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

CloudWatch Alarms

A CloudWatch alarm watches over the value of a single metric. If the metric crosses a threshold that you specify (and stays there), the alarm will take an action. For example, you might configure an alarm to take an action when the average CPU utilization for an instance exceeds 80% for five minutes. The action can be one of the following:

Notification using Simple Notification Service The *Simple Notification Service* (SNS) allows applications, users, and devices to send and receive notifications from AWS. SNS uses a publisher-subscriber model, wherein a publisher such as an AWS service generates a notification and a subscriber such as an end user receives it. The communication channel that SNS uses to map publishers and subscribers is called a *topic*. SNS can send notifications to subscribers via a variety of protocols including the following:

- HTTP(S)
- Simple Queue Service (SQS)
- Lambda
- Mobile push notification
- Email
- Email-JSON
- Short Message Service (SMS) text messages

Auto Scaling action By specifying an EC2 Auto Scaling action, the EC2 Auto Scaling service can add or remove EC2 instances in response to changing demand. For example, if a metric indicates that instances are overburdened, you can have EC2 Auto Scaling respond by adding more instances.

EC2 action If you're monitoring a specific instance that's having a problem, you can use an EC2 action to stop, terminate, or recover the instance. Recovering an instance migrates the instance to a new EC2 host, something you may need to do if there's a physical hardware problem on the hardware hosting the instance.

CloudWatch Dashboards

CloudWatch dashboards are your one-stop shop for keeping an eye on all of your important metrics. You can create multiple dashboards and add to them metric graphs, the latest values for a metric, and CloudWatch alarms. You can save your dashboards for future use and share them with others. Dashboards can also visualize metrics from multiple AWS Regions, so you can keep an eye on the global health of your infrastructure. Check out Figure 6.17 for a sample CloudWatch dashboard.

EC2 Auto Scaling

EC2 Auto Scaling automatically launches preconfigured EC2 instances. The goal of Auto Scaling is to ensure you have just enough computing resources to meet user demand without over-provisioning.

Launch Configurations and Launch Templates

Auto Scaling works by spawning instances from either a launch configuration or a launch template. For the purposes of Auto Scaling, both achieve the same basic purpose of defining the instance's characteristics, such as AMI, disk configuration, and instance type. You can also install software and make custom configurations by placing commands into a Userdata script that automatically runs when Auto Scaling launches a new instance.

But there are some differences between launch configurations and launch templates. Launch templates are newer and can be used to spawn EC2 instances manually, even without Auto Scaling. You can also modify them after you create them. Launch configurations, on the other hand, can be used only with Auto Scaling. And once you create a launch configuration, you can't modify it.

Auto Scaling Groups

Instances created by Auto Scaling are organized into an Auto Scaling group. All instances in a group can be automatically registered with an application load balancer target group. The application load balancer distributes traffic to the instances, spreading the demand out evenly among them.

Desired Capacity

When you configure an Auto Scaling group, you define a desired capacity—the number of instances that you want Auto Scaling to create. Auto Scaling creates this many instances and strives to maintain the desired capacity. If you raise or lower the capacity, Auto Scaling launches or terminates instances to match.

Self-Healing

Failed instances are self-healing. If an instance fails or terminates, Auto Scaling re-creates a replacement. This way you always get the number of instances you expect.

Auto Scaling can use EC2 or elastic load balancing (ELB) health checks to determine whether an instance is healthy. EC2 health checks consider the basic health of an instance, whether it's running, and whether it has network connectivity. ELB health checks look at the health of the application running on an instance. An unhealthy instance is treated as a failed instance and is terminated, and Auto Scaling creates another in its place.

Scaling Actions

Scaling actions control when Auto Scaling launches or terminates instances. You control how many instances Auto Scaling launches or terminates by specifying a minimum and maximum group size. Auto Scaling will ensure the number of instances never goes outside of this range. Naturally, the desired capacity must rest within the minimum and maximum bounds.

Dynamic Scaling

With dynamic scaling, Auto Scaling launches new instances in response to increased demand using a process called *scaling out*. It can also scale in, terminating instances when demand ceases. You can scale in or out according to a metric, such as average CPU utilization of your instances, or based on the number of concurrent application users.

Scheduled Scaling

Instead of or in addition to dynamic scaling, Auto Scaling can scale in or out according to a schedule. This is particularly useful if your demand has predictable peaks and valleys.

Predictive scaling is a feature that looks at historic usage patterns and predicts future peaks. It then automatically creates a scheduled scaling action to match. It needs at least one day's worth of traffic data to create a scaling schedule.

EC2 Auto Scaling fulfills one of the core principles of sound cloud architecture design: don't guess your capacity needs. Auto Scaling can save money by reducing your capacity when you don't need it and improve performance by increasing it when you do.

Configuration Management

Configuration management is an approach to ensuring accurate and consistent configuration of your systems. While automation is concerned with carrying out tasks, configuration management is primarily concerned with enforcing and monitoring the internal configuration state of your instances to ensure they're what you expect. Such configuration states primarily include but aren't limited to operating system configurations and what software is installed.

As with automation in general, configuration management tools use either imperative or declarative approaches. AWS offers both approaches using two tools to help you achieve configuration management of your EC2 and on-premises instances: Systems Manager and OpsWorks.

Systems Manager

Systems Manager uses the imperative approach to get your instances and AWS environment into the state that you want.

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.

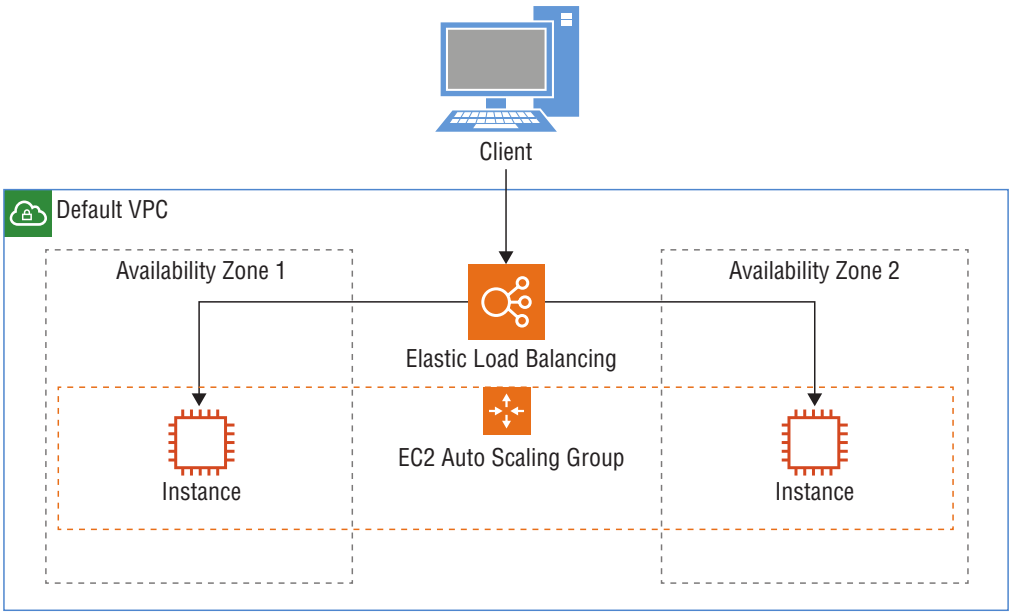
Cost optimization Automatically shut down or decommission unused resources. For example, implement S3 object life cycle configurations to delete unneeded objects. Or if you have EC2 instances that are used for testing only during business hours, you can automatically shut them down at the end of the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

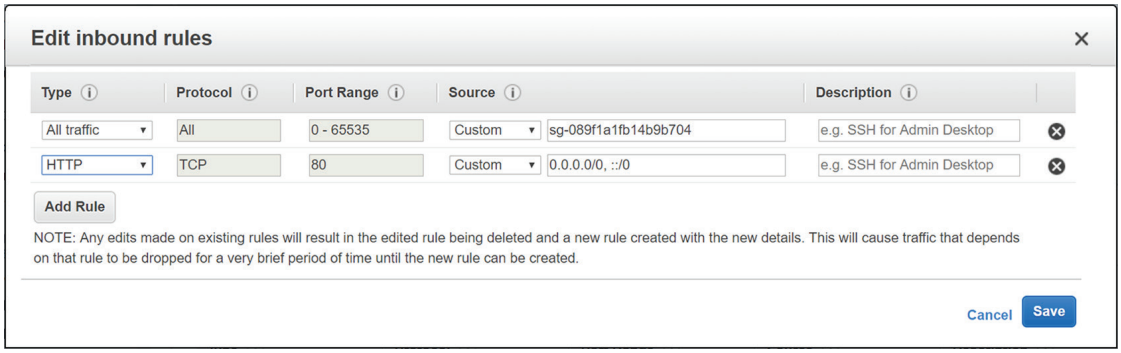
EXERCISE 12.1

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



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

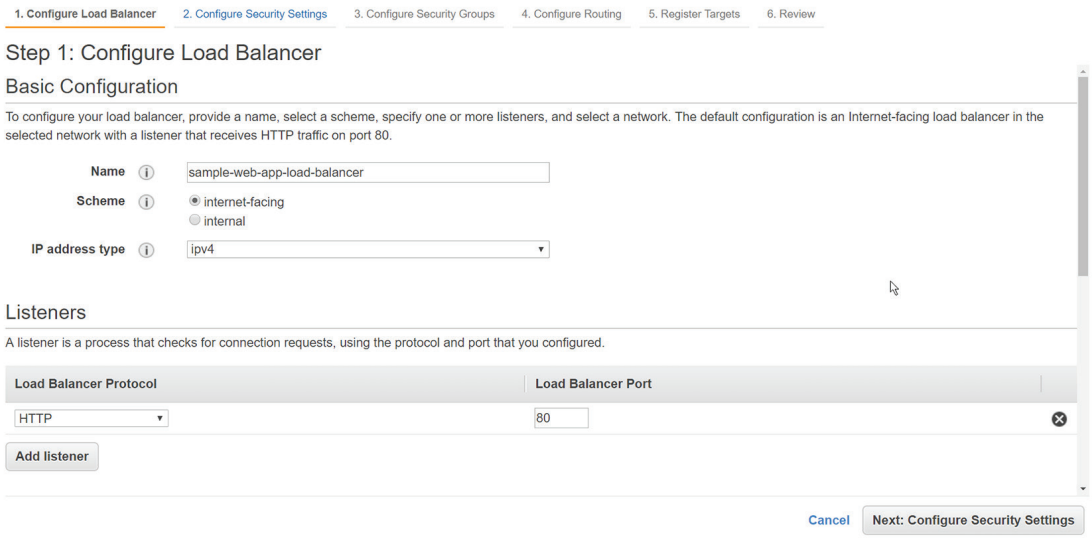
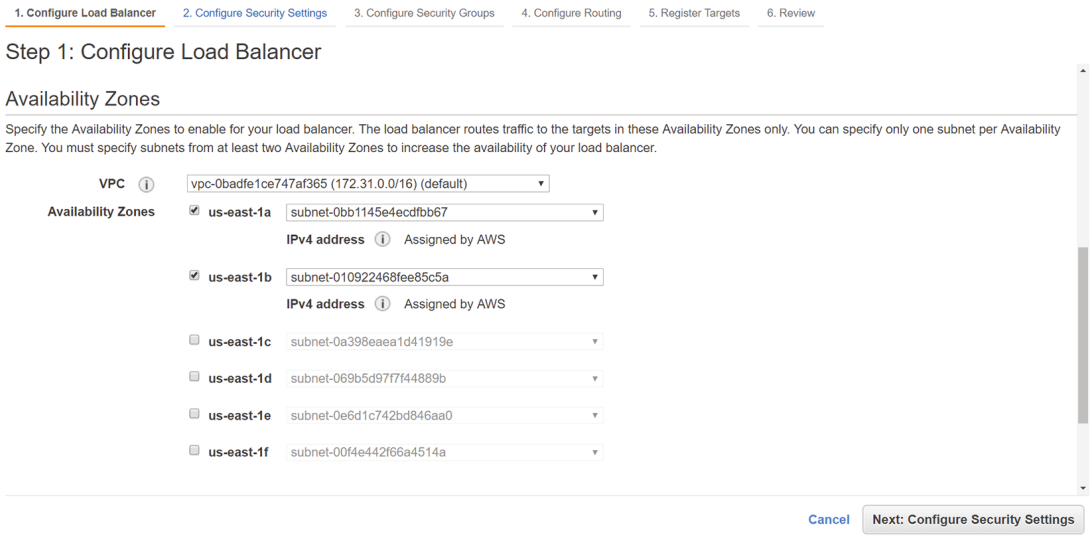


FIGURE 12.4 Application load balancer Availability Zones configuration



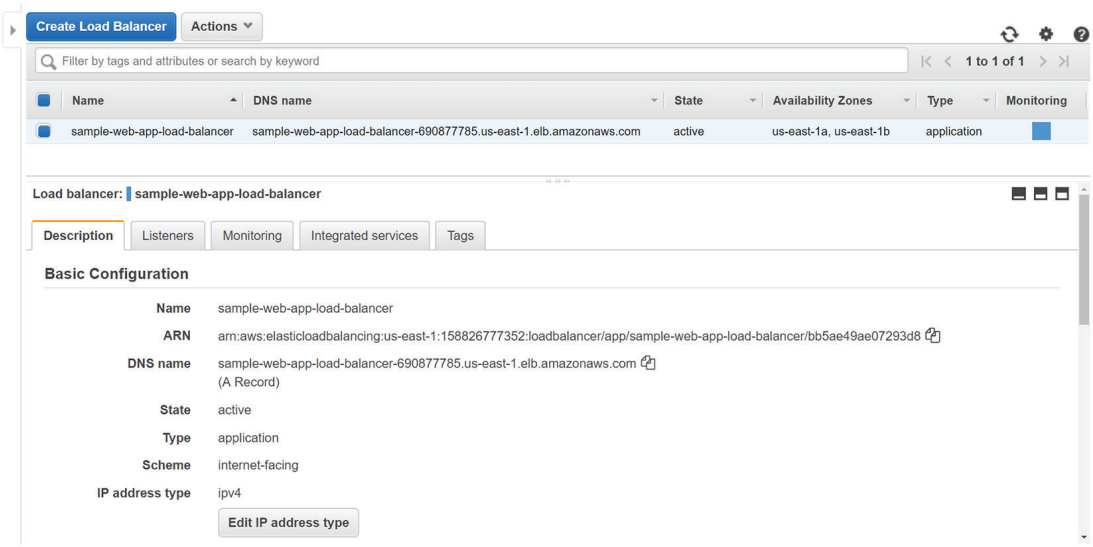
13. Select the button titled Next: Configure Routing.
14. Under Target Group, in the Target Group drop-down list, select New Target Group.
15. In the Name field, type **sample-web-app-target-group**.
16. Next to Target Type, select the Instance radio button.

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*.

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.

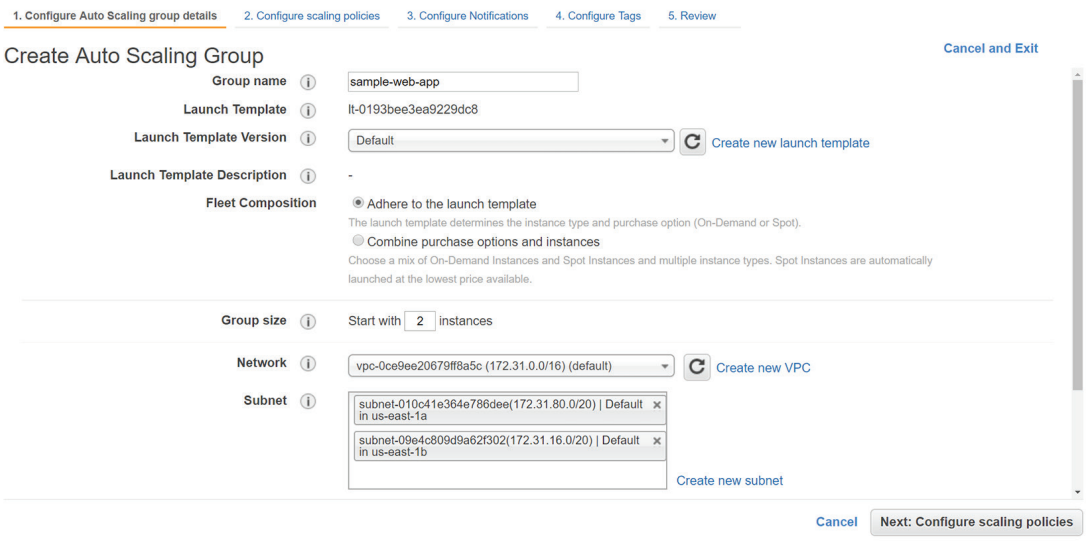
EXERCISE 12.4

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



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

When you're done experimenting, remember to delete the resources you created, including the Auto Scaling group, the instances it created, and the application load balancer.

Static Website Hosting Using S3

The next scenario you'll consider is a static website hosted on S3. Despite the term, a static website doesn't mean one that never changes. *Static* refers to the fact that the site's assets—HTML files, graphics, and other downloadable content such as PDF files—are just static files sitting in an S3 bucket. You can update these files at any time and as often as you want, but what's delivered to the end user is the same content that's stored in S3. This is in contrast to a dynamic website that uses server-side processing to modify the content on the fly just before sending it to the user. Some examples of dynamic websites are web-based email applications, database-backed ecommerce sites (like Amazon), and WordPress blogs. As a rule of thumb, if a website uses a database for storing any information, it's a dynamic website.

Setting up S3 to host a static website involves the following steps:

- Creating an S3 bucket
- Configuring it for static website hosting
- Uploading the web assets that contain the content you want to serve

By default, files in S3 buckets are not public and are accessible only by the account owner and any users the account owner has authorized. This has always been the case, but after several high-profile incidents of users unwittingly making files in their S3 buckets publicly available, AWS has taken steps to reduce this risk. This poses a challenge when you want everyone on the internet to be able to access your static website. Therefore, many of the steps you'll complete in Exercise 12.5 are there to overcome some of these important safeguards and ensure that your static website is in fact available to everyone.

EXERCISE 12.5

Create a Static Website Hosted Using S3

In this exercise, you're going to create a simple static website hosted by S3.

1. Browse to the S3 service console.
2. Choose the Create Bucket button.
3. Give the bucket a name of your choice. Try for something that's easy to type but unique (for this example, we used **benpiper2020**). Then choose the Next button.
4. Leave object versioning and encryption disabled by default. Enabling either won't impact your ability to use the bucket for hosting a static website. Choose the Next button.