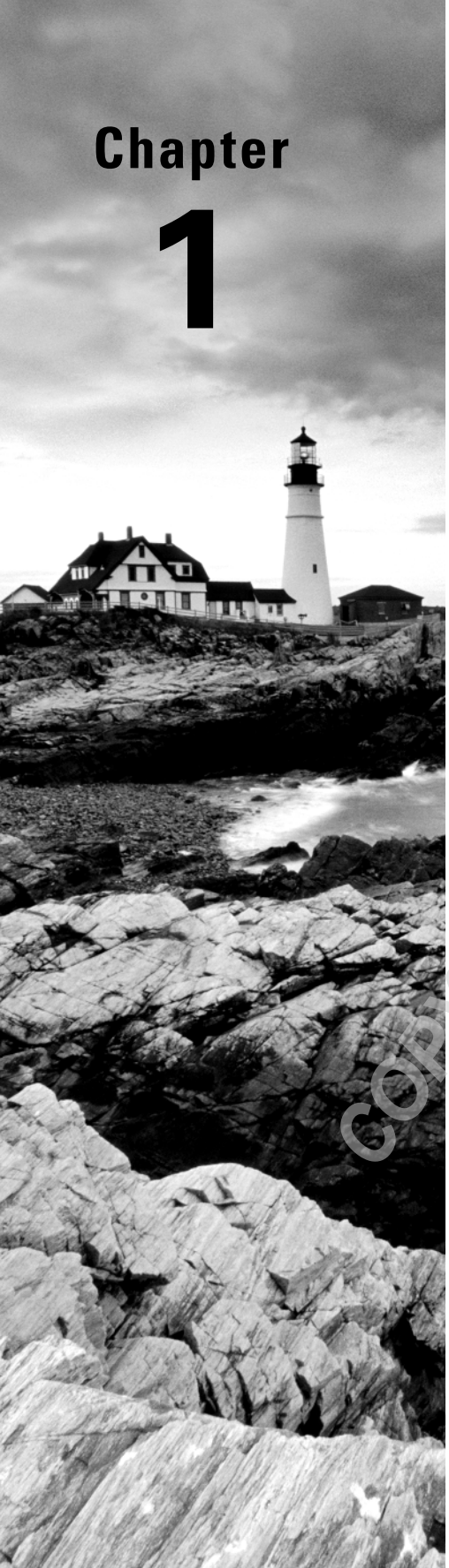


Chapter 1

The Cloud

THE AWS CERTIFIED CLOUD PRACTITIONER EXAM OBJECTIVES COVERED IN THIS CHAPTER MAY INCLUDE, BUT ARE NOT LIMITED TO, THE FOLLOWING:

Domain 1: Cloud Concepts

- ✓ **1.1 Define the AWS Cloud and its value proposition**
- ✓ **1.2 Identify aspects of AWS Cloud economics**
- ✓ **1.3 List the different cloud architecture design principles**

Introduction

If you want to make smart choices about how your organization is going to use Amazon Web Services' cloud platform, you'll first need to properly understand it. To get there, you'll need to figure out just what the cloud is, what technologies it's built on, what kinds of cost savings and operational advantages it can bring you, and how cloud-based applications work differently than their traditional cousins.

This chapter will introduce you to the basics. The rest of the book will fully flesh out the details.

What Is Cloud Computing?

Using a public cloud is about using other people's servers to run your digital workloads.

In a sense, there's no significant difference between running a software application on servers hosted in your own office versus locating it within Amazon's infrastructure. In both cases, you need to make sure you've got sufficient compute, memory, network, and storage resources. In both cases, fast deployments and avoiding over-provisioning are key goals.

But, particularly when it comes to the largest cloud providers, there are important differences. You see, the sheer size of a platform like AWS (and right now there's no platform on Earth that's bigger) means it can offer you service, cost, and reliability performance that you could probably never hope to re-create on your own.

Let's see how some of that works.

Highly Available and Scalable Resources

There's an awful lot a successful company like AWS can get done with a few hundred thousand networked servers and hundreds of the best trained engineers in the business:

- Design multiple layers of redundancy so that whenever one component fails, its workload is automatically and instantly moved to a healthy replacement.
- Connect resources in geographically remote locations so that the failure of one complete region could trigger a predefined relocation. This relocation can be supported by a similarly automated rerouting of network requests.

- Provide customers with access to as much compute power as they could possibly need, and deliver that power on-demand.
- Because of the scale and efficiency of the platform, AWS can do all that at a price that's often far below what it would cost to run comparable workloads locally.

Professionally Secured Infrastructure

IT security is a constantly moving target. As difficult as it's been to manage last year's threats, you know there's a whole new batch coming right behind them. As a business, you're already responsible for protecting the workstations and networking hardware running in your office along with securing your organization's data and code your developers put into your apps. The integrity of your underlying server infrastructure is just one more potential area of vulnerability for you to worry about.

No matter how good your IT security team is, they're probably not better informed, equipped, and trained than their counterparts at a major cloud provider. Because AWS is so good at what it does—and because it takes responsibility for the security of its platform's underlying networking and compute infrastructure—this is one area where outsourcing will usually make sense.

This won't relieve you of all worries. As you'll see in Chapter 4, "Understanding the AWS Environment," the terms of the AWS Shared Responsibility Model mean that, in many cases, the security and integrity of the resources you run *on* the cloud are still your problem. But *the cloud itself* is managed by AWS.

Metered Payment Model

One of the defining characteristics of any public cloud computing platform is the way it automatically allocates resources to meet client requests. Practically, this means that you can, for instance, log in to the AWS browser console, and define and launch a virtual server (called an *instance* in the AWS world), and moments later your new instance will be ready for you. There's no need to wait for manual intervention by AWS employees.

The flexibility of the self-serve system permits usage patterns that would have been impossible using traditional compute paradigms. Let's say you need to quickly test a possible application configuration you're working on. In the old days, even if the test lasted only an hour, you would still need to find free capacity on a physical server in the server room. Once the test ended, you'd still be paying the maintenance and ownership costs of that server capacity even if it was idle.

In the cloud, by contrast, you fire up an instance, run it for the length of time your test requires, and then shut it down. You'll be billed for only that testing time, which, in some cases, could cost you a fraction of a penny.

Since there's no human processing involved in cloud compute billing, it's as easy for a provider to charge a few pennies as it is thousands of dollars. This metered payment makes it possible to consider entirely new ways of testing and delivering your applications, and it often means your cost-cycle expenses will be considerably lower than they would if you were using physical servers running on-premises.

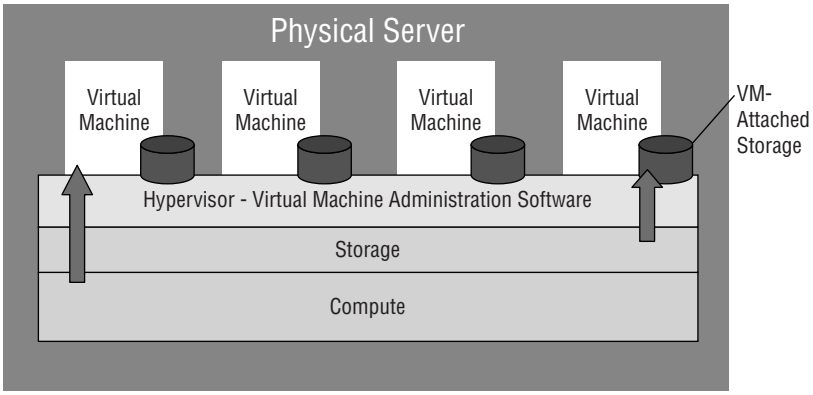
Comparing the costs of cloud deployments against on-premises deployments requires that you fully account for both capital expenses (*capex*) and operating expenses (*opex*). On-premises infrastructure tends to be very capex-heavy since you need to purchase loads of expensive hardware up front. Cloud operations, on the other hand, involve virtually no capex costs at all. Instead, your costs are ongoing, consisting mostly of per-hour resource "rental" fees.

You'll learn more about AWS billing in Chapter 2, "Understanding Your AWS Account."

Server Virtualization: The Basics

The secret sauce that lets cloud providers give their customers on-demand compute resources in such a wide range of configurations is virtualization. When you request a *virtual machine* (VM) with a particular processor speed, memory capacity, and storage size, AWS doesn't send some poor engineer running through the halls of its data center looking for an available machine with exactly that profile. Rather, as you can see illustrated in Figure 1.1, AWS carves the necessary resources from larger existing devices.

FIGURE 1.1 VMs accessing storage and compute resources from their host server



A 5 TB storage drive could, for instance, be divided into dozens of smaller virtual volumes, each associated with a different virtual server (or instance). And the resources of a single physical server could be invisibly shared between multiple instances. The operating systems installed on each of those instances could run, blissfully unaware that they're actually only masters over a small subset of a much larger server environment.

The virtualization model offers two compelling benefits:

- **Speed:** Defining, purchasing, provisioning, testing, and launching a new physical server can take months. Even a simple reboot can keep you waiting for a couple of minutes. The time lag between requesting a new cloud-based VM and logging in and getting to work can be seconds, but never more than a few minutes. Restarting a VM can sometimes happen faster than you can type your login details.

- **Efficiency:** It's rare to find a nonvirtualized physical server that utilizes anywhere near 100 percent of its capacity. More likely, either it'll spend its time running mostly empty or it'll be badly overused while you wait for more capacity to come online. Multiple virtual machines, on the other hand, can be tightly packed onto a physical server running a hypervisor (a common technology for hosting VMs). When space opens up on one server, you can quickly fill it with another virtual workload. When a server reaches capacity, overflow workloads can be moved to another machine. And the more workloads you're managing, the more flexible everything gets.

Amazon's formidable scale and logistical abilities mean that it's often able to leverage the benefits of virtualization to provide both superior performance and pricing.

Cloud Platform Models

Cloud services come in more than one flavor. Choosing the one that's right for your project will depend on your specific needs and how much fine control you'll need over the underlying gears and levers.

Infrastructure as a Service

Infrastructure as a Service (IaaS) products generally simulate the look and feel you'd get from managing physical resources. IaaS products give you direct access to a provider's compute, storage, and networking assets. Because it's you that's in there playing around at the hardware level, you—rather than the IaaS provider—are responsible for the consequences of any bad configurations. The trade-off is that you get to closely configure every layer of your operating stack.

You'll learn much more about these examples later in the book, but AWS IaaS products include Elastic Cloud Compute (EC2) for virtual machine instances, Elastic Block Store (EBS) for storage volumes, and Elastic Load Balancing.

Platform as a Service

Unlike IaaS, Platform as a Service (PaaS) products simplify the process of building an application by *hiding the complexity* of the infrastructure that runs it. You're given an interface through which you define the behavior and environment you want for your application. This will often include the code that will run your application.

AWS PaaS products include Elastic Beanstalk and Elastic Container Service (ECS).

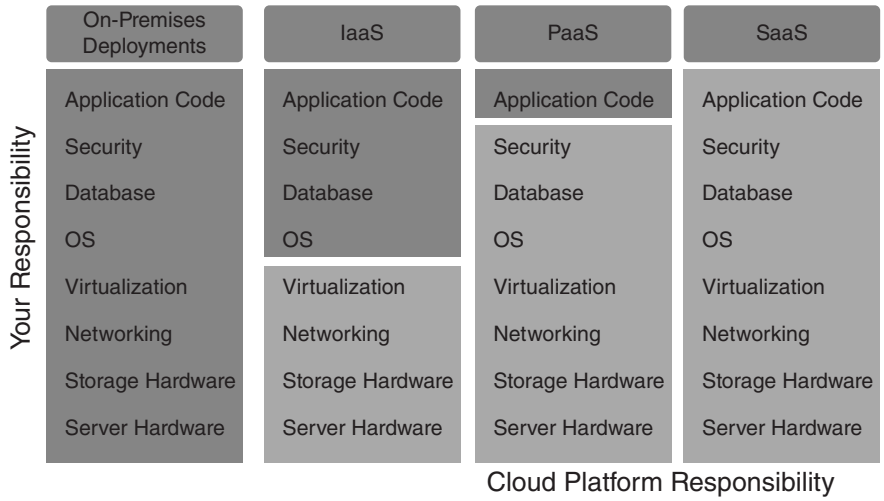
Software as a Service

Software as a Service (SaaS) products offer services meant to be accessed by end users. An easily recognizable illustration is Google's Gmail service, which allows users to manage their email by logging in to a browser interface or through an email client (like Microsoft Outlook) that's running locally.

While some may disagree with the designation, AWS SaaS products arguably include Simple Email Service and Amazon WorkSpaces.

Figure 1.2 compares the scope of responsibility you have on IaaS, PaaS, and SaaS platforms with the way it works for on-premises deployments.

FIGURE 1.2 The breakdown of responsibility across multiple infrastructure types



Serverless Workloads

Besides doing an excellent job emulating traditional server behavior, cloud providers can also enable entirely new ways to administrate applications and data. Perhaps the most obvious example is serverless computing.

Now don't be fooled by the name. You can't run a compute function without a computer environment (a "server") somewhere that'll host it. What "serverless" does allow is for individual developers to run their code for seconds or minutes at a time on some else's cloud servers.

The serverless model—as provided by services like AWS Lambda—makes it possible to design code that *reacts* to external events. When, for instance, a video file is uploaded to a repository (like an AWS S3 bucket or even an on-premises FTP site), it can trigger a Lambda function that will convert the file to a new video format. There's no need to maintain and pay for an actual instance running 24/7, just for the moments your code is actually running. And there's no administration overhead to worry about.

Scalability and Elasticity

The world's largest public cloud providers can accomplish a great deal through combining the wonders of server virtualization with the power that comes from owning vast data centers filled with racks upon racks of hardware resources. Elasticity and scalability are the two key principles through which a lot of this happens, and understanding exactly what they mean can help you optimize your design choices so you'll get the most bang for your cloud buck.

Note that there really are no precise and authoritative definitions for scalability and elasticity in the context of cloud computing—and any definitions you do see are bound to involve at least some overlap. Nevertheless, building some kind of picture in your mind of how these two principles work can be valuable.

Scalability

A scalable service will automatically grow in capacity to seamlessly meet any changes in demand. A well-designed cloud-based operation will constantly monitor the health of its application stack and respond whenever preset performance metrics might soon go unmet. The response might include automatically launching new server instances to add extra compute power to your existing cluster. But it will probably also involve prepopulating those instances with the application data and configuration settings they'll need to actually serve your application to your clients.

A large cloud provider like AWS will, for all practical purposes, have endless available capacity so the only practical limit to the maximum size of your application is your organization's budget (and default service limits imposed by AWS that you'll learn about later in Chapter 2).

Just how big is AWS? Well, if it can handle the capacity stresses required to keep millions of Netflix customers happy—and if you've ever watched a movie on the AWS-hosted Netflix, then you know it can—then AWS will surely be able to keep up with whatever trouble your applications send its way.

Elasticity

You can stretch an elastic band far beyond its resting state. But part of what makes it truly elastic is the fact that, when you let go of it, it immediately returns to its original size. The reason the word *elastic* is used in the names of so many AWS services (Elastic Compute Cloud, Elastic Load Balancing, Elastic Beanstalk, and so on) is because those services are built to be easily and automatically resized.

Generally, you set the maximum and minimum performance levels you want for your application, and the AWS service(s) you're using will automatically add or remove resources to meet changing usage demands. By way of illustration, a scalable ecommerce website

could be configured to function using just a single server during low-demand periods, but any number of additional servers could be automatically brought online as demand spikes. When demand drops back down, unused servers will be shut down automatically.

Assuming you don't already have one, now is the time to create your own AWS account. Working through the rest of this book will be pretty much impossible without an active account. You will have to provide a credit card, but you won't be charged anything unless you actually launch an actual resource. Work through Exercise 1.1 to make this happen.

EXERCISE 1.1

Create an AWS Account

1. Go to `https://aws.amazon.com`, and choose the Create An AWS Account button. If, instead, you see a Sign In button, you might have previously logged into an existing account using this browser. If you'd still like to create a new account, choose Sign In and then create a new, different account.
2. Enter the email address you want to associate with the account, a (strong) password, and a new AWS account name you'd like to use. Choose Continue.
3. Select an account type (either Professional or Personal—the only difference is the Company Name field) and the other requested information. Then agree to the AWS terms and choose Create Account And Continue. You will receive a confirmation email, after which you can continue with the activation process.
4. Enter a payment method and, if the payment address is different from the address you used in the previous step, enter a new address and choose Secure Submit.
5. On the Phone Verification page, enter the phone number AWS can use for an automated authentication call. After you choose Call Me Now, enter the PIN that will be displayed in the browser page on your phone's keypad.
6. One last step: on the Support Plan page, select an AWS support plan. If you're not sure which plan you want, just go with the Basic plan for now. It's free, and you can always upgrade later. You'll learn more about AWS support in Chapter 3, "Getting Support on AWS."

Once you're fully activated, you'll receive another email, this one confirming that your account is ready for you.

Summary

The size and quality of a major cloud provider like AWS means that its customers can often benefit from higher-quality security, availability, and reliability than they could provide locally.

While AWS customers are still responsible for the applications they run in the cloud, they don't need to worry about the underlying physical infrastructure that's managed by AWS.

Much of the attraction of cloud computing is the ability to pay for only the services you use, and only as you use them. This allows the provisioning of sophisticated applications with virtually no capital expenses (capex). You will, of course, need to assess and manage the operating expenses (opex).

Server virtualization makes it possible to more densely pack software operations on physical hardware, potentially driving down the costs and improving the time-to-deployment of compute workloads. An even more "virtualized" kind of virtualization is serverless computing, where customers are aware their code and the network events that trigger it.

Cloud-optimized workloads are designed to take advantage of the scalability and elasticity of cloud platforms.

Exam Essentials

Understand how a large and geographically dispersed infrastructure improves service quality. The sheer scale and geographic redundancy of the physical compute and networking resources owned by AWS mean that the company is able to guarantee a level of reliability and availability that would be hard to reproduce in any other environment.

Understand how metered, pay-per-use pricing makes for flexible compute options. Access to cloud infrastructure—sometimes for pennies per hour—makes it possible to experiment, sandbox, and regularly reassess and update application stacks.

Understand that cloud services come on a wide range of forms. IaaS gives you near-full control over virtualized hardware resources, closely emulating the way you would administrate actual physical servers. PaaS products abstract the underlying infrastructure, providing a simplified interface for you to add your application code. SaaS products provide services over a public network directly to end users.

Understand how serverless computing can be both cheap and efficient. Serverless services like AWS Lambda allow you access to AWS compute power for up to 15 minutes for a single function. This lets you operate code in response to real-time event triggers.

Understand how scalability allows applications to grow to meet need. A cloud-optimized application allows for automated provisioning of server instances that are designed from scratch to perform a needed compute function within an appropriate network environment.

Understand how elasticity matches compute power to both rising and falling demand. The scaling services of a cloud provider—like AWS Auto Scaling—should be configured to force compliance with your budget and application needs. You set the upper and lower limits, and the scaler handles the startups and shutdowns to optimize operations in between those limits.

Review Questions

1. Which of the following does not contribute significantly to the operational value of a large cloud provider like AWS?
   - **A.** Multiregional presence
   - **B.** Highly experienced teams of security engineers
   - **C.** Deep experience in the retail sphere
   - **D.** Metered, pay-per-use pricing
2. Which of the following are signs of a highly available application? (Select TWO.)
   - **A.** A failure in one geographic region will trigger an automatic failover to resources in a different region.
   - **B.** Applications are protected behind multiple layers of security.
   - **C.** Virtualized hypervisor-driven systems are deployed as mandated by company policy.
   - **D.** Spikes in user demand are met through automatically increasing resources.
3. How does the metered payment model make many benefits of cloud computing possible? (Select TWO.)
   - **A.** Greater application security is now possible.
   - **B.** Experiments with multiple configuration options are now cost-effective.
   - **C.** Applications are now highly scalable.
   - **D.** Full-stack applications are possible without the need to invest in capital expenses.
4. Which of the following are direct benefits of server virtualization? (Select TWO.)
   - **A.** Fast resource provisioning and launching
   - **B.** Efficient (high-density) use of resources
   - **C.** Greater application security
   - **D.** Elastic application designs
5. What is a hypervisor?
   - **A.** Hardware device used to provide an interface between storage and compute modules
   - **B.** Hardware device used to provide an interface between networking and compute modules
   - **C.** Software used to log and monitor virtualized operations
   - **D.** Software used to administrate virtualized resources run on physical infrastructure

6. Which of the following best describes server virtualization?
   - **A.** "Sharding" data from multiple sources into a single virtual data store
   - **B.** Logically partitioning physical compute and storage devices into multiple smaller virtual devices
   - **C.** Aggregating physical resources spread over multiple physical devices into a single virtual device
   - **D.** Abstracting the complexity of physical infrastructure behind a simple web interface

7. Which of the following best describes Infrastructure as a Service products?
   - **A.** Services that hide infrastructure complexity behind a simple interface
   - **B.** Services that provide a service to end users through a public network
   - **C.** Services that give you direct control over underlying compute and storage resources
   - **D.** Platforms that allow developers to run their code over short periods on cloud servers

8. Which of the following best describes Platform as a Service products?
   - **A.** Services that hide infrastructure complexity behind a simple interface
   - **B.** Platforms that allow developers to run their code over short periods on cloud servers
   - **C.** Services that give you direct control over underlying compute and storage resources
   - **D.** Services that provide a service to end users through a public network

9. Which of the following best describes Software as a Service products?
   - **A.** Services that give you direct control over underlying compute and storage resources
   - **B.** Services that provide a service to end users through a public network
   - **C.** Services that hide infrastructure complexity behind a simple interface
   - **D.** Platforms that allow developers to run their code over short periods on cloud servers

10. Which of the following best describes scalability?
    - **A.** The ability of an application to automatically add preconfigured compute resources to meet increasing demand
    - **B.** The ability of an application to increase or decrease compute resources to match changing demand
    - **C.** The ability to more densely pack virtualized resources onto a single physical server
    - **D.** The ability to bill resource usage using a pay-per-user model

11. Which of the following best describes elasticity?

- **A.** The ability to more densely pack virtualized resources onto a single physical server
- **B.** The ability to bill resource usage using a pay-per-user model
- **C.** The ability of an application to increase or decrease compute resources to match changing demand
- **D.** The ability of an application to automatically add preconfigured compute resources to meet increasing demand

12. Which of the following characteristics most help AWS provide such scalable services? (Select TWO.)

- **A.** The enormous number of servers it operates
- **B.** The value of its capitalized assets
- **C.** Its geographic reach
- **D.** Its highly automated infrastructure administration systems