Factor Authentication (MFA), a system checks for more than one authentication factor before granting access. This usually consists of a username/password combination (something you know) and the code from an authentication device (something you have). MFA provides an additional layer of security when users authenticate to the AWS cloud.

When you enable MFA, you need to provide a six-digit single-use code, in addition to a username/password, to gain access to your accounts. The single-use code may be provided by an authentication device you carry with you.

NOTE AWS MFA supports MFA through hardware devices such as Gemalto, virtual MFA devices such as Microsoft Authenticator and Google Authenticator, and Simple Message Service (SMS) messages that run on mobile hardware devices, including smartphones.

You can enable MFA for all the IAM users that you create. You can also add MFA protection for access across IAM accounts. This enables a user in one AWS account to use an IAM role to access AWS resources in a different AWS account. Before the user can assume the role, you can require the user to use MFA.

IAM Access Keys

As you know, access keys consist of two components: an access key ID and a secret access key. A user can have a maximum of two active access keys to enable continued access even while the active key is being rotated. Users can list and rotate their own access keys.

As a best practice, you must regularly rotate all your IAM user's access keys. You can view an access key's usage history to determine whether the key is being used and remove unnecessary active keys from users who don't use them. You can revoke an IAM user's access by disabling their access keys, which makes the keys inactive. You can also delete the access keys of a user.

As described in Chapter 2, the IAM access keys are by default stored in the ~/.aws/credentials file on a Linux server. AWS recommends that you not use the root access keys and that you lock these keys.

You are required to sign all your API requests with a digital signature to verify your identity. The text of your request and your secret access key serve as inputs to the hash function used to calculate the digital signature. Your applications must calculate the digital signature, but when you use the AWS SDKs, the signature is calculated for you. If a request doesn't reach AWS within 15 minutes of the request's timestamp, AWS turns down the request.

AWS also recommends that you not embed the access keys in your code. Using IAM roles is a safe and easy way to manage access key distribution. IAM roles offer temporary credentials that are automatically loaded to the target instances and are automatically rotated several times daily.

Key Pairs

Instead of passwords, EC2 instances use a public/private key pair to sign in via SSH. The public key is embedded in the EC2 instance. You sign in securely with your private key. You can have AWS generate a key pair for you automatically when you launch an instance, or you can generate your own and load it.

In addition to EC2, Amazon CloudFront also uses key pairs. It does so when creating signed URLs of private content, such as when you distribute restricted content that someone paid for.

X-509 Certificates

X-509 certificates contain a public key and additional metadata such as the certificate expiry date and are associated with a private key. These certificates are used to sign SOAP-based requests. They are also used as SSL/TLS certificates for users who use HTTPS for encrypting their data.

To create a request, you use your private key to create a digital signature and include the signature in the request, along with your X.509 certificate. AWS verifies your identity by decrypting the signature with the public key embedded in your certificate.

As with key pairs, you can let AWS create the certificate and private key for you to download, or you can upload your own certificate. If you're using the X-509 certificates for HTTPS, you can use a tool such as OpenSSL to create a unique private key, which you'll then use to create a Certificate Signing Request (CSR) to submit to a certificate authority (CA) to get the server certificate. Using the AWS CLI, you can upload the certificate, private key, and certificate chain to IAM.

Individual User Accounts

You use AWS IAM to create and manage individual users in your AWS account. A user can be a person, an application, or a system that works with AWS resources programmatically, from the command line, or through the AWS Management Console.

You use IAM to define permission policies that control what your users can do with your AWS resources. Grant users the minimum permissions they need to do their job.

Secure HTTP Access Points

AWS recommends that you use HTTPS (which uses the SSL/TLS protocol that uses public key cryptography to prevent unauthorized use) instead of HTTP for transmitting data. AWS services offer secure customer access points called *API endpoints* to help you establish HTTPS sessions.

Security Logs

AWS CloudTrail tracks and records all requests for AWS resources you own. The logs reveal the services accessed, actions performed, the time when the actions were performed, and the identity of the individual or service.

CloudTrail captures data about all API calls made to all your resources. It delivers its event logs every 5 minutes, and you can configure it so it stores aggregated log files from multiple regions into an Amazon S3 bucket. You can move the logs from S3 to Glacier for long-term auditing and compliance requirements.

You can upload CloudTrail logs from S3 to your own log management solutions to analyze them and detect unusual patterns. The CloudWatch Logs service tracks systems, applications, and more, from EC2 instances and other sources in near real-time, helping you to detect unauthorized login attempts, for example, by monitoring the web server log files.

AWS Trusted Advisor Security Checks

AWS Trusted Advisor offers a set of best-practice checks to help increase your security and performance. The advisor can also inspect your AWS environment and offer security remediation recommendations or close security gaps.

Managing Cryptographic Keys for Encryption

Earlier, you learned how to safeguard your public/private key pairs that help you log into your EC2 instances. You also learned how X-509 certificates help AWS to verify the authenticity of a requester by decrypting their digital signatures (created with a private key) with the public key and X-509 certificates to safeguard interactions with AWS.

You use encryption in various services, such as databases. To protect your cryptographic keys, AWS offers two important services or tools: the AWS Key Management Service (KMS), and the AWS cloud hardware security module (CloudHSM).

The AWS Key Management Service

AWS KMS is a managed service that helps you create and manage the encryption keys that you use to encrypt your data stored in the AWS cloud. KMS helps you create, import, rotate, disable, and define the usage policies for encryption keys that you use.

KMS uses FIPS 140-2 validated hardware security modules to safeguard the keys. KMS is integrated with most AWS services, which makes it easy for you to encrypt data you store in these services with encryption keys that you fully control. It's also well integrated with AWS CloudTrail, thus providing you with a record of all encryption key usage, which helps in meeting many regulatory and compliance needs.

Following are the key features of AWS KMS.

Centralized Key Management You can create and manage encryption keys from the AWS Management Console or with the AWS CLI and SDK, which offers you centralized control of your encryption keys. You can have KMS create the master keys or import them from somewhere else. These keys are stored in an encrypted format, and you can have KMS rotate the master keys once a year, without having to re-encrypt the previously encrypted data, since KMS keeps older master keys available.

Easy Encryption You can easily encrypt data through a one-click encryption in the AWS console or use the AWS SDK to incorporate encryption in your application code.

Audit Capabilities AWS CloudTrail can record all uses of a key that you store in KMS in a log file that it can deliver to an S3 bucket.

Scalability and High Availability AWS KMS scales automatically as your encryption needs grow over time. KMS stores multiple copies of the encrypted versions of your encryption keys to provide 99.999999999 percent durability. For high availability of the encryption keys, KMS is deployed in multiple AZs inside an AWS region.

Secure Safeguarding of Encryption Keys Your master keys are stored securely and cannot be exported out of KMS. No one, including AWS employees, can view the KMS service's plain-text keys, which are never written to disk and are pulled into volatile memory during cryptographic operations. KMS uses a FIPS 140-2 validated hardware security module (HSM) to safeguard the keys.

Compliance AWS KMS has been certified by several compliance standards that include PCI DSS Level 1, IS0 27018, and ISO 9001.

Storing and Managing Encryption Keys

You can use encryption for protecting data in your databases, document signing, transaction processing, and for digital rights management (DRM). Encryption strategies require encryption keys. You can use your own processes for managing encryption keys in the

AWS cloud or rely on server-side encryption with AWS key management and storage capabilities.

If you choose to manage encryption keys yourself, AWS recommends that you store keys in tamper-proof appliance such as an HSM. If you choose to store the keys on premises in an HSM, you can access the AWS keys over secure links such as IPsec VPNs, or AWS Direct Connect.

AWS offers an HSM service, AWS CloudHSM. If you use CloudHSM instead of your own on-premise HSM, you receive a dedicated single-tenant access to CloudHSM appliances, which are resources in your Amazon VPC. The appliance has a private IP and runs in a private subnet. You connect to the appliance from EC2 servers via SSL/TLS, with two-way digital certificate authentication and 256-bit SSL encryption.

If you use CloudHSM, try and choose a CloudHSM service in the same region as your EC2 instance to decrease network latency and thus improve application performance.

CloudHSM appliances can securely store and process cryptographic keys for database encryption, Public Key Infrastructure (PKI), authentication and authorization, transaction processing, and so on. They support strong cryptographic algorithms such as AES and RSA.

Although AWS is responsible for managing, monitoring, and maintaining the health of the CloudHSM appliance, only you control your security keys and the operations performed by CloudHSM. A *cryptographic domain* is a logical and physical security boundary that restricts access to your encryption keys. AWS doesn't have access to the cryptographic domain. You initialize and manage the cryptographic domain of CloudHSM.

When you start working with CloudHSM, you must set up one or more cryptographic partitions on it. A partition is a logical and physical security boundary that restricts access to your cryptographic keys. AWS can't see inside the HSM partitions, although it has admin credentials to the appliance itself in order to monitor the appliance's health and availability. Therefore, AWS can't perform cryptographic operations with your keys.

HSM also offers both physical and logical tamper detection and response capabilities that erase cryptographic key material and log the events when HSM detects tampering.

AWS Identity and Access Management

AWS IAM is a web service that helps you secure access to your AWS resources. IAM helps you centrally manage your users, security credentials (passwords and access keys), and permissions policies that determine which AWS services and resources users can access. More precisely, IAM enables you to do the following:

- Create users and groups in your AWS account.
- Assign unique security credentials for users in your AWS account.
- Share AWS resources among your users.
- Control user access to AWS services and resources.
- Control each user's permissions to perform tasks with AWS resources.

- Allow users in another AWS account to share your AWS resources.
- Create roles and define the users or services that can assume the roles.

IAM enables you to control how AWS products are administered, such as the creation and termination of EC2 instances. IAM controls the administrative tasks that you perform via the console, command line tools, or the AWS SDKs. You can work with IAM through the AWS Management Console, AWS CLI, and AWS SDKs.

It's important to understand that AWS services such as EC2 and Amazon RDS have their own ways of securing access to their resources. These access methods are separate from IAM.

IAM has two main components:

- **Identity** Who is authenticated legitimately (signed in)
- **Access management** Who has the authorization (permissions) to use the AWS resources

How IAM Works

IAM provides the infrastructure that manages authentication and authorization for AWS accounts. IAM uses the following elements to perform its tasks:

- Principals
- Requests
- Authentication
- Authorization
- Actions or operations
- Resources

NOTE By default, only the AWS account root user has access to all resources in that account. All other users must have permissions granted by a policy.

Principals

A *principal* is any entity that can act on an AWS resource. Originally, when you create your AWS account, you create the administrative IAM user as your first principal. Later, you can allow users and services to assume a role. You can also support applications to programmatically access your AWS account. All users and roles (including federated users and applications) are AWS principals.

Figure 3-2 shows how principals perform actions on resources through authorization granted via IAM policies.

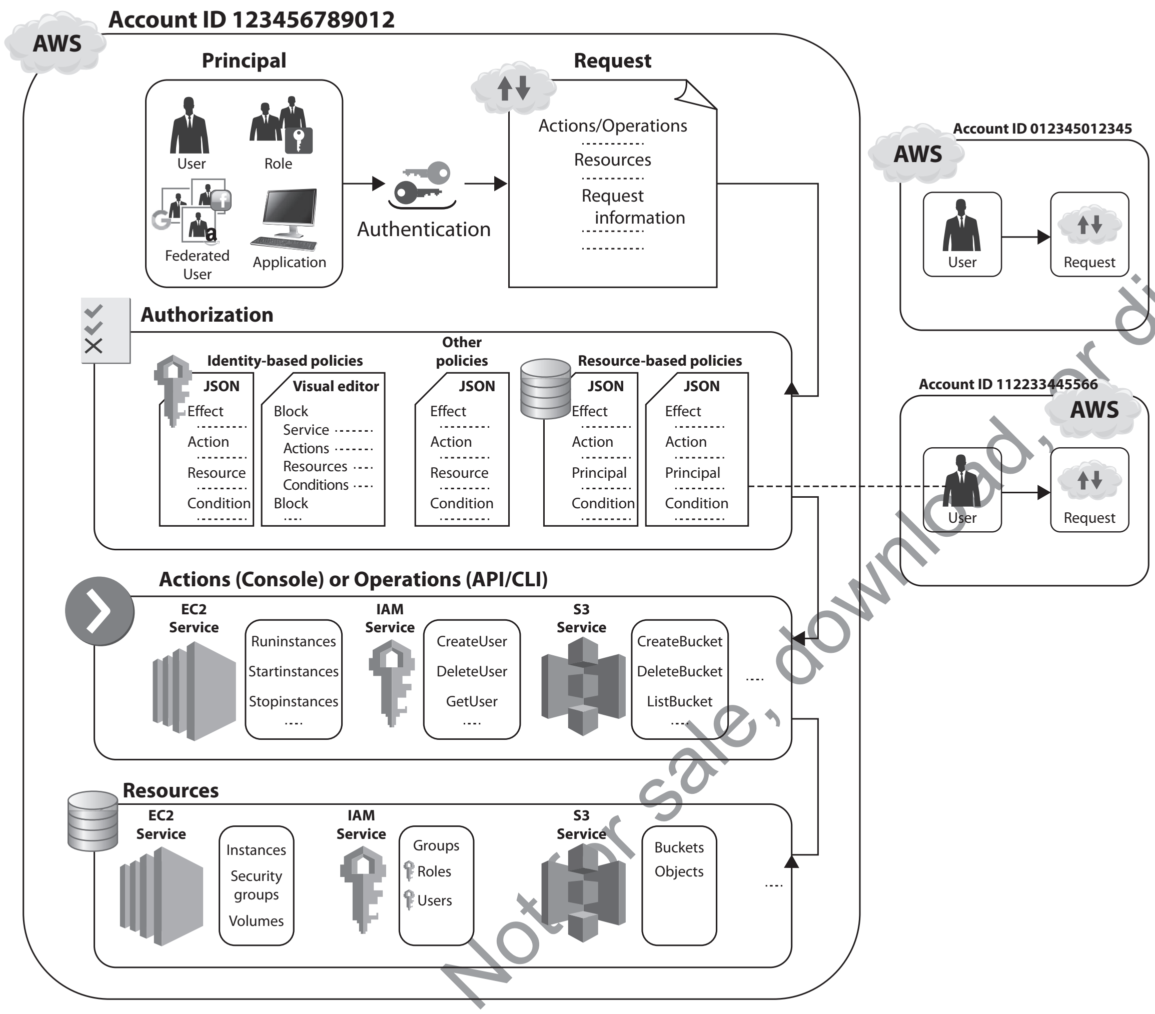

Figure 3-2 How principals perform actions on resources in IAM

Requests

Principals send requests via the AWS console, the AWS CLI, or the AWS API. A request contains information about the following:

- The principal (requestor), including the permissions granted to that principal via IAM policies
- The resources on which the user wants to perform the action, such as the name of a DynamoDB table or a tag associated with an Amazon EC2 instance
- The actions or operations that the principal wants to perform on the resources

AWS takes the request information such as principals, resources, actions, and environmental data (such as the IP address) and forms a request *context*, which it uses to evaluate the request and determine whether it should authorize it.

Authentication vs. Authorization

It's important that you understand the difference between *authentication* and *authorization*. AWS must authenticate and authorize your request before it approves the actions in the request.

Before a principal can access a resource and perform actions on it, AWS must first *authenticate* the user. You authenticate from the console by signing in with your username and password. You authenticate through the API or CLI by providing your access key and secret key.

In authorization, IAM checks the request contexts for matching policies so it can allow or deny the requests. AWS collects all the request information and puts it into a request context, which it uses to evaluate the request and determine whether it should authorize it. Policies specify the permissions that are either allowed or denied for principals or resources.

Actions (Operations)

Once IAM authenticates the user and authorizes the request, AWS will approve the action/operation that you specify in your request. The operations that you can perform are defined by a service, so they can vary. For example, you can perform actions such as the following on a user resource:

- Create user
- Delete user
- Update user

Resources

A resource is an object that exists within a service, such as an EC2 instance or an S3 bucket. I discuss resources in more details in the next section.

Managing the Identity Component of IAM

When you sign up for AWS, the first account that you create is an AWS account, and you use your e-mail address and password as credentials. You can use these credentials to log into the AWS Management Console. You can also create a set of access keys to use when you need to make programmatic calls to AWS via the CLI, the AWS SDKs, or API calls.

IAM enables you to create individual users in your AWS account with their own username/password. The users can log into the console using an account-specific URL. You can also create access keys for users to enable them to make programmatic calls to AWS resources.

NOTE Remember that as a best practice, AWS recommends that you create an IAM user for yourself (the administrator) and then not use your AWS account credentials for everyday access to AWS services and resources. Instead, use the IAM account.

You must understand two key things in the context of IAM:

- **Principal** This AWS entity can perform actions in AWS. A principal doesn't have to be a specific human user; it can be the AWS account root user, and IAM user, or a role.
- **Resource** This object exists within a service, such as an EC2 instance or an S3 bucket. An *Amazon Resource Name* (ARN) uniquely identifies an AWS resource. AWS requires you to use an ARN so that you can unambiguously specify resources across all of AWS, such as when you create an IAM policy or an Amazon RDS tag, for example. Following is the general format of an ARN:

  `arn:partition:service:region:account-id:resource,`

  - *partition* Where the resource is located. For standard AWS regions, the partition is aws. For resources that live in other partitions, the partition is aws-*partitionname*, as in aws-cn, which is the partition for resources location in the Beijing, China, AWS region.
  - *service* Identifies the AWS product. For IAM resources, the service is always IAM.
  - *region* The region the resource lives in. For IAM resources, you leave the region blank.
  - *account-id* The AWS account ID, such as 123456789012.
  - *resource* The part of the ARN that identifies a specific resource.

When working with AWS IAM, you must understand both the ARN format as well as IAM identifiers. I explain the two in the following sections.

IAM ARN Formats

An ARN can take slightly different formats, depending on the resource, as shown here:

```
arn:partition:service:region:account-id:resourcetype/resource
arn:partition:service:region:account-id:resourcetype/resource/qualifier
arn:partition:service:region:account-id:resourcetype/resource:qualifier
arn:partition:service:region:account-id:resourcetype:resource
arn:partition:service:region:account-id:resourcetype:resource:qualifier
```

Following are some examples of ARNs for various services.

AWS Batch

```
arn:aws:batch:us-east-1:123456789012:job-definition/my-job-definition:1
```

Amazon DynamoDB

```
arn:aws:dynamodb:us-east-1:123456789012:table/books_table
```

Amazon EC2

```
arn:aws:ec2:us-east-1:123456789012:dedicated-host/h-12345678
```

An AWS account

```
arn:aws:iam:123456789012:root
```

An IAM user in the AWS account

```
arn:aws:iam::123456789012:user/Sam
```

An IAM group

```
arn:aws:iam:123456789012:group/Developers
```

An IAM role

```
arn:aws:iam::123456789012:role/S3Access
```

You can specify all users, groups, or policies in your account by specifying a wildcard for the user, group, or policy portion of the ARN:

```
arn:aws:iam::123456789012:user/*
arn:aws:iam::123456789012:group/*
arn:aws:iam::123456789012:policy/*
```

IAM Identifiers

IAM employs different *identifiers* for users, roles, groups, policies, and server certificates. The three types of identifiers are friendly names and paths, IAM ARNs, and unique IDs.

Friendly Names and Paths When creating users, groups, or policies, or when you're uploading serer certificates, you can specify a friendly name for the entities, such as Sam, Administrators, ProdApp1, ManageCredentialsPermissions, or ProdServerCert.

When you use the AWS command line to create the IAM entities, you can optionally provide a path for an entity. The path can be a single path or a nested multiple path structure, such as a directory structure.

IAM ARNs Although many AWS resources have friendly names such as a user named Sam and a group named Administrators, you can't use these in permission policies, which require that you specify the ARN format instead when referring to resources such as users and groups.

Unique IDs IAM assigns a unique ID to each user, group, role, policy, instance profile, and server certificate that it creates. The unique ID is a string of letters and numbers, such as the following: AIDAJQABLZS4A3QDU576Q.

You use friendly names or IAM ARNs mostly when working with IAM entities such as users and groups, but the unique ID is helpful when you can't use friendly names. All IAM users have a unique ID, even if you assign an IAM user a previously used friendly name. In cases such as this, using unique IDs rather than friendly names helps tighten data access.

You can't get unique IDs for an IAM entity from the IAM console. You can get the IDs via the AWS CLI or an IAM API calls. In AWS CLI, for example, the `get-user`, `get-role`, and `get-group` commands will provide the unique IDs for a user, role, and group, respectively.

Managing the Authentication Component of IAM

AWS IAM enables you to manage secure access to your AWS resources and AWS services. The *identity* portion of AWS IAM deals with the creation and management of AWS users and groups. The *access management* part of IAM deals with how you use permission to allow and deny users access to your AWS resources.

With IAM, you can easily provide users secure access to your AWS resources. You can manage IAM users and their access by creating users in the AWS identity management system and assign those users various types of security credentials, such as access keys, passwords, and MFA devices. In addition to helping you manage IAM users, IAM helps you manage access to federated users in your corporate directory (such as Active Directory), without having to create IAM user accounts for them.

You can access AWS services and resources via different types of identities: the AWS account root user, IAM users, and IAM roles, as explained in the following sections.

AWS Account Root Users

The AWS account root user is the user account (with your e-mail address and password as the credentials) set up when you first create an AWS account. The AWS account root user account has full access to all AWS services and resources in your account. The root user has full, unrestricted access to your entire AWS account. Because you can't restrict the permissions granted to the main AWS account, the best practice is to use the root user to create your first IAM user and lock up the root user credentials. Even if you're the only person who works with your AWS account, it's a best practice to create an IAM user and use that user's identity (and credentials) when working with AWS.

IAM Users

IAM users are unique identities recognized by AWS services and applications. You create IAM users with specific permissions to access services and resources. For example, you can create an IAM user with permissions to create a hosted zone in Amazon DNS, Route 53.

TIP You can use the Active Directory Connector (AD Connector) to simplify identity management by sourcing the user identities directly from Active Directory. AD Connector is a directory gateway that redirects directory requests to your on-premises Microsoft Active Directory. Your IAM users can use their existing Active Directory credentials. You can reuse existing Active Directory security policies as well, such as those that control password expiration, and account lockout.

IAM helps you manage access to two broad sets of users: It supports IAM users you create and that are managed in AWS's IAM system. In addition, it enables you to grant access to your AWS resources to *federated users* that are managed outside AWS in your corporate directory.

IAM users can sign in to the AWS Management Console and use the AWS CLI or APIs to make programmatic requests to AWS services. Although you can directly attach permission policies to an IAM user, the recommended way to grant permissions is by making the user part of a group that has the required permission policies.

A user doesn't have to be a person: it can be an individual, system, or an application that requires access to an AWS service. Each user has a unique name and security credentials, such as a password and/or access keys (up to two access keys for use with the CLI or API).

Usernames can be a combination of up to 64 letters, digits, and the characters plus (+), equal (=), comma (,), period (.), at sign (@), and hyphen (-). Names aren't distinguished based on case. So, for example, sam@mycloud is a valid username and so is sam-mycloud, but not sam#mycloud.

IAM Roles

An IAM role is similar to an IAM user, but unlike a user, a role doesn't have or require credentials and it is temporary, set to expire after a defined time period (the default expiration time is 12 hours). Once a role expires, it can't be reused. An IAM role defines a set of permissions for making AWS service requests. It uses temporary security credentials that enable you to delegate access to users or services that need access to your AWS resources. You can create an IAM role with specific permissions that aren't tied to a specific user or group. Rather, entities such as IAM users, applications, or even AWS services such as EC instances assume the roles. The credentials are automatically loaded, to the target instances, and you don't need to embed them in code, which isn't safe. They are automatically rotated several times daily and stored securely.

NOTE IAM users can request temporary security credentials for their own use by calling the AWS STS GetSessionToken API. The default expiration for these temporary credentials is 12 hours; the minimum is 15 minutes, and the maximum is 36 hours.

You create a role similar to how you create an IAM user: you assign a name and attach an IAM policy to it. IAM roles can be used to solve several security problems. For example, say that you have 1000 employees in your organization for which you need to provide access to your new AWS cloud resources. The best way to do this is to create roles to authorize various users to access the different AWS resources. Creating roles is a best practice for AWS IAM.

An IAM role enables you to delegate access to trusted entities with defined permissions, without your having to share access keys long-term. Roles are helpful when you don't want to create regular IAM users with persistent permissions for accessing resources in your AWS account. With an appropriate IAM role, an IAM user can access not only resources in their own AWS account, but also resources in other AWS accounts.

NOTE You can use the AWS Security Token Service (STS) to provided trusted users with temporary security credentials that allow access to your AWS resources. You can assume an IAM role by calling the AWS STS AssumeRole APIs (`AssumeRole`, `AssumeRoleWithWebIdentity`, and `AssumeRoleWithSAML`). The APIs return a set of temporary security credentials that apps can use to sign their requests to AWS service APIs. There's no limit on the number of IAM roles that you can assume.

Before an IAM user, application, or service can use a role, you must grant the entity permission to switch the role through a permissions policy that you attach to the IAM user or user group. Use the `DurationSeconds` parameter to specify the duration of the role session, with a minimum value of 900 seconds (15 minutes), up to the maximum CLI/API session duration for the role.

IAM roles with temporary credentials are useful in the following cases, where you need to provide limited, controlled access.

Federated (Non-AWS) User Access You can assign a role to federated users (or applications) that don't have an AWS account when they request access through an identity provider such as Active Directory, Lightweight Directory Access Protocol (LDAP), or Kerberos. The temporary AWS credentials are assigned to the roles provided with the identity federation between AWS and non-AWS users in your organization's identity and access management system. With web identity federation, you don't need to manage identities of your users. Instead, the users of your apps can sign in via an identity provider (IdP) such as Amazon, Facebook, Google, or any other OpenID Connect (OIDC)–compatible IdP and receive an authentication token, which they exchange for temporary credentials in AWS that map to an IAM role with permissions to use resources in your account. The IdP enables you to manage user identities outside of IAM and grant these external users permissions to your AWS resources.

Create Trust Between Organizations If your company supports Security Assertion Markup Language (SAML) 2.0, you can create trust between your organization (identity provider) and other organizations (service providers).

Allow Externally Authenticated Users to Log In Roles also enable users to sign in to your applications by logging into Amazon, Facebook, or Google. The users can use information from their successful authentication to assume the roles and, thus, the temporary security credentials for accessing your AWS resources. If you're currently managing identities outside of AWS, you can use IAM IdPs instead of creating IAM users in your account. The best practice here is to use an IAM role that validates the calls to the DynamoDB database with IdPs.

Provide Cross-Account Access You can use roles to provide users with permissions in an AWS account to access your resources in another AWS account owned by your organization, thus ensuring that only temporary credentials are provided on an as-needed basis to the cross-account users. For details, see the section "Granting IAM Users Permissions to Switch to other Roles" a bit later in this chapter.

Allow Applications to Access AWS Resources Applications running on EC2 instances require security credentials when they make requests to an AWS resource such as S3 buckets. Roles help you grant AWS services permissions to access your AWS resources. Instead of creating individual IAM accounts for each application and storing access keys on the EC2 instances where applications are running, you can use IAM roles as a way of granting temporary credentials for the applications when they make AWS API requests. This is especially useful when dealing with a large number of EC2 instances that are spawned dynamically, through AWS Auto Scaling.

NOTE When dealing with IAM roles, remember that the permissions policy is attached to the role and not to the IAM user or IAM group member that temporarily assumes that IAM role.

IAM Groups

Use IAM groups to manage permissions that you want to grant to more than one user and to gather users based on their functional roles. Suppose, for example, that there are 50 IAM users in your AWS account. If your company introduces a new policy that changes the access of the IAM users, you don't need to apply the new policy at the individual user level. You can add the new security policy to the group. Remember that there's no default group in your AWS account. You create the groups as needed. An IAM user can be a member of more than one IAM group. When a user joins or leaves your organization, you can add or remove the user from the groups.

EXAM TIP Be sure you understand the things you can do with groups: You can add or remove users from a group and grant permissions to groups using access control policies. Also, groups can't belong to other groups, but a user can belong to multiple groups.

Know the Difference: IAM Users vs. Groups vs. Roles

A tricky part of IAM is in clearly understanding the differences and the relationships among IAM users, IAM groups, and IAM roles:

- An IAM user has permanent credentials and directly interacts with AWS services.
- An IAM group is a logical entity that you use to grant/modify/revoke permissions for a similar set of IAM users.
- An IAM role is an entity with permissions to make service requests to AWS services such as EC2. A role can't make direct requests to an AWS service; roles are strictly meant to be assumed by entities such as IAM users, applications, and AWS services. Roles help you delegate access within or between various AWS services.

Granting IAM Users Permissions to Switch to Other Roles

You can grant your IAM users permission to switch to other roles in your AWS account. AWS recommends that you adopt this approach to enforce the *principle of least access*—that is, grant users more advanced or more powerful permissions only for times they need them to perform specific tasks, such as terminate an EC2 instance. This reduces your security exposure and minimizes or prevents accidental changes to critical environments. Auditors will also appreciate the fact that you granted role permissions only when you needed to.

A user in an AWS account can switch to another role in the same or a different AWS account. When users switch to a different role, their original permissions are suspended, and they're restricted to just those actions and resources that are permitted by the role they switched to. The user's original permissions are restored when the user exits the new role.

For example, suppose that you have a development and a production environment, and you create separate AWS accounts for the two environments. Your IAM users in the development account may occasionally need to access resources in the production account. Instead of creating separate identities in both environments for the users that need this type of access, you can enable cross-account access to those users. Here's how to do this:

1. In the production account, create a role named ModifyApp and define a trust policy for that role and attach it to the role. This allows an IAM user to assume the role. The trust policy defines the development account as a Principal. This allows authorized users from the development account to use the ModifyApp role. Next, you define a IAM permissions policy for this role that specifies the actions that the IAM user is allowed to perform when the user assumes this IAM role. In this case, the permission policy grants access to the production resources (such as an Amazon S3 bucket in the production environment). The role ARN for an account such as 123456789012 will then look like the following:
   ```
   arn:aws:iam:123456789012:role/ModifyApp
   ```
2. As the AWS SysOps administrator, you must grant the users in the development account permission to switch to the role you've created in step 1 by granting the group—say, the Developers group—permission to call the AWS Security Token Service `AssumeRole` API for the ModifyApp role that you created in step 1. This enables the users in the Development group to switch to the ModifyApp role in the production account.
3. The users in the Development group can now request a switch to the ModifyApp role by choosing **Switch Role** in the AWS console, or they can use the AWS API or AWS CLI to do the same.
4. AWS STS verifies the request with the role's trust policy and returns temporary credentials to the AWS console or to the application.
5. The temporary credentials returned by AWS STS grant the user access to the AWS resources in the production account. The AWS console uses these credentials on behalf of the user for subsequent console actions, such as accessing an S3 bucket in the production account. Similarly, if the user has switched the role through the AWS API or AWS CLI, the application uses the temporary credentials to update the S3 bucket(s) in the production account.

Managing IAM Authorization Policies

So far, you've learned about the identity part of AWS IAM. The second part of IAM, *access management*, also known as *authorization*, enables you to configure what users and other entities can do in their account. Authorization is how you specify permissions that allow or deny users access to AWS resources, as well as specify the actions they can perform in AWS.

You define these permissions with the help of IAM policies. Permissions policies help you control which users can access AWS resources and what actions they can perform on those resources. For example, let's say that you want your IAM users to be able to access the IAM console only from within the organization, and not from outside. You create an IAM policy with a condition that denies access when the IP address range is from outside the organization. The following example shows an IAM policy with a `Deny` condition that prevents access from any IP addresses that are specified as the source IP addresses (`"`AWS:SourceIp`"`):

```
.{ "Version": "2012-10-17",  "Statement": {    "Effect": "Deny",    "Action":
"*",    "Resource": "*",    "Condition": {"NotIpAddress": {"AWS:SourceIp": [
"192.0.1.0/24",      "202.0.112.0/24"    ]}}  }}
```

All IAM users start with zero permissions. So, by default, a user can't do anything in your AWS account until you explicitly attach a policy to that user or add that user to a group with the relevant permissions.

EXAM TIP Often an IAM-related question tests your knowledge of the default limits for IAM entities such as users, groups, and roles. Here are some key default limits:

Customer-managed policies in your AWS account: 1500
Groups in your AWS account: 300
Roles in your AWS account: 1000
Managed policies that can be attached to an IAM role or an IAM user: 10

There are two broad types of policies:

- **Permissions policies** A permissions policy is an object that defines an entity's or a resource's permissions. These are mostly stored as JSON documents, which you can attach to an IAM identity or an AWS resource (such as an Amazon S3 bucket) to define their permissions (allow or deny).
- **Permissions boundaries** Permissions boundaries are policies that limit the maximum amount of permissions that a user or entity (principals) can have.

Earlier in the chapter, you learned about a request context. When a principal seeks to perform an action on an AWS resource such as an EC2 instance, IAM looks in the request context for matching permissions policies to determine whether to allow the request.

EXAM TIP An IAM policy is a document that lists permissions. Understand the different entities an IAM policy can assign permission to. You can use a policy to grant permissions to a user, group, role, or a resource.

Permissions Policies

An IAM policy is a JSON-formatted document with one or more statements. A statement has the following structure and syntax:

```
{
...... "Statement":[{
        "Effect":"effect",
        "Action":"action",
        "Resource":"arn",
        "Condition":{
         "condition":{
          "key":"value"
          }
         }
        }
       ]
}
```

Following are the key elements of a statement:

- `"Effect"` This can take the value `Allow` or `Deny`. By default, users don't have permissions to use resources or perform API actions, so all requests are denied. An explicit deny in the `"Effect"` element overrides any allows, and an explicit allow overrides the default (deny).
- `"Action"` This is the specific API action for which the policy grants or denies permission.
- `"Resource"` This is the resource that's impacted by the action. You specify the resource by using its Amazon Resource Name (ARN).
- `"Condition"` This is an optional attribute that allows you do things such as controlling when your policy is in effect. A condition has one or more key-value pairs, and when there are multiple keys in a condition, AWS evaluates them using a logical AND operation. All conditions must be met before permission is granted. For example, you can have `ec2:AccepterVpc` as the condition key, and `"ec2:AccepterVpc":"vpc-arn"` as the key-value pair. Here, `vpc-arn` is the VPC ARN of the accepter VPC in a VPC-peering connection. You can also set a policy condition that requests must originate from specific IP addresses, or that they use SSL, for example.

EXAM TIP Be sure that you aren't confused as to what the different elements of an IAM policy statement do. The four key elements in an IAM policy statement are *effect, actions, resources*, and *condition*. The effect describes the result of a user request: either allow or deny. Actions refer to the operations that you allow. Any actions that you don't explicitly allow are denied. That means that a user can't access any resources to which you've not granted that user permissions. Each AWS service has its own set of actions, such as the `ListBucket` action in Amazon S3., which returns information about the objects stored in an S3 bucket. Resources are the entities over which you allow the actions. Condition is an optional element.

Following is an example of a permissions policy that permits all requests except those coming from specific IP addresses (NotIpAddress) to terminate EC2 instances using the AWS API or the AWS CLI. Remember that unless a permission is granted explicitly, AWS by default disallows all actions, so the users to whom you attach this policy won't be able to terminate the EC2 instances via the console.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["ec2:TerminateInstances"],
      "Resource": ["*"]
    },
    {
      "Effect": "Deny",
      "Action": ["ec2:TerminateInstances"],
      "Condition": {
        "NotIpAddress": {
          "aws:SourceIp": [
            "192.0.2.0/24",
            "203.0.113.0/24"
          ]
        }
      },
      "Resource": ["*"]
    }
  ]

}
```

Here's what you need to remember about how IAM authorizes requests through permissions policies:

- By default, all requests are denied.
- An explicit allow in a permissions policy overrides the default.
- A permissions boundary (an AWS Organizations secure control policy [SCP] or a user or role boundary) or a policy used during AWS Security Token Service (STS) role assumption overrides the allow.
- An explicit deny in a permissions policy overrides any allows.
- If a single policy in a request includes a denied action, IAM denies the entire request.

NOTE Requests that are made using the AWS account root user credentials are always allowed for all resources in that account.

Types of Permissions Policies

There are four broad types of AWS permissions policies:

- **Identity-based policies** Attach managed or inline policies to IAM identities such as users, groups, and roles.

- **Resource-based policies** Attach inline policies to resources, such as an Amazon S3 bucket policy.
- **AWS Organizations SCPs** Apply permissions boundaries to an organizational unit (OU) or an AWS Organizations organization.
- **Access control lists (ACLs)** Specify the principals that can access various resources.

NOTE You can attach multiple permissions policies to a principal. Each of the policies can contain multiple permissions.

Identity-Based Permissions Policies You can attach permissions policies to IAM identities such as users, groups, and roles. There are two broad types of identity-based policies: *managed policies* and *inline policies*.

Managed policies are permissions policies that you can attach to users, groups, and roles in an AWS account. Managed policies offer various benefits, such as the following:

- **Automatic updates** AWS automatically updates AWS-managed policies when necessary, such as adding permissions for a new AWS service, and applies those changes to the principals.
- **Reusability** You can attach the same managed policy to multiple principals.
- **Central change management** A change to a managed policy applies to all principals to which you attach that policy.
- **Versioning and rollback** IAM stores up to five versions of customer-managed policies. You can revert to an older policy version if you need to.
- **Permissions management delegation** You can allow users to attach and detach permissions policies.

There are two types of managed policies: AWS-managed policies and customer-managed policies.

AWS-managed policies are standalone policies maintained by AWS. A standalone policy has its own ARN:

```
arn:aws:iam::aws:policy/IAMReadOnlyAccess
```

In this ARN, IAMReadOnlyAccess is the name of the AWS policy.

AWS recommends that you start with the AWS-managed policies since you don't have to write the policies yourself. You simply assign the permissions to the user, groups, or roles. You can attach a single AWS-managed policy to principals in different AWS accounts. You typically use these policies for defining permissions for service administrators. You also use them to grant full (AmazonDynamoDBFullAccess) or partial access (AmazonEC2ReadOnlyAccess) to AWS services.

AWS-managed policies are especially useful when you design the policies to match common job functions, such as the PowerUserAccess policy, which grants full access to a user for every AWS service (with limited access to IAM and Organizations).

EXAM TIP You're very likely to see a question or two about IAM policies, with the question containing a sample IAM policy. You need to focus on the key policy elements *Effect, Allow, Action,* and *Resource* to determine the entitlements that the policy grants to a user or a group.

Here's an example IAM policy for an IAM group that grants full access to all AWS services (similar to a full administrator access) for the IAM users who are members of this group:

```
?{   "Version": "2012-10-17",   "Statement": [       {         "Effect":
"Allow",          "Action": "*",          "Resource": "*"         }    ]}
```

NOTE You can't change any of the permissions defined in an AWS-managed policy. AWS can update the permissions in a policy, say, when it launches a new AWS service.

Customer-managed policies are policies that you create and manage. These allow a more fine-grained control than AWS-managed policies. Customer-managed policies are also standalone policies that you attach to users, groups, and roles. Customer-managed policies enable you to enforce identical permissions to a group of users by attaching the policy to a group (or groups). You can easily edit a policy to apply policy changes to all members of the group(s).

An easy way to ensure that you've created a policy correctly is by copying an AWS-managed policy and customizing it. You can then assign the policy to one or more principals. As an AWS administrator, you can use IAM policies to control which users can create, update, and delete customer-managed policies in your AWS account. Following is an example policy that allows a user to create, update, and delete customer-managed policies in your AWS account:

```
{
 "Version": "2012-10-17",
 "Statement": {
  "Effect": "Allow",
  "Action": [
   "iam:CreatePolicy",
   "iam:CreatePolicyVersion",
   "iam:DeletePolicy",
   "iam:DeletePolicyVersion",
   "iam:GetPolicy",
   "iam:GetPolicyVersion",
   "iam:ListPolicies",
   "iam:ListPolicyVersions",
   "iam:SetDefaultPolicyVersion"
  ],
  "Resource": "*"
 }
}
```

The second major identity-based permissions policy type, *inline policies*, are policies that you can create and embed directly into a user, group, or role. Unlike managed policies (both AWS-managed and customer-managed), inline policies aren't standalone policies. These policies are *part of a specific principal*, such as a user, group, or rule. So when you delete the principal, the embedded policies in the principal entity are deleted automatically. Inline policies are helpful when you want to establish a strict one-to-one relationship between a permissions policy and a principal. Multiple principals can share the same inline policy that you create.

NOTE Inline policies are ideal when you need to grant access to AWS resources in your account to an external vendor. You create an AWS account for the user and restrict access to your AWS resources for just that user for a brief period, using an inline policy. When you delete the user, the embedded inline policies are deleted automatically, preventing the permissions in the policy from being inadvertently attached to other principals. AWS recommends that you use AWS-managed policies instead of inline policies in most cases.

Resource-Based Permissions Policies Resource-based policies attach inline policies to AWS resources, rather than to IAM identities. For example, you can attach policies to resources such as Amazon S3 buckets, Amazon SQS queues, and so on. With each resource, you specify which principal can access the resource and the actions they can perform on the resource. All resource-based policies are inline, not managed, policies.

NOTE Only AWS account root users have permissions to perform any actions on the resources in that account. Even if an IAM user creates a resource, the user doesn't have automatic permissions on that resource.

AWS is composed of collections of resources such as IAM users and Amazon S3 buckets. User requests specify a resource, a principal, a principal account, and the action the user wants to perform on the resource. This information, along with other necessary request information, is part of the *resource context*.

AWS Organization SCPs AWS Organizations is a service that groups and manages all the AWS accounts that your organization owns. SCPs are JSON policies that apply permissions boundaries that control the maximum services and the actions that entities in an AWS Organizations organization, or an organization unit (OU), can access. The permissions apply to the AWS root user as well.

Access Control Lists Amazon S3 supports ACLs as a permission mechanism. An ACL is a list of permissions attached to an object, such as a file or an Amazon S3 bucket. The ACL specifies which users or processes can access the objects, as well as what operations they're allowed to perform on the objects.

ACLs are independent of IAM policies and permissions, although you can use both together. ACLs are similar to resource-based permissions policies, but they are the only resource type that doesn't use the JSON policy document structure.

NOTE A resource-based permissions policy is different from resource-level permissions. While you can attach a resource-based policy directly to a resource, a resource-level permission enables you to use ARNs to specify the individual resources within a policy. Only some AWS services support resource-based permissions. If a user with specific resource-based permissions requests a resource with a permissions policy attached to it, AWS evaluates both sets of permissions before it determines whether to grant the user access to that resource.

Permissions Boundaries

Permissions boundaries change the effective permissions for a user or role. While permissions policies (identity, resource-based, ACLs) define permissions for the objects to which you attach them, permissions boundaries help you limit the maximum permissions for an IAM principal or AWS Organizations. Permissions boundaries don't grant any access on their own, but they can *limit* permissions provided by permissions policies.

NOTE You use either an AWS-managed policy or a customer-managed policy to set the maximum permissions for a principal (user or role) using permissions boundaries. Here's an example that shows how to use a policy to set a permissions boundary for an IAM user:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "ec2:*"
      ],
      "Resource": "*"
    }
  ]
}
```

NOTE You can use a permissions boundary only to limit a user's permissions. It doesn't grant any permissions on its own.

This example shows how you can allow the IAM user to perform actions only in Amazon S3 and Amazon EC2. Even if you create a policy allowing the action `iam:CreateUser` and grant the policy to this IAM user, the operation fails, since the permissions boundary doesn't allow operations in IAM.

Creating IAM Policies

You can create policies from the command line or via the console. Here's an example that shows how to create a managed policy with the `CreatePolicy` action

```
https://iam.amazonaws.com/?Action=CreatePolicy
&PolicyDocument={"Version":"2012-10-17","Statement":[{"Effect":"Allow","Action":"s3:ListA
llMyBuckets",
"Resource":"arn:aws:s3:::*"},{"Effect":"Allow","Action":["s3:Get*","s3:List*"],"Resource":
["arn:aws:s3:::EXAMPLE-BUCKET","arn:aws:s3:::EXAMPLE-BUCKET/*"]}]}
&PolicyName=S3-read-only-example-bucket
&Version=2010-05-08
&AUTHPARAMS
```

You can also create an IAM policy in the AWS console. Choose from one of the following methods to create a new IAM policy:

- **Import** Import and customize either an AWS-managed or a customer-managed policy that you've created.
- **Visual editor** Create the JSON document for the policy in the visual editor.
- **JSON** Create a policy using JSON syntax in the JSON tab. You'll do this in Exercise 3-2 later in this chapter.

NOTE Remember that you can create both customer-managed and inline policies, but not AWS-managed policies.

IAM Best Practices

AWS recommends the following best practices for securing your AWS resources.

Restrict and Protect the AWS Account Root User Access Key

Your AWS account root user access key (consisting of an access key ID and secret access key) gives access to all your AWS resources, including billing details. Therefore, you should not use your AWS account root user access key. Instead, use your account e-mail address and password to log into the console and create an IAM user that you can use for all administrative work. In addition, you may also want to delete or rotate your AWS account keys and use a strong password to protect the key.

Create Individual IAM Users

Create individual IAM users for working with your AWS account. This enables you to grant a different set of permissions for each IAM user, as needed.

Grant Least Privilege

Granting least privilege means that when you create IAM policies, you must limit the privilege grants to the permissions required to perform a task that each user needs to perform. For example, if a user needs only read-only access to a resource, don't grant the user write permissions. When granting permissions on S3 service, for example, allow only a small set of required users to access Amazon S3 write actions, which enable a user to put objects into an S3 bucket or delete buckets. The Access Advisor tab in the IAM console shows information about the services by user, group, and role. This information helps you identify and remove unnecessary IAM policies.

EXAM TIP Review the AWS security best practices, along with IAM-specific best practices. Most security best practices include creating IAM users and groups, using roles, configuring users and groups with policies that grant access based on the least privilege principle, securing remote administrative access, and configuring MFA for the root account as well as for all privileged IAM users.

Use Roles and Groups to Delegate and Assign Permissions

Instead of granting permissions directly to individual users, try to create groups based on job functions, and grant permissions for each group. Once you add users to a group, the users inherit all the permissions you assigned to the group. Groups help you easily grant and revoke permissions from users as their assignments in an organization change over time.

NOTE To pass a role and its permissions to an AWS service, you must have permissions to pass the role to the service. You must grant the IAM user, group, or role the `PassRole` permission to allow the entity to pass a role to an AWS service. You can use the same IAM role on multiple EC2 instances. However, you can associate only one IAM role with an EC2 instance.

Granting credentials to IAM roles rather than users helps secure your AWS services. Applications running on EC2 instance, for example, can use a role's credentials to access AWS resources.

Use AWS-Defined Policies to Assign Permissions

Use the managed policies created by AWS to grant permissions. As you introduce new services, AWS maintains and updates the policies.

Monitor Activity in your AWS Account

Regularly review and monitor all your IAM policies to enhance your security. Ensure that the policies follow the principle of least privilege as the policies evolve over time.

AWS Component Security

Some AWS resources have their own security mechanism that operates beyond IAM, which controls access to management tasks that are performed from the console, command line, or AWS SDKs. The following sections summarize the product-specific security for important AWS services.

Amazon EC2 Security

In EC2, you log into the instances with a key pair (Linux) or use a username/password (Windows). You use security groups to control the traffic to the instance.

AWS offers the following recommendations as security best practices for Amazon EC2:

- Use and identify federation, IAM roles, and IAM users to manage access to all AWS resources and APIs.
- Establish credentials-management policies and procedures for the creation, revocation, and rotation of AWS access.
- Implement the least permissive rules for your security group.
- Perform regular patches and updates of the OS and applications on your EC2 instances.

AWS offers multiple layers of security for EC2 instance, at the host and guest OS levels, and via firewalls and signed API calls. The following sections summarize the OS, network, and security features of EC2 instances.

Managing OS Access to EC2 Instances

In the shared responsibility model, you keep the OS credentials to your EC2 instances. AWS helps you with the initial access to the OS. When you launch an EC2 instance from a standard Amazon Machine Image (AMI), you must authenticate at the OS level to access and configure the instance. You can use secure remote access protocols such as Secure Shell (SSH) or Windows Remote Desktop Protocol (RDP) to access that instance.

Once you authenticate to your new EC2 instance, you can set up standard OS authentication mechanisms, such as local OS accounts, Microsoft Active Directory, and X.509 certificate authentication.

EC2 Key Pairs for Authentication to EC2 Instances

AWS provides asymmetric EC2 key pairs to enable you to authenticate to EC2 instances. You learned about key pairs in Chapter 2. Public key cryptography uses a public key to encrypt something such as a password, and the recipient uses the private key to decrypt the data.

The big difference between the AWS account and IAM user credentials versus the EC2 key pairs is this: you use your account and IAM credentials to manage access to other AWS services, whereas an EC2 key pair controls access to a specific EC2 instance that you launch in your account.

NOTE You can have multiple EC2 key pairs and launch new instances with different key pairs.

AWS can generate EC2 key pairs for you. When you launch an instance, AWS presents both the private key and the public key that are part of the key pair. You can generate new key pairs through EC2 anytime you want to.

On a Linux server, the public key is stored with the /.ssh/authorized_keys file. You must provide the private key when you connect to the instance. AWS doesn't save the private key, so if you lose it, you must generate a new key pair. Therefore, it's a good idea to secure the private key of the Amazon EC2 key pair.

Instead of having AWS generate a key pair for you, you can use a standard tool such as OpenSSH (with the `keygen` utility) to generate your own EC2 key pair. You import only the public key of the key pair into AWS and securely store the private key.

Each Linux instance launches with a default Linux system user account such as *ec2-user* (or Amazon Linux and Red Hat), and *ubuntu* for the Ubuntu operating system. Instead of granting access to an account such as *ec2-user* to multiple users, to enhance EC2 security, you must create multiple user accounts. Once you create the users, set up access keys for the users to log into EC2.

NOTE You mustn't distribute the private key file that you use for the root account to multiple users. If multiple users require access to the same EC2 instance, you should add user accounts (through IAM) to your instance and create a key pair for each user. Then distribute the private key files to your users.

Controlling Access to EC2 Instances

You can access all AWS services with your security credentials. In addition, you have unlimited use of your AWS resources, such as EC2 instances. However, you don't have to share your own security credentials with other users. Instead, use IAM and other EC2 features to control EC2 resource usage by other users, services, and applications. Security groups control access to EC2 instances, and IAM helps you control how users use resources in your AWS account. Following is a summary of how you control access to EC2 instances.

EC2 Security Groups for Linux Instances to Control Network Access A security group is a virtual firewall that controls traffic into and out of one or more EC2 instances. If you don't specify a custom security group when you launch an instance, the instance uses the default security group. Your AWS account has a default security group (named default) for the default VPC in each AWS region. The default rules for each default security group are as follows:

- Allow all inbound traffic for all the other instances associated with the default security group.
- Allow all outbound traffic from the instance.

NOTE You can't delete the default security group.

You add rules to a security group to control which traffic you want to accept or deny access to an instance or instances. AWS evaluates all rules from a security group in determining whether to allow traffic to go to an EC2 instance.

Security groups control both inbound and outbound traffic at the instance level. You must set up rules in your security group to enable you to connect to a Linux instance from your IP address, via SSH. Here's how you'd add the rule to the security group using the AWS CLI:

```
$ aws ec2 authorize-security-group-ingress --group-id security_group_id --protocol
tcp --port 22 --cidr cidr_ip_range
```

If you create an EC2 instance in a VPC, you must specify a security group created for that VPC. A security group is associated with the network interfaces, and changing the security groups for an instance changes the security groups associated with the primary network interface (eth0).

NOTE You can maintain your own firewalls on top of the security groups on any of your EC2 instances.

EC2 Permission Attributes You can specify which of your AWS accounts can use your AMIs and EBS snapshots. You configure an AMI's `LaunchPermission` attribute to specify the AWS accounts that can access an AMI. The `create VolumePermission` attribute of an EBS snapshot helps you control which AWS accounts can use a snapshot.

IAM and EC2 You can use IAM with EC2 to control which users can perform tasks using specific EC2 API actions, and whether they can work with specific AWS resources.

By default, your IAM users can't create or modify EC2 resources. They also can't perform tasks with the EC2 API, the console, or the CLI. To enable your IAM users to work with EC2 resources and perform tasks, you must first create IAM policies that grant the users permission to use resources and perform API actions. You must then attach these policies to the users for groups that need the permissions.

An EC2 IAM policy contains two things:

- It must grant or deny permission to use one or more EC2 actions.
- It must specify the resources that can be used with the action. Because EC2 only partially supports resource-level permissions, for some EC2 API actions, you can't specify the resources a user is allowed to work with for the action.

You can create an IAM group and attach a policy to that group. For EC2, you can use the following AWS-managed policies:

- PowerUserAccess
- ReadOnlyAccess
- AmazonEC2FullAccess
- AmazonEC2ReadOnlyAccess

Once you create a group with one or more of these AWS-managed policies, you can add users to that group. When you attach the policy to the group or user, it grants or denies permission to perform the tasks that you specify on specific resources in your AWS account.

The following EC2 IAM policy shows how you can control permissions granted to IAM users for EC2 instances. This policy is for requests from the Amazon CLI or an AWS SDK. You can also create policies for working in the Amazon EC2 console. This policy enables a user to describe all EC2 instances (each chunk starting with `"Effect":` is a separate statement), but to start and stop only two specific instances and terminate instances only in a specific region and with a specific resource tag:

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
  "Effect": "Allow",
   "Action": "ec2:DescribeInstances",
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:StopInstances",
    "ec2:StartInstances"
   ],
   "Resource": [
   "arn:aws:ec2:us-east-1:123456789012:instance/i-1234567890abcdef0",
   "arn:aws:ec2:us-east-1:123456789012:instance/i-0598c7d356eba48d7"
   ]
  },
  {
   "Effect": "Allow",
   "Action": "ec2:TerminateInstances",
   "Resource": "arn:aws:ec2:us-east-1:123456789012:instance/*",
   "Condition": {
     "StringEquals": {
      "ec2:ResourceTag/purpose": "test"
     }
   }
  }

  ]
}
```

This policy shows how to control access to EC2 instances. You can also create policies that control access to volumes, snapshots, reserved instances, and those that restrict access to specific AWS regions.

Securing the Operating System and Applications

In the shared responsibility model, you are responsible for both OS- and application-level security. AWS recommends that you standardize the OS and application builds and maintain the security configurations in a secure build repository. Furthermore, you should build preconfigured AMIs that satisfy security hardening standards that address known security vulnerabilities.

Best practices for OS and application security include the following:

- Rotate credentials such as access keys.
- Run regular privilege checks using IAM user's Access Advisor and access key last used.

- Disable password-only access and use MFA to gain access to instances.
- Use *bastion hosts* to enforce control. A bastion host acts as a jump server that lets users hop into your AWS environment to access secure servers running within your private subnets. Ideally, all access to EC2 instances should be through a bastion host.
- Password-protect the .pem file on user servers.
- Restrict access to EC2 instances to a select range of IPs, using security groups (these act as firewalls).
- Use SSH network protocol to secure login to your Linux EC2 instances.
- Disable the root API access keys.
- Disable remote root login.
- Use command-line logging.
- Use sudo for privilege escalation.
- Generate your own key pairs, and don't share them with other customers (or even with AWS).
- Delete unnecessary keys from the authorized keys file on your EC2 instances. An instance's neighbors thus don't have privileged access to the instance compared to any host on the Internet, meaning that you can treat them as belonging to different physical hosts.

EXAM TIP The SysOps certification exam often includes a question relating to bastion hosts, which are part of a security best practice adopted by many to secure the assets that they run in their private subnets. A bastion host is a server in your network that's specifically designed and configured to withstand security attacks. You generally host a single application such as a proxy server on a bastion host and remove other services to reduce the threat to the server. In AWS, you place a bastion host in a public subnet. Users can log into a bastion host via SSH or RDP and use their session to manage other hosts that live in private subnets.

Securing the Hypervisor

EC2 uses a custom version of the Xen hypervisor, which uses paravirtualization for Linux VMs. Under a paravirtualized system, the guest OS has no privileged access to the CPU, since the guest runs less privileged mode, called a *ring*. This demarcation of the guest and hypervisor means strong security for you.

Instance Isolation

AWS uses the Xen hypervisor to isolate the instances running on a physical machine. All network packets pass through the AWS firewall, which is in the hypervisor layer, helping to control traffic to the VM instance's virtual network interfaces. AWS enforces instance isolation in multiple areas, such as CPU memory.

Your EC2 instances use virtualized disks and have no access to the raw disk devices. The disk virtualization layer resets all used storage blocks, so a customer's data isn't exposed to other users. Memory is returned to the free memory pool for new memory allocation only after the memory is fully scrubbed. The hypervisor scrubs all the memory it reallocates to users.

NOTE For additional security, AWS recommends that you encrypt the file systems that you store on the virtual disks.

Who Secures the Host and Guest Operating Systems? AWS uses specially designed, configured, and hardened servers to serve as administration hosts. It tightly controls access to these servers with MFA and logs and audits the access.

You, the customer, completely control your VM servers. You have root access or administrative control over all accounts and servers and the applications that run on these servers. AWS has no access rights to your VMs.

NOTE AWS has no access rights to your instances and guest operating systems.

API Calls

Amazon EC2 API calls, such as those that terminate instances, must be signed by your Amazon secret access key. This key could be the AWS account secret access key, or an IAM user's secret access key.

TIP AWS recommends that you always encrypt all API calls to EC2 instances with SSL.

Mandatory Firewall

The extent of security provided by a firewall depends on the network ports that you open and the duration for which you leave them open. EC2 comes with a mandatory inbound firewall that by default is in the deny-all mode—that is, no inbound traffic is allowed until you open the necessary ports. You can restrict network traffic by protocol, service, port, and IP address.

You can group instances into various classes, so you can assign different rules for the instances. For example, you can open port 80 (HTTP) and, optionally, port 443 (HTTPS) for all your web servers. Instances functioning as database servers, by running the Oracle database, would have port 1521 open.

You can't control the firewall through your VM's operating system. You need your X.509 certificate and key to authorize all firewall changes, thus offering additional security. AWS recommends that you tighten security by restricting inbound and outbound network traffic with additional host-based firewalls such as iptables (or Windows Firewall) and VPNs.

Managing the Security of Your Custom AMIs

You can create and publish custom AMIs for both internal and external use. You are responsible for the security of the AMI and for ensuring that the private AMIs don't violate AWS's Acceptable Use Policy.

NOTE AWS reserves the right to remove an AMI from the public catalog if the AMI is in violation of security best practices or if it poses a significant risk to users running that AMI. You can review AWS's Acceptable Use Policy at http://aws.amazon.com/aup/.

Ensure that your published AMIs are patched with the latest security patches, and perform the following cleanup and hardening tasks to minimize your exposure, to protect your credentials, and to conform with good governance practices:

- Disable all nonessential network services on startup.
- Delete all AWS and third-party credentials from disk and configuration files.
- Delete all additional certificates or key material from the servers.
- Check to ensure that installed software doesn't use default internal accounts and passwords.
- Ensure that the server doesn't violate the AWS Acceptable Use Policy, such as using open SMTP relays or proxy servers.

NOTE Don't leave sensitive credentials on AMIs that you share publicly.

You must also perform additional OS-specific cleanup tasks before publishing the AMIs, such as the following steps for a Linux AMI:

- Configure SSHD to allow only public key authentication.
- Generate a unique SSH host key when you launch an EC instance.
- Remove or disable passwords for all user accounts.
- Delete all user SSH public and private key pairs.
- Delete all shell history and system log files that may contain sensitive data.

Once you initiate the hardened AMI by following the security best practices, you can use bootstrapping applications such as Puppet and Chef to modify and update the security controls. These security software updates include the application of server patches, service packs, and critical updates.

Managing Patches

As with the OS software, you are responsible for all patch management for both AMIs and EC2 instances. AWS recommends that you formalize patch management by keeping an inventory of all software and system components and checking to ensure that the security patches you installed match the latest vendor security patch list. You must also set up processes to track the latest security vulnerabilities, and rank the vulnerabilities, especially the most crucial and highest risk vulnerabilities.

You oversee the updating and patching of your guest operating systems. This includes the application of the latest security updates. AWS regularly updates the Amazon-provided AMIs with the latest patches, enabling you to relaunch the instances with the latest APIs that are updated with the current patches. You can also use Amazon Linux YUM repositories to update the Amazon Linux AMI–based instances.

Elastic Block Store Security

AWS protects data that you store on an EBS volume through redundancy. Since EBS replication is entirely within the same AZ, AWS recommends that you regularly take EBS snapshots and store them in Amazon S3 for durability. The data is stored on multiple physical locations without any extra cost to you. AWS further recommends that you use database-driven backups (for example, Oracle RMAN backups or MySQL database dumps) to protect the database data stored on EBS volumes.

It's a good idea to encrypt EBS volumes and snapshots. The encryption, which is performed in the EC2 instances, protects data as it moves between EC2 instances and EBS storage.

NOTE Only EC2's larger and more powerful instance types such as M3, C3, R3, and G2 offer encryption.

Securing the Analytics Services

AWS offers several analytics services, such as EMR and Kinesis, to help you store and analyze large volumes of data. The following sections briefly discuss the security aspects of key AWS analytics services.

Securing Amazon EMR

Amazon EMR is AWS's Hadoop web service that lets you run Hadoop clusters in the AWS cloud. Data for EMR is stored in Amazon S3, and EMR stores its job outputs in S3 as well. Amazon EMR creates two EC2 security groups, one group for each of the master nodes and the other for the worker nodes in the EMR cluster. You can use SSH port access to SSH into the master instances. Worker instances interact only with the master instances. By default, neither security group allows external access.

By default, other IAM users can't view an EMR cluster launched by an IAM user. You can choose to make the cluster viewable and accessible to all IAM users in your AWS account.

You can launch your EMR clusters into an Amazon VPC, so you can control access to the entire private subnet in which the cluster runs. You can enable the EMR cluster to access resources from your local data center using a VPN connection.

If you encrypt data before uploading it to S3, you must also provide a way for the EMR job to decrypt the data when retrieving the encrypted data from S3.

Securing Amazon Kinesis

Amazon Kinesis is an AWS-managed service that performs real-time data streaming. Kinesis helps in real-time data ingestion and processing of data in data sources such as server logs, social media, and web clickstream data.

You control access to Kinesis resources by creating IAM users and configuring IAM policies that specify user permissions. To facilitate running applications, use IAM roles. This helps applications to have the permissions you associate with an IAM role, without using long-term security credentials.

Securing the Network

AWS offers several networking services to help you create a secure, isolated network for your infrastructure, and to establish a private network connection to the AWS cloud.

Amazon Elastic Load Balancing Security

The AWS ELB service automatically distributes application traffic across multiple targets such as EC2 instances, containers, IP addresses, and Lambda functions. ELB offers a predefined cipher set used for TLS negations during the establishing of connections between ELB and clients. ELB offers additional configurations for TLS protocols and ciphers to suit your standards and requirements (such as PCI and SOX).

ELB offers perfect forward secrecy (PFS) for enhanced communication privacy. This strategy uses ephemeral session keys. ELB also enables you to identify the originating IP address of clients, which can be useful when you need more information about visitors while analyzing traffic logs or whitelisting certain IP addresses, for example.

ELB logs all HTTP and TCP requests sent to it, and the logs contain details such as the client IP addresses and ports, the sizes of the requests and responses, the backend IP address of the instance that served the requests, and finally the request line from the client, such as `GET http://www.mycompany.com:80/HTTP/1.1`.

Securing Amazon Virtual Private Cloud

You use Amazon VPC to create a distinct, isolated portion of the AWS cloud for your use. When you create a VPC, you specify a range of IP addresses for the subnet in the form of a Classless Inter-Domain Routing (CIDR) block. To establish external connectivity, you can create and attach an Internet gateway, a virtual private gateway, or both to the VPC.

You launch EC2 instances inside your VPC and define subnets within the VPC to group sets of EC2 instances together based on IP address ranges. You specify a separate CIDR block for each subnet that you create, and this block is a subset of the VPC's CIDR block. You can use routing and security groups to control traffic flow into and out of your subnets and EC2 instances.

In addition to subnets, Amazon VPC contains several other complementary security features, such as security groups, network ACLs, routing tables, and external gateways.

Subnets and Route Tables

Each EC2 instance in a VPC is connected to one subnet. AWS blocks well-known layer 2 security attacks such as MAC and ARP spoofing.

A subnet is associated with a routing table, which filters all network traffic flowing out of the subnet to determine the traffic's destination. You use a public subnet (the subnet's traffic is routed to an Internet gateway) to run web applications that need to connect to the Internet, and a private subnet (the subnet is not routed to the Internet gateway) to run private applications and databases that don't need to connect to the Internet.

Security Groups (Firewalls)

As with EC2 instances, you can create security groups for an Amazon VPC to filter both incoming and outgoing networking traffic from EC2 instances. Here's what you need to know about the differences between EC2 security groups and VPC security groups:

- You must create VPC security groups explicitly for your Amazon VPC. The EC2 security groups that you may have created won't work within the VPC.
- You can do some things with VPC security groups that you can't with EC2 security groups, such as change the security group after the instance launch or specify multiple protocols instead of being limited to TCP, UDP, and ICMP.

Network Access Control Lists

Network ACLs strengthen VPC security by filtering all inbound and outbound traffic from a subnet within a VPC. The ACL rules can control incoming and outgoing network traffic based on protocol, service, port, and the source or destination IP addresses.

Network ACLs and security groups complement each other, as shown in Figure 3-3.

Virtual Private Gateways and Internet Gateways

Network gateways enable connectivity between networks, including the Internet, and your VPC. A virtual private gateway enables you to connect privately to your VPC and other networks. You can connect securely to your VPC from your data centers by establishing a VPN connection to the virtual private gateway from on-premise gateway devices.

You can also attach an Internet gateway to your VPC to connect to the Internet, as well as to AWS services such as Amazon S3. To be able to connect through an Internet gateway, an EC2 instance must have one of the following:

- An elastic IP address that's associated with the instance
- A NAT instance to route the traffic

Dedicated EC2 Instances

By default, your EC2 instances may share the underlying hardware with other tenants. You can create a VPC with dedicated tenancy, forcing all instances that you launch in that VPC to be physically isolated from other tenants at the hardware level. You may also

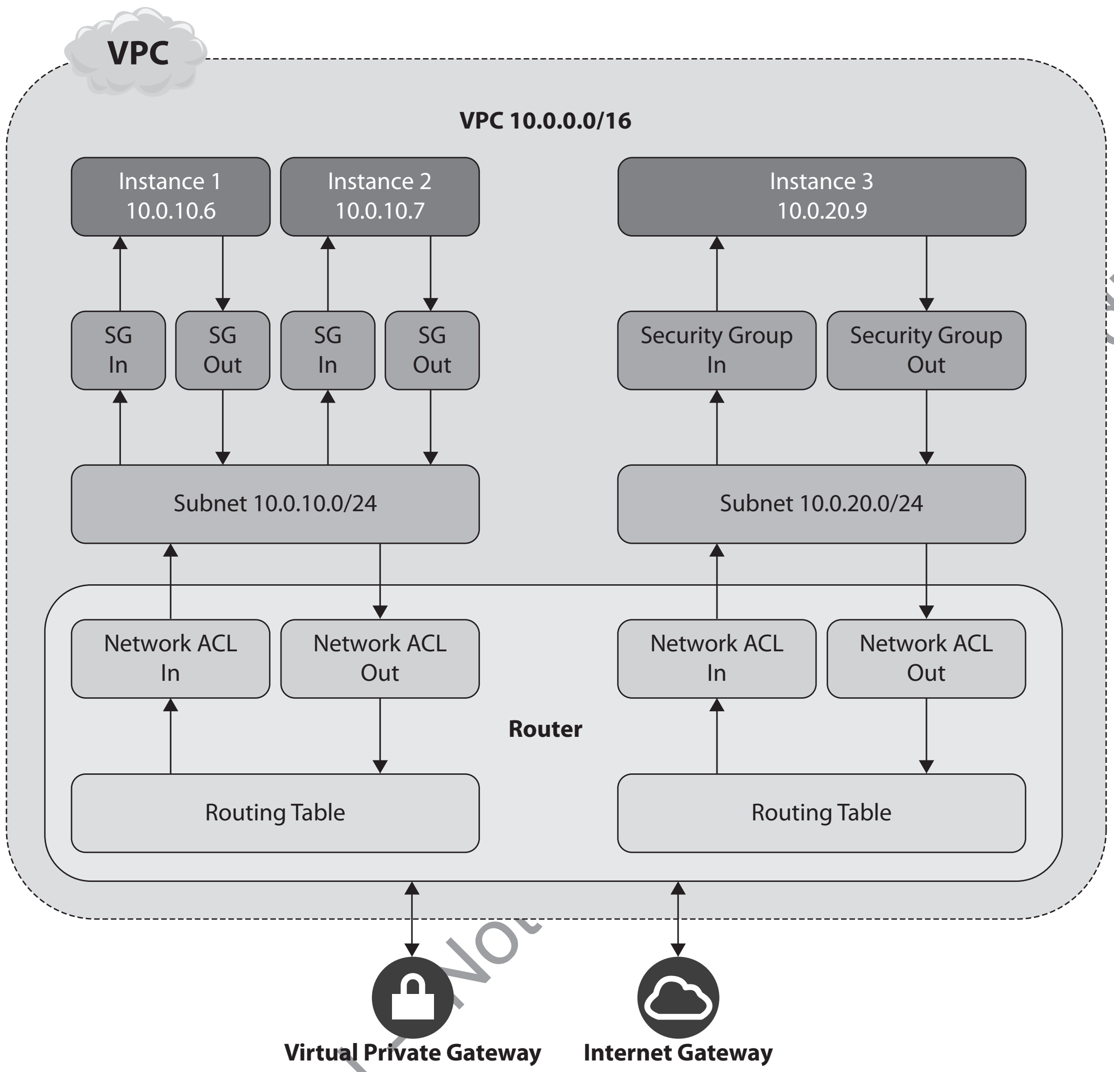

Figure 3-3 How network ACLs and security groups work together to protect instances within your Amazon VPC

specify that specific EC2 instances be created with dedicated tenancy. Be mindful of cost when selecting the dedicated instances.

Elastic Network Interfaces

All EC2 instances have a default network interface with a private IP address in your VPC network. You can add an elastic network interface (ENI) to an instance. ENIs are useful when creating a management network or using network and security appliances in your VPC, or to easily move the instances from one AZ to another.

Amazon Route 53 Security

Amazon Route 53 is Amazon's Domain Name System (DNS) that manages the IP addresses listed for your domain and translates domain names to IP addresses. The Route 53 control API is accessible only through SSL-encrypted endpoints. Route 53 requires all requests made to its control API to be authenticated, and the requests are signed with a signature calculated from the request and the secret access key of the user.

Amazon CloudFront Security

CloudFront is Amazon's content delivery network that delivers web content using a global network of edge locations. User requests for content are routed to the edge location nearest to them, substantially enhancing performance. CloudFront offers several security features to protect your static and dynamic web content.

Limiting Access to APIs

As with Route 53, CloudFront's control APIs are accessible only through SSL-encrypted endpoints. CloudFront also requires all requests made to its control APIs to be authenticated. The requests are signed with a signature calculated from the request and the secret access key of the user.

You can enable CloudFront's *private content feature* to limit who can download content from the service. CloudFront also supports geo restriction, which bases access to content on the viewer's location.

Controlling Access to Original Copies of Content

You can create origin access identities to control access of the original copies of your objects in S3. CloudFront uses the identities to retrieve objects from S3. To prevent the public from viewing the original copies of the object, you can limit access to the origin access identity by using S3 ACLs.

Restrict the Ability to Download Objects from CloudFront Locations

CloudFront uses a stringent system to control who can download objects from the service:

- Although, by default, CloudFront processes access requests in both HTTP and HTTPS, you can call for some objects to be transferred over an encrypted connection (HTTPS) only.
- All users must use a signed URL verification system that uses a public/private key pair.
- You may specify up to five trusted AWS accounts that can sign requests.
- You can create policy documents that specify under what conditions CloudFront serves the content. The document may include items such as the requested object name, the day and time the request was made, and the source IP of the client making the request.

NOTE By default, all content delivered by CloudFront is publicly readable, unless you enable the optional private content feature.

To reference your objects stored in CloudFront edge locations, you must include the encoded policy document and signature as query string parameters.

Enabling Access Logs You can enable CloudFront access logs by specifying an Amazon S3 bucket when you're configuring the CloudFront distribution.

Securing AWS Direct Connect

Direct Connect directly links your internal network in an AWS region through a dedicated connection that doesn't involve the public Internet; this lowers your network costs and enhances throughput and consistency. Direct Connect mandates the use of the Border Gateway Protocol (BGP) with an autonomous system number (ASN). You create virtual interfaces directly to the AWS cloud and Amazon VPC. To create the virtual interfaces, you must use either an AWS-generated BGP MD5 cryptographic key or provide your own key.

Securing the Storage Services

AWS offers several types of storage for backup and archiving. You can use Amazon S3 for object storage, for example, and Amazon Glacier for lower-cost archival storage.

Securing Simple Storage Service

Amazon S3 stores your data as objects inside storage buckets, with objects including various types of files, such as a text files or s video files, for example. By adding metadata when you store a file in S3, you can set permissions to control access to the file. S3 permissions can control the following:

- Access to a bucket to indicate who can create or delete objects in that bucket
- Viewing rights to a bucket and its objects access logs
- The choice of the geographical regions where S3 can store a bucket and its objects

Controlling Access to S3 Storage Buckets and Objects

An S3 bucket/object owner is the AWS account owner. The user who creates the bucket/object is *not* the owner of the object or bucket. Only an owner of a bucket or object can access the resources they create.

As the AWS account owner, you can grant or revoke access to S3 buckets and objects in three different ways:

- **IAM policies** Use IAM policies to control access of IAM users within your own AWS account to S3 buckets and objects. You attach S3 access policies to the IAM users in your AWS account.
- **ACLs** S3 access control lists (ACLs) help you manage access to S3 buckets and objects. Each bucket and object has an ACL attached to it that defines which AWS accounts or groups have access to the bucket or object. It's important to understand that ACLs allow you to grant access to AWS accounts and groups, but not to individual IAM users.

- **Bucket policies** S3 bucket policies are more broad-ranging that IAM policies and ACLs. Bucket policies enable you to control the access to some or all objects in an S3 bucket. You can attach a bucket policy to users and groups in your own AWS account, as well as other AWS accounts. You can also attach the bucket policies to S3 buckets.

EXAM TIP Carefully review the various types of Amazon S3 access policy options. Broadly speaking, you configure two types of S3 access policies: resource-based policies and user policies. You attach resource-based policies to your S3 resources, such as buckets and objects. Both bucket policies and ACLs are resource-based policies. You attach user policies to your IAM users. You can use user policies, resource policies, or a combination of the two to manage Amazon A3-related access permissions.

Granting Conditional Access to S3 Resources

You can use action-specific *policy keys* available in S3 to grant conditional access to specific S3 resources. You can use conditions such as request time, whether the request used SSL, the requester's IP address, or the requester's client application.

Query String Authentication

You can time-limit access to S3 objects through the use of *query string authentication.* This enables you to share your S3 objects through URLs that are valid for a specific length of time.

Protecting Data at Rest

You can manage your own encryption of S3 data by encryption data with the Amazon S3 encryption client (a client encryption library) before storing it in S3. You can also use Amazon S3 server-side encryption (SSE) to have Amazon S3 manage the encryption for you. You encrypt data with either an AWS-generated key or a key that you supply. Decryption is automatic upon retrieval of data.

TIP S3 metadata isn't encrypted; therefore, AWS suggests that you not put sensitive data in metadata.

S3 Access Logs

You can enable logging for an S3 bucket. Logging captures the access to the bucket and its objects, such as the requested resource, the request type (PUT, DELETE for example), the requestor's IP, and the time of the request.

Cross-Origin Resource Sharing (CORS)

Many web browsers use a same-origin policy that blocks JavaScript and HTML5 from allowing cross-origin requests from other sites or domains. You can enable CORS so that external web pages, style sheets, and HTML5 applications can safely access assets such as images stored in an S3 bucket.

Securing Glacier

Amazon Glacier is an archival service for infrequently used data. Archives are data such as photos, videos, and documents and can contain one or more files. Glacier stores the files as archives, inside vaults.

Security of Data Loads

You must compute and supply a tree hash to AWS when moving data to Glacier. A tree hash is a combination of hashes for each megabyte-sized data you send to Glacier, organized in a treelike fashion, to allow addition of new segments of that data. Glacier checks your tree hash to ensure that your data wasn't tampered with. Similarly, when you retrieve data from your Glacier archives, you can use the supplied checksums to ensure the file integrity.

Encryption

Glacier automatically encrypts all your data and offers an average annual durability of 99.999999999 percent for your archives. It protects your data by storing it in multiple data centers on multiple devices. It also performs regular data integrity checks and includes automatic self-healing features.

Securing the AWS Storage Gateway

The AWS Storage Gateway helps your local data center–based storage devices to connect with the Amazon S3 storage service. Using the Storage Gateway, you can securely upload data to S3 for backup and disaster recovery purposes.

The Storage Gateway moves offsite data to S3 storage via Amazon EBS snapshots, which are replicated over multiple AWS data centers and multiple storage devices. AWS stores the data in the region that you specify.

The Storage Gateway transfers data asynchronously from your data center–based storage devices to AWS via SSL. AWS then stores the data in S3, using the symmetric key advanced encryption standard Advanced Encryption Standard (AES) 256.

Securing AWS Import/Export

AWS Import/Export is a way to use portable storage devices to transfer large amounts of data to S3, EBS, or Glacier storage. You prepare the storage devices and ship them to a secure AWS facility and AWS transfers the data off the storage devices. You can similarly export data from AWS to a portable storage device.

To identify and authenticate your storage devices, you obtain a unique identifier for a job from AWS and a digital signature for authenticating the storage devices. AWS uses the signature file for authentication. When using S3, you place the signature file on your storage device's root directory, and for EBS, you affix the signature barcode to the device itself.

You can encrypt your storage devise in different ways, depending on whether you're importing or exporting and which AWS service (S3, EBS, or Glacier) you're using:

- **Importing to S3** Encrypt your storage device using a PIN-code device and/or TrueCrypt before shipping it. AWS decrypts the data and imports it into S3.

- **Exporting from S3** Provide a PIN code and/or a password so that AWS can encrypt your data on your storage device. You use a PIN-code device and/or TrueCrypt to decrypt the files.
- **Importing to Glacier or to EBS** Encrypt the data with your favorite encryption method before shipping the storage device. AWS doesn't decrypt your storage device.

Securing Databases

All AWS database services, including DynamoDB, RDS, Redshift, and ElastiCache, offer strong security features, as the following sections explain.

Securing DynamoDB

You can control who can access your DynamoDB resources and API by using IAM permissions policies. IAM policies help you control access at the resource level and the database level. Database-level permissions enable you to set up fine-grained access controls at the item (rows) and attribute (column) levels.

Instead of creating IAM users that can access DynamoDB, you can use web identity federation to grant access to users who authenticate by logging into Amazon, Facebook, or Google. IAM users log into an identity provider to obtain temporary security credentials from the AWS Security Token Service (STS). These temporary AWS credentials enable the applications to access specific DynamoDB tables.

All requests to the DynamoDB services must contain a HMAC-SHA256 signature. AWS SDKs automatically sign all your requests, but you must provide the signature when writing your own HTTP POST requests. To calculate the signature, you request temporary security credentials from AWS STS.

Securing RDS

You can secure access to your Amazon RDS DB instances and your RDS resources. You can use one or more of the following to control access to RDS.

Using a VPC to Control Network Access to RDS

For maximum network access control, run your DB instances in a VPC. Locating your RDS DB instances within a VPC enables you to run the instance in your own secure, private subnet. A common scenario for creating an RDS instance in a VPC is when you want the instance to share data with an application server running an EC2 instance within the same VPC.

To access a DB instance that lives in a different VPC from the one an EC2 instance lives in, you can use *VPC peering*. A VPC peering connection between two VPCs enables you to route traffic between the two VPCs using a private IP address, enabling instances to communicate as if they're within the same network. You can create VPC peering connections between your own VPCs or with a VPC in a different AWS account. The VPCs in a peering connection must all be in the same region, use security groups to control access, and must not have overlapping IP ranges. Chapter 6 discusses VPC peering.

Using IAM Policies to Control Access to Database Resources

Create IAM policies to grant permissions that control which users can manage RDS resources. You can, for example, create IAM policies to specify which of your users can create and delete DB instances.

Using DB Security Groups to Control Database Access

Use DB security groups to control which IP addresses can connect to your DB instances. A DB security group helps you secure DB instances running within your Amazon VPC by acting as a firewall to control network access to the DB instances. By default, when you create a database, the DB security group (firewall) disallows any database access unless you specify access through an existing EC2 security group rule. You must associate this security group with the DB instance.

NOTE A DB security group allows access to the database server port (such as 1521 for the Oracle database) and blocks all other ports.

Since the DB security group by default denies all access to the database, you must explicitly authorize network access by doing one of the following:

- Authorizing a network IP range
- Authorizing an existing EC2 security group

Using SSL Connections

An SSL connection encrypts the connections between your applications and your DB instances. Most of RDS's DB engines enable you to configure SSL connections to the DB instances, which enables the connections to be encrypted.

Encrypting Data

For an additional layer of data protection, you can use RDS encryption to encrypt data-at-rest in DB instances, automated backups, Read Replicas, and snapshots. RDS uses the standard AES-256 encryption algorithm to encrypt data. RDS handles the data decryption transparently, with little overhead.

For Oracle or SQL Server DB instances, you can also use Transparent Data Encryption (TDE) along with encryption-at-rest. For Oracle databases, you can use Oracle Native Network Encryption (NNE), which enables you to encrypt data as it moves via the network to and from an Oracle instance.

You use AWS Key Management Service (KMS) to create the encryption keys and define the policies for key usage. Remember that you must protect the use of the encryption keys by setting up appropriate access control policies. Encryption keys stored by KMS in a region can't be used in other regions, because key usage is limited to a specific region.

Using Built-in Database Security Features

All databases come with a host of built-in security features, such as password verification features. Use all the built-in security capabilities of your databases to protect your data.

When you create a DB instance in RDS, you'll create a master user account to manage the instance. The master user account is all powerful. After you create the database instance, you log into the database with the master user credentials and create additional user accounts, with limited capabilities.

Securing Redshift

Amazon Redshift, Amazon's petabyte-scale SQL data warehouse service, typically is run as a multinode cluster, with a leader node and two or more compute nodes. The leader node manages the connections and manages the query execution that the compute nodes perform. The compute nodes store data and perform the queries under the supervision of the leader node.

AWS runs the all-important compute node on a separate, isolated network, and you never directly access this node.

You can secure Amazon Redshift by controlling access to the database at four levels:

- Cluster management through IAM policies
- Cluster connectivity
- Database access
- Temporary database credentials and single sign-on (SSO)

Authentication and Access Control for Amazon Redshift

Any user who needs to access a Redshift database requires credentials that have permissions to access a Redshift cluster. You can control the ability of a user account to create, configure, and delete clusters by granting permissions to the users or accounts via IAM policies. Use IAM policies to control user access to a Redshift database and authenticate the user actions in the database.

The primary resource in Redshift is a cluster. There are also subresources, such as snapshots, parameter groups, and event subscriptions. Redshift provides a set of operations to use with the Redshift resources. A principal entity such as the root account, an IAM user, or an IAM role can create a resource such as a cluster. The resource owner is always the AWS account owner of the principal entity.

Earlier, you learned about IAM permissions policies such as identity-based permission policies, and policies that you attach to a resource, called resource-based policies. Redshift supports only identity-based IAM policies. You can attach these policies to IAM identities such as a user, group, or role. Customer-managed IAM policies can allow or deny access to RedShift actions and resources.

AWS provides several standalone IAM policies that it creates and administers. These AWS-managed policies provide permissions for common use cases. Here are the AWS-managed Redshift-specific policies that you can attach to your users:

- **AmazonRedshiftReadOnlyAccess** Grants read-only access to all Redshift resources in your AWS account
- **AmazonRedshiftFullAccess** Grants full access to all Redshift resources

- **AmazonRedshiftQueryEditor** Grants full access to the Query Editor in the Redshift console

Following is an example policy that allows an IAM user to create, delete, and modify a Redshift cluster.

```
[
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid":"AllowManageClusters",
   "Effect":"Allow",
   "Action": [
    "redshift:CreateCluster",
    "redshift:DeleteCluster",
    "redshift:ModifyCluster",
   ],
   "Resource":"*"
  }
 ]
```

When creating a policy, you can specify a condition such as one that specifies that a policy be applied only after a specific date. You can also use two condition keys to restrict access to resources based on their tags. The `redshift:Request` tag condition key applies to Redshift API actions that create a resource. The `redshift:Resource` tag restricts user access to resources based on tag keys and values.

Redshift uses IAM *service-linked roles*. A service-linked role is a predefined Redshift IAM role that includes the permissions the service requires to call AWS services on behalf of a Redshift cluster. When you create a cluster, Redshift creates a service-linked role in your account.

NOTE A service-linked role links to an AWS service, which is also called a *linked service* in this context. Only the linked service can assume the role, which means that you can't assume the role. You use a service-linked role to delegate permission to AWS services to manage AWS resources on your behalf.

The service-linked role named `AWSServiceRoleforRedshift` allows Redshift to call AWS services on your behalf. The role has predefined permissions that allow Redshift to do things such as describe VPCs and subnets.

Redshift Cluster Connectivity

By default, nobody has access to a Redshift cluster that you provision. To grant users access to the cluster, you must associate the cluster with a security group. The security group consists of rules that control cluster access, such as a range of IP addresses from which users can access Redshift. You can either use an existing VPC security group or define a new one and associate it with the cluster. You can associate multiple security groups with a cluster and also multiple clusters with a single security group.

Managing Redshift Database Security

Like all databases, Redshift offers built-in security features. Redshift logs all connection and use activity information for security purposes, a process called *database auditing*. It stores the logs in S3 buckets. The information in the logs is also stored in internal database system tables, but log files are easier to view than the log data stored in tables, since they don't require database permissions to query the data.

In addition to Redshift database audit logging, you can use AWS CloudTrail to view actions taken by a user, role, or AWS service in Redshift. You can find out the IP address from which a request was made to Redshift, as well as the time of the request and the identity of the requestor.

Database User Credentials

The common way to log into a Redshift database is by providing a database username and password. Instead of maintaining credentials in the Redshift database itself, you can allow users to create credentials to log into the database based on their IAM credentials. You can use IAM authentication to generate database user credentials. In other words, you can generate temporary database credentials based on permissions you grant through an IAM permissions policy.

The SQL client needs to call the `GetclusterCredentials` action on your behalf, and you must provide this authorization by creating an IAM user or role and attach a permissions policy that grants the permission to call the `redshift:getClusterCredentials`.

Redshift enables you to grant database permissions on a per-cluster level, and not on a per-table basis. The creator of an object is its owner, and only the owner can modify or delete the object. You can grant necessary permission to users or groups to use an object.

Backups

To protect the data, Redshift mirrors data on each of its nodes to disks on another node. In addition, it also backs up all its data to Amazon S3 through snapshots. You can have S3 store the backups from a period of 1 to 35 days.

Encrypting Redshift Databases

You can enable encryption for a Redshift cluster to protect data-at-rest. Although encryption is optional, AWS recommends that you use it for clusters with sensitive data, or when regulations such as the PPCI DSS and SOX require it.

There are two ways to handle the top-level encryption keys when you decide to encrypt Redshift databases. You can use the AWS Key Management Service (KMS) or an encryption hardware security module (HSM) to store the encryption keys.

Encryption with AWS KMS There is a four-tier hierarchy of encryption keys in AWS KMS:

- **Data encryption keys** These keys encrypt the data blocks in the Redshift cluster, using the database encryption key.
- **Database encryption key (DEK)** A randomly generated AES-256 key stored on disk separately from the Redshift database and encrypted by a master key.

- **Cluster encryption key (CEK)** Encrypts the database encryption key. You can use either AWS or a third-party HSM to store the CEK. HSMs enable you to manage keys separately from the application and the database.
- **Master key** Used for encrypting the cluster key, if you use AWS to store it (and not an HSM). If the cluster key is stored in an HSM, the master key encrypts the CEK-encrypted DEK.

NOTE *Key policies* are the main way to control access to the customer master keys (CMKs) in AWS KMS. You use key policies to configure who can access the keys in the KMS service.

Redshift randomly generates an AES-256 key that it uses as the DEK. It uses the decrypted DEK to encrypt and decrypt the keys for the data blocks in the Redshift database.

By default, Redshift uses the default key that's created for your AWS account (for use in Redshift) as the master encryption key. For more flexibility in creating, rotating, managing the access, and auditing the encryption keys that you use, you can create your own custom master key separately in AWS KMS.

After you choose a master key, Redshift requests AWS KMS to generate a CEK and encrypt it with the master key. AWS KMS stores the customer master key and sends the encrypted CEK to Redshift, which stores it on a separate network from the cluster.

Encryption Using an HSM Instead of using AWS KMS, you can use an HSM to manage encryption keys. As explained earlier in the section, "Storing and Managing Encryption Keys," an HSM is a device for generating and managing encryption keys. Because key management is separated from the database layer, an HSM offers more security.

Rotating the Encryption Keys You can rotate the encryption keys for an encrypted Redshift cluster. Redshift rotates the CEK for the cluster and its snapshots. It also rotates the DEK for the cluster, but it can't do so for the snapshots while they are stored in Amazon S3.

The frequency of key rotation depends on your regulatory compliance and industry standard requirements. A cluster is temporarily unavailable during the key rotation process because it's put in the `ROTATING_KEYS` state during this time. So be careful about frequently rotating the encryption keys!

Securing ElastiCache

Amazon ElastiCache is a web service that helps you set up and manage a distributed in-memory cache environment in the AWS cloud. ElastiCache helps web applications serve information faster by enabling them to retrieve information from a fast in-memory caching site, instead of slow disks.

ElastiCache lets you create cache clusters, which are collections of one or more cache nodes, each running an instance of the Memcached (or Redis) service. The cache nodes are chunks of network-attached RAM, and each cache node runs a Memcached (or Redis) service instance, with its own DNS name and port.

TIP All clients of an ElastiCache cache cluster must be a part of the EC2 network and authorized through ingress rules in cache security groups.

You can control access to an ElastiCache cluster by setting up cache security groups, which are like firewalls that control network access to the cache cluster. By default, network access is denied for all your cache clusters, and you must configure a cache security group and associate it with your cache clusters. By setting up ingress rules, you must explicitly enable access from hosts in the EC2 security group so your applications can access your cache cluster.

After you create a cache security group, you can execute the `authorize cache security group` ingress API or CLI command to authorize the EC2 security group. This determines which EC2 instances, and therefore which applications running on those instances, can access the cache cluster.

Data Security

ElastiCache offers encryption features for data on clusters running in Redis and Memcached. There are two types of encryption in ElastiCache: in-transit encryption and at-rest encryption.

In-Transit Data Encryption In-transit data encryption is designed to protect data moving between two places, such as the cluster and its applications, or between a primary node and a read replica node within a replication group.

In-transit encryption does the following:

- Encrypts connections
- Encrypts replicated data moving between primary and replica nodes
- Enables clients to authenticate that they're connecting to the right server
- Authenticates clients using a security feature such as Redis `AUTH`

NOTE The Redis `AUTH` command enhances security by requiring users to enter a password before they're allowed to execute Redis commands on a Redis server.

At-Rest Encryption At-rest encryption protects data stored on disk, including during backup operations. It's designed to enhance data security by encrypting data during sync, backup, and snapshot operations. Because of the processing involved in encryption/decryption, the overhead may impact performance.

Securing Network Access

ElastiCache is fully integrated with Amazon VPC, meaning that your cluster is automatically deployed into a VPC. If you don't define an ElastiCache security group for your VPC, ElastiCache uses the default security group. When creating an ElastiCache cluster in a VPC, you must specify a subnet group. ElastiCache will then choose the subnet and IP address to associate with your nodes. If you don't specify a subnet when launching the cluster, it's launched into your default VPC.

Application Services Security

AWS offers services to support applications, such as the Amazon Simple Queue Service (SQS) and the Amazon Simple Notification Service (SNS). Let's review how you secure the application services.

Securing Amazon SQS

Amazon SQS is a message queueing service that enables asynchronous messaging between a distributed application's components, such as EC2 instances and web servers. As with all other AWS services, SQS access requires credentials that have permissions to access resources such as SQS queues and messages. SQS has a resource-based permission system that's written in the same language as IAM policies. So you can achieve similar things with IAM and SQS policies. In most cases, the end results are the same with the SQS policy system or the IAM policy system.

The only SQS resource is the *queue*. SQL offers a set of actions that work with the queue resource.

Configuring Access with SQS Policies

Although you can use SQS policies to specify the AWS accounts that can work with it through IAM policies, SQS has a separate policy infrastructure. You can create SQS policies for a queue to control AWS account access to that queue. You can specify access conditions such as those that grant permissions to send or receive messages if the request is made before a certain date.

SQS doesn't support resource-based permissions policies; it supports only identity-based (IAM) policies. Here's an example policy:

```
{
  "Version": "2012-10-17",
  "Statement": [{
   "Effect": "Allow",
   "Action": "sqs:*",
   "Resource": "arn:aws:sqs:*:123456789012:sam_queue_*"
  }]
}
```

The SQS service defines a set of actions for each resource. SQS defines the set of actions you can specify in a policy to grant permissions for the actions. You can create SQS policies to allow users to create queues, to allow developers to write messages to shared queues, to allow a partner to send messages to a queue, and so on.

To grant permissions, you can specify conditions for when a policy should take effect by using the SQS Access Policy Language.

Protecting Data with Server-Side Encryption and AWS KMS

AWS doesn't encrypt data stored in Amazon SQS, but you can encrypt data before it is uploaded to Amazon SQS. SSE helps you transmit sensitive data in encrypted queues. It uses keys that are managed by AWS KMS to protect message contents in SQS queues. It's important to remember that all requests that are made to queues for which you've enabled SSE must use HTTPS Signature Version 4, which is a process that adds authentication information to AWS requests. When you use KMS, the data keys that encrypt the messages are encrypted as well as stored together with the data. SSE encrypts the message body in an SQS queue without affecting the normal functioning of SQS. To be able to use SSE for Amazon SQS, you must configure KMS key policies to allow the encryption techniques as well as messages that are stored in the queues. You can do this with either IAM policies or AWS SMS key policies.

You can choose to have SQS encrypt messages stored in both standard and first in, first out (FIFO) queues using an encryption key provided by AWS KMS. You can do this when you create your queue or later.

NOTE SSE encrypts the body of the message, but not the queue and message metadata or the per-queue metrics. Adding encryption to an existing queue doesn't encrypt the backlogged messages. Removing encryption doesn't affect any encrypted messages in the queue.

Securing Amazon SNS

Amazon SNS is a web service that enables users and applications to send and receive notifications from the AWS cloud. SNS coordinates and manages the delivery of messages to endpoints or clients. The two types of clients in SNS are called *publishers* and *subscribers* (or producers and consumers). Publishers send messages asynchronously to a *topic*, which is a communication channel, and subscribers such as web servers and AWS Lambda functions consume the messages by subscribing to the topic.

You control access to a topic by defining policies that specify which publishers and subscribers can use the topic. You allow an account to publish to a topic using the Amazon SNS API action `AddPermission`.

NOTE An SQS policy controls access to a queue, and an SNS policy controls access to a topic.

The key resource in SNS is a *topic*, and you must decide whether you want to grant AWS accounts the ability to perform specific topic actions, such as publishing messages to a specific topic. The SNS access control mechanism enables you to secure topics and messages against access by unauthorized users. A topic owner can set access policies for the topic to control who can subscribe or publish to it. The topic owner can also specify that all message delivery mechanisms must use HTTPS.

By default, access to a topic is limited to the AWS account that created the topic. You can allow other IAM users access to SNS through an SNS-generated policy or a policy that you configure.

AWS Monitoring Tools and Services that Help with Security

AWS provides several useful monitoring tools that help you secure your AWS infrastructure and AWS resources through event notifications and other means. Following are the key monitoring services that help with security:

- Amazon CloudWatch
- AWS Trusted Advisor
- Amazon Inspector
- AWS Config
- Amazon CloudTrail
- AWS Web Application Firewall (WAF)
- AWS Certificate Manager

Amazon CloudWatch

Amazon CloudWatch is a monitoring tool that helps you track your AWS resources and the applications that you run in AWS. CloudWatch collects and processes raw data into useful, near real-time, easily readable and understood performance and security metrics.

CloudWatch is a metrics repository. AWS services such as EC2 store operational metrics in this repository, and you retrieve standard summary statistics based on the metrics. You can also place custom metrics into this repository. You can present the summary metric data in the CloudWatch console in a graphical format. CloudWatch logs enable the monitoring of OS and other service logs. CloudWatch logs can also monitor CloudTrail logs in real time.

In Chapter 6, you'll learn about Amazon VPC Flow Logs, which capture information pertaining to the IP traffic flowing into and out of the network interfaces in your VPC. The CloudWatch Logs service stores the VPC Flow Logs data.

You can configure CloudWatch with alarm actions that stop, start, or terminate EC2 instances when certain criteria are met. An alarm can automatically initiate actions on your behalf.

AWS Trusted Advisor Tool

Some Premium Support plans include access to the AWS Trusted Advisor tool, which offers a snapshot of your AWS infrastructure and helps you identify potential security misconfigurations.

Trusted Advisor checks for compliance with the following security configurations:

- Configuration of IAM to ensure that AWS access controls are in place
- Enablement of MFA to provide two-factor authentication for your root AWS account
- Limited access to common administrative ports, such as the ports 22 (SSH), 23 (Telnet), 3389 (RDP), and 5500 (VNC)
- Limited access to common database ports, including ports 1433 (SQL Server), 3306 (MySQL), and Oracle (1521)

Amazon Inspector

Amazon Inspector is a security vulnerability service that helps you secure your AWS resources and comply with various security regulations. Amazon Inspector detects deviations from security best practices and security vulnerabilities in AW services and resources, and it produces a comprehensive list of findings ordered by severity level.

As with other security vulnerability tools, Amazon Inspector relies on a knowledge base of security rules that map to known security vulnerabilities and common security standard definitions. AWS security researchers maintain the security knowledge base and keep it up-to-date.

You can monitor Inspector with CloudWatch, which enables you to access historical information and understand how Inspector is performing. By default, Inspector sends data to CloudWatch every 5 minutes.

Following are the key features of Amazon Inspector:

- **Automation** You can fully automate Inspector via an API, including executing security checks and reporting their results.
- **Configuration scanning and activity monitoring engine** Inspector provides an engine that analyzes configuration of resources and monitors activity. This offers a comprehensive picture of the assessment targets and their security and compliance postures.
- **Built-in content library** Inspector contains a built-in library of rules that checks against best practices, common security standards, and known security vulnerabilities. It also provides detailed recommendations for resolving the security loopholes.

Service-Linked Roles for Amazon Inspector

Inspector is easy to set up because you don't need to add permissions manually. Inspector uses an IAM service-linked role, which is a unique type of IAM rule that links directly to Amazon Inspector. A service-linked role is predefined and includes all the permissions the service requires to all other AWS services on your behalf. Inspector defines the permissions of its service-linked roles, including the trust policy and the permissions policy.

AWS Config

AWS Config is a tool that offers a detailed view of your AWS resources, such as their configuration, their interrelationships, and how both have changed over time. You use AWS Config for troubleshooting, resource discovery, change management, audit compliance, and security analysis.

AWS Config helps you do the following:

- Get a snapshot of the current resource configurations.
- Retrieve the history of the configuration of the resources.
- Receive a notification when a resource is created, deleted, or altered.
- View interrelationships among resources—for example, finding all resources that share a security group.

AWS Config continually tracks resources as you create, delete, or modify them. When AWS Config detects that a resource isn't compliant with your security or other rules, it marks that resource as noncompliant and notifies you. For continuous configuration assessment, you can use AWS Config rules, both the set of prebuilt rules as well as custom rules of your own, to check the configuration changes that AWS Config captures.

AWS Config helps spot potential security weaknesses by providing detailed history information about key resource configuration, such as the IAM permissions that you've granted to users or the EC2 security groups you've configured to control access to AWS resources.

You can configure AWS Config to trigger automatic evaluation of rules when resource configurations change (say, a security group's rules are changed). You can also configure periodic evaluation of rules, such as every 24 hours or every 7 days.

AWS Config enables you to view the IAM policies assigned to users, groups, or roles. This helps you ascertain whether a user or role had permissions to perform certain actions at a specific time. By tracking the configuration of the EC2 security groups, you determine things such as whether specific port rules were open or whether incoming TCP traffic on that port was blocked at a certain time.

AWS CloudTrail

AWS CloudTrail is an AWS service that helps perform governance, compliance, and operations and risk auditing of an AWS account. All activity that occurs in an AWS account is recorded in a CloudTrail event. The activity could be an action that you, a rule, or an AWS service performs via the AWS Management Console, the AWS CLI, and AWS SDKs and APIs. Using CloudTrail, you can get a history of AWS API calls for your account.

CloudTrail helps you identify the users and accounts that called AWS APIs. It captures information about the API calls made to an AWS resource, such as the IP address the calls were made from and the time of the API calls. If the API call resulted in an error, CloudTrail shows the details of the error, including authorization failure messages if any. CloudTrail also captures all AWS console sign-in events and records when an AWS account user, federated user, or an IAM user signed into the console.

NOTE When you turn on CloudTrail logging, CloudWatch writes log files to the Amazon S3 bucket that you listed when configuring CloudWatch.

CloudTrail delivers event logs every 5 minutes to an Amazon S3 bucket that you specify. You can review recent events in the CloudTrail console by visiting Event History, where you can view and download the past 90 days of activity.

You can integrate CloudTrail into applications and automate trail creation for your organization. You can create a CloudTrail *trail* to archive changes to your AWS resources. A trail enables delivery of events to an Amazon S3 bucket you specify. In addition, you can deliver events to the CloudWatch Logs and CloudWatch Events services, and analyze the events there.

CloudTrail log files are stored indefinitely by default. The files are automatically encrypted using Amazon's S3 SSE. You can set up S3 lifecycle configuration rules to delete old logs automatically or move them to Amazon Glacier if you need long-term log storage.

CloudTrail Events

A CloudTrail *event*, which is in the JSON format, is a record of an activity in your AWS account performed by a user, role, or service. CloudTrail events offer a history of both API and non-API account activities. These actions could be from the AWS Management Console, AWS SDKs, CLI, or other AWS services. A trail enables the delivery of CloudTrail events to an S3 bucket, CloudWatch logs, or CloudWatch events. You can encrypt the event log files and set up SNS notifications for log file delivery. There are two types of trail events: *management events* and *data events*.

Management Events

Management events track management operations (also called control plane operations) you perform on AWS resources. For example, a management event can relate to configuration of security through an IAM `AttachRolePolicy` API operation or a network routing data configuration rule event (EC2 `CreateSubnet` API operation). Management events also can be non-API events such as when a user signs in to your AWS account, which is logged as the `ConsoleLogin` event.

Data Events

Data events, also called data plane operations, track the operations performed on your AWS resources. For example, a data event can be a `DeleteObject` API operation in Amazon S3. A data event can also be the Invoke API for executing AWS Lambda functions.

AWS Web Application Firewall

AWS WAF protects the applications from common web security exploits such as SQL injection and cross-site scripting. Using WAF, you determine which traffic is allowed or blocked with custom web security rules. WAF pricing is based on the number of rules you deploy and the number of web requests the applications receive.

NOTE You may deploy AWS WAF on CloudFront, the Application Load Balancer that fronts your web servers, or ElastiCache origin servers running on EC2.

Instead of creating your own set of security rules, you can use the managed rules for AWS WAF, a set of rules provided and managed by the AWS marketplace. These managed rules help secure your web applications and APIs against commons threats such as the OWASP (Open Web Application Security Project) Top 10 security risks and vulnerabilities listed in Common Vulnerabilities and Exposures (CVE).

AWS Certificate Manager

ACM is a service that helps you provision and manage public and private SSL/TLS certificates for using with various AWS services and your AWS resources. The SSL/TLS certificates help secure network communications and establish web site identity over the Internet.

You can request certificates and deploy them on AWS resources such as Elastic Load Balancing and CloudFront distributions and APIs on the API Gateway. ACM also helps create private certificates for your internal use and takes care of the certificate renewals.

Both public and private certificates that you provision through ACM for use with AWS services are free. If you use the ACM Private Certificate Authority, you pay a monthly charge for operating the private CA for the certificates that you issue.

Chapter Review

This chapter provided a thorough review of AWS security topics. You learned about the AWS shared responsibility security model, which explains how AWS and you share the security responsibilities for your AWS resources and AWS services running in the AWS cloud.

AWS Identity and Access Management helps you handle the two key security functions—authentication and authorization—through the creation of IAM principals such as users, groups, and roles, and granting these entities permissions to perform actions on AWS services.

AWS offers several ways to secure your network and storage services. Each of the database services offered by AWS, such as RDS, DynamoDB, and Redshift, provides built-in security measures that you should use.

AWS provides several monitoring tools and services to help you monitor AWS security and resource configuration, such as Amazon Inspector, AWS Config, and AWS CloudTrail.

Exercises

These exercises are designed to teach you how to perform important AWS security-related administrative tasks through the console. If you make a mistake, retrace your steps and ensure that you've correctly performed the steps. You'll know that you've correctly completed the exercise if you achieve the goal set for you in each exercise, such as creating an IAM user.

Exercise 3-1: Create an IAM user from the AWS Management Console.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose Users in the navigation pane, and then choose Add User.
3. Enter the username for the new IAM user (names aren't distinguished by case, and they must be truly unique).
4. Select the type of access you want to grant to the new IAM user. You can select one of the following, or both:
   - **Programmatic Access** If the user requires access to the API, AWS CLI, or the tools for Windows PowerShell
   - **AWS Management Console** If the user requires access to the console (this creates the password for the new user)
5. For the Console password type, choose Autogenerated Password or Custom Password. AWS recommends that you choose Require Password Reset to force users to change their password when they sign in the first time.
6. Choose Next:Permissions.
7. On the Set Permissions page, choose how you want to assign permissions to the new user. You must select from one of the following three options:
   - **Add User To Group** Add the new user to an existing group or choose Create Group to create a group.
   - **Copy Permissions From Existing User** Copy the group memberships, attached managed policies, embedded inline policies, and any existing permissions boundaries from a current IAM user to the new IAM user.
   - **Attach Existing Policies To User Directly** Attach an AWS-managed or custom-managed policy to the new user. Or choose Create Policy to create a new policy. AWS recommends, as a best practice, to attach policies to a group and make users members of the appropriate group.
     Optionally, you can set a permissions boundary by opening the Set Permissions Boundary section and choosing Use A Permissions Boundary to set the maximum user permissions.
8. Choose Next:Review to review your choices, and if you're satisfied, choose Create User.
9. Save the new user's access keys (access key ID and secret access key) by choosing Download .csv and saving the file. The new IAM user needs the secret keys to use the AWS API.

Exercise 3-2: Create a new IAM permissions policy from the AWS Management Console.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose Policies in the navigation pane.
3. In the Welcome To Managed policies page, choose Get Started.
4. Choose Create Policy.
5. Choose one of the following options and follow the steps in that option to create the policy:
   - Import Existing Managed Policies
   - Create Policy With The Visual Editor
   - Create Policies In The JSON Tab

After you create the policy, you can attach it to IAM users, groups, or roles.

Exercise 3-3: Create an Administrator IAM user and an Administrators Group from the console (create the group, assign the user to the group).

1. Log into the IAM console as the AWS account root user, using your AWS account e-mail address/password: https://console.aws.amazon.com/iam.
2. Select Users | Add User in the Navigation pane.
3. Type **Administrator** for the User Name.
4. Check the box next to AWS Management Console Access. Select Custom Password and enter a password in the text box.
5. Select Next:Permissions
6. In the Set Permissions page, select Add User To Group.
7. Select Create Group.
8. In the Create Group dialog box, type **Administrators** for Group Name.
9. For Filter Policies, select the check box for AWS Managed – Job Function.
10. In the policy list, select the check box for AdministratorAccess. Then select Create Group.
11. In the list of groups, select the new group (you may have to select Refresh before you see your new group appears in the list).
12. Select Next:Review to see the group memberships AWS IAM grants to the Administrator user. Select Create User.

Exercise 3-4: Create an IAM role.

1. Log into the IAM console as the AWS account root user, using your AWS account e-mail address/password: https://console.aws.amazon.com/iam/.
2. Select Role | Create Role.
3. For Role Type, select Another AWS Account.
4. Under Account ID, enter the AWS account ID for which you want to grant access.
5. Select Next:Permissions.
6. Select an existing AWS-managed or customer-managed policy in your account. Or select Create Policy to create a new policy, as in Exercise 3-2.
7. Select Next:Tagging.
8. Select Next:Review.
9. Under Role Name, type a name for the role (for example: devrole).
10. Review the role and select Create Role.

Exercise 3-5: Disable an IAM user's access keys via the console.

1. Log into the IAM console as the AWS account root user, using your AWS account e-mail address/password: https://console.aws.amazon.com/iam/.
2. Select Users.
3. Select the name of the user and choose the Security Credentials tab.
4. In the Access Keys section, select Make Inactive to disable the active access key.

Questions

The following questions will help you measure your understanding of the material presented in this chapter. Read all the choices carefully because there may be more than one correct answer. Choose all the correct answers for each question.

1. Which of the following AWS services can you use to track and visualize changes made to the resources in your AWS account?
   - **A.** AWS Config
   - **B.** Amazon Inspector
   - **C.** AWS CloudFormation
   - **D.** Amazon CloudTrail

2. When you are in the AWS cloud, which aspects of security is AWS responsible for? (Choose three)

  A. Redundant power supplies

  B. Fire detection and protection

  C. Climate control in the AWS data centers

  D. Database security

3. Which of the following credentials must an IAM user have to be able to log into the AWS Management Console and the AWS Command Line Interface (CLI)? (Choose two)

  A. E-mail address/password combination

  B. Username/password combination

  C. A key pair

  D. Access keys

4. Which of the following does AWS CloudTrail provide?

  A. Information about the incoming IP traffic

  B. Utilization information regarding the AWS resources in your account

  C. A trail of configuration changes made to AWS resources in your account

  D. Logs of the API requests for AWS resources in your account

5. Which of the following IAM permissions policies are standalone policies? (Choose two)

  A. Inline policy

  B. AWS-managed policy

  C. Customer-managed policy

  D. AWS built-in permissions policy

6. Which of the following statements are true regarding a CloudHSM? (Choose two)

  A. You and AWS share the responsibility for managing, monitoring, and maintaining the health of the CloudHSM appliance.

  B. AWS administrators manage, monitor, and maintain the health of a CloudHSM appliance.

  C. AWS initializes and manages the cryptographic domain of the CloudHSM.

  D. You initiate and manage the cryptographic domain of the CloudHSM.

7. Which of the following are AWS account security features? (Choose several)

  A. Key pairs

  B. X-509 certificates

  C. Access keys

  D. AWS Multi-Factor Authentication (MFA)

8. Which of the following aspects of security are your responsibility? (Choose two)
   - **A.** Providing a whitelist of users that can enter an AWS data center
   - **B.** Updating the hypervisors your EC2 instances run on
   - **C.** Configuring security groups to restrict access to EC2 instances
   - **D.** Configuring network ACLs to restrict access to and from the subnets in your Amazon VPC
9. Which of the following statements is/are true in securing Amazon SQS and Amazon SNS?
   - **A.** Both Amazon SQS and Amazon SNS encrypt data-at-rest.
   - **B.** Neither Amazon SQS nor Amazon SNS encrypts data-at-rest.
   - **C.** Amazon SQS encrypts data-at-rest, but Amazon SNS doesn't.
   - **D.** Amazon SNS encrypts data-at-rest, but Amazon SQS doesn't.
10. Which of the following AWS services handles centralized management of authentication and authorization of AWS users?
    - **A.** AWS Security Config
    - **B.** AWS Identity and Access Management service
    - **C.** Amazon Cloud Directory
    - **D.** AWS Key Management Service
11. Which of the following Amazon VPC security features can you implement to protect an RDS instance you're running inside an Amazon VPC? (Choose two)
    - **A.** Security groups
    - **B.** Network ACLs
    - **C.** Key pairs
    - **D.** Encryption with a hardware security module (CloudHSM)
12. Which of the following do you need to use to work with an AWS service from the AWS Command Line Interface (CLI)?
    - **A.** An access key ID and a secret access key
    - **B.** Your e-mail address for the AWS account and a password
    - **C.** Your private key and AWS's public key
    - **D.** Secret access ID and secret access key
13. Which of the following can you do to protect data-at-rest inside an Amazon DynamoDB database?
    - **A.** AWS-provided Server-Side Encryption (SSE)
    - **B.** Secure Socket Layer (SSL) connections
    - **C.** Appropriate AWS IAM permissions policies
    - **D.** Client-side encryption that you perform

14. Which of the following will enable you to send traffic security from your local data center network to your AWS resources within your Amazon VPC (in other words, what type of gateway can you use on the VPC)?

- **A.** Virtual private gateway
- **B.** Internet gateway
- **C.** Gateway VPC endpoints
- **D.** NAT gateway

15. Which of the following are AWS security best practices for securing AWS accounts? (Choose three)

- **A.** Creating individual IAM users and not using the root AWS account for routine work
- **B.** Requiring Multi-Factor Authentication (MFA) for root access
- **C.** Sharing AWS credentials to ensure secure cross-account access
- **D.** Disabling remote login for the root user

16. Which of the following does Amazon CloudTrail track and record?

- **A.** Flow of IP traffic passing through all the network interfaces in your VPC
- **B.** Configuration changes to AWS resources owned by you
- **C.** Resource utilization patterns for AWS resources owned by you
- **D.** All API request activity, such as who made the call, the time of the call, and the change made by the call

17. Which of the following is true when using AWS Identity and Access Management?

- **A.** All IAM members belong to a default user group if you don't place them in a custom group.
- **B.** An IAM user can be a member of a single group only.
- **C.** IAM roles can be members of a group.
- **D.** You can grant roles to a group or a member of a group.

18. Which feature of an Amazon VPC can you use to control network traffic flowing to an Amazon EC2 instance that lives in an Amazon VPC, based on the source IP address and port number?

- **A.** Main route table
- **B.** Security groups
- **C.** Subnets
- **D.** Network access controls lists (ACLs)

19. Which of the following statements are true regarding network access control lists and security groups? (Choose two)
    - **A.** Subnets use network access control lists to control the network traffic flowing into and out of the subnet.
    - **B.** EC2 instances use network access control lists to control the traffic flowing into and out of an EC2 instance.
    - **C.** Subnets use security groups to control the network traffic flowing into and out of the subnets.
    - **D.** EC2 instances use security groups to control the network traffic flowing into and out of the EC2 instances.
20. Which of the following logs does Amazon CloudWatch store? (Choose three)
    - **A.** EC2 operating system logs
    - **B.** AWS CloudTrail logs
    - **C.** Amazon VPC logs
    - **D.** Amazon RDS login activity logs
21. Which AWS monitoring and security tool performs automated security assessments to enhance the security and compliance of applications and operating systems that you deploy on Amazon EC2 instances?
    - **A.** AWS Trusted Advisor
    - **B.** Amazon Inspector
    - **C.** Amazon Config
    - **D.** Amazon Secure
22. Which of the following types of Multi-Factor Authentication (MFA) devices can you use for the AWS IAM service? (Choose three)
    - **A.** Amazon Simple Message Service (SMS) via mobile devices such as smartphones
    - **B.** Hardware devices such as Gemalto
    - **C.** Virtual MFA applications such as Microsoft Authenticator or Google Authenticator
    - **D.** Amazon Simple Queue Service (SQS)
23. Which of the following does Amazon CloudFront require as query string parameters in a request for content from the service?
    - **A.** Valid policy documents only
    - **B.** Matching signatures only
    - **C.** Valid policy documents and matching signatures
    - **D.** Valid policy documents, matching signatures, and an origin access identity

24. Which of the following are ways to control access to Amazon S3 buckets and objects? (Choose three)

- **A.** Identity and Access Management (IAM) policies
- **B.** Access control lists (ACLs)
- **C.** Security groups
- **D.** Bucket policies

25. Which of the following does the AWS Security Token Service (STS) help you do? (Choose two)

- **A.** Users who sign in with an identity provider can obtain temporary security credentials from AWS STS.
- **B.** You can use AWS STS to request temporary credentials when calculating the signature for your requests to the DynamoDB service.
- **C.** STS enables you to grant temporary security credentials for the new IAM users that you create.
- **D.** STS enables you to create fine-grained access controls for Amazon DynamoDB.

Answers

1. **A.** AWS Config enables you to track and visualize changes in resources.
2. **A, B, C.** AWS is responsible for the security of its cloud infrastructure. This includes the provision of redundant power supplies to ensure uninterrupted power, fire detection and protection, as well as climate control at the AWS data centers.
3. **B, D.** IAM users need an IAM username and password to log into the AWS Management Console. IAM users also require access keys to work with AWS services through the AWS CLI.
4. **D.** AWS CloudTrail logs all API requests made to all the AWS resources in your AWS account.
5. **B, C.** Both AWS-managed and customer-managed policies are standalone policies.
6. **B, D. B** is correct because the AWS administrators are responsible for managing and monitoring and maintaining the health of a CloudHSM appliance. **D** is correct because you initiate and manage the cryptographic domain of the CloudHSM.
7. **A, B, C, D.** All four of the choices are AWS account security features.
8. **C, D.** You are fully responsible for creating both security groups for controlling access to your EC2 instances, and configuring network ACLs for controlling traffic flowing into and out of the subnets in your Amazon VPC. AWS doesn't secure your VPC, subnets, EC2 instances, and services such as databases that you install and run on those instances.
9. **C.** You can configure SQS to encrypt messages stored in the SQS queues, but SQS does not encrypt data-at-rest.

10. **B.** AWS Identity and Access Management (IAM) service helps you manage authentication and authorization of AWS users through the provision of identities (principals) and permissions policies that control the actions the identities can perform on your AWS resources.
11. **A, B. A** is correct because you can configure rules in a security group to control traffic to the EC2 instances that host the RDS database instances. **B** is correct because network ACLs help you control the traffic flow to and from the subnet in which the EC2 instances are running.
12. **A.** You need an access key ID and a secret access key to work with AWS services from the AWS CLI.
13. **D.** You must perform client-side encryption of the data-at-rest in an Amazon DynamoDB database to protect sensitive data. DynamoDB doesn't support SSE.
14. **A.** To transmit data securely from your local data center network to your resources inside an Amazon VPC, you must use a virtual private gateway. All other gateways mentioned here aren't designed for safe and protected transmission of data.
15. **A, B, D.** Not using the root AWS account for performing routine tasks, requiring Multi-Factor Authentication for the root user, and disabling remote login capability for the root user are all part of AWS-recommended best practices for securing your AWS account.
16. **D.** Amazon CloudTrail records all API calls made to all AWS resources in your account. The information includes the identity of the user that made the API call, the day and time when the API call was made, and the specific changes made by the API call.
17. **D.** You can grant roles to an IAM group or to individual members of a group**.**
18. **D.** Network access controls lists (ACLs) enable you to control network traffic based on IP address and port number.
19. **A, D. A** is correct because you configure a network ACL to control network traffic flowing into and out of a subnet in your VPC. **D** is correct because you configure a security group to control the network traffic going into and out an EC2 instance.
20. **A, B, C.** Amazon CloudWatch stores the EC2 OS logs, AWS CloudTrail logs, and the Amazon VPC logs. It doesn't store any database login activity related logs.
21. **B.** Amazon Inspector uses a vulnerability database to check for known security vulnerabilities by performing automatic security assessment of the applications and operating systems that you deploy on Amazon EC2 instances.
22. **A, B, C.** You can use the Amazon SMS, a hardware device, or a virtual MFA application such as the Microsoft Authenticator to provide MFA.
23. **C.** A request for content from Amazon CloudFront must have valid policy documents and matching signatures as two of the query parameters for the service to honor the request.

24. A, B, D. IAM policies, ACLs (permissions policies), and bucket policies are all valid ways to control access to buckets and objects in Amazon S3.

25. A, B. A is correct because AWS STS returns temporary AWS credentials to applications that sign in to an identity provider. **B** is correct because AWS STS also grants you temporary security credentials to calculate the signature that you must provide to sign your requests to Amazon DynamoDB.

This page intentionally left blank