

AWS[®] Certified Cloud Practitioner **STUDY GUIDE**

FOUNDATIONAL (CLF-C01) EXAM

Includes interactive online learning environment and study tools:

Two custom practice exams

100 electronic flashcards

Searchable key term glossary

**BEN PIPER
DAVID CLINTON**

 **SYBEX[®]**
A Wiley Brand

in the additional labor and time involved in performing database backups, upgrades, and monitoring, you may find that using RDS to handle these tasks for you would be more cost effective.

Operational Excellence

Operational excellence is fundamentally about automating the processes required to achieve and maintain the other four goals of reliability, performance efficiency, cost optimization, and security. In reality, few organizations automate everything, so operational excellence is more of a journey than a goal. But the idea behind operational excellence is to incrementally improve and automate more activities for the purpose of strengthening the other pillars. Here are some simple examples of how automation can help achieve operational excellence:

Reliability Use elastic load balancing health checks to monitor the health of an application running on several EC2 instances. When an instance fails, route users away from the failed instance and create a new one.

Performance efficiency Use EC2 Auto Scaling dynamic scaling policies to scale in and out automatically, ensuring you always have as many instances as you need and no more.

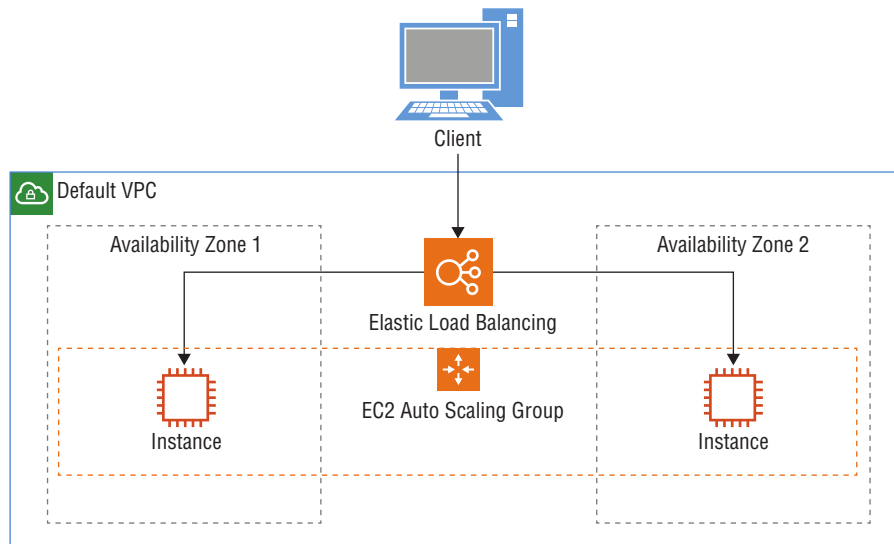
Security Use CodeBuild to automatically test new application code for security vulnerabilities. When deploying an application, use CloudFormation to automatically deploy fresh, secure infrastructure, rather than following a manual checklist.

Cost optimization Automatically shut down or decommission unused resources. For example, implement S3 object life cycle configurations to delete unneeded objects. Or if you have EC2 instances that are used for testing only during business hours, you can automatically shut them down at the end of the workday and restart them the following morning.

Keep in mind that these examples are by no means exhaustive. Many opportunities for automation aren't obvious until there's a failure. For instance, if someone deletes an application load balancer that's operating as the front end of a critical web application, recovering from such an event would entail performing some quick manual work-arounds and re-creating the load balancer. Understanding how failures affect your application can help you avoid such failures in the future and automate recovery when they occur.

A Highly Available Web Application Using Auto Scaling and Elastic Load Balancing

The first scenario we'll consider is a highly available web application that you'll implement using the topology shown in Figure 12.1.

FIGURE 12.1 A highly available web application using Auto Scaling and elastic load balancing

Don't worry if this looks intimidating. We'll start by taking a high-level look at how these components fit together, and then we'll dig into the configuration specifics. EC2 Auto Scaling will initially provision two EC2 instances running in different Availability Zones inside the default VPC. Every default VPC has a default subnet in each Availability Zone, so there's no need to create subnets explicitly. An application load balancer will proxy the connections from clients on the internet and distribute those connections to the instances that are running the popular open-source Apache web server.

We'll implement this scenario in four basic steps:

1. Create an inbound security group rule in the VPC's default security group.
2. Create an application load balancer.
3. Create a launch template.
4. Create an Auto Scaling group.

Creating an Inbound Security Group Rule

Every VPC comes with a default security group. Recall that security groups block traffic that's not explicitly allowed by a rule, so we'll start by adding an inbound rule to the VPC's default security group to allow only inbound HTTP access to the application load balancer and the instances. Complete Exercise 12.1 to create the inbound rule.

EXERCISE 12.1**Create an Inbound Security Group Rule**

You'll start by modifying the default security group to allow inbound HTTP access. The application load balancer that you'll create later, as well as the instances that Auto Scaling will create, will use this security group to permit users to connect to the web application.

1. Browse to the EC2 service console. In the navigation pane, choose the Security Groups link.
2. Select the Create Security Group button.
3. Select the default security group for the default VPC.
4. Select the Inbound tab.
5. Select the Edit button.
6. Select the Add Rule button.
7. Under the Type drop-down list, select HTTP. The Management Console automatically populates the Protocol field with TCP and the Port Range field with 80, which together correspond to the HTTP protocol used for web traffic. It also populates the Source field with the IP subnet 0.0.0.0/0 and IPv6 subnet ::0/0, allowing traffic from all sources. You can optionally change the Source field to reflect the subnet of your choice or a specific IP address. For example, to allow only the IP address 198.51.100.100, you'd enter **198.51.100.100/32**. Refer to Figure 12.2 to see what your new inbound rule should look like.
8. Select the Save button.

FIGURE 12.2 Modifying the default security group

Type	Protocol	Port Range	Source	Description
All traffic	All	0 - 65535	Custom sg-089f1a1fb14b9b704	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom 0.0.0.0/0, ::0	e.g. SSH for Admin Desktop

Add Rule

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Cancel Save

Creating an Application Load Balancer

Next, you need to create an application load balancer that will distribute connections to the instances in the Auto Scaling group that you'll create in a later exercise. The load balancer will route traffic to healthy instances using a round-robin algorithm that distributes traffic evenly to the instances without regard for how busy those instances are.

You'll configure the load balancer to perform a health check to monitor the health of the application on each instance. If the application fails on an instance, the load balancer will route connections away from the failed instance. The Auto Scaling group will also use this health check to determine whether an instance is healthy or needs to be replaced. Follow the steps in Exercise 12.2 to create the application load balancer.

EXERCISE 12.2

Create an Application Load Balancer

Now you'll create an application load balancer to receive incoming connections from users. The application load balancer will distribute those connections to the instances in the Auto Scaling group that you'll create later.

1. While in the EC2 service console, in the navigation pane on the left, choose the Load Balancers link.
 2. Select the Create Load Balancer button.
 3. Under Application Load Balancer, select the Create button.
 4. In the Name field, type **sample-web-app-load-balancer**.
 5. For the Scheme, select the radio button next to Internet-facing. This will assign the load balancer a publicly resolvable domain name and allow the load balancer to receive connections from the internet.
 6. In the IP Address Type drop-down list, select IPv4.
 7. Under Listeners, make sure the Load Balancer Protocol field is HTTP and the Load Balancer Port field is 80. Refer to Figure 12.3 for an example of what your basic load balancer configuration should look like.
 8. On the Configure Load Balancer page, under the Availability Zones section, in the VPC drop-down, make sure your default VPC is selected.
 9. Select at least two Availability Zones. Refer to Figure 12.4 for an example.
 10. Select the button titled Next: Configure Security Settings.
 11. You may see a message warning you that your load balancer isn't using a secure listener. Select the button titled Next: Configure Security Groups.
 12. Select the default security group for the VPC.
-

FIGURE 12.3 Application load balancer basic configuration

1. Configure Load Balancer 2. Configure Security Settings 3. Configure Security Groups 4. Configure Routing 5. Register Targets 6. Review

Step 1: Configure Load Balancer

Basic Configuration

To configure your load balancer, provide a name, select a scheme, specify one or more listeners, and select a network. The default configuration is an Internet-facing load balancer in the selected network with a listener that receives HTTP traffic on port 80.

Name ⓘ sample-web-app-load-balancer

Scheme ⓘ ☒ internet-facing ☐ internal

IP address type ⓘ ipv4

Listeners

A listener is a process that checks for connection requests, using the protocol and port that you configured.

Load Balancer Protocol	Load Balancer Port
HTTP	80

[Add listener](#)

[Cancel](#) [Next: Configure Security Settings](#)

FIGURE 12.4 Application load balancer Availability Zones configuration

1. Configure Load Balancer 2. Configure Security Settings 3. Configure Security Groups 4. Configure Routing 5. Register Targets 6. Review

Step 1: Configure Load Balancer

Availability Zones

Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. You can specify only one subnet per Availability Zone. You must specify subnets from at least two Availability Zones to increase the availability of your load balancer.

VPC ⓘ vpc-0badfe1ce747af365 (172.31.0.0/16) (default)

Availability Zones

Availability Zone	Subnet	IPv4 address
<input checked="" type="checkbox"/> us-east-1a	subnet-0bb1145e4ecd4bb67	Assigned by AWS
<input checked="" type="checkbox"/> us-east-1b	subnet-010922468fee85c5a	Assigned by AWS
<input type="checkbox"/> us-east-1c	subnet-0a398eaa1d41919e	
<input type="checkbox"/> us-east-1d	subnet-069b5d977f744889b	
<input type="checkbox"/> us-east-1e	subnet-0e6d1c742bd846aa0	
<input type="checkbox"/> us-east-1f	subnet-00f4e442f66a4514a	

[Cancel](#) [Next: Configure Security Settings](#)

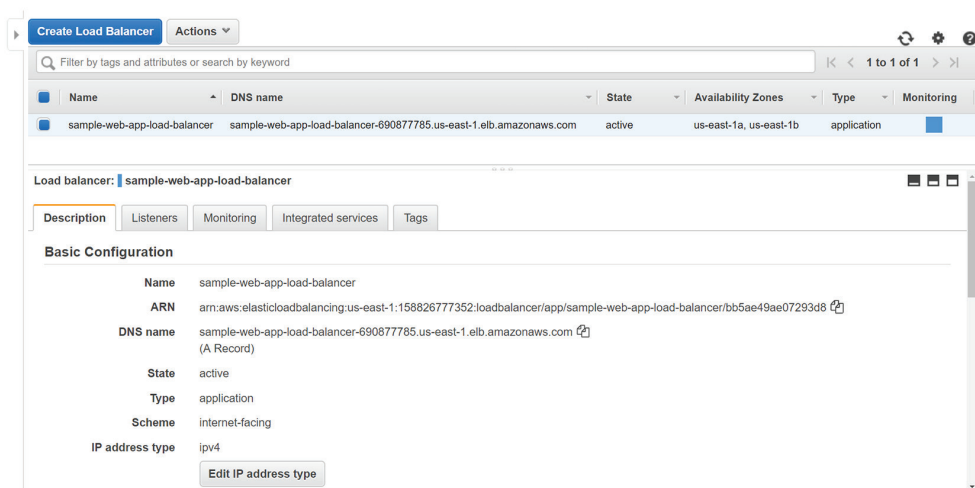
13. Select the button titled Next: Configure Routing.
14. Under Target Group, in the Target Group drop-down list, select New Target Group.
15. In the Name field, type **sample-web-app-target-group**.
16. Next to Target Type, select the Instance radio button.

EXERCISE 12.2 (continued)

17. For Protocol and Port, select HTTP and 80, respectively.
18. Under Health Checks, make sure Protocol and Path are HTTP and /, respectively.
19. Select the button titled Next: Register Targets.
20. The Auto Scaling group that you'll create will add instances to the target group, so there's no need to do that manually here. Select the button titled Next: Review.
21. Review your settings, and select the Create button. It may take a few minutes for your load balancer to become active.

Once AWS has provisioned the load balancer, you should be able to view its publicly resolvable DNS name and other details, as shown in Figure 12.5. Make a note of the DNS name because you'll need it later.

FIGURE 12.5 Application load balancer details



Creating a Launch Template

Before creating the Auto Scaling group, you need to create a launch template that Auto Scaling will use to launch the instances and install and configure the Apache web server software on them when they're launched. Because creating the launch template by hand would be cumbersome, you'll instead let CloudFormation create it by deploying the CloudFormation template at <https://s3.amazonaws.com/aws-ccp/launch-template.yaml>. The launch template that CloudFormation will create will install Apache on each instance that Auto Scaling provisions. You can also create a custom launch template for your own application. Complete Exercise 12.3 to get some practice with CloudFormation and create the launch template.

EXERCISE 12.3**Create a Launch Template**

In this exercise, you'll use CloudFormation to deploy an EC2 launch template that Auto Scaling will use to launch new instances.

1. Browse to the CloudFormation service console. Make sure you're in the AWS Region where you want your instances created.
2. Select the Create Stack button.
3. Under Choose A Template, select the radio button titled Specify An Amazon S3 Template URL.
4. In the text field, enter **`https://s3.amazonaws.com/aws-ccp/launch-template.yaml`**.
5. Select the Next button.
6. In the Stack Name field, enter **`sample-app-launch-template`**.
7. From the drop-down list, select the instance type you want to use, or stick with the default t2.micro instance type.
8. Select the Next button.
9. On the Options screen, stick with the defaults and select the Next button.
10. Review your settings, and select the Create button.

CloudFormation will take you to the Stacks view screen. Once the stack is created, the status of the sample-app-launch-template stack will show as `CREATE_COMPLETE`, indicating that the EC2 launch template has been created successfully.

Creating an Auto Scaling Group

EC2 Auto Scaling is responsible for provisioning and maintaining a certain number of healthy instances. By default, Auto Scaling provides reliability by automatically replacing instances that fail their EC2 health check. You'll reconfigure Auto Scaling to monitor the ELB health check and replace any instances on which the application has failed.

To achieve performance efficiency and make this configuration cost-effective, you'll create a dynamic scaling policy that will scale the size of the Auto Scaling group in or out between one and three instances, depending on the average aggregate CPU utilization of the instances. If the CPU utilization is greater than 50 percent, it indicates a heavier load, and Auto Scaling will scale out by adding another instance. On the other hand, if the utilization drops below 50 percent, it indicates that you have more instances than you need, so Auto Scaling will scale in. This is called a *target tracking policy*.

**NOTE**

EC2 reports these metrics to CloudWatch, where you can graph them to analyze your usage patterns over time. CloudWatch stores metrics for up to 15 months.

This may sound like a lot to do, but the wizard makes it easy. Complete Exercise 12.4 to create and configure the Auto Scaling group.

EXERCISE 12.4

Create an Auto Scaling Group

In this exercise, you'll create an Auto Scaling group that will provision your web server instances using the launch template.

1. Browse to the EC2 service console.
2. In the navigation pane, choose the Launch Templates link.
3. Select the launch template.
4. Select the Action button, and choose Create Auto Scaling Group.
5. In the Group Name field, enter **sample-web-app**.
6. In the Group Size field, enter **2**.
7. In the Network drop-down list, select the default VPC.
8. In the Subnet drop-down list, select at least two subnets. Refer to Figure 12.6 to get an idea of what the basic configuration should look like.

FIGURE 12.6 Auto Scaling group basic configuration

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

Cancel and Exit

Create Auto Scaling Group

Group name ⓘ sample-web-app

Launch Template ⓘ lt-0193bee3ea9229dc8

Launch Template Version ⓘ Default [Create new launch template](#)

Launch Template Description ⓘ -

Fleet Composition

- ☒ Adhere to the launch template
The launch template determines the instance type and purchase option (On-Demand or Spot).
- ☐ Combine purchase options and instances
Choose a mix of On-Demand instances and Spot instances and multiple instance types. Spot instances are automatically launched at the lowest price available.

Group size ⓘ Start with 2 instances

Network ⓘ vpc-0ce9ee20679ff8a5c (172.31.0.0/16) (default) [Create new VPC](#)

Subnet ⓘ

- subnet-010c41e364e786dee(172.31.80.0/20) | Default x in us-east-1a
- subnet-09e4c809d9a62f302(172.31.16.0/20) | Default x in us-east-1b

[Create new subnet](#)

[Cancel](#) [Next: Configure scaling policies](#)

9. Expand the Advanced Details section.
10. Select the check box next to Receive Traffic From One Or More Load Balancers.

11. In the Target Groups field, select sample-web-app-target-group.
12. Next to Health Check Type, select the ELB radio button.
13. Select the button titled Next: Configure Scaling Policies.
14. Under Create Auto Scaling Group, select the Use Scaling Policies To Adjust The Capacity Of This Group radio button. This will create a dynamic scaling policy that will automatically scale in or out based on the average aggregate CPU utilization of the instances in the Auto Scaling group.
15. Adjust the minimum and maximum group size to scale between one and three instances. This will ensure that the group always has at least one instance but never more than three instances.
16. Under Scale Group Size, select the Metric Type Average CPU Utilization.
17. For the Target Value, enter 50. This will cause Auto Scaling to attempt to keep the average CPU utilization of each instance at 50 percent. If the average CPU utilization falls below 50 percent, Auto Scaling will scale in, leaving only one instance in the group. If the average CPU utilization rises above 50 percent, Auto Scaling will add more instances to the group for a total of up to three.
18. Select the button titled Next: Configure Notifications.
19. Select the button titled Next: Configure Tags.
20. Select the Review button.
21. Review the settings, and select the Create Auto Scaling Group button.
22. Select the View Your Auto Scaling Groups link, and wait a few minutes for Auto Scaling to provision the instances.
23. Open a web browser, and browse to the DNS name of the application load balancer that you noted in Exercise 12.2. You should see the Apache Linux AMI test page, as shown in Figure 12.7.

FIGURE 12.7 The Apache Linux AMI test page

