SWINBURNE UNIVERSITY OF TECHNOLOGY

Cloud Computing Architecture

Lecture 7 Scalable Architectures

includes material from
ACF Module 10 – ELB, CloudWatch and Auto Scaling
ACA Module 3 – Designing for High Availability

Reminders

- **First MCQ exam September 24th, 3 pm sharp, 60 minutes**
  - Access from Canvas
  - You must pass MCQs to pass the unit
  - Practice test available.
- **Assignment 2 - Due date: Thursday 8th October 9 am**
  - Implement on specified Vocareum classroom (SDCC - Assignments)
  - Due to Canvas
    Late submission penalty: 10% of total available marks per day.
  - Discussion board provided
    - Feel free to ask and answer questions (don't post code)

SWINBURNE UNIVERSITY OF TECHNOLOGY

Last week - Security

- Access Control concepts
- Security in the Cloud: AWS Shared Responsibility Model
- AWS IAM
  - Users, Groups, Roles
  - Authorization using Policies
  - Securing your AWS account
- AWS Authentication
  - Cognito, Directory service, STS, Web Identity
- Securing Data
  - In transit, at rest.
  - Encryption
- Securing the System
  - DDOS, AWS WAF
- Auditing
  - AWS CloudTrail, Config

Quizzes:
ACF Security
ACA Mod 10 Well Architected Security

SWINBURNE UNIVERSITY OF TECHNOLOGY

This week – Scalable Architectures

- **High availability and Scaling**
- Automated Scaling and Monitoring on AWS
  - ☐Elastic Load Balancing
  - ☐Amazon CloudWatch
  - ☐Amazon EC2 Auto Scaling
- Labs
  - ☐ *ACF Lab 6*
  - ☐ACA Module 3

Quizzes:
ACF 2.0.5 Balancing, Scaling and Monitoring
ACA Mod 4 Designing for High Availability

SWINBURNE UNIVERSITY OF TECHNOLOGY

What Is High Availability?

High availability (HA) is about ensuring that your application's downtime is minimized as much as possible, without the need for human intervention.

Levels of Availability:

	Percent of Uptime	Max Downtime per Year	Equivalent Downtime per Day
1 Nine	90%	36.5 days	2.4 hrs
2 Nines	99%	3.65 days	14 min
3 Nines	99.9%	8.76 hrs	86 sec
4 Nines	99.99%	52.6 min	8.6 sec
5 Nines	99.999%	5.25 min	.86 sec

Availability refers to the amount of time your system is in a functioning condition. In general terms, your availability is referred to as 100% minus your system's downtime. Since events which may disrupt your system's availability are never entirely predictable, there are always ways to make an application more available, however, improving availability typically leads to increased cost. When considering how to make your environment more available, it's important to balance the cost of the improvement with the benefit to your users. HA might mean that you ensure your application is always alive/reachable or does it mean that the app is servicing requests within an acceptable level of performance?

High Availability Factors

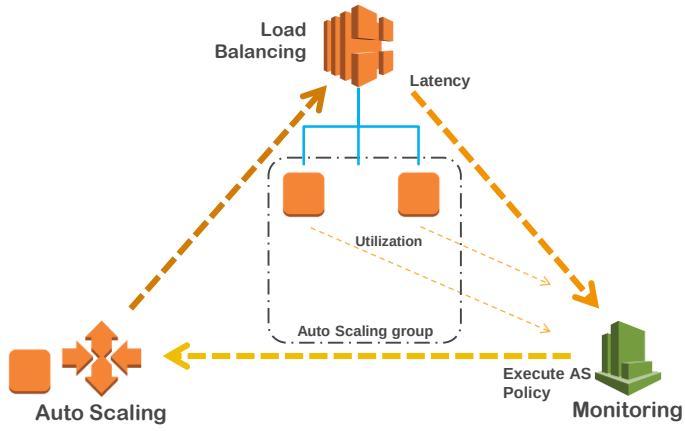
Need to balance *capacity* with *demand*

- **Scalability:**
  The ability of an application to accommodate growth without changing design.
- **Fault tolerance**:
  The built-in redundancy of an application's components.
- **Recoverability:** *(Week 9)*
  The process, policies, and procedures related to restoring service after a catastrophic event.

Fault tolerance, recoverability, and scalability are three primary factors that determine the overall availability of your application.
Fault tolerance is often confused with high availability, but fault tolerance refers to the built-in redundancy of an application's components. Does it avoid single points of failure? We will talk more about fault tolerance within this module.

Recoverability is often overlooked as a component of availability. In the event a natural disaster makes one or more of your components unavailable, or destroys your primary data source, are you able to restore service quickly and without lost data? We will talk more about specific disaster recovery strategies in a later module.
Scalability is a question of how quickly your application's infrastructure can respond to increased capacity needs to ensure your application is available and performing within your required standards. It does not guarantee availability, but is one part of your application's availability.

Services for Highly Available Web Application

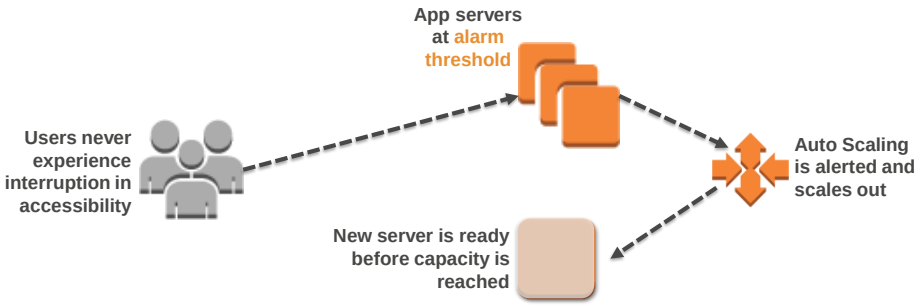


Auto Scaling works as a triad of services working in sync. Elastic Load Balancing and EC2 instances feed metrics to Amazon CloudWatch. Auto Scaling defines a group with launch configurations and Auto Scaling policies. Amazon CloudWatch alarms execute Auto Scaling policies to affect the size of your fleet. All of these services work well individually, but together they become more powerful and increase the control and flexibility our customers demand.

Best Practice: Enable Automated Scalability

Ensure that your architecture can handle changes in demand.

A key advantage of a cloud-based infrastructure is how quickly you can respond to changes in resource needs.



Best practice

1. Using the best practice of enabling scalability, this pattern is designed to anticipate need and have more capacity available before it's too late.
2. A monitoring solution (such as Amazon CloudWatch) detects that the total load across the fleet of servers has reached a specified threshold of load. This could be anything, such as "Stayed above 60% CPU utilization for longer than 5 minutes", or anything related to the use of resources, and with CloudWatch, you can even design custom metrics based around your specific application, which can trigger scaling in whatever way you need. When that alarm is triggered, Amazon EC2 Auto Scaling immediately launches a new instance.
3. That instance is then ready before capacity is reached, providing a seamless experience for users, who never know that capacity was in danger of being reached.

Ideally, you should also design this system to scale down once demand drops off again, so that you're not running instances that are no longer needed.

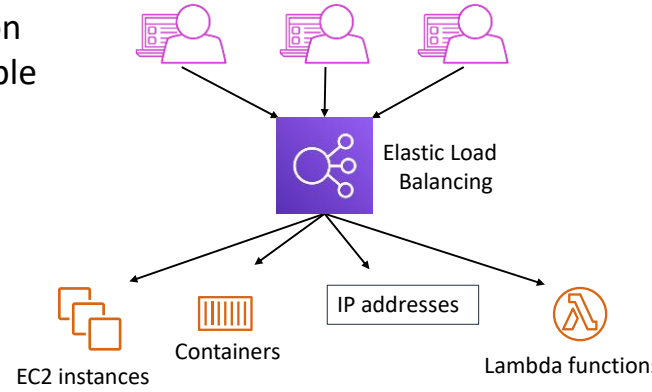
This week – Scalable Architectures

- High availability and Scaling
- Automated Scaling and Monitoring on AWS
  - **Elastic Load Balancing**
  - Amazon CloudWatch
  - Amazon EC2 Auto Scaling

SWINBURNE UNIVERSITY OF TECHNOLOGY

Elastic Load Balancing

- Distributes incoming application or network traffic across multiple targets in a single Availability Zone or across multiple Availability Zones.
- Scales your load balancer as traffic to your application changes over time.



Modern high-traffic websites must serve hundreds of thousands—if not millions—of concurrent requests from users or clients, and then return the correct text, images, video, or application data in a fast and reliable manner. Additional servers are generally required to meet these high volumes.

Elastic Load Balancing is an AWS service that distributes incoming application or network traffic across multiple targets—such as Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, internet protocol (IP) addresses, and Lambda functions—in a single Availability Zone or across multiple Availability Zones. Elastic Load Balancing scales your load balancer as traffic to your application changes over time. It can automatically scale to most workloads.

Types of load balancers

Application Load Balancer	Network Load Balancer	Classic Load Balancer (Previous Generation)
• Load balancing of HTTP and HTTPS traffic	• Load balancing of TCP, UDP, and TLS traffic where extreme performance is required	• Load balancing of HTTP, HTTPS, TCP, and SSL traffic
• Routes traffic to targets based on content of request • Provides advanced request routing targeted at the delivery of modern application architectures, including microservices and containers	• Routes traffic to targets based on IP protocol data • Can handle millions of requests per second while maintaining ultra-low latencies • Is optimized to handle sudden and volatile traffic patterns	• Load balancing across multiple EC2 instances
• Operates at the application layer (OSI model layer 7)	• Operates at the transport layer (OSI model layer 4)	• Operates at both the application and transport layers.

Elastic Load Balancing is available in three types:

- An *Application Load Balancer* operates at the application level (Open Systems Interconnection, or OSI, model layer 7). It routes traffic to targets—Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, Internet Protocol (IP) addresses, and Lambda functions—based on the content of the request. It is ideal for advanced load balancing of Hypertext Transfer Protocol (HTTP) and Secure HTTP (HTTPS) traffic. An Application Load Balancer provides advanced request routing that is targeted at delivery of modern application architectures, including microservices and container-based applications. An Application Load Balancer simplifies and improves the security of your application by ensuring that the latest Secure Sockets Layer/Transport Layer Security (SSL/TLS) ciphers and protocols are used at all times.

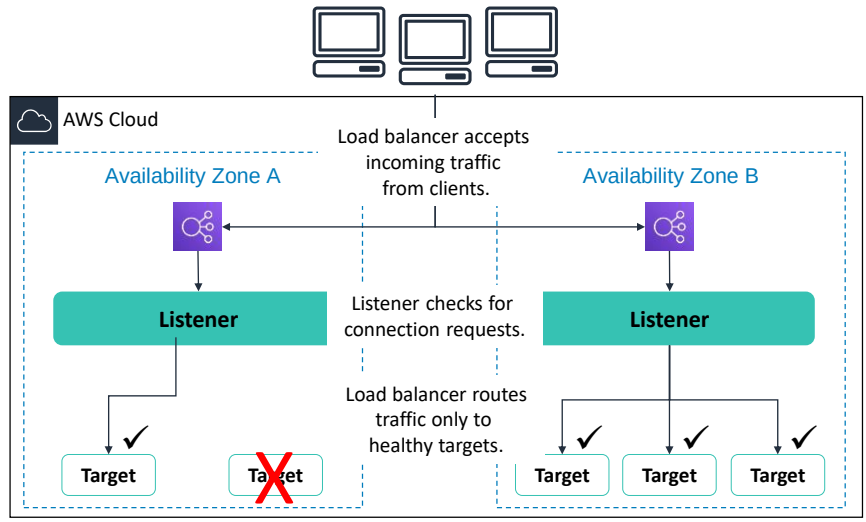
- A *Network Load Balancer* operates at the network transport level (OSI model layer 4), routing connections to targets—EC2 instances, microservices, and containers—based on IP protocol data. It works well for load balancing both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) traffic. A Network Load Balancer is capable of handling millions of requests per second while maintaining ultra-low latencies. A Network Load Balancer is optimized to handle sudden and volatile network traffic patterns.

- A *Classic Load Balancer* provides basic load balancing across multiple EC2 instances, and it operates at both the application level and network transport level. A Classic Load Balancer supports the load balancing of applications that use HTTP, HTTPS, TCP, and SSL. The Classic Load Balancer is an older implementation. When possible, AWS recommends that you use a dedicated Application Load Balancer or Network Load Balancer.

To learn more about the differences between the three types of load balancers, see *Product comparisons* on the Elastic Load Balancing Features page.

How Elastic Load Balancing works

- With Application Load Balancers and Network Load Balancers, you register targets in target groups, and route traffic to the target groups.
- With Classic Load Balancers, you register instances with the load balancer.



Load balancer performs health checks to monitor health of registered targets.

A load balancer accepts incoming traffic from clients and routes requests to its registered targets (such as EC2 instances) in one or more Availability Zones.

You configure your load balancer to accept incoming traffic by specifying one or more *listeners*. A listener is a process that checks for connection requests. It is configured with a protocol and port number for connections from clients to the load balancer. Similarly, it is configured with a protocol and port number for connections from the load balancer to the targets.

You can also configure your load balancer to perform *health checks*, which are used to monitor the health of the registered targets so that the load balancer only sends requests to the healthy instances. When the load balancer detects an unhealthy target, it stops routing traffic to that target. It then resumes routing traffic to that target when it detects that the target is healthy again.

There is a key difference in how the load balancer types are configured. With Application Load Balancers and Network Load Balancers, you register targets in *target groups*, and route traffic to the target groups. With Classic Load Balancers, you register instances with the load balancer.

Elastic Load Balancing use cases

Highly available and fault-tolerant applications

Containerized applications

Elasticity and scalability

Virtual private cloud (VPC)

Hybrid environments

Invoke Lambda functions over HTTP(S)

There are many reasons to use a load balancer:

- *Achieve high availability and better fault tolerance for your applications* – Elastic Load Balancing balances traffic across healthy targets in multiple Availability Zones. If one or more of your targets in a single Availability Zone are unhealthy, Elastic Load Balancing will route traffic to healthy targets in other Availability Zones. After the targets return to a healthy state, load balancing will automatically resume traffic to them.

- *Automatically load balance your containerized applications* – With enhanced container support for Elastic Load Balancing, you can now load balance across multiple ports on the same EC2 instance. You can also take advantage of deep integration with Amazon Elastic Container Service (Amazon ECS), which provides a fully-managed container offering. You only need to register a service with a load balancer, and Amazon ECS transparently manages the registration and de-registration of Docker containers. The load balancer automatically detects the port and dynamically reconfigures itself.

- *Automatically scale your applications* – Elastic Load Balancing works with Amazon CloudWatch and Amazon EC2 Auto Scaling to help you scale your applications to the demands of your customers. Amazon CloudWatch alarms can trigger auto scaling for your EC2 instance fleet when the latency of any one of your EC2 instances exceeds a preconfigured threshold. Amazon EC2 Auto Scaling then provisions new instances and your applications will be ready to serve the next customer request. The load balancer will register the EC2 instance and direct traffic to it as needed.

- *Use Elastic Load Balancing in your virtual private cloud (VPC)* – You can use Elastic Load Balancing to create a public entry point into your VPC, or to route request traffic between tiers of your application within your VPC. You can assign security groups to your load balancer to control which ports are open to a list of allowed sources. Because Elastic Load Balancing works with your VPC, all your existing network access control lists (network ACLs) and routing tables continue to provide additional network controls. When you create a load balancer in your VPC, you can specify whether the load balancer is public (default) or internal. If you select internal, you do not need to have an internet gateway to reach the load balancer, and the private IP addresses of the load balancer will be used in the load balancer's Domain Name System (DNS) record.

- *Enable hybrid load balancing* – Elastic Load Balancing enables you to load balance across AWS and on-premises resources by using the same load balancer. For example, if you must distribute application traffic across both AWS and on-premises resources, you can register all the resources to the same target group and associate the target group with a load balancer. Alternatively, you can use DNS-based weighted load balancing across AWS and on-premises resources by using two load balancers, with one load balancer for AWS and other load balancer for on-premises resources. You can also use hybrid load balancing to benefit separate applications where one application is in a VPC and the other application is in an on-premises location. Put the VPC targets in one target group and the on-premises targets in another target group, and then use content-based routing to route traffic to each target group.

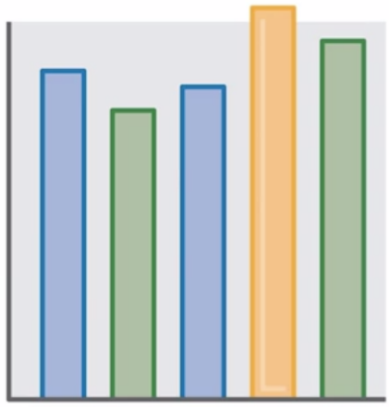
- *Invoking Lambda functions over HTTP(S)* – Elastic Load Balancing supports invoking Lambda functions to serve HTTP(S) requests. This enables users to access serverless applications from any HTTP client, including web browsers. You can register Lambda functions as targets and use the support for content-based routing rules in Application Load Balancers to route requests to different Lambda functions. You can use an Application Load Balancer as a common HTTP endpoint for applications that use servers and serverless computing. You can build an entire website by using Lambda functions, or combine EC2 instances, containers, on-premises servers, and Lambda functions to build applications.

Activity: Elastic Load Balancing

You must support traffic to a containerized application.	Application Load Balancer
You have extremely spiky and unpredictable TCP traffic.	Network Load Balancer
You need simple load balancing with multiple protocols.	Classic Load Balancer
You need to support a static or Elastic IP address, or an IP target outside a VPC.	Network Load Balancer
You need a load balancer that can handle millions of requests per second while maintaining low latencies.	Network Load Balancer
You must support HTTPS requests.	Application Load Balancer

For this activity, name the load balancer you would use for the given scenario.

Load balancer monitoring



- **Amazon CloudWatch metrics** – Used to verify that the system is performing as expected and creates an alarm to initiate an action if a metric goes outside an acceptable range.
- **Access logs** – Capture detailed information about requests sent to your load balancer.
- **AWS CloudTrail logs** – Capture the who, what, when, and where of API interactions in AWS services.

You can use the following features to monitor your load balancers, analyze traffic patterns, and troubleshoot issues with your load balancers and targets:

- *Amazon CloudWatch metrics* – Elastic Load Balancing publishes data points to Amazon CloudWatch for your load balancers and your targets. CloudWatch enables you to retrieve statistics about those data points as an ordered set of time series data, known as metrics. You can use metrics to verify that your system is performing as expected. For example, you can create a CloudWatch alarm to monitor a specified metric and initiate an action (such as sending a notification to an email address) if the metric goes outside what you consider an acceptable range.

- *Access logs* – You can use access logs to capture detailed information about the requests that were made to your load balancer and store them as log files in Amazon Simple Storage Service (Amazon S3). You can use these access logs to analyze traffic patterns and to troubleshoot issues with your targets or backend applications.

- *AWS CloudTrail logs* – You can use AWS CloudTrail to capture detailed information about the calls that were made to the Elastic Load Balancing application programming interface (API) and store them as log files in Amazon S3. You can use these CloudTrail logs to determine who made the call, what calls were made, when the call was made, the source IP address of where the call came from, and so on.

ELB takeaways

- Elastic Load Balancing distributes incoming application or network traffic across multiple targets in one or more Availability Zones.
- Elastic Load Balancing supports three types of load balancers:
  - Application Load Balancer
  - Network Load Balancer
  - Classic Load Balancer
- ELB offers instance health checks, security, and monitoring.

Some key takeaways from this section of the module include:

- Elastic Load Balancing distributes incoming application or network traffic across multiple targets (such as Amazon EC2 instances, containers, IP addresses, and Lambda functions) in one or more Availability Zones.
- Elastic Load Balancing supports three types of load balancers:
  - Application Load Balancer
  - Network Load Balancer
  - Classic Load Balancer
- Elastic Load Balancing offers several monitoring tools for continuous monitoring and logging for auditing and analytics.

This week – Scalable Architectures

- High availability and Scaling
- Automated Scaling and Monitoring on AWS
  - ☐Elastic Load Balancing
  - ☐**Amazon CloudWatch**
  - ☐Amazon EC2 Auto Scaling

SWINBURNE UNIVERSITY OF TECHNOLOGY

Monitoring AWS resources

To use AWS efficiently, you need insight into your AWS resources:

- How do you know when you should **launch more Amazon EC2 instances**?
- Is your **application's performance or availability** being affected by a lack of sufficient capacity?
- How much of your infrastructure is actually **being used**?

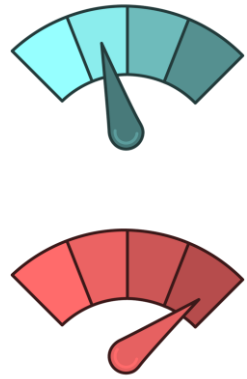
To use AWS efficiently, you need insight into your AWS resources.

For example, you might want to know:
- When you should launch more Amazon EC2 instances?
- If your application’s performance or availability is being affected by a lack of sufficient capacity?
- How much of your infrastructure is actually being used?

How do you capture this information?

Amazon CloudWatch

Amazon CloudWatch



- Monitors –
  - AWS resources
  - Applications that run on AWS
- Collects and tracks –
  - Standard metrics
  - Custom metrics
- Alarms –
  - Send notifications to an Amazon SNS topic
  - Perform Amazon EC2 Auto Scaling or Amazon EC2 actions
- Events –
  - Define rules to match changes in AWS environment and route these events to one or more target functions or streams for processing

You can capture this information with Amazon CloudWatch.

Amazon CloudWatch is a monitoring and observability service that is built for DevOps engineers, developers, site reliability engineers (SRE), and IT managers. CloudWatch monitors your AWS resources (and the applications that you run on AWS) in real time. You can use CloudWatch to collect and track metrics, which are variables that you can measure for your resources and applications.

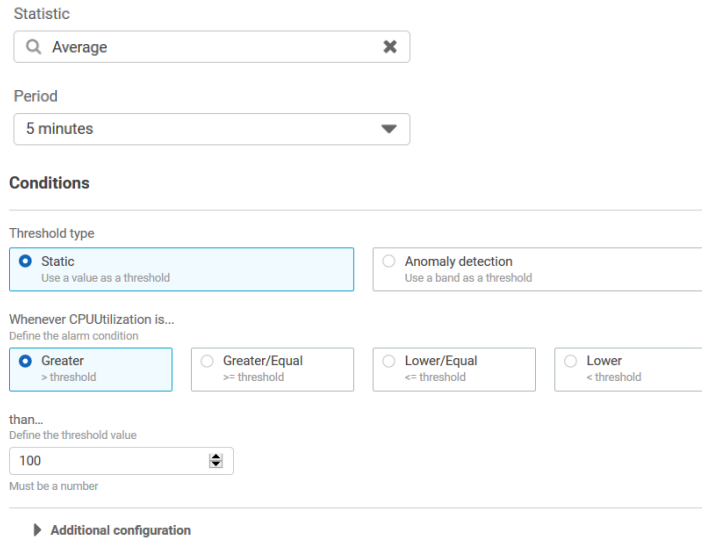
You can create an alarm to monitor any Amazon CloudWatch metric in your account and use the alarm to automatically send a notification to an Amazon Simple Notification Service (Amazon SNS) topic or perform an Amazon EC2 Auto Scaling or Amazon EC2 action. For example, you can create alarms on the CPU utilization of an EC2 instance, Elastic Load Balancing request latency, Amazon DynamoDB table throughput, Amazon Simple Queue Service (Amazon SQS) queue length, or even the charges on your AWS bill. You can also create an alarm on custom metrics that are specific to your custom applications or infrastructure.

You can also use Amazon CloudWatch Events to define rules that match incoming events (or changes in your AWS environment) and route them to targets for processing. Targets can include Amazon EC2 instances, AWS Lambda functions, Kinesis streams, Amazon ECS tasks, Step Functions state machines, Amazon SNS topics, Amazon SQS queues, and built-in targets. CloudWatch Events becomes aware of operational changes as they occur. CloudWatch Events responds to these operational changes and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.

With CloudWatch, you gain system-wide visibility into resource utilization, application performance, and operational health. There is no upfront commitment or minimum fee; you simply pay for what you use. You are charged at the end of the month for what you use.

CloudWatch alarms

- Create alarms based on –
  - Static threshold
  - Anomaly detection
  - Metric math expression
- Specify –
  - Namespace
  - Metric
  - Statistic
  - Period
  - Conditions
  - Additional configuration
  - Actions



You can create a CloudWatch alarm that watches a single CloudWatch metric or the result of a math expression based on CloudWatch metrics. You can create a CloudWatch alarm based on a static threshold, anomaly detection, or a metric math expression.

When you create an alarm based on a static threshold, you choose a CloudWatch metric for the alarm to watch and the threshold for that metric. The alarm goes to ALARM state when the metric breaches the threshold for a specified number of evaluation periods.

For an alarm based on a static threshold, you must specify the:

- *Namespace* – A namespace contains the CloudWatch metric that you want, for example, *AWS/EC2*.
- *Metric* – A metric is the variable you want to measure, for example, *CPU Utilization*.
- *Statistic* – A statistic can be an average, sum, minimum, maximum, sample count, a predefined percentile, or a custom percentile.
- *Period* – A period is the evaluation period for the alarm. When the alarm is evaluated, each period is aggregated into one data point.
- *Conditions* – When you specify the conditions for a static threshold, you specify whenever the metric is *Greater*, *Greater or Equal*, *Lower or Equal*, or *Lower* than the threshold value, and you also specify the threshold value.
- *Additional configuration information* – This includes the number of data points within the evaluation period that must be breached to trigger the alarm, and how CloudWatch should treat missing data when it evaluates the alarm.
- *Actions* – You can choose to send a notification to an Amazon SNS topic, or to perform an Amazon EC2 Auto Scaling action or Amazon EC2 action.

For more information on creating CloudWatch alarms, see the topics under Using Alarms in the AWS Documentation.

Activity: Amazon CloudWatch

 Amazon EC2	If average CPU utilization is > 60% for 5 minutes…	Correct!
 Amazon RDS	If the number of simultaneous connections is > 10 for 1 minute…	Correct!
 Amazon S3	If the maximum bucket size in bytes is around 3 for 1 day…	Incorrect. *Around* is not a threshold option. You must specify a threshold of >, >=, <=, or <.
 Elastic Load Balancing	If the number of healthy hosts is < 5 for 10 minutes…	Correct!
 Amazon Elastic Block Store	If the volume of read operations is > 1,000 for 10 seconds…	Incorrect. You must specify a statistic (for example, *average volume*).

For this activity, see if you can identify which are correct CloudWatch alarms. For the ones that are incorrect, see if you can identify the error.

Section 2 key takeaways

- Amazon CloudWatch helps you monitor your AWS resources—and the applications that you run on AWS—in real time.
- CloudWatch enables you to –
  - Collect and track standard and custom metrics.
  - Set alarms to automatically send notifications to SNS topics, or perform Amazon EC2 Auto Scaling or Amazon EC2 actions.
  - Define rules that match changes in your AWS environment and route these events to targets for processing.

Some key takeaways from this section of the module include:

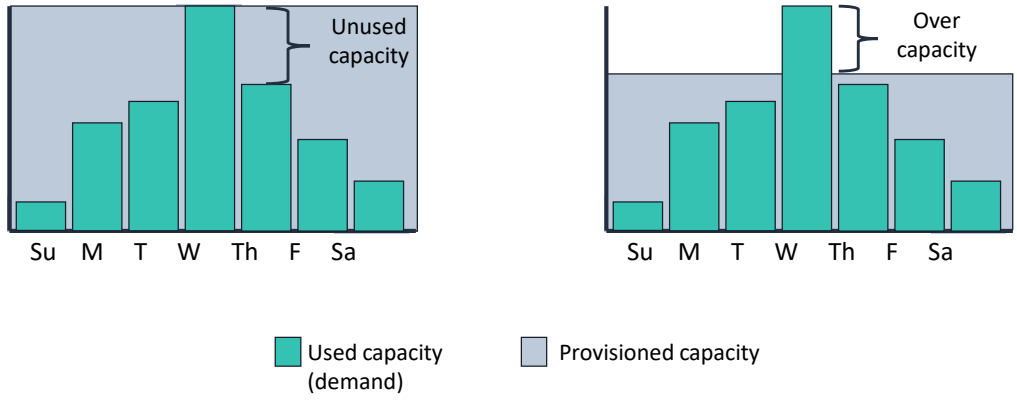
- Amazon CloudWatch helps you monitor your AWS resources—and the applications that you run on AWS—in real time.
- CloudWatch enables you to –
  - Collect and track standard and custom metrics.
  - Set alarms to automatically send notifications to SNS topics or perform Amazon EC2 Auto Scaling or Amazon EC2 actions based on the value of the metric or expression relative to a threshold over a number of time periods.
  - Define rules that match changes in your AWS environment and route these events to targets for processing.

This week – Scalable Architectures

- High availability and Scaling
- Automated Scaling and Monitoring on AWS
  - ☐Elastic Load Balancing
  - ☐Amazon CloudWatch
  - ☐ Amazon EC2 Auto Scaling

SWINBURNE UNIVERSITY OF TECHNOLOGY

Why is scaling important?

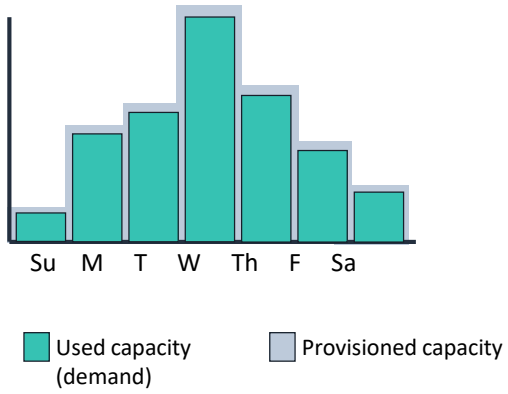


Scaling is the ability to increase or decrease the compute capacity of your application. To understand why scaling is important, consider this example of a workload that has varying resource requirements. In this example, the most resource capacity is required on Wednesday, and the least resource capacity is required on Sunday.

One option is to allocate more than enough capacity so you can always meet your highest demand—in this case, Wednesday. However, this situation means that you are running resources that will be underutilized most days of the week. With this option, your costs are not optimized.

Another option is to allocate less capacity to reduce costs. This situation means that you are under capacity on certain days. If you don't solve your capacity problem, your application could underperform or potentially even become unavailable for users.

Amazon EC2 Auto Scaling



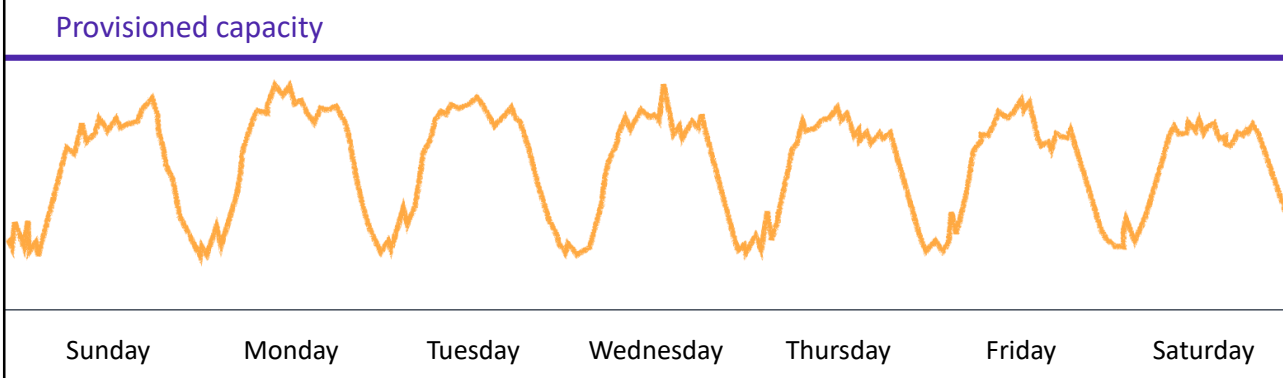
- Helps you maintain application availability
- Enables you to automatically add or remove EC2 instances according to conditions that you define
- Detects impaired EC2 instances and unhealthy applications, and replaces the instances without your intervention
- Provides several scaling options – Manual, scheduled, dynamic or on-demand, and predictive

In the cloud, because computing power is a programmatic resource, you can take a flexible approach to scaling. Amazon EC2 Auto Scaling is an AWS service that helps you maintain application availability and enables you to automatically add or remove EC2 instances according to conditions you define. You can use the fleet management features of EC2 Auto Scaling to maintain the health and availability of your fleet.

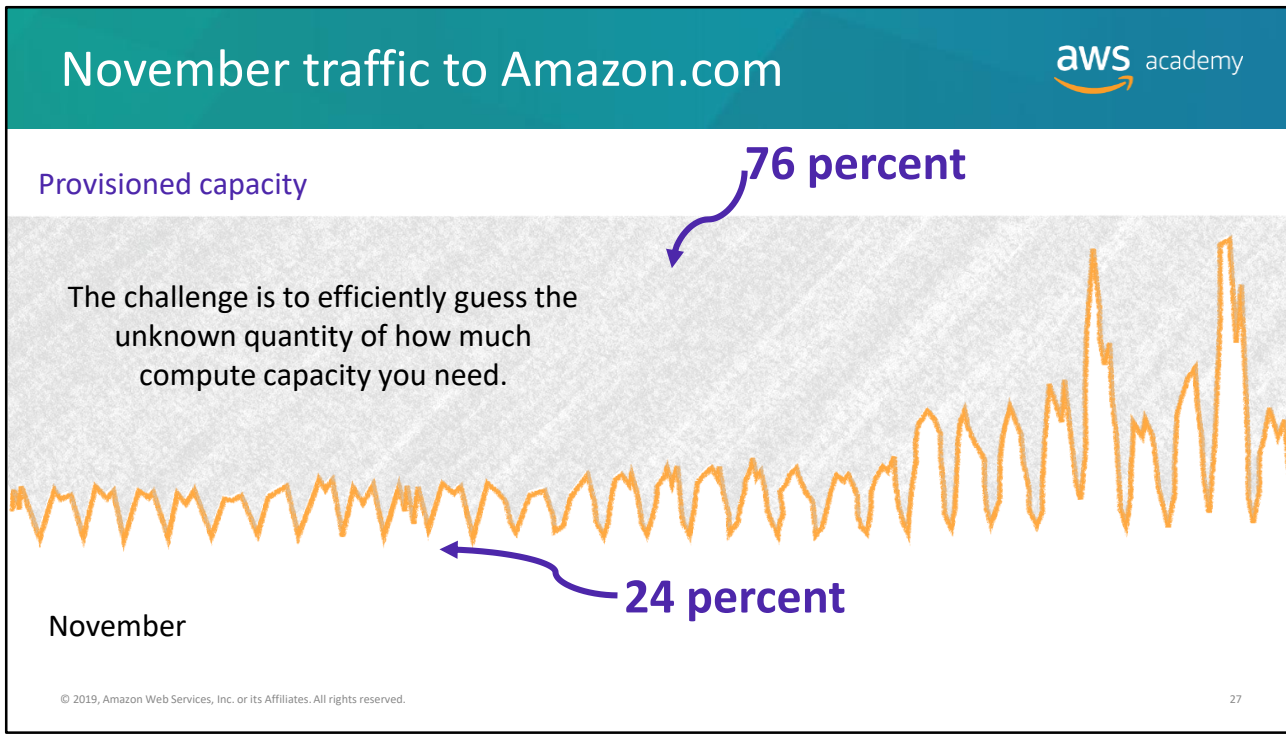
Amazon EC2 Auto Scaling provides several ways to adjust scaling to best meet the needs of your applications. You can add or remove EC2 instances manually, on a schedule, in response to changing demand, or in combination with AWS Auto Scaling for predictive scaling. Dynamic scaling and predictive scaling can be used together to scale faster.

To learn more about Amazon EC2 Auto Scaling, see the Amazon EC2 Auto Scaling product page.

Typical weekly traffic at Amazon.com

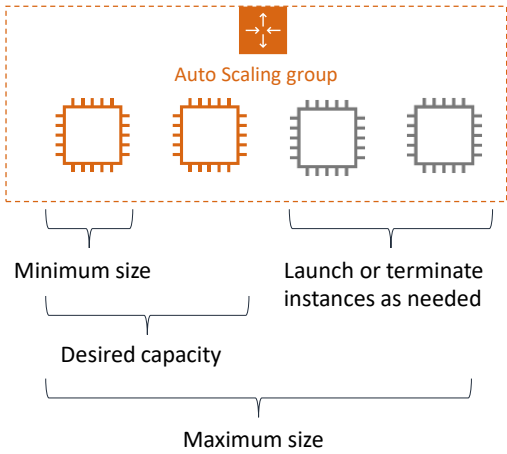


Automatic scaling is useful for predictable workloads—for example, the weekly traffic at the retail company Amazon.com.



Automatic scaling is also useful for dynamic on-demand scaling. Amazon.com experiences a seasonal peak in traffic in November (on Black Friday and Cyber Monday, which are days at the end of November when US retailers hold major sales). If Amazon provisions a fixed capacity to accommodate the highest use, 76 percent of the resources are idle for most of the year. Capacity scaling is necessary to support the fluctuating demands for service. Without scaling, the servers could crash due to saturation, and the business would lose customer confidence.

Auto Scaling groups



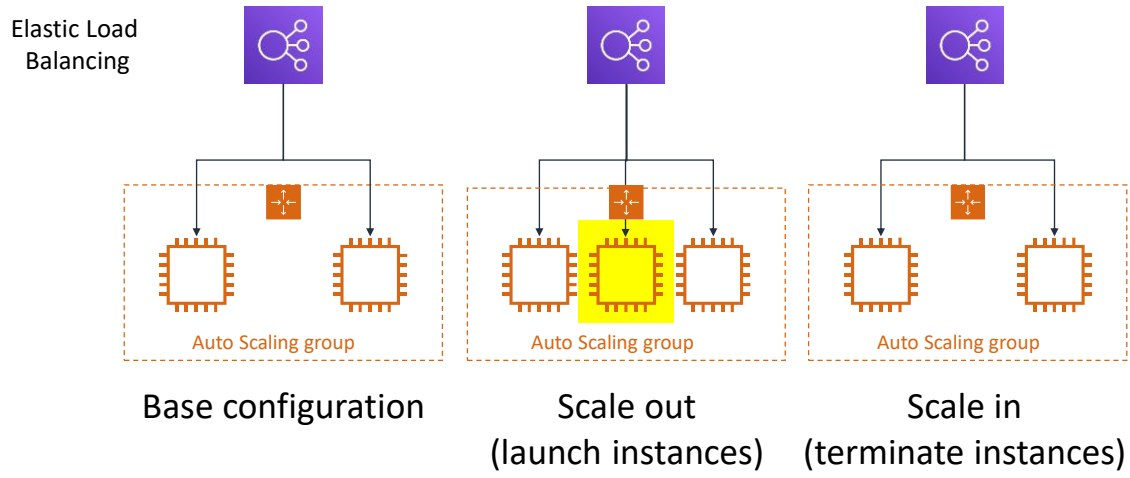
An **Auto Scaling group** is a collection of EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management.

An Auto Scaling group is a collection of Amazon EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management. The size of an Auto Scaling group depends on the number of instances you set as the *desired capacity*. You can adjust its size to meet demand, either manually or by using automatic scaling.

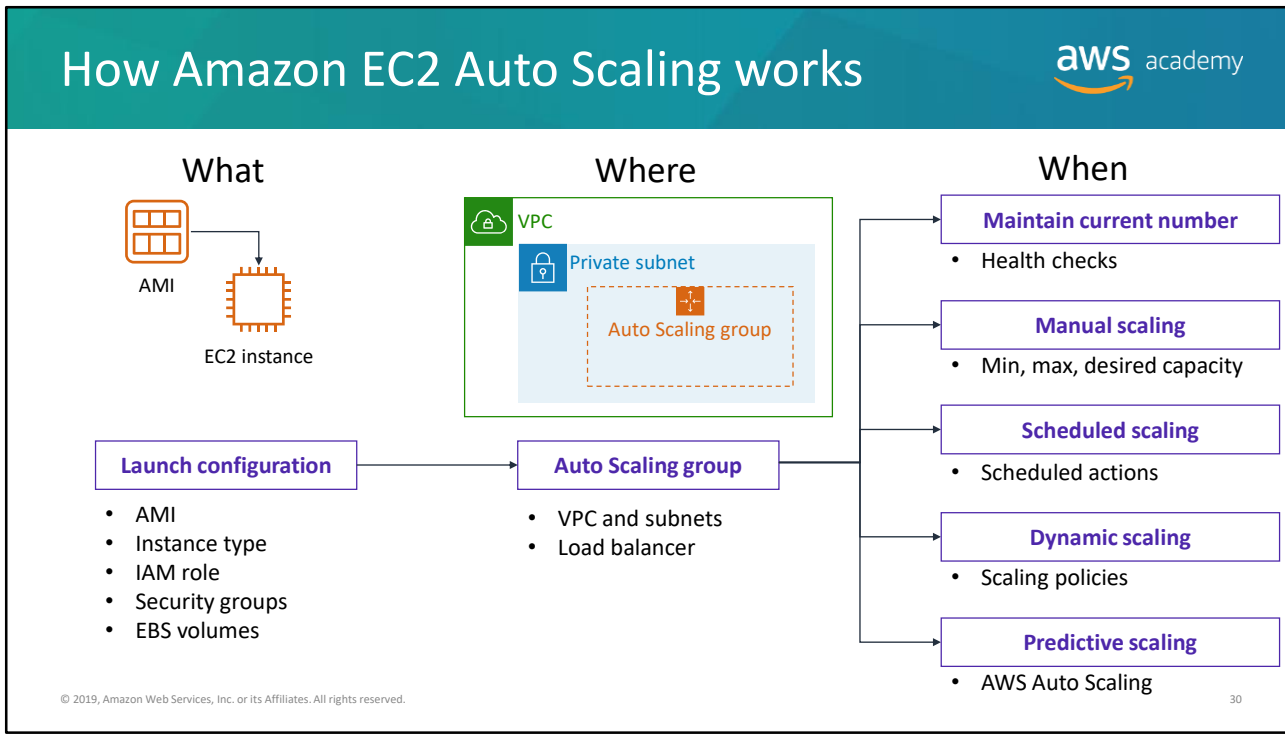
You can specify the minimum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling is designed to prevent your group from going below this size. You can specify the maximum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling is designed to prevent your group from going above this size. If you specify the desired capacity, either when you create the group or at any time afterwards, Amazon EC2 Auto Scaling is designed to adjust the size of your group so it has the specified number of instances. If you specify scaling policies, then Amazon EC2 Auto Scaling can launch or terminate instances as demand on your application increases or decreases.

For example, this Auto Scaling 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.

Scaling out versus scaling in



With Amazon EC2 Auto Scaling, launching instances is referred to as *scaling out*, and terminating instances is referred to as *scaling in*.



To launch EC2 instances, an Auto Scaling group uses a launch configuration, which is an instance configuration template. You can think of a launch configuration as *what* you are scaling. When you create a launch configuration, you specify information for the instances. The information you specify includes the ID of the Amazon Machine Image (AMI), the instance type, AWS Identity and Access Management (IAM) role, additional storage, one or more security groups, and any Amazon Elastic Block Store (Amazon EBS) volumes.

You define the minimum and maximum number of instances and desired capacity of your Auto Scaling group. Then, you launch it into a subnet within a VPC (you can think of this as *where* you are scaling). Amazon EC2 Auto Scaling integrates with Elastic Load Balancing to enable you to attach one or more load balancers to an existing Auto Scaling group. After you attach the load balancer, it automatically registers the instances in the group and distributes incoming traffic across the instances.

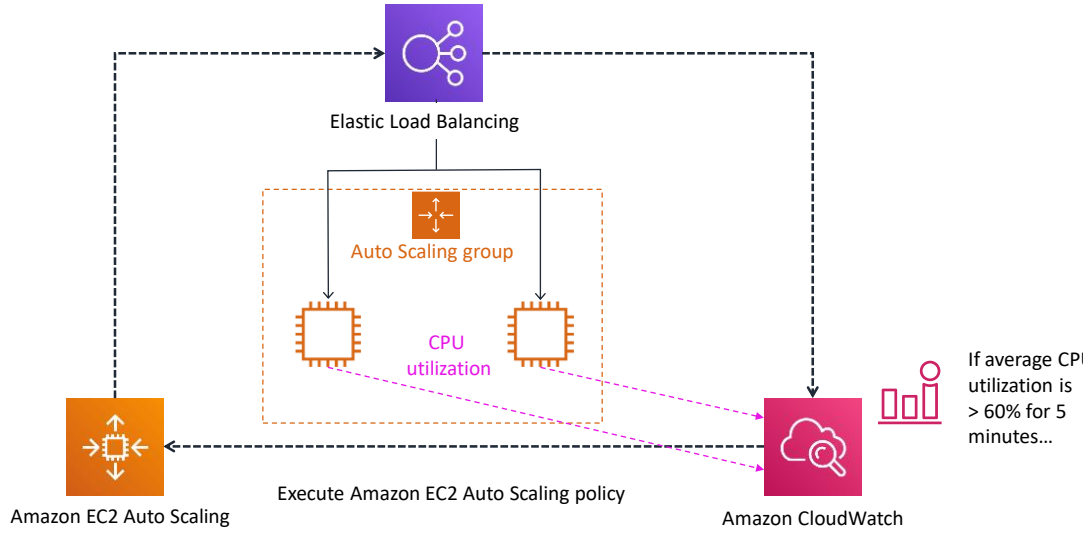
Finally, you specify *when* you want the scaling event to occur. You have many scaling options:

- *Maintain current instance levels at all times* – You can configure your Auto Scaling group to maintain a specified number of running instances at all times. To maintain the current instance levels, Amazon EC2 Auto Scaling performs a periodic health check on running instances in an Auto Scaling group. When Amazon EC2 Auto Scaling finds an unhealthy instance, it terminates that instance and launches a new one.
- Manual scaling – With manual scaling, you specify only the change in the maximum, minimum, or desired capacity of your Auto Scaling group.

- Scheduled scaling – With scheduled scaling, scaling actions are performed automatically as a function of date and time. This is useful for predictable workloads when you know exactly when to increase or decrease the number of instances in your group. For example, say that 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 actions based on the predictable traffic patterns of your web application. To implement scheduled scaling, you create a scheduled action.
- Dynamic, on-demand scaling – A more advanced way to scale your resources enables you to define parameters that control the scaling process. For example, you have a web application that currently runs on two instances and you want the CPU utilization of the Auto Scaling group to stay close to 50 percent when the load on the application changes. This option is useful for scaling in response to changing conditions, when you don't know when those conditions will change. Dynamic scaling gives you extra capacity to handle traffic spikes without maintaining an excessive amount of idle resources. You can configure your Auto Scaling group to scale automatically to meet this need. The scaling policy type determines how the scaling action is performed. You can use Amazon EC2 Auto Scaling with Amazon CloudWatch to trigger the scaling policy in response to an alarm.
- Predictive scaling – You can use Amazon EC2 Auto Scaling with AWS Auto Scaling to implement predictive scaling, where your capacity scales based on predicted demand. Predictive scaling uses data that is collected from your actual EC2 usage, and the data is further informed by billions of data points that are drawn from our own observations. AWS then uses well-trained machine learning models to predict your expected traffic (and EC2 usage), including daily and weekly patterns. The model needs at least 1 day of historical data to start making predictions. It is re-evaluated every 24 hours to create a forecast for the next 48 hours. The prediction process produces a scaling plan that can drive one or more groups of automatically scaled EC2 instances.

To learn more about these options, see Scaling the Size of Your Auto Scaling Group in the AWS Documentation.

Implementing dynamic scaling



One common configuration for implementing dynamic scaling is to create a CloudWatch alarm that is based on performance information from your EC2 instances or load balancer. When a performance threshold is breached, a CloudWatch alarm triggers an automatic scaling event that either scales out or scales in EC2 instances in the Auto Scaling group.

To understand how it works, consider this example:

- You create an Amazon CloudWatch alarm to monitor CPU utilization across your fleet of EC2 instances and execute automatic scaling policies if the average CPU utilization across the fleet goes above 60 percent for 5 minutes.
- Amazon EC2 Auto Scaling instantiates a new EC2 instance into your Auto Scaling group based on the launch configuration that you create.
- After the new instance is added, Amazon EC2 Auto Scaling makes a call to Elastic Load Balancing to register the new EC2 instance in that Auto Scaling group.
- Elastic Load Balancing then performs the required health checks and starts distributing traffic to that instance. Elastic Load Balancing routes traffic between EC2 instances and feeds metrics to Amazon CloudWatch.

Amazon CloudWatch, Amazon EC2 Auto Scaling, and Elastic Load Balancing work well individually. Together, however, they become more powerful and increase the control and flexibility over how your application handles customer demand.

AWS Auto Scaling

AWS Auto Scaling

- Monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost
- Provides a simple, powerful user interface that enables you to build scaling plans for resources, including –
  - Amazon EC2 instances and Spot Fleets
  - Amazon Elastic Container Service (Amazon ECS) Tasks
  - Amazon DynamoDB tables and indexes
  - Amazon Aurora Replicas

So far, you learned about scaling EC2 instances with Amazon EC2 Auto Scaling. You also learned that you can use Amazon EC2 Auto Scaling with AWS Auto Scaling to perform predictive scaling.

AWS Auto Scaling is a separate service that monitors your applications. It automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost. The service provides a simple, powerful user interface that enables you to build scaling plans for resources, including:

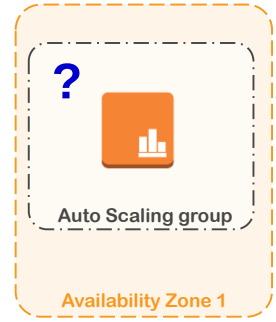
- Amazon EC2 instances and Spot Fleets
- Amazon Elastic Container Service (Amazon ECS) tasks
- Amazon DynamoDB tables and indexes
- Amazon Aurora Replicas

If you are already using Amazon EC2 Auto Scaling to dynamically scale your EC2 instances, you can now use it with AWS Auto Scaling to scale additional resources for other AWS services.

To learn more information about AWS Auto Scaling, see AWS Auto Scaling.

How Do You Decide On Minimum Capacity Size?

- Auto Scaling group defines:
  - Desired capacity
  - Minimum capacity
  - Maximum capacity
- What would be a good **minimum** capacity to set it to?
- What would be a good **maximum** capacity to set it to?



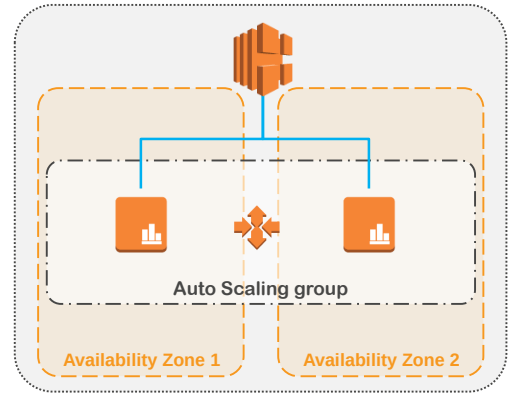
You don't have to specify the desired capacity because minimum capacity is the initial desired capacity. You pay for what you use; therefore, you don't need to launch more instances than you need and auto scale based on demand.

Deciding on the minimum capacity size depends on the type of applications that you are running. Here are some things to consider:

- If you are processing batches that run once a week, you might want to set the minimum to zero.
- Remember that it takes minutes to launch a new instance (depending on the complexity of your launch configuration). You may not be able to afford zero minimum capacity to start with.

How Do You Decide On Minimum Capacity Size?

What about high availability?



Minimum = two instances (# of AZs)

Desired capacity = two instances (Min.)

If you require at least one instance per AZ to start with, the minimum capacity size should be set to the number of AZs.

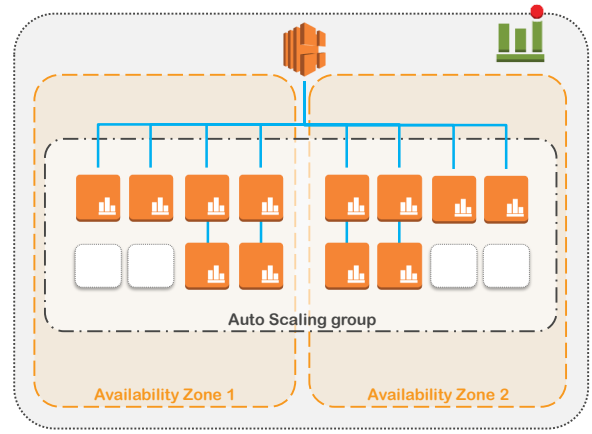
Maximum Capacity Size And Auto Scaling

Scenario:

Auto Scaling group
- Minimum = 2
- Maximum = **12**

Auto Scaling policy
- When CPU utilization is greater than 60%
- Add 100% of group = **double the capacity**

CPU utilization triggers the alarm: capacity is doubled until CPU utilization drops below 60% or max capacity is reached.



This scenario demonstrates how maximum capacity size plays a role in the way Auto Scaling works.

As a discussion, you set the Auto Scaling policy to **double** the current group capacity when the CPU utilization becomes greater than 60%. You start with a minimum of two instances.

1. The first time the alarm triggers the Auto Scaling policy to take effect, the desired capacity becomes four (2x2).
2. Load keeps increasing and when CPU utilization reaches greater than 60%, the desired capacity becomes eight (2x4).
3. The next time the alarm gets triggered, you can no longer double the current group capacity because the maximum capacity is set to 12.
4. Auto Scaling adds four more instances equally balanced across the two AZs.

Selecting a reasonable maximum capacity size depends on your **application** and possibly a **budget**. You pay for what you use; therefore, your organization may limit you from running more than a certain number of instances. Even if you don't have a budget restriction, there is no single answer.

Auto Scaling key takeaways

- Scaling enables you to respond quickly to changes in resource needs.
- Amazon EC2 Auto Scaling maintains application availability by automatically adding or removing EC2 instances.
- An Auto Scaling group is a collection of EC2 instances.
- A launch configuration is an instance configuration template.
- Dynamic scaling uses Amazon EC2 Auto Scaling, CloudWatch, and Elastic Load Balancing.
- AWS Auto Scaling is a separate service from Amazon EC2 Auto Scaling.

Some key takeaways from this section of the module include:

- Scaling enables you to respond quickly to changes in resource needs.
- Amazon EC2 Auto Scaling helps you maintain application availability, and enables you to automatically add or remove EC2 instances according to your workloads.
- An Auto Scaling group is a collection of EC2 instances.
- A launch configuration is an instance configuration template.
- You can implement dynamic scaling with Amazon EC2 Auto Scaling, Amazon CloudWatch, and Elastic Load Balancing.

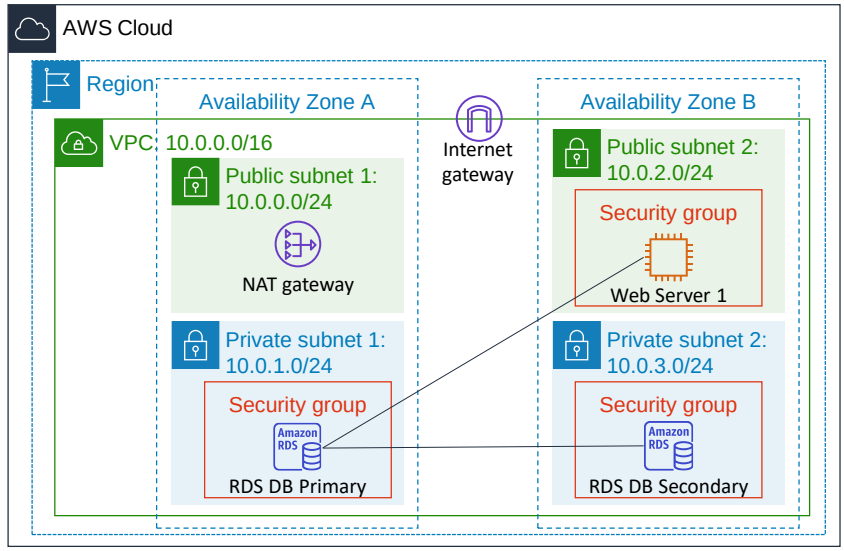
AWS Auto Scaling is a separate service that monitors your applications, and it automatically adjusts capacity for the following resources:

- Amazon EC2 instances and Spot Fleets
- Amazon ECS tasks
- Amazon DynamoDB tables and indexes
- Amazon Aurora Replicas

ACF Lab 6:
Scale and Load Balance Your Architecture

You will now complete Lab 6: Scale and Load Balance Your Architecture.

ACF Lab 6: Scenario (optional – not assessed)



In this lab, you will use Elastic Load Balancing and Amazon EC2 Auto Scaling to load balance and scale your infrastructure. You will start with the given infrastructure.

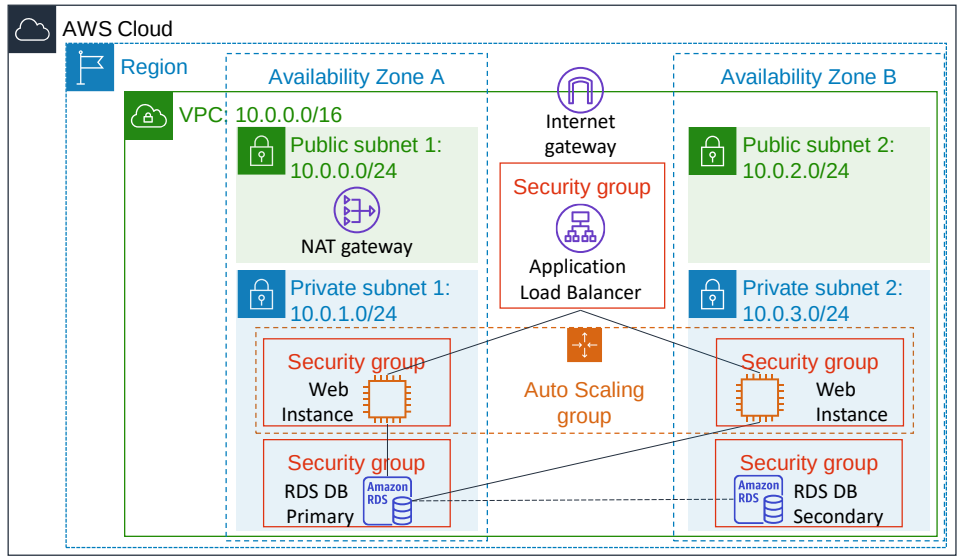
Lab 6: Tasks

- Create an Amazon Machine Image (AMI) from a running instance.
- Create an Application Load Balancer.
- Create a launch configuration and an Auto Scaling group.
- Automatically scale new instances within a private subnet.
- Create Amazon CloudWatch alarms and monitor performance of your infrastructure.

In this lab, you will complete the following tasks:

- Create an Amazon Machine Image (AMI) from a running instance.
- Create an Application Load Balancer.
- Create a launch configuration and an Auto Scaling group.
- Automatically scale new instances within a private subnet.
- Create Amazon CloudWatch alarms and monitor performance of your infrastructure.

Lab 6: Final product



The diagram summarizes what you will have built after you complete the lab.

ACA Mod 3 Lab: Making Your Environment Highly Available

~ 60 minutes

This lab will provide you with a basic overview of launching, resizing, managing, and monitoring an Amazon EC2 instance. You will use Amazon EC2, VPC, ELB, NAT Gateway and Amazon Machine Images, or AMIs, to deploy a highly available PHP web application. This lab includes Automatic Scaling.

ACA Mod 3 Lab: Scenario

In this lab, you will take a web application and make it **highly available**.

The lab will use these services:

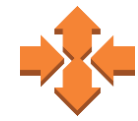

Amazon EC2


Amazon VPC


Amazon EC2 AMI

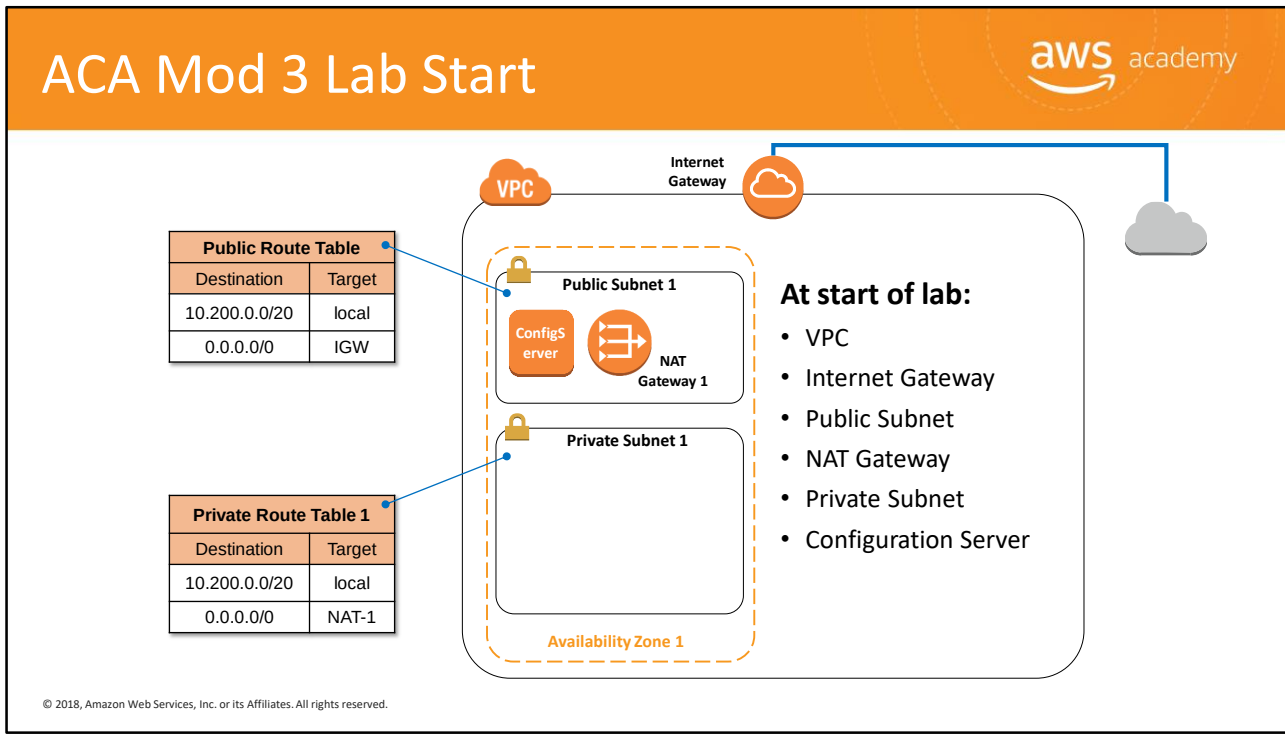

Elastic Load Balancing


NAT Gateway


Automatic Scaling

This lab demonstrates how to make a highly available web application.

You will use several AWS services to implement the application, including Amazon EC2, Amazon VPC, Amazon EC2 AMI, Elastic Load Balancing, NAT Gateway, and Automatic Scaling.



At the start of the lab, you will be given:

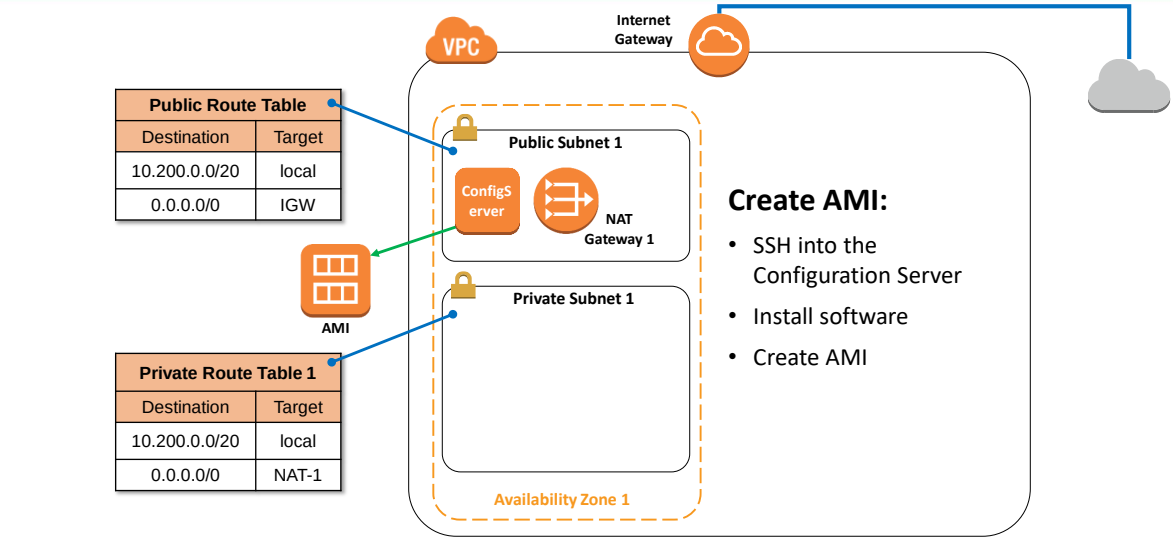
- A VPC in one Availability Zone
- An internet gateway
- A public subnet with a route table that sends internet-bound traffic to the internet gateway
- A NAT Gateway for connecting private Amazon EC2 instances to the internet
- A private subnet
- And an Amazon EC2 “Configuration Server” that will be used to install the web application

The lab consists of eight tasks:

- Creating an image of an existing Amazon EC2 instance and use it to launch new instances.
- Expanding an Amazon VPC to additional Availability Zones.
- Creating VPC subnets and route tables.
- Creating an AWS NAT Gateway.
- Creating a load balancer.
- And create an Auto Scaling group.

The lab is expected to take approximately 60 minutes.

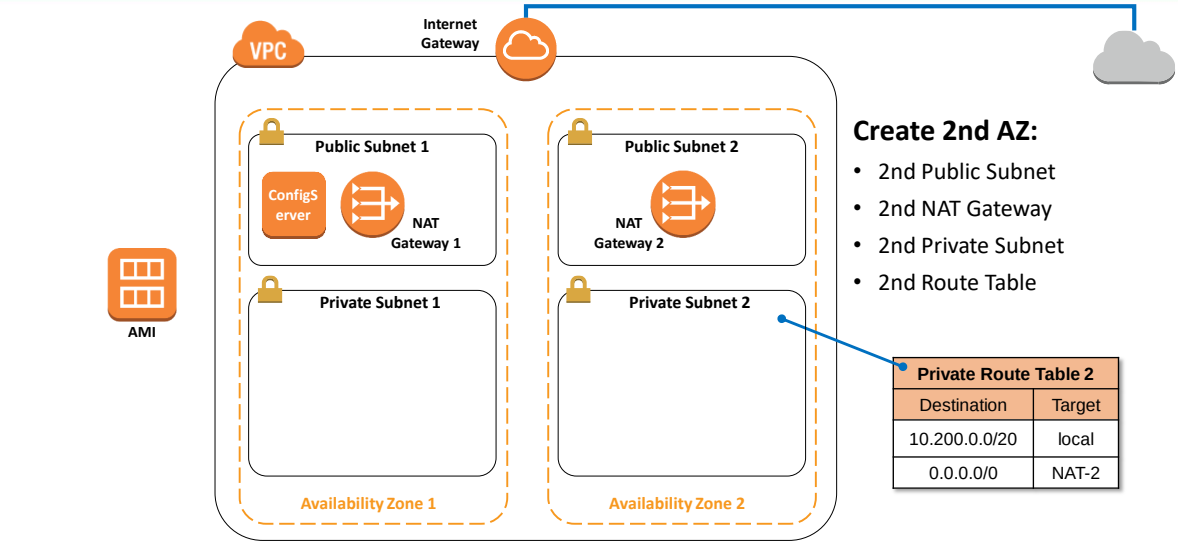
ACA Mod 3 Lab: Configure, Create AMI



In this lab, you will configure the Amazon EC2 instance to run your web application.

You will then create an Amazon Machine Image, or (AMI) of the instance, which can be used to launch more instances.

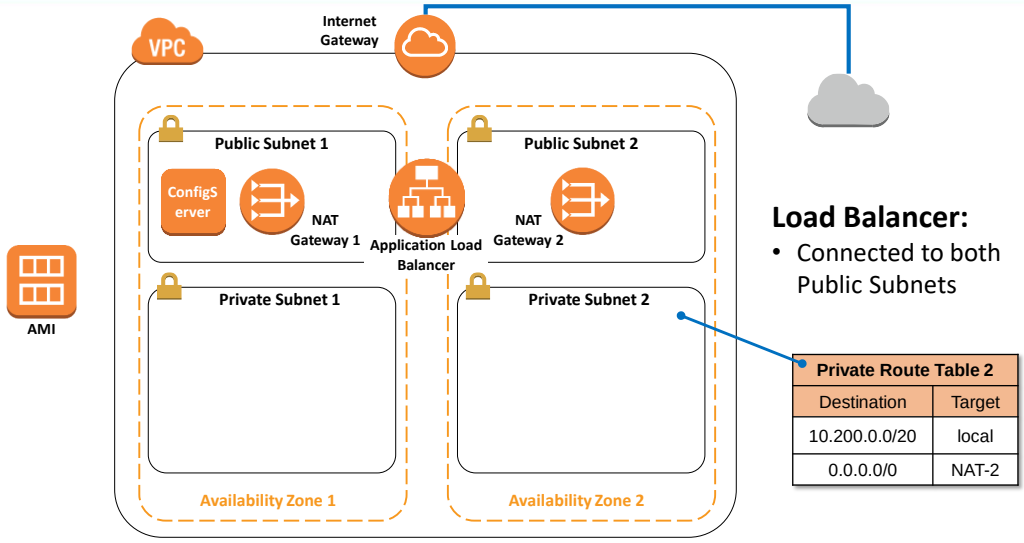
ACA Mod 3 Lab: Create Second AZ



To make the application highly available, you will create:

- A second public subnet
- A second NAT Gateway—if Availability Zone 1 fails, this will continue providing access
- A second private subnet
- And a second route table that sends internet-bound traffic through the second NAT Gateway

ACA Mod 3 Create Load Balancer



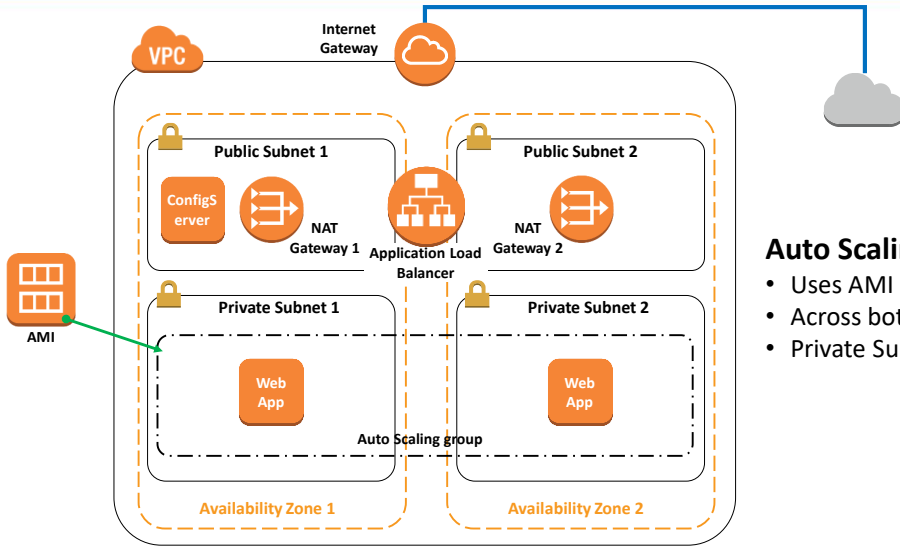
Load Balancer:

- Connected to both Public Subnets

Private Route Table 2	
Destination	Target
10.200.0.0/20	local
0.0.0.0/0	NAT-2

You will launch an Application load balancer to receive incoming requests.

ACA Mod 3 Lab: Auto Scaling Group

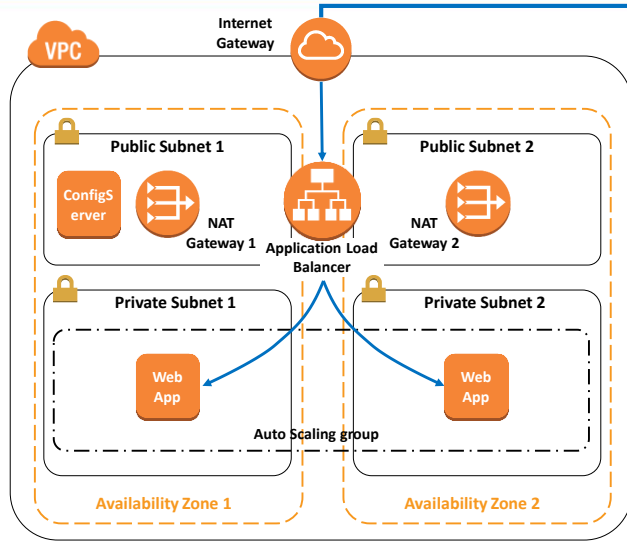


Auto Scaling

- Uses AMI
- Across both
- Private Subnets

Now, launch an Auto Scaling group in the private subnets. Auto Scaling will use the AMI that you created earlier.

ACA Mod 3 Lab: Incoming Request

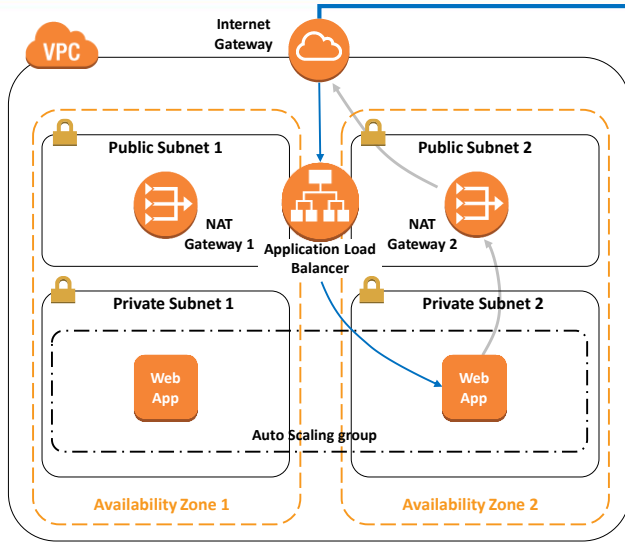


Incoming Request:

- Comes into Load Balancer
- Distributed amongst instances

Incoming requests will come into the load balancer, which will forward each request to one of the Amazon EC2 instances.

ACA Mod 3 Lab: Use NAT Gateway



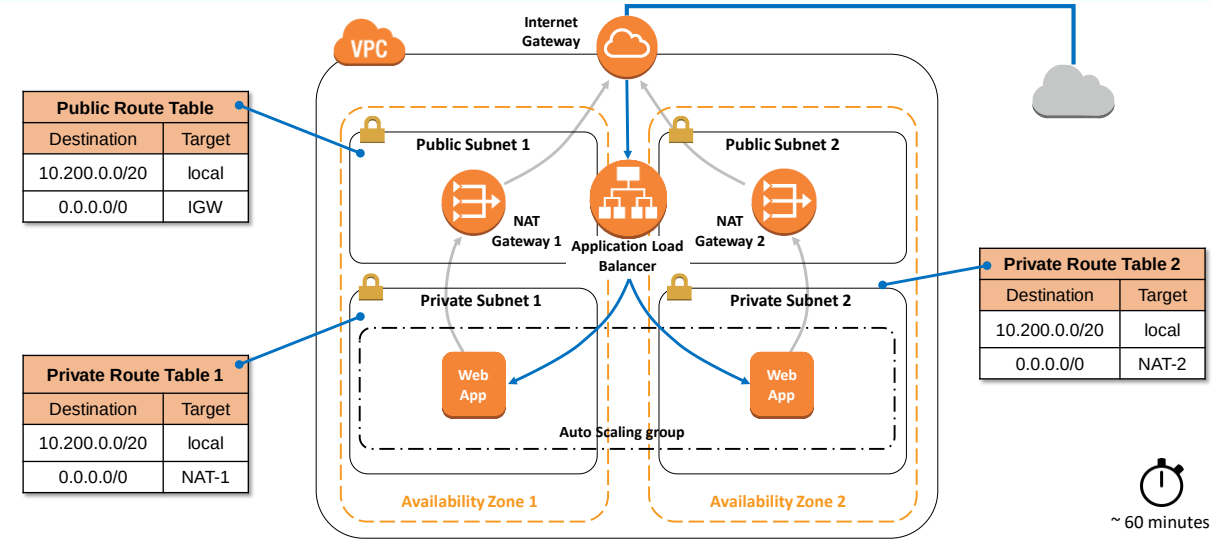
App queries freegeoip.net:

- Web application accesses Internet
- Request goes out NAT Gateway
- Web application responds to user

The web application requires internet access to query information from freegeoip.net.

The web application servers are in private subnets, so these queries will be sent via the NAT Gateway in the same Availability Zone.

ACA Mod 3 Lab: Final Product



The final product is a highly available application.

Next week

- Highly Available Architectures II
- Cloud Formation

SWINBURNE UNIVERSITY OF TECHNOLOGY