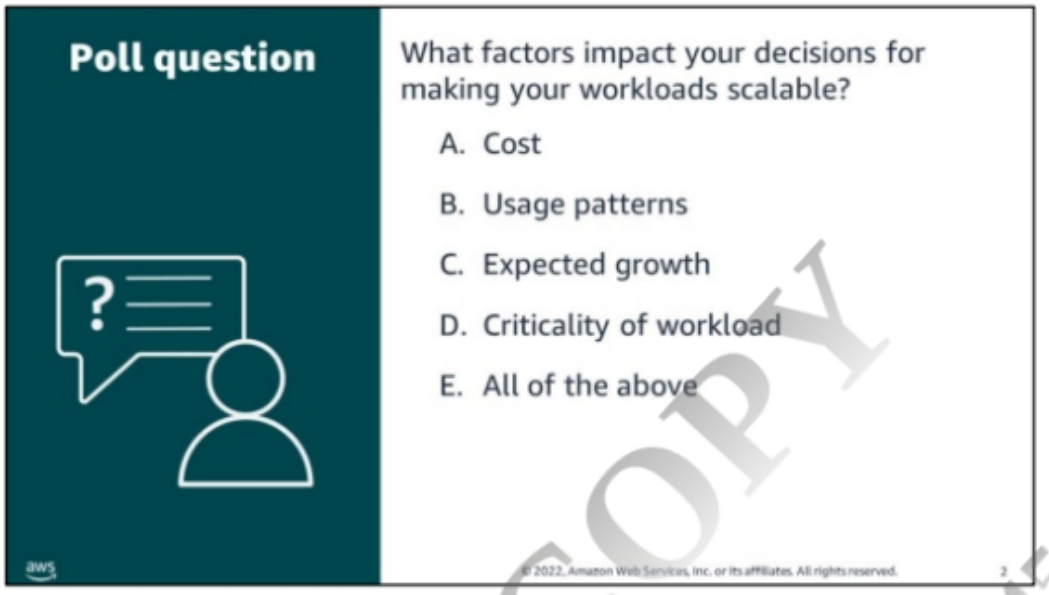
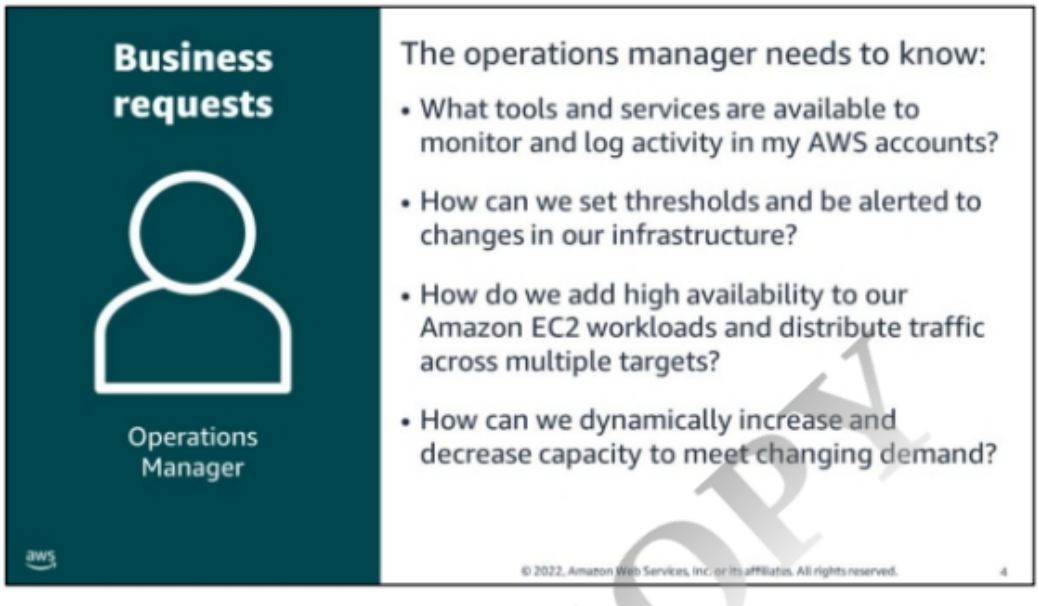
Printed by: 2d35e8483186bd2@placeholder.44518.edu. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



Module overview

- Business requests
- Monitoring
- Alarms and events
- Load balancing
- Auto scaling
- Present solutions
- Capstone check-in
- Knowledge check
- Lab 4: Configure high availability in your Amazon VPC

 3

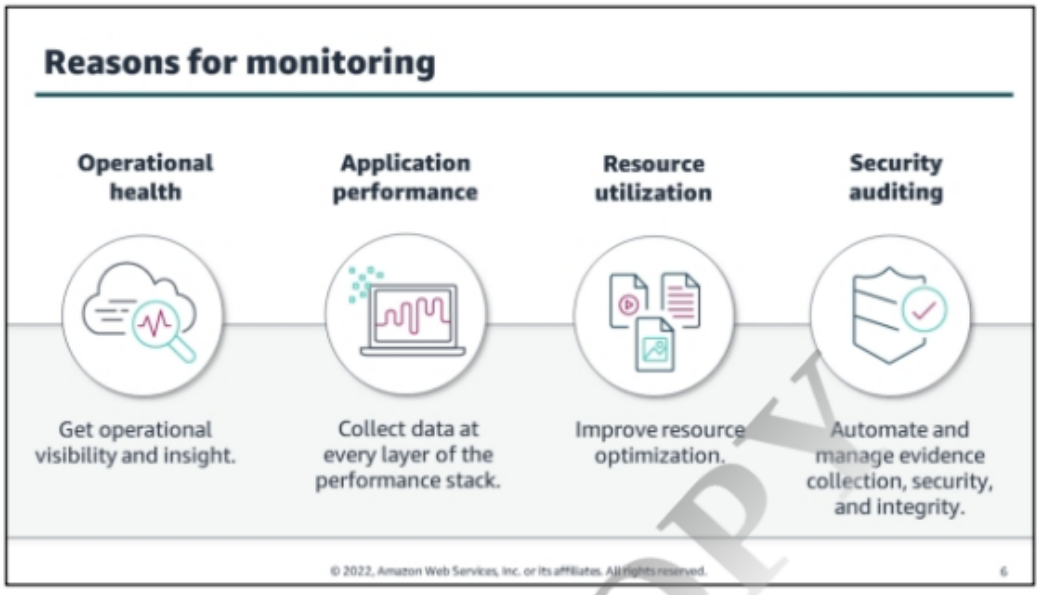


Imagine your operations manager meets with you to discuss how to monitor resources and scale operations in your AWS accounts. Here are some questions they are asking about monitoring and scaling.

At the end of this module, you meet with the operations manager and present some solutions.

The operations manager asks, "What tools and services are available to monitor and log activity in our AWS accounts?"

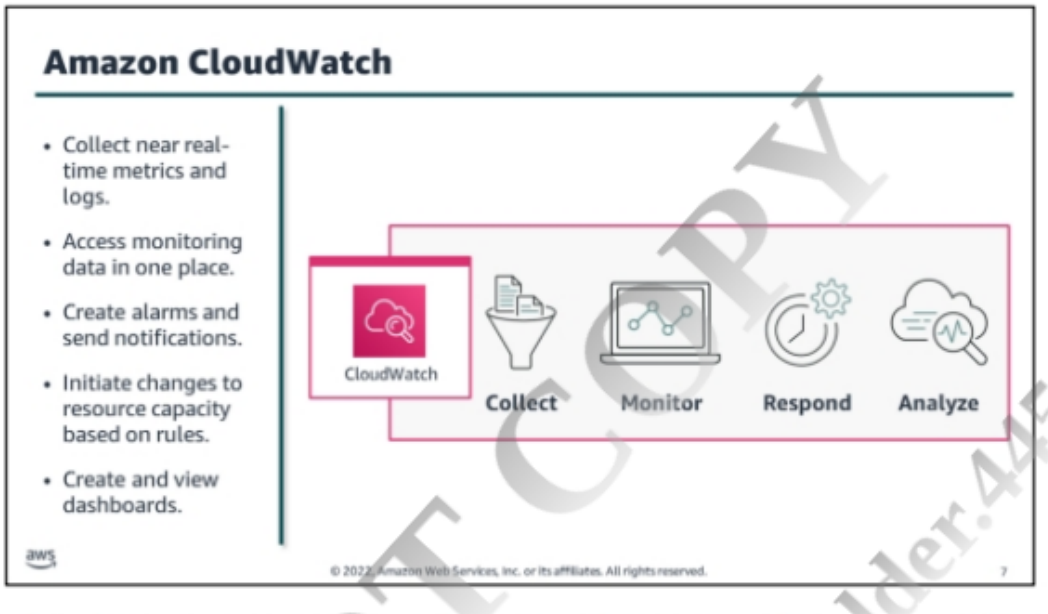
You and the operations team need to examine tools used in monitoring activity so that you can help build an easy-to-manage architecture.



Monitoring your environment is one of the most important things to think about when creating architecture. You will always need a way to keep track of how your resources are operating and performing. Monitoring gives you the information to answer the question: Does something need to change?

Here are a few points to remember:

- With monitoring, you gather information about resource utilization and application performance. Monitoring measures whether your infrastructure is satisfying demand. It helps you build an architecture that scales up for more demand and pulls back when there is less demand.
- This kind of scaling provides a better user experience for your customers and saves you money.
- Monitoring is very important for security. With parameters in place, you can see if and when users are accessing parts of your environment, and verify permissions.

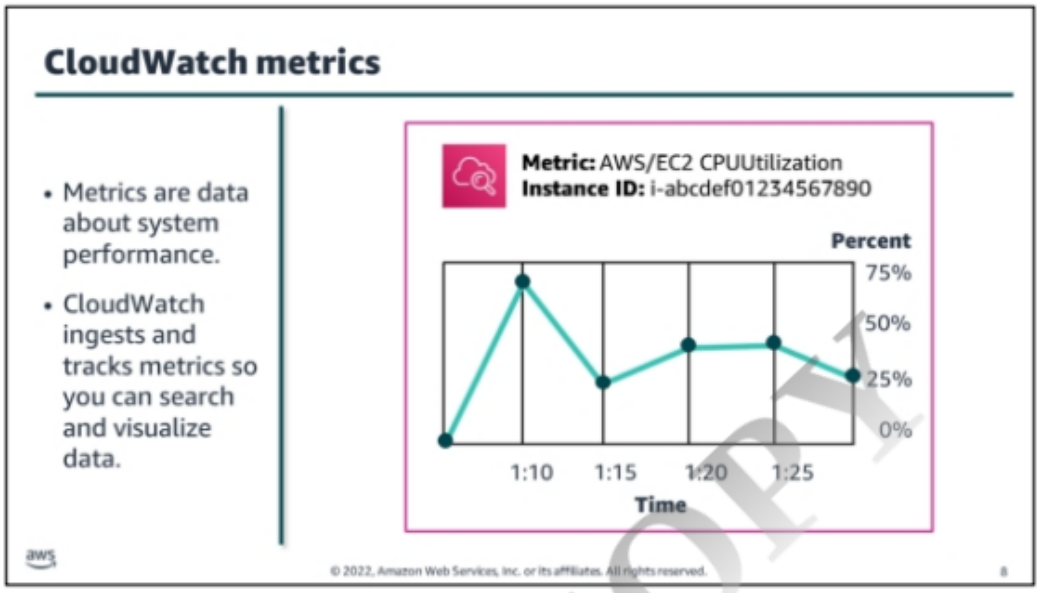


Amazon CloudWatch is an AWS service that provides a near real-time stream of system events. The events describe changes to your AWS resources. With CloudWatch, you can respond quickly to operational changes and take corrective action. CloudWatch alarms send notifications or automatically make changes to the resources you are monitoring based on rules that you define.

For example, you can monitor the CPU usage and disk reads and writes of your Amazon Elastic Compute Cloud (Amazon EC2) instances. This data can be used to determine whether you should launch additional instances to handle increased load. You can also use this data to stop underused instances to save money. Besides monitoring the built-in metrics that come with AWS, you can create and monitor custom metrics. With CloudWatch, you gain system-wide visibility into resource use, application performance, and operational health.

You can collect, access, and correlate this data in one place from across all of your AWS resources, applications, and services. CloudWatch also collects from on-premises servers. To optimize performance and resource use, CloudWatch provides automatic dashboards, data with one-second granularity, and up to 15 months of metrics storage and retention.

For more information about CloudWatch, see "What is Amazon CloudWatch?" in the *Amazon CloudWatch* User Guide (https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html).



Metrics are data about the performance of your systems. By default, many services provide you with metrics for resources, such as Amazon EC2 instances, Amazon Elastic Block Store (Amazon EBS) volumes, and Amazon Relational Database Service (Amazon RDS) DB instances. CloudWatch stores data about a metric as a series of data points. Each data point has an associated timestamp.

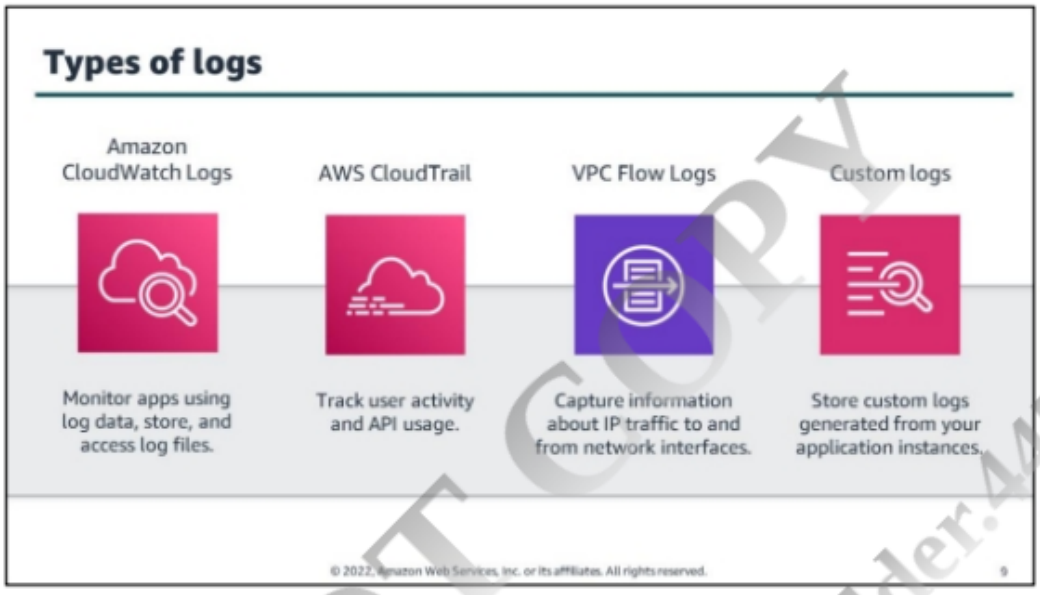
You can publish your own metrics to CloudWatch. Use the AWS Management Console to view statistical graphs of your published metrics.

You can also turn on detailed monitoring for some resources, such as your Amazon EC2 instances, or publish your own application metrics. Amazon CloudWatch can load all of the metrics in your account (both AWS resource metrics and application metrics that you provide) for search, graphing, and alarms. Metric data is kept for 15 months, so you can view both up-to-the-minute data and historical data.

Components of a CloudWatch metric include the following:

- A *namespace* is a container for CloudWatch metrics. Metrics in different namespaces are isolated from each other so that metrics from different applications are not mistakenly aggregated into the same statistics. You can specify a namespace name when you create a metric. The AWS namespaces use the following naming convention: AWS/*service*. In the example, the namespace is AWS/EC2.
- A *metric* represents a time-ordered set of data points that are published to CloudWatch. Think of a metric as a variable to monitor, and the data points represent the values of that variable over time. Each metric data point must be associated with a *timestamp*. If you do not provide a timestamp, CloudWatch creates a timestamp for you based on the time the data point was received.
- A *dimension* is a name-value pair that uniquely identifies a metric. You can assign up to 10 dimensions to a metric. Every metric has specific characteristics that describe it. You can think of dimensions as categories for those characteristics. In the example, the dimension is the instance ID.

To graph metrics in the console, you can use CloudWatch Metrics Insights, a high-performance structured query language (SQL) query engine. Use it to identify real-time trends and patterns within all of your metrics.

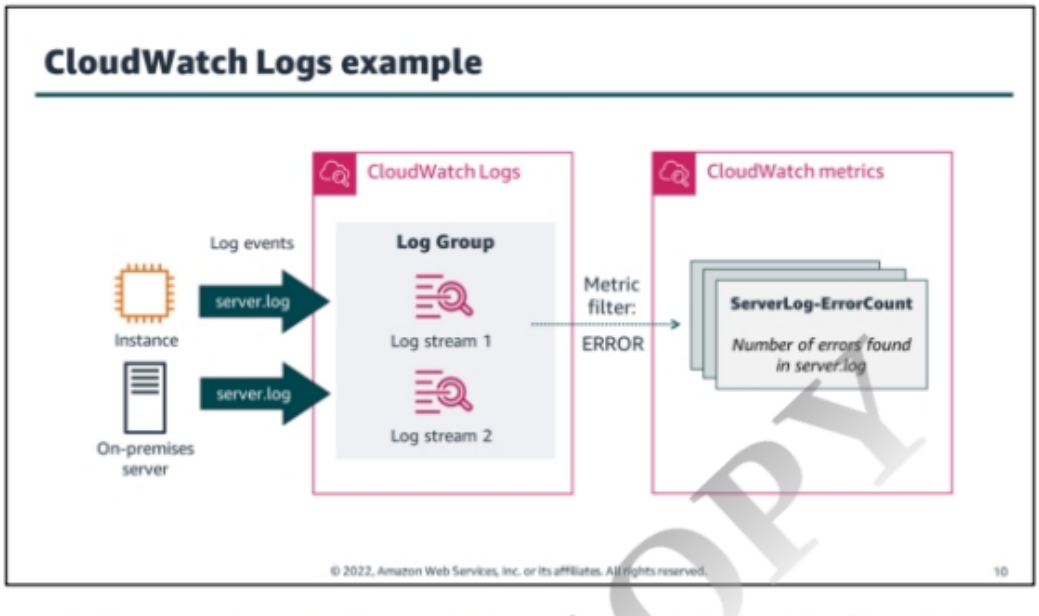


Plan for logging as you build. Review the following services to understand how they support logs:

- **Amazon CloudWatch Logs** monitor, store, and access your log files from Amazon EC2 instances, AWS CloudTrail, Amazon Route 53, and other resources.
- **AWS CloudTrail** provides event history of your account activity, including actions taken through the console, AWS SDK, command line interface (CLI), and AWS services. This event history simplifies security analysis, resource change tracking, and troubleshooting. CloudTrail facilitates governance, compliance, and operational and risk auditing. With CloudTrail, you can log, continuously monitor, and retain account activity related to actions across your AWS infrastructure.
- **VPC Flow Logs** captures information about the IP traffic going to and from network interfaces in your virtual private cloud (VPC).
- **Load balancing** provides access logs that capture detailed information about requests sent to your load balancer. You can use **custom logs** from your applications.

For more information about logging and events, see "Logging and Events" in the *AWS Technical Guide* (https://docs.aws.amazon.com/whitepapers/latest/aws-security-incident-response-guide/logging-and-events.html).

For more information about application log files, see "Store and Monitor OS & Application Log Files with Amazon CloudWatch" on the *AWS News Blog* (https://aws.amazon.com/blogs/aws/cloudwatch-log-service/).



In the example, there are two identical web servers that record data to the same log file, called server.log. One server is in Amazon EC2, while the other is a virtual machine on premises.

Both servers are able to publish events in the log file to a *log stream*. A log stream is a sequence of log events that share the same source. Each separate source of logs in CloudWatch Logs makes up a separate log stream.

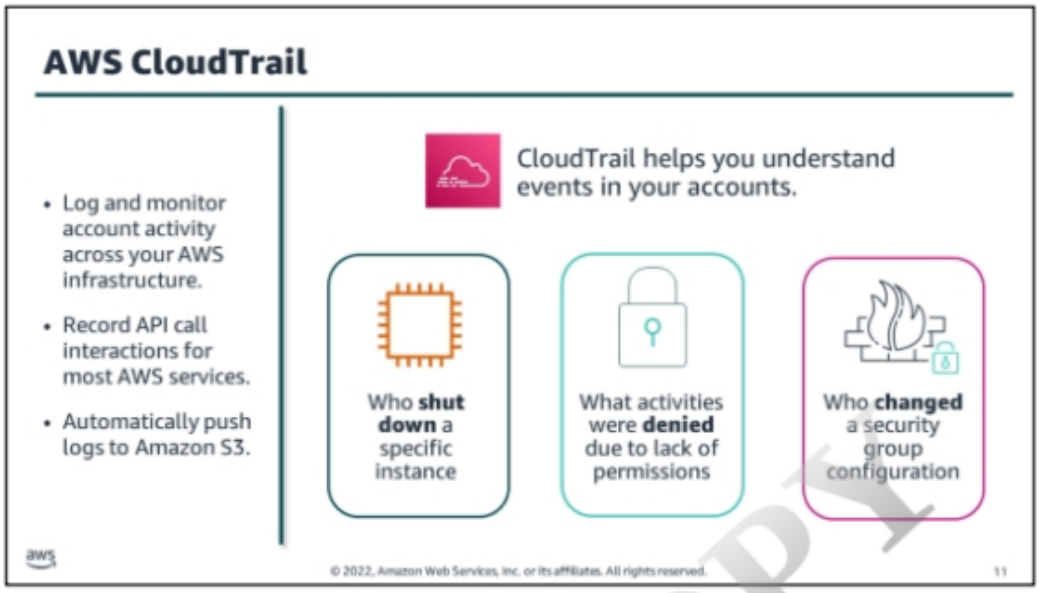
Multiple log streams can be collected in a single *log group*. A log group shares the same retention, monitoring, and access control settings across all streams. You can define log groups and specify which streams to put into each group. There is no limit on the number of log streams that can belong to one log group.

Once you create a log group, you can use *metric filters* to search for and match terms, phrases, or values in your log events. When a metric filter finds one of the terms, phrases, or values in your log events, you can increment the value of a CloudWatch metric. For example, you can count occurrences of a single term, such as the word "error," in a metric called *LogFile-ErrorCount*.

Metrics can be monitored and invoke alarms, which is covered later in this module.

For more information about CloudWatch log functionality, see "Working with log groups and log streams" in the *Amazon CloudWatch Logs User Guide* (https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/Working-with-log-groups-and-streams.html).

For more information about collecting metrics and logs, see "Collecting metrics and logs from Amazon EC2 instances and on-premises servers with the CloudWatch agent" in the *Amazon CloudWatch User Guide* (https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/Install-CloudWatch-Agent.html).



AWS CloudTrail provides insight into who did what and when by tracking user activity and API usage. With CloudTrail, you can get a history of AWS API calls in your account, including those made through the console, AWS SDKs, CLI, and higher-level AWS services. CloudTrail records the AWS API calls and delivers the log files to you. The information includes the source IP address and identity of the API caller, the time of the call, the request parameters, and the response elements returned by the AWS service.

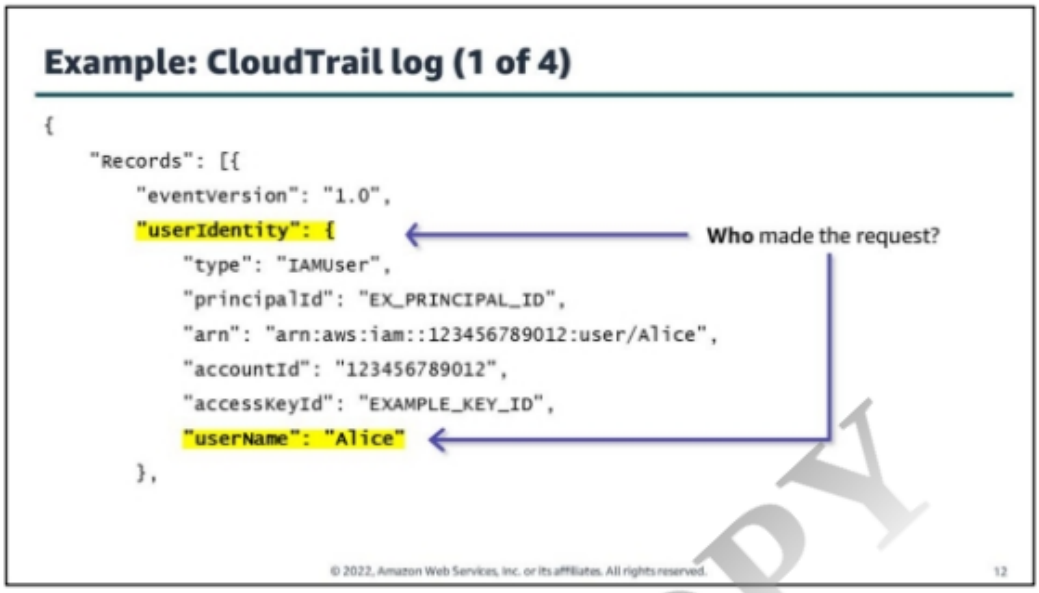
The AWS API call history produced by CloudTrail facilitates security analysis, tracking of resource changes, and compliance auditing.

You turn on CloudTrail on a per-Region basis. If you use multiple Regions, you can choose where log files are delivered for each Region. CloudTrail saves the logs in your designated Amazon Simple Storage Service (Amazon S3) bucket. For example, you can have a separate Amazon S3 bucket for each Region or you can aggregate log files from all Regions into a single Amazon S3 bucket.

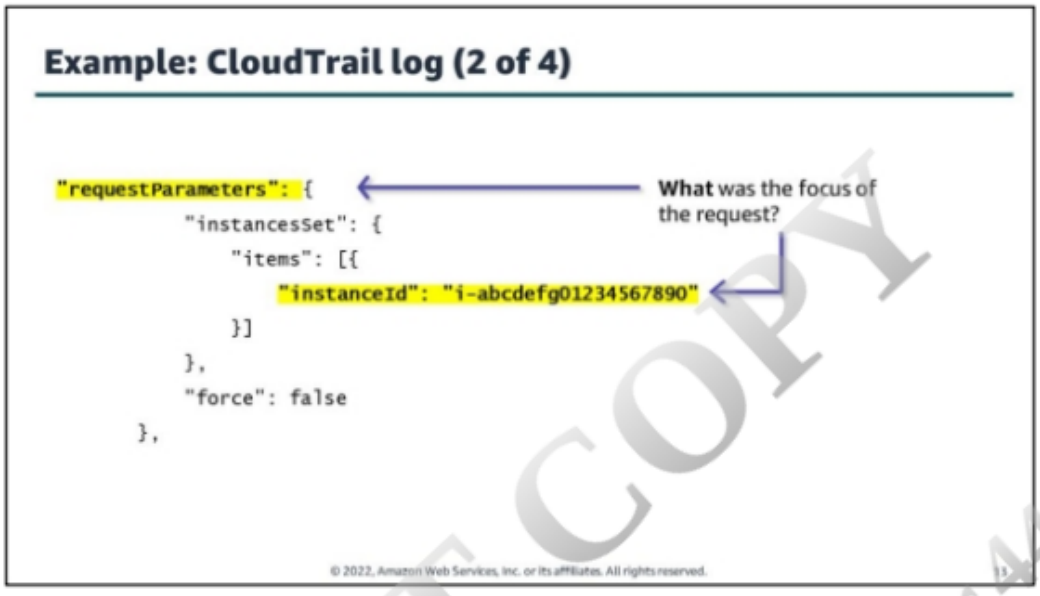
CloudTrail helps you answer questions requiring detailed analysis. Store your CloudTrail API usage logs in an Amazon S3 bucket. You can analyze those logs later to answer compelling questions, such as the following:

- Why was a long-running instance terminated and who terminated it? (*Organizational traceability and accountability*)
- Who changed a security group configuration? (*Accountability and security auditing*)
- What activities were denied due to lack of permissions? (*Potential internal or external attack against the network*)

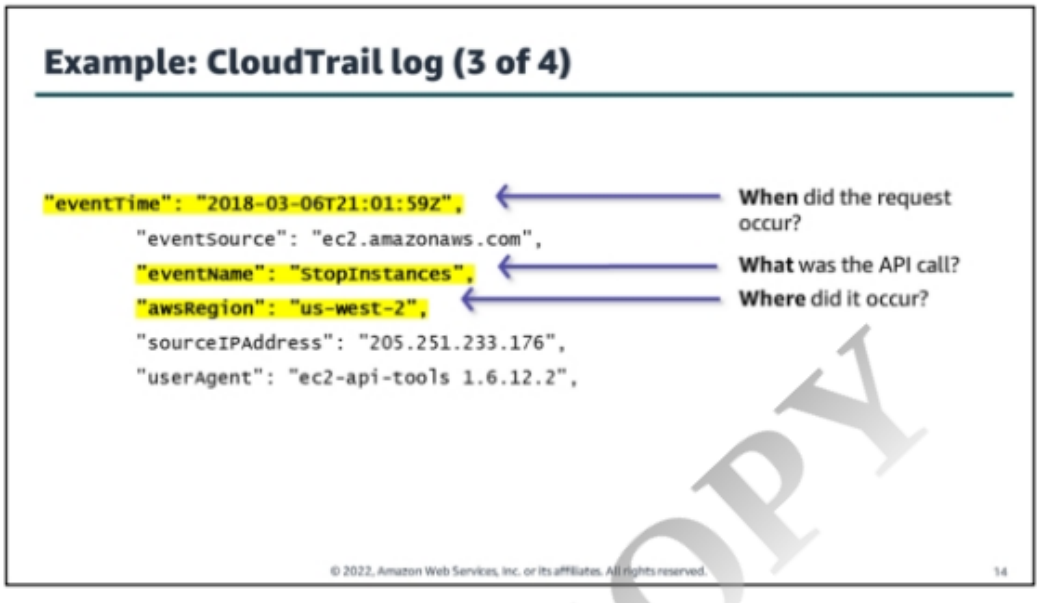
For more information about CloudTrail supported services and integrations, see "CloudTrail supported services and integrations" in the *AWS CloudTrail User Guide* (http://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-supported-services.html).



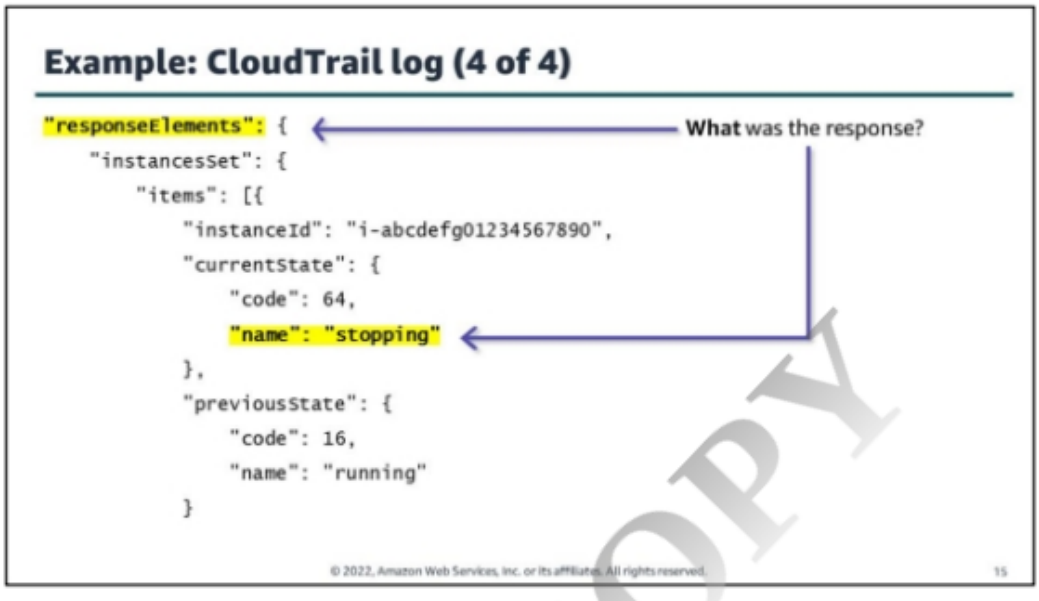
This example log file shows that an AWS Identity and Access Management (IAM) user named Alice called the EC2 StopInstances action by using the ec2-stop-instances command in AWS CLI. In this section of the log, you get information about who made the request.



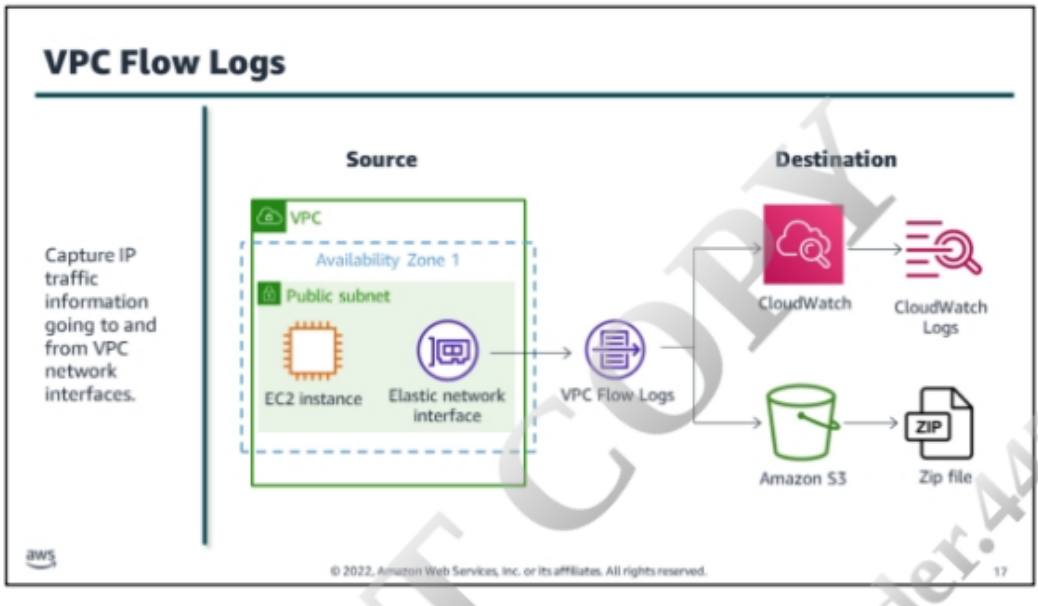
In this section of the script, you get information about the focus of the request. In this case, the focus of the request was an instance, and you can see the instance ID: i-abcdefg01234567890.



In this section of the script, you can find when the API call occurred, what the API call was (StopInstances), and in what Region it occurred.



In this section of the script, you get information about the response. In this case, the instance was stopped.



VPC Flow Logs capture IP traffic information to and from VPC network interfaces.

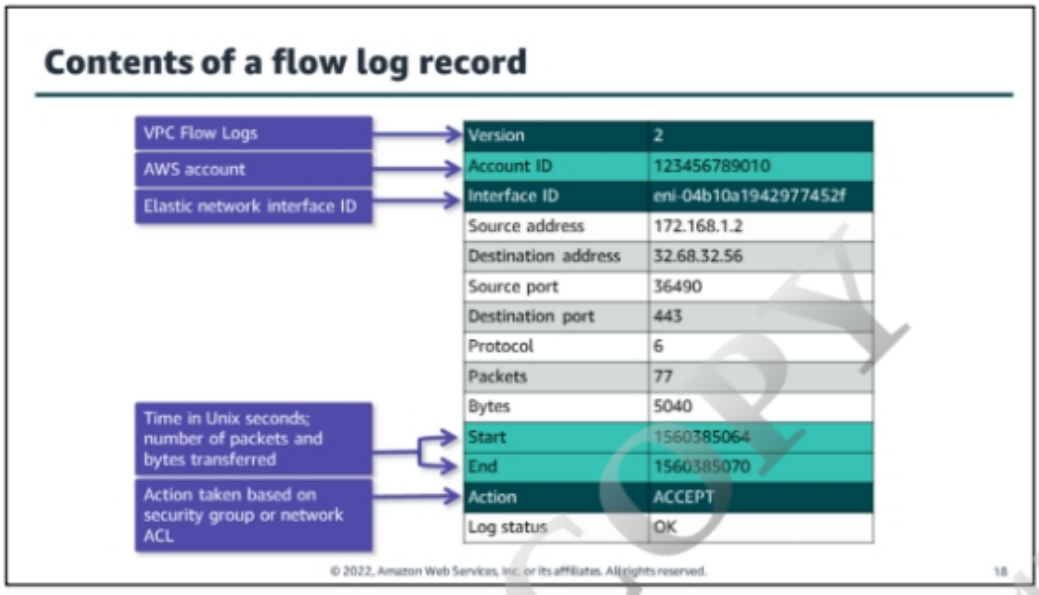
- They can be configured to record traffic per VPC, subnet, or network interface.
- You can view information about your flow logs in the Amazon EC2 and Amazon VPC consoles by choosing the *Flow Logs* tab for a specific resource.
- Flow logs are turned off by default. You need to opt in to collect flow log data.

Flow logs are published to either Amazon S3 buckets or CloudWatch log groups. Data is collected outside the path of your network traffic, and therefore does not affect network throughput or latency. Flow logs can help you perform several tasks, such as the following:

- Diagnose overly restrictive security group rules.
- Monitor the traffic that is reaching your instance.
- Determine the direction of the traffic to and from the network interfaces.

For more information about flow logs, see "Logging IP traffic using VPC Flow Logs" in the *Amazon VPC User Guide* (https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html).

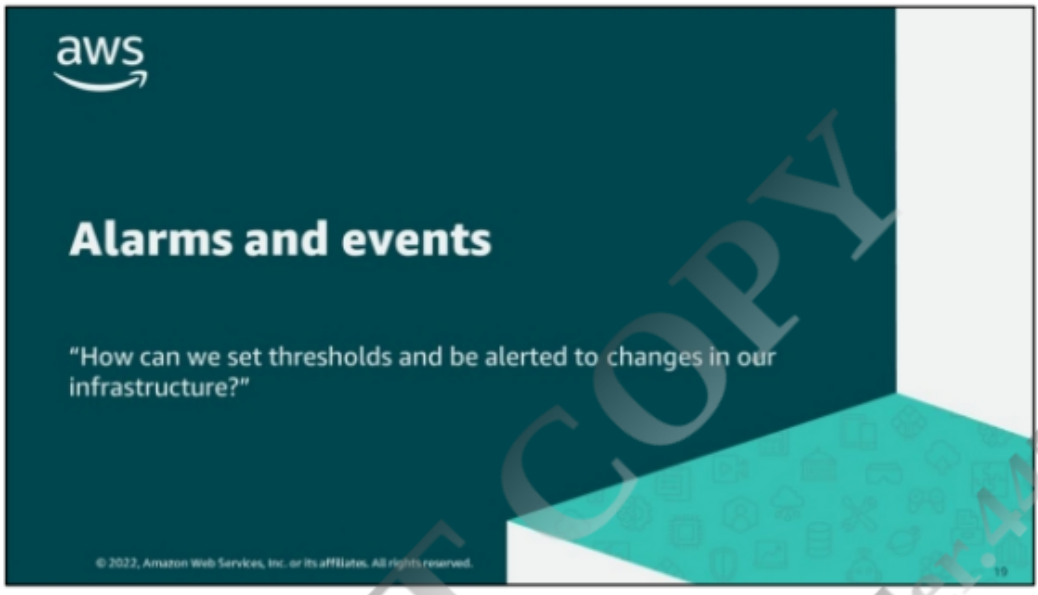
For more information about improving security by analyzing VPC flow logs, see "Improve security by analyzing VPC flow logs with Amazon CloudWatch Contributor Insights" in the *AWS Cloud Operations & Migrations Blog* (https://aws.amazon.com/blogs/mt/improve-security-by-analyzing-vpc-flow-logs-with-amazon-cloudwatch-contributor-insights/).



Each record captures a network IP traffic flow for a specific capture window and contains five different values, also known as a 5-tuple. A record includes different components of IP flow, such as the source, destination, and protocol. You can create alarms that will activate if certain types of traffic are detected, and metrics to help you to identify trends and patterns.

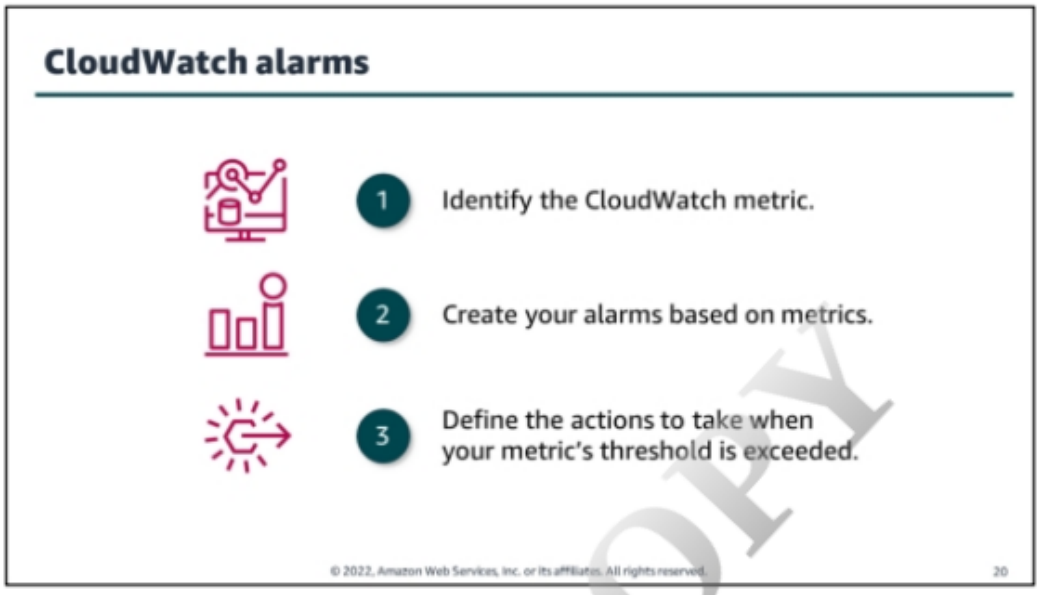
The information includes an ACCEPT or REJECT action, based on security group and network access control list (ACL) rules. It also includes source and destination IP addresses, ports, the Internet Assigned Numbers Authority (IANA) protocol number, and packet and byte counts (a time interval during which the flow was observed.)

Flow logs don't capture everything in your network. VPC logging does not include Domain Name System (DNS) traffic and Dynamic Host Configuration Protocol (DHCP) requests or responses. If you're running your own DNS server, you can log request resolution traffic. But many users rely on internal AWS DNS servers, and VPC Flow Logs will not capture activity between the servers and AWS DNS services. DHCP traffic is also not recorded.

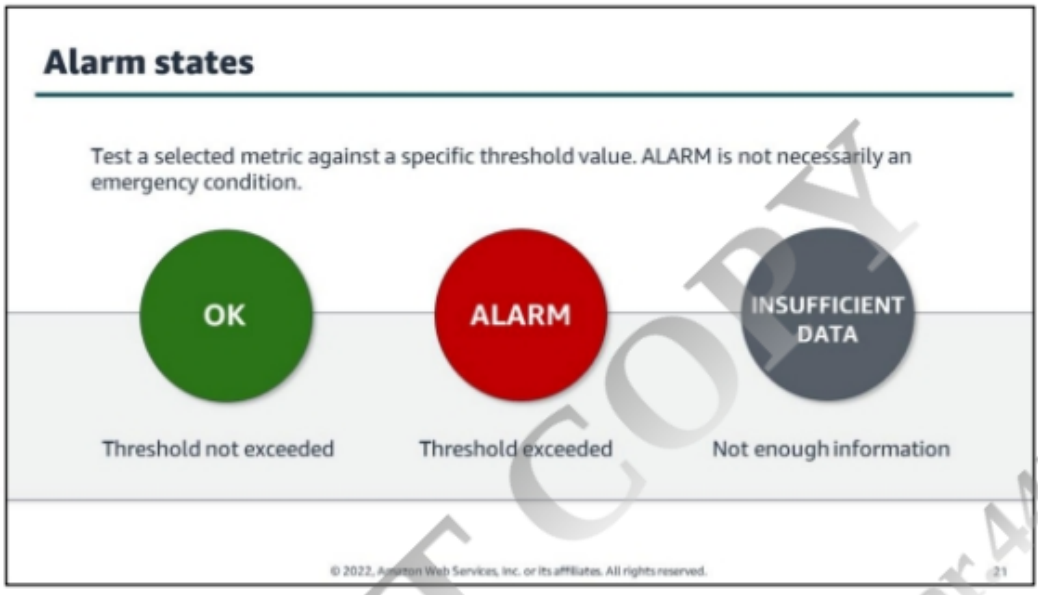


The operations manager asks, "How can we set thresholds and be alerted to changes in our infrastructure?"

The operations team needs to examine AWS services used to help automate actions based on events.



A metric alarm watches a single CloudWatch metric. The alarm performs one or more actions based on the value of the metric relative to a threshold over a number of time periods. The action can be an Amazon EC2 action, an auto scaling action, or a notification sent to an Amazon Simple Notification Service (Amazon SNS) topic.

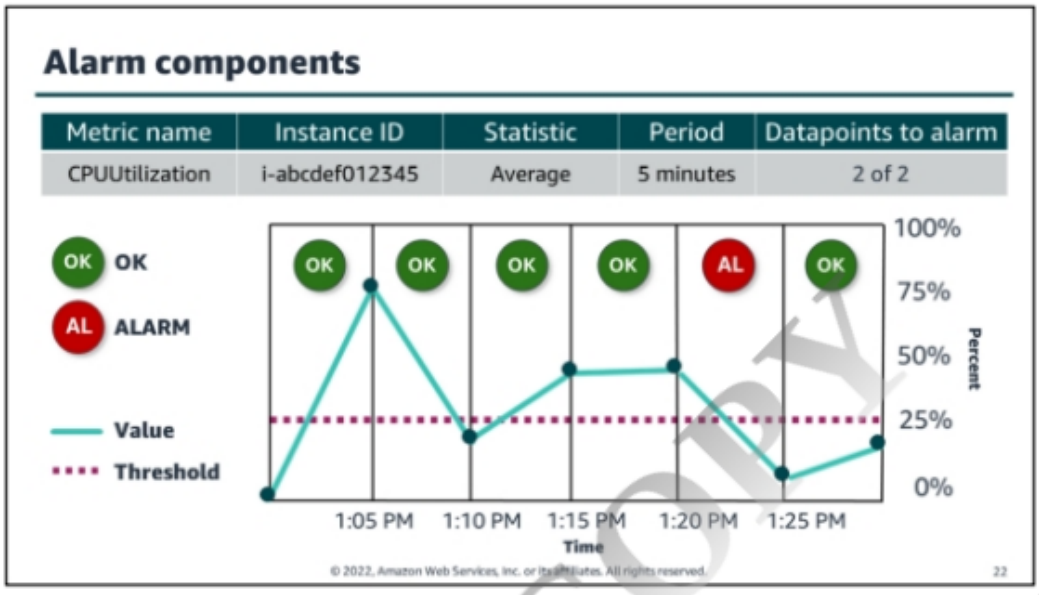


An alarm has three possible states:

- **OK** – The metric is within the defined threshold.
- **ALARM** – The metric is outside the defined threshold.
- **INSUFFICIENT_DATA** – The alarm has started, the metric is not available, or not enough data is available for the metric to determine the alarm state.

ALARM is only a name given to the state and does not necessarily signal an emergency condition requiring immediate attention. It means that the monitored metric is equal to, greater than, or less than a specified threshold value. You could, for example, define an alarm that tells you when your CPU utilization for a given EC2 instance is too high. You might process this notification programmatically to suspend a CPU-intensive job on the instance. You can also send a notification to take action to notify the application owner.

INSUFFICIENT_DATA can be returned when no data exists for a given metric. An example of this is the depth of an empty Amazon Simple Queue Service (Amazon SQS) queue. This can also be an indicator that something is wrong in your system.



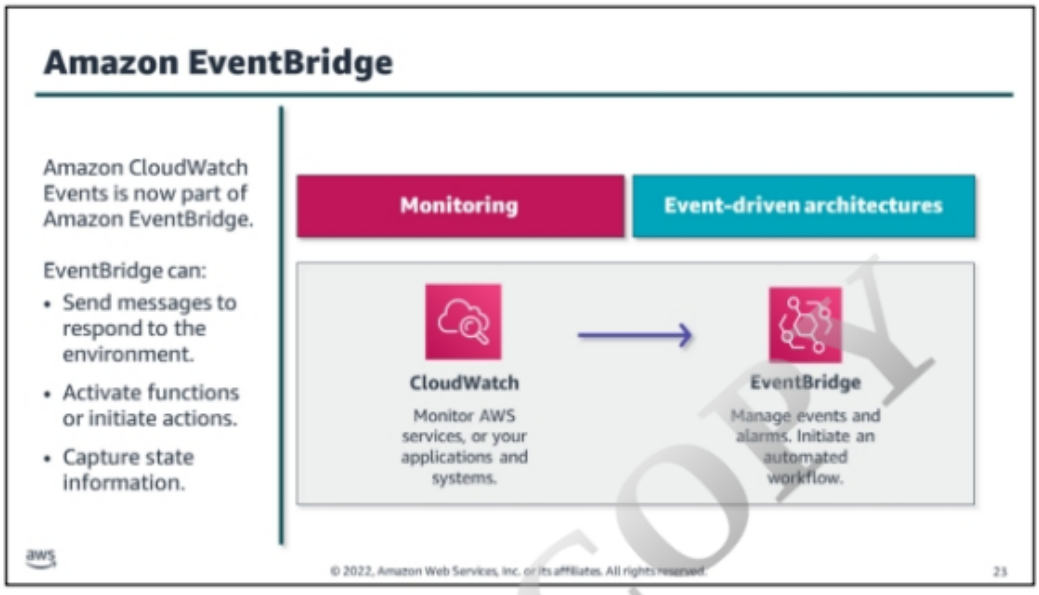
To create an alarm based on a metric math expression, choose one or more CloudWatch metrics to use in the expression. Then, specify the expression, threshold, and evaluation periods.

- *Statistics* are metric data aggregations over specified periods of time. For a metric alarm, it is evaluated on one statistic. Some of the most common statistics you can choose are sample count, sum, average, maximum, minimum, and percentile.
- *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. If you choose one minute as the period, the alarm evaluates the metric once per minute.
- *Evaluation Periods* is the number of the most recent periods, or data points, to evaluate when determining the alarm state. In the example table, this is the second number in the "2 of 2" value.
- *Datapoints to Alarm* is the number of data points within the Evaluation Periods that must be breaching to cause the alarm to go to the ALARM state. The breaching data points don't have to be consecutive, but they must all be within the last number of data points equal to the Evaluation Period.

In the example, the alarm threshold is set to 25 percent and the minimum breach is 2 periods. That is, the alarm state changes and it invokes its action only when the threshold is breached for two consecutive periods. You control this value. You can also set how the alarm treats missing data points. In some cases, you may want to build your alarm to go to the INSUFFICIENT_DATA state for missing data rather than an ALARM state.

In the metric graph, this happens with the third through fourth and fifth periods, and the alarm's state is set to ALARM at 1:20 PM. At period six, the value dips below the threshold and the state reverts to OK at 1:25 PM.

For more information about CloudWatch alarms, see "Using Amazon CloudWatch alarms" in the *Amazon CloudWatch User Guide* (https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/AlarmThatSendsEmail.html).



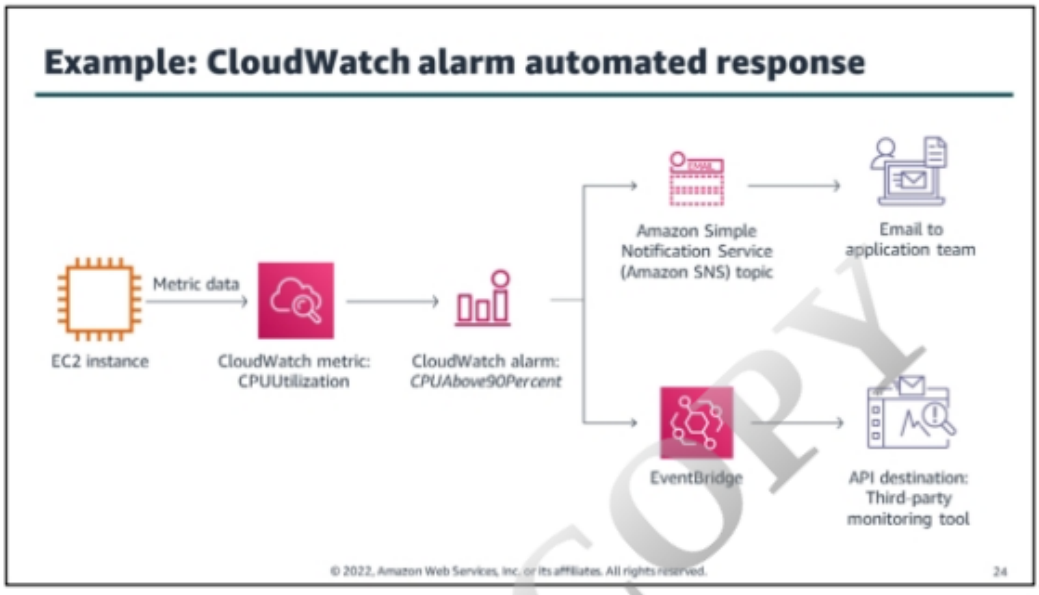
Amazon EventBridge removes the friction of writing *point-to-point* integrations. You can access changes in data that occur in both AWS and software as a service (SaaS) applications through a highly scalable, central stream of events.

EventBridge is the preferred way to manage your events captured in CloudWatch. CloudWatch Events and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console.

With EventBridge, you get a simple programming model where event publishers are decoupled from event subscribers. You can build loosely coupled, independently scaled, and highly reusable event-driven applications.

It's fully managed, so it handles everything from event ingestion and delivery to security, authorization, and error-handling, so you can build scalable, event-driven applications. EventBridge is serverless, so there is no infrastructure to manage and you only pay for the events you consume.

You only pay for events generated by your own applications or SaaS applications.



In the example, an EC2 instance reports the CPUUtilization metric data to CloudWatch. A custom alarm is created and configured, called "CPUAbove90Percent," so that you will know when the EC2 instance is being overused.

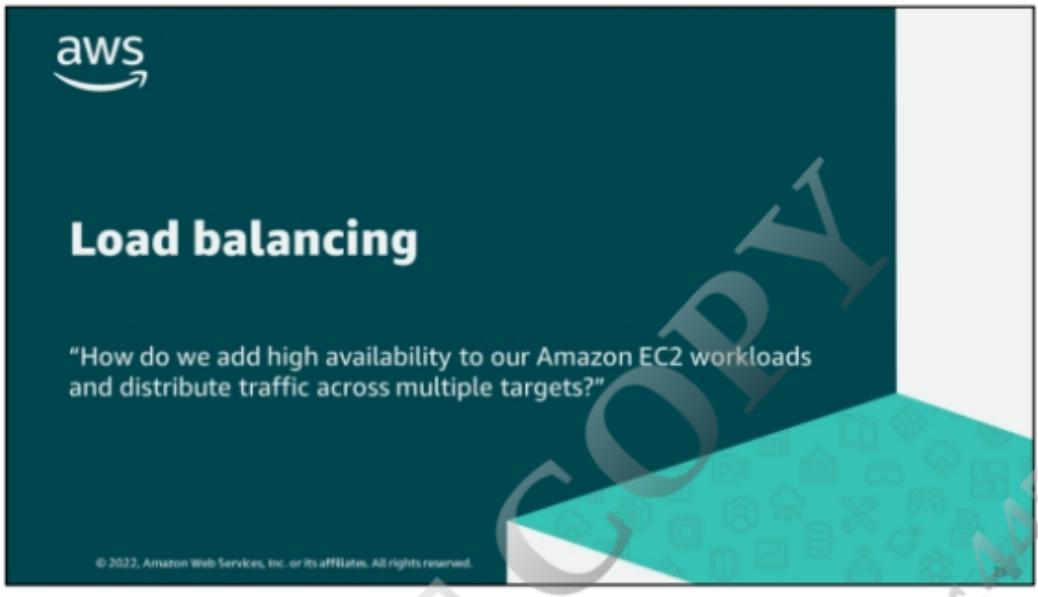
EventBridge rules are built to notify your support team when the CPUAbove90Percent alarm is in ALARM state, so that they can investigate and take action. EventBridge takes two actions: it sends an email to subscribed recipients using the Amazon Simple Notification Service (Amazon SNS) topic, and sends a rich notification to the operation team's third-party monitoring tool.

Think about how you could take this further.

- What other actions could you invoke using EventBridge that could automate a response to high CPU utilization on the EC2 instance?
- Are there tools in AWS services that can help you redirect traffic while you take action?
- How could you scale out to prevent an event caused by overutilized CPU?

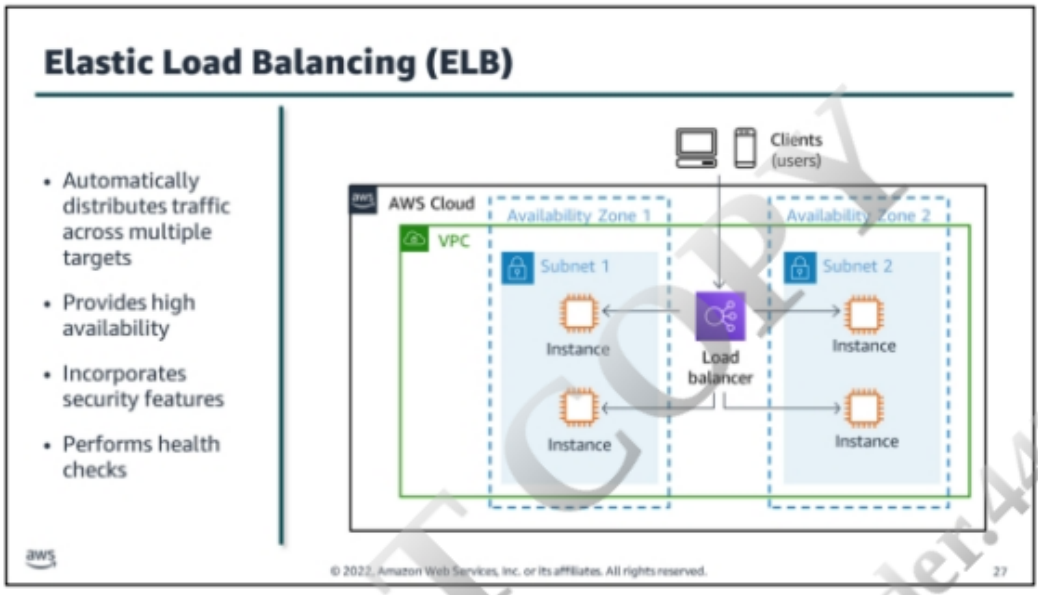
In this course, you learn how to build architectures that support your operations team to maintain a healthy environment.

For more information about EventBridge integrations, see "Amazon EventBridge Integrations" (https://aws.amazon.com/eventbridge/integrations/).



The operations manager asks, "How do we add high availability to our Amazon EC2 workloads and distribute traffic across multiple targets?"

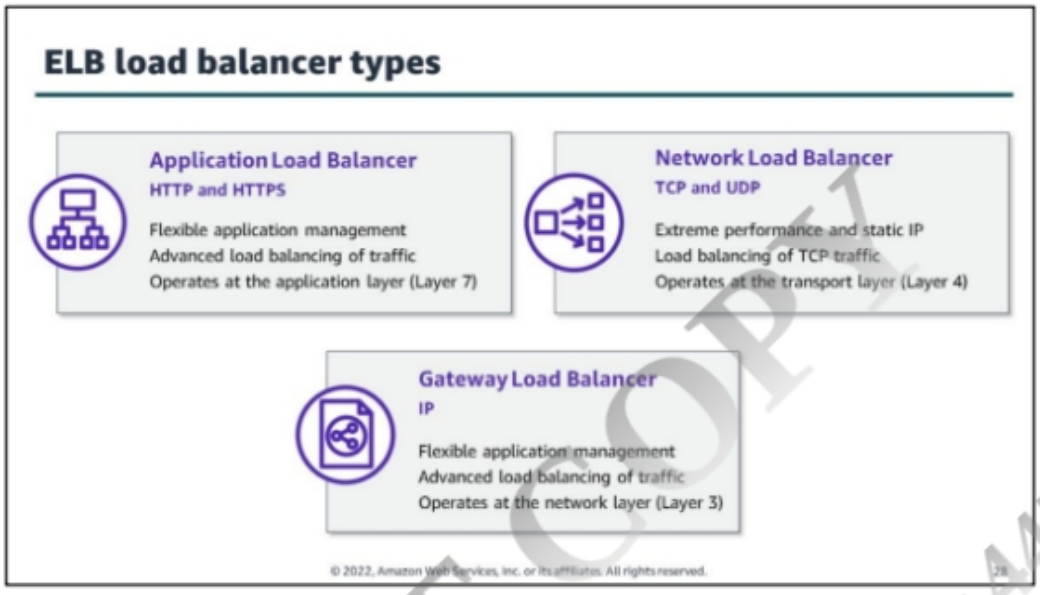
The team needs to examine the AWS services used in event-driven infrastructure, or applications, to monitor and manage events.



Elastic Load Balancing (ELB) makes up one of the most widely used AWS service categories. It has been adopted by organizations of all sizes, in all geographies, and across every industry. ELB load balancers are the only load balancers available on AWS that natively connect users to your EC2 instances, container deployments, and AWS Lambda functions. Some key feature sets include the following:

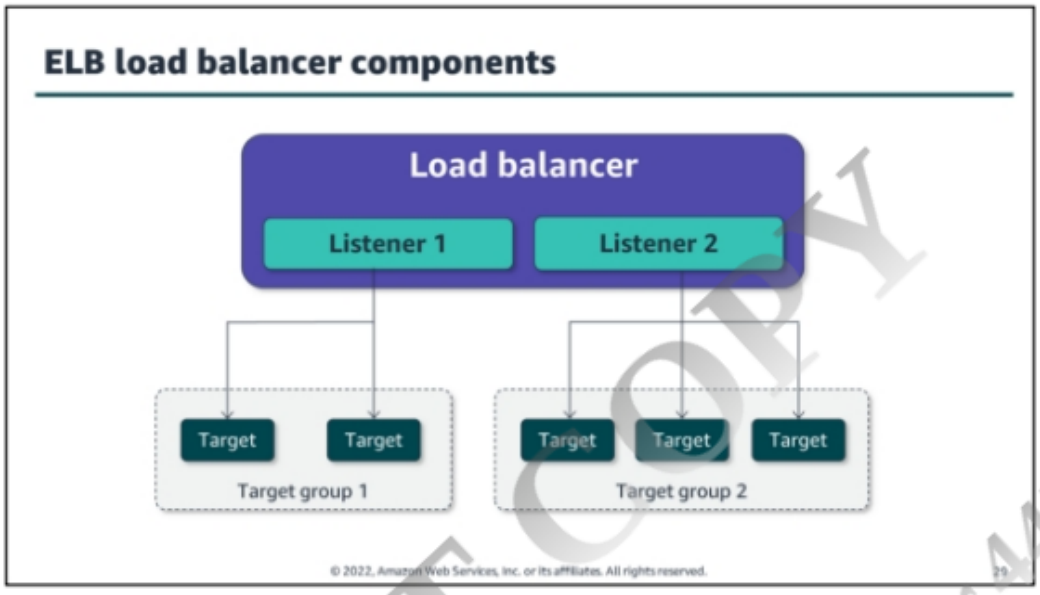
- **High availability –** ELB automatically distributes your traffic across multiple targets in a single Availability Zone or multiple Availability Zones. Examples of targets include EC2 instances, containers, and IP addresses.
- **Layer 4 or Layer 7 HTTP and HTTPS load balancing –** You can load balance your HTTP or HTTPS applications for Layer 7-specific features. Alternatively, you can use strict Layer 4 load balancing for applications that rely purely on the TCP.
- **Security features –** Use Amazon VPC to create and manage security groups associated with load balancers to provide additional networking and security options. You can also create an internal (non-internet-facing) load balancer.
- **Health checks –** ELB load balancers can detect unhealthy targets, stop sending traffic to them, and spread the load across the remaining healthy targets.
- **Monitoring operations –** To monitor real-time application performance, ELB integrates with CloudWatch metrics and provides request tracing.

In this example, the load balancer receives requests from desktop and mobile clients (users). There are two subnets in the same VPC, with two EC2 instances each. Each subnet is in a separate availability zone. All four EC2 instances are registered to the same load balancer and receive traffic.



Types of load balancers available for you to use include the following:

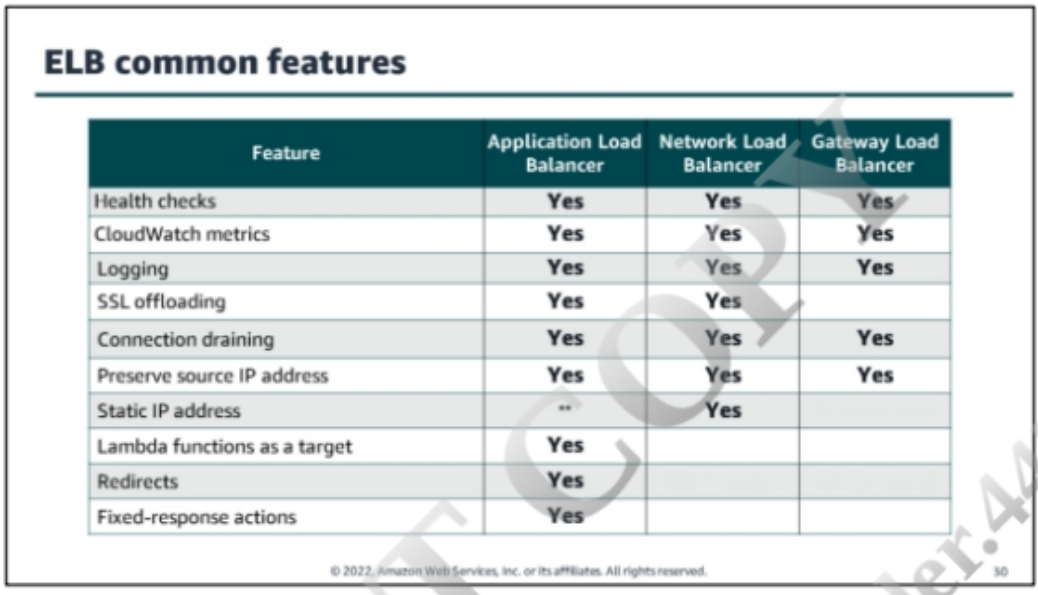
- **Application Load Balancer** – This load balancer functions at the application layer, the seventh layer of the Open Systems Interconnection (OSI) model. Application Load Balancer supports content-based routing, applications that run in containers, and open standard protocols (WebSocket and HTTP/2). This type of balancer is ideal for advanced load balancing of HTTP and HTTPS traffic.
- **Network Load Balancer** – This load balancer is designed to handle tens of millions of requests per second while maintaining high throughput at ultra low-latency. Network Load Balancer operates at the transport layer (Layer 4), routing connections to targets based on IP protocol data. Targets include EC2 instances, containers, and IP addresses. It is ideal for balancing TCP and User Diagram Protocol (UDP) traffic.
- **Gateway Load Balancer** – You can use this load balancer to deploy, scale, and manage your third-party virtual appliances. It provides one gateway for distributing traffic across multiple virtual appliances, and scales them up or down, based on demand. This distribution reduces potential points of failure in your network and increases availability. Gateway Load Balancer passes all Layer 3 traffic transparently through third-party virtual appliances. It is invisible to the source and destination.
- **Classic Load Balancer** – This is a previous-generation load balancer that is not discussed deeply in this course.



An ELB load balancer distributes your incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones to increase the availability of your application. Your load balancers are configured with *listeners*. You can have more than one listener per load balancer. Listeners have different functions for different types of load balancers. Your load balancer can forward requests to one or more target groups based on rules and settings that you define.

Each *target group* routes requests to one or more registered targets (for example, EC2 instances) using the specified protocol and port number. You can register a target with multiple target groups.

In the example, a load balancer has two listeners. Listener 1 has one target group called target group 1, with two registered targets. A second listener in the same load balancer has one target group with three different registered targets.



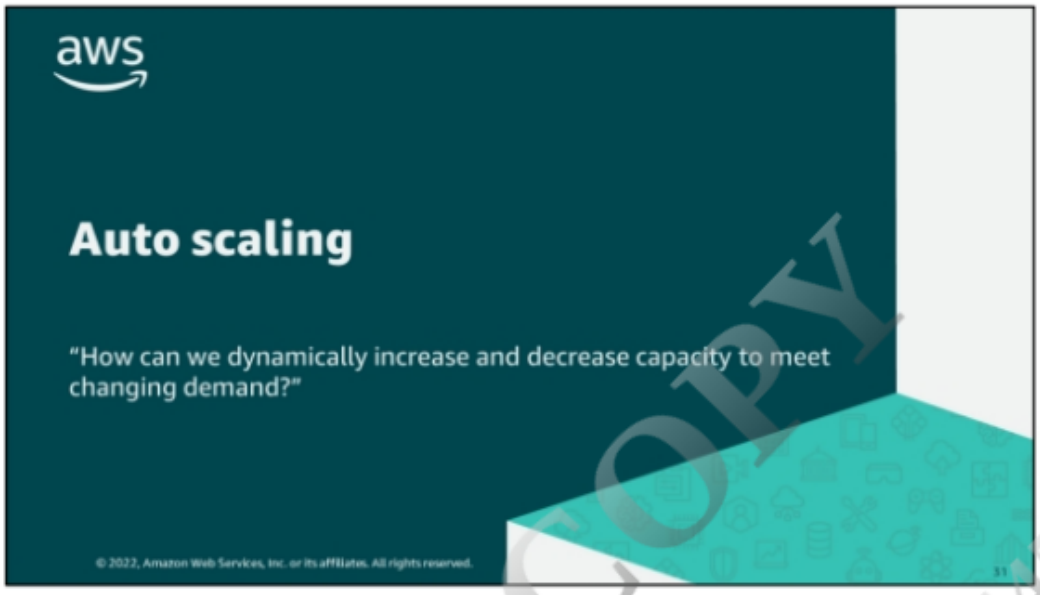
The table in the graphic outlines support for a set of load balancer features.

If you need flexible application management, consider an Application Load Balancer. If you need extreme performance and a static IP, consider a Network Load Balancer. If you need to manage your third-party virtual appliances, consider a Gateway Load Balancer.

**Elastic Load Balancing now supports forwarding traffic directly from Network Load Balancer to Application Load Balancer. With this feature, you can use AWS PrivateLink and expose static IP addresses for applications built on Application Load Balancer.

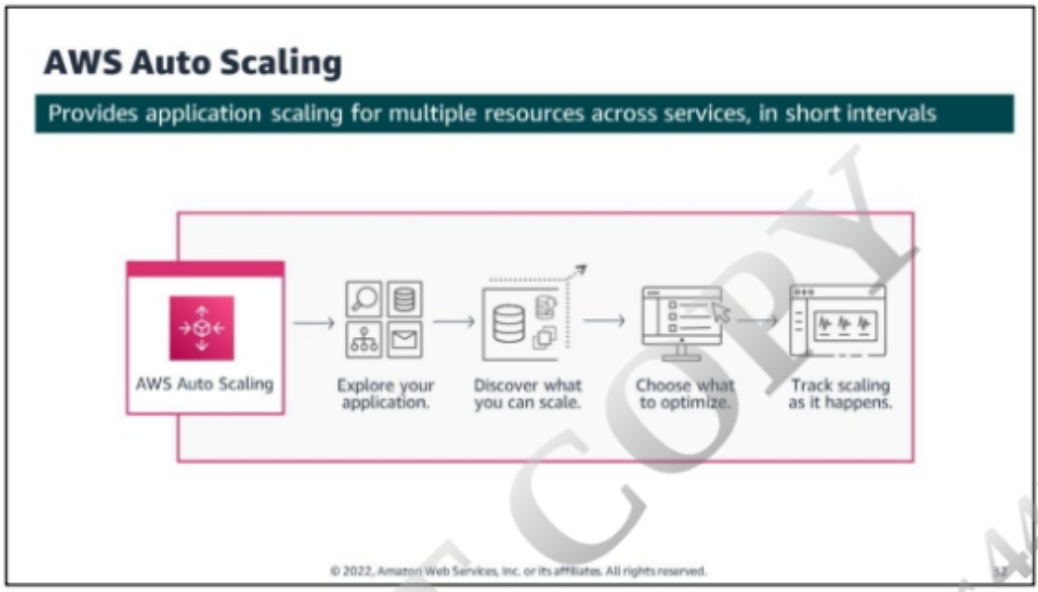
For more information and a full, up-to-date comparison of ELB products, see "Elastic Load Balancing features" (https://aws.amazon.com/elasticloadbalancing/features/?nc=sn&loc=2).

For more information about exposing static IP addresses for applications built on ALB, see "Application Load Balancer now enables AWS PrivateLink and static IP addresses by direct integration with Network Load Balancer" (https://aws.amazon.com/about-aws/whats-new/2021/09/application-load-balancer-aws-privatelink-static-ip-addresses-network-load-balancer/).



The operations manager asks, "How can we dynamically increase and decrease capacity to meet changing demand?"

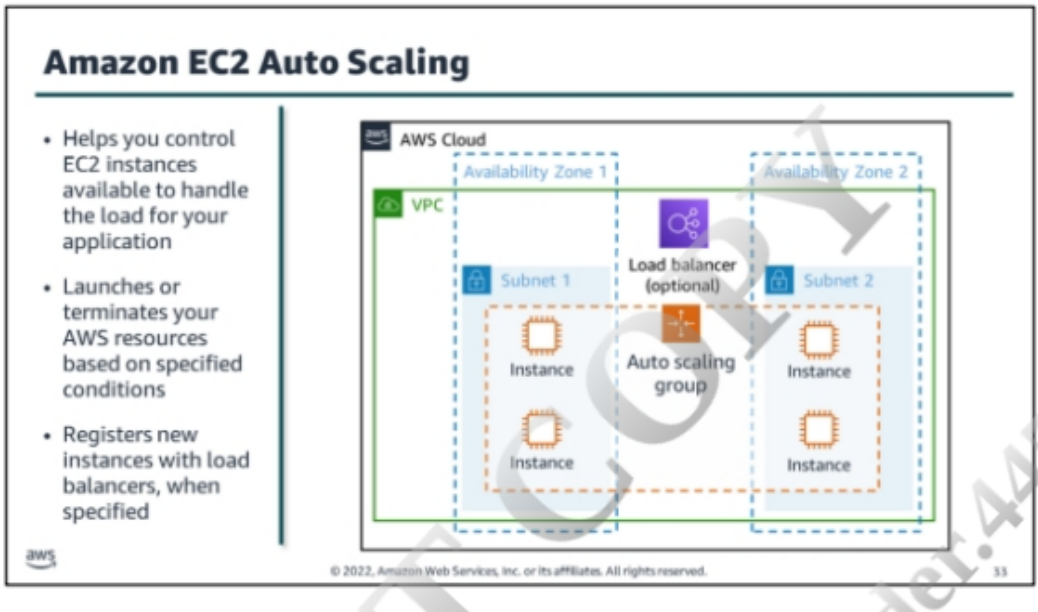
Your operations team needs to compare auto scaling features to determine best practices.



AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost. Using AWS Auto Scaling, you can set up application scaling for multiple resources across multiple services in minutes.

The service provides a simple, powerful user interface that you can use to build scaling plans for resources, including EC2 instances and Spot Fleets, and other compute and database services. AWS Auto Scaling makes scaling simple with recommendations for you to optimize performance and costs, or to balance between them.

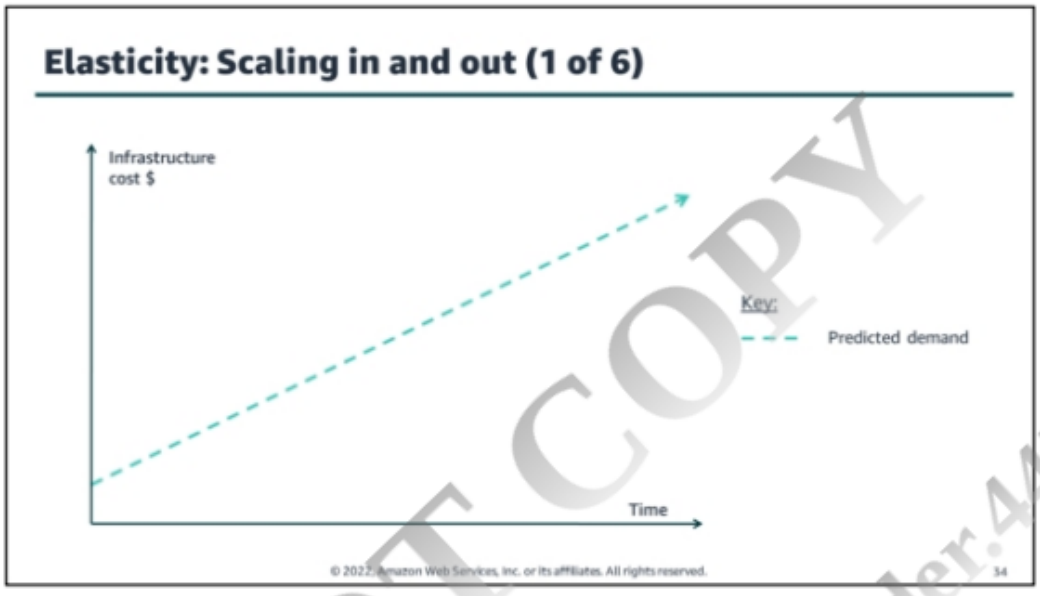
AWS Auto Scaling provides application scaling for multiple resources such as Amazon EC2, Amazon DynamoDB, Amazon Aurora, and many more across multiple services in short intervals.



With Amazon EC2 Auto Scaling, you can build scaling plans that automate how groups of different EC2 resources respond to changes in demand. You can optimize availability, costs, or a balance of both.

If you specify scaling policies, then Amazon EC2 Auto Scaling can launch or terminate instances as demand on your application increases or decreases. Amazon EC2 Auto Scaling integrates with ELB so you can attach one or more load balancers to an existing Amazon EC2 Auto Scaling group. After you attach the load balancer, it automatically registers the instances in the group and distributes incoming traffic across the instances.

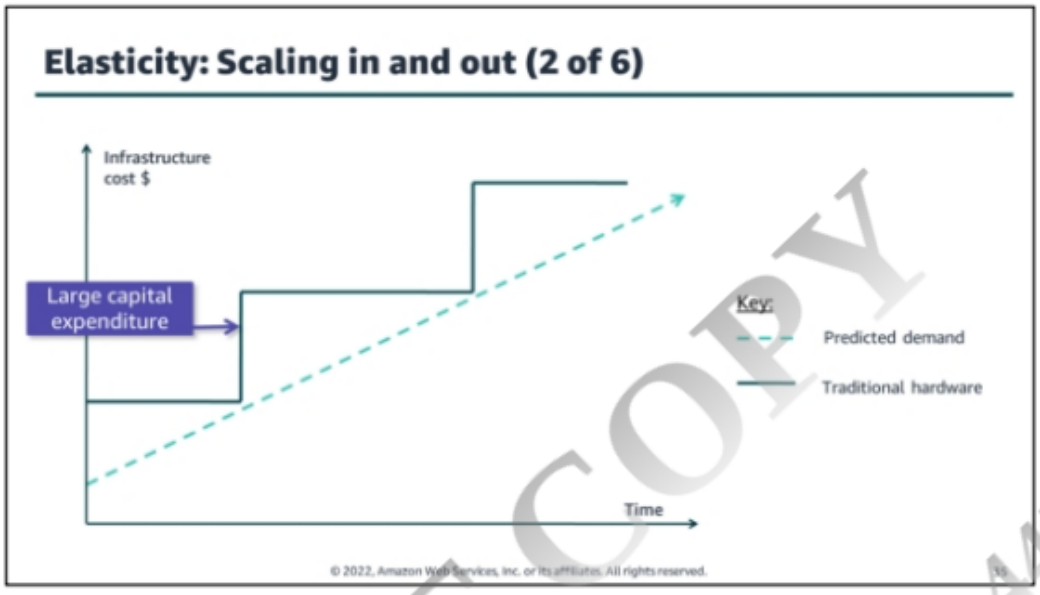
In the example, one VPC has two subnets in two separate availability zones. Two EC2 instances are launched in each subnet as part of one single auto scaling group. An existing load balancer is shown separately, but registering the auto scaling group is optional.



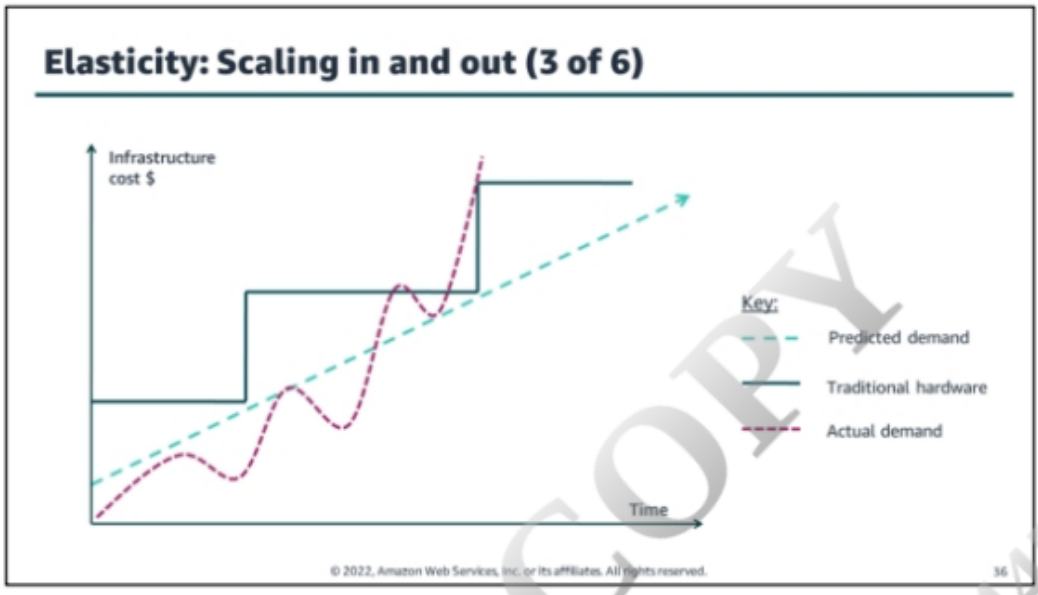
Cloud computing addresses the problem of overuse and underuse of resources through the concept of scaling. What does this mean? As your business grows and demand for your applications increases, your infrastructure costs will also increase over time. An elastic infrastructure can intelligently expand and contract as its capacity needs change.

Examples include the following:

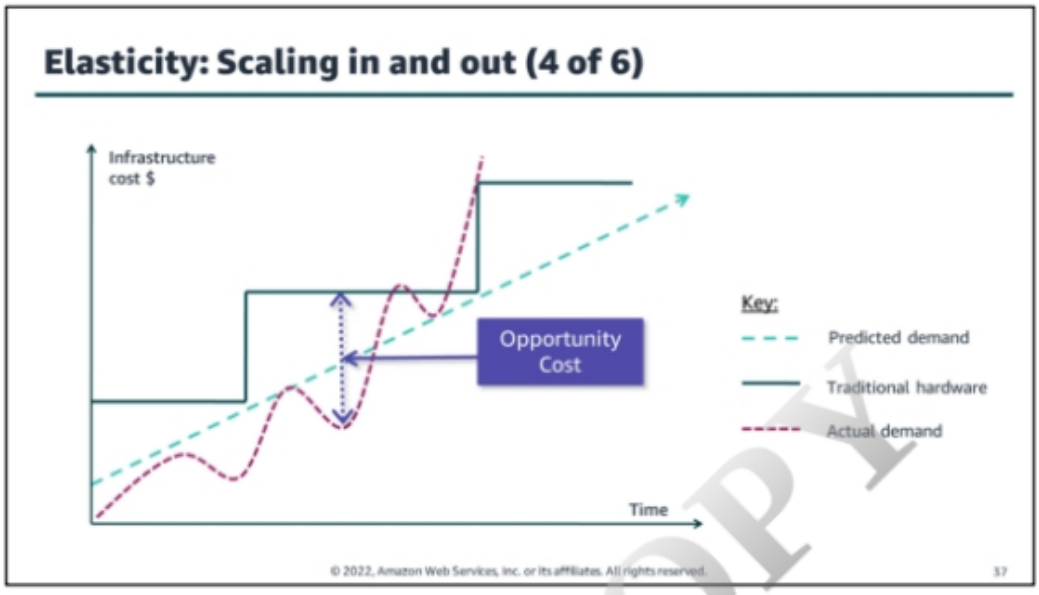
- Increase the number of web servers when traffic spikes.
- Lower write capacity on your database when that traffic goes down.
- Handle the day-to-day fluctuation of demand throughout your architecture.



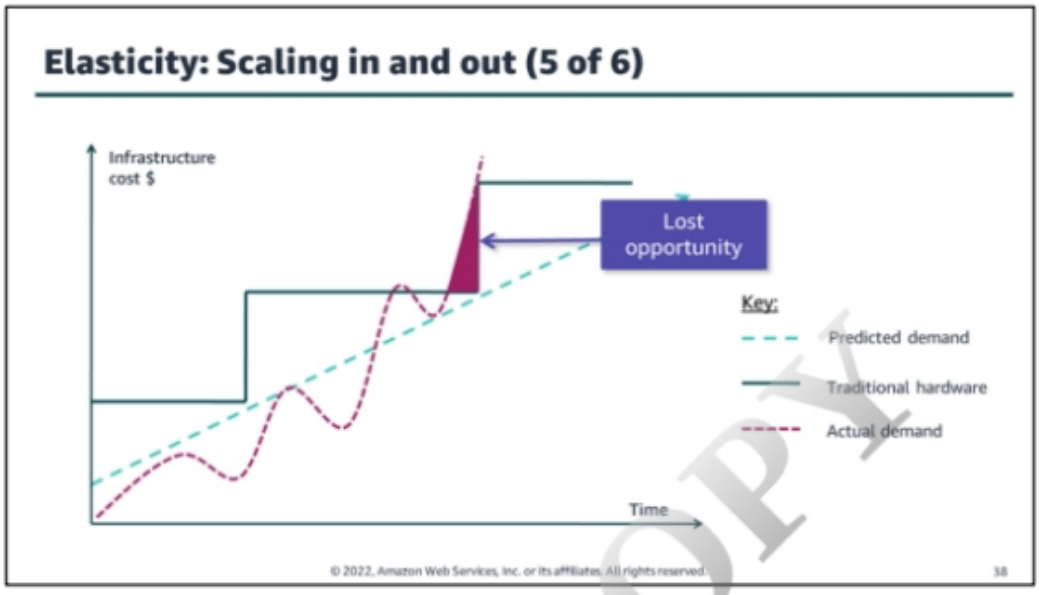
In a traditional IT procurement model, to cope with this increase in demand, businesses would need to regularly invest up front in large purchases of infrastructure. They would continue to use this infrastructure until they needed more capacity, and then make another large investment.



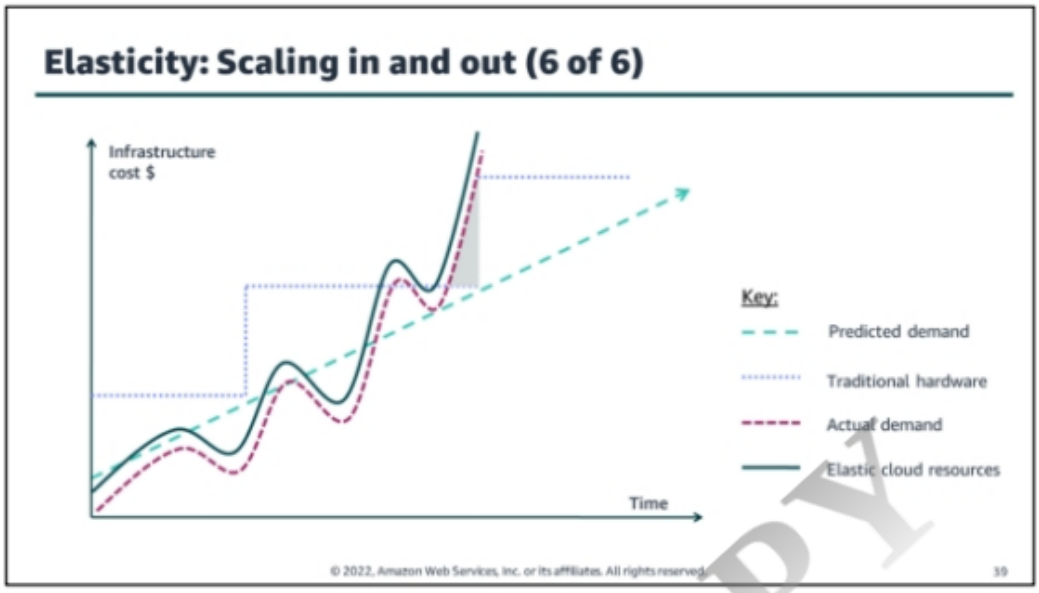
However, demand is rarely linear or predictable. It is far more likely that workloads are spiky and variable.



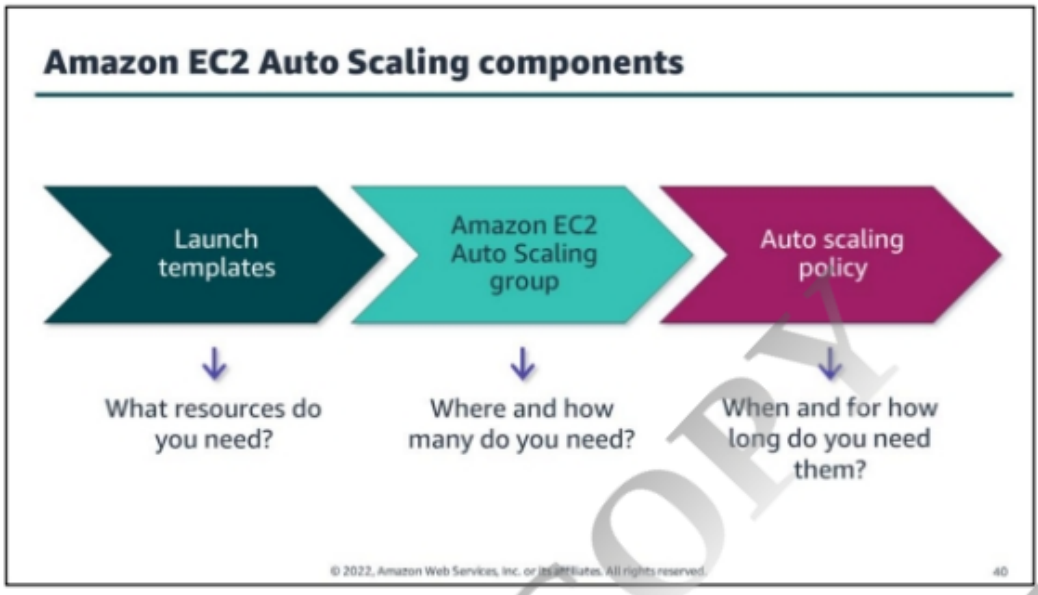
This means a big opportunity cost exists for businesses that have invested in high levels of infrastructure that aren't used when demand is low. Consider how businesses could have reallocated these upfront costs if they had known these demand patterns.



The alternative is that a business underprovisions their infrastructure to save money or because they couldn't have predicted a peak. It experiences a lost opportunity when demand outstrips the available environment. This can create a negative end user experience. Consider the familiar feeling of a website crashing when you are trying to purchase newly released tickets to a popular show.

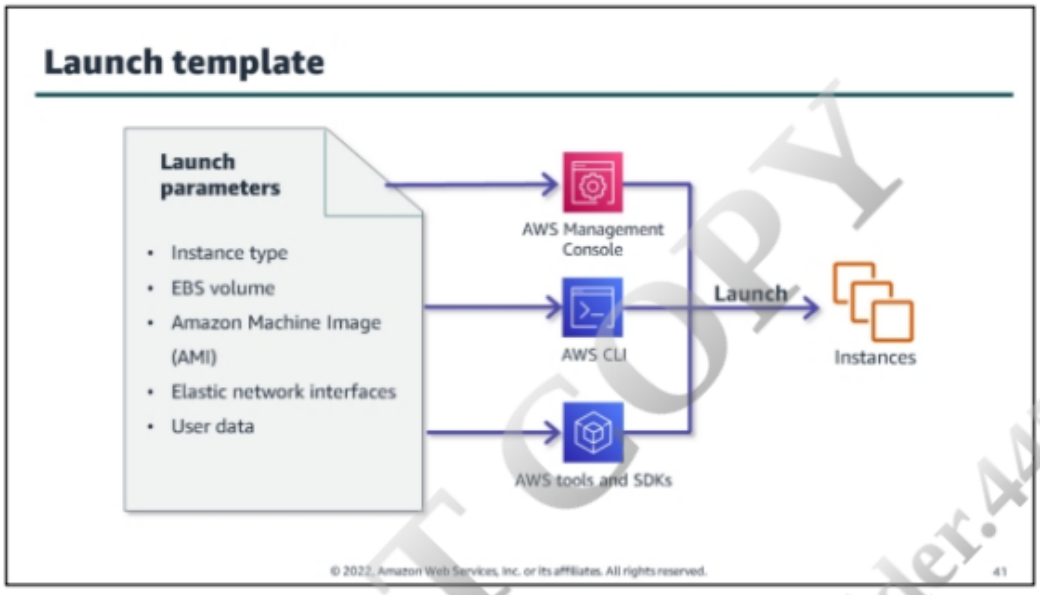


Using cloud technology, you can fit supply to demand through elastic cloud resources. Unlike traditional infrastructure, you don't need to provision resources months in advance. You don't need to keep resources that are not used and you don't need to worry as much about an inability to meet forecast customer demand.



Amazon EC2 Auto Scaling helps you to have the correct number of Amazon EC2 instances available to handle application load. You create collections of EC2 instances, called *Auto Scaling groups*. You can specify the minimum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling manages your group to never go below this size. You can specify the maximum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling manages your group to never go above this size.

The graphic shows how we set up a group from start to finish. We cover each of the three steps in the following slides.



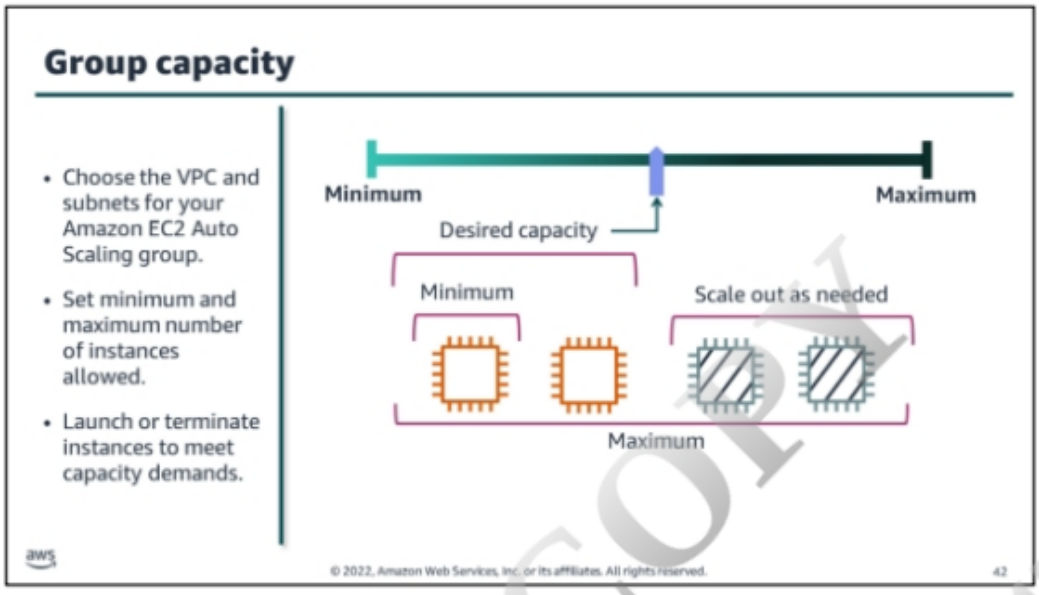
Before you can create an Auto Scaling group using a *launch template*, you must create a launch template that includes the parameters required to launch an EC2 instance, such as the ID of the Amazon Machine Image (AMI) and an instance type. A launch template provides full functionality for Amazon EC2 Auto Scaling and also newer features of Amazon EC2, such as the current generation of Amazon Elastic Block Store (Amazon EBS) Provisioned IOPS volumes (io2), EBS volume tagging, T2 Unlimited instances, Elastic Inference, and Dedicated Hosts.

We strongly recommend that you create Auto Scaling groups from launch templates to get the latest features from Amazon EC2. A *launch configuration* is an instance configuration template that an Auto Scaling group uses to launch EC2 instances. When you create a launch configuration, you must specify information about the EC2 instances to launch. Include the AMI, instance type, key pair, security groups, and block device mapping.

Alternatively, you can create a launch configuration using attributes from a running EC2 instance.

For more information about templates, see "Create a launch template for an Auto Scaling group" in the *Amazon EC2 Auto Scaling User Guide* (https://docs.aws.amazon.com/autoscaling/ec2/userguide/create-launch-template.html).

For more information about launch configurations, see "Create a launch configuration using an EC2 instance" in the *Amazon EC2 Auto Scaling User Guide* (https://docs.aws.amazon.com/autoscaling/ec2/userguide/create-lc-with-instanceID.html).

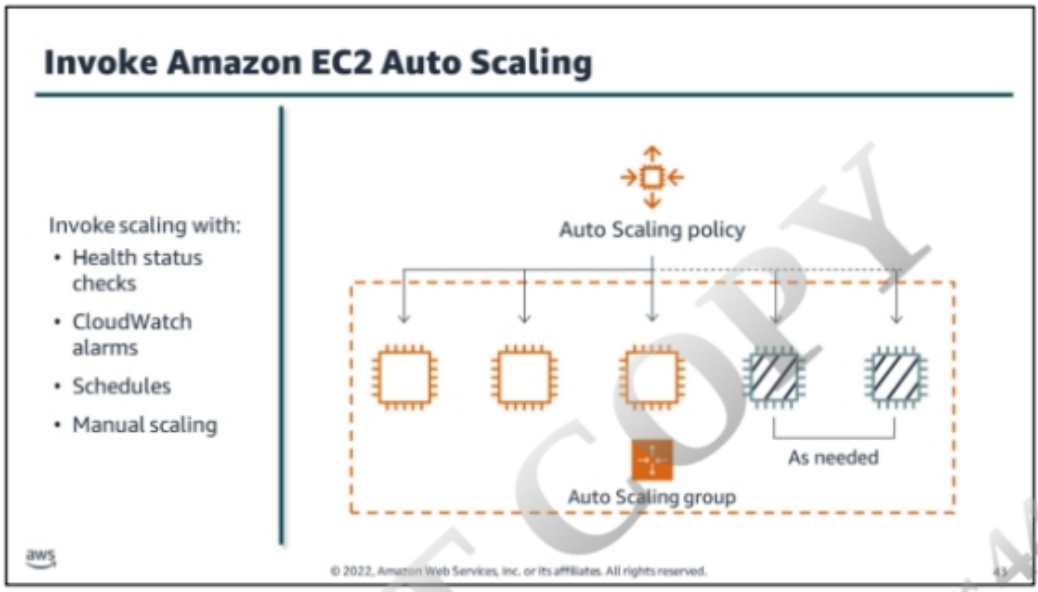


A group contains a collection of EC2 instances that are treated as a logical grouping for automatic scaling and management. With a group, you can also use Amazon EC2 Auto Scaling features such as health check replacements and scaling policies. You can specify the minimum number of instances in each group and Amazon EC2 Auto Scaling controls the group to never go below this size.

If you specify the desired capacity, either when you create the group or at any time thereafter, Amazon EC2 Auto Scaling manages your group to have this many instances. If you specify scaling policies, Amazon EC2 Auto Scaling can launch or terminate instances as demand on your application increases or decreases.

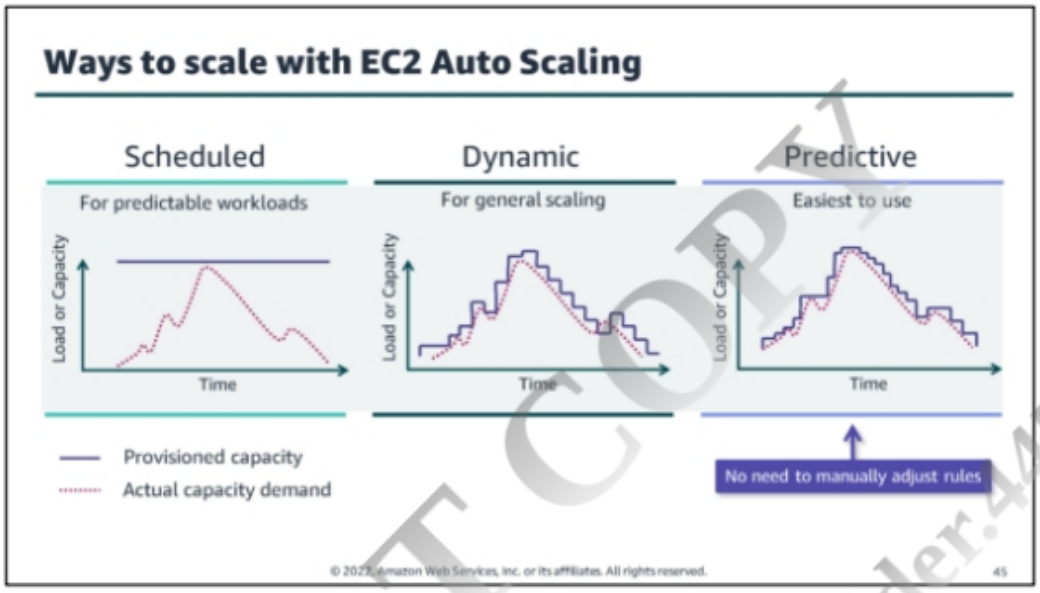
For example, this group has a minimum size of one instance, a desired capacity of two instances, and a maximum size of four instances. The scaling policies that you define adjust the number of instances, within your minimum and maximum number of instances, based on the criteria that you specify.

For more information, see "Auto Scaling groups" in the *Amazon EC2 Scaling User Guide* (https://docs.aws.amazon.com/autoscaling/ec2/userguide/AutoScalingGroup.html).



You can use the following tools to invoke scaling in your groups:

- **Health status checks** – You can configure your group to maintain a specified number of running instances at all times. If an instance becomes unhealthy, the group terminates the unhealthy instance and launches another instance to replace it.
- **CloudWatch alarms** – A scaling policy instructs Amazon EC2 Auto Scaling to track a specific CloudWatch metric. It defines what action to take when the associated CloudWatch alarm is in the ALARM state.
- **Schedules** – You can scale by schedule. Actions are performed automatically as a function of time and date. Scaling by schedule is useful when you know exactly when to increase or decrease the number of instances in your group.
- **Manual scaling** – Manual scaling is the most basic way to scale your resources. Specify only the change in the maximum, minimum, or desired capacity of your group. Amazon EC2 Auto Scaling manages the process of creating or terminating instances to maintain the updated capacity.



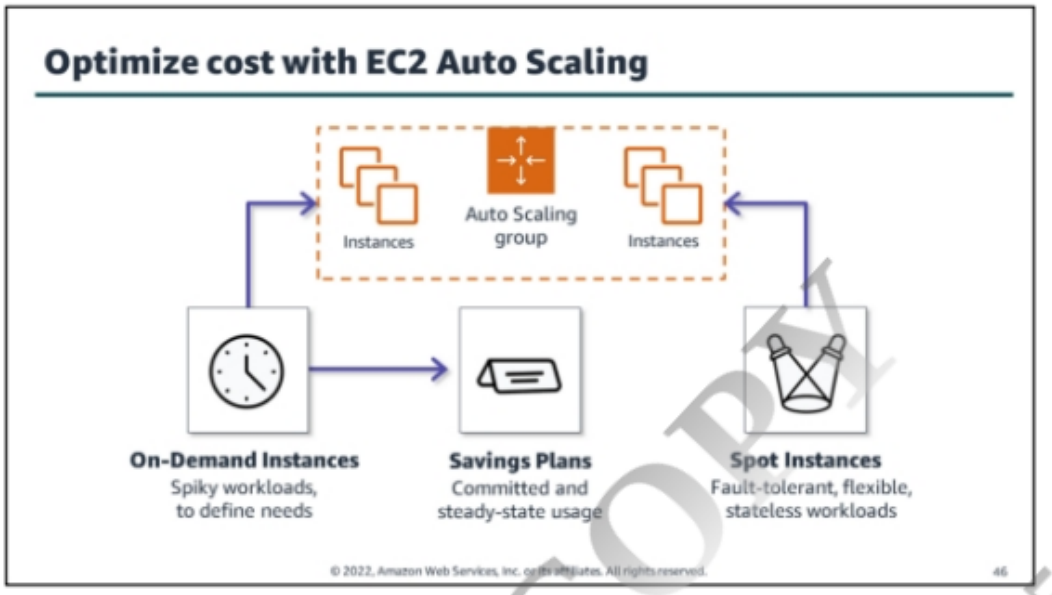
With *scheduled scaling*, you can scale your application before known load changes. For example, every week, the traffic to your web application starts to increase on Wednesday, remains high on Thursday, and starts to decrease on Friday. You can plan your scaling activities based on the known traffic patterns of your web application.

With *dynamic scaling*, you define how to scale the capacity of your Amazon EC2 Auto Scaling group in response to changing demand. For example, suppose that you have a web application that currently runs on two instances and you want the CPU utilization of the group to stay at about 50 percent when the load on the application changes. This gives you extra capacity to handle traffic spikes without maintaining an excessive number of idle resources.

Use *predictive scaling* to increase the number of EC2 instances in your group in advance of daily and weekly patterns in traffic flows. Predictive scaling is well suited for situations where you have the following:

- Cyclical traffic, such as high use of resources during regular business hours and low use of resources during evenings and weekends
- Recurring on-and-off workload patterns, such as batch processing, testing, or periodic data analysis
- Applications that take a long time to initialize, causing a noticeable latency impact on application performance during scale-out events

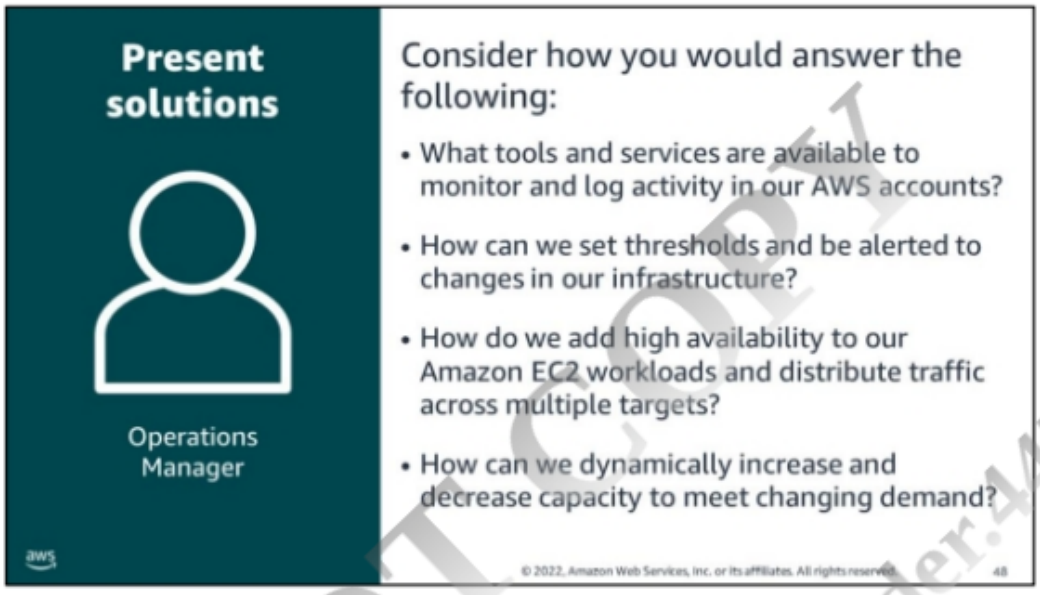
It is recommended for you to scale out early and fast, while you scale in slowly over time.



Amazon EC2 Auto Scaling supports multiple purchasing options within the same group. You can launch and automatically scale a fleet of On-Demand Instances and Spot Instances within a single Auto Scaling group. In addition to receiving discounts for using Spot Instances, you can use Reserved Instances or a Savings Plan to receive discounted rates of the regular On-Demand Instance pricing. All of these factors combined help you to optimize your cost savings for EC2 instances, while making sure that you obtain the desired scale and performance for your application.

Using *Amazon EC2 Fleet*, you can define a combination of EC2 instance types to make up the desired capacity of your group. This is defined as a percentage of each type of purchasing option. Amazon EC2 Auto Scaling will maintain the desired cost optimization as your group scales in or out. Groups made up of mixed fleets still support the same lifecycle hooks, instance health checks, and scheduled scaling as a single-fleet group.

To learn more about optimizing cost, see "Auto Scaling groups with multiple instance types and purchase options" in the *Amazon EC2 Auto Scaling User Guide* (https://docs.aws.amazon.com/autoscaling/ec2/userguide/asg-purchase-options.html).



Imagine you are now ready to talk to the operations manager and present solutions that meet their architectural needs.

Think about how you would answer the questions from the beginning of the lesson.

Your answers should include the following solutions:

- Use tools like CloudWatch metrics, CloudWatch Logs, CloudTrail, and VPC Flow Logs to monitor and log activity in your accounts.
- Configure CloudWatch alarms for existing metrics and use EventBridge to take action when a metric is in the ALARM state.
- Explore options available to you in Elastic Load Balancing. Choose either an Application Load Balancer, Network Load Balancer, or Gateway Load Balancer based on your use case.
- Prepare for changes in demand using AWS Auto Scaling. For compute, use Amazon EC2 Auto Scaling to invoke scaling based on the expected demand for that workload. Use launch templates to build your configuration to support scaling in and out.

Module review

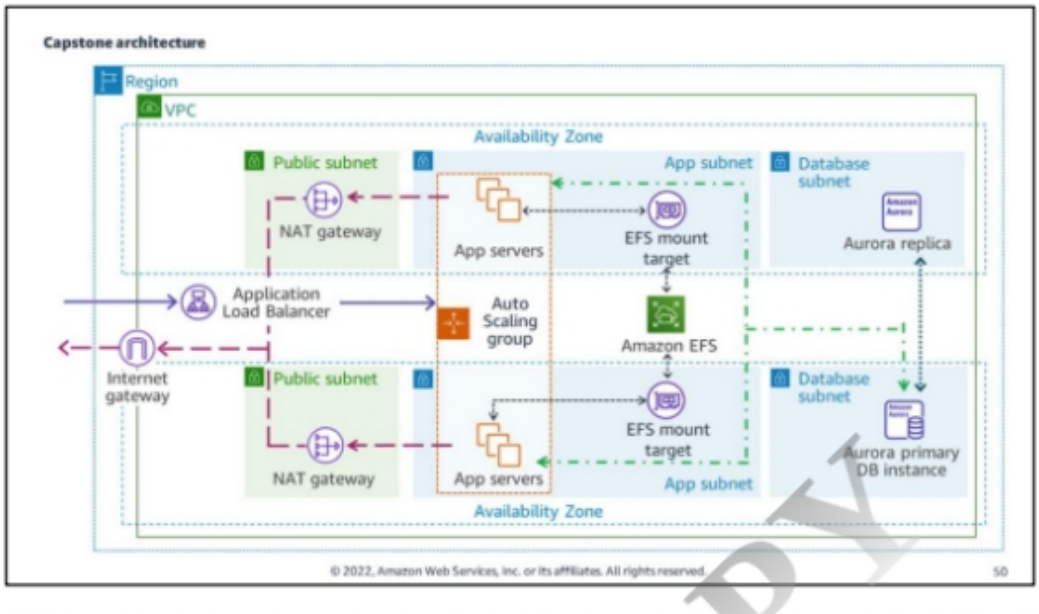
In this module you learned about:

- ✓ Monitoring
- ✓ Alarms and events
- ✓ Load balancing
- ✓ Auto scaling

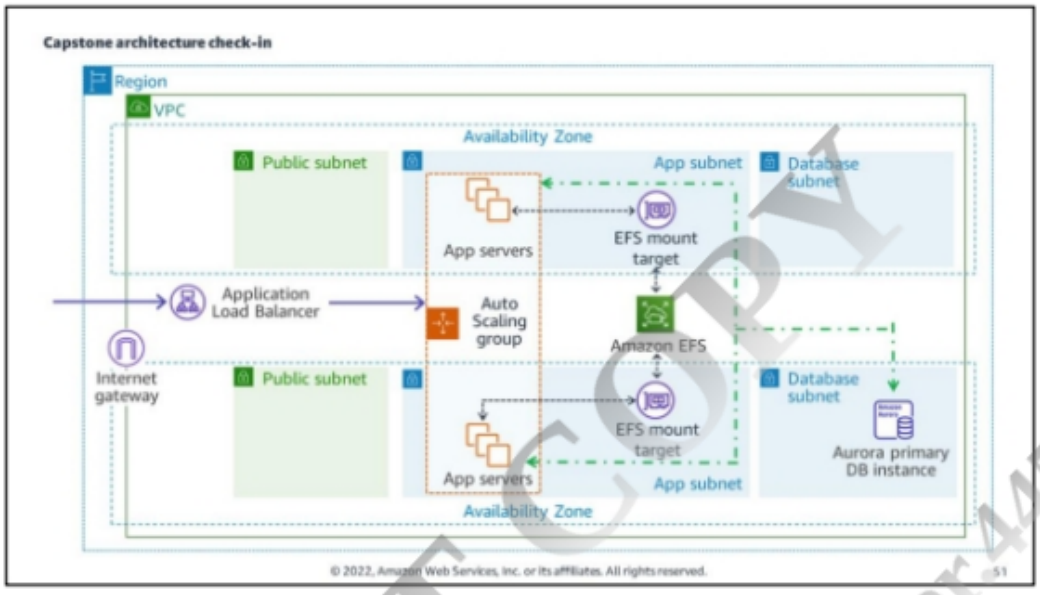
Next, you will review:

- Capstone check-in
- Lab introduction
- Knowledge check

 49



At the end of this course is a Capstone Lab project. You will be provided a scenario and asked to build an architecture based on project data, best practices, and the Well-Architected Framework.



You notice an ELB load balancer is added to balance the WordPress traffic across multiple EC2 instances. This also gives you fault-tolerance if one of your EC2 instances fails a health check for application traffic.

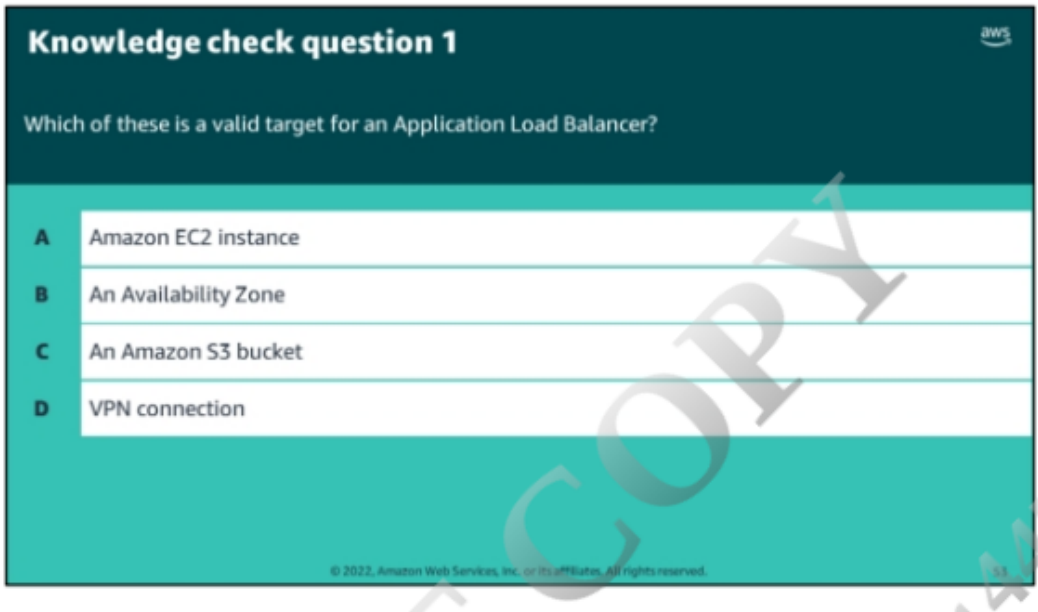
Your application servers added in the Compute module have been placed in an Amazon EC2 Auto Scaling group. This adds scalability to your application.

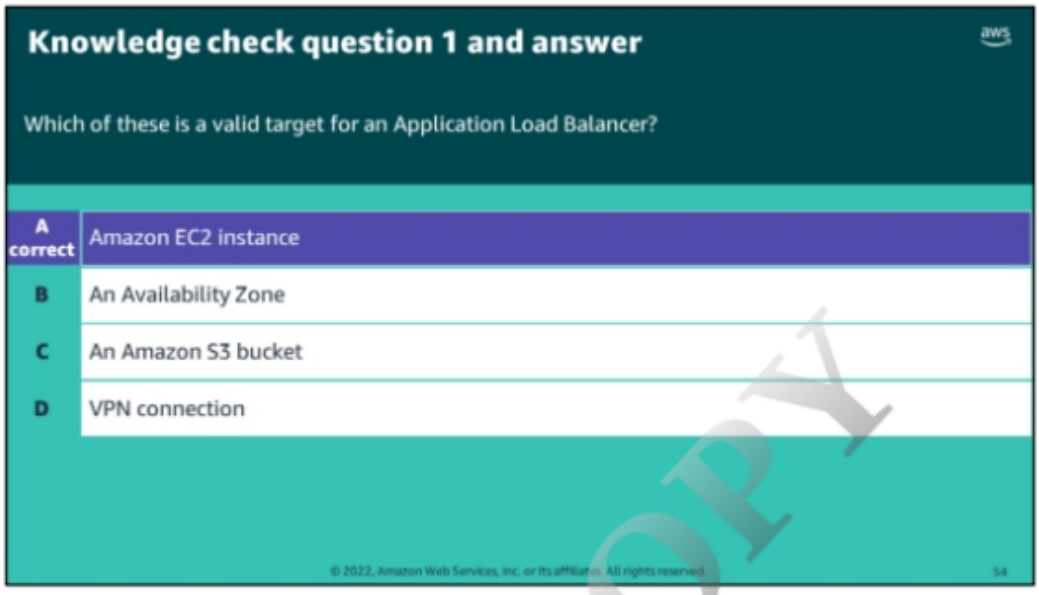
As your Auto Scaling group scales out, the launch template will use your AMI to launch application servers, and user data in the launch template is processed during instance launch. You can use this to mount your Amazon Elastic File System (Amazon EFS) automatically at instance launch. The EC2 instances in the auto scaling group will launch with security groups that allow incoming traffic from the Amazon Aurora primary DB instance.

You have now learned about all of the services used in the Capstone Lab. You will revisit this architecture again on Day 3 to strengthen your knowledge on these topics.

Consider how you could continue expanding on this architecture by reviewing the following questions:

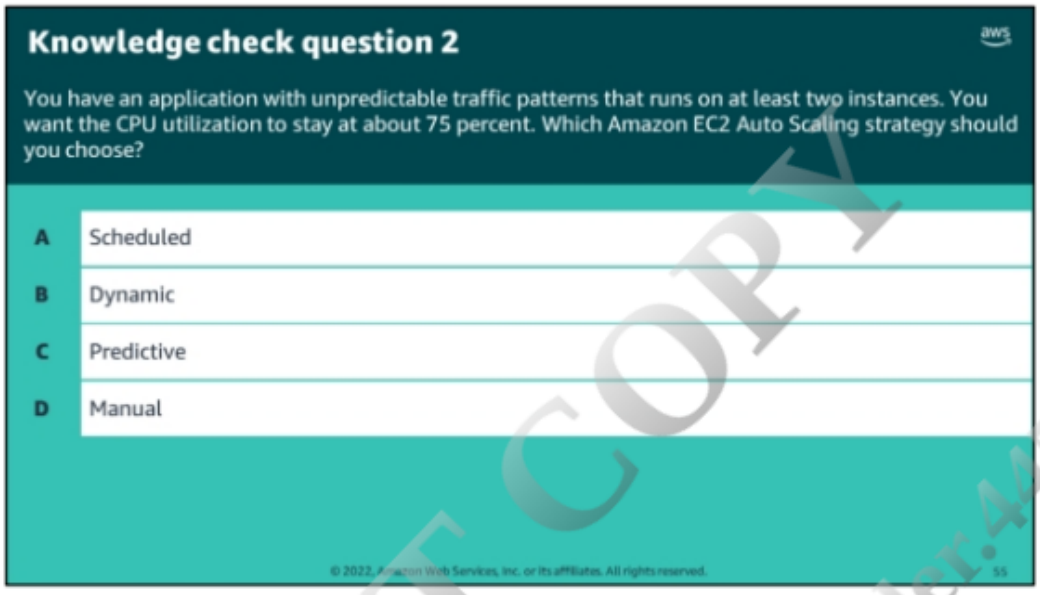
- How would you automate the creation of resources?
- How would you duplicate this deployment in more than one account without manual effort?
- Can you use containers instead of virtual machines to control the application?
- Can you integrate this application with your on-premises network?
- Do you need more than one account to be able to access this application?
- What serverless strategies can you use to help with easy management of the application?
- What is your disaster recovery strategy for this environment?

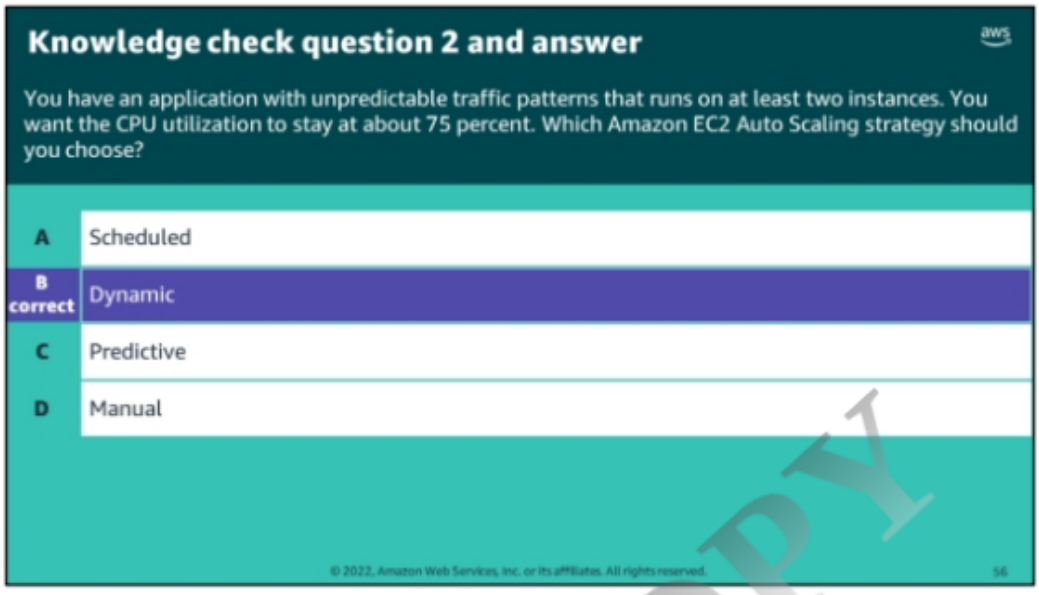




The correct answer is A, Amazon EC2 instance.

A load balancer distributes incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones to increase the availability of your application.

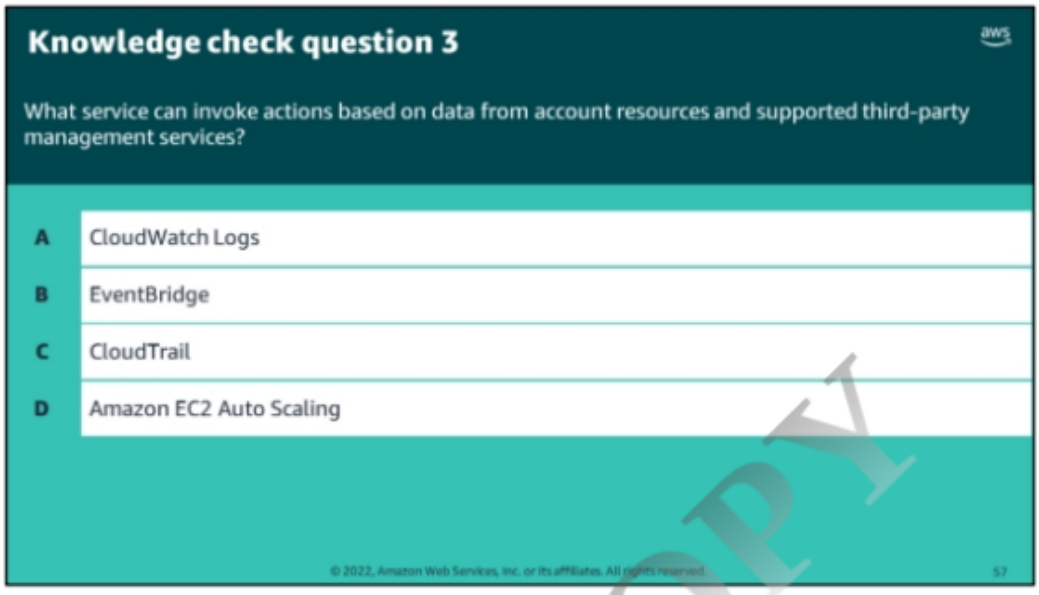


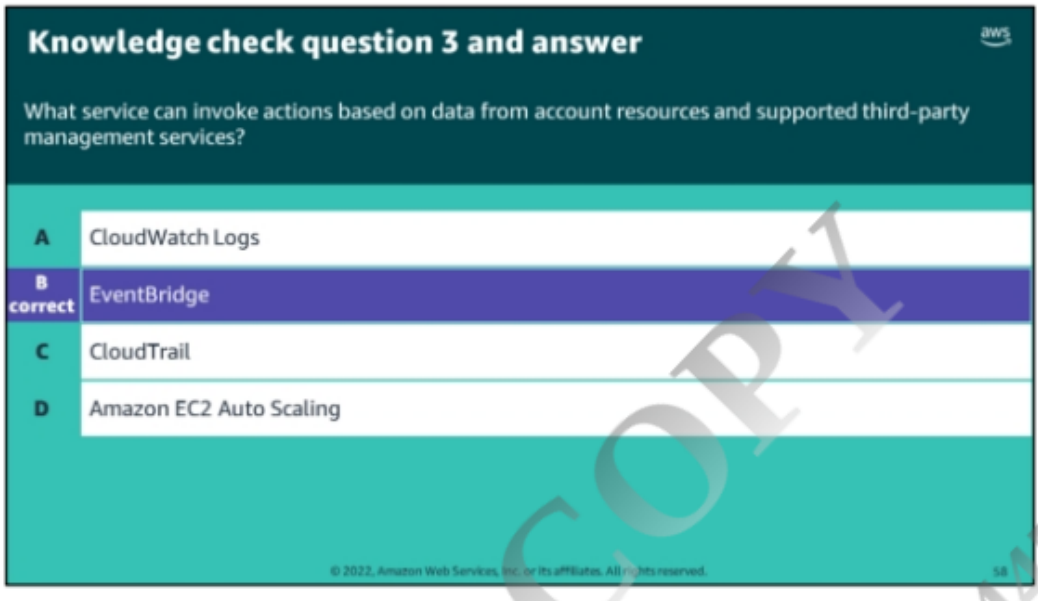


The answer is B, dynamic.

This option gives you extra capacity to handle traffic spikes without maintaining an excessive number of idle resources.

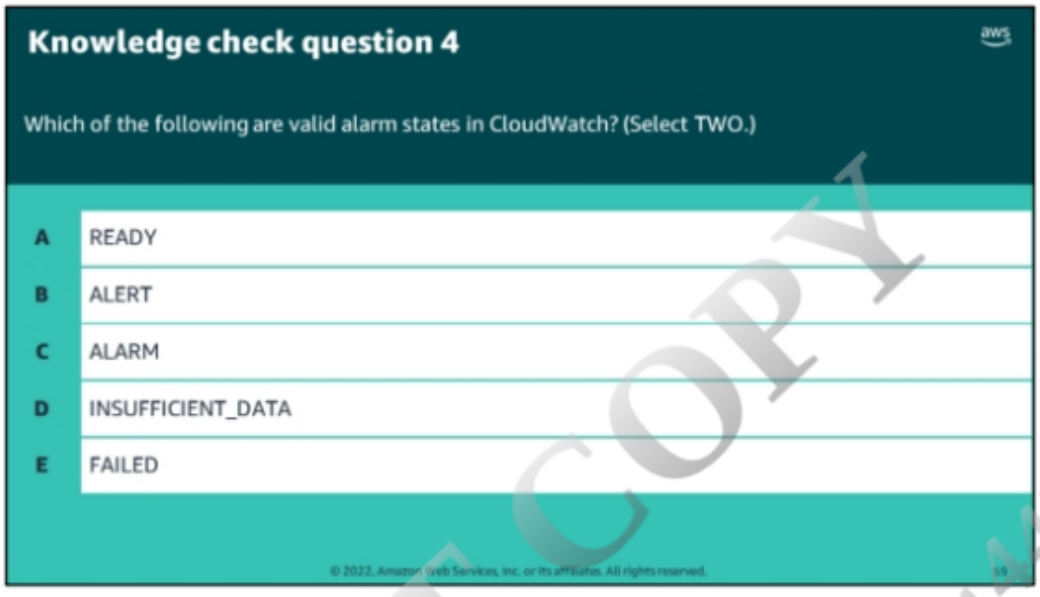
Predictive scaling would be best for cyclical traffic, recurring on-and-off workload patterns, or applications that take a long time to initialize.

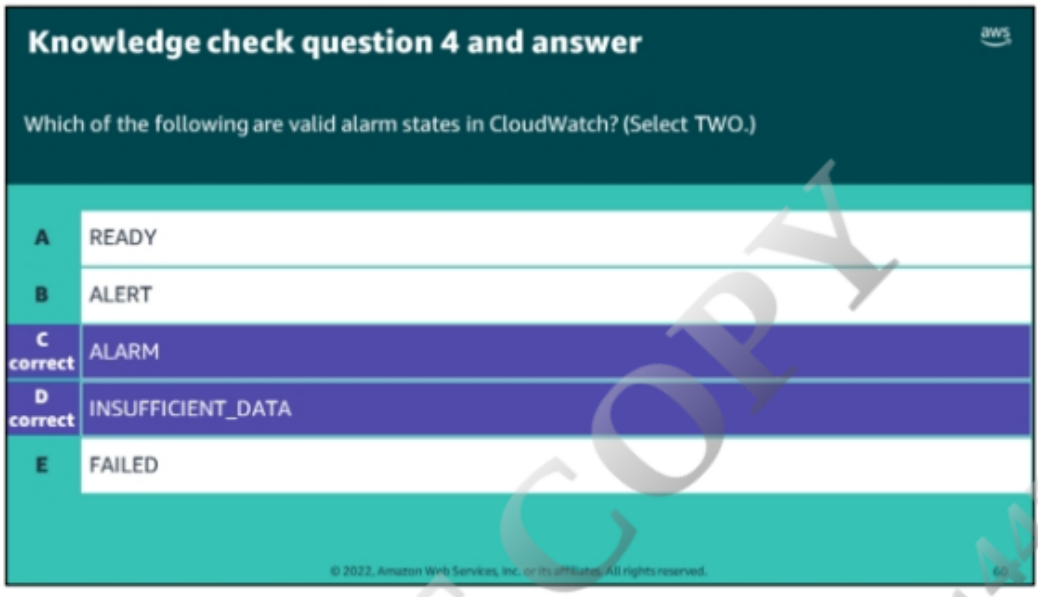




The correct answer is B, EventBridge.

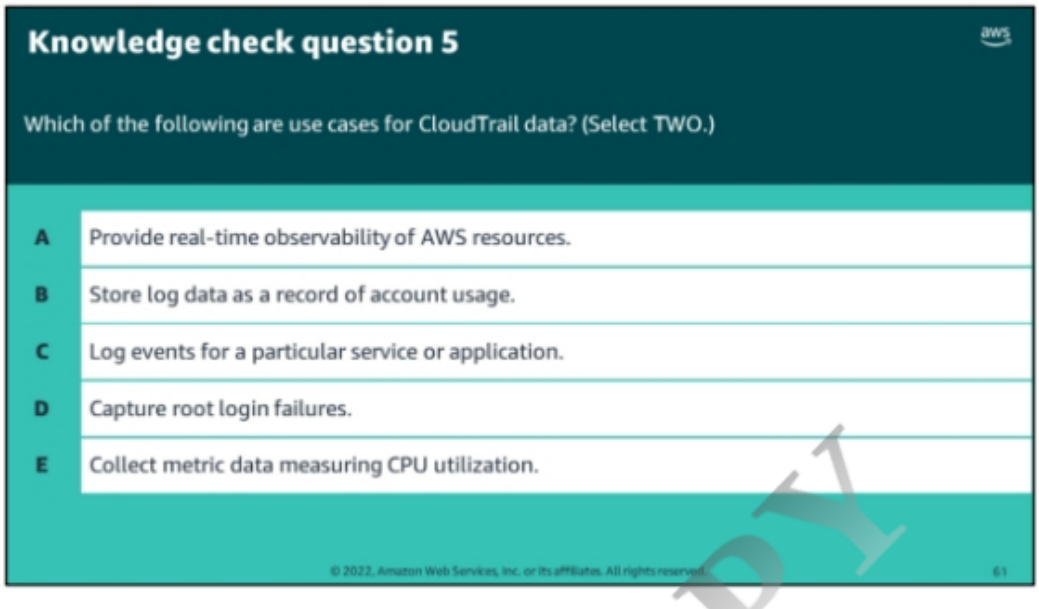
EventBridge builds upon and extends CloudWatch Events. It uses the same service API and endpoint, and the same underlying service infrastructure. EventBridge reacts to events generated by other AWS services.

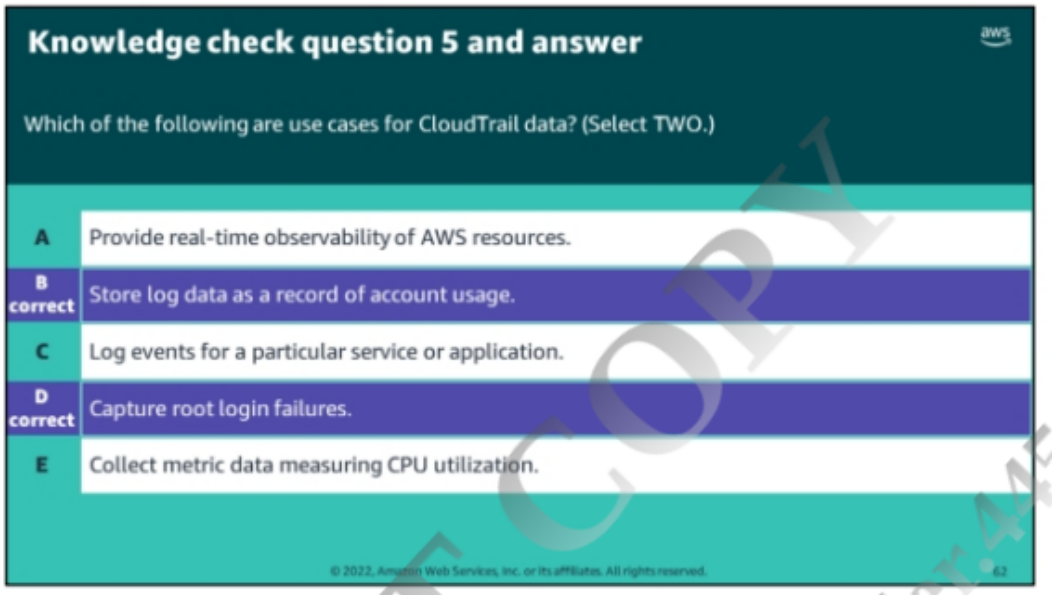




The correct answer is C and D, alarm and insufficient data.

Amazon CloudWatch alarms have three possible states: OK, ALARM, and INSUFFICIENT_DATA.

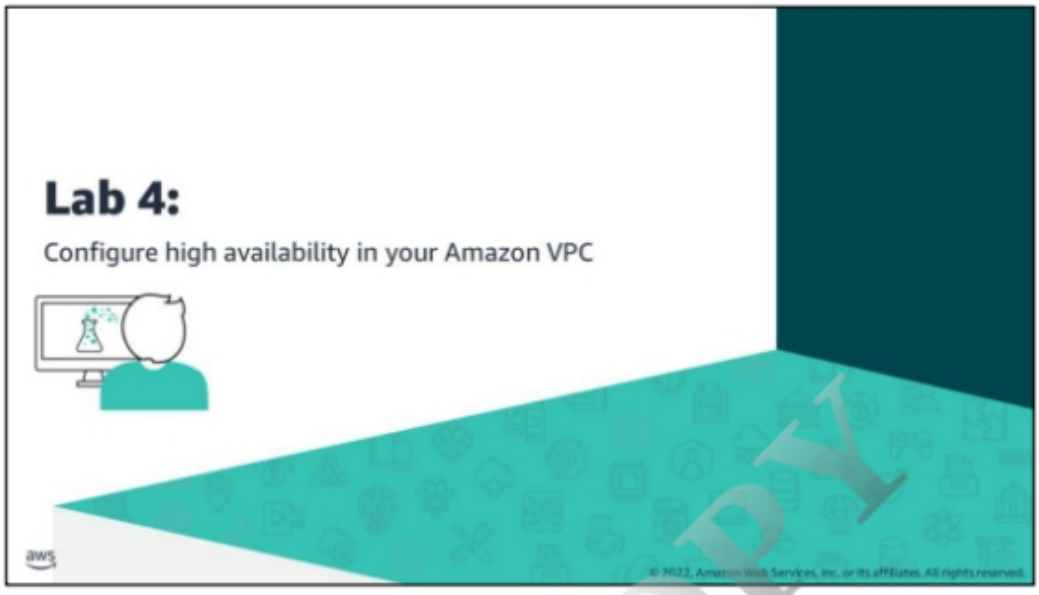


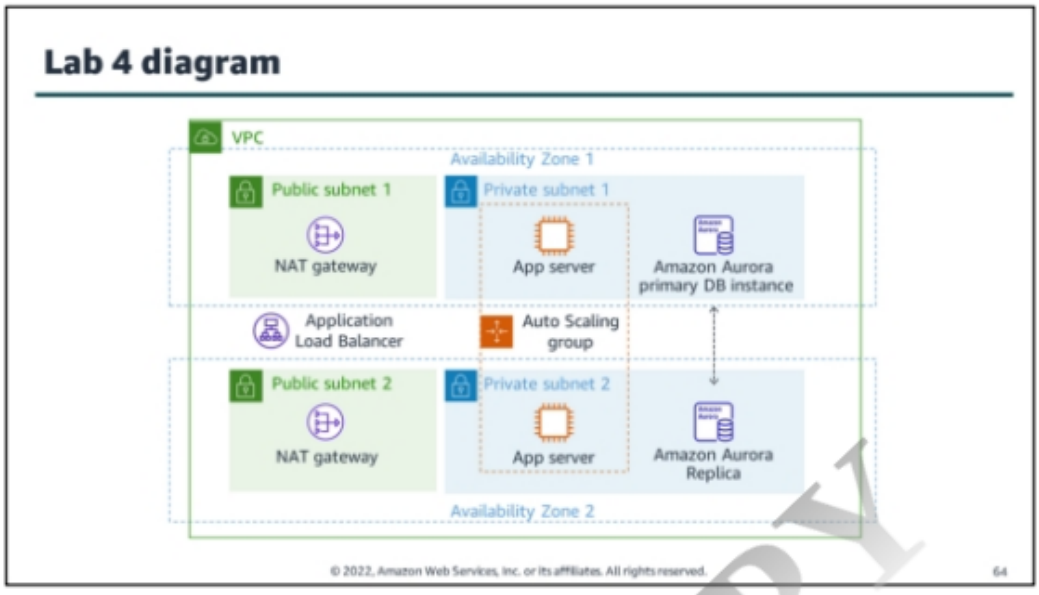


The correct answer is B and D, store log data as a record of account usage and capture root login failures.

CloudTrail provides insight into who is doing what and when, by tracking user activity and API usage. CloudTrail does the following:

- Monitors and logs account activity across your AWS infrastructure
- Records API call interactions for most AWS services
- Automatically pushes logs to Amazon S3
- Captures root login failures





Lab 4 uses a VPC across two Availability Zones with a public and private subnet in each. Each private subnet contains an app server in the same Auto Scaling group. Private subnet 1 has an Aurora primary DB instance, while private subnet 2 has an Aurora Replica. The primary DB instance and replica communicate with each other across Availability Zones. Each public subnet has a NAT gateway. An Application Load Balancer is in the VPC.