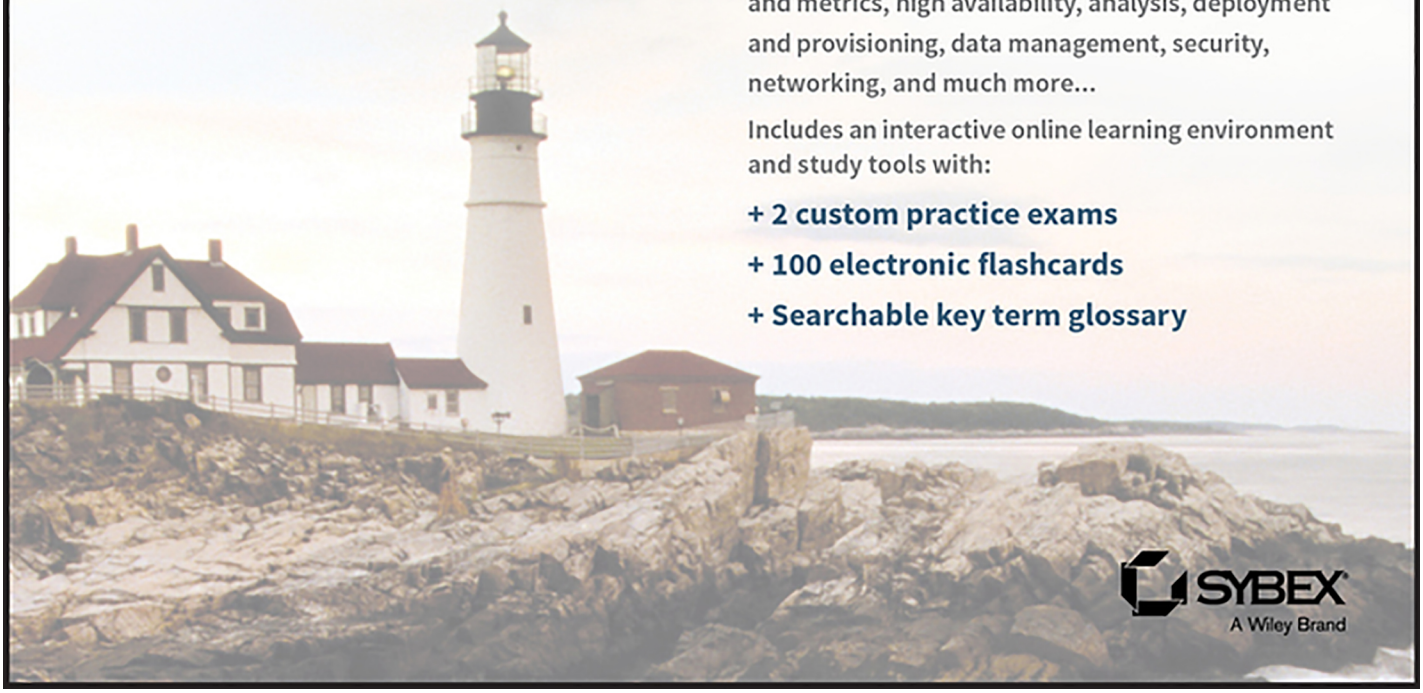
amazon web services

Stephen Cole, Gareth Digby, Chris Fitch,
Steve Friedberg, Shaun Qualheim, Jerry Rhoads,
Michael Roth, Blaine Sundrud

AWS Certified SysOps Administrator

OFFICIAL STUDY GUIDE

ASSOCIATE EXAM



Covers exam objectives, including monitoring and metrics, high availability, analysis, deployment and provisioning, data management, security, networking, and much more...

Includes an interactive online learning environment and study tools with:

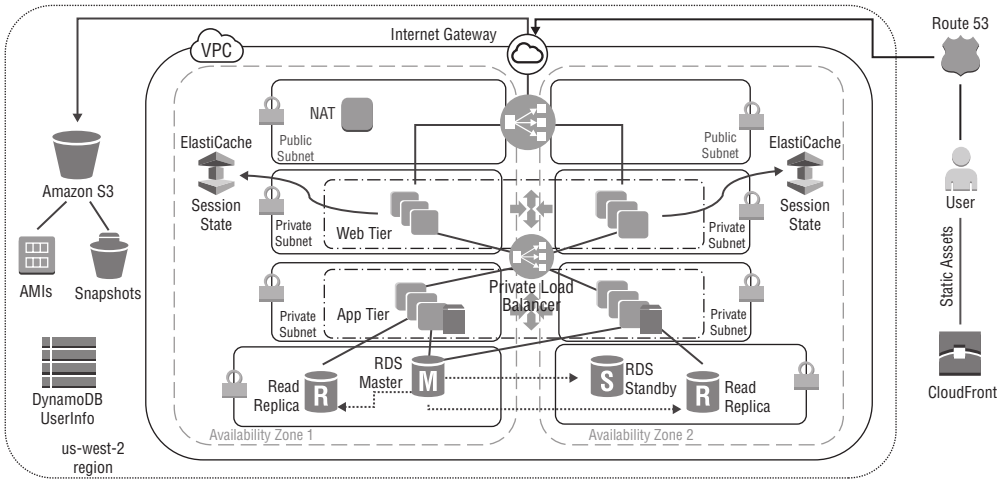
+ 2 custom practice exams

+ 100 electronic flashcards

+ Searchable key term glossary

SYBEX
A Wiley Brand

The Solution



As we examine the pieces of the solution, we start by breaking down the components of the architecture. Then we focus on how systems operators interact with the individual pieces and begin thinking about how those pieces fit into the certification exam.

Environment

Architectures live inside *AWS Regions*; in this scenario, in us-west-2 (Oregon, United States). Regions are made up of multiple *Availability Zones*, which provide the foundation for highly available architectures. Although this is a systems operation exam, it is critical to understand the nature of AWS Regions and Availability Zones.

Each AWS Region is a separate geographic area. Each AWS Region has multiple, isolated locations known as *Availability Zones*. *AWS Regions* and *Availability Zones* are discussed in Chapter 5, "Networking."

Networking

Networking components start inside the AWS Region with Amazon Virtual Private Cloud (Amazon VPC). *Amazon VPC* is a private network in the AWS Region that isolates all traffic from the millions of other applications running in AWS. A deep dive into Amazon VPC (and the rest of its components) is found in Chapter 5.

Amazon VPC is divided into *subnets*; all assets running in your Amazon VPC are assigned to a subnet. Unlike on-premises subnetting decisions that can affect latency between servers, Amazon VPC subnets only affect access. Access between subnets is

controlled through *network Access Control Lists (nACLs)*, and access in and out of Amazon VPC is controlled through attached gateways. In this scenario, the only gateway is the *Internet Gateway (IGW)*, and it allows traffic to and from external (public IP) sources.

By granting route table access to the gateway only to specific subnets, ingress and egress can be tightly controlled. In this scenario, public subnets indicate IGW access. Without IGW access, the subnets become private; that is, they are accessible only to private IP networks.

To learn about the other gateways that could be leveraged to create hybrid or other private architectures, refer to Chapter 5.

Security groups are often part of the networking discussion. They provide stateful firewalls that operate at the Hypervisor levels for all individual*Amazon Elastic Compute Cloud (Amazon EC2)* instances and other Amazon VPC objects. In this scenario, we potentially have seven different security groups:

Public Elastic Load Balancing The only security group that allows full public access

Web Tier Amazon EC2 This accepts traffic only from public Elastic Load Balancing.

Private Elastic Load Balancing This accepts traffic only from Web Tier Amazon EC2.

Application Tier Amazon EC2 This accepts traffic only from private Elastic Load Balancing.

Amazon ElastiCache This accepts traffic only from Application Tier Amazon EC2.

Amazon Relational Database Service (Amazon RDS) This accepts traffic only from Application Tier Amazon EC2.

Network Address Translation (NAT) This is used only for internally initiated outbound traffic.

By specifically stacking security groups in this manner, you can provide layers of network security that surround the database portion of the three-tier design.

Compute

In this scenario, you use traditional compute methods, such as Linux servers running on Amazon EC2. Amazon EC2 comes in many sizes (how many CPUs, how much memory, how much network capacity, and so on), known as *instances*. Based on the Amazon Machine Image (AMI), each Amazon EC2 instance can run a wide range of Linux- or Windows-based operating systems as well as preinstalled software packages. Amazon EC2 instances also support runtime configuration as required.

The requirements for the scenario include scalable solutions. AWS provides Auto Scaling as an engine that can take predefined launch configurations and dynamically add or remove instances from the web or the Application Tier based on metrics.

as that right is explicitly reserved for the AWS account that created the volume. An Amazon EBS snapshot is a block-level view of an entire Amazon EBS volume. Note that data that is not visible through the file system on the volume, such as files that have been deleted, may be present in the Amazon EBS snapshot. If you want to create shared snapshots, you should do so carefully. If a volume has held sensitive data or has had files deleted from it, you should create a new Amazon EBS volume to share. The data to be contained in the shared snapshot should be copied to the new volume and the snapshot created from the new volume.

Amazon EBS volumes are presented to you as raw unformatted *block devices*, which have been wiped prior to being made available for use. Wiping occurs immediately before reuse so that you can be assured that the wipe process completed. If you have procedures requiring that all data be wiped via a specific method, you have the ability to do so on Amazon EBS. You should conduct a specialized wipe procedure prior to deleting the volume for compliance with your established requirements.

Encryption of sensitive data is generally a good security practice, and AWS provides the ability to encrypt Amazon EBS volumes and their snapshots with AES-256. The encryption occurs on the servers that host the Amazon EC2 instances, providing encryption of data as it moves between Amazon EC2 instances and Amazon EBS storage. In order to be able to do this efficiently and with low latency, the Amazon EBS encryption feature is only available on Amazon EC2's more powerful instance types.

Networking

AWS provides a range of networking services that enable you to create a logically isolated network that you define, establish a private network connection to the AWS Cloud, use a highly available and scalable Domain Name System (DNS) service, and deliver content to your end users with low latency at high data transfer speeds with a content delivery web service.

Elastic Load Balancing Security

Elastic Load Balancing is used to manage traffic on a fleet of Amazon EC2 instances, distributing traffic to instances across all Availability Zones within a region. Elastic Load Balancing has all of the advantages of an on-premises load balancer, plus several security benefits:

- Takes over the encryption and decryption work from the Amazon EC2 instances and manages it centrally on the load balancer
- Offers clients a single point of contact and can also serve as the first line of defense against attacks on the customer's network
- When used in an Amazon VPC, supports creation and management of security groups associated with Elastic Load Balancing to provide additional networking and security options
- Supports end-to-end traffic encryption using TLS (previously SSL) on those networks that use HTTPS connections. When TLS is used, the TLS server certificate used to terminate client connections can be managed centrally on the load balancer, instead of on every individual instance.

HTTPS/TLS uses a long-term secret key to generate a short-term session key to be used between the server and the browser to create the encrypted message. Elastic Load Balancing configures your load balancer with a predefined cipher set that is used for TLS negotiation when a connection is established between a client and your load balancer. The predefined cipher set provides compatibility with a broad range of clients and uses strong cryptographic algorithms. However, some customers may have requirements for allowing only specific ciphers and protocols (for example, PCI DSS, Sarbanes-Oxley Act [SOX]) from clients to ensure that standards are met. In these cases, Elastic Load Balancing provides options for selecting different configurations for TLS protocols and ciphers. You can choose to enable or disable the ciphers depending on your specific requirements.

To help ensure the use of newer and stronger cipher suites when establishing a secure connection, you can configure the load balancer to have the final say in the cipher suite selection during the client-server negotiation. When the server order preference option is selected, the load balancer will select a cipher suite based on the server's prioritization of cipher suites rather than the client's. This gives you more control over the level of security that clients use to connect to your load balancer.

For even greater communication privacy, Elastic Load Balancing allows the use of perfect forward secrecy, which uses session keys that are ephemeral and not stored anywhere. This prevents the decoding of captured data, even if the secret long-term key itself is compromised.

Elastic Load Balancing allows you to identify the originating IP address of a client connecting to your servers, whether you're using HTTPS or TCP load balancing. Typically, client connection information, such as IP address and port, is lost when requests are proxied through a load balancer. This is because the load balancer sends requests to the server on behalf of the client, making your load balancer appear as though it is the requesting client. Having the originating client IP address is useful if you need more information about visitors to your applications in order to gather connection statistics, analyze traffic logs, or manage whitelists of IP addresses.

Elastic Load Balancing access logs contain information about each HTTP and TCP request processed by your load balancer. This includes the IP address and port of the requesting client, the back-end IP address of the instance that processed the request, the size of the request and response, and the actual request line from the client (for example, `GET http://www.example.com: 80/HTTP/1.1`). All requests sent to the load balancer are logged, including requests that never make it to back-end instances (more on Elastic Load Blancing can be found in Chapter 5).

Amazon Virtual Private Cloud (Amazon VPC) Security

Normally, each Amazon EC2 instance you launch is randomly assigned a public IP address in the Amazon EC2 address space. *Amazon VPC* enables you to create an isolated portion of the AWS Cloud and launch Amazon EC2 instances that have private (RFC 1918) addresses in the range of your choice (for example, 10.0.0.0/16). You can define subnets in your Amazon VPC, grouping similar kinds of instances based on IP address range, and then set up routing and security to control the flow of traffic in and out of the instances and subnets.

groups to control access on a specific interface. With administrative access, you again can control access with placement of Amazon EC2 instances in an Amazon VPC, the use of security groups, and the use of public/private key pairs both to encrypt traffic and control access.

Controlling Administrative Access

AWS provides a number of tools to manage the security of your instances. These tools include the following:

- IAM
- AWS Trusted Advisor
- AWS Service Catalog
- Amazon Inspector

IAM allows you to create policies that control access to APIs and apply those policies to users, groups, and roles.

AWS Trusted Advisor, which comes as part of AWS Support, looks at things like open ports on security groups and level of access allowed to Amazon EC2 instances, and it makes recommendations to improve the security of your AWS infrastructure.

AWS Service Catalog allows IT administrators to create, manage, and distribute portfolios of approved products to end users who can then access the products they need in a personalized portal. AWS Service Catalog allows organizations to manage commonly deployed IT services centrally, and it helps organizations achieve consistent governance and meet compliance requirements while enabling users to deploy quickly only the approved IT services they need within the constraints the organization sets.

Amazon Inspector is a security vulnerability assessment service that helps improve the security and compliance of your AWS resources. Amazon Inspector automatically assesses resources for vulnerabilities or deviations from best practices and then produces a detailed list of security findings prioritized by level of severity.

AWS also has a number of developer documents, whitepapers, articles, and tutorials to help you in securing your environment. These are available on the Cloud Security Resources page of the AWS website.

Amazon EC2 Container Service (Amazon ECS)

Amazon ECS is a highly scalable container management service. Amazon ECS makes it easy to run, stop, and manage Docker containers on Amazon EC2 instances.

With Amazon ECS, you can launch and place containers across your cluster using API calls. You schedule the placement of containers based on your resource needs, isolation

policies, and availability requirements. You can use Amazon ECS both for cluster management and for scheduling deployments of containers onto hosts.

Amazon ECS eliminates the need for you to operate your own cluster management and configuration management systems. Amazon ECS can be used to manage and scale both batch and Extract, Transform, Load (ETL) workloads. It can also be used to build application architectures based on the microservices model.

Amazon ECS is a regionally-based service that can be used to run application containers in a highly available manner across all Availability Zones within an AWS Region.

Implementation

To implement Amazon ECS, you need to install an Amazon ECS agent on your Amazon EC2 instances. If you use Amazon ECS-optimized AMIs, that agent is already installed. Additionally, the container instance needs to have an IAM role that authenticates to your account and will need external network access to communicate with the Amazon ECS service endpoint.

Clusters are the logical grouping of container instances that you can place tasks on. Clusters are region specific and can contain multiple, different instance types (though an instance can only be a member of a single cluster).

Amazon ECS obtains a Docker image for repository. This repository can be on AWS or on other infrastructure.

Deploying a Container

To deploy a container, you need to do the following:

1. **Define a task**. This is where you assign the name, provide the image name (important for locating the correct image), and decide on the amount of memory and CPU needed.
2. **Define the service**. In this step, you decide how many instances of the task you want to run in the cluster and any Elastic Load Balancing load balancers that you want to register the instances with.
3. **Create the Amazon ECS cluster**. This is where the cluster is created and also where you specify the number of instances to run. The cluster can run across multiple Availability Zones.
4. **Create the stack**. A stack of instances is created based on the configuration information provided. You can monitor the creation of the stack in the AWS Management Console. Creation of the stack is usually completed in less than five minutes.

Management

Amazon ECS can be configured using the AWS Management Console, the AWS CLI, or the Amazon ECS CLI.

Monitoring Amazon ECS

The primary tool used for monitoring your Amazon ECS clusters is Amazon CloudWatch. Amazon CloudWatch collects Amazon ECS metric data in one-minute periods and sends them to Amazon CloudWatch. Amazon CloudWatch stores these metrics for a period of two weeks. You can monitor the CPU and memory reservation and utilization across your cluster as a whole and the CPU and memory utilization on the services in your cluster. You can use Amazon CloudWatch to trigger alarms, set notifications, and create dashboards to monitor the services.

Once it's set up, you can view Amazon CloudWatch metrics in both the Amazon ECS console and the Amazon CloudWatch console. The Amazon ECS console provides a maximum 24-hour view while the Amazon CloudWatch console provides a fine-grained and customizable view of running services.

The other tool available is AWS CloudTrail, which will log all Amazon ECS API calls.

AWS Trusted Advisor is another source for monitoring all of your AWS resources, including Amazon ECS, to improve performance, reliability, and security.

There is no additional cost for using Amazon ECS. The only charges are for the Amazon EC2 instances or AWS Lambda requests and compute time.

Security

With Amazon ECS, you need to do the following:

1. Control who can create task definitions.
2. Control who can deploy clusters.
3. Control who can access the Amazon EC2 instances.

IAM is the tool used for the first two necessities. For controlling access to Amazon EC2 instances, the tools described in the Amazon EC2 section still apply. You can use IAM roles, security groups, and (because these Amazon EC2 instances are located in an Amazon VPC) network Access Control Lists (ACLs) and route tables to control access to the Amazon EC2 instances.

AWS Elastic Beanstalk

AWS Elastic Beanstalk enables you to deploy and manage applications in the AWS Cloud without worrying about the infrastructure that runs those applications. AWS Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and AWS Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.

Languages Supported in AWS Elastic Beanstalk

AWS Elastic Beanstalk supports applications developed in the following languages:

- Java
- PHP
- .NET
- Node.js
- Python
- Ruby

Services that AWS Elastic Beanstalk Deploys

AWS Elastic Beanstalk automates the deployment of Amazon VPC, multiple compute instances (across multiple Availability Zones), security groups (both for instances and load balancers), Auto Scaling, Amazon S3 buckets, Amazon CloudWatch alarms, and domain names. Even though AWS Elastic Beanstalk automates the deployment of your environment, you can make modifications as you see fit.

Implementation

There are a number of tools that you can use to create and manage the AWS Elastic Beanstalk environment. These tools are as follows:

- AWS Management Console
- AWS Elastic Beanstalk CLI
- AWS SDK for Java
- AWS Toolkit for Eclipse
- AWS SDK for .NET
- AWS Toolkit for Visual Studio
- AWS SDK for Node.js
- AWS SDK for PHP (requires PHP 5.2 or later)
- Boto (AWS SDK for Python)
- AWS SDK for Ruby

You deploy a new application on AWS Elastic Beanstalk by uploading a source bundle. This source bundle can be a ZIP or a WAR file (you can include multiple WAR files inside of your ZIP file). This source bundle cannot be larger than 512 MB.

Management

AWS Elastic Beanstalk is by definition highly available. It creates multiple compute instances, locates those compute instances in two Availability Zones, and creates a load

balancer (a Classic Elastic Load Balancing load balancer) to spread the compute load across these multiple instances.

You can configure Auto Scaling to establish a minimum number of instances, a maximum number of instances, and what parameters would trigger either an increase or decrease in the number of instances. These triggers can either be time-based or event-based.

When you go into the AWS Elastic Beanstalk console, it displays all of the environments running in that region, sorted by application. You can further drill down by application name to see the environments, application versions, and saved configurations. There are limits on the number of applications, application versions, and environments that you can have.

As with Amazon EC2, you are responsible for patching and updating both the guest operating system and your application with AWS Elastic Beanstalk.

Making Changes in Your AWS Elastic Beanstalk Environment

It is recommended to use the AWS Elastic Beanstalk console and CLI when making changes to your AWS Elastic Beanstalk environments and configuration. Although it is possible to modify your AWS Elastic Beanstalk environment using other service consoles, CLI commands, or AWS SDKs, those changes will not be reflected in the AWS Elastic Beanstalk console and you will not be able to monitor, change, or save your environment. It will also cause issues when you attempt to terminate your environment without using the AWS Elastic Beanstalk console.

Monitoring Your AWS Elastic Beanstalk Environment

You can monitor the overall health of your environment using either the basic health reporting or enhanced health reporting and monitoring systems. Basic health reporting gives an overall status (via color key) of the environment. It does not publish any metrics to Amazon CloudWatch. Enhanced health reporting and monitoring not only provides status via color key, but it also provides a descriptor that helps indicate the severity and cause of the problem. You can also configure enhanced health reporting and monitoring to publish metrics to Amazon CloudWatch as custom metrics.

You can retrieve log files from the individual instances or have those log files uploaded to an Amazon S3 bucket for more permanent storage.

Security

When you create an environment with AWS Elastic Beanstalk, you are prompted to create two IAM roles:

Service role This is used by AWS Elastic Beanstalk to access other AWS Cloud services.

Instance profile This is applied to the instances created in your environment, and it allows them to upload logs to Amazon S3 and perform other tasks.

In addition to these two roles, you can create polices in IAM and attach them to users, groups, and roles. Just as in Amazon EC2, in AWS Elastic Beanstalk you need to create an Amazon EC2 key pair to access the guest operating system within your compute instances. Access would be via either Port 22 (for SSH) or Port 3389 (for RDP).

AWS Elastic Beanstalk allows web developers to deploy highly scalable and highly available web infrastructure without having to know or understand services like Elastic Load Balancing, Auto Scaling groups, or Availability Zones.

AWS Lambda

AWS Lambda is a serverless compute service. It runs your code in response to events. AWS Lambda automatically manages the underlying compute resources with no server configuration from you. There is no need to provision or manage servers.

The execution of code (called an AWS Lambda function) can be triggered by events internal to your AWS infrastructure or external to it.

AWS Lambda is a regionally-based service running across multiple Availability Zones. This makes it highly available and highly scalable. Because the code is stateless, it is executed almost automatically without deployment or configuration delays.

Implementation

Implementation for AWS Lambda involves the following steps:

1. Configure an AWS Lambda function.
2. Create a deployment package.
3. Upload the deployment package.
4. Create an invocation type.

The languages available to write your code are Node.js, Java, Python, and C#. The language that you choose also controls what tools are available for authoring your code. These tools can include the AWS Lambda console, Visual Studio, .NET Core, Eclipse, or your own authoring environment. You use these languages to create a deployment package.

Once a deployment package has been created, you then upload it to create an AWS Lambda function. The tools you can use to do so are the AWS Lambda console, CLI, and AWS SDKs.

The code that you have written is part of what is called an *AWS Lambda function*. The other elements of an AWS Lambda function are as follows:

- Memory
- Maximum execution time
- IAM role
- Handler name

The amount of memory that you specify controls the CPU power. The default amount of memory allocated per AWS Lambda function is 512 MB, but this can range from 128 MB to 1,536 MB in 64 MB increments.

Load Balancing

Elastic Load Balancing distributes incoming application traffic across multiple Amazon EC2 instances, in multiple Availability Zones within a single AWS Region. Elastic Load Balancing supports two types of load balancers:

- Application Load Balancers
- Classic Load Balancers

The Elastic Load Balancing load balancer serves as a single target, which increases the availability of your application. You can add and remove instances from your load balancer as your needs change without disrupting the overall flow of requests to your application. Elastic Load Balancing, working in conjunction with Auto Scaling, can scale your load balancer as traffic to your application changes over time. Refer to Chapter 10, "High Availability," for details on how Auto Scaling works.

You can configure health checks and send requests only to the healthy instances. You can also offload the work of encryption and decryption to your load balancer. See Table 5.6 for differences between the Classic Load Balancer and the Application Load Balancer.

TABLE 5.6 Classic Load Balancer and Application Load Balancer

Feature	Classic Load Balancer	Application Load Balancer
Protocols	HTTP, HTTPS, TCP, SSL	HTTP, HTTPS
Platforms	Amazon EC2-Classic, Amazon EC2-VPC	Amazon EC2-VPC
Sticky sessions (cookies)	✔	Load balancer generated
Idle connection timeout	✔	✔
Connection draining	✔	✔
Cross-zone load balancing	Can be enabled	Always enabled
Health checks	✔	Improved
Amazon CloudWatch metrics	✔	Improved
Access logs	✔	Improved
Host-based routing		✔
Path-based routing		✔

TABLE 5.6 Classic Load Balancer and Application Load Balancer *(continued)*

Feature	Classic Load Balancer	Application Load Balancer
Route to multiple ports on a single instance		✔
HTTP/2 support		✔
WebSocket support		✔
Load balancer deletion protection		✔

Load Balancing Implementation

Because the Classic Load Balancer and the Application Load Balancer distribute traffic in different ways, implementation is different. Let's start with the Classic Load Balancer. Both Load Balancers will be examined in this section of the chapter.

Classic Load Balancer

With a *Classic Load Balancer*, there are a number of parameters that you need to configure. These include the following:

- Choose VPC
- Choose subnets
- Define protocols and ports
- Choose Internet facing or internal
- Determine security groups
- Configure health check
- Assign Amazon EC2 instances

If you have multiple VPCs, you determine which VPC will contain the Application Load Balancer. Load balancers cannot be shared among VPCs. From there, you are given a list of Availability Zones in which you have created subnets. In order for your application to be considered highly available, you must specify at least two Availability Zones. If you need additional Availability Zones, you need to create subnets in those Availability Zones.

High Availability

You can distribute incoming traffic across your Amazon EC2 instances in a single Availability Zone or multiple Availability Zones. The Classic Load Balancer automatically scales its request handling capacity in response to incoming application traffic.

Health Checks

The Classic Load Balancer can detect the health of Amazon EC2 instances. When it detects unhealthy Amazon EC2 instances, it no longer routes traffic to those instances and spreads the load across the remaining healthy instances.

Security Features

When using Amazon Virtual Private Cloud (Amazon VPC), you can create and manage security groups associated with Classic Load Balancers to provide additional networking and security options. You can also create a Classic Load Balancer without public IP addresses to serve as an internal (non-Internet-facing) load balancer.

SSL Offloading

Classic Load Balancers support SSL termination, including offloading SSL decryption from application instances, centralized management of SSL certificates, and encryption to back-end instances with optional public key authentication.

Flexible cipher support allows you to control the ciphers and protocols the load balancer presents to clients.

Sticky Sessions

Classic Load Balancers support the ability to stick user sessions to specific EC2 instances using cookies. Traffic will be routed to the same instances as the user continues to access your application.

IPv6 Support

Classic Load Balancers support the use of Internet Protocol versions 4 and 6 (IPv4 and IPv6). IPv6 support is currently unavailable for use in VPC.

Layer 4 or Layer 7 Load Balancing

You can load balance HTTP/HTTPS applications and use layer 7-specific features, such as X-Forwarded and sticky sessions. You can also use strict layer 4 load balancing for applications that rely purely on the TCP protocol.

Operational Monitoring

Classic Load Balancer metrics, such as request count and request latency, are reported by Amazon CloudWatch.

Logging

Use the Access Logs feature to record all requests sent to your load balancer and store the logs in Amazon S3 for later analysis. The logs are useful for diagnosing application failures and analyzing web traffic.

You can use AWS CloudTrail to record classic load balancer API calls for your account and deliver log files. The API call history enables you to perform security analysis, resource change tracking, and compliance auditing.

You can assign the Classic Load Balancer to any Amazon EC2 instance that is in your VPC, including Amazon EC2 instances not in the Availability Zones that you specified. If you do so, the Classic Load Balancer will not route traffic to those Amazon EC2 instances.

Application Load Balancer

With an *Application Load Balancer*, there are a number of parameters that you need to configure. These include the following:

- Name of your load balancer
- Security groups
- The VPC
- Availability Zones to be used
- Internet-facing or internal
- IP addressing scheme to be used
- Configure one or more listeners
- Configure listener rules
- Define target group

Table 5.6 demonstrates the features of the Application Load Balancer. In order to become a proficient Systems Operator on AWS, you need to know what these features do. The features of the Application Load Balancer are defined in the following sections.

Content-Based Routing

If your application is composed of several individual services, an Application Load Balancer can route a request to a service based on the content of the request.

Host-Based Routing

You can route a client request based on the Host field of the HTTP header, allowing you to route to multiple domains from the same load balancer.

Path-Based Routing

You can route a client request based on the URL path of the HTTP header.

Containerized Application Support

You can now configure an Application Load Balancer to load balance containers across multiple ports on a single Amazon EC2 instance. Amazon EC2 Container Service (Amazon ECS)

allows you to specify a dynamic port in the ECS task definition, giving the container an unused port when it is scheduled on the EC2 instance. The ECS scheduler automatically adds the task to the ELB using this port.

HTTP/2 Support

HTTP/2 is a new version of the Hypertext Transfer Protocol (HTTP) that uses a single, multiplexed connection to allow multiple requests to be sent on the same connection. It also compresses header data before sending it out in binary format, and it supports TLS connections to clients.

WebSockets Support

WebSockets allows a server to exchange real-time messages with end users without the end users having to request (or poll) the server for an update. The WebSockets protocol provides bi-directional communication channels between a client and a server over a long-running TCP connection.

Native IPv6 Support

Application Load Balancers support native Internet Protocol version 6 (IPv6) in a VPC. This will allow clients to connect to the Application Load Balancer via IPv4 or IPv6.

Sticky Sessions

Sticky sessions are a mechanism to route requests from the same client to the same target. The Application Load Balancer supports sticky sessions using load balancer generated cookies. If you enable sticky sessions, the same target receives the request and can use the cookie to recover the session context. Stickiness is defined at a target group level.

Health Checks

An Application Load Balancer routes traffic only to healthy targets. With an Application Load Balancer, you get improved insight into the health of your applications in two ways:

1. Health check improvements that allow you to configure detailed error codes from 200-399. The health checks allow you to monitor the health of each of your services behind the load balancer.
2. New metrics that give insight into traffic for each of the services running on an Amazon EC2 instance.

High Availability

An Application Load Balancer requires you to specify more than one Availability Zone. You can distribute incoming traffic across your targets in multiple Availability Zones. An Application Load Balancer automatically scales its request-handling capacity in response to incoming application traffic.

Layer-7 Load Balancing

You can load balance HTTP/HTTPS applications and use layer 7-specific features, such as X-Forwarded-For (XFF) headers.

HTTPS Support

An Application Load Balancer supports HTTPS termination between the clients and the load balancer. Application Load Balancers also offer management of SSL certificates through AWS Identity and Access Management (IAM) and AWS Certificate Manager for predefined security policies.

Operational Monitoring

Amazon CloudWatch reports Application Load Balancer metrics, such as request counts, error counts, error types, and request latency.

Logging

You can use the Access Logs feature to record all requests sent to your load balancer and store the logs in Amazon S3 for later analysis. The logs are compressed and have a `.gzip` file extension. The compressed logs save both storage space and transfer bandwidth, and they are useful for diagnosing application failures and analyzing web traffic.

You can also use AWS CloudTrail to record Application Load Balancer API calls for your account and deliver log files. The API call history enables you to perform security analysis, resource change tracking, and compliance auditing.

Delete Protection

You can enable deletion protection on an Application Load Balancer to prevent it from being accidentally deleted.

Request Tracing

The Application Load Balancer injects a new custom identifier "X-Amzn-Trace-Id" HTTP header on all requests coming into the load balancer. Request tracing allows you to track a request by its unique ID as the request makes its way across various services that make up your websites and distributed applications. You can use the unique trace identifier to uncover any performance or timing issues in your application stack at the granularity of an individual request.

AWS Web Application Firewall

You can now use AWS WAF to protect your web applications on your Application Load Balancers. AWS WAF is a web application firewall that helps protect your web applications from common web exploits that could affect application availability, compromise security, or consume excessive resources.

Load Balancing Management

Elastic Load Balancing is a highly available and scalable service.

You can create, access, and manage your load balancers using any of the following interfaces:

AWS Management Console Provides a web interface that you can use to access Elastic Load Balancing.

AWS CLI Provides commands for a broad set of AWS Cloud services, including Elastic Load Balancing, and it is supported on Windows, Mac, and Linux.

AWS SDKs Provides language-specific APIs and takes care of many of the connection details, such as calculating signatures, handling request retries, and error handling.

Query API Provides low-level API actions that you call using HTTPS requests. Using the Query API is the most direct way to access Elastic Load Balancing, but it requires that your application handle low-level details such as generating the hash to sign the request and error handling.

There are three tools for managing both Classic Load Balancers and Application Load Balancers. These are Amazon CloudWatch, AWS CloudTrail, and access logs. In addition, Application Load Balancers has a feature that allows request tracing to track HTTP requests from clients to targets.

Amazon CloudWatch

Amazon CloudWatch provides a number of metrics available to track the performance of the Elastic Load Balancing load balancers. These metrics include number of health hosts, average latency, number of requests, and number of connections, among others. These metrics are updated on a 60-second interval.

AWS CloudTrail

AWS CloudTrail can capture the API calls used by the Elastic Load Balancing load balancers and deliver those logs to an Amazon S3 bucket that you specify. From there you can analyze the logs as needed. The information these logs will include are things like the time the API was invoked, what invoked the API, and the parameters of the request.

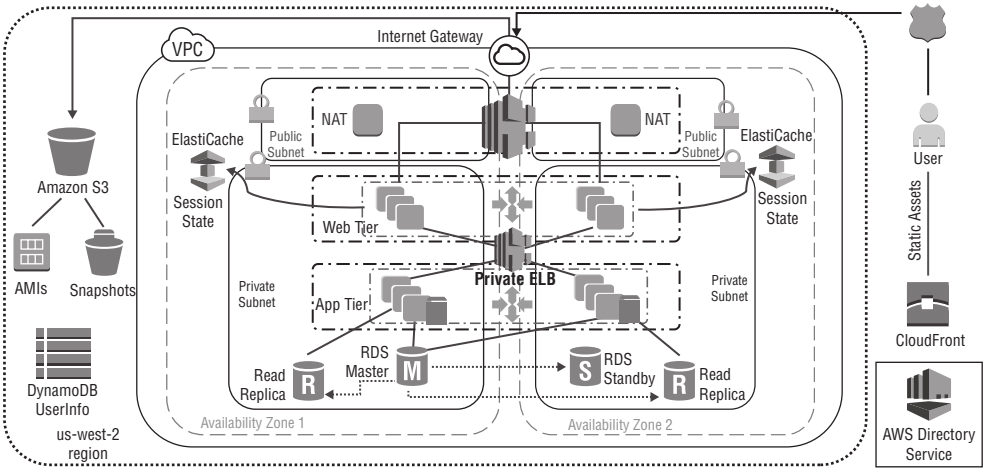
Access Logs

Access logs are an optional feature of Elastic Load Balancing. Access logs capture greater detailed information than AWS CloudTrail does, including such information as latencies, request paths, IP addresses of clients making the requests, and the instance processing the request, among other information.

Request Tracing

Request tracing is used to track HTTP requests from clients to targets or other services. When the load balancer receives a request from a client, it adds or updates the X-Amzn-Trace-Id header before sending the request to the target. Any services or applications between the load balancer and the target can also add or update this header. The contents of the header are logged in access logs.

FIGURE 10.5 Highly available three-tier architecture



Network Address Translation (NAT) Gateways

Network Address Translation (NAT) gateways were covered in Chapter 5. NAT servers allow traffic from private subnets to traverse the Internet or connect to other AWS Cloud services. Individual NAT servers can be a single point of failure. The NAT gateway is a managed device, and each NAT gateway is created in a specific Availability Zone and implemented with redundancy in that Availability Zone. To achieve optimum availability, use NAT gateways in each Availability Zone.

If you have resources in multiple Availability Zones, they share one NAT gateway. When the NAT gateway's Availability Zone is down, resources in the other Availability Zones will lose Internet access. To create an Availability Zone-independent architecture, create a NAT gateway in each Availability Zone and configure your routing to ensure that resources use the NAT gateway in the same Availability Zone.

Elastic Load Balancing

As discussed in Chapter 5, Elastic Load Balancing comes in two types: Application Load Balancer and Classic Load Balancer. *Elastic Load Balancing* allows you to decouple an application's web tier (or front end) from the application tier (or back end). In the event of a node failure, the Elastic Load Balancing load balancer will stop sending traffic to the affected Amazon EC2 instance, thus keeping the application available, although in a deprecated state. Self-healing can be achieved by using Auto Scaling. For a refresher on Elastic Load Balancing, refer to Chapter 5.