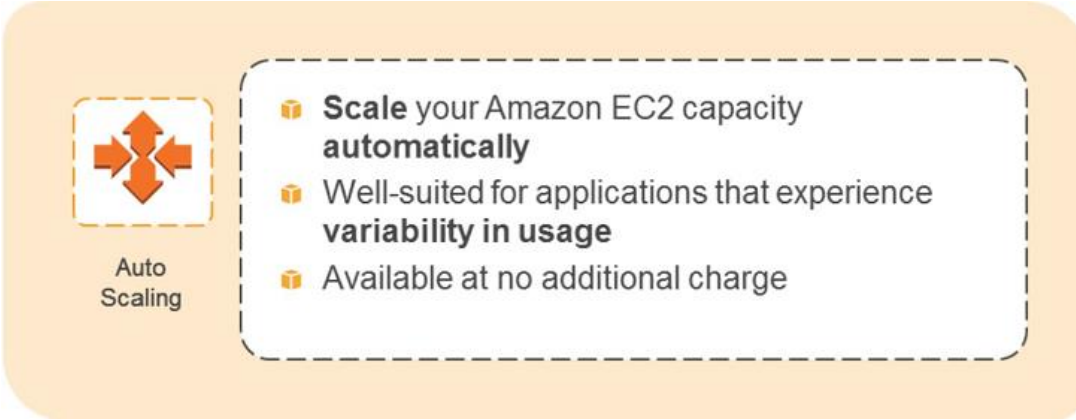
AWS ELASTICITY

AWS Elasticity

- Auto Scaling
- Elastic Beanstalk
- Elastic Load balance

Auto Scaling

- Auto Scaling is particularly well suited for applications that experience hourly, daily, or weekly variability in usage. Auto Scaling is enabled by Amazon CloudWatch and available at no additional charge beyond the Amazon CloudWatch fees.

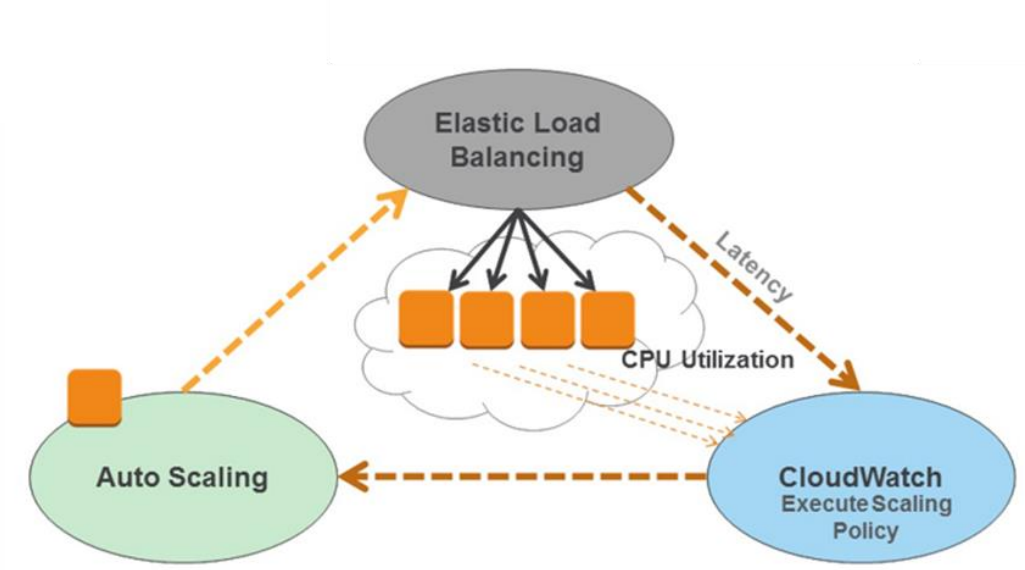


Benefits

- Elastic capacity:
  - With Auto Scaling, you can ensure that the number of Amazon EC2 instances you are use increases seamlessly during demand spikes to maintain performance, and decreases automatically during demand lulls to minimize costs.
- Cost savings:
  - Save compute costs by terminating underused instances automatically and launching new instances when you need them, without the need for manual intervention.
- Ease of use:
  - Manage your instances as a single collective entity and define rules for when instances should be added and removed. Replace lost or unhealthy instances automatically based on predefined thresholds.
- Multi-AZ:
  - Distribute, scale, and balance applications automatically over multiple Availability Zones within a region to support scalability and geographic redundancy.

Tres Mosqueteiros

- 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.



Beanstalk

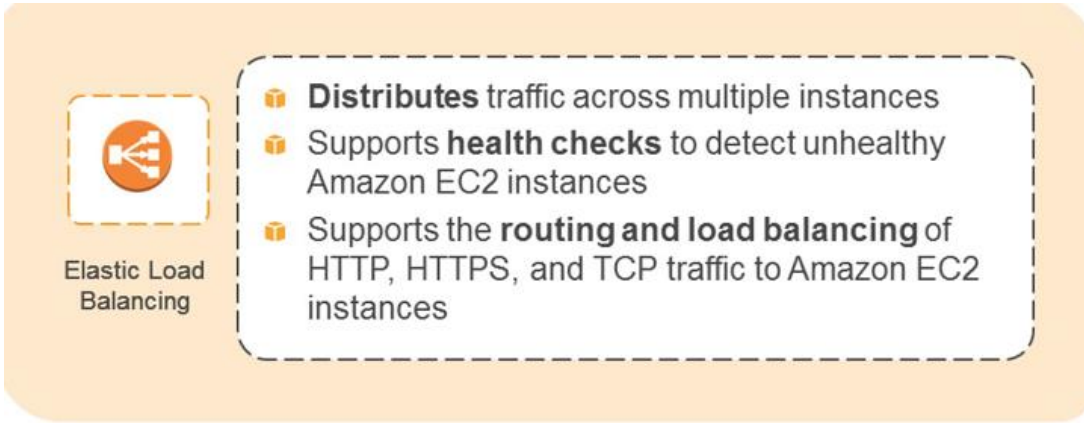
- With AWS Elastic Beanstalk, you simply upload your application, and AWS Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring.
- You can retain full control over the AWS resources powering your application.AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, Python, PHP, Ruby, and Docker on familiar services such as Apache, Nginx, Passenger, and IIS.

Demo

Tutorial

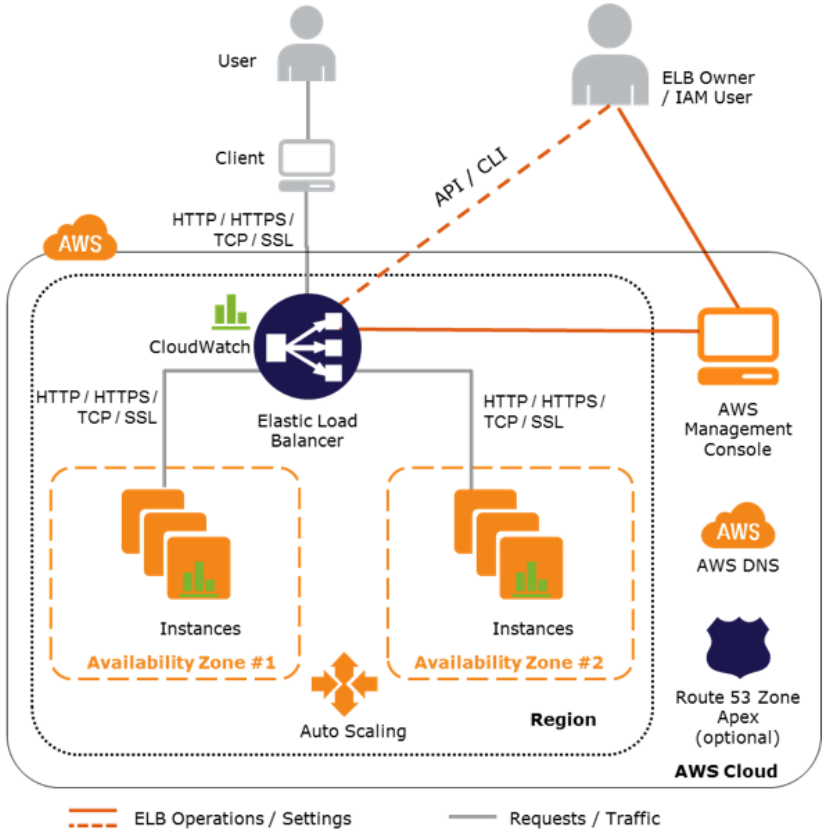
Elastic Load Balance

- Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances.
- It enables fault tolerance by providing the amount of load balancing capacity needed in response to incoming application traffic.
- Elastic Load Balancing detects unhealthy instances within a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored.
- Elastic Load Balancing can bewithin a single Availability Zone or across multiple zones for even more consistent application performance.



ELB Example

- Elastic Load Balancing automatically scales its request handling capacity in response to incoming traffic.
- The diagram shows how the various components of Elastic Load Balancing work together.
- You can access and work with your load balancer using one of the following interfaces:
  - AWS Management Console—A simple web-browser interface that you can use to create and manage your load balancers without using additional software or tools.
  - Command Line Interfaces —A Java-based command-line client that wraps the SOAP API



Features of Elastic Load Balancing

Feature	Classic Load Balancer	Application Load Balancer
Protocols	HTTP, HTTPS, TCP, SSL	HTTP, HTTPS
Platforms	EC2-Classic, EC2-VPC	EC2-VPC
Sticky sessions (cookies)	✔	load balancer generated
Idle connection timeout	✔	✔
Connection draining	✔	✔
Cross-zone load balancing †	✔	Always enabled
Health checks † †	✔	Improved
CloudWatch metrics	✔	Improved
Access logs	✔	Improved
Host-based routing		✔
Path-based routing		✔
Route to multiple ports on a single instance		✔
HTTP/2 support		✔
Websockets support		✔
Load balancer deletion protection		✔

Demo


amazon
web services
Training and
Certification
Introduction to
Elastic Load Balancing

ELB: H/W

- Create a VPC, e.g., MyVPC, 10.200.0.0/16
- Create two subnets, sub1a, sub1b, associated with us-east-1a and us-east-1b AZ's, respectively. A subnet should have 10.200.10.0/24 and another one 10.200.20.0/24. Make sure you enable DNS and public IP
- Create an Internet Gateway, MyIntGat, and associate it to MyVpc
- Create a security group where ports 22 and 80 are open to the internet
- Create a route table with 2 rules
  - 10.200.0.0/16 target local and
  - 0.0.0.0/0 target MyIntGat
- Launch 2 Ubuntu EC2 inside sub1a and sub1b respectively.
- Connect to each EC2 and install apache 2
  - sudo apt-get update
  - sudo apt-get install apache2
- Create an ELB and make sure these 2 EC2s are part of the ELB target group
- Access the ELB via the web
- Stop an EC2 and access again the ELB via the web
- Stop the 2nd DC2 and access again the ELB via the web
- Describe results