



Joe Baron, Hisham Baz, Tim Bixler, Biff Gaut,
Kevin E. Kelly, Sean Senior, John Stamper

AWS Certified Solutions Architect

OFFICIAL STUDY GUIDE

ASSOCIATE EXAM

Covers exam objectives, including designing highly available, cost efficient, fault tolerant, scalable systems, implementation and deployment, data security, troubleshooting, and much more...

Includes interactive online learning environment and study tools with:

- + 2 custom practice exams
- + More than 100 electronic flashcards
- + Searchable key term glossary



columnar storage technology that improves I/O efficiency and parallelizing queries across multiple nodes, Amazon Redshift is able to deliver fast query performance. The Amazon Redshift architecture allows organizations to automate most of the common administrative tasks associated with provisioning, configuring, and monitoring a cloud data warehouse.

Amazon ElastiCache

Amazon ElastiCache is a web service that simplifies deployment, operation, and scaling of an in-memory cache in the cloud. The service improves the performance of web applications by allowing organizations to retrieve information from fast, managed, in-memory caches, instead of relying entirely on slower, disk-based databases. As of this writing, Amazon ElastiCache supports Memcached and Redis cache engines.

Management Tools

AWS provides a variety of tools that help organizations manage your AWS resources. This section provides an overview of the management tools that AWS provides to organizations.

Amazon CloudWatch

Amazon CloudWatch is a monitoring service for AWS Cloud resources and the applications running on AWS. It allows organizations to collect and track metrics, collect and monitor log files, and set alarms. By leveraging Amazon CloudWatch, organizations can gain system-wide visibility into resource utilization, application performance, and operational health. By using these insights, organizations can react, as necessary, to keep applications running smoothly.

AWS CloudFormation

AWS CloudFormation gives developers and systems administrators an effective way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion. AWS CloudFormation defines a JSON-based templating language that can be used to describe all the AWS resources that are necessary for a workload. Templates can be submitted to AWS CloudFormation and the service will take care of provisioning and configuring those resources in appropriate order (see [Figure 1.4](#)).

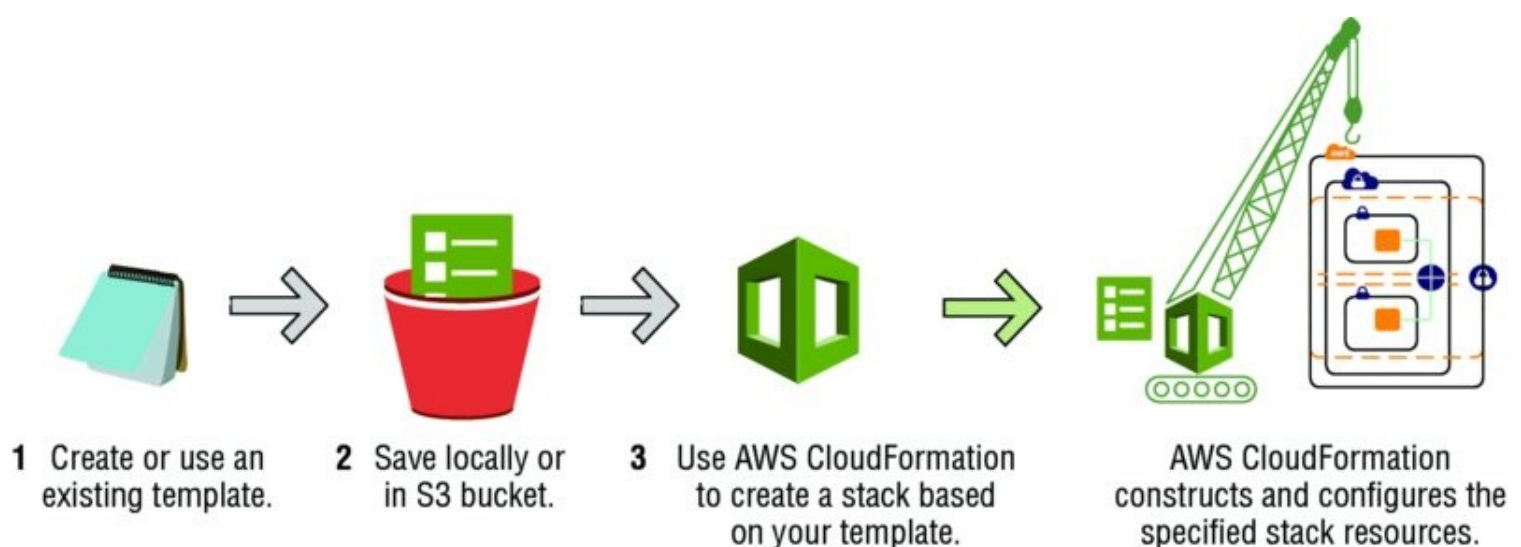


FIGURE 1.4 AWS CloudFormation workflow summary

Chapter 5

Elastic Load Balancing, Amazon CloudWatch, and Auto Scaling

THE AWS CERTIFIED SOLUTIONS ARCHITECT EXAM TOPICS COVERED IN THIS CHAPTER MAY INCLUDE, BUT ARE NOT LIMITED TO, THE FOLLOWING:

Domain 1.0: Designing highly available, cost-effective, fault-tolerant, scalable systems

- ✓ **1.1 Identify and recognize cloud architecture considerations, such as fundamental components and effective designs.**
 - Elasticity and scalability

Domain 2.0: Implementation/Deployment

- ✓ **2.1 Identify the appropriate techniques and methods using Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), AWS Elastic Beanstalk, AWS CloudFormation, AWS OpsWorks, Amazon Virtual Private Cloud (Amazon VPC), and AWS Identity and Access Management (IAM) to code and implement a cloud solution.**

Content may include the following:

- Launch instances across the AWS global infrastructure

Domain 3.0: Data Security

- ✓ **3.1 Recognize and implement secure practices for optimum cloud deployment and maintenance.**
 - CloudWatch Logs

Domain 4.0: Troubleshooting

Content may include the following:

- General troubleshooting information and questions



Introduction

In this chapter, you will learn how Elastic Load Balancing, Amazon CloudWatch, and Auto Scaling work both independently and together to help you efficiently and cost-effectively deploy highly available and optimized workloads on AWS.

Elastic Load Balancing is a highly available service that distributes traffic across Amazon Elastic Compute Cloud (Amazon EC2) instances and includes options that provide flexibility and control of incoming requests to Amazon EC2 instances.

Amazon CloudWatch is a service that monitors AWS Cloud resources and applications running on AWS. It collects and tracks metrics, collects and monitors log files, and sets alarms. Amazon CloudWatch has a basic level of monitoring for no cost and a more detailed level of monitoring for an additional cost.

Auto Scaling is a service that allows you to maintain the availability of your applications by scaling Amazon EC2 capacity up or down in accordance with conditions you set.

This chapter covers all three services separately, but it also highlights how they can work together to build more robust and highly available architectures on AWS.

Amazon CloudWatch

Amazon CloudWatch is a service that you can use to monitor your AWS resources and your applications in real time. With Amazon CloudWatch, you can collect and track metrics, create alarms that send notifications, and make changes to the resources being monitored based on rules you define.

For example, you might choose to monitor CPU utilization to decide when to add or remove Amazon EC2 instances in an application tier. Or, if a particular application-specific metric that is not visible to AWS is the best indicator for assessing your scaling needs, you can perform a `PUT` request to push that metric into Amazon CloudWatch. You can then use this custom metric to manage capacity.

You can specify parameters for a metric over a time period and configure alarms and automated actions when a threshold is reached. Amazon CloudWatch supports multiple types of actions such as sending a notification to an Amazon Simple Notification Service (Amazon SNS) topic or executing an Auto Scaling policy.

Amazon CloudWatch offers either basic or detailed monitoring for supported AWS products. *Basic monitoring* sends data points to Amazon CloudWatch every five minutes for a limited number of preselected metrics at no charge. *Detailed monitoring* sends data points to Amazon CloudWatch every minute and allows data aggregation for an additional charge. If you want to use detailed monitoring, you must enable it—basic is the default.

Amazon CloudWatch supports monitoring and specific metrics for most AWS Cloud services, including: Auto Scaling, Amazon CloudFront, Amazon CloudSearch, Amazon DynamoDB, Amazon EC2, Amazon EC2 Container Service (Amazon ECS), Amazon ElastiCache, Amazon Elastic Block Store (Amazon EBS), Elastic Load Balancing, Amazon Elastic MapReduce (Amazon EMR), Amazon Elasticsearch Service, Amazon Kinesis Streams, Amazon Kinesis Firehose, AWS Lambda, Amazon Machine Learning, AWS OpsWorks, Amazon Redshift, Amazon Relational Database Service (Amazon RDS), Amazon Route 53, Amazon SNS, Amazon Simple Queue Service (Amazon SQS), Amazon S3, AWS Simple Workflow Service (Amazon SWF), AWS Storage Gateway, AWS WAF, and Amazon WorkSpaces.

Read Alert

You may have an application that leverages Amazon DynamoDB, and you want to know when read requests reach a certain threshold and alert yourself with an email. You can do this by using `ProvisionedReadCapacityUnits` for the Amazon DynamoDB table for which you want to set an alarm. You simply set a threshold value during a number of consecutive periods and then specify email as the notification type. Now, when the threshold is sustained over the number of periods, your specified email will alert you to the read activity.

Amazon CloudWatch metrics can be retrieved by performing a `GET` request. When you use detailed monitoring, you can also aggregate metrics across a length of time you specify. Amazon CloudWatch does not aggregate data across regions but can aggregate across

Availability Zones within a region.

AWS provides a rich set of metrics included with each service, but you can also define custom metrics to monitor resources and events AWS does not have visibility into—for example, Amazon EC2 instance memory consumption and disk metrics that are visible to the operating system of the Amazon EC2 instance but not visible to AWS or application-specific thresholds running on instances that are not known to AWS. Amazon CloudWatch supports an Application Programming Interface (API) that allows programs and scripts to PUT metrics into Amazon CloudWatch as name-value pairs that can then be used to create events and trigger alarms in the same manner as the default Amazon CloudWatch metrics.

Amazon CloudWatch Logs can be used to monitor, store, and access log files from Amazon EC2 instances, AWS CloudTrail, and other sources. You can then retrieve the log data and monitor in real time for events—for example, you can track the number of errors in your application logs and send a notification if an error rate exceeds a threshold. Amazon CloudWatch Logs can also be used to store your logs in Amazon S3 or Amazon Glacier. Logs can be retained indefinitely or according to an aging policy that will delete older logs as no longer needed.

A *CloudWatch Logs agent* is available that provides an automated way to send log data to CloudWatch Logs for Amazon EC2 instances running Amazon Linux or Ubuntu. You can use the Amazon CloudWatch Logs agent installer on an existing Amazon EC2 instance to install and configure the CloudWatch Logs agent. After installation is complete, the agent confirms that it has started and it stays running until you disable it.

Amazon CloudWatch has some limits that you should keep in mind when using the service. Each AWS account is limited to 5,000 alarms per AWS account, and metrics data is retained for two weeks by default (at the time of this writing). If you want to keep the data longer, you will need to move the logs to a persistent store like Amazon S3 or Amazon Glacier. You should familiarize yourself with the limits for Amazon CloudWatch in the Amazon CloudWatch Developer Guide.

Auto Scaling

A distinct advantage of deploying applications to the cloud is the ability to launch and then release servers in response to variable workloads. Provisioning servers on demand and then releasing them when they are no longer needed can provide significant cost savings for workloads that are not steady state. Examples include a website for a specific sporting event, an end-of-month data-input system, a retail shopping site supporting flash sales, a music artist website during the release of new songs, a company website announcing successful earnings, or a nightly processing run to calculate daily activity.

Auto Scaling is a service that allows you to scale your Amazon EC2 capacity automatically by scaling out and scaling in according to criteria that you define. With Auto Scaling, you can ensure that the number of running Amazon EC2 instances increases during demand spikes or peak demand periods to maintain application performance and decreases automatically during demand lulls or troughs to minimize costs.

Embrace the Spike

Many web applications have unplanned load increases based on events outside of your control. For example, your company may get mentioned on a popular blog or television program driving many more people to visit your site than expected. Setting up Auto Scaling in advance will allow you to embrace and survive this kind of fast increase in the number of requests. Auto Scaling will scale up your site to meet the increased demand and then scale down when the event subsides.

Auto Scaling Plans

Auto Scaling has several schemes or plans that you can use to control how you want Auto Scaling to perform.

Maintain Current Instance Levels

You can configure your Auto Scaling group to maintain a minimum or specified number of running instances at all times. To maintain the current instance levels, Auto Scaling performs a periodic health check on running instances within an *Auto Scaling group*. When Auto Scaling finds an unhealthy instance, it terminates that instance and launches a new one.



Steady state workloads that need a consistent number of Amazon EC2 instances at all times can use Auto Scaling to monitor and keep that specific number of Amazon EC2 instances running.

Manual Scaling

Manual scaling is the most basic way to scale your resources. You only need to specify the change in the maximum, minimum, or desired capacity of your Auto Scaling group. Auto

Scaling manages the process of creating or terminating instances to maintain the updated capacity.



Manual scaling out can be very useful to increase resources for an infrequent event, such as the release of a new game version that will be available for download and require a user registration. For extremely large-scale events, even the Elastic Load Balancing load balancers can be pre-warmed by working with your local solutions architect or AWS Support.

Scheduled Scaling

Sometimes you know exactly when you will need to increase or decrease the number of instances in your group, simply because that need arises on a predictable schedule. Examples include periodic events such as end-of-month, end-of-quarter, or end-of-year processing, and also other predictable, recurring events. Scheduled scaling means that scaling actions are performed automatically as a function of time and date.



Recurring events such as end-of-month, quarter, or year processing, or scheduled and recurring automated load and performance testing, can be anticipated and Auto Scaling can be ramped up appropriately at the time of the scheduled event.

Dynamic Scaling

Dynamic scaling lets you define parameters that control the Auto Scaling process in a scaling policy. For example, you might create a policy that adds more Amazon EC2 instances to the web tier when the network bandwidth, measured by Amazon CloudWatch, reaches a certain threshold.

Auto Scaling Components

Auto Scaling has several components that need to be configured to work properly: a *launch configuration*, an *Auto Scaling group*, and an optional *scaling policy*.

Launch Configuration

A *launch configuration* is the template that Auto Scaling uses to create new instances, and it is composed of the configuration name, *Amazon Machine Image (AMI)*, Amazon EC2 instance type, security group, and instance key pair. Each Auto Scaling group can have only one launch configuration at a time.

The CLI command that follows will create a launch configuration with the following attributes:

Name: myLC

AMI: ami-0535d66c

Instance type: m3.medium

Security groups: sg-f57cde9d

Instance key pair: myKeyPair

```
> aws autoscaling create-launch-configuration --launch-configuration-name myLC --  
image-id ami-0535d66c --instance-type m3.medium --security-groups sg-f57cde9d --  
key-name myKeyPair
```

Security groups for instances launched in EC2-Classic may be referenced by security group name such as “SSH” or “Web” if that is what they are named, or you can reference the security group IDs, such as sg-f57cde9d. If you launched the instances in Amazon VPC, which is recommended, you must use the security group IDs to reference the security groups you want associated with the instances in an Auto Scaling launch configuration.

The default limit for launch configurations is 100 per region. If you exceed this limit, the call to create-launch-configuration will fail. You may view and update this limit by running describe-account-limits at the command line, as shown here.

```
> aws autoscaling describe-account-limits
```

Auto Scaling may cause you to reach limits of other services, such as the default number of Amazon EC2 instances you can currently launch within a region, which is 20. When building more complex architectures with AWS, it is important to keep in mind the service limits for all AWS Cloud services you are using.



When you run a command using the CLI and it fails, check your syntax first. If that checks out, verify the limits for the command you are attempting, and check to see that you have not exceeded a limit. Some limits can be raised and usually defaulted to a reasonable value to limit a race condition, an errant script running in a loop, or other similar automation that might cause unintended high usage and billing of AWS resources. AWS service limits can be viewed in the AWS General Reference Guide under AWS Service Limits. You can raise your limits by creating a support case at the AWS Support Center online and then choosing Service Limit Increase under Regarding. Then fill in the appropriate service and limit to increase value in the online form.

Auto Scaling Group

An Auto Scaling group is a collection of Amazon EC2 instances managed by the Auto Scaling service. Each Auto Scaling group contains configuration options that control when Auto Scaling should launch new instances and terminate existing instances. An Auto Scaling group must contain a name and a minimum and maximum number of instances that can be in the group. You can optionally specify desired capacity, which is the number of instances that the group must have at all times. If you don't specify a desired capacity, the default desired capacity is the minimum number of instances that you specify.

The CLI command that follows will create an Auto Scaling group that references the previous launch configuration and includes the following specifications:

Name: myASG

Launch configuration: myLC

Availability Zones: us-east-1a and us-east-1c

Minimum size: 1

Desired capacity: 3

Maximum capacity: 10

Load balancers: myELB

```
> aws autoscaling create-auto-scaling-group --auto-scaling-group-name myASG --  
launch-configuration-name myLC --availability-zones us-east-1a, us-east-1c --min-  
size 1 --max-size 10 --desired-capacity 3 --load-balancer-names myELB
```

[Figure 5.1](#) depicts deployed AWS resources after a load balancer named myELB is created and the launch configuration myLC and Auto Scaling Group myASG are set up.

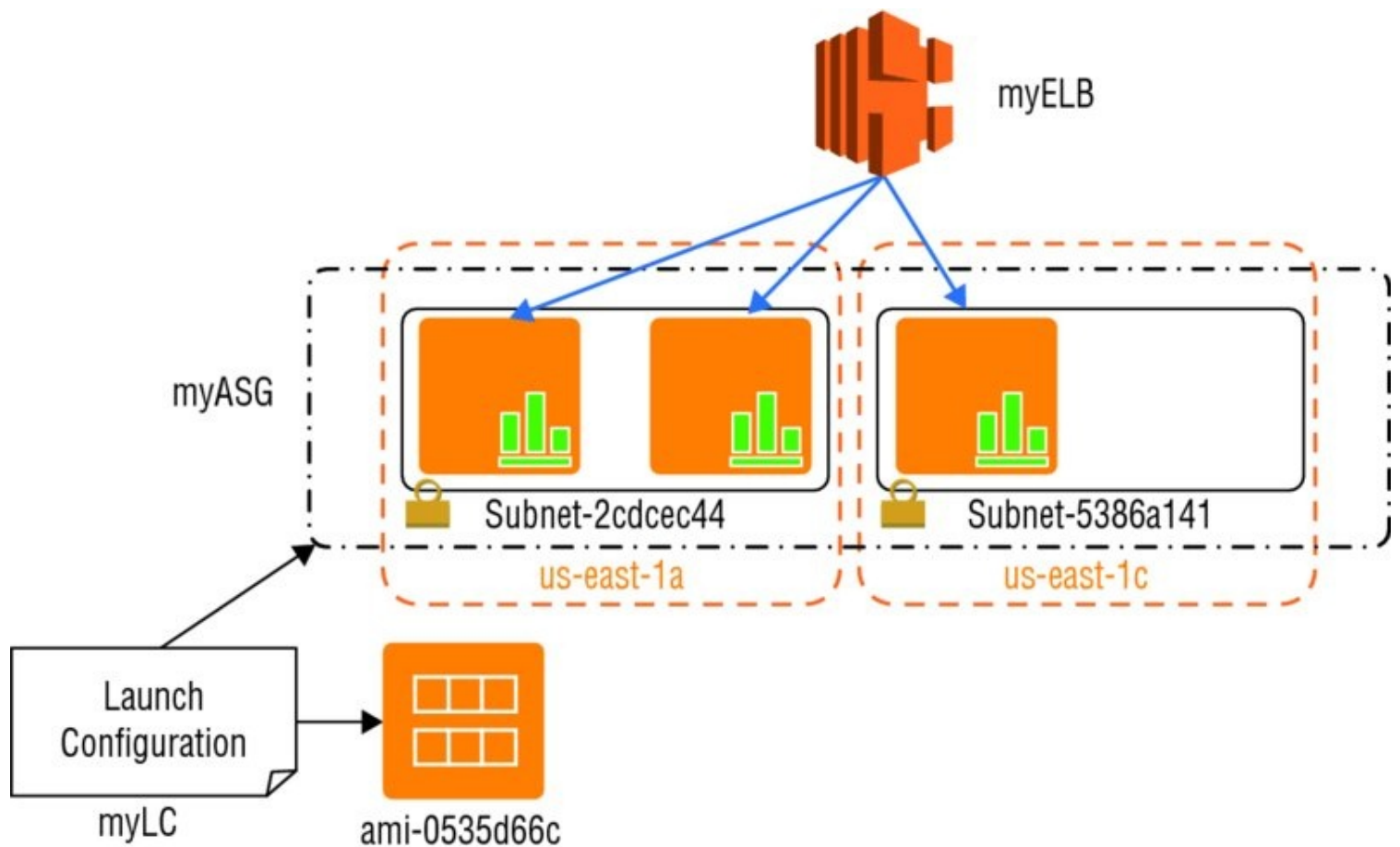


FIGURE 5.1 Auto Scaling group behind an Elastic Load Balancing load balancer

An Auto Scaling group can use either On-Demand or Spot Instances as the Amazon EC2 instances it manages. On-Demand is the default, but Spot Instances can be used by referencing a maximum bid price in the launch configuration (`--spot-price "0.15"`) associated with the Auto Scaling group. You may change the bid price by creating a new launch configuration with the new bid price and then associating it with your Auto Scaling group. If instances are available at or below your bid price, they will be launched in your Auto Scaling group. Spot Instances in an Auto Scaling group follow the same guidelines as Spot

Instances outside an Auto Scaling group and require applications that are flexible and can tolerate Amazon EC2 instances that are terminated with short notice, for example, when the Spot price rises above the bid price you set in the launch configuration. A launch configuration can reference On-Demand Instances or Spot Instances, but not both.

Spot On!

Auto Scaling supports using cost-effective Spot Instances. This can be very useful when you are hosting sites where you want to provide additional compute capacity but are price constrained. An example is a “freemium” site model where you may offer some basic functionality to users for free and additional functionality for premium users who pay for use. Spot Instances can be used for providing the basic functionality when available by referencing a maximum bid price in the launch configuration (`-spot-price "0.15"`) associated with the Auto Scaling group.

Scaling Policy

You can associate Amazon CloudWatch alarms and *scaling policies* with an Auto Scaling group to adjust Auto Scaling dynamically. When a threshold is crossed, Amazon CloudWatch sends alarms to trigger changes (scaling in or out) to the number of Amazon EC2 instances currently receiving traffic behind a load balancer. After the Amazon CloudWatch alarm sends a message to the Auto Scaling group, Auto Scaling executes the associated policy to scale your group. The policy is a set of instructions that tells Auto Scaling whether to scale out, launching new Amazon EC2 instances referenced in the associated launch configuration, or to scale in and terminate instances.

There are several ways to configure a scaling policy: You can increase or decrease by a specific number of instances, such as adding two instances; you can target a specific number of instances, such as a maximum of five total Amazon EC2 instances; or you can adjust based on a percentage. You can also scale by steps and increase or decrease the current capacity of the group based on a set of scaling adjustments that vary based on the size of the alarm threshold trigger.

You can associate more than one scaling policy with an Auto Scaling group. For example, you can create a policy using the trigger for CPU utilization, called *CPU Load*, and the CloudWatch metric *CPU Utilization* to specify scaling out if CPU utilization is greater than 75 percent for two minutes. You could attach another policy to the same Auto Scaling group to scale in if CPU utilization is less than 40 percent for 20 minutes.

The following CLI commands will create the scaling policy just described.

```
> aws autoscaling put-scaling-policy --auto-scaling-group-name myASG --policy-name CPULoadScaleOut --scaling-adjustment 1 --adjustment-type ChangeInCapacity --cooldown 30
> aws autoscaling put-scaling-policy --auto-scaling-group-name myASG --policy-name CPULoadScaleIn --scaling-adjustment -1 --adjustment-type ChangeInCapacity --cooldown 600
```

The following CLI commands will associate Amazon CloudWatch alarms for scaling out and scaling in with the scaling policy, as shown in [Figure 5.2](#). In this example, the Amazon CloudWatch alarms reference the scaling policy by Amazon Resource Name (ARN).

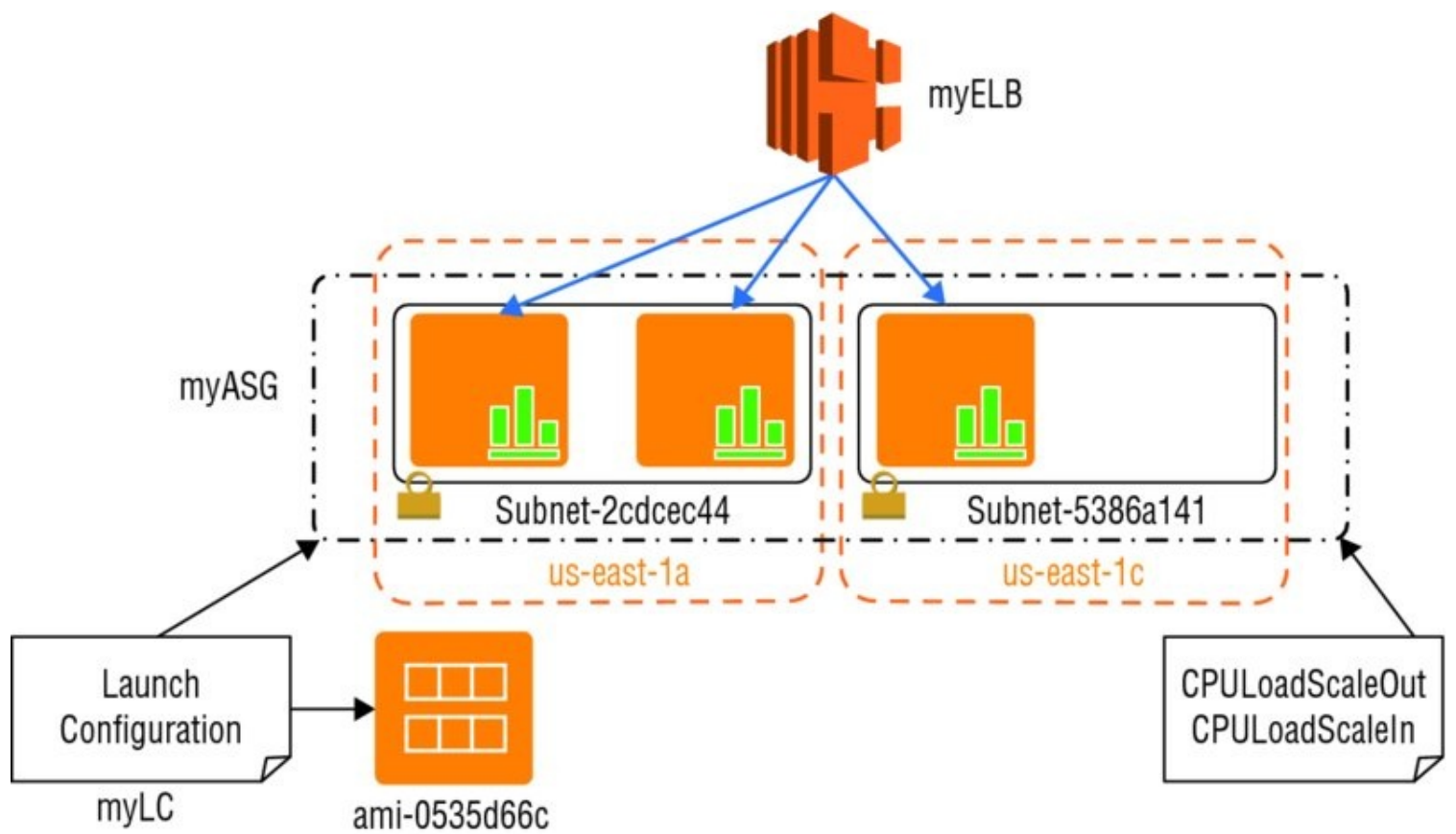


FIGURE 5.2 Auto Scaling group with policy

```
> aws cloudwatch put-metric-alarm --alarm-name capacityAdd --metric-name
CPUUtilization --namespace AWS/EC2 --statistic Average --period 300 --threshold 75
--comparison-operator GreaterThanOrEqualToThreshold --dimensions
"Name=AutoScalingGroupName, Value=myASG" --evaluation-periods 1 --alarm-actions
arn:aws:autoscaling:us-east-1:123456789012:scalingPolicy:12345678-90ab-cdef-
1234567890ab:autoScalingGroupName/myASG:policyName/CPUloadScaleOut --unit Percent
> aws cloudwatch put-metric-alarm --alarm-name capacityReduce --metric-name
CPUUtilization --namespace AWS/EC2 --statistic Average --period 1200 --threshold 40
--comparison-operator GreaterThanOrEqualToThreshold --dimensions
"Name=AutoScalingGroupName, Value=myASG" --evaluation-periods 1 --alarm-actions
arn:aws:autoscaling:us-east-1:123456789011:scalingPolicy:11345678-90ab-cdef-
1234567890ab:autoScalingGroupName/myASG:policyName/CPUloadScaleIn --unit Percent
```

If the scaling policy defined in the previous paragraph is associated with the Auto Scaling group named myASG, and the CPU utilization is over 75 percent for more than five minutes, as shown in [Figure 5.3](#), a new Amazon EC2 instance will be launched and attached to the load balancer named myELB.

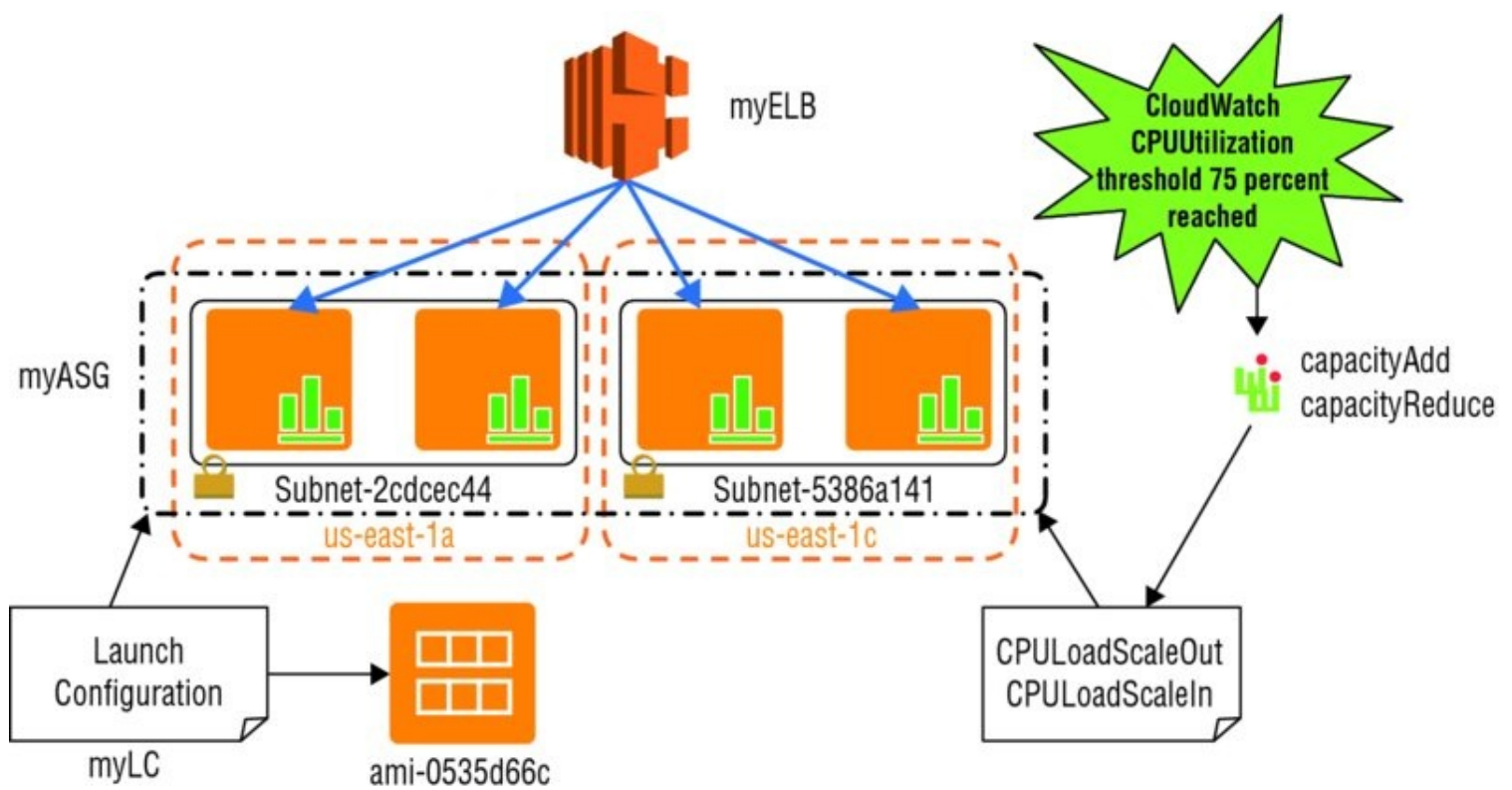


FIGURE 5.3 Amazon CloudWatch alarm triggering scaling out

A recommended best practice is to scale out quickly and scale in slowly so you can respond to bursts or spikes but avoid inadvertently terminating Amazon EC2 instances too quickly, only having to launch more Amazon EC2 instances if the burst is sustained. Auto Scaling also supports a *cooldown period*, which is a configurable setting that determines when to suspend scaling activities for a short time for an Auto Scaling group.

If you start an Amazon EC2 instance, you will be billed for one full hour of running time. Partial instance hours consumed are billed as full hours. This means that if you have a permissive scaling policy that launches, terminates, and relaunches many instances an hour, you are billing a full hour for each and every instance you launch, even if you terminate some of those instances in less than hour. A recommended best practice for cost effectiveness is to scale out quickly when needed but scale in more slowly to avoid having to relaunch new and separate Amazon EC2 instances for a spike in workload demand that fluctuates up and down within minutes but generally continues to need more resources within an hour.



Scale out quickly; scale in slowly.

It is important to consider bootstrapping for Amazon EC2 instances launched using Auto Scaling. It takes time to configure each newly launched Amazon EC2 instance before the instance is healthy and capable of accepting traffic. Instances that start and are available for load faster can join the capacity pool more quickly. Furthermore, instances that are more stateless instead of stateful will more gracefully enter and exit an Auto Scaling group.

Rolling Out a Patch at Scale

In large deployments of Amazon EC2 instances, Auto Scaling can be used to make rolling out a patch to your instances easy. The launch configuration associated with the Auto Scaling group may be modified to reference a new AMI and even a new Amazon EC2 instance if needed. Then you can deregister or terminate instances one at a time or in small groups, and the new Amazon EC2 instances will reference the new patched AMI.

Summary

This chapter introduced three services:

- Elastic Load Balancing, which is used to distribute traffic across a group of Amazon EC2 instances in one or more Availability Zones to achieve greater levels of fault tolerance for your applications.
- Amazon CloudWatch, which monitors resources and applications. Amazon CloudWatch is used to collect and track metrics, create alarms that send notifications, and make changes to resources being monitored based on rules you define.
- Auto Scaling, which allows you to automatically scale your Amazon EC2 capacity out and in using criteria that you define.

These three services can be used very effectively together to create a highly available application with a resilient architecture on AWS.

Exam Essentials

Understand what the Elastic Load Balancing service provides. Elastic Load Balancing is a highly available service that distributes traffic across Amazon EC2 instances and includes options that provide flexibility and control of incoming requests to Amazon EC2 instances.

Know the types of load balancers the Elastic Load Balancing service provides and when to use each one. An Internet-facing load balancer is, as the name implies, a load balancer that takes requests from clients over the Internet and distributes them to Amazon EC2 instances that are registered with the load balancer.

An internal load balancer is used to route traffic to your Amazon EC2 instances in VPCs with private subnets.

An HTTPS load balancer is used when you want to encrypt data between your load balancer and the clients that initiate HTTPS sessions and for connections between your load balancer and your back-end instances.

Know the types of listeners the Elastic Load Balancing service provides and the use case and requirements for using each one. A listener is a process that checks for connection requests. It is configured with a protocol and a port for front-end (client to load balancer) connections and a protocol and a port for back-end (load balancer to back-end instance) connections.

Understand the configuration options for Elastic Load Balancing. Elastic Load Balancing allows you to configure many aspects of the load balancer, including idle connection timeout, cross-zone load balancing, connection draining, proxy protocol, sticky sessions, and health checks.

Know what an Elastic Load Balancing health check is and why it is important. Elastic Load Balancing supports health checks to test the status of the Amazon EC2 instances behind an Elastic Load Balancing load balancer.

Understand what the Amazon CloudWatch service provides and what use cases there are for using it. Amazon CloudWatch is a service that you can use to monitor your AWS resources and your applications in real time. With Amazon CloudWatch, you can collect and track metrics, create alarms that send notifications, and make changes to the resources being monitored based on rules you define.

For example, you might choose to monitor CPU utilization to decide when to add or remove Amazon EC2 instances in an application tier. Or, if a particular application-specific metric that is not visible to AWS is the best indicator for assessing your scaling needs, you can perform a PUT request to push that metric into Amazon CloudWatch. You can then use this custom metric to manage capacity.

Know the differences between the two types of monitoring—basic and detailed—for Amazon CloudWatch. Amazon CloudWatch offers basic or detailed monitoring for supported AWS products. Basic monitoring sends data points to Amazon CloudWatch every five minutes for a limited number of preselected metrics at no charge. Detailed monitoring sends data points to Amazon CloudWatch every minute and allows data aggregation for an

additional charge. If you want to use detailed monitoring, you must enable it—basic is the default.

Understand Auto Scaling and why it is an important advantage of the AWS Cloud.

A distinct advantage of deploying applications to the cloud is the ability to launch and then release servers in response to variable workloads. Provisioning servers on demand and then releasing them when they are no longer needed can provide significant cost savings for workloads that are not steady state.

Know when and why to use Auto Scaling. Auto Scaling is a service that allows you to scale your Amazon EC2 capacity automatically by scaling out and scaling in according to criteria that you define. With Auto Scaling, you can ensure that the number of running Amazon EC2 instances increases during demand spikes or peak demand periods to maintain application performance and decreases automatically during demand lulls or troughs to minimize costs.

Know the supported Auto Scaling plans. Auto Scaling has several schemes or plans that you can use to control how you want Auto Scaling to perform. The Auto Scaling plans are named Maintain Current Instance Levels, Manual Scaling, Scheduled Scaling, and Dynamic Scaling.

Understand how to build an Auto Scaling launch configuration and an Auto Scaling group and what each is used for. A launch configuration is the template that Auto Scaling uses to create new instances and is composed of the configuration name, AMI, Amazon EC2 instance type, security group, and instance key pair.

Know what a scaling policy is and what use cases to use it for. A scaling policy is used by Auto Scaling with CloudWatch alarms to determine when your Auto Scaling group should scale out or scale in. Each CloudWatch alarm watches a single metric and sends messages to Auto Scaling when the metric breaches a threshold that you specify in your policy.

Understand how Elastic Load Balancing, Amazon CloudWatch, and Auto Scaling are used together to provide dynamic scaling. Elastic Load Balancing, Amazon CloudWatch, and Auto Scaling can be used together to create a highly available application with a resilient architecture on AWS.

Exercises

For assistance in completing the following exercises, refer to the Elastic Load Balancing Developer Guide located at <http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elastic-load-balancing.html>, the Amazon CloudWatch Developer Guide at <http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/WhatIsCloudWatch.html> and the Auto Scaling User Guide at <http://docs.aws.amazon.com/autoscaling/latest/userguide/WhatIsAutoScaling.html>.

EXERCISE 5.1

Create an Elastic Load Balancing Load Balancer

In this exercise, you will use the AWS Management Console to create an Elastic Load Balancing load balancer.

1. Launch an Amazon EC2 instance using an AMI with a web server on it, or install and configure a web server.
2. Create a static page to display and a health check page that returns HTTP 200. Configure the Amazon EC2 instance to accept traffic over port 80.
3. Register the Amazon EC2 instance with the Elastic Load Balancing load balancer, and configure it to use the health check page to evaluate the health of the instance.

EXERCISE 5.2

Use an Amazon CloudWatch Metric

1. Launch an Amazon EC2 instance.
2. Use an existing Amazon CloudWatch metric to monitor a value.

EXERCISE 5.3

Create a Custom Amazon CloudWatch Metric

1. Create a custom Amazon CloudWatch metric for memory consumption.
2. Use the CLI to PUT values into the metric.

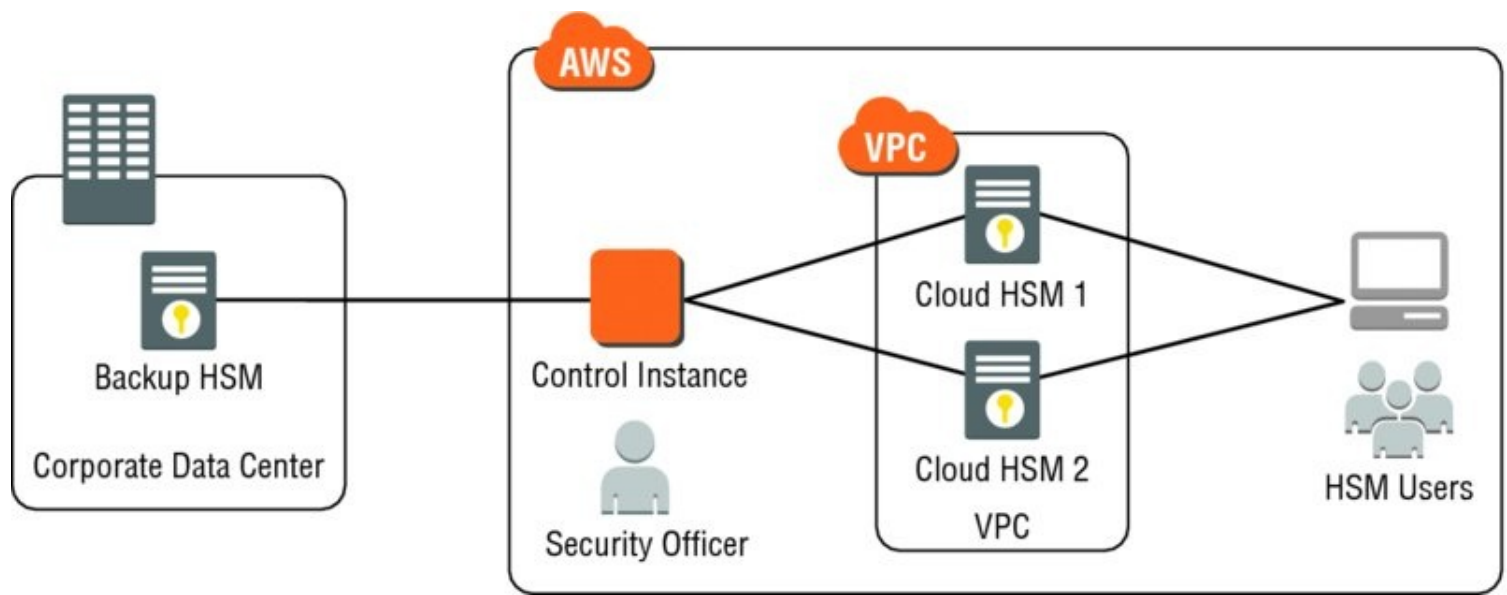


FIGURE 11.2 High availability CloudHSM architecture

AWS CloudHSM allows you to protect your encryption keys within HSMs that are designed and validated to government standards for secure key management. You can securely generate, store, and manage the cryptographic keys used for data encryption in a way that ensures that only you have access to the keys. AWS CloudHSM helps you comply with strict key management requirements within the AWS cloud without sacrificing application performance.

Use Cases

The AWS key management services address several security needs that would require extensive effort to deploy and manage otherwise, including, but not limited to:

Scalable Symmetric Key Distribution Symmetric encryption algorithms require that the same key be used for both encrypting and decrypting the data. This is problematic because transferring the key from the sender to the receiver must be done either through a known secure channel or some “out of band” process.

Government-Validated Cryptography Certain types of data (for example, Payment Card Industry—PCI—or health information records) must be protected with cryptography that has been validated by an outside party as conforming to the algorithm(s) asserted by the claiming party.

AWS CloudTrail

AWS CloudTrail provides visibility into user activity by recording API calls made on your account. AWS CloudTrail records important information about each API call, including the name of the API, the identity of the caller, the time of the API call, the request parameters, and the response elements returned by the AWS service. This information helps you to track changes made to your AWS resources and to troubleshoot operational issues. AWS CloudTrail makes it easier to ensure compliance with internal policies and regulatory standards.

Overview

AWS CloudTrail captures AWS API calls and related events made by or on behalf of an AWS account and delivers log files to an Amazon S3 bucket that you specify. Optionally, you can configure AWS CloudTrail to deliver events to a log group monitored by Amazon CloudWatch Logs. You can also choose to receive Amazon Simple Notification Service (Amazon SNS) notifications each time a log file is delivered to your bucket. You can create a *trail* with the AWS CloudTrail console, the AWS Command Line Interface (CLI), or the AWS CloudTrail API. A trail is a configuration that enables logging of the AWS API activity and related events in your account.

You can create two types of trails:

A Trail That Applies to All Regions When you create a trail that applies to all AWS regions, AWS CloudTrail creates the same trail in each region, records the log files in each region, and delivers the log files to the single Amazon S3 bucket (and optionally to the Amazon CloudWatch Logs log group) that you specify. This is the default option when you create a trail using the AWS CloudTrail console. If you choose to receive Amazon SNS notifications for log file deliveries, one Amazon SNS topic will suffice for all regions. If you choose to have AWS CloudTrail send events from a trail that applies to all regions to an Amazon CloudWatch Logs log group, events from all regions will be sent to the single log group.

A Trail That Applies to One Region You specify a bucket that receives events only from that region. The bucket can be in any region that you specify. If you create additional individual trails that apply to specific regions, you can have those trails deliver event logs to a single Amazon S3 bucket.

By default, your log files are encrypted using Amazon S3 SSE. You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically.

AWS CloudTrail typically delivers log files within 15 minutes of an API call. In addition, the service publishes new log files multiple times an hour, usually about every five minutes. These log files contain API calls from all of the account's services that support AWS CloudTrail.



Enable AWS CloudTrail on all of your AWS accounts. Instead of configuring a trail for one region, you should enable trails for all regions.

Use Cases

AWS CloudTrail is beneficial for several use cases:

External Compliance Audits Your business must demonstrate compliance to a set of regulations pertinent to some or all data being transmitted, processed, and stored within your AWS accounts. Events from AWS CloudTrail can be used to show the degree to which you are compliant with the regulations.

Unauthorized Access to Your AWS Account AWS CloudTrail records all sign-on attempts to your AWS account, including AWS Management Console login attempts, AWS

Software Development Kit (SDK) API calls, and AWS CLI API calls. Routine examination of AWS CloudTrail events will provide the needed information to determine if your AWS account is being targeted for unauthorized access.