SCHOOL OF ENGINEERING

# AMAZON WEB SERVICES(AWS)

## MODULE 1

**Presentation Material**

**Department of Computer Science & Engineering**

**Course Code: 21CS4703** Semester: VII

**Course Title:** AMAZON WEB SERVICES(AWS) Year: IV

**Faculty Name:** Dr. Gokulakrishnan S

# Course Learning Objectives:

1. Understand fundamental of Cloud Computing and Amazon Elastic Compute Cloud and Amazon Elastic Block Store using AWS Platform

2.Devise the process of services Storage and Virtual Private Cloud in AWS.

3. Develop the Database services in AWS platform.

4. Recognize the importance of Authentication and Authorization of Security in AWS.

5. Apply Domain Name System and Network Routing for an EC2 Web Server.

# Course Outcome:

| Course Outcome | Description | Bloom's Taxonomy Level |
|---|---|---|
| At the end of the course the student will be able to: | | |
| CO1 | Utilize the fundamental concepts of Cloud computing, Amazon EC2, load balancing and Auto scaling in developing AWS cloud platform. | L3 |
| CO2 | Examine the services Storage and Virtual Private Cloud runs on AWS platform. | L3 |
| CO3 | Design and develop the latest Database Services on AWS Platform. | L3 |
| CO4 | Develop Amazon Authentication and Authorization, CloudTrial, Cloud Watch using AWS tools. | L3 |
| CO5 | Apply DNS, Network Routing, SQS. | L4 |

**TEXT BOOKS:**

1. Ben Piper, David Clinton, “AWS Certified Solutions Architect Study Guide: Associate SAA- C02 Exam (Aws Certified Solutions Architect Official: Associate Exam)” – February 2021.

**REFERENCE BOOKS:**

1. Learning Amazon Web Services (AWS): A Hands-On Guide to the Fundamentals of AWS Cloud,

Mark Wilkins, Pearson Education, November 2019.

2. AWS Certified Cloud Practitioner, by Anthony J. Sequeira, Pearson Education, August 2020.

**E-Resources:**

1. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html
2. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html
3. https://docs.aws.amazon.com/AWSEC2/latest/APIReference/OperationList-query-v
4. https://aws.amazon.com/cloudfront/?nc=sn&loc=0

| Exam | Total | Converted |
|---|---|---|
| CIA | 60 | 60 |
| SEE | 100 | 40 |

**Passing mark = CIA + SEE = 40**

| CIA Components | | |
|---|---|---|
| **CIA Components** | **Total** | **Converted** |
| **MSE1** | 40 | **20** |
| **MSE2** | 40 | **20** |
| **Assignment** | 10 | **10** |
| **Case Study** | 10 | **10** |

**AAT (Alternate Assessment)**

i. **First Component: Assignment for 10 Marks**
ii. **Second Component: Case Study for 10 Marks**

# MODULE 1: Contents

**Introduction:** Introduction to Cloud Computing and AWS: Cloud Computing and Virtualization, Cloud Computing Optimization, The AWS Cloud, AWS Platform Architecture, AWS Reliability and Compliance, Working with AWS, Amazon Elastic - Compute Cloud and Block Store: EC2 Instances, EC2 Storage Volumes, EC2 Auto Scaling, AWS Systems Manager, AWS CLI Example.

Textbook 1- Chapter 1:3-14, Chapter 2:21-51

# Introduction to Cloud Computing and AWS

Amazon Web Services offers a broad set of global cloud-based products including **compute, storage, databases, analytics, networking, mobile, developer tools, management tools, IoT, security, and enterprise applications**: on-demand, available in seconds, with pay-as-you-go pricing.

From data warehousing to deployment tools, directories to content delivery, over **200 AWS services** are available.

New services can be provisioned quickly, **without the upfront fixed expense**. This allows enterprises, start-ups, small and medium-sized businesses, and customers in the public sector to access the building blocks they need to respond quickly to changing business requirements.

# Introduction to Cloud Computing and AWS

In 2006, Amazon Web Services (AWS) began offering IT infrastructure services to businesses as web services—now commonly known as cloud computing.

One of the key benefits of cloud computing is the opportunity to replace upfront capital infrastructure expenses with low variable costs that scale with your business.

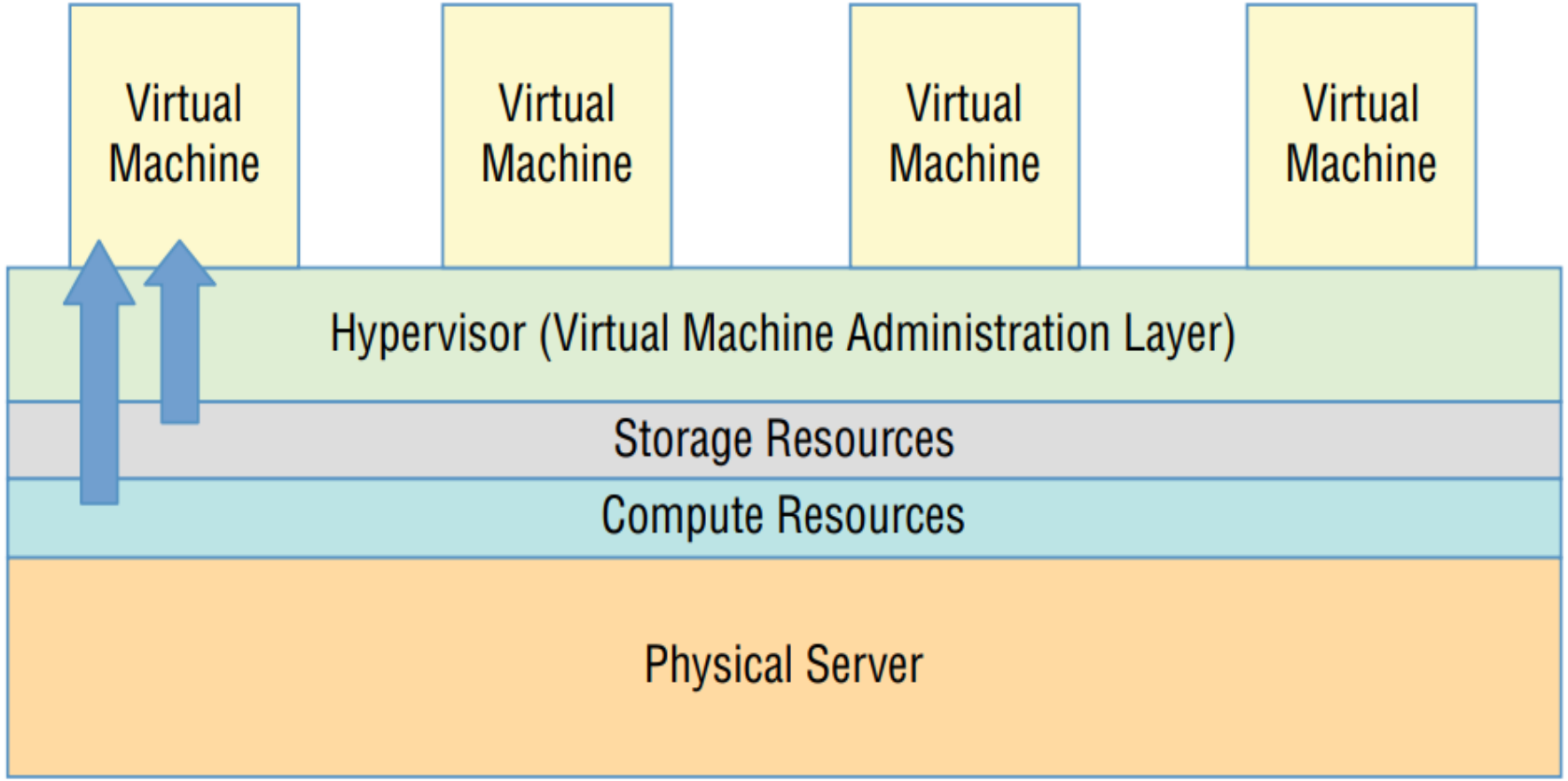
With the cloud, businesses no longer need to plan for and procure servers and other IT infrastructure weeks or months in advance. Instead, they can instantly spin up hundreds or thousands of servers in minutes and deliver results faster.

Today, AWS provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers hundreds of thousands of businesses in 190 countries around the world.

# Cloud Computing and Virtualization

- The technology that lies at the core of all cloud operations is virtualization.
- As illustrated in Figure 1.1, virtualization lets you divide the hardware resources of a single physical server into smaller units.
- That physical server could therefore host multiple virtual machines (VMs) running their own complete operating systems, each with its own memory, storage, and network access.

**FIGURE 1.1** A virtual machine host



# Cloud Computing and Virtualization

- Virtualization's flexibility makes it possible to provision a virtual server in a matter of seconds, run it for exactly the time your project requires, and then shut it down.
- The resources released will become instantly available to other workloads.
- The usage density you can achieve lets you squeeze the greatest value from your hardware and makes it easy to generate experimental and sandboxed environments.

# Cloud Computing Architecture

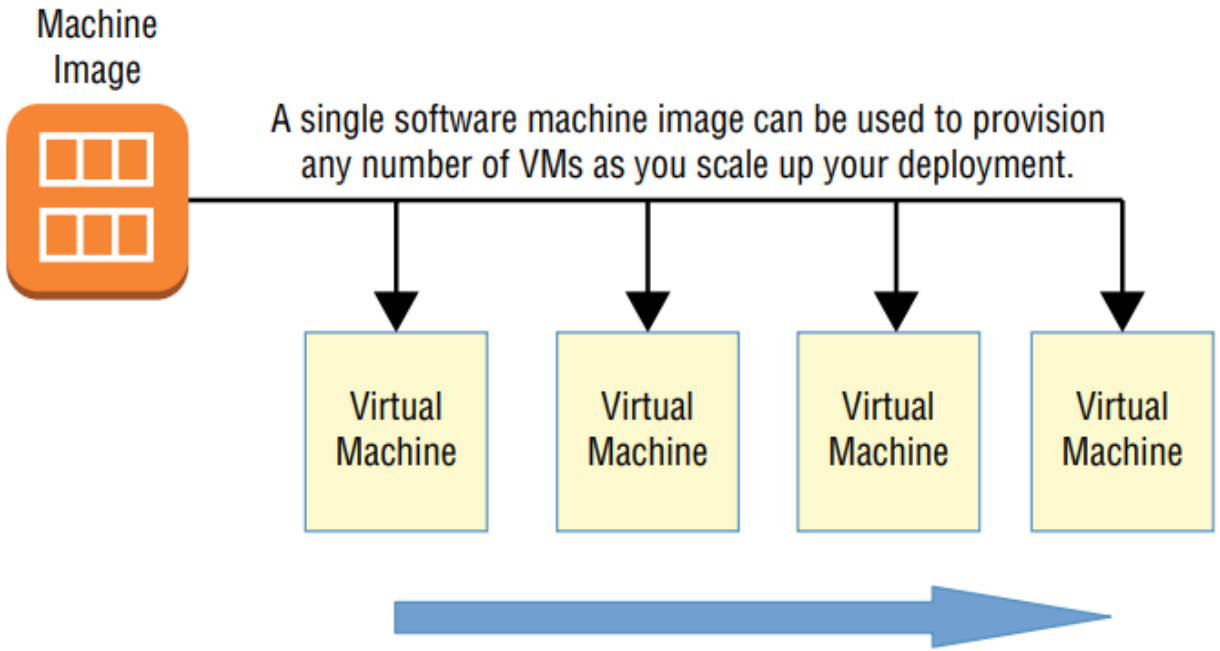
- Major cloud providers like AWS have enormous server farms where hundreds of thousands of servers and disk drives are maintained along with the network cabling necessary to connect them.
- A well-built virtualized environment could provide a virtual server using storage, memory, compute cycles, and network bandwidth collected from the most efficient mix of available sources it can find.
- A cloud computing platform offers on-demand, self-service access to pooled compute resources where your usage is metered and billed according to the volume you consume.
- Cloud computing systems allow for precise billing models, sometimes involving fractions of a penny for an hour of consumption.

# Cloud Computing Optimization

Effective deployment provisioning will require some insight into those three features.

- **Scalability** A scalable infrastructure can efficiently meet unexpected increases in demand for your application by automatically adding resources.
- AWS offers its autoscaling service through which you define a machine image that can be instantly and automatically replicated and launched into multiple instances to meet demand.

**FIGURE 1.2** Copies of a machine image are added to new VMs as they're launched.



# Cloud Computing Optimization

- **Elasticity**
- The principle of elasticity covers some of the same ground as scalability—both address how the system manages changing demand.
- However, though the images used in a scalable environment let you ramp up capacity to meet rising demand, an elastic infrastructure will automatically reduce capacity when demand drops.
- This makes it possible to control costs, since you'll run resources only when they're needed.

# Cloud Computing Optimization

- **Cost Management**
- Besides the ability to control expenses by closely managing the resources you use, cloud computing transitions your IT spending from a capital expenditure (capex) framework into something closer to operational expenditure (opex).
- In practical terms, this means you no longer have to spend $10,000 up front for every new server you deploy—along with associated electricity, cooling, security, and rack space costs.

# Cloud Computing Optimization

- To help you understand the full implications of cloud compute spending, AWS provides a free Total Cost of Ownership (TCO) Calculator at aws.amazon.com/tco-calculator.
- This calculator helps you perform proper “apples-to-apples” comparisons between your current data center costs and what an identical operation would cost you on AWS.

# The AWS Service Categories

**TABLE 1.1** AWS service categories

| Category | Function |
|---|---|
| Compute | Services replicating the traditional role of local physical servers for the cloud, offering advanced configurations including autoscaling, load balancing, and even serverless architectures (a method for delivering server functionality with a very small footprint) |
| Networking | Application connectivity, access control, and enhanced remote connections |
| Storage | Various kinds of storage platforms designed to fit a range of both immediate accessibility and long-term backup needs |

# The AWS Cloud

| Category | Function |
|---|---|
| Database | Managed data solutions for use cases requiring multiple data formats: relational, NoSQL, or caching |
| Application management | Monitoring, auditing, and configuring AWS account services and running resources |
| Security and identity | Services for managing authentication and authorization, data and connection encryption, and integration with third-party authentication management systems |

**TABLE 1.2** Core AWS services (by category)

| Category | Service | Function |
|---|---|---|
| Compute | Elastic Compute Cloud (EC2) | EC2 server instances provide virtual versions of the servers you would run in your local data center. EC2 instances can be provisioned with the CPU, memory, storage, and network interface profile to meet any application need, from a simple web server to one part of a cluster of instances providing an integrated multi-tiered fleet architecture. Since EC2 instances are virtual, they're resource-efficient and deploy nearly instantly. |
| | Lambda | Serverless application architectures like the one provided by Amazon's Lambda service allow you to provide responsive public-facing services without the need for a server that's actually running 24/7. Instead, network events (like consumer requests) can trigger the execution of a predefined code-based operation. When the operation (which can currently run for as long as 15 minutes) is complete, the Lambda event ends, and all resources automatically shut down. |
| | Auto Scaling | Copies of running EC2 instances can be defined as image templates and automatically launched (or *scaled up*) when client demand can't be met by existing instances. As demand drops, unused instances can be terminated (or *scaled down*). |
| | Elastic Load Balancing | Incoming network traffic can be directed between multiple web servers to ensure that a single web server isn't overwhelmed while other servers are underused or that traffic isn't directed to failed servers. |

# The AWS Cloud

| | | |
|---|---|---|
| Networking | Virtual Private Cloud (VPC) | VPCs are highly configurable networking environments designed to host your EC2 (and RDS) instances. You use VPC-based tools to secure and, if desired, isolate your instances by closely controlling inbound and outbound network access. |
| | Direct Connect | By purchasing fast and secure network connections to AWS through a third-party provider, you can use Direct Connect to establish an enhanced direct tunnel between your local data center or office and your AWS-based VPCs. |
| | Route 53 | Route 53 is the AWS DNS service that lets you manage domain registration, record administration, routing protocols, and health checks, which are all fully integrated with the rest of your AWS resources |
| | CloudFront | CloudFront is Amazon's distributed global content delivery network (CDN). When properly configured, a CloudFront distribution can store cached versions of your site's content at edge locations around the world so that they can be delivered to customers on request with the greatest efficiency and lowest latency. |

# The AWS Cloud

| | | |
|---|---|---|
| Storage | Simple Storage Service (S3) | S3 offers highly versatile, reliable, and inexpensive object storage that's great for data storage and backups. It's also commonly used as part of larger AWS production processes, including through the storage of script, template, and log files. |
| | S3 Glacier | A good choice for when you need large data archives stored cheaply over the long term and can live with retrieval delays measuring in the hours. Glacier's lifecycle management is closely integrated with S3. |
| | Elastic Block Store (EBS) | EBS provides the persistent virtual storage drives that host the operating systems and working data of an EC2 instance. They're meant to mimic the function of the storage drives and partitions attached to physical servers. |

| | | |
|---|---|---|
| Database | Relational Database Service (RDS) | RDS is a managed service that builds you a stable, secure, and reliable database instance. You can run a variety of SQL database engines on RDS, including MySQL, Microsoft SQL Server, Oracle, and Amazon's own Aurora. |
| | DynamoDB | DynamoDB can be used for fast, flexible, highly scalable, and managed nonrelational (NoSQL) database workloads. |
| Application management | CloudWatch | CloudWatch can be set to monitor process performance and resource utilization and, when preset thresholds are met, either send you a message or trigger an automated response. |
| | CloudFormation | This service enables you to use template files to define full and complex AWS deployments. The ability to script your use of any AWS resources makes it easier to automate, standardizing and speeding up the application launch process. |
| | CloudTrail | CloudTrail collects records of all your account's API events. This history is useful for account auditing and troubleshooting purposes. |
| | Config | The Config service is designed to help you with change management and compliance for your AWS account. You first define a desired configuration state, and Config evaluates any future states against that ideal. When a configuration change pushes too far from the ideal baseline, you'll be notified. |

# The AWS Cloud

| | | |
|---|---|---|
| Security and identity | Identity and Access Management (IAM) | You use IAM to administrate user and programmatic access and authentication to your AWS account. Through the use of users, groups, roles, and policies, you can control exactly who and what can access and/or work with any of your AWS resources. |
| | Key Management Service (KMS) | KMS is a managed service that allows you to administrate the creation and use of encryption keys to secure data used by and for any of your AWS resources. |

# The AWS Cloud

| Category | Service | Function |
|---|---|---|
| | Directory Service | For AWS environments that need to manage identities and relationships, Directory Service can integrate AWS resources with identity providers like Amazon Cognito and Microsoft AD domains. |
| Application integration | Simple Notification Service (SNS) | SNS is a notification tool that can automate the publishing of alert *topics* to other services (to an SQS Queue or to trigger a Lambda function, for instance), to mobile devices, or to recipients using email or SMS. |
| | Simple Workflow (SWF) | SWF lets you coordinate a series of tasks that must be performed using a range of AWS services or even non-digital (meaning, human) events. |
| | Simple Queue Service (SQS) | SQS allows for event-driven messaging within distributed systems that can decouple while coordinating the discrete steps of a larger process. The data contained in your SQS messages will be reliably delivered, adding to the fault-tolerant qualities of an application. |
| | API Gateway | This service enables you to create and manage secure and reliable APIs for your AWS-based applications. |

# AWS Platform Architecture

- https://aws.amazon.com/about-aws/global-infrastructure/regions_az/
- https://aws.amazon.com/about-aws/global-infrastructure/
- AWS maintains data centers for its physical servers around the world. Because the centers are so widely distributed, you can reduce your own services' network transfer latency by hosting your workloads geographically close to your users.
- It can also help you manage compliance with regulations requiring you to keep data within a particular legal jurisdiction.
- It's important to always be conscious of the region you have selected when you launch new AWS resources; pricing and service availability can vary from one to the next.

# The AWS Cloud

**TABLE 1.3** A list of publicly accessible AWS regions

| Region name | Region | Endpoint |
|---|---|---|
| US East (Ohio) | us-east-2 | us-east-2.amazonaws.com |
| US West (N. California) | us-west-1 | us-west-1.amazonaws.com |
| US West (Oregon) | us-west-2 | us-west-2.amazonaws.com |
| Asia Pacific (Hong Kong) | ap-east-1 | ap-east-1.amazonaws.com |
| Asia Pacific (Mumbai) | ap-south-1 | ap-south-1.amazonaws.com |
| Asia Pacific (Seoul) | ap-northeast-2 | ap-northeast-2.amazonaws.com |
| Asia Pacific (Osaka-Local) | ap-northeast-3 | ap-northeast-3.amazonaws.com |
| Asia Pacific (Singapore) | ap-southeast-1 | ap-southeast-1.amazonaws.com |
| Asia Pacific (Sydney) | ap-southeast-2 | ap-southeast-2.amazonaws.com |
| Asia Pacific (Tokyo) | ap-northeast-1 | ap-northeast-1.amazonaws.com |
| Canada (Central) | ca-central-1 | ca-central-1.amazonaws.com |
| China (Beijing) | cn-north-1 | cn-north-1.amazonaws.com.cn |
| China (Ningxia) | cn-northwest-1 | cn-northwest-1.amazonaws.com.cn |
| EU (Frankfurt) | eu-central-1 | eu-central-1.amazonaws.com |

# AWS Reliability and Compliance

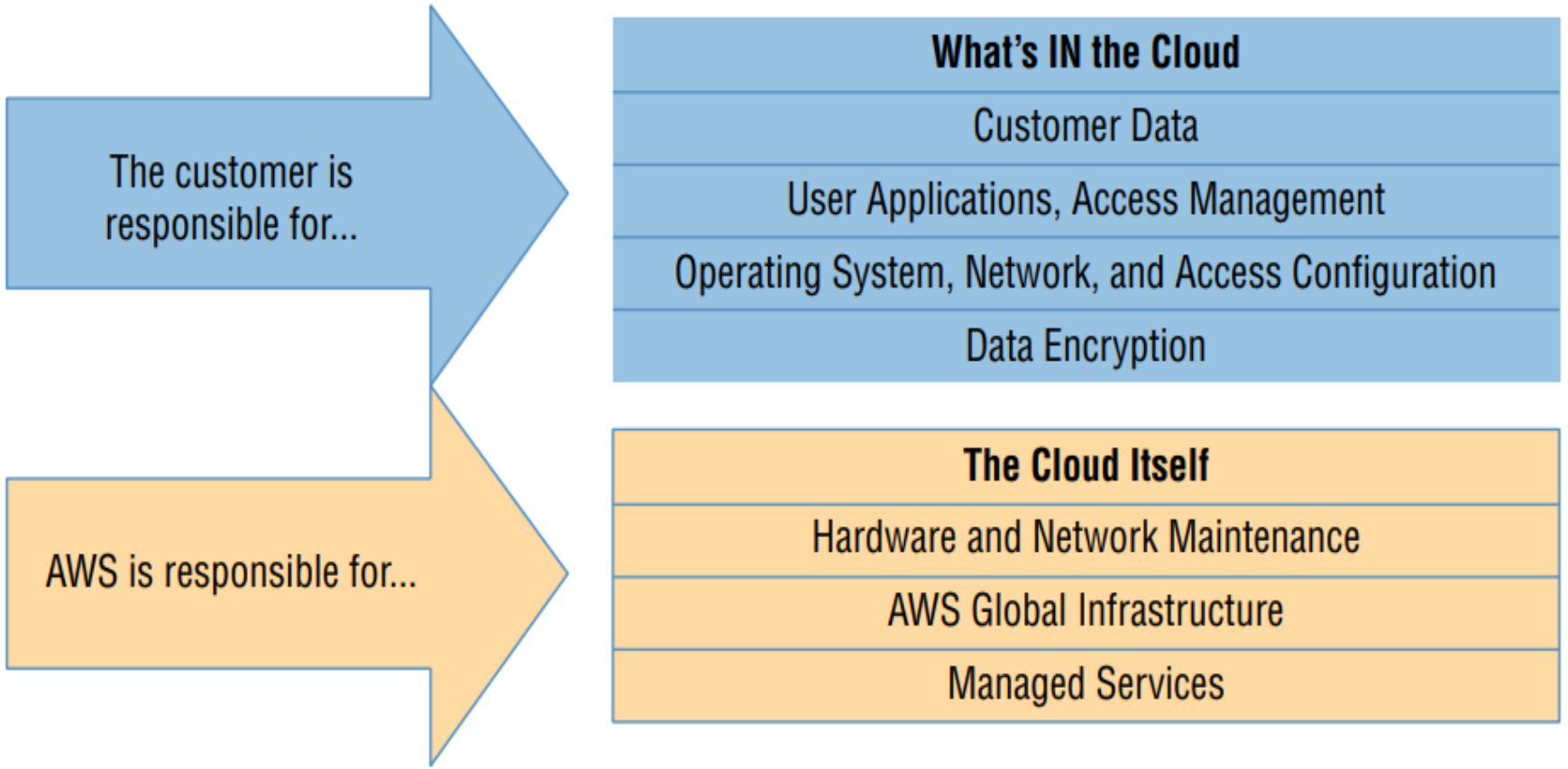


- AWS has a lot of the basic regulatory, legal, and security groundwork covered before you even launch your first service.
- AWS has invested significant planning and funds into resources and expertise relating to infrastructure administration.
- Its heavily protected and secretive data centers, layers of redundancy, and carefully developed best-practice protocols would be difficult or even impossible for a regular enterprise to replicate.
- Where applicable, resources on the AWS platform are compliant with dozens of national and international standards, frameworks, and certifications, including ISO 9001, FedRAMP, NIST, and GDPR. (See **aws.amazon.com/compliance/programs** for more information.)

# The AWS Shared Responsibility Model

- AWS guarantees the secure and uninterrupted operation of its "cloud." That means its physical servers, storage devices, networking infrastructure, and managed services.
- AWS customers, as illustrated in **Figure 1.3**, are responsible for whatever happens within that cloud.
- This covers the security and operation of installed operating systems, client-side data, the movement of data across networks, end-user authentication and access, and customer data.

**FIGURE 1.3** The AWS Shared Responsibility Model



# The AWS Service Level Agreement

- By "guarantee," AWS doesn't mean that service disruptions or security breaches will never occur.
- Drives may stop spinning, major electricity systems may fail, and natural disasters may happen.
- But when something does go wrong, AWS will provide service credits to reimburse customers for their direct losses whenever uptimes fall below a defined threshold.
- The exact percentage of the guarantee will differ according to service.

# The AWS Service Level Agreement

- The service level agreement (SLA) rate for AWS EC2, for instance, is set to 99.99 percent—meaning that you can expect your EC2 instances, ECS containers, and EBS storage devices to be available for all but around four minutes of each month.

# Working with AWS

- Whatever AWS services you choose to run, you'll need a way to manage them all.
- The browser-based management console is an excellent way to introduce yourself to a service's features and learn how it will perform in the real world.
- There are few AWS administration tasks that you can't do from the console, which provides plenty of useful visualizations and What's IN the Cloud Customer Data The customer is responsible for... AWS is responsible for... User Applications, Access Management Operating System, Network, and Access Configuration Data Encryption

# The AWS CLI



- The AWS Command Line Interface (CLI) lets you run complex AWS operations from your local command line.
- Once you get the hang of how it works, you'll discover that it can make things much simpler and more efficient.
- As an example, suppose you need to launch a half-dozen EC2 instances to make up a microservices environment.
- Each instance is meant to play a separate role and therefore will require a subtly different provisioning process.

# The AWS CLI

- Clicking through window after window to launch the instances from the console can quickly become tedious and time-consuming, especially if you find yourself repeating the task every few days.
- But the whole process can alternatively be incorporated into a simple script that you can run from your local terminal shell or PowerShell interface using the AWS CLI.
- Installing and configuring the AWS CLI on Linux, Windows, or macOS machines is not hard at all, but the details might change depending on your platform.
- For the most up-to-date instructions, see **docs.aws.amazon.com/cli/latest/userguide/installing.html**.

# AWS SDKs

- If you want to incorporate access to your AWS resources into your application code, you'll need to use an AWS software development kit (SDK) for the language you're working with.
- AWS currently offers SDKs for nine languages, including Java, .NET, and Python, and a number of mobile SDKs that include Android and iOS.
- There are also toolkits available for IntelliJ, Visual Studio, and Visual Studio Team Services (VSTS).
- You can see a full overview of AWS developer tools at **aws.amazon.com/tools**.

# Amazon Elastic Compute Cloud and Amazon Elastic Block Store

- **Introduction** The ultimate focus of a traditional data center/server room is its precious servers. But, to make those servers useful, you'll need to add racks, power supplies, cabling, switches, firewalls, and cooling.
- AWS's Elastic Compute Cloud (EC2) is designed to replicate the data center/server room experience as closely as possible.
- At the center of it all is the EC2 virtual server, known as an instance.

**This topic will explore the tools and practices used to fully leverage the power of the EC2 ecosystem, including the following:**

- ■ Provisioning an EC2 instance with the right hardware resources for your project
- ■ Configuring the right base operating system for your application needs
- ■ Building a secure and effective network environment for your instance
- ■ Adding scripts to run as the instance boots to support (or start) your application
- ■ Choosing the best EC2 pricing model for your needs
- ■ Understanding how to manage and leverage the EC2 instance lifecycle

**This topic will explore the tools and practices used to fully leverage the power of the EC2 ecosystem, including the following:**

- ■ Choosing the right storage drive type for your needs
- ■ Securing your EC2 resources using key pairs, security groups, network access lists, and Identity and Access Management (IAM) roles
- ■ Scaling the number of instances up and down to meet changing demand using Auto Scaling
- ■ Accessing your instance as an administrator or end-user client

# EC2 Instances

- An EC2 instance may only be a virtualized and abstracted subset of a physical server, but it behaves just like the real thing.
- It will have access to storage, memory, and a network interface, and its primary storage drive will come with a fresh and clean operating system running.
- It's up to you to decide what kind of hardware resources you want your instance to have, what operating system and software stack you'd like it to run, and, ultimately, how much you'll pay for it.

# EC2 Amazon Machine Images

- An AMI is really just a template document that contains information telling EC2 what OS and application software to include on the root data volume of the instance it's about to launch.
- There are four kinds of AMIs:
- **Amazon Quick Start AMIs** Amazon Quick Start images appear at the top of the list in the console when you start the process of launching a new instance. The Quick Start AMIs are popular choices and include various releases of Linux or Windows Server OSs and some specialty images for performing common operations (like deep learning and database). These AMIs are up-to-date and officially supported.
- **AWS Marketplace AMIs** AMIs from the AWS Marketplace are official, productionready images provided and supported by industry vendors like SAP and Cisco.

# EC2 Amazon Machine Images

- **Community AMIs** More than 100,000 images are available as community AMIs. Many of these images are AMIs created and maintained by independent vendors and are usually built to meet a specific need. This is a good catalog to search if you're planning an application built on a custom combination of software resources.

- **Private AMIs** You can also store images created from your own instance deployments as private AMIs. Why would you want to do that? You might, for instance, want the ability to scale up the number of instances you've got running to meet growing demand. Having a reliable, tested, and patched instance image as an AMI makes incorporating autoscaling easy. You can also share images as AMIs or import VMs from your local infrastructure (by way of AWS S3) using the AWS VM Import/Export tool.

# Instance Types

- AWS allocates hardware resources to your instances according to the instance type—or hardware profile—you select.
- The particular workload you're planning for your instance will determine the type you choose.
- The idea is to balance cost against your need for compute power, memory, and storage space.
- Ideally, you'll find a type that offers exactly the amount of each to satisfy both your application and budget.
- As listed in Table 2.1, there are currently more than 75 instance types organized into five instance families, although AWS frequently updates their selection.
- You can view the most recent collection at aws.amazon.com/ec2/instance-types.

**TABLE 2.1** EC2 instance type family and their top-level designations

| Instance type family | Types |
|---|---|
| General purpose | A1, T3, T3a, T2, M6g, M5, M5a, M5n, M4 |
| Compute optimized | C5, C5n, C4 |
| Memory optimized | R5, R5a, R5n, X1e, X1, High Memory, z1d |
| Accelerated computing | P3, P2, Inf1, G4, G3, F1 |
| Storage optimized | I3, I3en, D2, H1 |

## Instance Types

- **General Purpose** The General Purpose family includes T3, T2, M5, and M4 types, which all aim to provide a balance of compute, memory, and network resources. T2 types, for instance, range from the t2.nano with one virtual CPU (vCPU0) and half a gigabyte of memory all the way up to the t2.2xlarge with its eight vCPUs and 32 GB of memory. Because it's eligible as part of the Free Tier, the t2.micro is often a good choice for experimenting. But there's nothing stopping you from using it for light-use websites and various development-related services.

- **Compute Optimized** For more demanding web servers and high-end machine learning workloads, you'll choose from the Compute Optimized family that includes C5 and C4 types. C5 machines—currently available from the c5.large to the c5d.24xlarge—give you as much as 3.5 GHz of processor speed and strong network bandwidth.

# Instance Types

- **Memory Optimized** Memory Optimized instances work well for intensive database, data analysis, and caching operations. The X1e, X1, and R4 types are available with as much as 3.9 terabytes of dynamic random-access memory (DRAM)-based memory and low-latency solid-state drive (SSD) storage volumes attached.

- **Accelerated Computing** You can achieve higher-performing general-purpose graphics processing unit (GPGPU) performance from the P3, P2, G3, and F1 types within the Accelerated Computing group. These instances make use of various generations of high-end NVIDIA GPUs or, in the case of the F1 instances, an Xilinx Virtex UltraScale+ field-programmable gate array (FPGA—if you don't know what that is, then you probably don't need it). Accelerated Computing instances are recommended for demanding workloads such as 3D visualizations and rendering, financial analysis, and computational fluid dynamics.

# Instance Types

- **Storage Optimized** The H1, I3, and D2 types currently make up the Storage Optimized family that have large, low-latency instance storage volumes (in the case of I3en, up to 60 TB of slower hard disk drive [HDD] storage). These instances work well with distributed filesystems and heavyweight data processing applications.

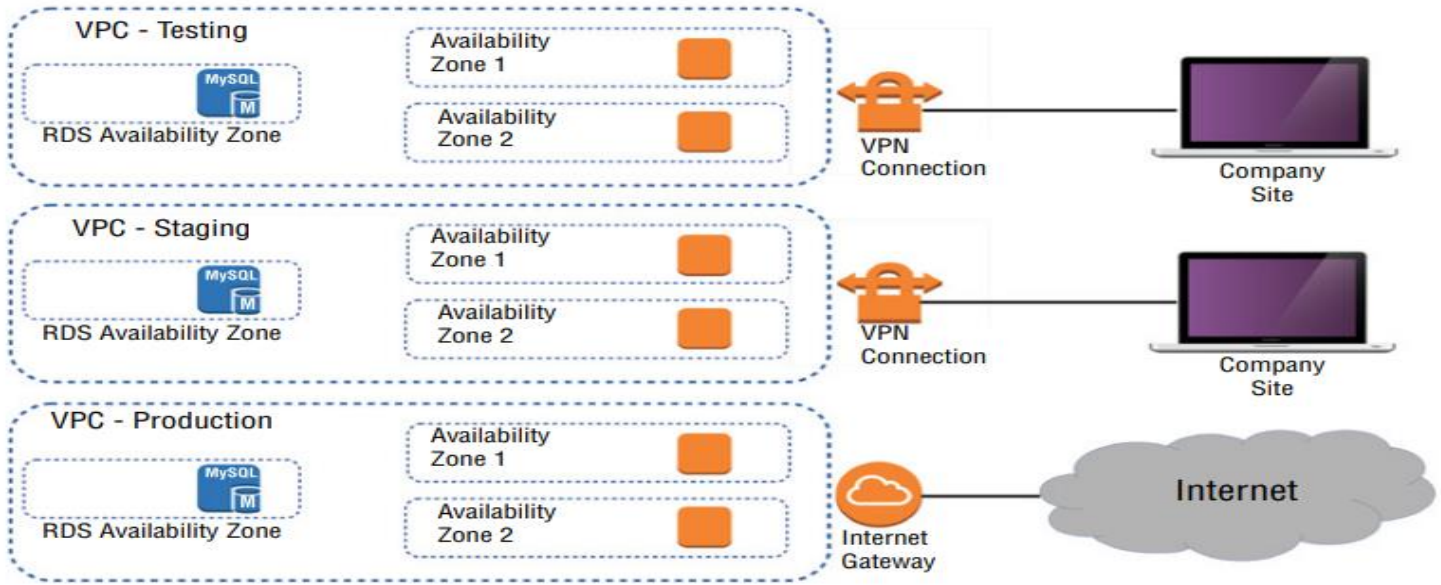
# Configuring an Environment for Your Instance

- Deciding where your EC2 instance will live is as important as choosing a performance configuration.
- Here, there are **three primary** details to get right: **geographic region, virtual private cloud (VPC), and tenancy model.**
- **AWS Regions** As you learned earlier, AWS servers are housed in data centers around the world and organized by geographical region. You'll generally want to launch an EC2 instance in the region that's physically closest to the majority of your customers or, if you're working with data that's subject to legal restrictions, within a jurisdiction that meets your compliance needs. EC2 resources can be managed only when you're "located within" their region. You set the active region in the console through the drop-down menu at the top of the page and through default configuration values in the AWS CLI or your SDK. You can update your CLI configuration by running aws configure.

# Configuring an Environment for Your Instance

- **VPCs Virtual private clouds (VPCs)** are easy-to-use AWS network organizers and great tools for organizing your infrastructure. Because it's so easy to isolate the instances in one VPC from whatever else you have running, you might want to create a new VPC for each one of your projects or project stages. For example, you might have one VPC for early application development, another for beta testing, and a third for production (see Figure 2.1).

**FIGURE 2.1** A multi-VPC infrastructure for a development environment



# Configuring an Environment for Your Instance

- **Tenancy** When launching an EC2 instance, you'll have the opportunity to choose a tenancy model. The default setting is shared tenancy, where your instance will run as a virtual machine on a physical server that's concurrently hosting other instances. Those other instances might well be owned and operated by other AWS customers, although the possibility of any kind of insecure interaction between instances is remote

# Configuring Instance Behavior

- You can optionally tell EC2 to execute commands on your instance as it boots by pointing to user data in your instance configuration (this is sometimes known as **bootstrapping**). Whether you specify the data during the console configuration process or by using the --user-data value with the AWS CLI, you can have script files bring your instance to any desired state. User data can consist of a few simple commands to install a web server and populate its web root, or it can be a sophisticated script setting up the instance as a working node within a Puppet Enterprise–driven platform.

# Placement Groups

- By default AWS will attempt to spread your instances across their infrastructure to create a profile that will be optimal for most use cases.
- But the specific demands of your operation might require a different setup. EC2 placement groups give you the power to define nonstandard profiles to better meet your needs.
- There are, at this time**, three kinds of placement groups**:

- ■ **Cluster groups** launch each associated instance into a single availability zone within close physical proximity to each other. This provides low-latency network interconnectivity and can be useful for high-performance computing (HPC) applications, for instance.

# Placement Groups

- **Spread groups** separate instances physically across distinct hardware racks and even availability zones to reduce the risk of failure-related data or service loss. Such a setup can be valuable when you're running hosts that can't tolerate multiple concurrent failures. If you're familiar with VMware's Distributed Resource Scheduler (DRS), this is similar to that.

- **Partition groups** let you associate some instances with each other, placing them in a single "partition." But the instances within that single partition can be kept physically separated from instances within other partitions. This differs from spread groups where no two instances will ever share physical hardware.

# Instance Pricing

- You can purchase the use of EC2 instances through one of three models.
- For always-on deployments that you expect to run for less than 12 months, you'll normally pay for each hour your instance is running through the on-demand model.
- Ondemand is the most flexible way to consume EC2 resources since you're able to closely control how much you pay by stopping and starting your instances according to your need.
- But, per hour, it's also the most expensive. If you're planning to keep the lights burning 24/7 for more than a year, then you'll enjoy a significant discount by purchasing a reserve instance—generally over a term commitment of between one and three years.

# Instance Pricing

- Table 2.2 gives you a sense of how costs can change between models. These estimates assume a Linux platform, all up-front payments, and default tenancy. Actual costs may vary over time and between regions.

**TABLE 2.2** Pricing estimates comparing on-demand with reserve costs

| Instance type | Pricing model | Cost/hour | Cost/year |
|---|---|---|---|
| t2.micro | On-demand | $0.0116 | $102.00 |
| t2.micro | Reserve (three-year term) | | $38.33 |
| g3.4xlarge | On-demand | $1.14 | $9986.40 |
| g3.4xlarge | Reserve (three-year term) | | $4429.66 |

# Instance Lifecycle

- The state of a running EC2 instance can be managed in a number of ways.
- Terminating the instance will shut it down and cause its resources to be reallocated to the general AWS pool.
- Terminating an instance will, in most cases, destroy all data kept on the primary storage.
- The exception to this would be an Elastic Block Store (EBS) volume that has been set to persist after its instance is terminated.

# Resource Tags

- The more resources you deploy on your AWS account, the harder it can be to properly keep track of things.
- Having constantly changing numbers of EC2 instances—along with accompanying storage volumes, security groups, and elastic IP addresses—all spread across two or three VPCs can get complicated.

# Resource Tags

**TABLE 2.3** A sample key/value tagging convention

| Key | Value |
|---|---|
| production-server | server1 |
| production-server | server2 |
| production-server | security-group1 |
| staging-server | server1 |
| staging-server | server2 |
| staging-server | security-group1 |
| test-server | server1 |
| test-server | security-group1 |

# Service Limits

- By default, each AWS account has limits to the number of instances of a particular service you're able to launch.
- Sometimes those limits apply to a single region within an account, and others are global.
- As examples, you're allowed only five VPCs per region and 5,000 Secure Shell (SSH) key pairs across your account.
- If necessary, you can ask AWS to raise your ceiling for a particular service.
- You can find up-to-date details regarding the limits of all AWS services at **docs.aws.amazon.com/general/latest/gr/aws_service_limits.html**.

# EC2 Storage Volumes

- Storage drives are for the most part virtualized spaces carved out of larger physical drives.
- To the OS running on your instance, though, all AWS volumes will present themselves exactly as though they were normal physical drives.
- But there's actually more than one kind of AWS volume, and it's important to understand how each type works.

# Elastic Block Store Volumes

- You can attach as many **Elastic Block Store (EBS)** volumes to your instance as you like (although one volume can be attached to no more than a single instance at a time) and use them just as you would hard drives, flash drives, or USB drives with your physical server.

- And as with physical drives, the type of EBS volume you choose will have an impact on both performance and cost. The AWS SLA guarantees the reliability of the data you store on its EBS volumes (promising at least 99.99 percent availability), so you don't have to worry about failure.

-

# Elastic Block Store Volumes

- When an EBS drive does fail, its data has already been duplicated and will probably be brought back online before anyone notices a problem.

- So, practically, the only thing that should concern you is how quickly and efficiently you can access your data. **There are currently four EBS volume types**, two using SSD technologies and two using the older spinning hard drives.
- The performance of each volume type is measured in maximum IOPS/volume (where IOPS means input/output operations per second).

# Elastic Block Store Volumes

- **EBS-Provisioned IOPS SSD** If your applications will require intense rates of I/O operations, then you should consider provisioned IOPS, which provides a maximum IOPS/volume of 64,000 and a maximum throughput/volume of 1,000 MB/s. Provisioned IOPS—which in some contexts is referred to as EBS Optimized—can cost $0.125/GB/month in addition to $0.065/provisioned IOPS.
- **EBS General-Purpose SSD** For most regular server workloads that, ideally, deliver low-latency performance, general-purpose SSDs will work well. You'll get a maximum of 16,000 IOPS/volume, and it will cost you $0.10/GB/month. For reference, a general-purpose SSD used as a typical 8 GB boot drive for a Linux instance would, at current rates, cost you $9.60/year.

# Elastic Block Store Volumes

- **Throughput-Optimized HDD** Throughput-optimized HDD volumes can provide reduced costs with acceptable performance where you're looking for throughput-intensive workloads, including log processing and big data operations. These volumes can deliver only 500 IOPS/ volume but with a 500 MB/s maximum throughput/volume, and they'll cost you only $0.045/GB/month.

- **Cold HDD** When you're working with larger volumes of data that require only infrequent access, a 250 IOPS/volume type might meet your needs for only $0.025/GB/month.

# Elastic Block Store Volumes

**TABLE 2.4** Sample costs for each of the four EBS storage volume types

| | EBS-provisioned IOPS SSD | EBS general-purpose SSD | Throughput-optimized HDD | Cold HDD |
|---|---|---|---|---|
| Volume size | 4 GB–16 TB | 1 GB–16 TB | 500 GB–16 TB | 500 GB–16 TB |
| Max IOPS/volume | 64,000 | 16,000 | 500 | 250 |
| Max throughput/volume (MB/s) | 1,000 | 250 | 500 | 250 |
| Price (/month) | $0.125/GB + $0.065/prov IOPS | $0.10/GB | $0.045/GB | $0.025/GB |

# EBS Volume Features

- All EBS volumes can be copied by creating a snapshot.
- Existing snapshots can be used to generate other volumes that can be shared and/or attached to other instances or converted to images from which AMIs can be made.
- You can also generate an AMI image directly from a running instance-attached EBS volume—although, to be sure no data is lost, you should shut down the instance first.
- EBS volumes can be encrypted to protect their data while at rest or as it's sent back and forth to the EC2 host instance.
- EBS can manage the encryption keys automatically behind the scenes or use keys that you provide through the AWS Key Management Service (KMS).

# Instance Store Volumes

- Unlike EBS volumes, instance store volumes are ephemeral.
- This means that when the instances they're attached to are shut down, their data is permanently lost.
- So, why would you want to keep your data on an instance store volume more than on EBS?
- ■ Instance store volumes are SSDs that are physically attached to the server hosting your instance and are connected via a fast NVMe (Non-Volatile Memory Express) interface.
- ■ The use of instance store volumes is included in the price of the instance itself.
- ■ Instance store volumes work especially well for deployment models where instances are launched to fill short-term roles (as part of autoscaling groups, for instance), import data from external sources, and are, effectively, disposable.

# Accessing Your EC2 Instance

- Like all networked devices, EC2 instances are identified by unique IP addresses. All instances are assigned at least one private IPv4 address that, by default, will fall within one of the blocks shown in Table 2.5.

**TABLE 2.5** The three IP address ranges used by private networks

| From | To |
|---|---|
| 10.0.0.0 | 10.255.255.255 |
| 172.16.0.0 | 172.31.255.255 |
| 192.168.0.0 | 192.168.255.255 |

# Securing Your EC2 Instance

- You are responsible for configuring appropriate and effective access controls to protect your EC2 instances from unauthorized use.
- Broadly speaking, AWS provides four tools to help you with this task: security groups, Identity and Access Management (IAM) roles, network address translation (NAT) instances, and key pairs.

- **Security Groups** An EC2 security group plays the role of a firewall. By default, a security group will deny all incoming traffic while permitting all outgoing traffic. You define group behavior by setting policy rules that will either block or allow specified traffic types. From that point on, any data packet coming into or leaving the perimeter will be measured against those rules and processed accordingly

## IAM Roles

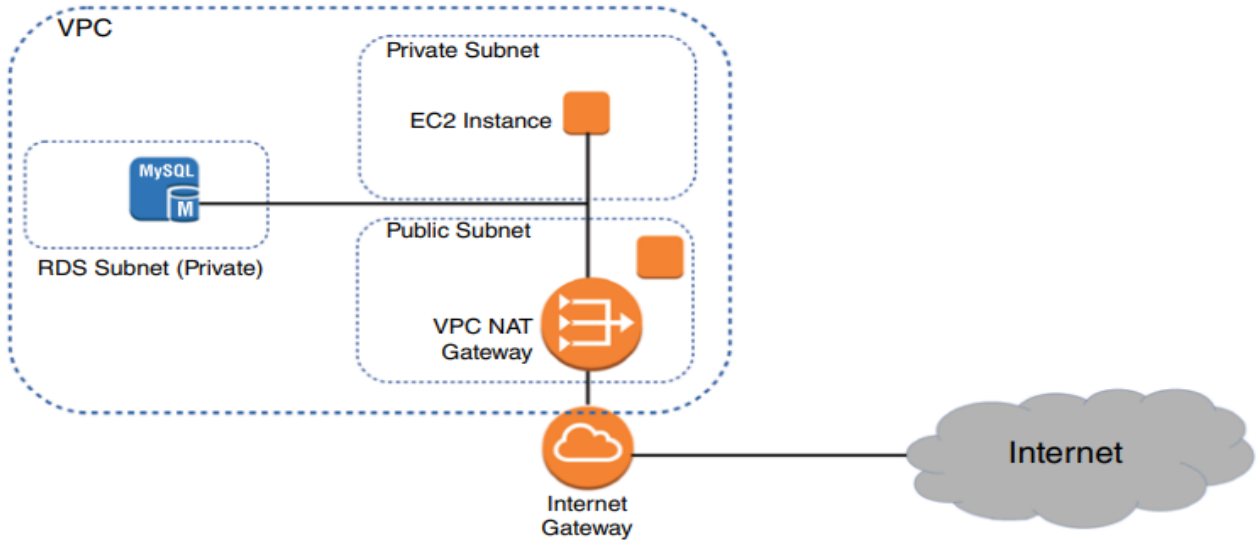
- You can also control access to AWS resources—including EC2 instances—through the use of IAM roles. You define an IAM role by giving it permissions to perform actions on specified services or resources within your AWS account. When a particular role is assigned to a user or resource, they'll gain access to whichever resources were included in the role policies.

- Using roles, you can give a limited number of entities (other resources or users) exclusive access to resources like your EC2 instances. But you can also assign an IAM role to an EC2 instance so that processes running within it can access the external tools—like an RDS database instance—it needs to do its work.

# NAT Devices

- Sometimes you'll need to configure an EC2 instance without a public IP address to limit its exposure to the network. Naturally, that means it won't have any Internet connectivity. But that can present a problem because you'll probably still need to give it Internet access so that it can receive security patches and software updates.
- One solution is to use network address translation (NAT) to give your private instance access to the Internet without allowing access to it from the Internet.

**FIGURE 2.2** A NAT gateway providing network access to resources in private subnets



# Key Pairs

- As any professional administrator will know, remote login sessions on your running instances should never be initiated over unencrypted plain-text connections. To ensure properly secured sessions, you'll need to generate a key pair, save the public key to your EC2 server, and save its private half to your local machine. If you're working with a Windows AMI, you'll use the private key file to retrieve the password you'll need to authenticate into your instance. For a Linux AMI, the private key will allow you to open an SSH session.

# EC2 Auto Scaling

- The EC2 Auto Scaling service offers a way to both avoid application failure and recover from it when it happens.
- Auto Scaling works by provisioning and starting on your behalf a specified number of EC2 instances. It can dynamically add more instances to keep up with increased demand.
- And when an instance fails or gets terminated, Auto Scaling will automatically replace it.
- **Launch Configurations** When you create an instance manually, you have to specify many configuration parameters, including an AMI, instance type, SSH key pair, security group, instance profile, block device mapping, whether it's EBS optimized, placement tenancy, and user data, such as custom scripts to install and configure your application.
- A launch configuration is essentially a named document that contains the same information you'd provide when manually provisioning an instance.

# Launch Templates

- Launch templates are similar to launch configurations in that you can specify the same settings. But the uses for launch templates are more versatile.
- You can use a launch template with Auto Scaling, of course, but you can also use it for spinning up one-off EC2 instances or even creating a spot fleet.
- Launch templates are also versioned, allowing you to change them after creation. Any time you need to make changes to a launch template, you create a new version of it.
- AWS keeps all versions, and you can then flip back and forth between versions as needed. This makes it easier to track your launch template changes over time.

# Auto Scaling Groups

- An Auto Scaling group is a group of EC2 instances that Auto Scaling manages.
- When creating an Auto Scaling group, you must first specify either the launch configuration or launch template you created.
- When you create an Auto Scaling group, you must specify how many running instances you want Auto Scaling to provision and maintain using the launch configuration or template you created.
- You must specify the minimum and maximum size of the Auto Scaling group.

# Auto Scaling Groups

You may also optionally set the desired number of instances you want Auto Scaling to provision and maintain.

- **Minimum Auto Scaling** will ensure the number of healthy instances never goes below the minimum. If you set this to 0, Auto Scaling will not spawn any instances and will terminate any running instances in the group.
- **Maximum Auto Scaling** will make sure the number of healthy instances never exceeds this amount. This might seem strange but remember that you might have budget limitations and need to be protected from unexpected (and unaffordable) usage demands.
- **Desired Capacity** The desired capacity is an optional setting that must lie within the minimum and maximum values. If you don't specify a desired capacity, Auto Scaling will launch the number of instances as the minimum value. If you specify a desired capacity, Auto Scaling will add or terminate instances to stay at the desired capacity.

# Specifying an Application Load Balancer Target Group

If you want to use an application load balancer (ALB) to distribute traffic to instances in your Auto Scaling group, just plug in the name of the ALB target group when creating the Auto Scaling group. Whenever Auto Scaling creates a new instance, it will automatically add it to the ALB target group.

**Health Checks Against Application Instances** When you create an Auto Scaling group, Auto Scaling will strive to maintain the minimum number of instances, or the desired number if you've specified it. If an instance becomes unhealthy, Auto Scaling will terminate and replace it.

# Auto Scaling Options

Once you create an Auto Scaling group, you can leave it be and it will continue to maintain the minimum or desired number of instances indefinitely. However, maintaining the current number of instances is just one option. Auto Scaling provides several other options to scale out the number of instances to meet demand.

**Manual Scaling** If you change the minimum, desired, or maximum values at any time after creating the group, Auto Scaling will immediately adjust. For example, if you have the desired capacity value set to 2 and change it to 4, Auto Scaling will launch two more instances. If you have four instances and set the desired capacity value to 2, Auto Scaling will terminate two instances. Think of the desired capacity as a thermostat.

# Dynamic Scaling Policies

Most AWS-managed resources are elastic—that is, they automatically scale to accommodate increased load.

Some examples include S3, load balancers, Internet gateways, and NAT gateways. Regardless of how much traffic you throw at them, AWS is responsible for ensuring that they remain available while continuing to perform well.

But when it comes to your EC2 instances, you're responsible for ensuring that they're powerful and plentiful enough to meet demand.

**Auto Scaling generates the following aggregate metrics for all instances within the group:**

- Aggregate CPU utilization
- Average request count per target
- Average network bytes in
- Average network bytes out

# Simple Scaling Policies

With a simple scaling policy, whenever the metric rises above the threshold, Auto Scaling simply increases the desired capacity. How much it increases the desired capacity, however, depends on which of the following adjustment types you choose:

**ChangeInCapacity** Increases the capacity by a specified amount. For instance, you could start with a desired capacity value of 4 and then have Auto Scaling increase the value by 2 when the load increases.

**ExactCapacity** Sets the capacity to a specific value, regardless of the current value. For example, suppose the desired capacity value is 4. You create a policy to change the value to 6 when the load increases.

**PercentChangeInCapacity** Increases the capacity by a percentage of the current amount. If the current desired capacity value is 4 and you specify the percent change in capacity as 50 percent, then Auto Scaling will bump the desired capacity value to 6.

# Step Scaling Policies

- ✓ If the demand on your application is rapidly increasing, a simple scaling policy may not add enough instances quickly enough.
- ✓ Using a step scaling policy, you can instead add instances based on how much the aggregate metric exceeds the threshold.
- ✓ To illustrate, suppose your group starts out with four instances.
- ✓ You want to add more instances to the group as the average CPU utilization of the group increases.
- ✓ When the utilization hits 50 percent, you want to add two more instances. When it goes above 60 percent, you want to add four more instances.

# Step Scaling Policies

You must then specify at least one step adjustment. Each step adjustment consists of the following:

- A lower bound
- An upper bound
- The adjustment type
- The amount by which to increase the desired capacity

**Target Tracking Policies** If step scaling policies are too involved for your taste, you can instead use target tracking policies. All you do is select a metric and target value, and Auto Scaling will create a CloudWatch Alarm and a scaling policy to adjust the number of instances to keep the metric near that target.

# Scheduled Actions

Scheduled actions are useful if you have a predictable load pattern and want to adjust your capacity proactively, ensuring you have enough instances before demand hits.

When you create a scheduled action, you must specify the following:

■ A minimum, maximum, or desired capacity value

■ A start date and time You may optionally set the policy to recur at regular intervals, which is useful if you have a repeating load pattern. You can also set an end time, after which the scheduled policy gets deleted.

**FIGURE 2.3** Scheduled action setting the desired capacity to 2 every Saturday

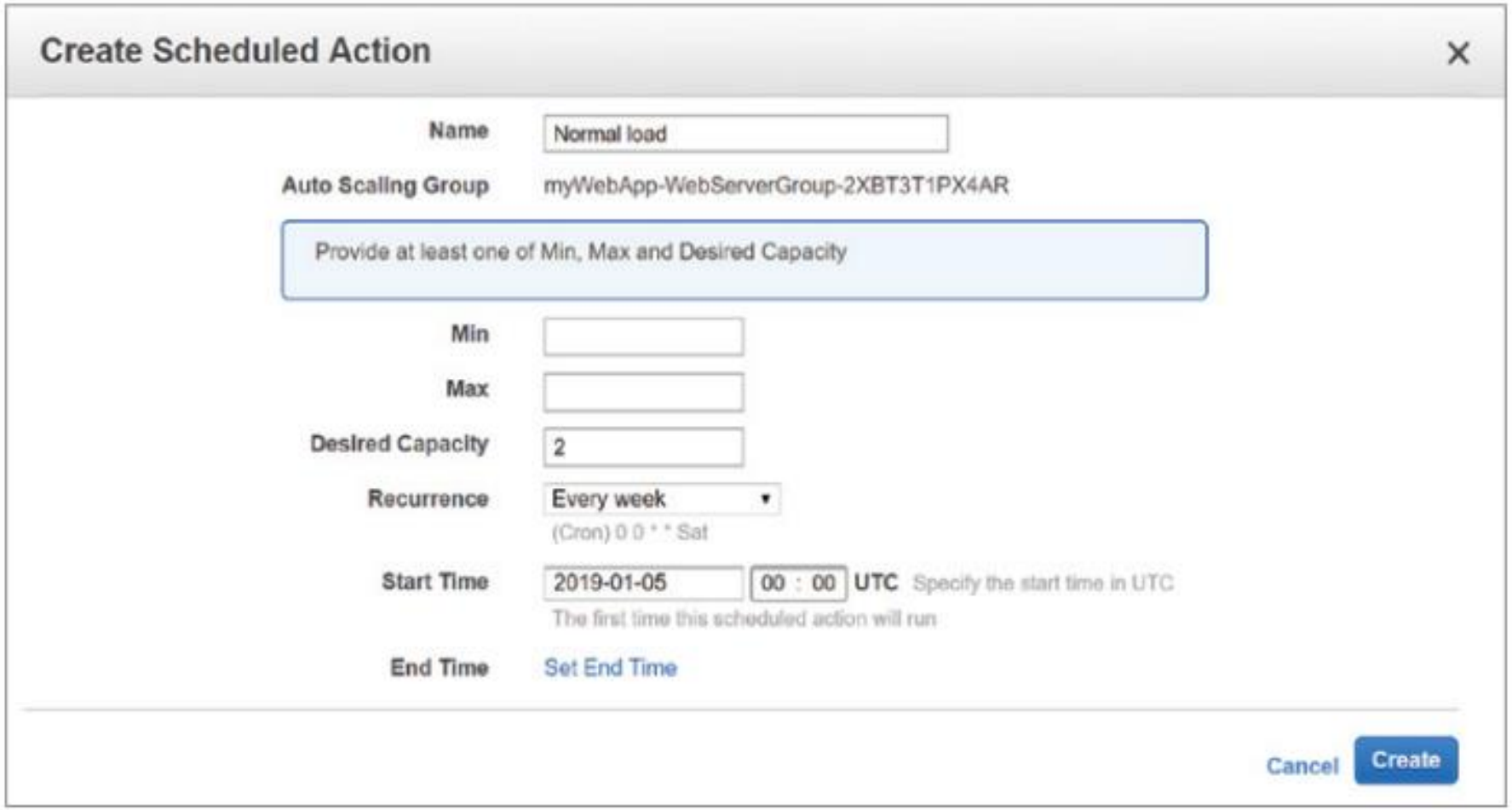
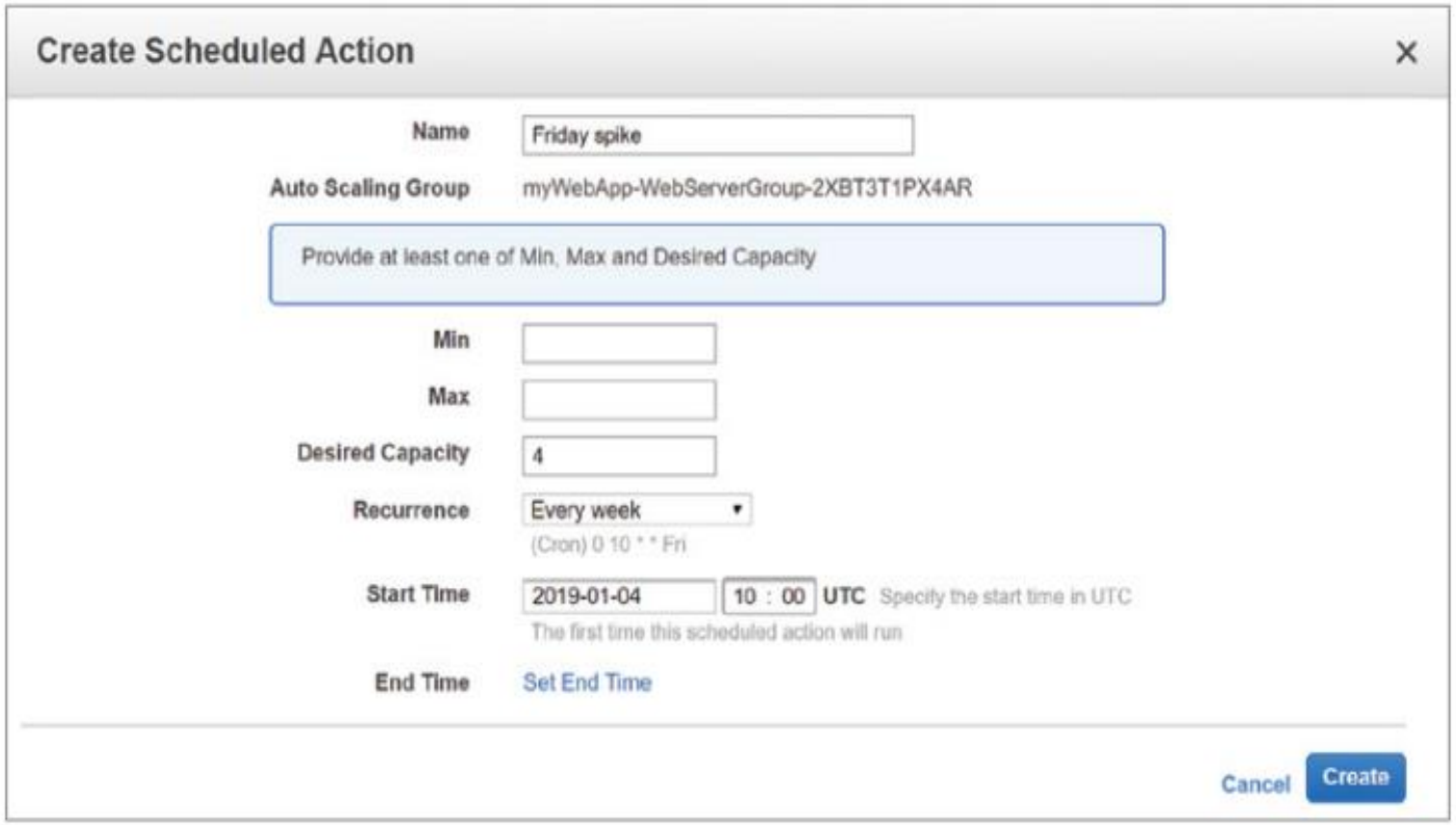


**FIGURE 2.4** Scheduled action setting the desired capacity to 4 every Friday



# AWS Systems Manager

AWS Systems Manager, formerly known as EC2 Systems Manager and Simple Systems Manager (SSM), lets you automatically or manually perform actions against your AWS resources and on-premises servers.

From an operational perspective, Systems Manager can handle many of the maintenance tasks that often require manual intervention or writing scripts. For on-premises and EC2 instances, these tasks include upgrading installed packages, taking an inventory of installed software, and installing a new application.

Systems Manager provides the following two capabilities:

- Actions
- Insights

# AWS Systems Manager

Actions let you automatically or manually perform actions against your AWS resources, either individually or in bulk.

These actions must be defined in documents, which are divided into three types:

- Automation—actions you can run against your AWS resources
- Command—actions you run against your Linux or Windows instances
- Policy—defined processes for collecting inventory data from managed instances

**Automation** Automation enables you to perform actions against your AWS resources in bulk. For example, you can restart multiple EC2 instances, update CloudFormation stacks, and patch AMIs.

While automation lets you automate tasks against your AWS resources, Run commands let you execute tasks on your managed instances that would otherwise require logging in or using a third-party tool to execute a custom script. Systems Manager accomplishes this via an agent installed on your EC2 and on-premises managed instances.

The Systems Manager agent is installed by default on more recent Windows Server, Amazon Linux, and Ubuntu Server AMIs. You can manually install the agent on other AMIs and on-premises servers.

**Session Manager** Session Manager lets you achieve interactive Bash and PowerShell access to your Linux and Windows instances, respectively, without having to open up inbound ports on a security group or network ACL or even having your instances in a public subnet. You don't need to set up a protective bastion host or worry about SSH keys. All Linux versions and Windows Server 2008

Patch Manager Patch Manager helps you automate the patching of your Linux and Windows instances.

It will work for supporting versions of the following operating systems:

- Windows Server
- Ubuntu Server
- Red Hat Enterprise Linux (RHEL)
- SUSE Linux Enterprise Server (SLES)
- CentOS
- Amazon Linux
- Amazon Linux 2

**State Manager** While Patch Manager can help ensure your instances are all at the same patch level, State Manager is a configuration management tool that ensures your instances have the software you want them to have and are configured in the way you define. More generally, State Manager can automatically run command and policy documents against your instances, either one time only or on a schedule. For example, you may want to install antivirus software on your instances and then take a software inventory.

I**nsights** Insights aggregate health, compliance, and operational details about your AWS resources into a single area of AWS Systems Manager. Some insights are categorized according to AWS resource groups, which are collections of resources in an AWS region. You define a resource group based on one or more tag keys and optionally tag values.

Built-in Insights Built-in insights are monitoring views that Systems Manager makes available to you by default.

Built-in insights include the following:

**AWS Config Compliance** This insight shows the total number of resources in a resource group that are compliant or noncompliant with AWS Config rules, as well as compliance by resource. It also shows a brief history of configuration changes tracked by AWS Config.

**CloudTrail Events** This insight displays each resource in the group, the resource type, and the last event that CloudTrail recorded against the resource.

**Personal Health Dashboard** The Personal Health Dashboard contains alerts when AWS experiences an issue that may impact your resources. For example, some service APIs occasionally experience increased latency. It also shows you the number of events that AWS resolved within the last 24 hours.

**Trusted Advisor Recommendations** The AWS Trusted Advisor tool can check your AWS environment for optimizations and recommendations related to cost optimization, performance, security, and fault tolerance. It will also show you when you've exceeded 80 percent of your limit for a service.

**Business and Enterprise support customers get access to all Trusted Advisor checks. All AWS customers get the following security checks for free:**

- Public access to an S3 bucket, particularly upload and delete access
- Security groups with unrestricted access to ports that normally should be restricted, such as TCP port 1433 (MySQL) and 3389 (Remote Desktop Protocol)
- Whether you've created an IAM user
- Whether multifactor authentication is enabled for the root user
- Public access to an EBS or RDS snapshot

**Inventory Manager** The Inventory Manager collects data from your instances, including operating system and application versions.

Inventory Manager can collect data for the following:

- Operating system name and version
- Applications and filenames, versions, and sizes
- Network configuration, including IP and media access control (MAC) addresses
- Windows updates, roles, services, and registry values
- CPU model, cores, and speed

**AWS CLI Example**

The following example code shows how you can use an AWS CLI command to deploy an EC2 instance that includes many of the features you learned about in this chapter. Naturally, the image-id, security-group-ids, and subnet-id values are not real. Those you would replace with actual IDs that fit your account and region.

```
aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 \
--instance-type t2.micro --key-name MyKeyPair \
--security-group-ids sg-xxxxxxxx --subnet-id subnet-xxxxxxxx \
--user-data file://my_script.sh \
--tag-specifications \
'ResourceType=instance,Tags=[{Key=webserver,Value=production}]' \
'ResourceType=volume,Tags=[{Key=cost-center,Value=cc123}]'
```

**How did AWS get started?**

- Origins began in 2002 when Amazon started the Amazon.com web service.
  - Offered tools for developers to work on the Amazon product catalog
- In 2003, Amazon realized that its infrastructure services could give them an advantage over the competition.
  - Provided hardware power, storage, and databases along with the software tools to control them
- In 2004, Amazon publicly announced that it was working on a cloud service.
- In 2006, Amazon launched AWS with just a few of the services that are still around today.
  - Amazon Simple Storage Service (Amazon S3)
  - Amazon EC2
  - Amazon Simple Queue Service (Amazon SQS)
- By 2009, AWS added more services
  - Amazon Elastic Block Store (Amazon EBS)
  - Amazon CloudFront – a content delivery network (CDN)

| Type of Cloud Service | What It Does | Examples |
|---|---|---|
| Infrastructure as a service (IaaS) | Compute power, networking, and storage provided over the internet | Amazon Elastic Compute Cloud (Amazon EC2), Rackspace, Google Compute Engine |
| Platform as a service (PaaS) | Tools provided over the internet for making programs and applications | AWS Elastic Beanstalk, Microsoft Azure, Google App Engine |
| Software as a service (SaaS) | Applications and programs that are accessed and provided over the internet | Dropbox, Slack, Spotify, YouTube, Microsoft Office 365, Gmail |

**•A pediatrician with a private practice has so many patient files, that she is running out of room in her filing cabinets. For this reason, she wants to move her data into the cloud. She wants to be sure the data is secure, but also wants her patients to be able to access their medical records and communicate with her online in a secure way. Describe one way that you can use each type of cloud service and how it would benefit her business.**

Create a rubric where you rank the factors for a business to consider when choosing a Region to locate their cloud services. Then explain why those factors would be most (or least) important when choosing a Region.
Factors to consider:
•AWS cost
•Availability of services
•Speed or latency
•Resiliency of AWS components
•Data rights
•Audience