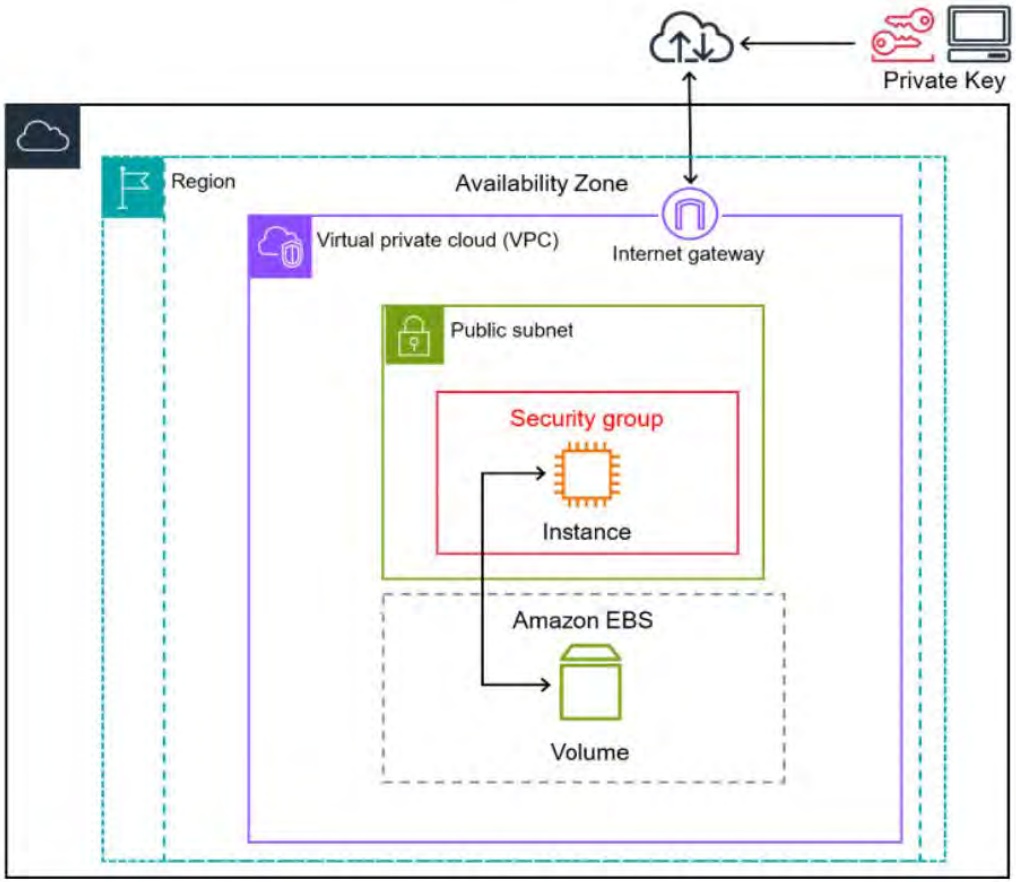
iamneo

Amazon EC2

Introduction to Amazon EC2

- Amazon Elastic Compute Cloud (Amazon EC2) is a web service that
- provides secure, resizable compute capacity in the cloud.
- Access reliable, scalable infrastructure on demand. Scale capacity within minutes with SLA commitment of 99.99% availability. Provide secure compute for your applications.

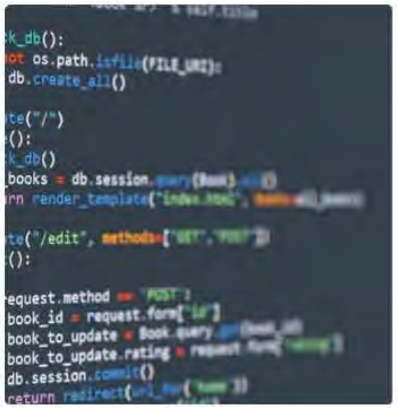
Introduction to Amazon EC2



Introduction to Amazon EC2

1. A computing powerhouse
2. Flexible and reliable
3. Affordable and cost-effective

What can you do with Amazon EC2?

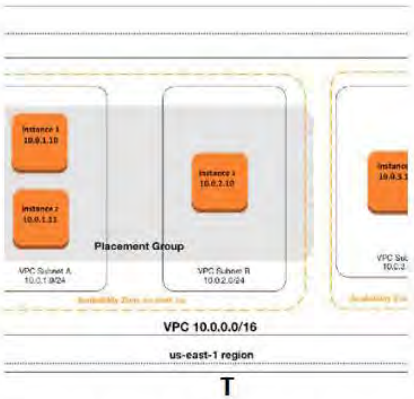


Application Development

Use EC2 as an environment to develop, test, and deploy applications, from simple web apps to complex enterprise solutions.

Big Data

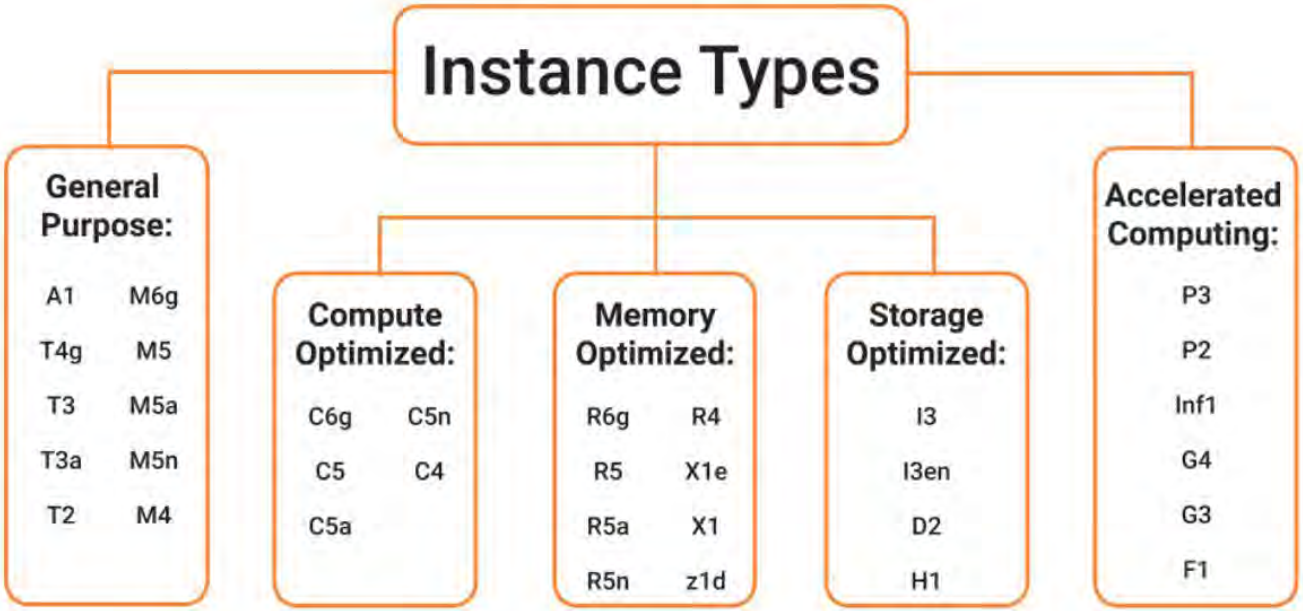
Run big data applications and workloads, including Hadoop and Spark, on EC2's powerful clusters of instances.



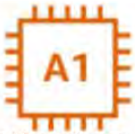
Team Collaboration

EC2 can be used for team collaboration and project management. Share instances and allow various group-level permissions to drive collaboration and productivity.

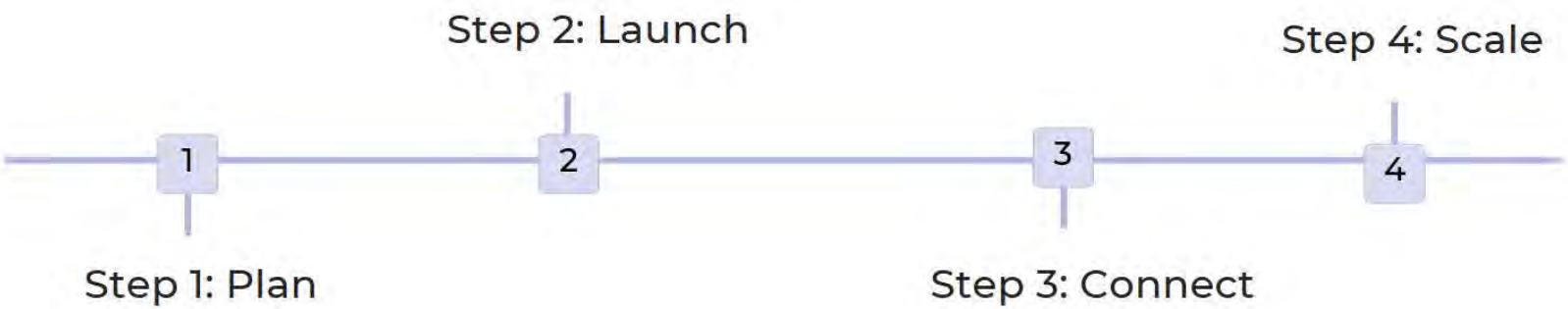
Instance Types



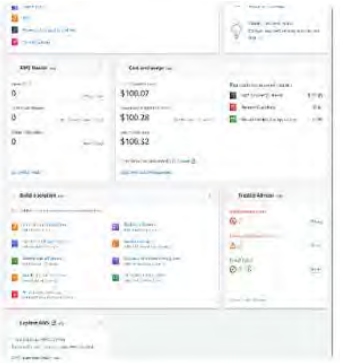
Instance Types

General Purpose	Compute Optimised	Memory Optimised	Accelerated Computing	Storage Optimised
 ARM based core and custom silicon	 Compute - CPU intensive apps and DBs	 RAM - Memory intensive apps and DB's	 Processing optimised- Machine Learning	 High Disk Throughput - Big data clusters
 Tiny - Web servers and small DBs		 Xtreme RAM - For SAP/Spark	 Graphics Intensive - Video and streaming	 IOPS - NoSQL DBs
 Main - App servers and general purpose		 High Compute and High Memory - Gaming	 Field Programmable - Hardware acceleration	 Dense Storage - Data Warehousing

Launching and Configuring EC2 Instances



Introduction to AWS Management Console



Sign Up or Log In

Create and log in to your AWS Management Console account. Follow the instructions to access the console.

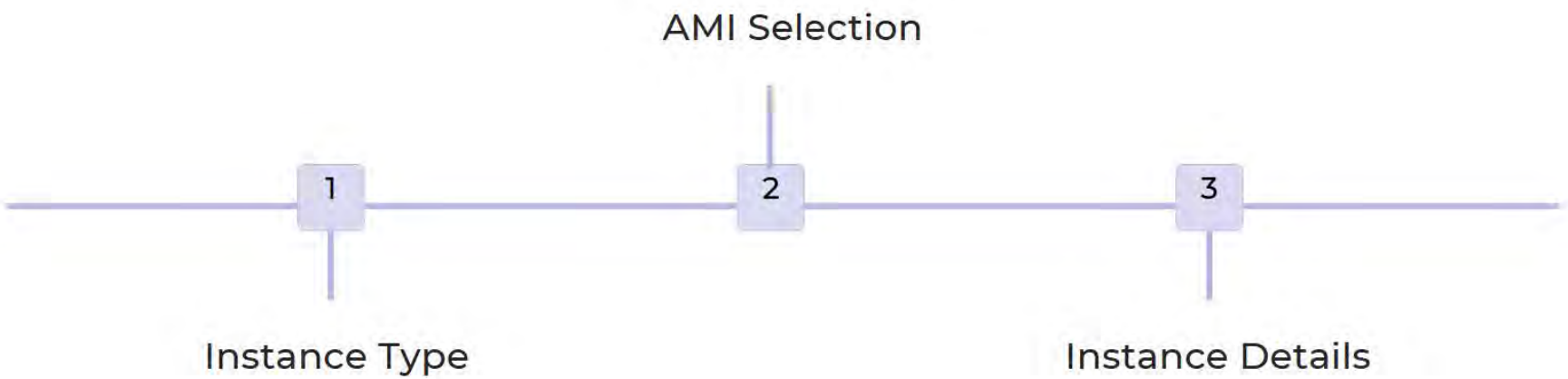
AWS Dashboard

The dashboard is the home page of your AWS Management Console. Explore the various services AWS offers.

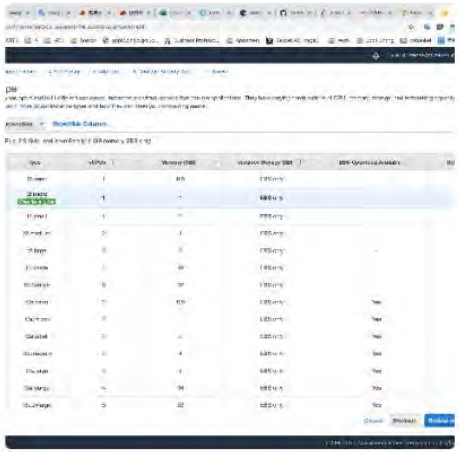
Configuring Settings

Quickly access, configure, and customize your settings and preferences according to your needs.

Launching an EC2 Instance



Configuring Instance Details



Instance Name

Name your instance and specify the purpose so you can easily identify it later on.

Instance Type

Specify the details for the instance type.

Configuring Instance Details

Port range	Protocol	Source
22	TCP	[redacted]/3
443	TCP	[redacted]/3
80	TCP	[redacted]/3
0 - 65535	TCP	0.0.0.0/0

Port range	Protocol	Destination
All	All	0.0.0.0/0

Security Groups

Choose the security groups you want your instance to be associated with.

Key Pairs

Choose the key pair to log in to your instance.

Choosing Storage Options

EBS Volume

Elastic Block Store (EBS) lets you store data separately from the instance. It can be easily attached or detached to the instance.

Instance Store Volume

Instance Store Volumes work similarly to EBS, but the data is tied to the instance's lifecycle.

Snapshot

Creating a snapshot ensures that you have a backup of your instance. Snapshots can be created on demand or scheduled.

Setting Up Security Groups

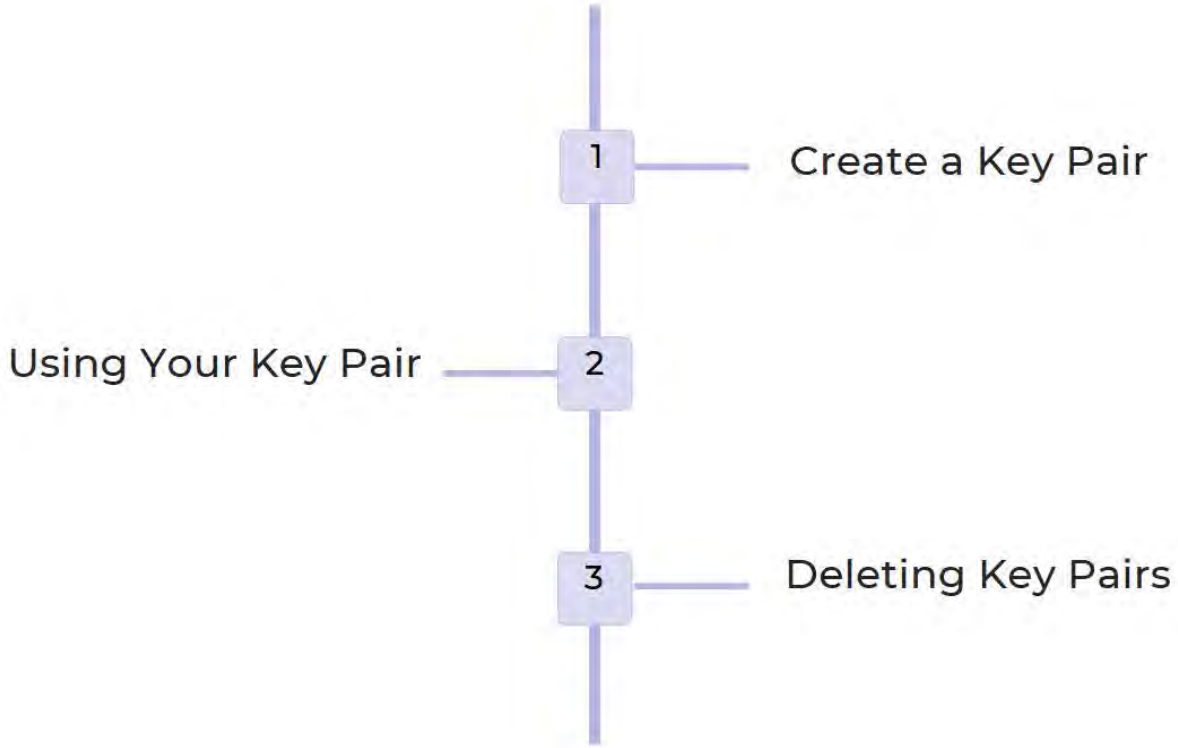
Inbound Rules

- Control access to the instance
- Specify ports and protocols
- Allow unrestricted access

Outbound Rules

- Control instance access to the internet
- Specify ports and protocols
- Allow unrestricted access

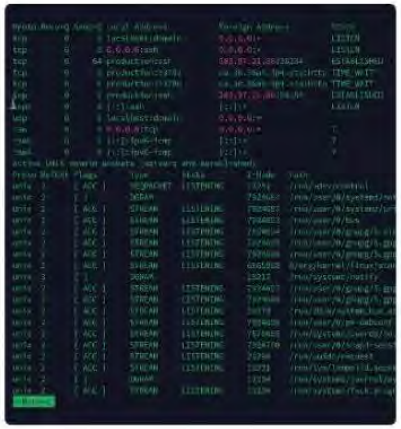
Creating and Using Key Pairs



Connecting to Instances using SSH or RDP

Windows -RDP

Mac -SSH



Linux -SSH

Understanding EC2 Instance States and Lifecycle

Running Instances

EC2 instances launched and running in a specific Availability Zone.

Stopped Instances

EC2 instances that have been stopped and can be re-launched when needed.

Terminated Instances

EC2 instances that have been terminated, and their data cannot be recovered.

Managing EC2 Instances using AWS Management Console

Launch an Instance

Reboot and Terminate

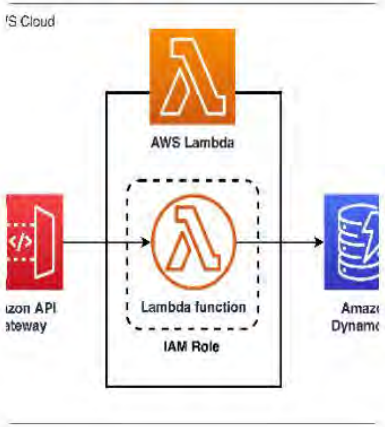
Scaling and Load Balancing

Monitoring and Troublehooting

Managing EC2 Instances using CLI

Command Line Interface

Automation using Shell Scripts



AWS Lambda Functions

Managing EC2 Instances using SDKs

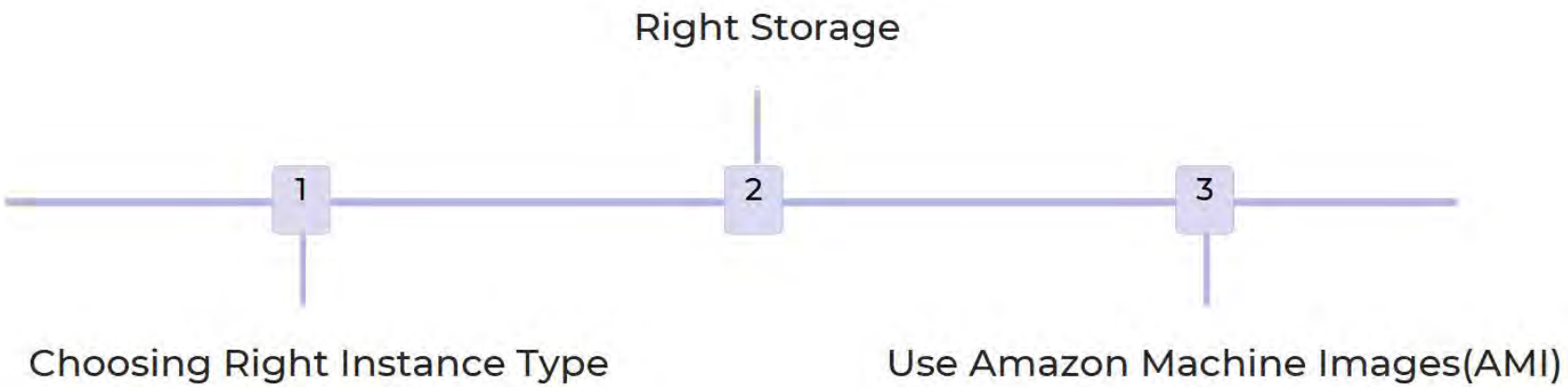
Amazon SDKs

Use AWS SDKs to manage EC2 instances programmatically from your preferred programming languages such as Node.js, Java, Python, etc.

AWS CloudFormation

Create your instances along with all required dependencies, security, network, and storage using AWS CloudformationStacks!

Best Practices for Optimizing EC2 Instances for Cost



Best Practices for Optimizing EC2 Instances for Cost

Use AWS Compute Optimizer

Utilize available AWS Compute Optimizer tools to efficiently choose the most cost-effective instance type for your workload

Reserved Instances

Save you up to 75% on EC2 instances and provide capacity reservation when you need it most.

Spot Instances

Utilize the unused capacity of Amazon EC2 instances at highly reduced prices and perform cost-efficient batch processing or run other workloads with flexible start and end times.

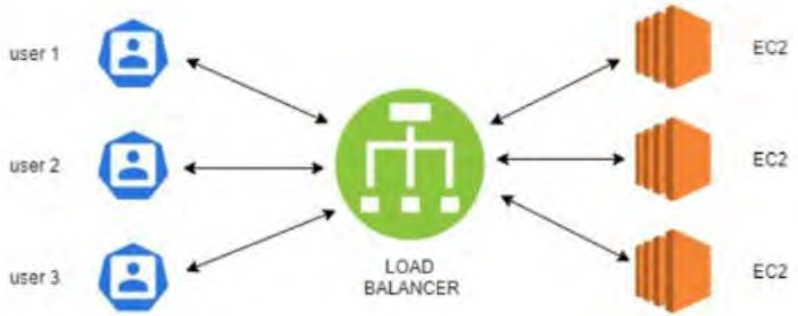
iamneo

AWS Load Balancer

AWS Load Balancer

Amazon Web Services (AWS) Load Balancer distributes traffic to multiple targets such as EC2 instances or containers, which increases the availability of your applications. In this presentation, we will explore different types of Load Balancers and how to create and configure them.

Load Balancing



- Single point of access (DNS) to your application. Provide
- SSL termination (HTTPS) for your websites.
- Separate public traffic from private traffic.
- It is integrated with many AWS offerings / services.
- You can setup internal (private) or external (public) ELBs.
- AWS guarantees that it will be working AWS takes care of upgrades, maintenance, high availability.

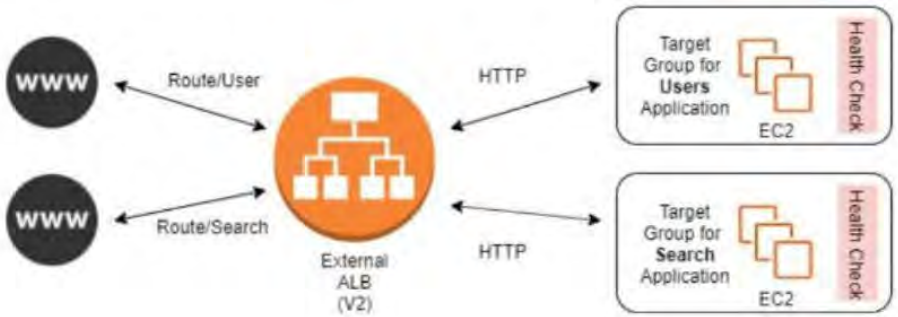
Types of AWS Load Balancers

- Application Load Balancer
- Network Load Balancer
- Classic LoadBalancer
- Gateway Load Balancer

Types of AWS Load Balancers

- Application Load Balancer
- Network Load Balancer
- Classic LoadBalancer
- Gateway Load Balancer

Application Load Balancer (V2)



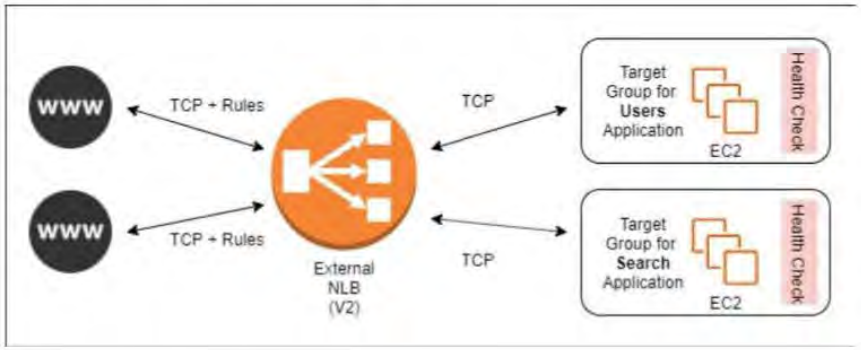
Application load balancers (Layer 7) allow you to do:

- ➢ Load balancing for multiple HTTP applications across machines (target groups).
- ➢ Load balancing for multiple applications running on the same machine(ex: containers).
- ➢ Load balancing based on route (path) in URL.
- ➢ Load balancing based on Hostname in URL.

Application Load Balancer (V2)

- ALB is perfect for micro services & container-based applications.
- Port mapping feature to redirect to a dynamic port.
- Previously we used to create one Classic Load Balancer per application.
- That was very expensive and inefficient!
- Stickiness can be enabled at the target group level.
- ALB support HTTP/HTTPS & Web sockets protocols.
- The application servers don't see the IP of the client directly. Instead, it is inserted in the header X-Forwarded-For.

Network Load Balancer (V2)

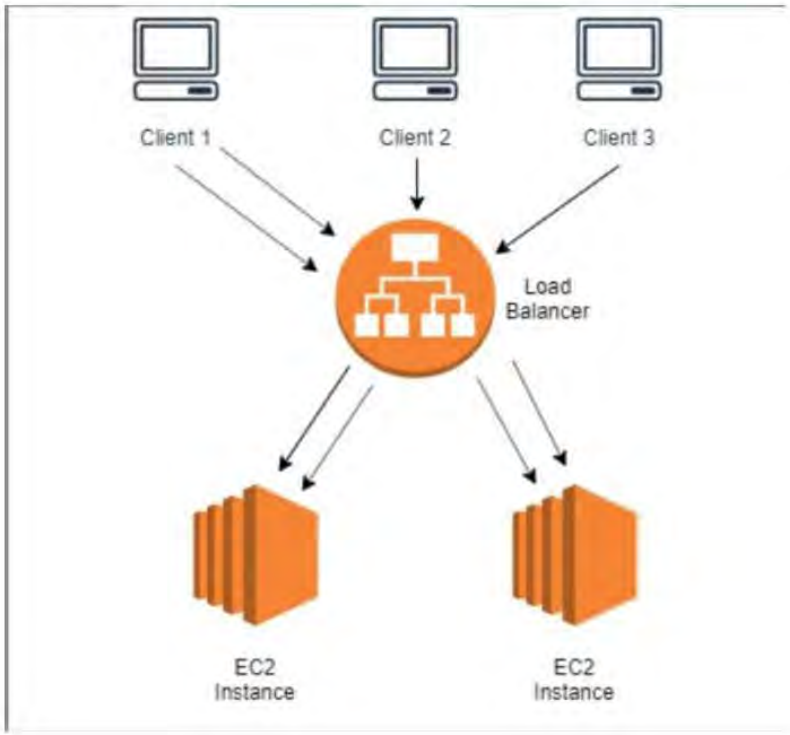


- Network load balancers (Layer 4) allow you to :
  - Forward TCP traffic to your instances
  - Handle millions of request per seconds
  - Support for static IP or elastic IP
  - Support Cross Zone Balancing.
- Network Load Balancers are mostly used for extreme performance and should not be the default load balancer.

Important Notes:

- CLB, ALB & NLB support SSL certificates and provide SSL termination.
- All Load Balancers have health check capability.
- ALB is a great fit with ECS (Docker).
- All Load Balancers in AWS has a static host name.
- Do not change and use underlying IP.
- Network Load Balancer can directly see the client IP.
- 4xx errors are client induced errors.
- 5xx errors are application induced errors.
- 503 means at capacity or no registered target
- If the Load Balancer can't connect to your application, check your security groups.

Load Balancer Stickiness



- Stickiness is nothing but the same client is always redirected to the same instance behind a load balancer.
- This works for Classic Load Balancers & Application Load Balancers.
- "Cookies" are used for stickiness and those cookies has an Expiry date which you can control.
- Enabling stickiness may bring imbalance to the load over the backend EC2 instances as it will stick to one instance and send traffic only to that instance for certain amount of time

Steps for Creating and Configuring Load Balancers using AWS

Follow these steps to create and configure a load balancer in AWS:

1. Choose the appropriate type of load balancer for your use case, such as Application Load Balancer or Network Load Balancer.
2. Create a target group for your load balancer that specifies the resources you want to distribute traffic to, such as EC2 instances or containers.
3. Configure your load balancer listeners to specify the protocols and ports to use for incoming traffic.
4. Configure your load balancer routing to specify the rules for distributing traffic among your resources.
5. Set up health checks to ensure that your load balancer is sending traffic only to healthy resources.
6. Configure Auto Scaling to automatically adjust the number of resources in your target group based on traffic demand.
7. Manage your load balancer security groups to ensure that only authorized traffic is allowed.

Configuring Listeners, Target Groups, and Routing Rules

Listeners, target groups, and routing rules are the building blocks of an AWS load balancer.

Listeners

A listener is a process that checks for connection requests. Add a listener in the AWS Management Console by selecting your load balancer, going to the Listeners tab, and configuring the protocol and port.

Target Groups

A target group is a group of resources that the load balancer distributes traffic to. Create a target group in the AWS Management Console by navigating to Target Groups, clicking Create target group, and specifying the target type, protocol, port, and VPC.

Routing Rules

Routing rules determine how the load balancer distributes traffic among your target groups. Configure routing rules by selecting your listener, adding rules under the Rules tab, and setting conditions and actions to route traffic to specific target groups.

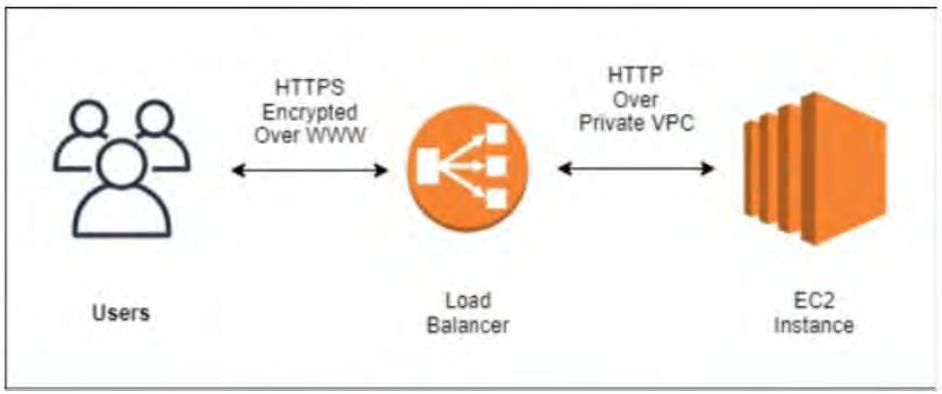
Enabling Health Checks for Automatic Instance Failover

Health checks are a crucial component of an AWS load balancer, as they allow the load balancer to detect when a target is unhealthy and automatically failover to a healthy target. Here's how to enable health checks for your load balancer:

1. Create a target group for your load balancer that specifies the resources you want to distribute traffic to, such as EC2 instances or containers.
2. Configure health checks for the target group to ensure that the load balancer sends traffic only to healthy resources. Health checks can be configured to check the status of the instance, the status of the application running on the instance, or both.
3. Configure your load balancer listeners to specify the target group that the listener should forward traffic to.
4. Configure your load balancer routing rules to specify the actions that the load balancer should take for requests that match the conditions.

Load Balancers SSL Certificates

- The load balancer uses an X.509 certificate (SSL/TLS server certificate)
- You can create upload your own certificates alternatively
- HTTPS listener:
  - You must specify a default certificate
  - You can add an optional list of certs to support multiple domains
  - Clients can use SNI (Server Name Indication) to specify the hostname they reach



Configuring Cross-Zone Load Balancing and Connection Draining

Cross-zone load balancing and connection draining are advanced features of an AWS load balancer that can improve the performance and reliability of your infrastructure. Here's how to configure them:

Cross-Zone Load Balancing

To enable cross-zone load balancing for your load balancer:

1. Open the Amazon EC2 console and navigate to the load balancer that you want to configure.
2. Select the "Attributes" tab and click "Edit".
3. Set "Cross-Zone Load Balancing" to "Enabled".
4. Click "Save" to save your changes.

Configuring Cross-Zone Load Balancing and Connection Draining

Connection Draining

Connection draining is a mechanism that allows the load balancer to complete in-flight requests before terminating a target that has become unhealthy. To enable connection draining for your load balancer:

1. Open the Amazon EC2 console and navigate to the load balancer that you want to configure.
2. Select the "Attributes" tab and click "Edit".
3. Set "Connection Draining" to "Enabled".
4. Specify the amount of time, in seconds, that the load balancer should wait before terminating an unhealthy target.
5. Click "Save" to save your changes.

Integrating with Auto Scaling for Dynamic Scaling of Instances Behind the Load Balancer

Auto Scaling is a powerful tool that allows you to automatically scale the number of instances in your infrastructure up or down based on demand. By integrating your load balancer with Auto Scaling, you can ensure that your infrastructure is always right-sized to handle your workloads. Here's how to configure Auto Scaling with your load balancer:

Step 1: Create an Auto Scaling Group

1. Open the Amazon EC2 console and navigate to the Auto Scaling groups page.
2. Click "Create Auto Scaling group" and follow the on-screen instructions to create your Auto Scaling group.
3. Specify the desired capacity, minimum capacity, and maximum capacity for your group, as well as any other configuration options that you need.
4. Configure your scaling policies to define how your group should scale up and down based on demand.

Step 2: Register Your Instances with the Load Balancer

To register your instances with your load balancer, you can use either the Amazon EC2 console or the command line interface. Here's how to do it using the console:

1. Open the Amazon EC2 console and navigate to the Instances page.
2. Select the instances that you want to register with your load balancer.
3. Click "Actions", then "Add to Load Balancer".
4. Select the load balancer that you want to register your instances with, then click "Add".

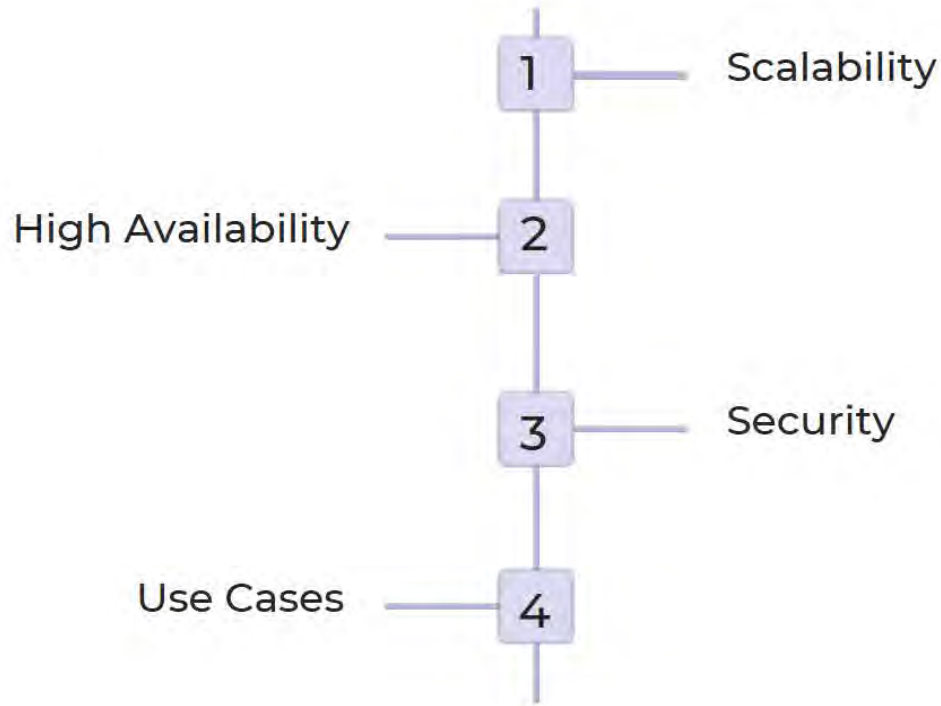
Step 3: Configure Your Load Balancer to Use Your Auto Scaling Group

To configure your load balancer to use your Auto Scaling group, you need to update your target group. Here's how to do it:

1. Open the Amazon EC2 console and navigate to the target groups page.
2. Select the target group that you want to update.
3. Click "Edit", then select your Auto Scaling group from the "Registered" tab.
4. Click "Save" to save your changes.

By integrating your load balancer with Auto Scaling, you can ensure that your infrastructure is always optimized for your workloads. Your Auto Scaling group will automatically adjust the number of instances in your infrastructure based on demand, while your load balancer will distribute traffic evenly across all healthy instances, ensuring that your users receive a high-quality experience.

Benefits and Use Cases of AWS Load Balancers



Pricing of AWS Load Balancers

	Application Load Balancer	Network Load Balancer	Classic Load Balancer
Price per hour Price per	$0.0225	$0.024	$0.025
LCU Load Balancer	$0.008	$0.008	$0.025
Capacity Units (LCU)	2,048 LCUs per hour	1 LCU per hour	1 LCU per hour

Load Balancer pricing is based on the number of hours and Load Balancer Capacity Units (LCUs) used per hour. Application Load Balancers are cheaper than Network and Classic Load Balancers.

Troubleshooting Common Issues with AWS Load Balancers

1 High Latency

Check the network and application performance, validate the health of your targets, and explore options for scaling your instances.

2 HTTP 503 Errors

Ensure that the target instances are configured correctly, and check that they're receiving traffic from the Load Balancer. Switch to a healthy target group if needed.

3 SSL Certificate Issues

Ensure that the certificate is valid and review the security group rules to enable incoming traffic on appropriate ports.

Best Practices for Implementing AWS Load Balancers

Security

Automation

Logging and Monitoring

iamneo

AWS AutoScaling

What is autoscaling and why is it important?

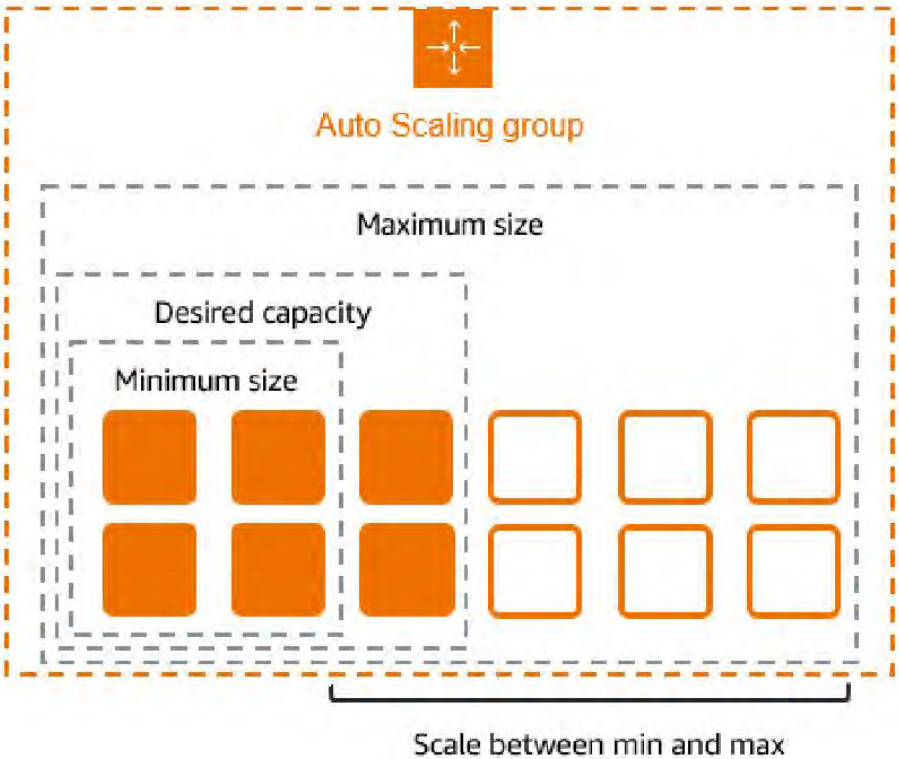
- Autoscaling is a feature of cloud computing that allows you to automatically adjust the number of computing resources in your system based on demand. This means that if your application suddenly becomes very popular and starts receiving a lot of traffic, autoscaling will automatically add more resources to handle the increased load.

- Autoscaling is important because it helps you ensure that your application is always available and responsive, even during times of high traffic.

- It also helps you save costs by only using the number of resources that you need, rather than paying for resources that are sitting idle.

- With autoscaling, you can achieve better performance, higher availability, and lower costs.

What is autoscaling and why is it important?



What is autoscaling and why is it important?

Scaling based on demand

Multiple instance types supported

Integration with CloudWatch

Why Autoscaling Matters?

Scalability and Flexibility

Cost Efficiency

Performance and Reliability

How does AWS Auto Scaling work?

1

Step 1

Set triggers to monitor your application's resource usage.

2

Step 2

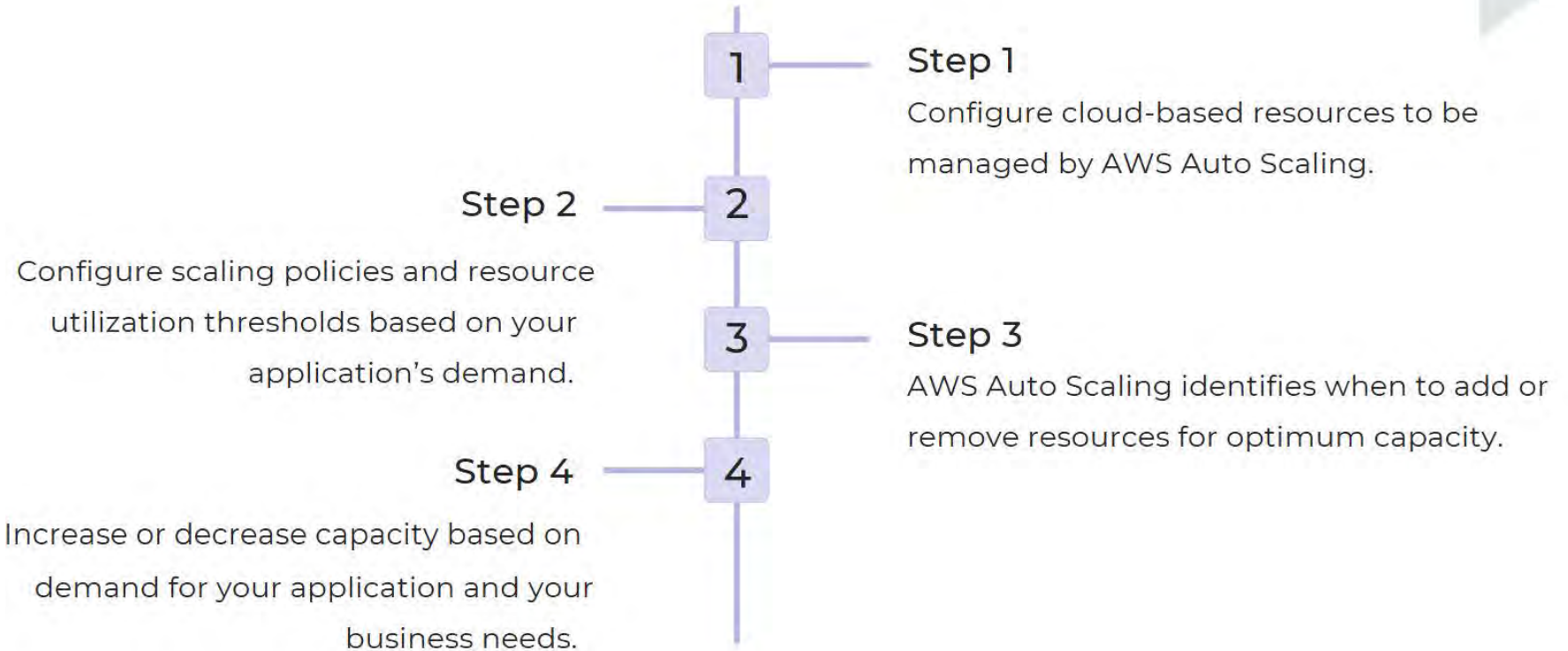
AWS will automatically adjust your resources as traffic changes.

3

Step 3

Automatic scaling ensures consistent performance and optimizes costs.

Implementation of AWS Auto Scaling



Understanding the components of Auto Scaling

Auto Scaling is an AWS service that allows you to automatically adjust the capacity of your EC2 instances based on demand. There are three main components to Auto Scaling:

- **Auto Scaling groups:** A collection of EC2 instances that are created and managed together. You can specify the minimum and maximum number of instances in the group, and Auto Scaling will automatically adjust the number of instances based on demand.
- **Launch configurations:** A template that defines the settings for new instances that are launched by the Auto Scaling group. This includes the AMI, instance type, and security groups.
- **Scaling policies:** Rules that determine when and how to adjust the capacity of the Auto Scaling group. For example, you might create a scaling policy that adds 2 instances when CPU usage exceeds 80% for 5 minutes.

By using these components together, you can ensure that your application always has the right amount of capacity to handle traffic. You can also save money by only paying for the instances that you need.

AutoScaling Groups

Defining Group Size Limits

Creating an AutoScaling Group

Configuring AutoScaling Triggers

Creating Multi-Zone Deployments

Understanding Autoscaling Groups

- An autoscaling group is a collection of Amazon EC2 instances that are designed to work together to handle incoming traffic.
- When you create an autoscaling group, you specify the minimum and maximum number of instances that should be running at any given time.
- If the traffic to your application increases, the autoscaling group will automatically add more instances to handle the load. If the traffic decreases, the autoscaling group will remove some of the instances to save costs. This allows your application to handle variable levels of traffic without any manual intervention.

Creating Auto Scaling groups and defining launch configurations

- Auto Scaling groups are a core component of AWS Auto Scaling. An Auto Scaling group contains a collection of Amazon EC2 instances that are created from a common Amazon Machine Image (AMI).

- The group automatically scales the number of instances up or down in response to changes in demand for the application. To create an Auto Scaling group, you'll need to:

  1.Create an Amazon Machine Image (AMI) that contains the software and configuration for your application.

  2.Create a launch configuration that describes the settings for the instances that will be launched by the Auto Scaling group. This includes the AMI ID, instance type, and security groups.

  3.Create an Auto Scaling group and specify the minimum and maximum number of instances that the group should maintain. You'll also need to specify the launch configuration that you created in step 2.

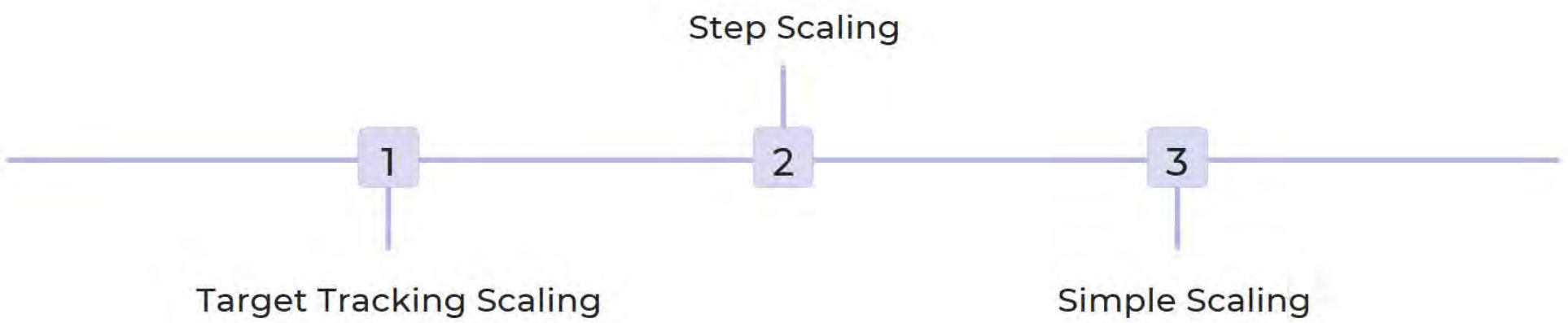
Configuring Autoscaling Group Capacity

- When creating an autoscaling group, you need to specify the minimum, maximum, and desired capacity. The minimum capacity is the smallest number of instances that can be running at any time. The maximum capacity is the largest number of instances that can be running at any time. The desired capacity is the number of instances that should be running at any given time.

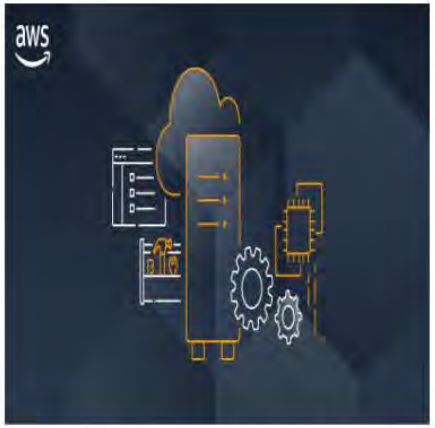
·When the autoscaling group launches, it will start with the desired capacity. If the traffic to your application increases and exceeds the desired capacity, the autoscaling group will automatically add more instances up to the maximum capacity. If the traffic decreases and goes below the desired capacity, the autoscaling group will remove some of the instances down to the minimum capacity.

·It's important to choose appropriate values for these parameters based on the expected traffic to your application. If the minimum capacity is too high, you'll be paying for instances that you don't need. If the maximum capacity is too low, your application won't be able to handle spikes in traffic.

AutoScaling Policies



Customizing AutoScaling with Lifecycle Hooks



Extending AWS Services

Terminate Protection

EBS Volume Creation

Key Features and Benefits

Cost Optimization💰

Quick and Accurate Scaling🚀

Improved Reliability🛡️

Use Cases

E-commerce applications

Mobile apps

Cloud infrastructure

Challenges and Limitations

1. Application design
2. Scaling limitations
3. Costs

Best Practices for AWS Auto Scaling

Algorithm selection

Choose the algorithm that best meets the needs of your application.

Application architecture

Consider implementing per-application scaling for optimal resource allocation.

Monitoring and testing

Continuously monitor and test to ensure AWS Auto Scaling is working optimally with your applications.

Scheduling

Use AWS Auto Scaling scheduled actions to proactively manage capacity.

AutoScaling Case Studies: Examples of Successful Implementation

Netflix

Netflix relies heavily on AutoScaling to stream content for millions of users worldwide. By dynamically allocating streaming resources as needed, Netflix ensures that users can watch their favorite shows and movies whenever they want.

Nasa JPL

NASA's Jet Propulsion Laboratory (JPL) uses AutoScaling to process data and simulate workloads for various space missions. By scaling up or down based on demand, JPL ensures that its compute resources are optimized and cost-effective.

Kajabi

Kajabi, an e-learning platform, uses predictive scaling to optimize compute resources during their peak hours. By predicting workload patterns and scaling in advance, they ensure students have smooth interaction with courses delivered on the platform.

iamneo

Amazon Lambda

Serverless Computing Made Easy with AWS Lambda

What is Serverless Computing?

Cloud-based

Serverless computing enables developers to focus on writing code for specific tasks, rather than managing servers or infrastructure.

Cost-effective

Because serverless computing only requires payment for actual usage, it's often more cost-effective than traditional hosting or infrastructure.

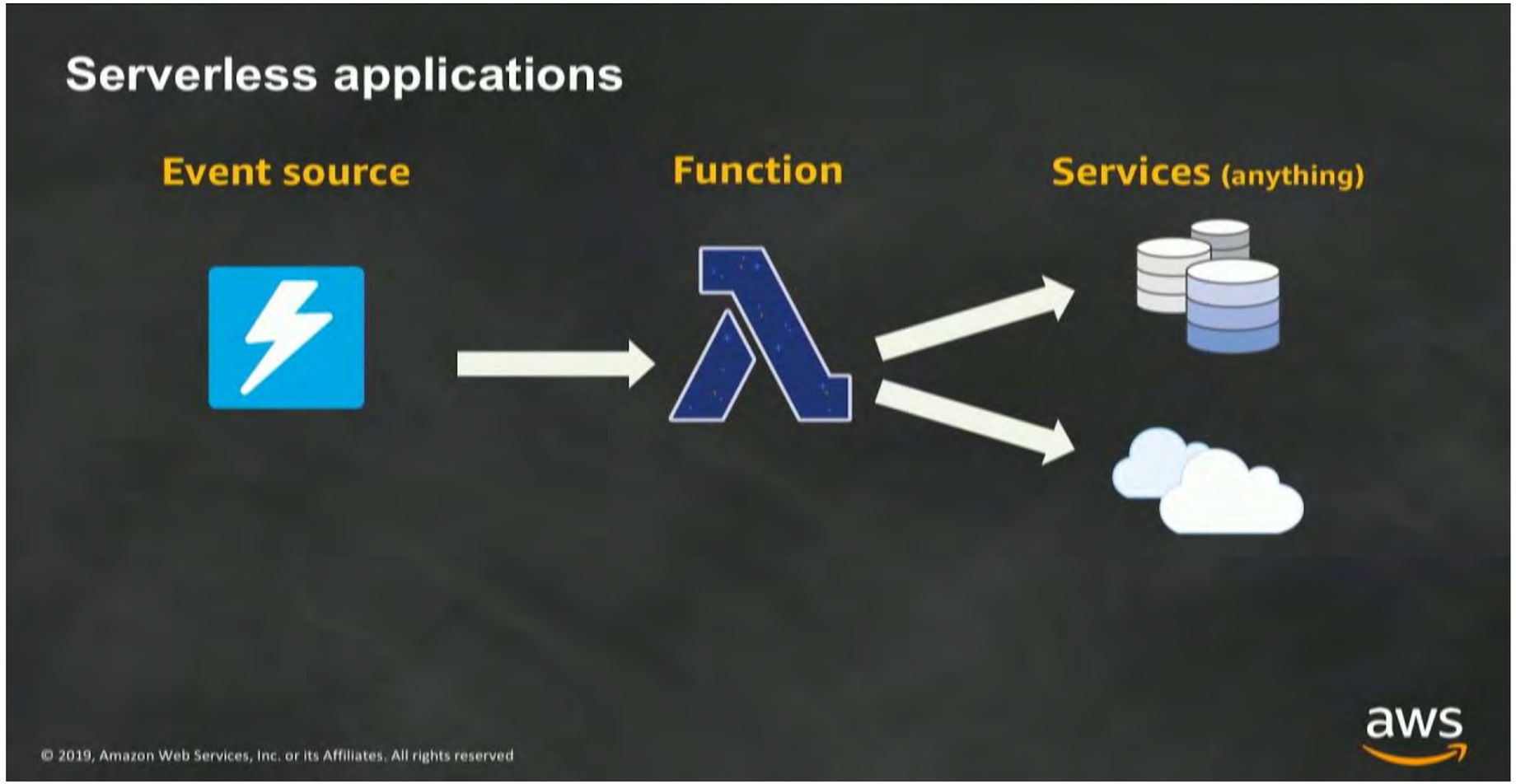
Event-driven

With serverless computing, code is executed only in response to events, such as data changes or user actions, making it highly efficient and scalable.

Why AWS Lambda?

- **Flexible**
- **Scalable**
- **Cost-effective**
- **Integrative**

How AWS Works



How AWS Works

Amazon Web Services

AWS is a cloud computing platform that provides a wide range of scalable services, including AWS Lambda, to businesses and individuals.

Code and Triggers

Developers create code in the appropriate language, set up triggers that launch the code, and rely on AWS to execute the code securely and efficiently.

Leveraging the Cloud

With AWS Lambda, developers can leverage the power of the cloud, without worrying about server maintenance and setup.

Use Cases for AWS Lambda

1

Build a Real-time File Processing System

Process incoming files, store them in S3, send notifications via Amazon SNS, and analyze data using Amazon Athena.

2

Create a Full-stack Web Application

Use AWS services such as API Gateway and DynamoDB to build a serverless web application that scales automatically with user demand.

3

Automate Your Data Pipeline

Use AWS Lambda to automate data collection, processing, and storage, so you can focus on using your data to drive insights and innovation.

Building Your First Lambda Function

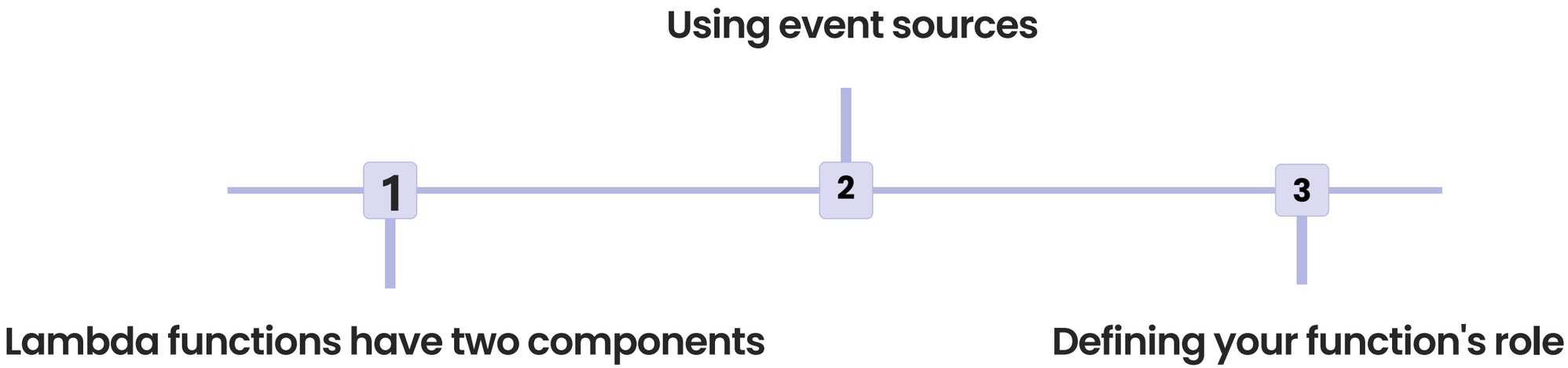
Step 1: Choose a programming language

Step 2: Define a handler function

Step 3: Configure triggers

Step 4: Testing and debugging

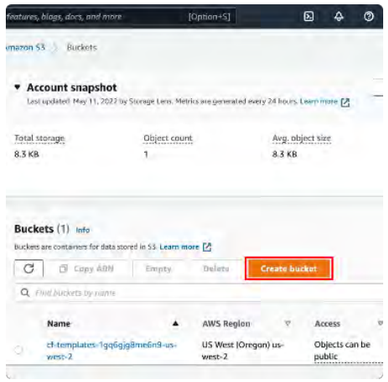
Structuring Your Lambda Functions



Triggering and Invoking Lambda Functions

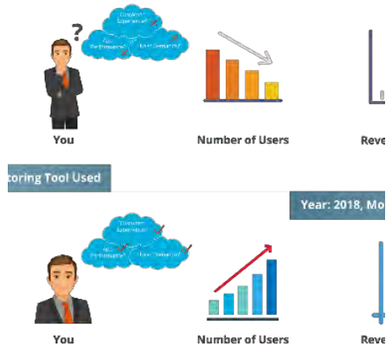
Manual Invocation	You can invoke a Lambda function manually using the AWS console.
API Gateway Trigger	Create an API Gateway and configure it to trigger your Lambda function based on HTTP requests.
Event Trigger	Your function can be triggered based on events in other AWS resources, such as S3 object creation, Kinesis Data Streams, and DynamoDB.

Integrating AWS Lambda Functions with Other Services



Integrating with Amazon S3

Integrating with Amazon Alexa



Managing and Monitoring Your Lambda Functions

Managing and Monitoring Lambda Functions