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

Compliance If any of your resources must adhere to specific compliance requirements such as the Health Insurance Portability and Accountability Act (HIPAA) or the Payment Card Industry Data Security Standard (PCI DSS), you can tag those resources accordingly.

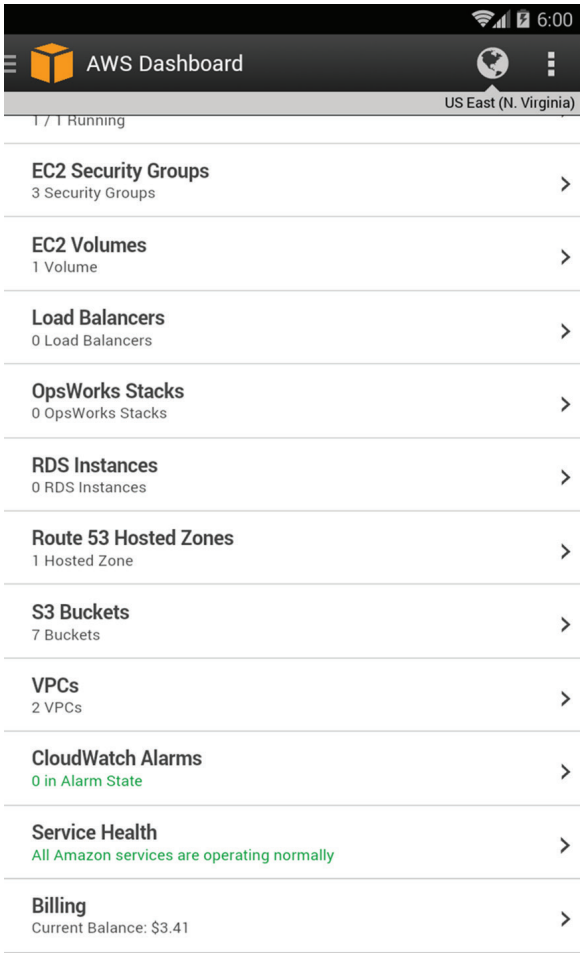
The AWS Console Mobile Application

The *AWS Console Mobile Application* is a smartphone application that lets you manage your AWS account and resources on the go. The application features a dashboard showing key information about your AWS account and resources, including the following:

- Service Health—View any current health issues with AWS services across different regions.
- CloudWatch Alarms—View alarm graphs and current alarm status.
- Billing—View your current billing balance and a graph of usage charges.

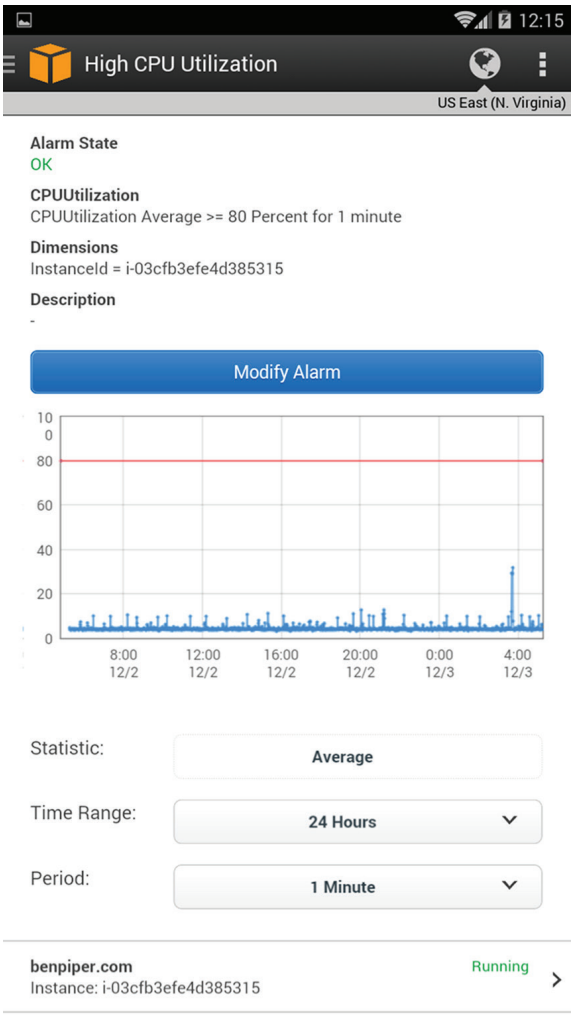
Take a look at Figure 6.11 for an example of what kind of information the dashboard can show you.

FIGURE 6.11 The AWS Console Mobile Application dashboard



You can use the application to make limited changes to some AWS resources, including CloudWatch Alarms, EC2 security groups, EC2 instances, and CloudFormation stacks. For example, you can view or modify a CloudWatch alarm, as shown in Figure 6.12.

FIGURE 6.12 Viewing a CloudWatch alarm from the AWS Console Mobile Application



Or you can stop or reboot an EC2 instance, as shown in Figure 6.13.

CloudWatch

Amazon CloudWatch is a key service that helps you plan, monitor, and fine-tune your AWS infrastructure and applications. It lets you collect, search, and visualize data from your applications and AWS resources in the form of logs, metrics, and events. Common CloudWatch use cases include the following:

Infrastructure monitoring and troubleshooting Visualize performance metrics to discover trends over time and spot outliers that might indicate a problem. Correlate metrics and logs across your application and infrastructure stacks to understand the root cause of failures and performance issues.

Resource optimization Save money and help with resource planning by identifying over-used or underused resources. Ensure performance and availability by using AWS Auto Scaling to automatically provision new EC2 instances to meet demand.

Application monitoring Create CloudWatch alarms to alert you and take corrective action when a resource's utilization, performance, or health falls outside of a threshold that you define.

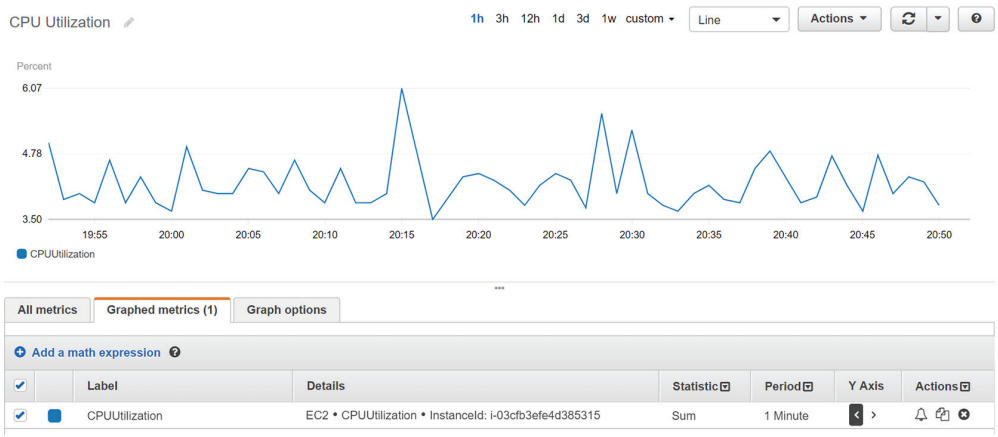
Log analytics Search, visualize, and correlate logs from multiple sources to help with troubleshooting and identify areas for improvement.

CloudWatch Metrics

CloudWatch Metrics is a feature that collects numeric performance metrics from both AWS and non-AWS resources such as on-premises servers. A *metric* is a variable that contains a time-ordered set of data points. Each data point contains a timestamp, a value, and optionally a unit of measure. For example, a data point for the CPU Utilization metric for an EC2 instance may contain a timestamp of December 25, 2018 13:37, a value of 75, and Percent as the unit of measure.

All AWS resources automatically send their metrics to CloudWatch. These metrics include things such as EC2 instance CPU utilization, S3 bucket sizes, and DynamoDB consumed read and write capacity units. CloudWatch stores metrics for up to 15 months. You can graph metrics to view trends and how they change over time, as illustrated in Figure 6.16.

FIGURE 6.16 Using CloudWatch to graph the CPU Utilization metric for an EC2 Instance



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

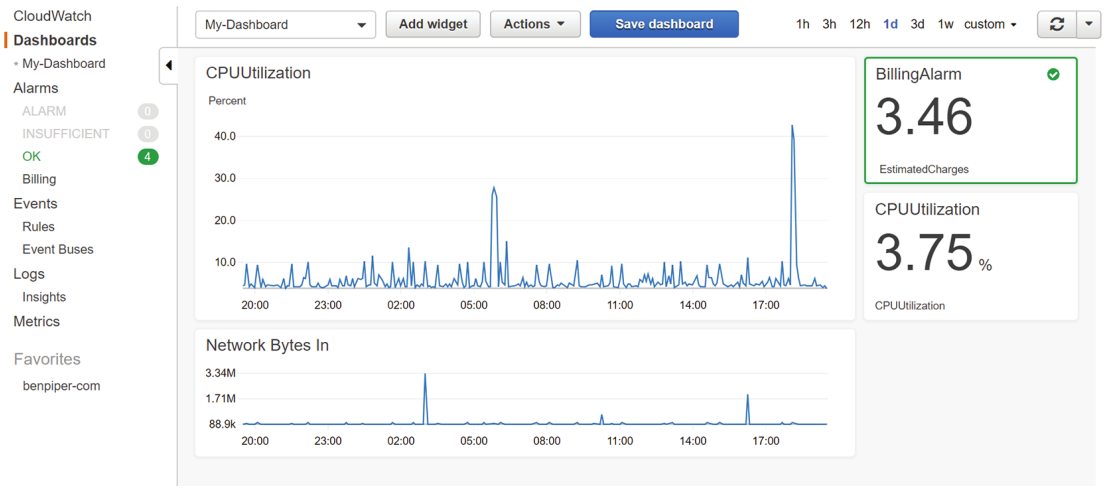
Auto Scaling action By specifying an EC2 Auto Scaling action, the EC2 Auto Scaling service can add or remove EC2 instances in response to changing demand. For example, if a metric indicates that instances are overburdened, you can have EC2 Auto Scaling respond by adding more instances.

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.

FIGURE 6.17 A CloudWatch dashboard



CloudWatch Logs

CloudWatch Logs collects and stores log files from AWS and non-AWS sources and makes it easy to view, search, and extract custom metrics from them.

Log Events, Streams, and Groups

You configure your applications and AWS services to send log events to CloudWatch Logs. A log event is analogous to a line in a log file and always contains a timestamp and an event message. Many AWS services produce their own logs called *vended logs* that you can stream to CloudWatch Logs. Such logs include Route 53 DNS query logs, VPC flow logs, and CloudTrail logs. CloudWatch Logs can also receive custom logs from your applications, such as web server access logs.

CloudWatch Logs organizes log events by log streams by storing log events from the same source in a single log stream. For example, web server access logs from a specific EC2 instance would be stored in one log stream, while Route 53 DNS query logs would be stored in a separate log stream.

CloudWatch further organizes log streams into log groups. To organize related log streams, you can place them into the same log group. For instance, if you have several log streams that are collecting web server log events from multiple web servers, you can group all of those log streams into a single log group.

CloudWatch Logs stores log events indefinitely by default, but you can configure a log group's retention settings to delete events automatically. Retention settings range from 1 day to 10 years. You can also archive your logs by exporting them to an S3 bucket.

Metric Filters

A metric filter extracts data from log events in a log group and stores that data in a custom CloudWatch metric. For example, suppose a log event from a database server contains the time in milliseconds it takes to run a query. You may extract that value and store it as a CloudWatch metric so you can graph it and create an alarm to send a notification when it exceeds a certain threshold.

You can also use metric filters to track the number of times a particular string occurs. This is useful for counting the number of times a particular event occurs in a log, such as an error code. For example, you might want to track how many times a 403 Forbidden error appears in a web server log. You can configure a metric filter to count the number of times the error occurs in a given timeframe—five minutes, for example—and record that value in a CloudWatch custom metric.

Metric filters let you extract or derive quantitative data from log events, so metric values will always be numeric. You can't store a non-numeric string such as an IP address in a metric.

CloudWatch Events

The CloudWatch Events feature lets you continuously monitor for specific events that represent a change in your AWS resources—particularly write-only API operations—and take an action when they occur. For example, an EC2 instance going from the running state to the stopped state would be an event. An IAM user logging into the AWS Management Console would also be an event. CloudWatch Events can then automatically and immediately take actions in response to those events.

You start by creating a rule to define the events to monitor, as well as the actions you want to take in response to those events. You define the action to take by selecting a target, which is an AWS resource. Some targets you can choose from include the following:

- Lambda functions
- EC2 instances
- SQS queues
- SNS topics
- ECS tasks

CloudWatch responds to events as they occur, in real time. Unlike CloudWatch alarms, which take action when a metric crosses and remains crossing a numeric threshold, CloudWatch events trigger immediately. For example, you can create a CloudWatch event to send an SNS notification whenever an EC2 instance terminates. Or you could trigger a Lambda function to process an image file as soon as it hits an S3 bucket.

Cost Explorer updates report data at least once every 24 hours.

Summary

Starting out, you'll spend most of your time interacting with AWS using the AWS Management Console. It's always changing, but even when it does, AWS takes great care to let you know what changed. Sometimes AWS will even let you preview new console features before they go live, giving you the chance to adjust to the change before it's rolled out permanently.

As you find yourself working with AWS more and getting more familiar with the services, you'll begin to use the AWS Command Line Interface for many common tasks. The AWS Command Line Interface is a must for scripting AWS tasks and collecting information from your AWS resources in bulk.

CloudWatch collects metrics from AWS services. You can create alarms to take some action, such as a notification, when a metric crosses a threshold. CloudWatch receives and stores logs from AWS and non-AWS services and even extracts metrics from those logs using metric filters.

CloudTrail records events that occur against your AWS account. By default, the CloudTrail event history log captures the last 90 days of management events in each region. If you want to log more than this or customize the events that it logs, you must create a trail to cause CloudTrail to store events in an S3 bucket. You can also configure a trail to stream logs to CloudWatch Logs for storage, viewing, and searching.

Exam Essentials

Understand when to use the AWS Management Console versus the AWS CLI. The Management Console is required if you want to use the point-and-click interface and want to view visual elements such as CloudWatch graphs or Cost Explorer graphs. You can log into the Management Console using an email address and password for the root account. If you're logging in as an IAM user, you'll need the account alias or number, IAM username, and password. If MFA is set up, you'll be prompted for an MFA one-time passcode. The AWS CLI is what you'll use to manage your AWS resources manually from the command line or using scripts. It's good for repetitive or bulk tasks that would take a long time using the Web. To use the CLI, you need an access key ID and secret key.

Know how to use resource tags and resource groups. Resource tags are keys associated with your AWS resources. A key can optionally contain a value. You can use tags to label your resources according to whatever you like, be it owner, business unit, or

environment. You can group resources into a resource group according to resource tags or CloudFormation stacks.

Be able to identify use cases for CloudWatch. CloudWatch can collect logs and metrics from AWS and non-AWS services. Many AWS services such as EC2 automatically send metric data to CloudWatch. You can create alarms to trigger when a metric falls above or below a threshold. In response to an alarm, you can send a notification using SNS, or you can take an action using an Auto Scaling action or EC2 action. You can also graph metrics to view trends visually. CloudWatch Logs lets you aggregate and search log files. Some services, such as VPC and Route 53, can be configured to stream vended logs to CloudWatch logs. You can extract metrics from these logs using metric filters. CloudWatch events let you take actions in response to specific events that occur with your AWS resources, such as launching an EC2 instance or creating an S3 bucket. Unlike alarms that are triggered by metrics crossing a threshold, CloudWatch Events acts in response to specific API operations.

Know the options for developing applications that integrate with AWS. AWS offers SDKs for a variety of programming languages and platforms. You can use the SDKs to quickly develop desktop, server, web-based, or mobile apps that use AWS services. Although many AWS services offer the HTTPS-based AWS Query API that you can interface with directly, the SDKs handle the heavy lifting of request authentication, serialization, and connection management, freeing you up to write your application without having to learn the nitty-gritty API details of every AWS service you want to use.

Understand what CloudTrail does and how it differs from and integrates with CloudWatch. CloudTrail logs management and data operations on your account. By default, it logs 90 days of management events per region. If you want to log more than this or customize which events it logs, you can create a trail to log those events and store them in an S3 bucket. You can optionally stream CloudTrail logs to CloudWatch for storage, searching, and analysis.

Geolocation A Geolocation policy lets you route users based on their specific continent, country, or state.

Multivalue Answer A Multivalue Answer policy allows you to evenly distribute traffic across multiple resources. Unlike Weighted policies that return a single record, a Multivalue Answer policy returns all records, sorted in a random order.

Health Checks

All routing policies with the exception of Simple can use health checks to determine whether they should route users to a given resource. A health check can check one of three things: an endpoint, a CloudWatch alarm, or another health check. All health checks occur every 10 seconds or 30 seconds.

Endpoint Endpoint health checks work by connecting to the endpoint you want to monitor via HTTP, HTTPS, or TCP. Route 53 has health checkers in several AWS Regions, and you can choose which health checkers a health check uses. This lets you ensure that an endpoint is reachable from various locations around the world.

CloudWatch alarm A Route 53 health check can monitor the status of a CloudWatch alarm. This is useful if you want to consider a resource unhealthy if it's experiencing high latency or is servicing a high number of connections.

Calculated This type of health check monitors the status of other health checks. For example, if you want to consider the status of both an Endpoint health check and a CloudWatch alarm health check, you can create a Calculated health check to take both into account.

Traffic Flow and Traffic Policies

If you require complex routing scenarios for a public hosted zone, creating multiple resource records with a variety of different routing policies can become an administrative nightmare. As an alternative to manually engineering routing policies, you can use the Route 53 Traffic Flow visual editor to create a diagram to represent the desired routing.

The diagram you create represents a traffic policy that you can save and associate with a domain name by creating a policy record. Route 53 doesn't create the individual resource records but instead hides the routing behind the single policy record. The cost is currently $50 USD per month per policy record.

You can use the same routing policies that are available with normal resource records: Simple, Weighted, Latency, Failover, Geolocation, and Multivalue Answer. But in addition, Traffic Flow offers another routing policy that's not otherwise available: Geoproximity. The Geoproximity routing policy lets you direct users to a resource based on how close they are to a geographic location. This differs from the Geolocation routing policy that routes based on the user's specific continent, country, or state.

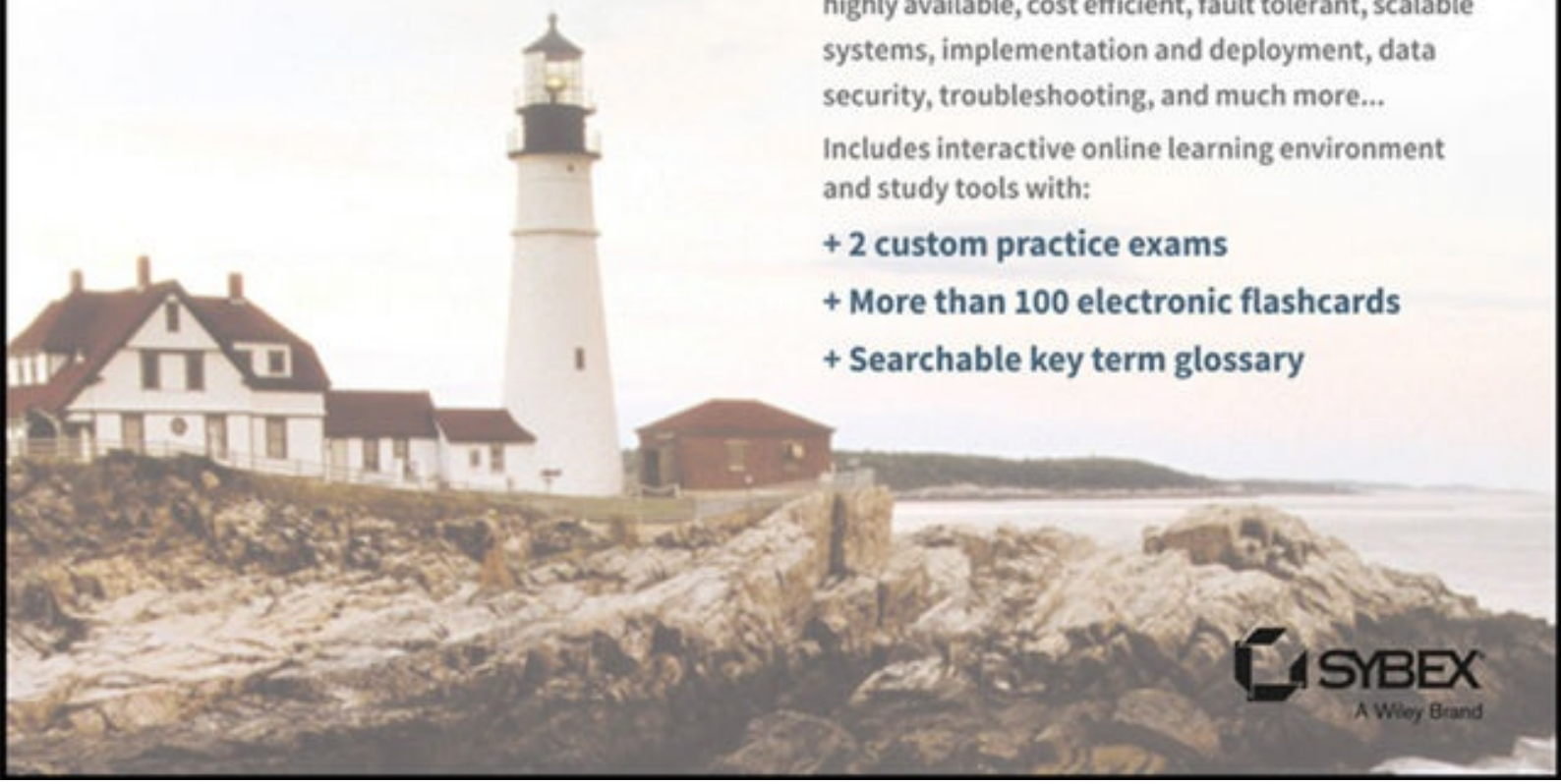
amazon web services

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

SYBEX
A Wiley Brand

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).

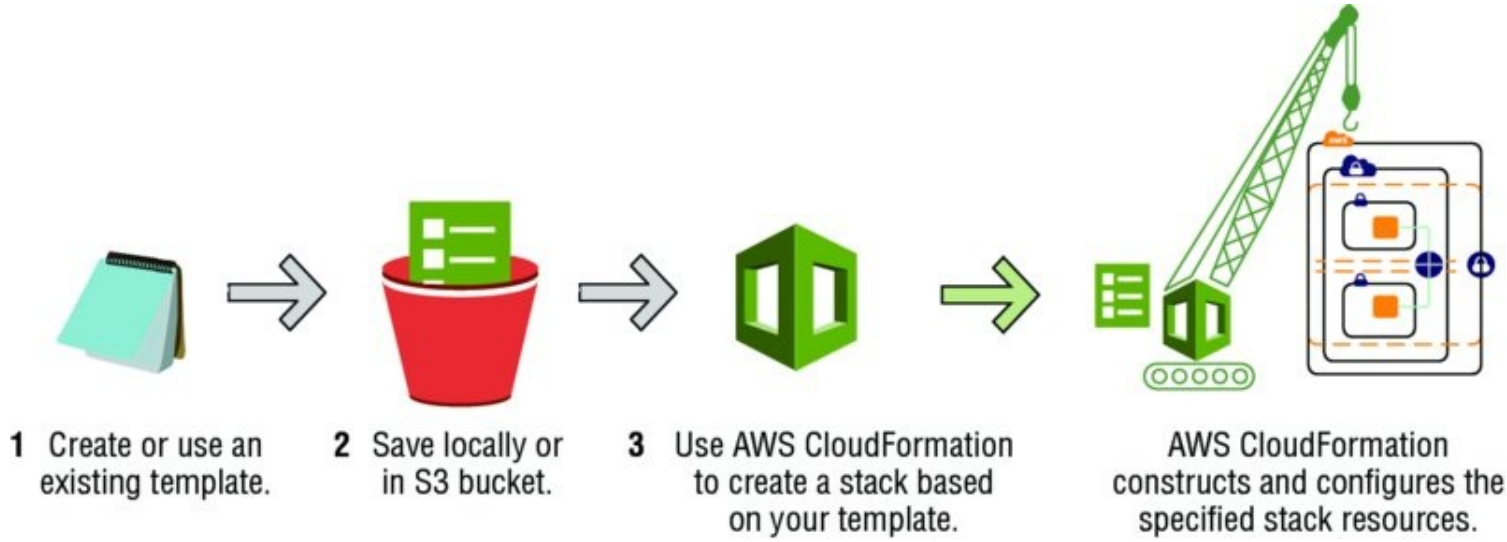


FIGURE 1.4 AWS CloudFormation workflow summary

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.

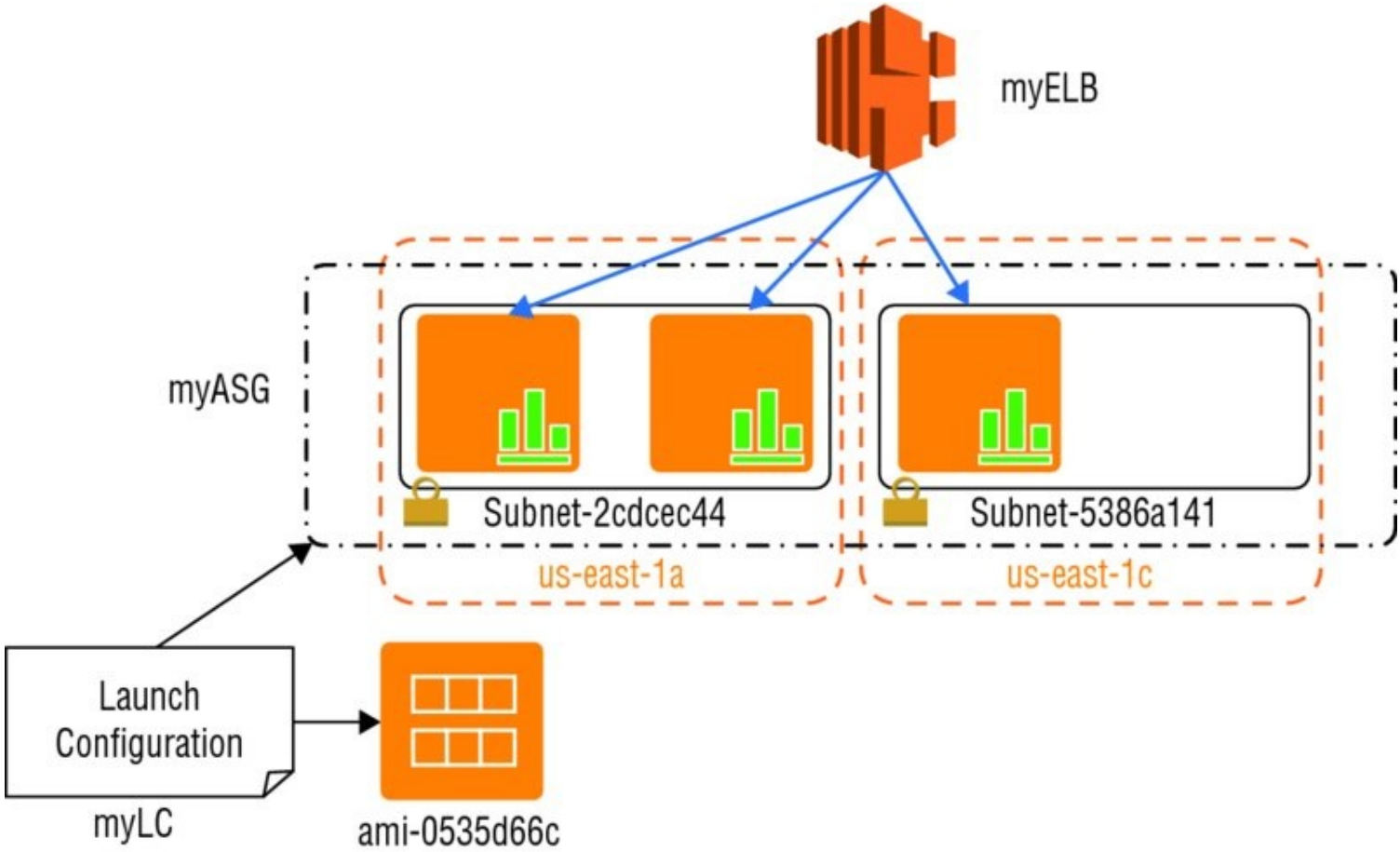


FIGURE 5.1 Auto Scaling group behind an Elastic Load Balancing load balancer

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 *CPULoad*, and the CloudWatch metric *CPUUtilization* 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).

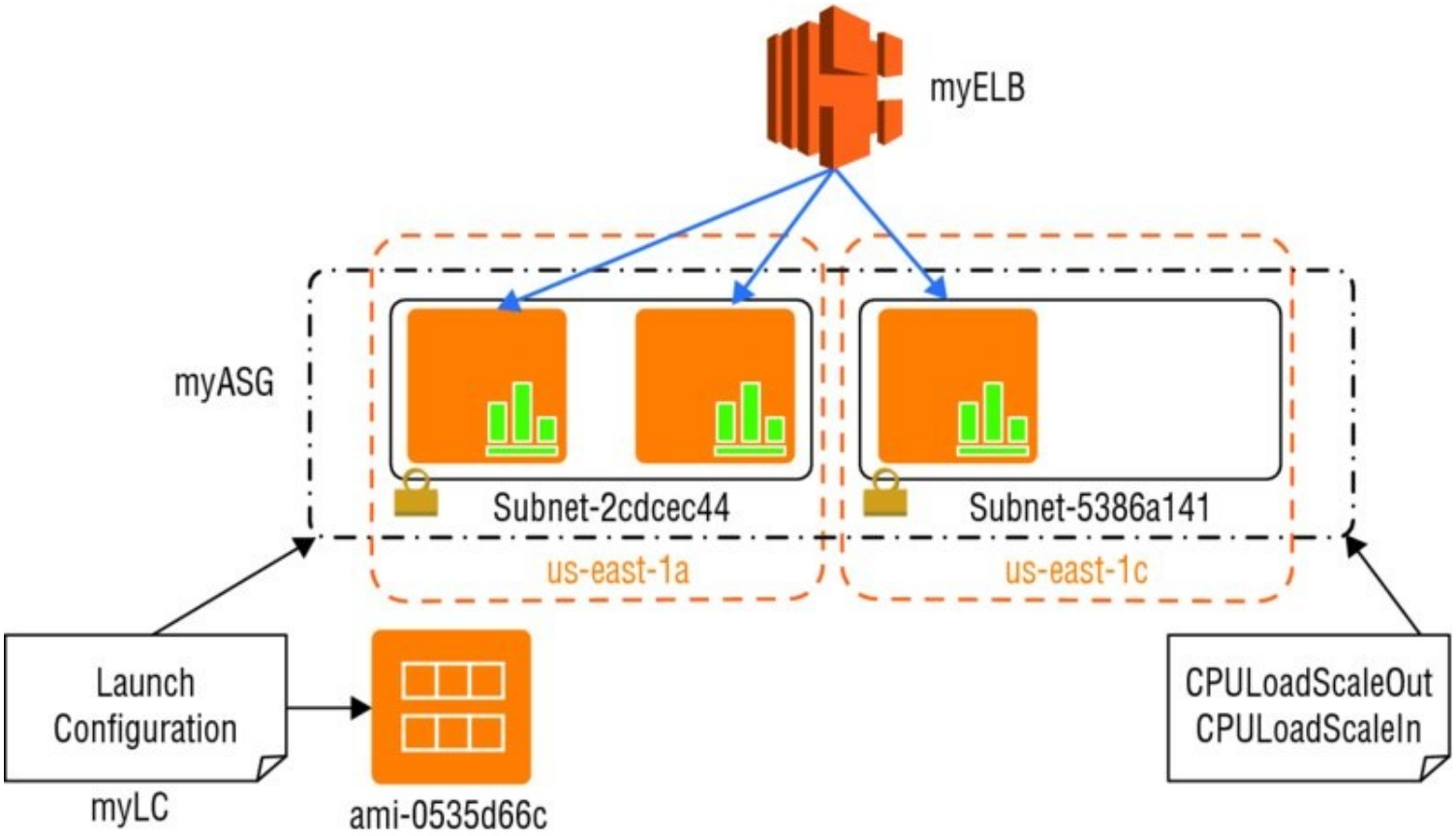


FIGURE 5.2 Auto Scaling group with policy

```
> 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`.

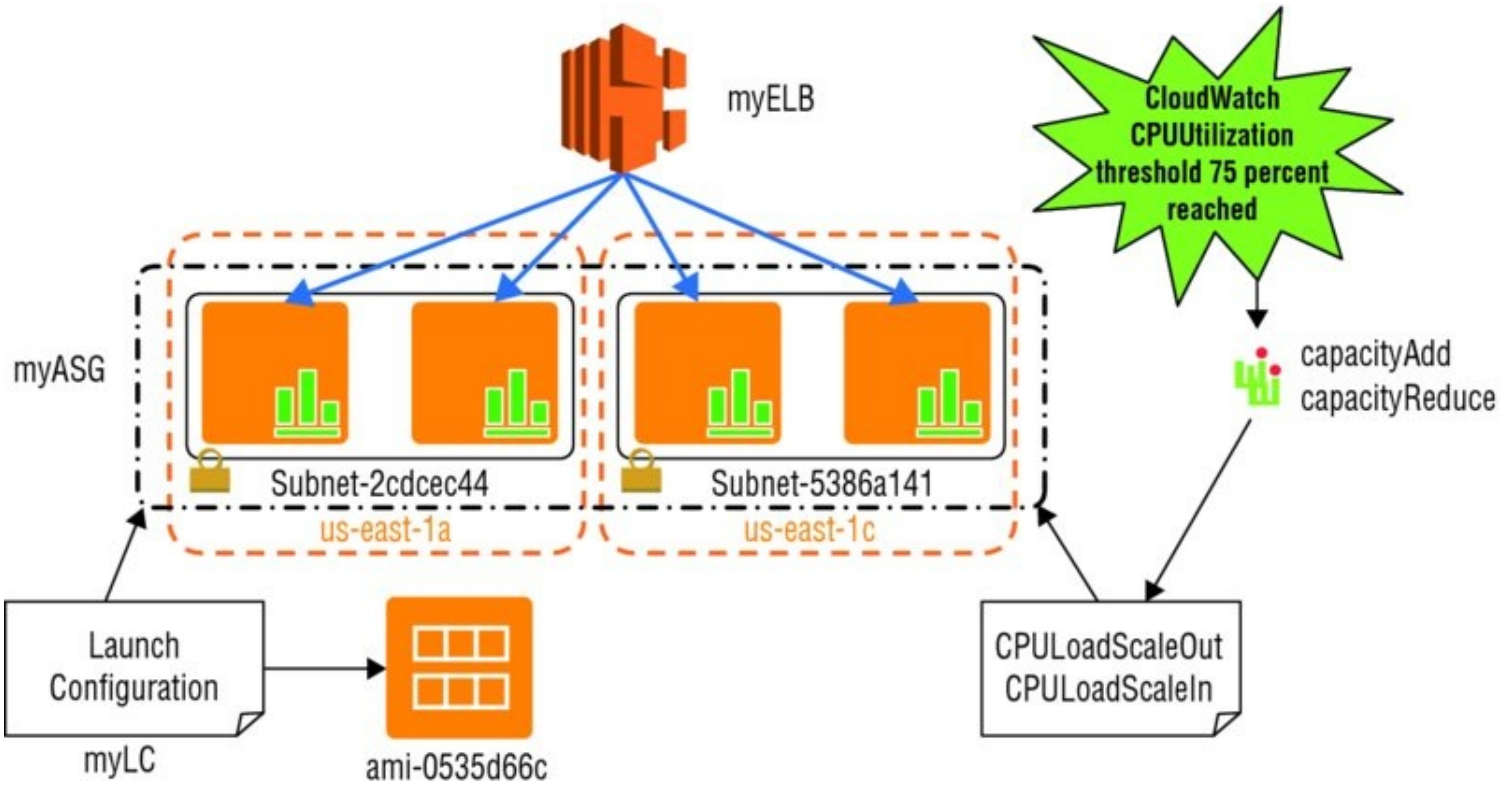


FIGURE 5.3 Amazon CloudWatch alarm triggering scaling out

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 Instant 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.h 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.

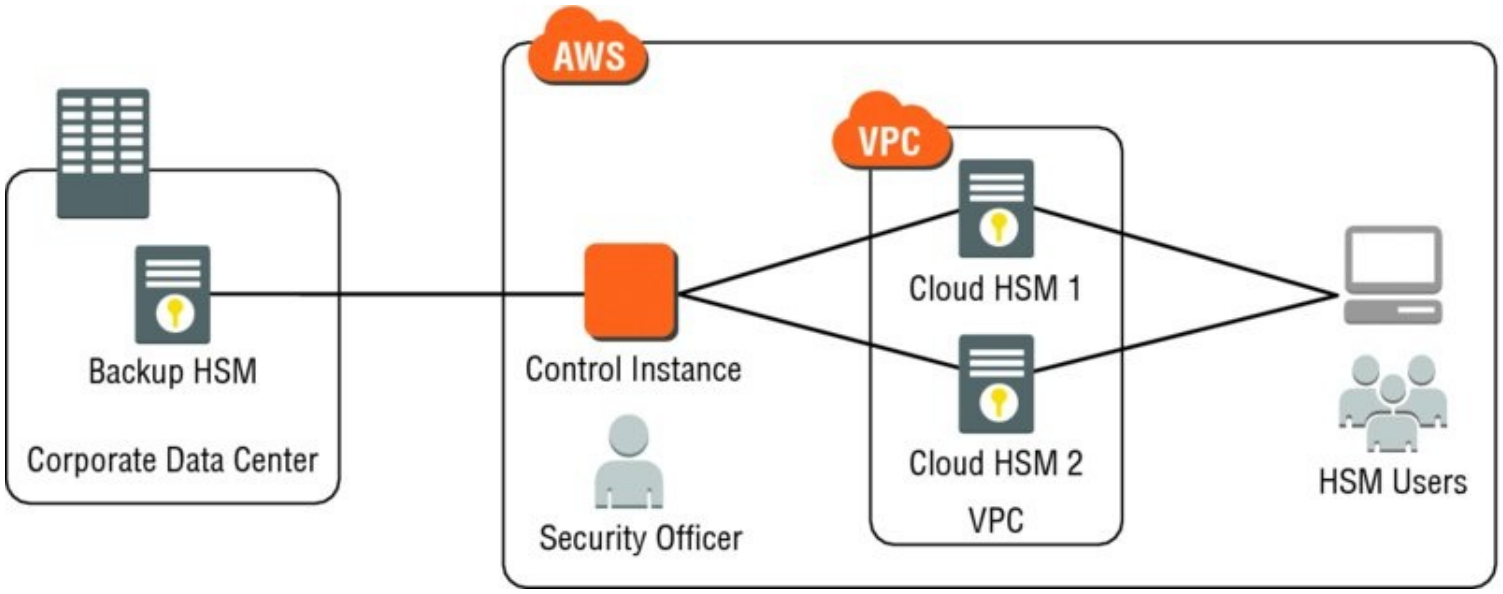


FIGURE 11.2 High availability CloudHSM architecture

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.

JON BONSO AND KENNETH SAMONTE

AWS CERTIFIED

DEVOPS ENGINEER PROFESSIONAL

Tutorials Dojo
Study Guide and Cheat Sheets

3. Amazon CloudWatch Events
4. Amazon CloudWatch Alarms
5. AWS CodePipeline
6. AWS CodeDeploy
7. AWS CodeBuild
8. AWS CodeCommit
9. AWS Config
10. AWS Systems Manager
11. Amazon ECS
12. Amazon Elastic Beanstalk
13. AWS CloudTrail
14. AWS OpsWorks
15. AWS Trusted Advisor

The FAQs provide a good summary for each service, however, the AWS documentation contains more detailed information that you'll need to study. These details will be the deciding factor in determining the correct choice from the incorrect choices in your exam. To supplement your review of the services, we recommend that you take a look at Tutorials Dojo's AWS Cheat Sheets. Their contents are well-written and straight to the point, which will help reduce the time spent going through FAQs and documentations.

Common Exam Scenarios

Scenario	Solution
Software Development and Lifecycle (SDLC) Automation	
An Elastic Beanstalk application must not have any downtime during deployment and requires an easy rollback to the previous version if an issue occurs.	Set up Blue/Green deployment, deploy a new version on a separate environment then swap environment URLs on Elastic Beanstalk.
A new version of an AWS Lambda application is ready to be deployed and the deployment should not cause any downtime. A quick rollback to the previous Lambda version must be available.	Publish a new version of the Lambda function. After testing, use the production Lambda Alias to point to this new version.
In an AWS Lambda application deployment, only 10% of the incoming traffic should be routed to the new version to verify the changes before eventually allowing all production traffic.	Set up Canary deployment for AWS Lambda. Create a Lambda Alias pointed to the new Version. Set Weighted Alias value for this Alias as 10%.
An application hosted in Amazon EC2 instances behind an Application Load Balancer. You must	Launch the application in Amazon EC2 that runs the new version with an Application Load

provide a safe way to upgrade the version on Production and allow easy rollback to the previous version.	Balancer (ALB) in front. Use Route 53 to change the ALB A-record Alias to the new ALB URL. Rollback by changing the A-record Alias to the old ALB.
An AWS OpsWorks application needs to safely deploy its new version on the production environment. You are tasked to prepare a rollback process in case of unexpected behavior.	Clone the OpsWorks Stack. Test it with the new URL of the cloned environment. Update the Route 53 record to point to the new version.
A development team needs full access to AWS CodeCommit but they should not be able to create/delete repositories.	Assign the developers with the AWSCodeCommitPowerUser IAM policy
During the deployment, you need to run custom actions before deploying the new version of the application using AWS CodeDeploy.	Add lifecycle hook action BeforeAllowTraffic
You need to run custom verification actions after the new version is deployed using AWS CodeDeploy.	Add lifecycle hook action AfterAllowTraffic
You need to set up AWS CodeBuild to automatically run after a pull request has been successfully merged using AWS CodeCommit	Create CloudWatch Events rule to detect pull requests and action set to trigger CodeBuild Project. Use AWS Lambda to update the pull request with the result of the project Build
You need to use AWS CodeBuild to create artifact and automatically deploy the new application version	Set CodeBuild to save artifact to S3 bucket. Use CodePipeline to deploy using CodeDeploy and set the build artifact from the CodeBuild output.
You need to upload the AWS CodeBuild artifact to Amazon S3	S3 bucket needs to have versioning and encryption enabled.
You need to review AWS CodeBuild Logs and have an alarm notification for build results on Slack	Send AWS CodeBuild logs to CloudWatch Log group. Create CloudWatch Events rule to detect the result of your build and target a Lambda function to send results to the Slack channel (or SNS notification)
Need to get a Slack notification for the status of the application deployments on AWS CodeDeploy	Create CloudWatch Events rule to detect the result of CodeDeploy job and target a notification

	to AWS SNS or a Lambda function to send results to Slack channel
Need to run an AWS CodePipeline every day for updating the development progress status	Create CloudWatch Events rule to run on schedule every day and set a target to the AWS CodePipeline ARN
Automate deployment of a Lambda function and test for only 10% of traffic for 10 minutes before allowing 100% traffic flow.	Use CodeDeploy and select deployment configuration CodeDeployDefault.LambdaCanary10Percent10Minutes
Deployment of Elastic Beanstalk application with absolutely no downtime. The solution must maintain full compute capacity during deployment to avoid service degradation.	Choose the “Rolling with additional Batch” deployment policy in Elastic Beanstalk
Deployment of Elastic Beanstalk application where the new version must not be mixed with the current version.	Choose the “Immutable deployments” deployment policy in Elastic Beanstalk
Configuration Management and Infrastructure-as-Code	
The resources on the parent CloudFormation stack needs to be referenced by other nested CloudFormation stacks	Use Export on the Output field of the main CloudFormation stack and use Fn::ImportValue function to import the value on the other stacks
On which part of the CloudFormation template should you define the artifact zip file on the S3 bucket?	The artifact file is defined on the AWS::Lambda::Function code resource block
Need to define the AWS Lambda function inline in the CloudFormation template	On the AWS::Lambda::Function code resource block, the inline function must be enclosed inside the ZipFile section.
Use CloudFormation to update Auto Scaling Group and only terminate the old instances when the newly launched instances become fully operational	Set AutoScalingReplacingUpdate : WillReplace property to TRUE to have CloudFormation retain the old ASG until the instances on the new ASG are healthy.
You need to scale-down the EC2 instances at night when there is low traffic using OpsWorks.	Create *Time-based* instances for automatic scaling of predictable workload.

Can't install an agent on on-premises servers but need to collect information for migration	Deploy the Agentless Discovery Connector VM on your on-premises data center to collect information.
Syntax for CloudFormation with an Amazon ECS cluster with ALB	Use the AWS::ECS::Service element for the ECS Cluster, AWS::ECS::TaskDefinition element for the ECS Task Definitions and the AWS::ElasticLoadBalancingV2::LoadBalancer element for the ALB.
Monitoring and Logging	
Need to centralize audit and collect configuration setting on all regions of multiple accounts	Setup an Aggregator on AWS Config.
Consolidate CloudTrail log files from multiple AWS accounts	Create a central S3 bucket with bucket policy to grant cross-account permission. Set this as destination bucket on the CloudTrail of the other AWS accounts.
Ensure that CloudTrail logs on the S3 bucket are protected and cannot be tampered with.	Enable Log File Validation on CloudTrail settings
Need to collect/investigate application logs from EC2 or on-premises server	Install CloudWatch Logs Agent to send the logs to CloudWatch Logs for storage and viewing.
Need to review logs from running ECS Fargate tasks	Enable awslogs log driver on the Task Definition and add the required logConfiguration parameter.
Need to run real-time analysis for collected application logs	Send logs to CloudWatch Logs, create a Lambda subscription filter, Elasticsearch subscription filter, or Kinesis stream filter.
Need to be automatically notified if you are reaching the limit of running EC2 instances or limit of Auto Scaling Groups	Track service limits with Trusted Advisor on CloudWatch Alarms using the ServiceLimitUsage metric.
Policies and Standards Automation	
Need to secure the buildspec.yml file which contains the AWS keys and database password stored in plaintext.	Store these values as encrypted parameter on SSM Parameter Store

Using default IAM policies for AWSCodeCommitPowerUser but must be limited to a specific repository only	Attach additional policy with Deny rule and custom condition if it does not match the specific repository or branch
You need to secure an S3 bucket by ensuring that only HTTPS requests are allowed for compliance purposes.	Create an S3 bucket policy that Deny if checks for condition aws:SecureTransport is **false**
Need to store a secret, database password, or variable, in the most cost-effective solution	Store the variable on SSM Parameter Store and enable encryption
Need to generate a secret password and have it rotated automatically at regular intervals	Store the secret on AWS Secrets Manager and enable key rotation.
Several team members, with designated roles, need to be granted permission to use AWS resources	Assign AWS managed policies on the IAM accounts such as, ReadOnlyAccess, AdministratorAccess, PowerUserAccess
Apply latest patches on EC2 and automatically create an AMI	Use Systems Manager automation to execute an Automation Document that installs OS patches and creates a new AMI.
Need to have a secure SSH connection to EC2 instances and have a record of all commands executed during the session	Install SSM Agent on EC2 and use SSM Session Manager for the SSH access. Send the session logs to S3 bucket or CloudWatch Logs for auditing and review.
Ensure that the managed EC2 instances have the correct application version and patches installed.	Use SSM Inventory to have a visibility of your managed instances and identify their current configurations.
Apply custom patch baseline from a custom repository and schedule patches to managed instances	Use SSM Patch Manager to define a custom patch baseline and schedule the application patches using SSM Maintenance Windows
Incident and Event Response	
Need to get a notification if somebody deletes files in your S3 bucket	Setup Amazon S3 Event Notifications to get notifications based on specified S3 events on a particular bucket.
Need to be notified when an RDS Multi-AZ failover happens	Setup Amazon RDS Event Notifications to detect specific events on RDS.

Get a notification if somebody uploaded IAM access keys on any public GitHub repositories	Create a CloudWatch Events rule for the AWS_RISK_CREDENTIALS_EXPOSED event from AWS Health Service. Use AWS Step Functions to automatically delete the IAM key.
Get notified on Slack when your EC2 instance is having an AWS-initiated maintenance event	Create a CloudWatch Events rule for the AWS Health Service to detect EC2 Events. Target a Lambda function that will send a notification to the Slack channel
Get notified of any AWS maintenance or events that may impact your EC2 or RDS instances	Create a CloudWatch Events rule for detecting any events on AWS Health Service and send a message to an SNS topic or invoke a Lambda function.
Monitor scaling events of your Amazon EC2 Auto Scaling Group such as launching or terminating an EC2 instance.	Use Amazon EventBridge or CloudWatch Events for monitoring the Auto Scaling Service and monitor the EC2 Instance-Launch Successful and EC2 Instance-Terminate Successful events.
View object-level actions of S3 buckets such as upload or deletion of object in CloudTrail	Set up Data events on your CloudTrail trail to record object-level API activity on your S3 buckets.
Execute a custom action if a specific CodePipeline stage has a FAILED status	Create CloudWatch Event rule to detect failed state on the CodePipeline service, and set a target to SNS topic for notification or invoke a Lambda function to perform custom action.
Automatically rollback a deployment in AWS CodeDeploy when the number of healthy instances is lower than the minimum requirement.	On CodeDeploy, create a deployment alarm that is integrated with Amazon CloudWatch. Track the MinimumHealthyHosts metric for the threshold of EC2 instances and trigger the rollback if the alarm is breached.
Need to complete QA testing before deploying a new version to the production environment	Add a Manual approval step on AWS CodePipeline, and instruct the QA team to approve the step before the pipeline can resume the deployment.
Get notified for OpsWorks auto-healing events	Create a CloudWatch Events rule for the OpsWorks Service to track the auto-healing events

Automatically Run CodeBuild Tests After a Developer Creates a CodeCommit Pull Request

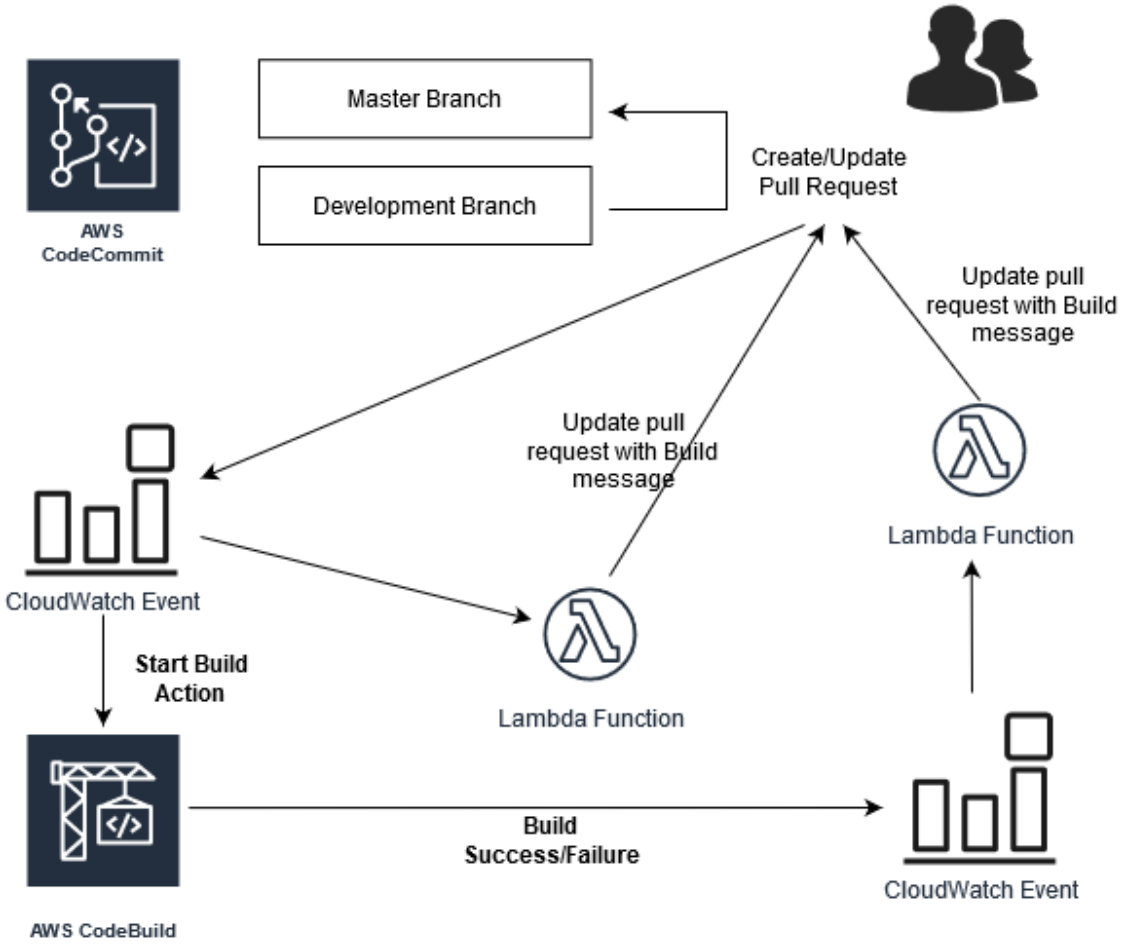
AWS CodeCommit allows developers to create multiple branches, which can be used for developing new features or fixing application bugs. These changes can then be merged on the master branch, which is usually used for the production release. To merge the changes to the master branch on CodeCommit, the developers create a pull request. But these code changes need to be validated in order to make sure that the changes integrate properly with the current code base.

To validate your changes on the code base, you can run an AWS CodeBuild project to the build and test your pull request and based on the result, decide on whether to accept the merging of the code or reject the pull request.

DevOps Exam Notes:

You can automate the validation of AWS CodeCommit pull requests with AWS CodeBuild and AWS Lambda with the help of CloudWatch Events. Basically, CloudWatch Events will detect the pull requests on your CodeCommit repository and then trigger the AWS CodeBuild project with the Lambda function updating the comments on the pull request. The results of the build are also detected by AWS CloudWatch Events and will trigger the Lambda function to update the pull request with the results.

The following diagram shows an example workflow on how you can automate the validation of a pull request with AWS CodeCommit, AWS CodeBuild, CloudWatch Event, and AWS Lambda.



1. The AWS CodeCommit repository contains two branches, the master branch that contains approved code, and the development branch, where changes to the code are developed.

2. Push the new code on the AWS CodeCommit Development branch.

3. Create a pull request to merge their changes to the master branch.

4. Create a CloudWatch Events rule to detect the pull request and have it trigger an AWS Lambda function that will post an automated comment to the pull request that indicates that a build to test the changes is about to begin.

5. With the same CloudWatch Events rule, have it trigger an AWS CodeBuild project that will build and validate the changes.

6. Create another CloudWatch Events rule to detect the output of the build. Have it trigger another Lambda function that posts an automated comment to the pull request with the results of the build and a link to the build logs.

Based on this automated testing, the developer who opened the pull request can update the code to address any build failures, and then update the pull request with those changes. The validation workflow will run again and will produce the updated results.

Once the pull request is successfully validated, you can accept the pull request to merge the changes to the master branch.

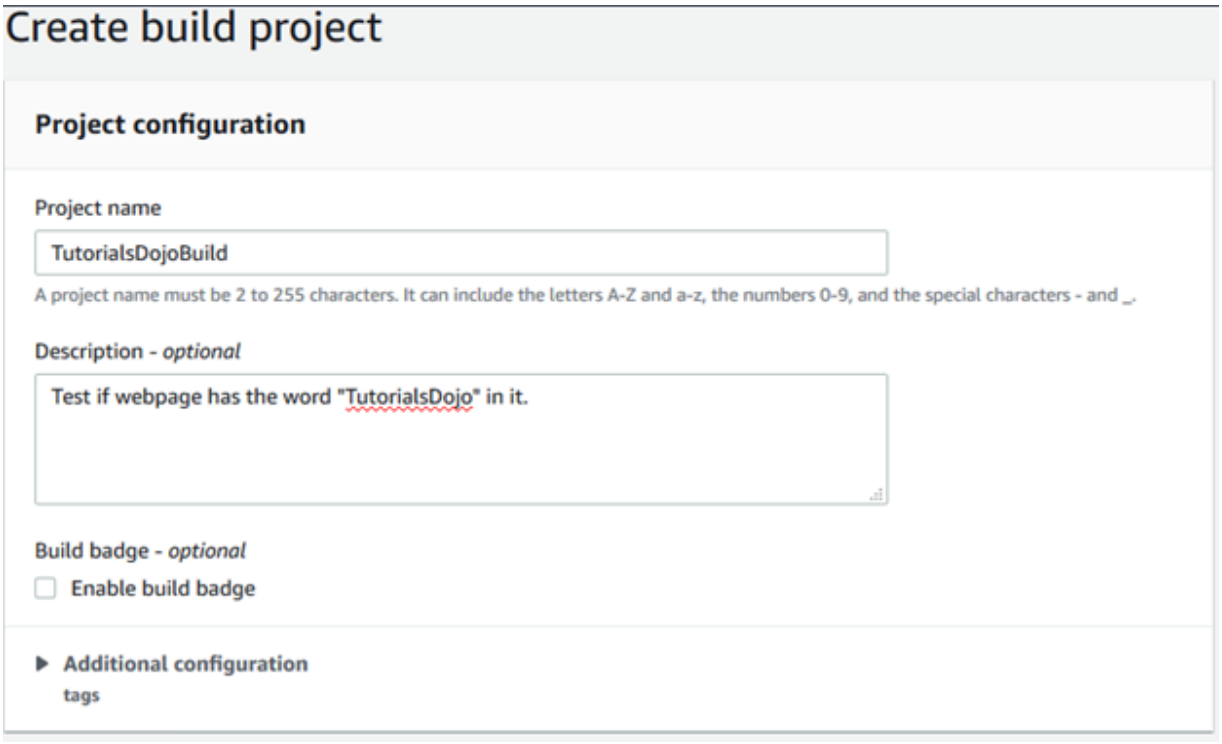
Sources:
https://docs.aws.amazon.com/codebuild/latest/userguide/how-to-create-pipeline.html
https://aws.amazon.com/blogs/devops/validating-aws-codecommit-pull-requests-with-aws-codebuild-and-aws-lambda/

CodeBuild with CloudWatch Logs, Metrics, and Alarms

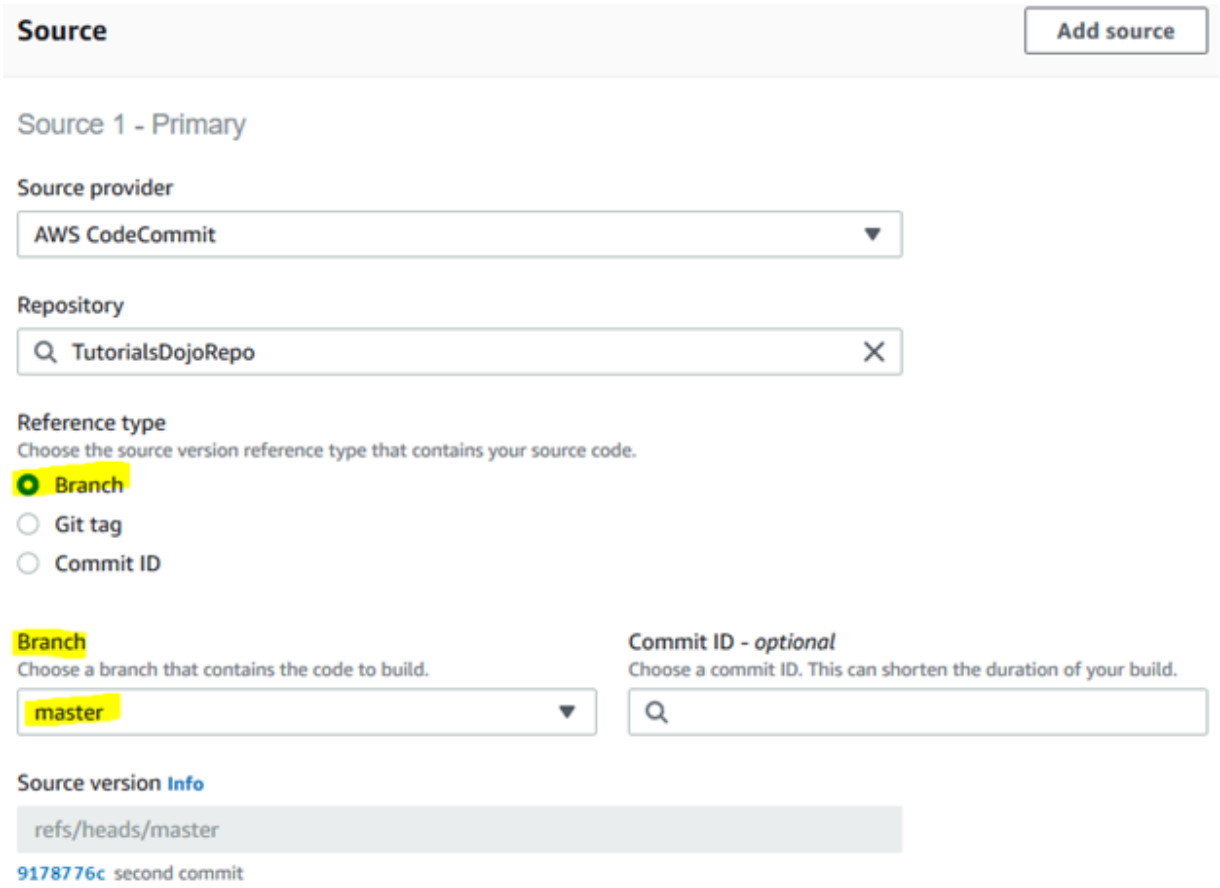
AWS Codebuild allows you to compile, build, run tests, and produce an artifact of your code. This can be integrated into your CI/CD process. For example, you are using AWS CodeCommit as version control for your source code repository. After a developer pushes new code to the Development branch, you can have an automatic trigger for CodeBuild to run your project build, test your application, and then upload the output artifact to S3. The artifact file on S3 can then be used by deployment tools such as AWS CodeDeploy to have it deployed on your EC2 instances, ECS, or Lambda functions.

Here are the steps on how to create a CodeBuild build project and save the artifact on an S3 bucket. We will also demonstrate how CodeBuild integrates with AWS CloudWatch.

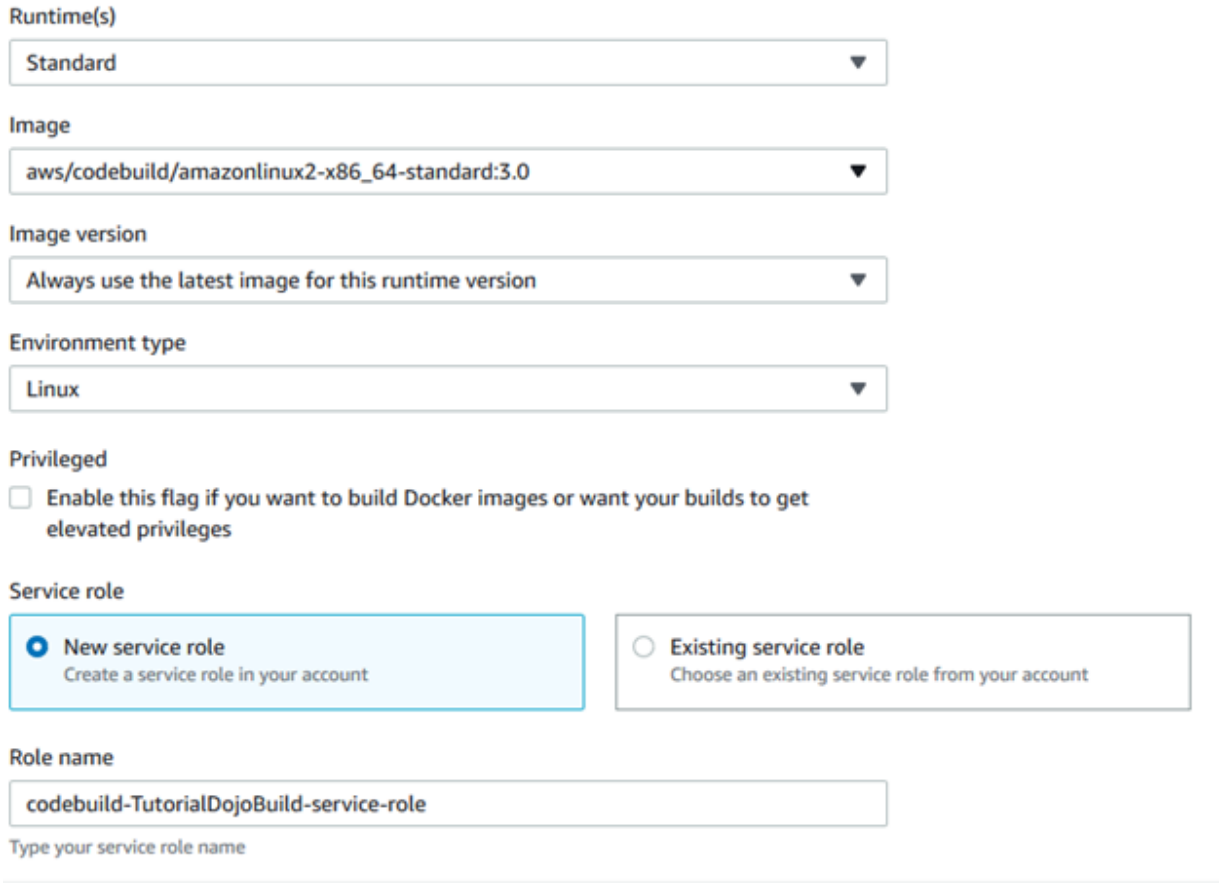
1. Go to AWS Code Build > Build Projects and click Create Build Project. Input your details for this project.



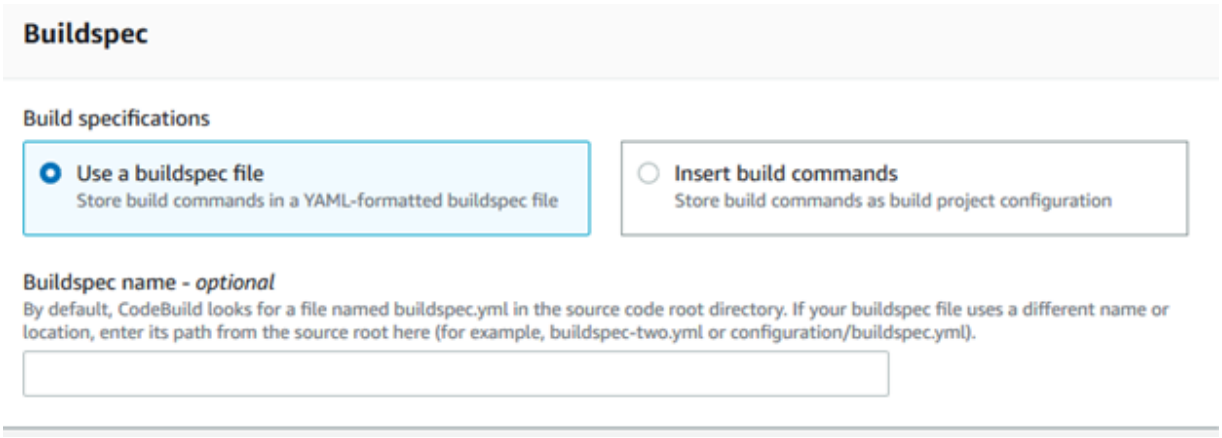
2. CodeBuild supports several sources for your application code, including Amazon S3, AWS CodeCommit, GitHub, Bitbucket, etc. Using CodeBuild, select your source repository and branch.



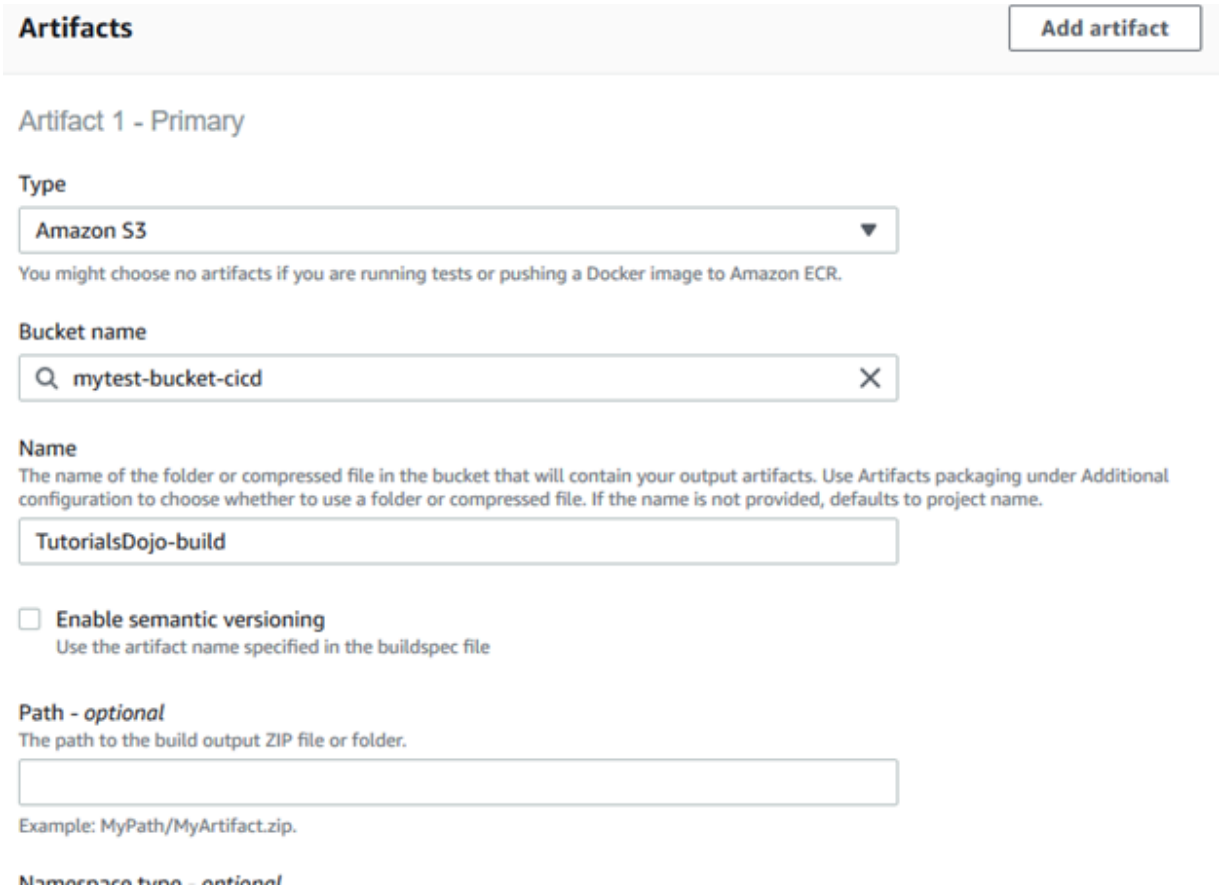
3. Use the Amazon Linux runtime since we'll build this for the Amazon Linux AMI.



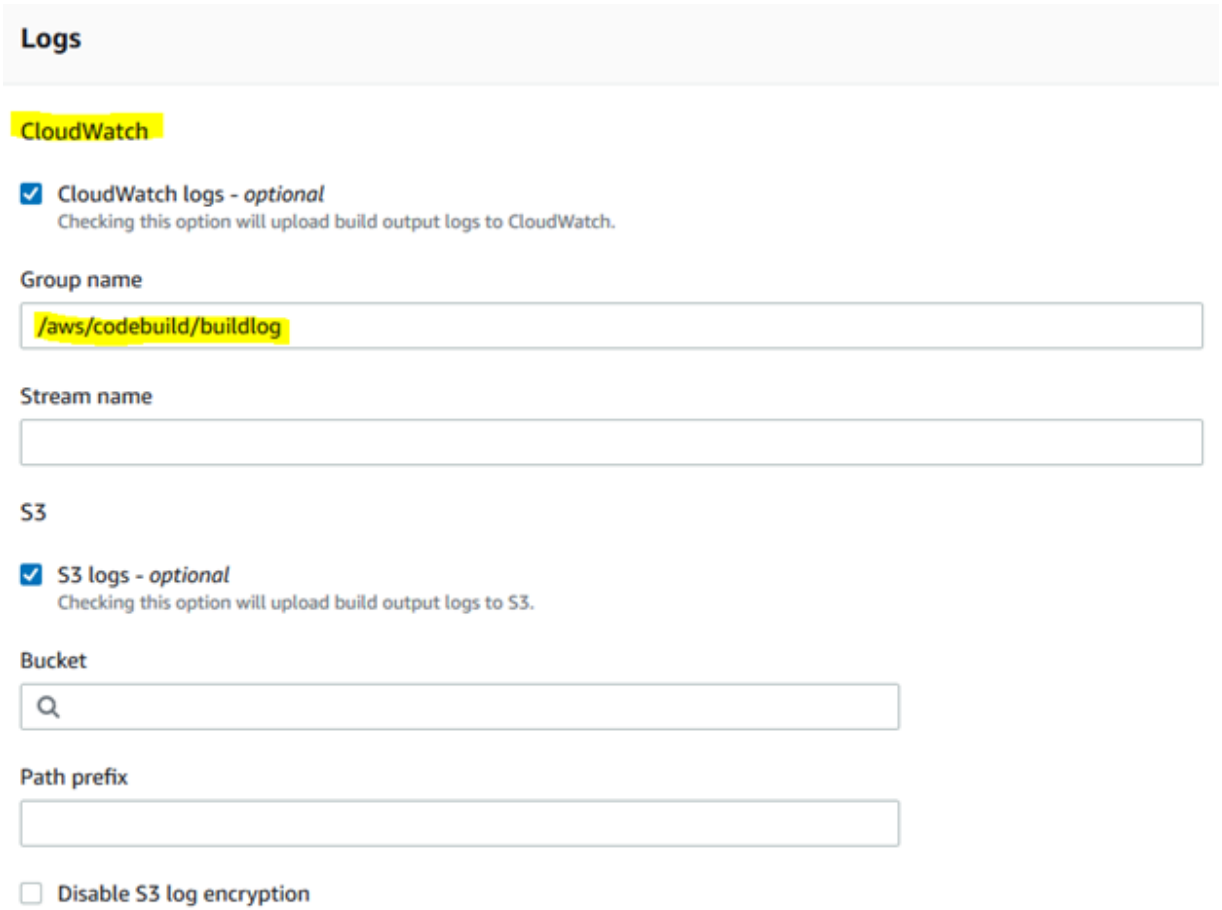
4. By default, the build specification filename is “buildspec.yml”. This should be in the root folder of your application code.



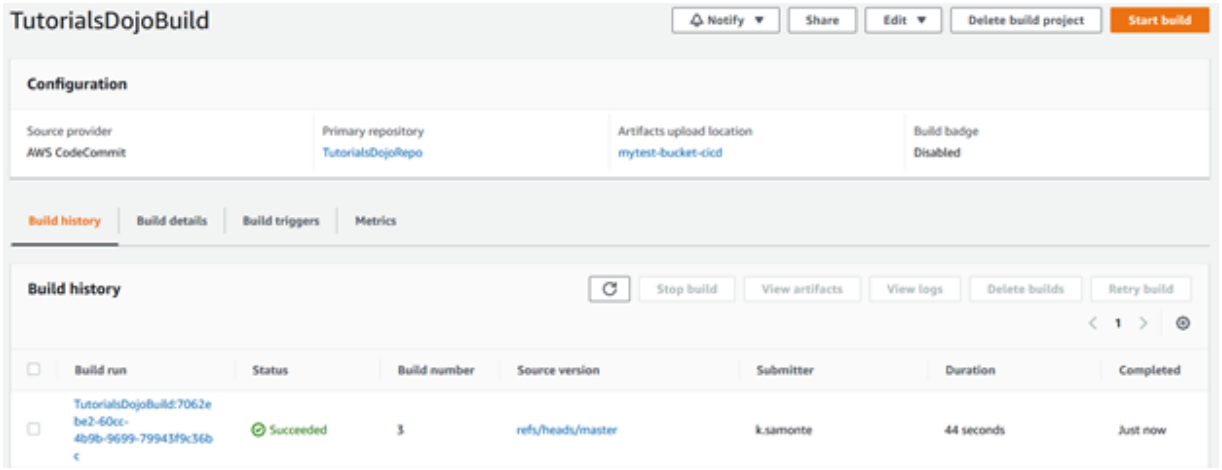
5. Specify which S3 bucket you are going to send the output artifact of your build.



6. On the logs part, you can have the option to send the build logs to CloudWatch Log group or send to an S3 bucket. The AWS CloudWatch log group or S3 bucket must exist first before you specify it here.

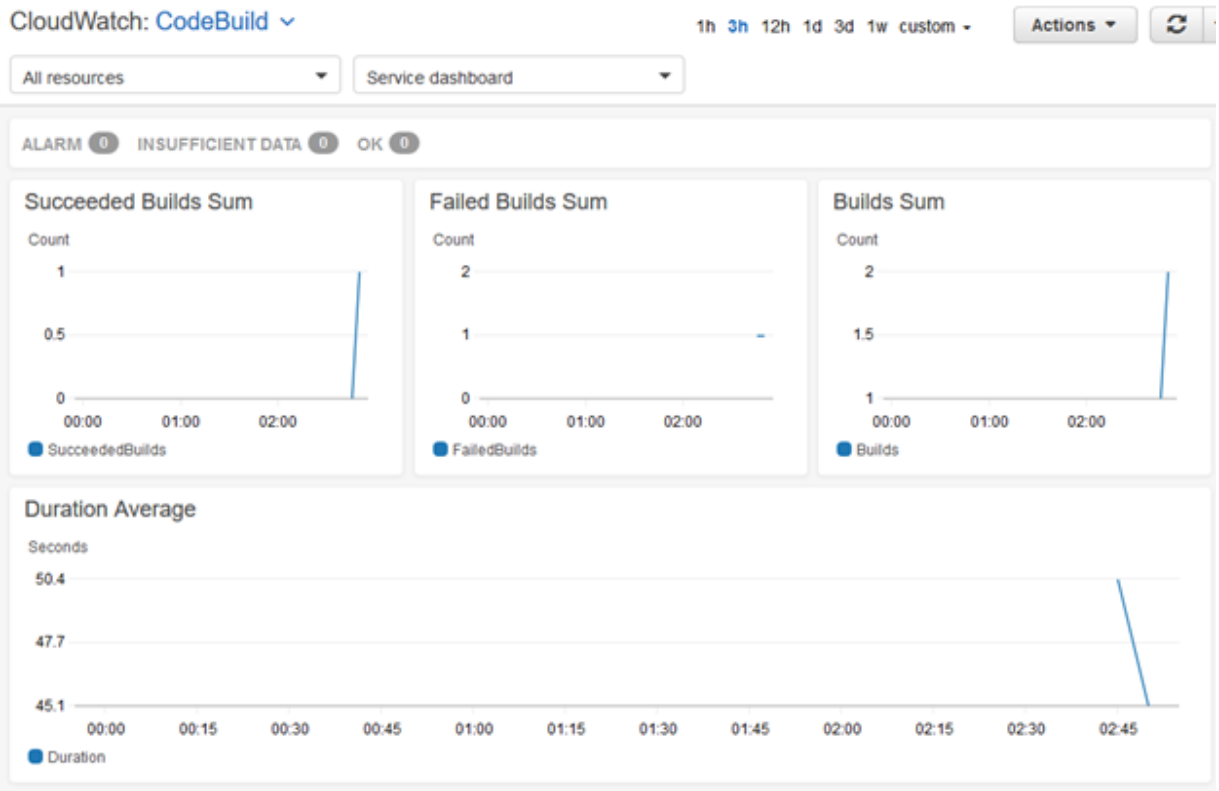


7. Click “Create build project”. Select your project and click the “Start build” button.

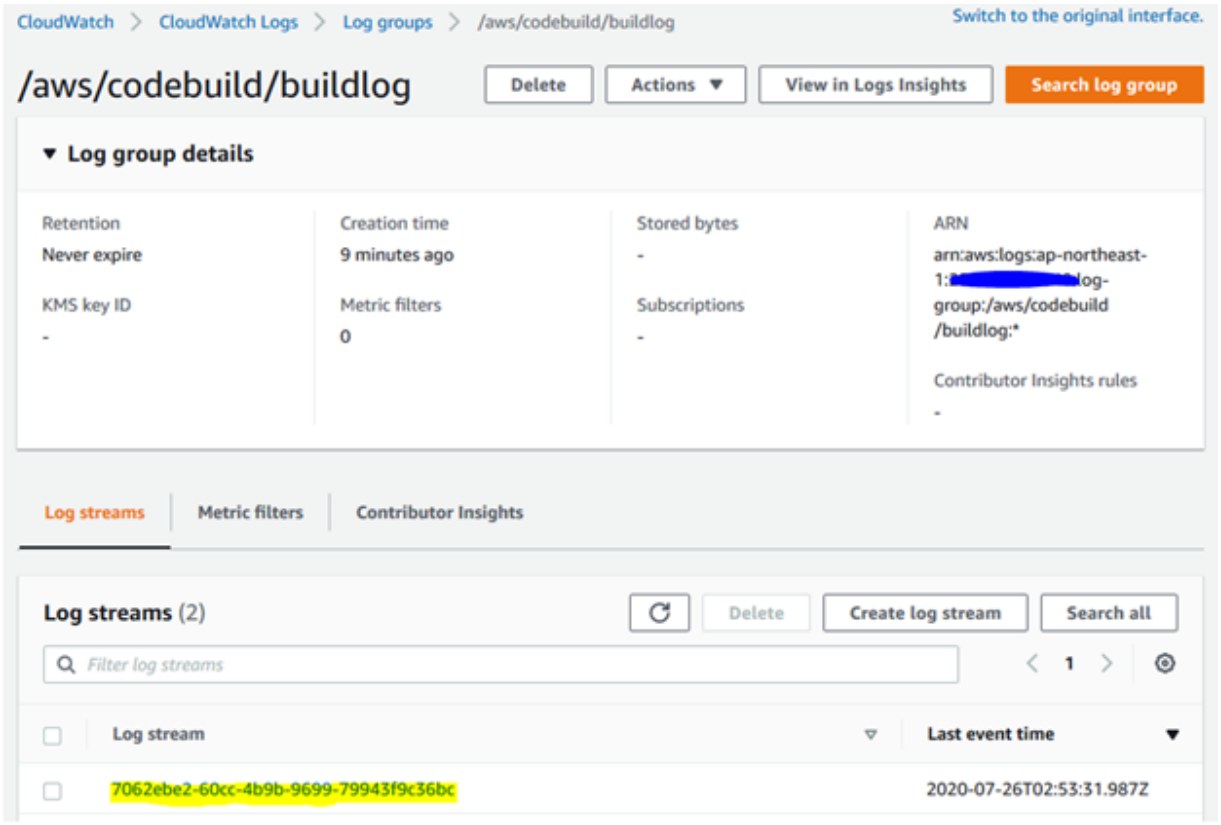


8. After the build, you should see the artifact on the S3 bucket. You will also see the build logs of your project if you click on the Build run ID of your project.

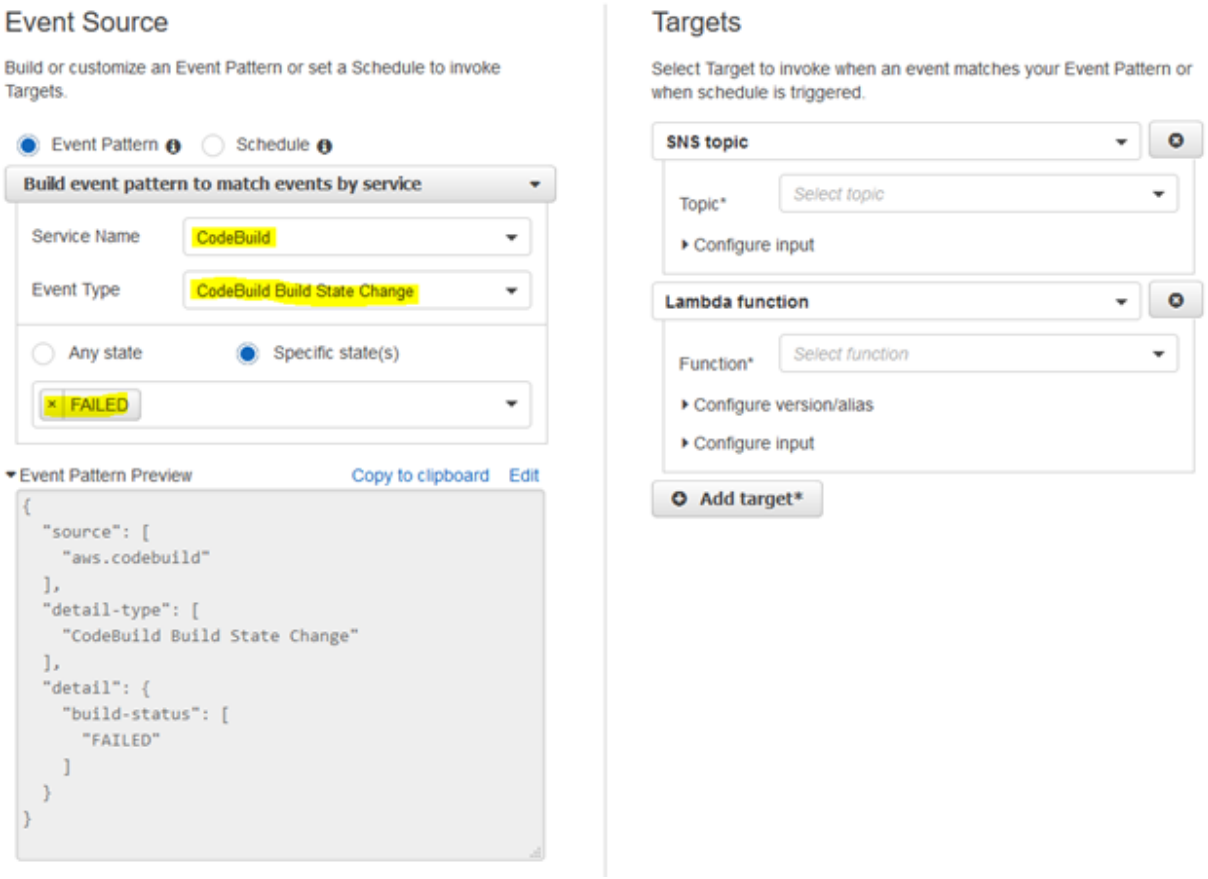
9. AWS CodeBuild is also integrated on CloudWatch Logs. When you go to CloudWatch Dashboard, you can select the pre-defined CodeBuild Dashboard where you will see the metrics such as Successful builds, Failed builds, etc.



10. You should also be able to see the AWS CloudWatch Log Group you created and that CodeBuild logs are delivered to it.



11. Using CloudWatch Event rules, you can also create a rule to detect Successful or Failed builds, and then use the Trigger to invoke a Lambda function to send you a slack notification, or use an SNS Topic target to send you an email notification about the build status.



DevOps Exam Notes:

AWS CodeBuild is integrated with AWS CloudWatch. A predefined dashboard is created on CloudWatch to view metrics regarding project builds on your account. You can send CodeBuild build logs to a CloudWatch log group so you have a central location to review them. You can set filters and alarms on these logs and set up a notification. You can also use CloudWatch Events to detect changes on your build project, such as FAILED builds and have it invoke a Lambda function to send you a message on your Slack channel, or SNS topic to send you an email notification.

Sources:
https://docs.aws.amazon.com/codebuild/latest/userguide/create-project.html
https://docs.aws.amazon.com/codebuild/latest/userguide/create-project-console.html

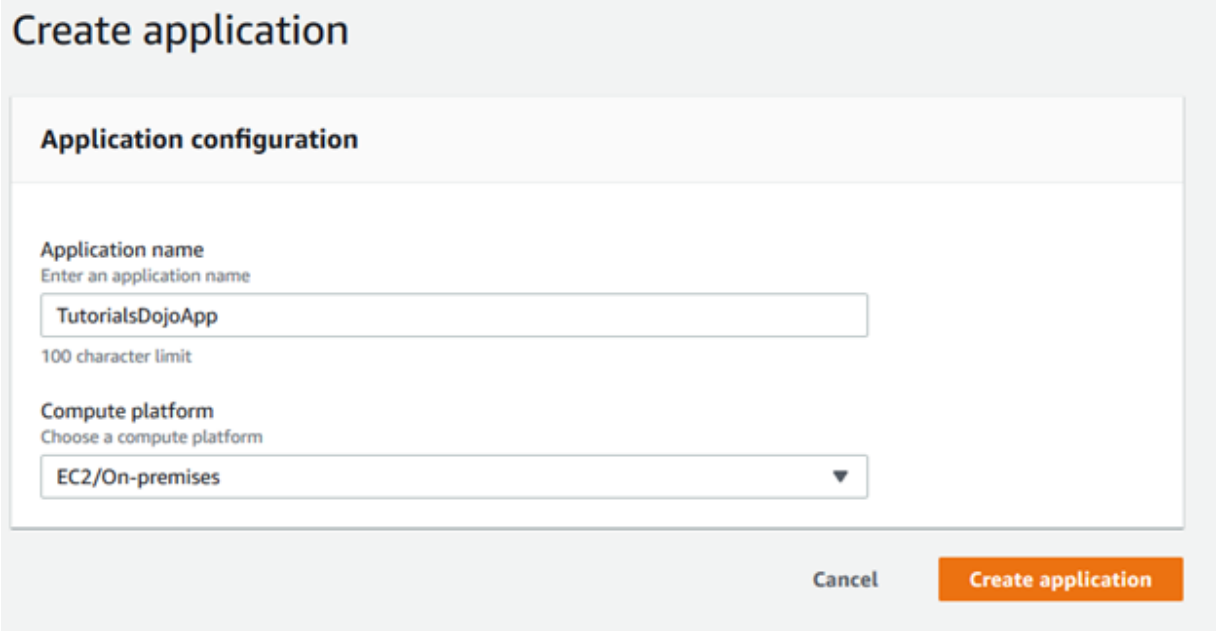
CodeDeploy with CloudWatch Logs, Metrics, and Alarms

AWS CodeDeploy allows you to automate software deployments to a variety of services such as EC2, AWS Fargate, AWS Lambda, and your on-premises servers.

For example, after you have produced from AWS CodeBuild, you can have CodeDeploy fetch the artifact from the AWS S3 bucket and then deploy it to your AWS instances. Please note that the target instances need to have the AWS CodeDeploy agent installed on them for this to be successful and have proper Tags.

Here are the steps to create a deployment on AWS CodeDeploy, including a discussion on how it integrates with AWS CloudWatch.

1. Go to AWS CodeDeploy > Applications and click “Create Application”. Input details of your application and which platform it is running.

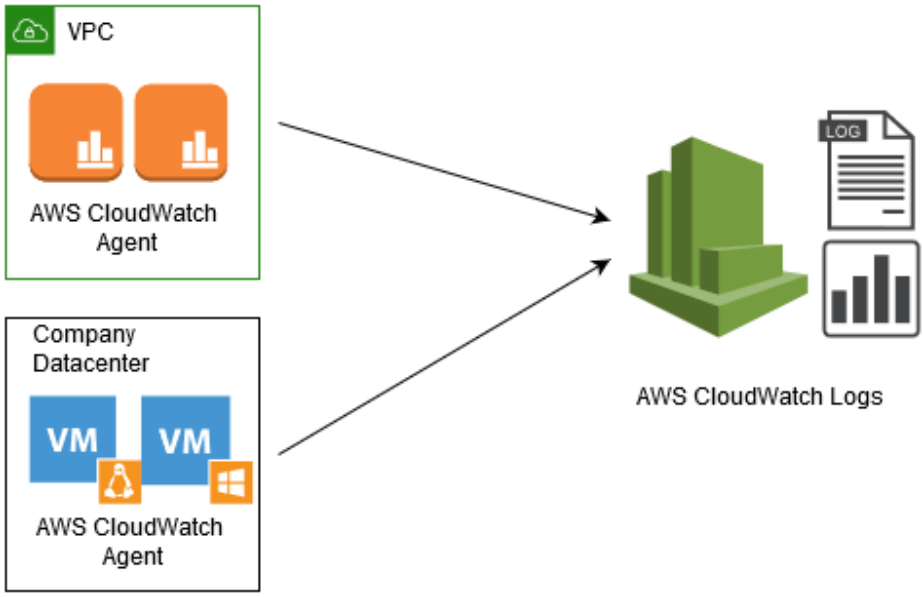


2. Select your application and create a Deployment Group. CodeDeploy needs to have an IAM permission to access your targets as well as read the AWS S3 bucket containing the artifact to be deployed.

Fetching Application Logs from Amazon EC2, ECS and On-premises Servers

Application logs are vital for monitoring, troubleshooting, and regulatory compliance of every enterprise system. Without it, your team will waste a lot of time trying to find the root cause of an issue that can be easily detected by simply checking the logs. These files often live inside the server or a container. Usually, you have to connect to the application server via SSH or RDP before you can view the logs. This manual process seems to be inefficient, especially for high-performance organizations with hybrid network architectures.

Using the Amazon CloudWatch Logs agent, you can collect system metrics and logs from your Amazon EC2 instances and on-premises application servers. Gone are the days of spending several minutes connecting to your server and manually retrieving the application logs. For Linux servers, you don't need to issue a *tail - f* command anymore since you can view the logs on the CloudWatch dashboard on your browser in near real-time. It also collects both system-level and custom metrics from your EC2 instances and on-premises servers, making your monitoring tasks a breeze.



You have to manually download and install the Amazon CloudWatch Logs agent to your EC2 instances or on-premises servers using the command line. Alternatively, you can use AWS Systems Manager to automate the installation process. For your EC2 instances, it is preferable to attach an IAM Role to allow the application to send data to CloudWatch. For your on-premises servers, you have to create a separate IAM User to integrate your server to CloudWatch. Of course, you should first establish a connection between your on-premises data center and VPC using a VPN or a Direct Connect connection. You have to use a named profile in your local server that contains the credentials of the IAM user that you created.

If you are running your containerized application in Amazon Elastic Container Service (ECS), you can view the different logs from your containers in one convenient location by integrating Amazon CloudWatch. You can configure your Docker containers' tasks to send log data to CloudWatch Logs by using the *awslogs* log driver.


```
"logConfiguration": {
    "logDriver": "awslogs",
    "options": {
        "awslogs-group": "awslogs-wordpress",
        "awslogs-region": "us-west-2",
        "awslogs-stream-prefix": "awslogs-example"
    }
```

If your ECS task is using a Fargate launch type, you can enable the *awslogs* log driver and add the required *logConfiguration* parameters to your task definition. For EC2 launch types, you have to ensure that your Amazon ECS container instances have an attached IAM role that contains *logs:CreateLogStream* and *logs:PutLogEvents* permissions. Storing the log files to Amazon CloudWatch prevents your application logs from taking up disk space on your container instances.

Sources:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/UseCloudWatchUnifiedAgent.html
https://docs.aws.amazon.com/AmazonECS/latest/userguide/using_awslogs.html

CloudWatch Logs Agent to CloudWatch Logs Subscription

You can install CloudWatch Agent on your on-premises instances or EC2 instances to allow them to send detailed metrics to CloudWatch or send application logs to CloudWatch Logs. The logs will be sent to your configured CloudWatch log group for viewing and searching.

DevOps Exam Notes:

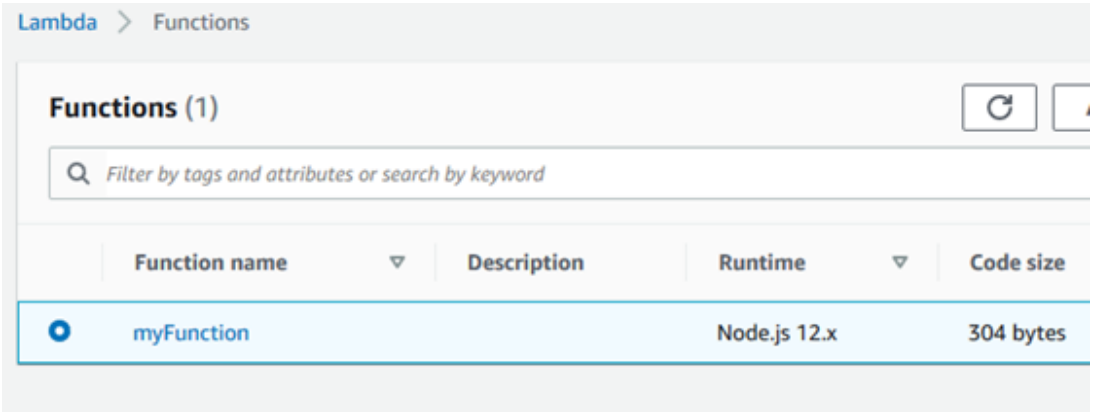
Additionally, you can use CloudWatch Logs subscriptions to get access to a real-time feed of log events from CloudWatch Logs and have it delivered to other services such as an Amazon Kinesis stream, an Amazon Kinesis Data Firehose stream, or AWS Lambda for custom processing, analysis, or loading to other systems. This way, you can perform near real-time analysis of the logs, further log processing using AWS Lambda, or have advanced searching capability using Elasticsearch.

To begin subscribing to log events, create the receiving source such as a Kinesis stream or Lambda function where the events will be delivered.

A subscription filter defines the filter pattern to use to sort out which log events get delivered to your AWS resource, as well as information about where to send matching log events to.

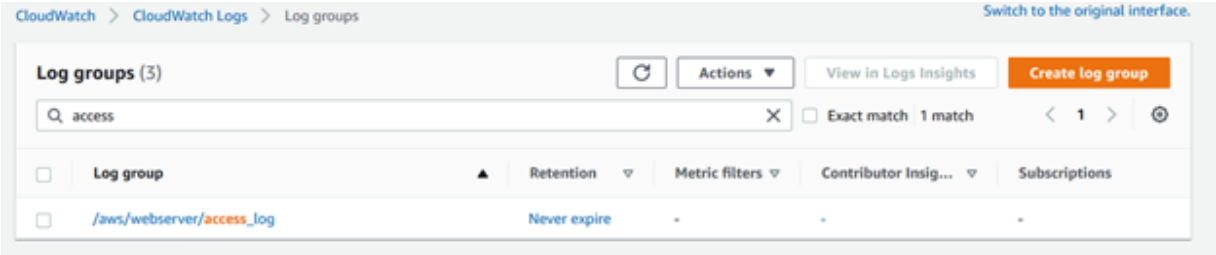
Here are the steps to setup CloudWatch logs subscription:

1. Create a receiving source, such as a Kinesis stream, Elasticsearch cluster, or Lambda function.

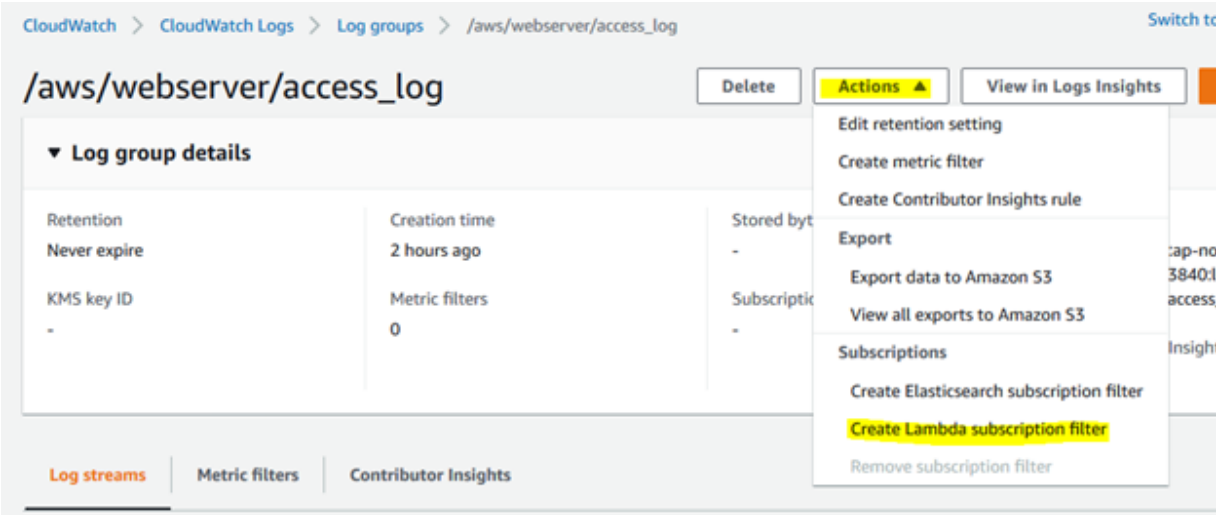


2. Install CloudWatch Unified Agent on the EC2 instance (Linux or Windows) and configure it to send application logs to CloudWatch log group.

3. Create the CloudWatch log group that will access the logs.



4. Create a subscription filter for the log group and select the Lambda function or Elasticsearch cluster that you created. If you need to set a Kinesis Data Firehose stream as the subscription filter, you will need to use AWS CLI as the web console does not support it yet.



Sources:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/QuickStartEC2Instance.html
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/Subscriptions.html
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/SubscriptionFilters.html

Monitoring Service Limits with Trusted Advisor

AWS Trusted Advisor provides you real-time guidance to help you provision your resources following AWS best practices. It provides recommendations on 5 categories – Cost Optimization, Performance, Security, Fault Tolerance and Service Limits.

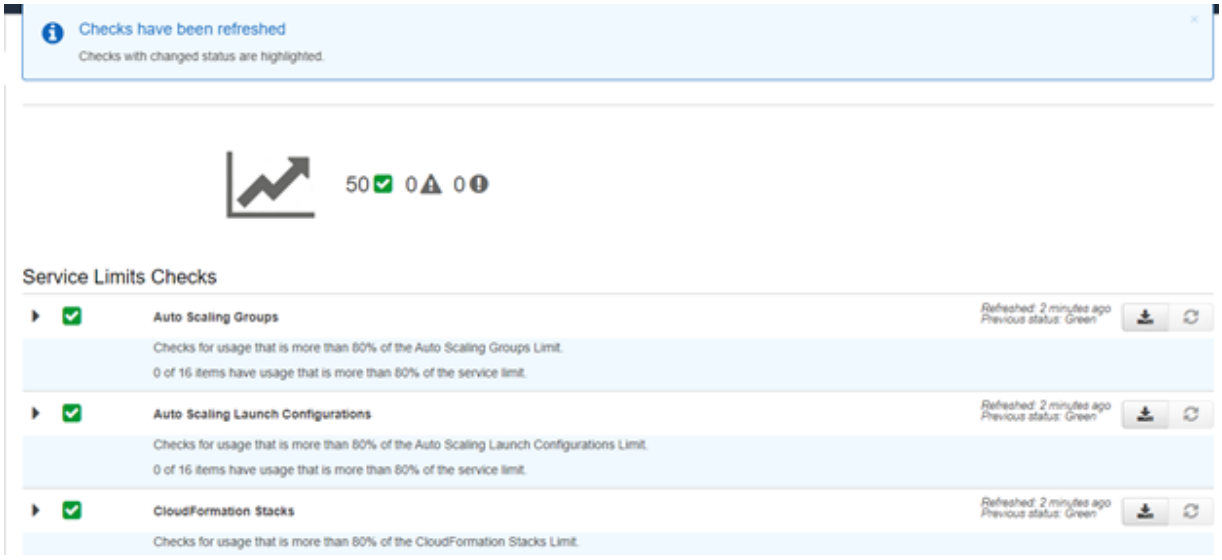
With Trusted Advisor's Service Limit Dashboard, you can view, refresh, and export utilization and limit data on a per-limit basis. These metrics are published on AWS CloudWatch in which you can create custom alarms based on percentage of service utilization against limits, understand the number of resources by each check, or view time-aggregate views of check results across service categories.

DevOps Exam Notes:

You need to understand that service limits are important when managing your AWS resources. In the exam you can be given a scenario in which you have several Auto Scaling groups in your AWS account and you need to make sure that you are not reaching the service limit when you perform your blue/green deployments for your application. You can track service limits with Trusted Advisor and CloudWatch Alarms. The ServiceLimitUsage metric on CloudWatch Alarms is only visible for Business and Enterprise support customers.

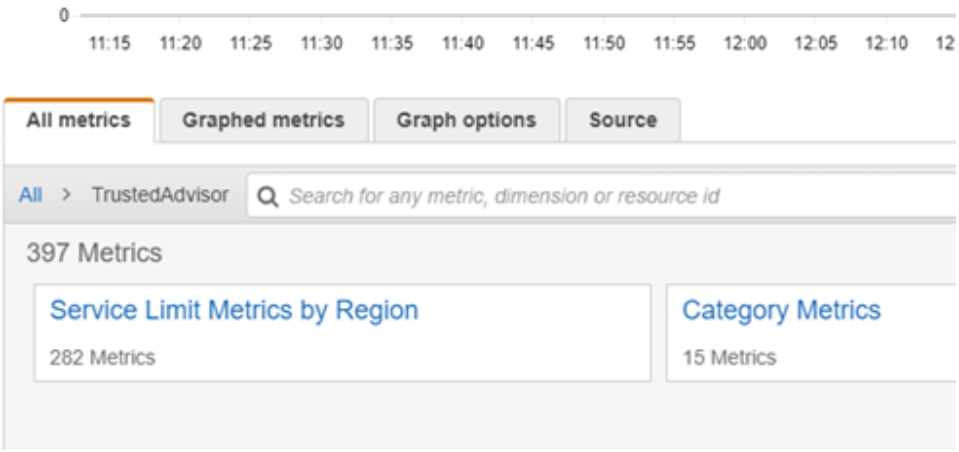
Here's how you can create a CloudWatch Alarm to detect if you are nearing your auto scaling service limit and send a notification so you can request for a service limit increase to AWS support.

1. First, head over to AWS Trusted Advisor > Service Limits and click the refresh button. This will refresh the service limit status for your account.

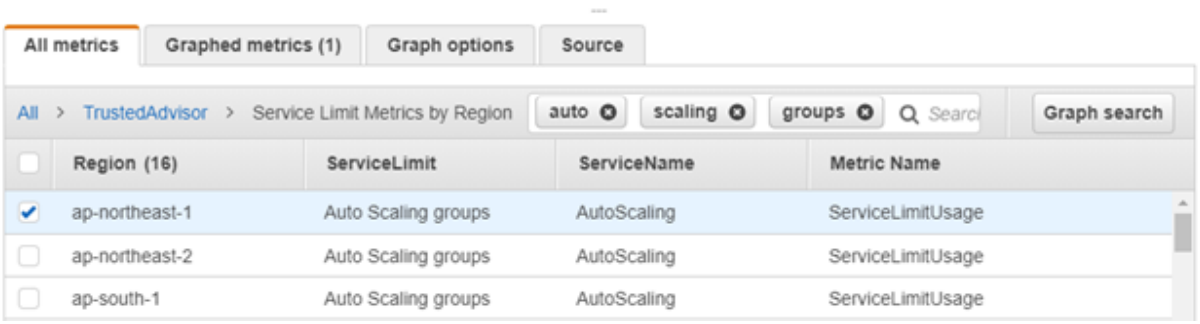


2. Go to CloudWatch > Alarms. Make sure that you are in the North Virginia region. Click the “Create Alarm” button and click “Select Metric”.

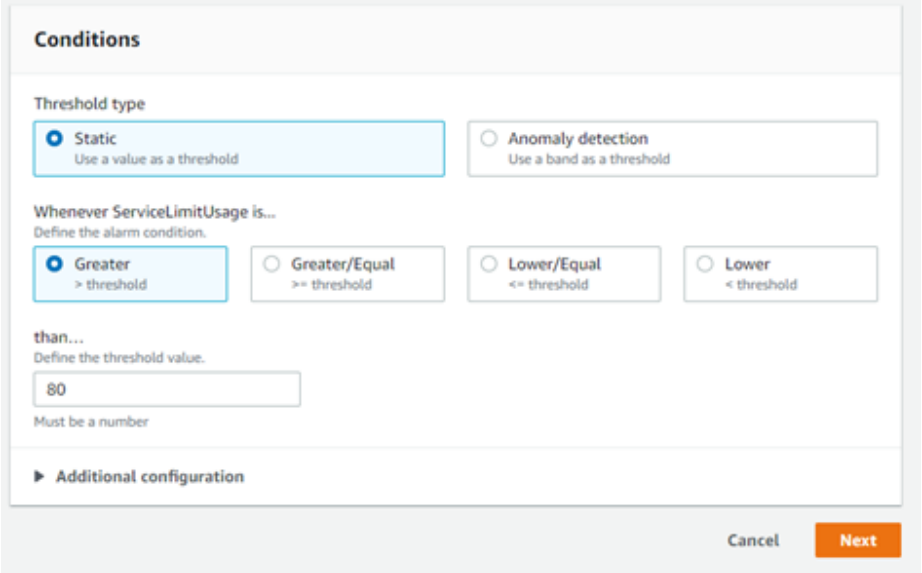
3. In the “All metrics” tab, click “Trusted Advisor” category and you will see “Service Limits by Region”.



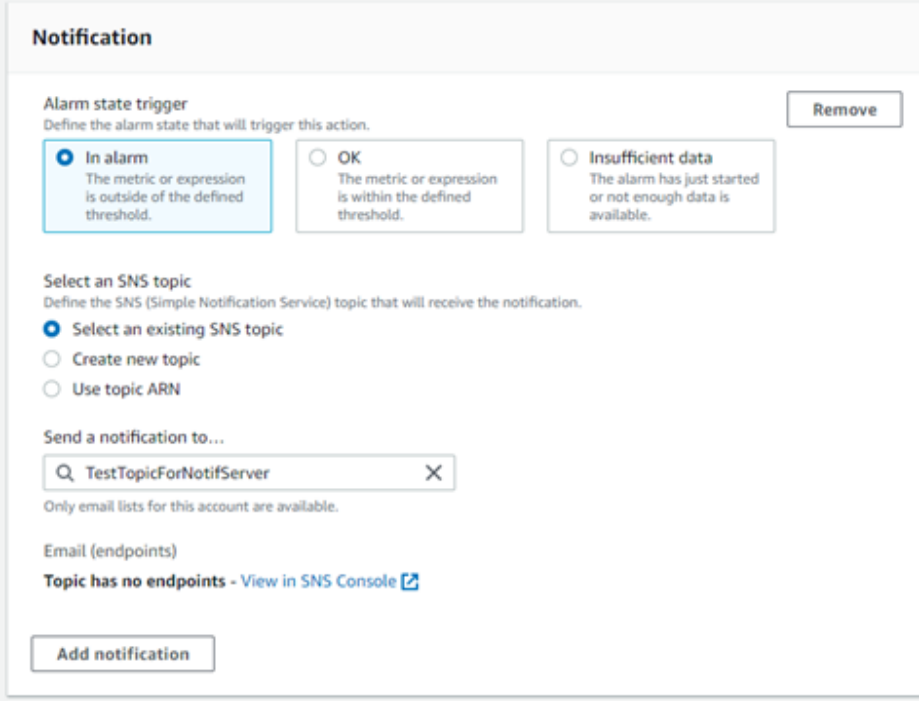
4. Search for Auto Scaling groups on your desired region and click Select Metric.



5. We can set the condition for this Alarm so that when your Auto Scaling group reaches 80 for that particular region, the alarm is triggered.



6. You can then configure an SNS topic to receive a notification for this alarm.



7. Click Next to Preview the alarm and click “Create Alarm” to create the alarm.

Sources:
https://aws.amazon.com/about-aws/whats-new/2017/11/aws-trusted-advisor-adds-service-limit-dashboard-and-cloudwatch-metrics/
https://aws.amazon.com/blogs/mt/monitoring-service-limits-with-trusted-advisor-and-amazon-cloudwatch/

Incident and Event Response Management on AWS

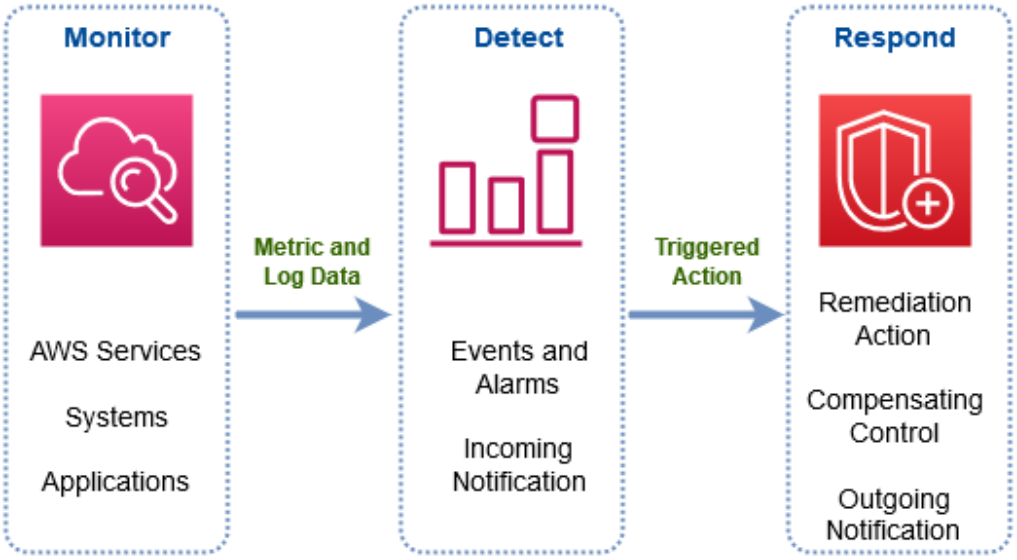
There is always a lot of action in a typical IT environment. The development team builds the new features within a given sprint and rectifies any software defects along the way. The operations team continuously monitors the underlying resources used by the enterprise applications and troubleshoots technical problems throughout the day. Critical incidents might suddenly happen that could adversely affect your production environments and many other system events that could potentially bring your business operations into a screeching halt! Thus, automating the incident and event management of your infrastructure is of utmost importance to maintain maximum efficiency and to reduce unnecessary interruptions.

AWS provides a myriad of services and features to automate manual and repetitive IT tasks. Gone are the days of receiving outraged emails, calls, or tweets from your customers because your production server was down and you're not even aware of it. By using Amazon CloudWatch Events and CloudWatch Alarms, your teams can immediately be notified of any system events or breach of a specified threshold. You can also enable the Auto Healing feature in your AWS OpsWorks Stack or place your EC2 instances in an Auto Scaling group to automatically replace failed instances without manual intervention. Deployment issues can quickly be resolved or prevented through deployment monitoring and automatic rollbacks using AWS CodeDeploy and CloudWatch Events. S3 Events enables you to monitor unauthorized actions in your S3 buckets continuously, and RDS Events keeps you in the know for any failover, configuration change, or backup-related events that affect your database tier. Amazon EventBridge can also track all the changes in your AWS services, your custom applications, and external Software-as-a-Service (SaaS) applications in real-time. These AWS features and services complement your existing security information and event management (SIEM) solutions to manage your entire cloud infrastructure properly.

DevOps Exam Notes:

An event indicates a change in a resource that is routed by an ‘event bus’ to its associated rule. A custom event bus can receive rules from AWS services, custom applications, and SaaS partner services. Both Amazon CloudWatch Events and Amazon EventBridge have the same API and underlying service, but the latter provides more features. Amazon EventBridge is the ideal service to manage your events, but take note that CloudWatch Events is still available and not entirely deprecated.

Automated Remediation Flow



The process of auditing your applications, systems, and infrastructure services in AWS is also simplified as all events and activities are appropriately tracked. Within minutes, you can identify the root cause of a recent production incident by checking the event history in AWS CloudTrail. Real-time logs feed on CloudWatch can be delivered to an Amazon Kinesis stream, an Amazon Kinesis Data Firehose stream, or AWS Lambda for processing, analysis, or integration to other systems through CloudWatch Subscription Filters. Security incidents can be remediated immediately by setting up custom responses to Amazon GuardDuty findings using Amazon CloudWatch Events. In this way, any security vulnerability in your AWS resources, such as an SSH brute force attack on one of your EC2 instances, can immediately be identified.

AWS-Scheduled Maintenance Notification to Slack Channel via CloudWatch Events

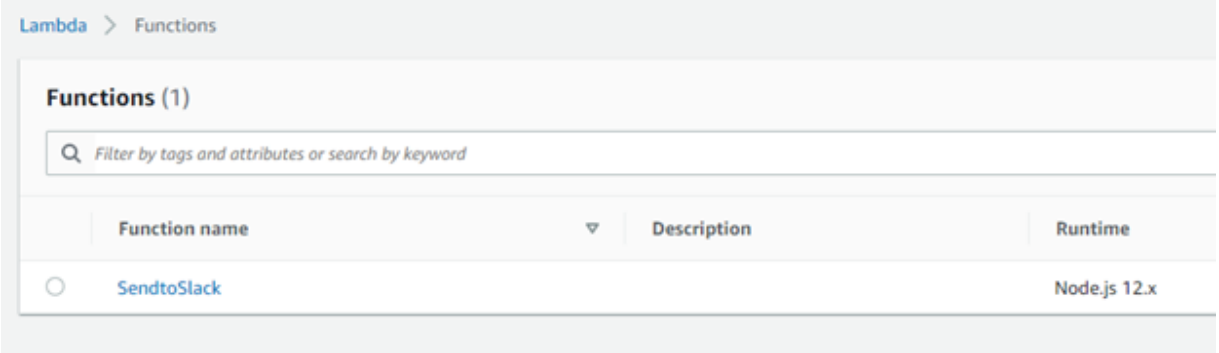
AWS Scheduled maintenance events are listed on AWS Health Dashboard. For example, if AWS needs to perform maintenance on the underlying host of your EC2 instance in which the EC2 instance usually needs to shut down, you will see an event on your AWS Health Dashboard for it. You can use CloudWatch Events to detect these changes and you trigger notifications so you will be notified for these events. You can then perform the needed actions based on the events

You can choose the following types of targets when using CloudWatch Events as a part of your AWS Health workflow:

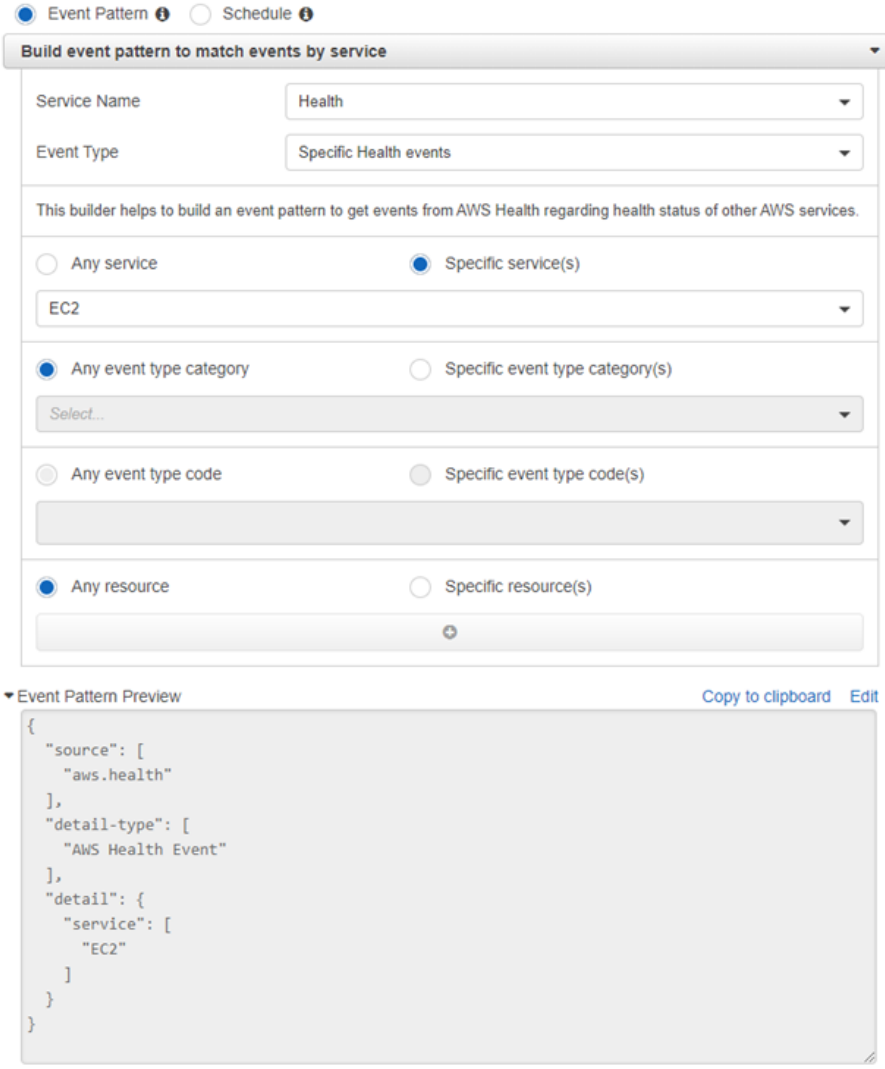
- AWS Lambda functions
- Amazon Kinesis Data Streams
- Amazon Simple Queue Service (Amazon SQS) queues
- Built-in targets (CloudWatch alarm actions)
- Amazon Simple Notification Service (Amazon SNS) topics

For example, you can use a Lambda function to pass a notification to a Slack channel when an AWS Health event occurs. Here are the steps to do this.

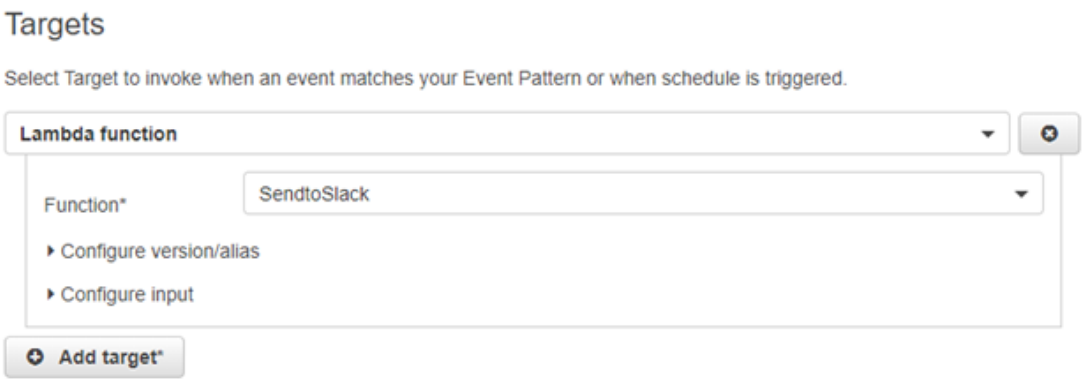
1. Create a Lambda function that will send a message to your Slack Channel. A Nodejs or Python script will suffice. The function will call an API URL for the Slack channel passing along the message.



2. Go to AWS CloudWatch Events and create a rule.
3. Set a rule for the AWS Health Service, and EC2 Service Event Type.



4. Add a Target for this event to run the Lambda function and save the CloudWatch Events Rule.



Sources:
https://aws.amazon.com/premiumsupport/knowledge-center/cloudwatch-notification-scheduled-events/
https://docs.aws.amazon.com/health/latest/ug/cloudwatch-events-health.html

Using AWS Health API and CloudWatch Events for Monitoring AWS-Scheduled Deployments/Changes

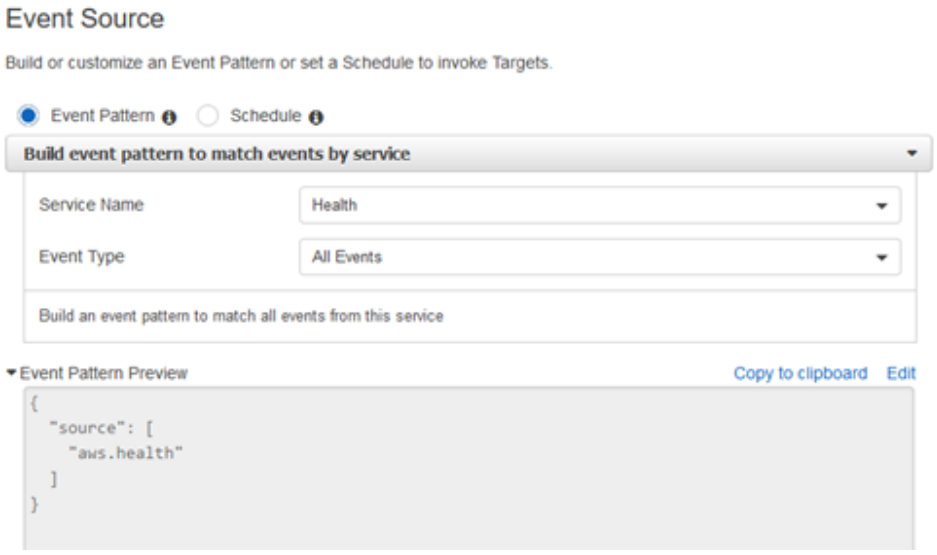
AWS Personal Health Dashboard provides you with alerts and remediation status when AWS is experiencing events that may impact your resources. These events may be scheduled or unscheduled. For example, scheduled events such as changes on your underlying EC2 hosts may shutdown or terminate your instances, or AWS RDS scheduled upgrades that may reboot your RDS instance.

DevOps Exam Notes:

You can monitor these AWS Health events using CloudWatch Events by calling the AWS Health API. Then, you can set a target to an SNS topic to inform you of that Event, or you can trigger a Lambda function to perform a custom action based on the Event.

Here are the steps on to set this up:

1. Go to CloudWatch > Events > Rules and create a rule that will detect AWS Health Events.
2. Select the Services in Event Type if you want to monitor a particular service, or choose All Events to be notified for all events.

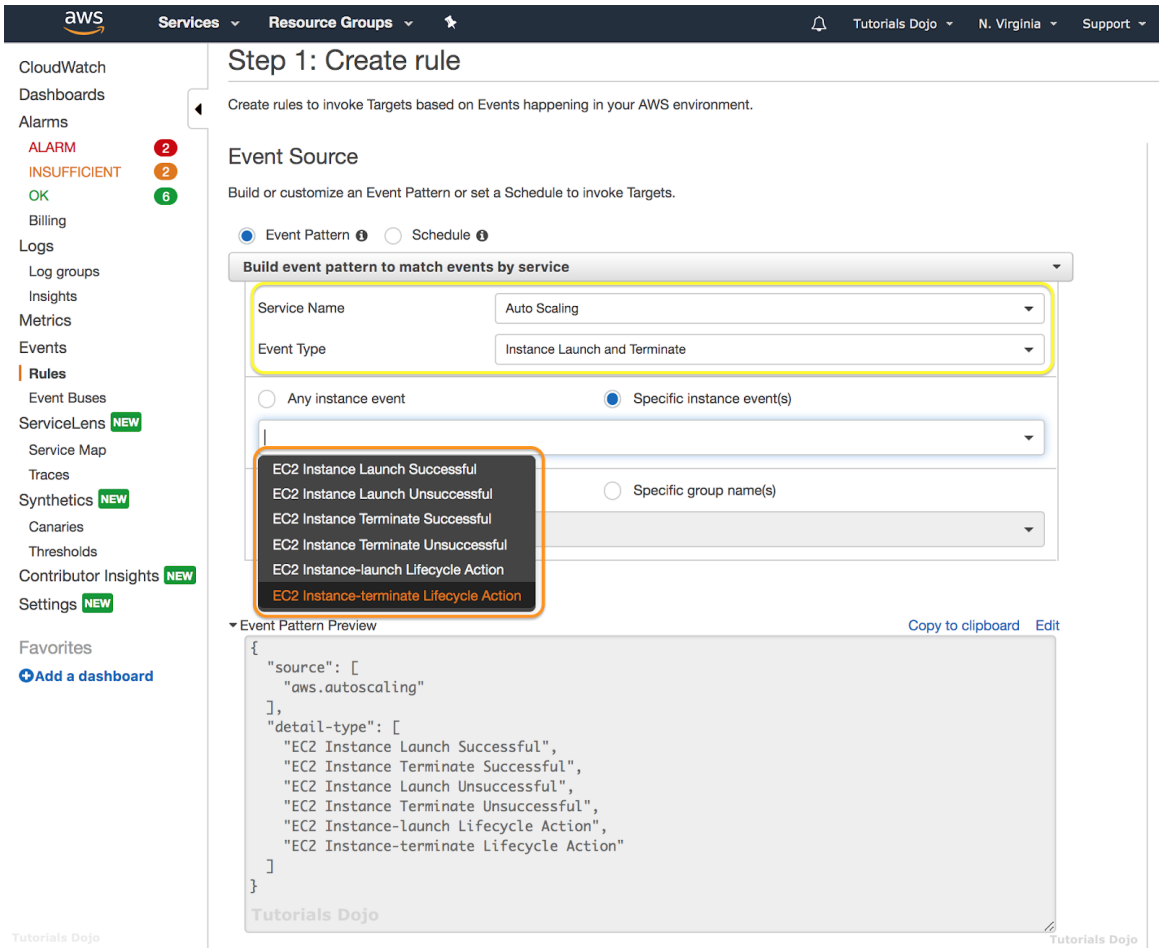


3. You can select a Target such as an SNS Topic if you want to get notified of the event, or Lambda function that can perform a custom action based on your requirements.
4. Click the Configure details to save and activate this CloudWatch Events rule.

Monitoring Amazon EC2 Auto Scaling Events

Auto Scaling provides fault tolerance, high availability, and cost management to your computing resources. It automatically scales in (terminates existing EC2 instances) if the demand is low and scales out (launch new EC2 instances) if the incoming traffic exceeds the specified threshold. AWS offers various ways of monitoring your fleet of Amazon EC2 instances as well responding to Auto Scaling events.

You can either use Amazon EventBridge or Amazon CloudWatch Events to track the Auto Scaling Events and run a corresponding custom action using AWS Lambda. Amazon EventBridge extends the capabilities of Amazon CloudWatch Events to integrate internal and external services. It allows you to track the changes of your Auto Scaling group in near real-time, including your custom applications, Software-as-a-Service (SaaS) partner apps, and other AWS services.



Amazon EventBridge or Amazon CloudWatch Events can be used to send events to the specified target when the following events occur:

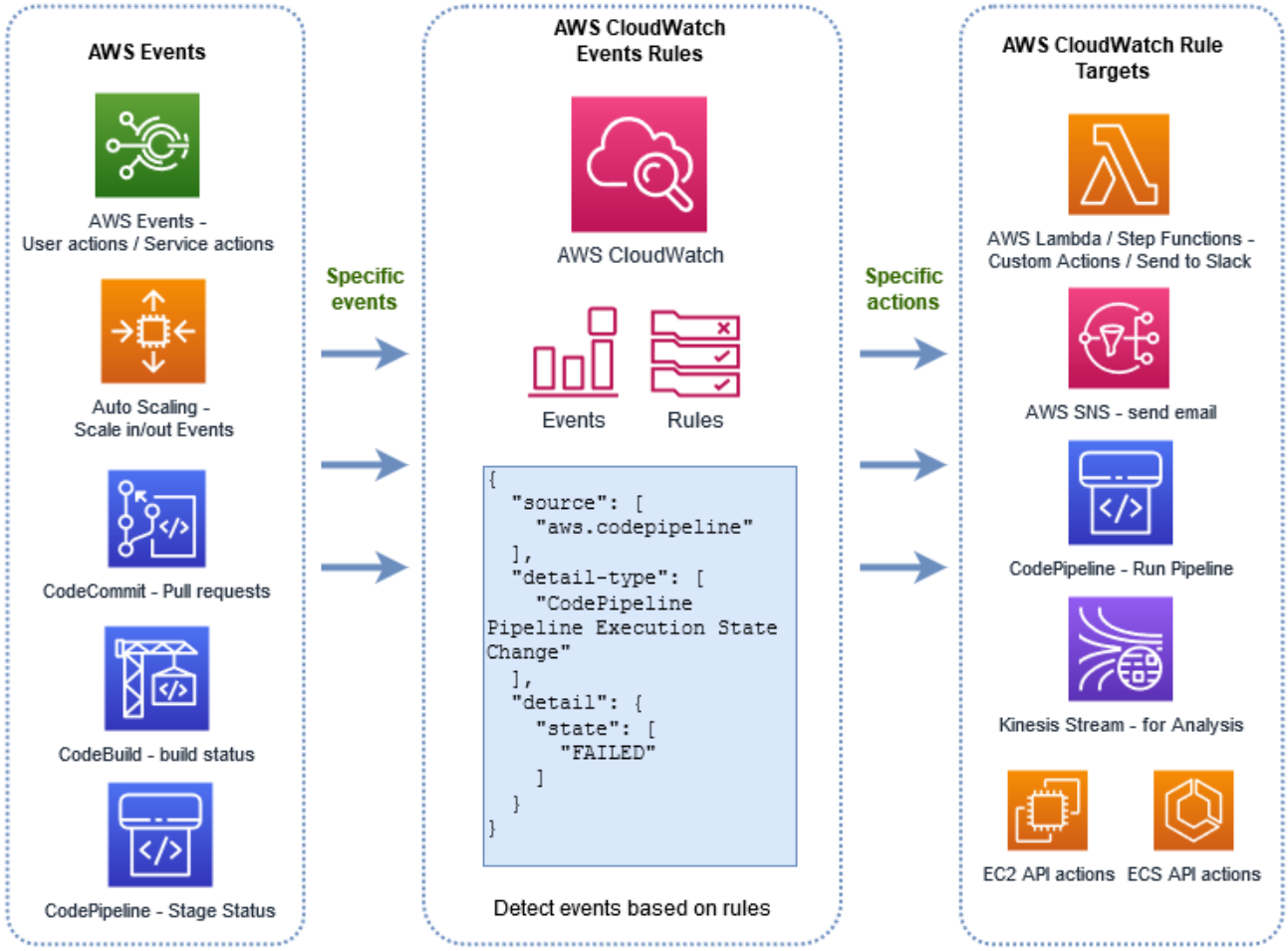
- EC2 Instance-launch Lifecycle Action
- EC2 Instance Launch Successful
- EC2 Instance Launch Unsuccessful
- EC2 Instance-terminate Lifecycle Action
- EC2 Instance Terminate Successful
- EC2 Instance Terminate Unsuccessful

The **`EC2 Instance-launch Lifecycle Action`** is a scale-out event in which the Amazon EC2 Auto Scaling moved an EC2 instance to a **`Pending:Wait`** state due to a lifecycle hook. Conversely, the **`EC2 Instance-terminate Lifecycle Action`** is a scale-in event in which EC2 Auto Scaling updates an instance to a **`Terminating:Wait`** state.

Source:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/cloud-watch-events.html

AWS CodePipeline Event Patterns

You can detect and respond to certain changes of your pipeline state in AWS CodePipeline using Amazon CloudWatch Events. You can use AWS CodePipeline (aws.codepipeline) as the event source of your CloudWatch Events rule and then associate an Amazon SNS topic to send notification or a Lambda function to execute a custom action. CloudWatch Events can automatically detect the state changes of your pipelines, stages, or actions in AWS CodePipeline which improves incident and event management of your CI/CD processes.



You can specify both the state and type of CodePipeline execution that you want to monitor. An item can be in a **STARTED**, **SUCCEEDED**, **RESUMED**, **FAILED**, **CANCELED** or **SUPERSEDED** state. Refer to the table below for the list of available detail types that you can use.

Entity	Detail Type
Pipeline	**CodePipeline Pipeline Execution State Change**
Stage	**CodePipeline Stage Execution State Change**
Action	**CodePipeline Action Execution State Change**

You can use this sample event pattern to capture failed **deploy** and **build** actions across all your pipelines in AWS CodePipeline:

```
{
  "source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Deploy", "Build"]
    }
  }
}
```

The sample event pattern below captures all **`rejected`** or **`failed`** approval actions across all the pipelines:

```
{
  "source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Approval"]
    }
  }
}
```

The following sample event pattern tracks all the events from the specified pipelines (**`TD-Pipeline-Manila`**, **`TD-Pipeline-Frankfurt`** and **`TD-Pipeline-New-York`**)

```
{
"source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Pipeline Execution State Change",
    "CodePipeline Action Execution State Change",
    "CodePipeline Stage Execution State Change"
  ],
  "detail": {
    "pipeline": ["TD-Pipeline-Manila",
                 "TD-Pipeline-Frankfurt",
                 "TD-Pipeline-New-York"]
  }
}
```

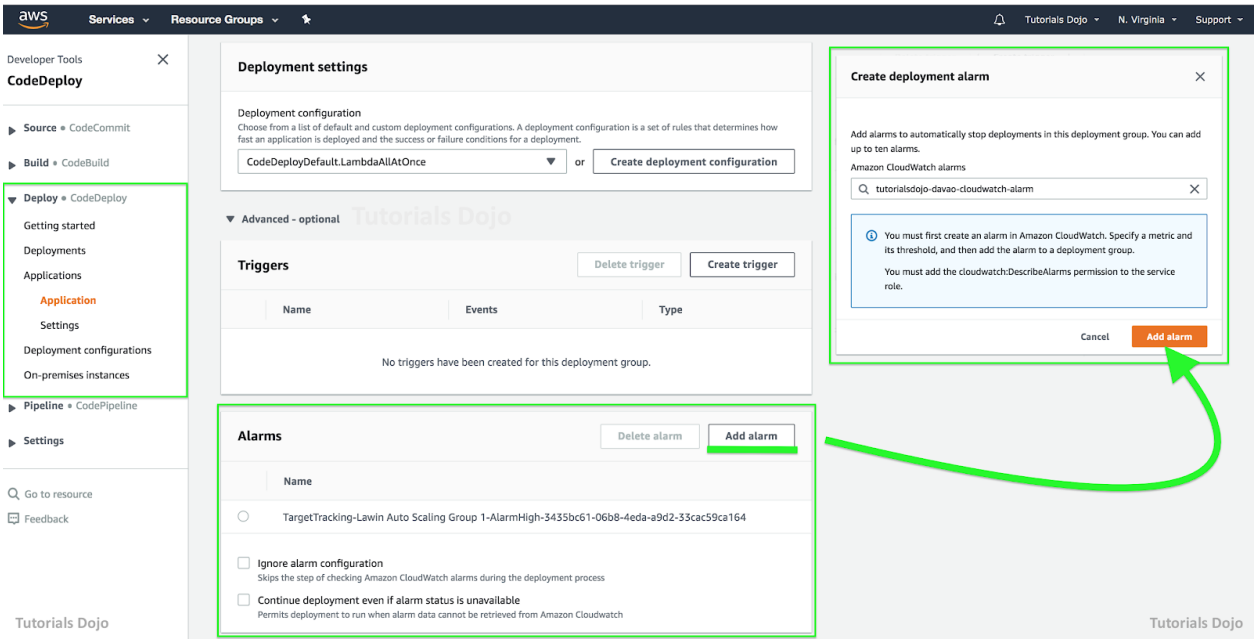
Source:
https://docs.aws.amazon.com/codepipeline/latest/userguide/detect-state-changes-cloudwatch-events.html

Monitoring Deployments in AWS CodeDeploy with CloudWatch Events and Alarms

Amazon CloudWatch Events helps you detect any changes in the state of an instance or a deployment in AWS CodeDeploy. You can also send notifications, collect state information, rectify issues, initiate events, or execute other actions using CloudWatch Alarms. This type of monitoring is useful if you want to be notified via Slack (or in other channels) whenever your deployments fail or push deployment data to a Kinesis stream for real-time status monitoring. If you integrated Amazon CloudWatch Events in your AWS CodeDeploy operations, you can specify the following as targets to monitor your deployments:

- AWS Lambda functions
- Kinesis streams
- Amazon SQS queues
- Built-in targets (CloudWatch alarm actions)
- Amazon SNS topics

Integrating AWS CodeDeploy and CloudWatch Alarms provides you an automated way to roll back your release when your deployment fails or if certain thresholds are not met. You can easily track the minimum number of healthy instances (`MinimumHealthyHosts`) that should be available at any time during the deployment. The `HOST_COUNT` or `FLEET_PERCENT` deployment configuration parameters can also be utilized to monitor the absolute number or just the relative percentage of healthy hosts respectively.



Source:
https://docs.aws.amazon.com/codedeploy/latest/userguide/monitoring-create-alarms.html

    - IAM roles
- Restrict access by only allowing trusted hosts or networks to access ports on your instance.
- A **security group** acts as a virtual firewall that controls the traffic for one or more instances.
    - Evaluates all the rules from all the security groups that are associated with an instance to decide whether to allow traffic or not.
    - By default, security groups allow **all outbound traffic**.
    - Security group rules are **always permissive**; you can't create rules that deny access.
    - Security groups are **stateful**
- You can replicate the network traffic from an EC2 instance within your Amazon VPC and forward that traffic to security and monitoring appliances for content inspection, threat monitoring, and troubleshooting.

Networking

- An **Elastic IP address** is a static IPv4 address designed for dynamic cloud computing. With it, you can mask the failure of an instance or software by rapidly remapping the address to another instance in your account.
- If you have not enabled auto-assign public IP address for your instance, you need to associate an Elastic IP address with your instance to enable communication with the internet.
- An elastic **network interface** is a logical networking component in a VPC that represents a virtual network card, which directs traffic to your instance
- Scale with **EC2 Scaling Groups** and distribute traffic among instances using **Elastic Load Balancer**.

Monitoring

- EC2 items to monitor
    - CPU utilization, Network utilization, Disk performance, Disk Reads/Writes using EC2 metrics
    - Memory utilization, disk swap utilization, disk space utilization, page file utilization, log collection using a monitoring agent/CloudWatch Logs
- Automated monitoring tools include:
    - System Status Checks - monitor the AWS systems required to use your instance to ensure they are working properly. These checks detect problems with your instance that require AWS involvement to repair.
    - Instance Status Checks - monitor the software and network configuration of your individual instance. These checks detect problems that require your involvement to repair.
    - Amazon CloudWatch Alarms - watch a single metric over a time period you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods.
    - Amazon CloudWatch Events - automate your AWS services and respond automatically to system events.

    - Amazon CloudWatch Logs - monitor, store, and access your log files from Amazon EC2 instances, AWS CloudTrail, or other sources.
- Monitor your EC2 instances with CloudWatch. By default, EC2 sends metric data to CloudWatch in 5-minute periods.
- You can also enable detailed monitoring to collect data in 1-minute periods.

Instance Metadata and User Data

- **Instance metadata** is data about your instance that you can use to configure or manage the running instance.
- View all categories of instance metadata from within a running instance at **http://169.254.169.254/latest/meta-data/**
- Retrieve user data from within a running instance at **http://169.254.169.254/latest/user-data**

    - When using the Fargate launch type with tasks within your cluster, ECS manages your cluster resources.
    - Enabling managed Amazon ECS cluster auto scaling allows ECS to manage the scale-in and scale-out actions of the Auto Scaling group.
- Services
    - ECS allows you to run and maintain a specified number of instances of a task definition simultaneously in a cluster.
    - In addition to maintaining the desired count of tasks in your service, you can optionally run your service behind a load balancer.
    - There are two deployment strategies in ECS:
        - **Rolling Update**
            - This involves the service scheduler replacing the current running version of the container with the latest version.

          Blue/Green Deployment with AWS CodeDeploy
            - This deployment type allows you to verify a new deployment of a service before sending production traffic to it.
            - The service must be configured to use either an Application Load Balancer or Network Load Balancer.
- Container Agent (AWS ECS Agent)
    - The **container agent** runs on each infrastructure resource within an ECS cluster.
    - It sends information about the resource's current running tasks and resource utilization to ECS, and starts and stops tasks whenever it receives a request from ECS.
    - Container agent is only supported on Amazon EC2 instances.

AWS Fargate

- You can use Fargate with ECS to run containers without having to manage servers or clusters of EC2 instances.
- You no longer have to provision, configure, or scale clusters of virtual machines to run containers.
- Fargate only supports container images hosted on Elastic Container Registry (ECR) or Docker Hub.

Task Definitions for Fargate Launch Type

- Fargate task definitions require that the network mode is set to *awsvpc*. The *awsvpc* network mode provides each task with its own elastic network interface.
- Fargate task definitions only support the *awslogs* log driver for the log configuration. This configures your Fargate tasks to send log information to Amazon CloudWatch Logs.
- Task storage is **ephemeral**. After a Fargate task stops, the storage is deleted.

Monitoring

- You can configure your container instances to send log information to CloudWatch Logs. This enables you to view different logs from your container instances in one convenient location.
- With CloudWatch Alarms, watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods.
- Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs.

AWS Management Tools

Amazon CloudWatch

- Monitoring tool for your AWS resources and applications.
- Display metrics and create alarms that watch the metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached.
- CloudWatch is basically a metrics repository. An AWS service, such as Amazon EC2, puts metrics into the repository and you retrieve statistics based on those metrics. If you put your own custom metrics into the repository, you can retrieve statistics on these metrics as well.
- CloudWatch does not aggregate data across regions. Therefore, metrics are completely separate between regions.
- **CloudWatch Concepts**
    - **Namespaces** - a container for CloudWatch metrics.
        - There is no default namespace.
        - The AWS namespaces use the following naming convention: AWS/*service*.
    - **Metrics** - represents a time-ordered set of data points that are published to CloudWatch.
        - Exists only in the region in which they are created.
        - By default, several services provide free metrics for resources. You can also enable **detailed monitoring**, or publish your own application metrics.
        - **Metric math** enables you to query multiple CloudWatch metrics and use math expressions to create new time series based on these metrics.
        - **Important note for EC2 metrics:** CloudWatch does not collect memory utilization and disk space usage metrics right from the get go. You need to install CloudWatch Agent in your instances first to retrieve these metrics.
    - **Dimensions** - a name/value pair that uniquely identifies a metric.
        - You can assign up to 10 dimensions to a metric.
        - Whenever you add a unique dimension to one of your metrics, you are creating a new variation of that metric.
    - **Statistics** - metric data aggregations over specified periods of time.
        - Each statistic has a unit of measure. Metric data points that specify a unit of measure are aggregated separately.
        - You can specify a unit when you create a custom metric. If you do not specify a unit, CloudWatch uses *None* as the unit.
        - A *period* is the length of time associated with a specific CloudWatch statistic. The default value is 60 seconds.
        - CloudWatch aggregates statistics according to the period length that you specify when retrieving statistics.
        - For large datasets, you can insert a pre-aggregated dataset called a *statistic set*.

- **Alarms** - watches a single metric over a specified time period, and performs one or more specified actions, based on the value of the metric relative to a threshold over time.
    - You can create an alarm for monitoring CPU usage and load balancer latency, for managing instances, and for billing alarms.
    - When an alarm is on a dashboard, it turns red when it is in the *ALARM* state.
    - Alarms invoke actions for sustained state changes only.
    - Alarm States
        - **OK**—The metric or expression is within the defined threshold.
        - **ALARM**—The metric or expression is outside of the defined threshold.
        - **INSUFFICIENT_DATA**—The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state.
    - You can also monitor your estimated AWS charges by using Amazon CloudWatch Alarms. However, take note that you can only track the estimated AWS charges in CloudWatch and not the actual utilization of your resources. Remember that you can only set coverage targets for your reserved EC2 instances in AWS Budgets or Cost Explorer, but not in CloudWatch.
    - When you create an alarm, you specify three settings:
        - **Period** is the length of time to evaluate the metric or expression to create each individual data point for an alarm. It is expressed in seconds.
        - **Evaluation Period** is the number of the most recent periods, or data points, to evaluate when determining alarm state.
        - **Datapoints to Alarm** is the number of data points within the evaluation period that must be breaching to cause the alarm to go to the ALARM state. The breaching data points do not have to be consecutive, they just must all be within the last number of data points equal to **Evaluation Period**.

CloudWatch Dashboard

- Customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those spread across different regions.
- There is no limit on the number of CloudWatch dashboards you can create.
- All dashboards are **global**, not region-specific.
- You can add, remove, resize, move, edit or rename a graph. You can metrics manually in a graph.

CloudWatch Events

- Deliver near real-time stream of system events that describe changes in AWS resources.
- Events respond to these operational changes and take corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.
- Concepts
    - **Events** - indicates a change in your AWS environment.
    - **Targets** - processes events.
    - **Rules** - matches incoming events and routes them to targets for processing.

CloudWatch Logs

- Features

- Monitor logs from EC2 instances in real-time
- Monitor CloudTrail logged events
- By default, logs are kept indefinitely and never expire
- Archive log data
- Log Route 53 DNS queries

CloudWatch Agent

- Collect more logs and system-level metrics from EC2 instances and your on-premises servers.
- Needs to be installed.

Comparison of AWS Services

AWS CloudTrail vs Amazon CloudWatch

- CloudWatch is a monitoring service for AWS resources and applications. CloudTrail is a web service that records API activity in your AWS account. They are both useful monitoring tools in AWS.
- By default, CloudWatch offers free basic monitoring for your resources, such as EC2 instances, EBS volumes, and RDS DB instances. CloudTrail is also enabled by default when you create your AWS account.
- With CloudWatch, you can collect and track metrics, collect and monitor log files, and set alarms. CloudTrail, on the other hand, logs information on who made a request, the services used, the actions performed, parameters for the actions, and the response elements returned by the AWS service. CloudTrail Logs are then stored in an S3 bucket or a CloudWatch Logs log group that you specify.
- You can enable detailed monitoring from your AWS resources to send metric data to CloudWatch more frequently, with an additional cost.
- CloudTrail delivers one free copy of management event logs for each AWS region. Management events include management operations performed on resources in your AWS account, such as when a user logs in to your account. Logging data events are charged. Data events include resource operations performed on or within the resource itself, such as S3 object-level API activity or Lambda function execution activity.
- CloudTrail helps you ensure compliance and regulatory standards.
- CloudWatch Logs reports on application logs, while CloudTrail Logs provide you specific information on what occurred in your AWS account.
- CloudWatch Events is a near real time stream of system events describing changes to your AWS resources. CloudTrail focuses more on AWS API calls made in your AWS account.
- Typically, CloudTrail delivers an event within 15 minutes of the API call. CloudWatch delivers metric data in 5 minutes periods for basic monitoring and 1 minute periods for detailed monitoring. The CloudWatch Logs Agent will send log data every five seconds by default.