Load balancing

What Is Elastic Load Balancing?

Elastic Load Balancing distributes incoming application traffic across multiple EC2 instances, in multiple Availability Zones. This increases the fault tolerance of your applications.

What Is Elastic Load Balancing?

You can configure health checks, which are used to monitor the health of the registered instances so that the load balancer can send requests only to the healthy instances. You can also offload the work of encryption and decryption to your load balancer so that your instances can focus on their main work.

Features of Elastic Load Balancing

Elastic Load Balancing supports three types of load balancers: Application Load Balancers, Network Load Balancers, and Classic Load Balancers. You can select a load balancer based on your application needs. For more information, see Comparison of Elastic Load Balancing Products.

For more information about using each load balancer, see the User Guide for Application Load Balancers, the User Guide for Network Load Balancers, and the User Guide for Classic Load Balancers.

Accessing Elastic Load Balancing

You can create, access, and manage your load balancers using any of the following interfaces:

AWS Management Console
AWS Command Line Interface (AWS CLI)
AWS SDKs
Query API

Related Services

Elastic Load Balancing works with the following services to improve the availability and scalability of your applications.

Amazon EC2
Amazon ECS
Auto Scaling
Amazon CloudWatch
Route 53

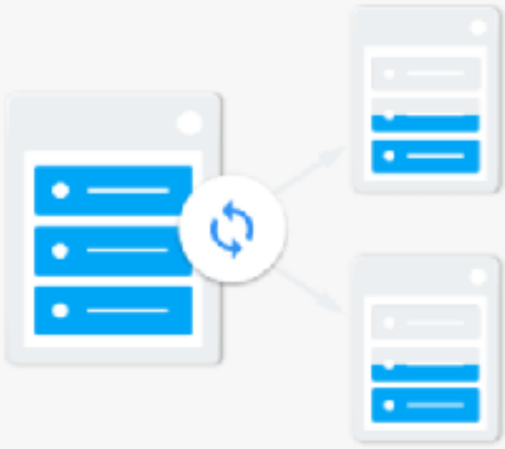
Google Cloud

Load balancing

- Load Balancing
  - Network-Based Load Balancing
  - Content-Based Load Balancing
  - Cross-region Load Balancing
  - HTTPS Load Balancing using NGINX
  - HTTP(S) Load Balancing using Microsoft IIS Backends
  - Internal Load Balancing using HAProxy
  - Autoscaled Internal Load Balancing using HAProxy and Consul

Worldwide Autoscaling and Load Balancing

Scale your applications on Google Compute Engine from zero to full-throttle with Google Cloud Load Balancing, **with no pre-warming needed**. Distribute your load-balanced compute resources in single or multiple regions, close to your users and to meet your high availability requirements. Cloud Load Balancing can put your resources behind a single anycast IP and **scale your resources up or down with intelligent Autoscaling**. Cloud Load Balancing comes in a variety of flavors and is integrated with Google Cloud CDN for optimal application and content delivery.

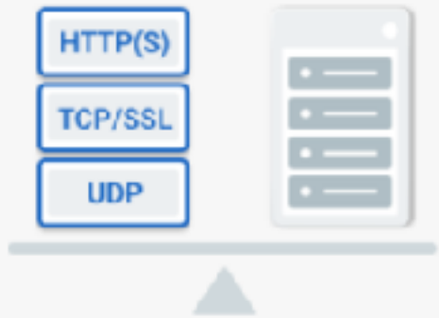


Global Load Balancing with Single Anycast IP

With Cloud Load Balancing, a single anycast IP front-ends all your backend instances in regions around the world. It provides cross-region load balancing including automatic multi-region failover which gently moves traffic in fractions if backends become unhealthy. In contrast to DNS-based Global Load Balancing solutions, Cloud Load Balancing reacts instantaneously to changes in users, traffic, network, backend health and other related conditions.

Software-Defined Load Balancing

Cloud Load Balancing is a fully distributed, software-defined, managed service for all your traffic. It is not an instance or device based solution, so you won't be locked into physical load balancing infrastructure or face the HA, scale and management challenges inherent in instance based LBs. You can apply Cloud Load Balancing to all of your traffic: HTTP(S), TCP/SSL, and UDP. You can also terminate your SSL traffic with HTTPS Load Balancing and SSL proxy.

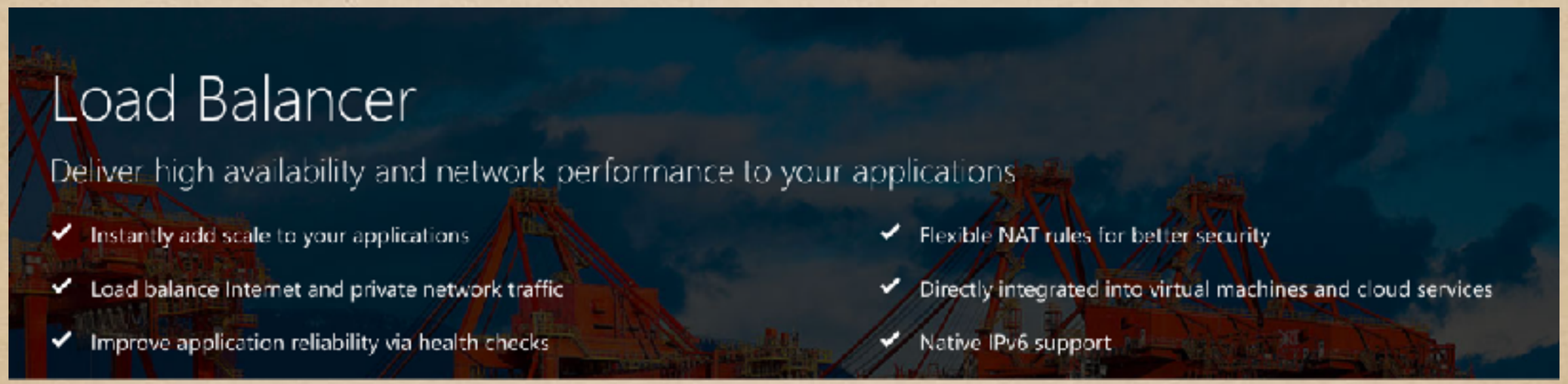


Internal Load Balancing

Internal Load Balancing enables you to build scalable and highly available internal services for your internal client instances without requiring your load balancers to be exposed to the Internet. GCP Internal Load Balancing is architected using Andromeda, Google's software-defined network virtualization platform.

Microsoft Azure

Load balancing



Simplify load balancing for applications

With built-in load balancing for cloud services and virtual machines, you can create highly-available and scalable applications in minutes. Azure Load Balancer supports TCP/UDP-based protocols such as HTTP, HTTPS, and SMTP, and protocols used for real-time voice and video messaging applications.

Related products and services

Virtual Network

Provision private networks, optionally connect to on-premises datacenters

Virtual Machines

Provision Windows and Linux virtual machines in seconds

Cloud Services

Create highly-available, infinitely-scalable cloud applications and APIs

Learn more

Documentation

Purchase options

Resources

Load Balancer status

Region availability

Support options

Videos

Service updates

General availability: Standard Load Balancer
3/22/2018

More updates

Auto scaling

What Is Amazon EC2 Auto Scaling?

Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups. You can specify the minimum number of instances in each Auto Scaling group, and Auto Scaling ensures that your group never goes below this size. You can specify the maximum number of instances in each Auto Scaling group, and Auto Scaling ensures that your group never goes above this size. If you specify the desired capacity, either when you create the group or at any time thereafter, Auto Scaling ensures that your group has this many instances. If you specify scaling policies, then Auto Scaling can launch or terminate instances as demand on your application increases or decreases.

What Is Application Auto Scaling?

Use Application Auto Scaling to scale the following compute and data resources for your cloud-based web applications:

Amazon ECS services

Spot Fleet requests

Amazon EMR clusters

AppStream 2.0 fleets

DynamoDB tables and global secondary indexes

Aurora replicas

Amazon SageMaker endpoint variants

Features of Application Auto Scaling

Target tracking scaling—Scale a resource based on a target value for a specific CloudWatch metric.

Step scaling— Scale a resource based on a set of scaling adjustments that vary based on the size of the alarm breach.

Scheduled scaling—Scale a resource based on the date and time.

Serverless computing

Serverless computing is a cloud computing execution model in which the cloud provider dynamically manages the allocation of machine resources. Pricing is based on the actual amount of resources consumed by an application, rather than on pre-purchased units of capacity.[1] It is a form of utility computing.

Google Cloud

Auto scaling

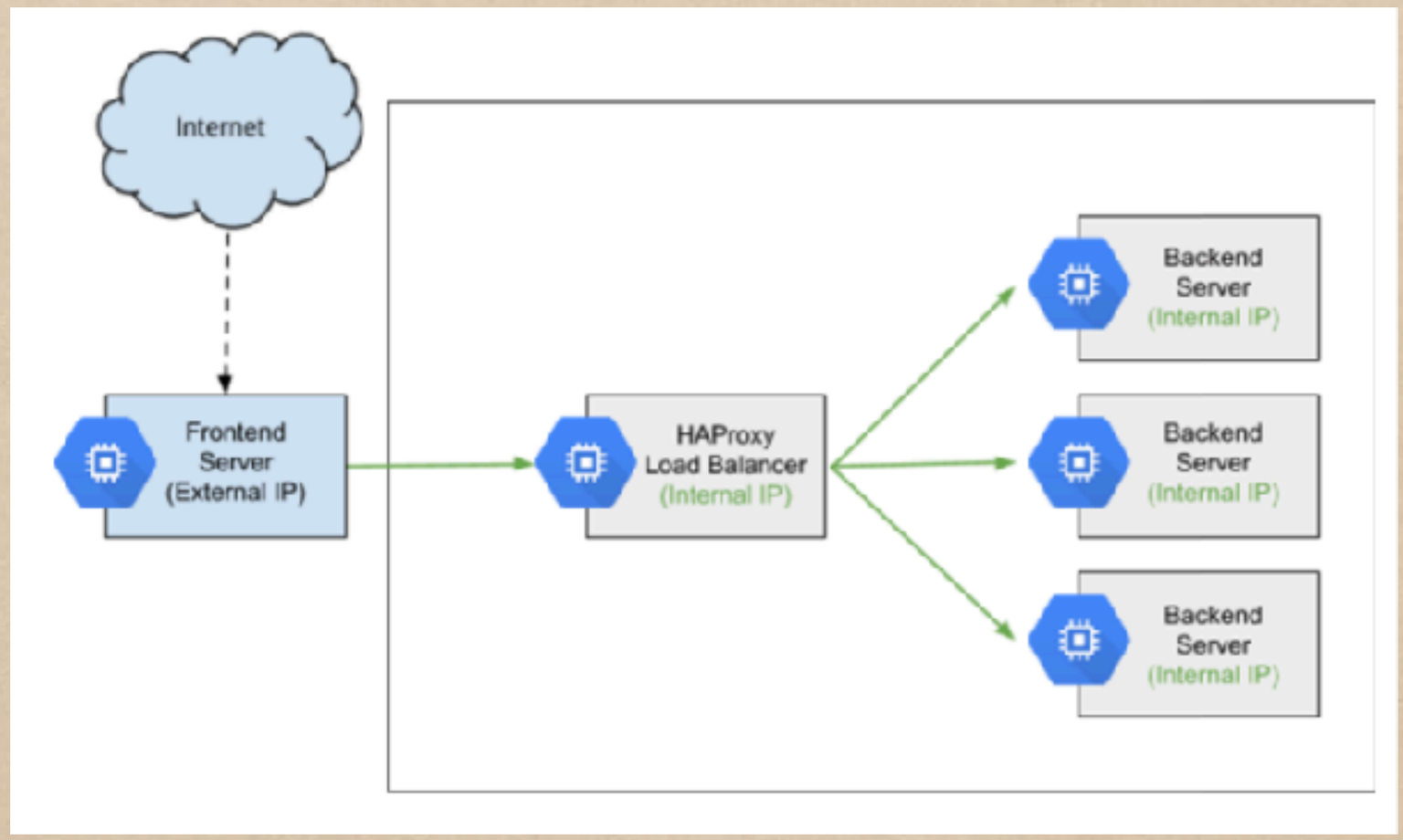
发送反馈

Autoscaled Internal Load Balancing using HAProxy and Consul on Compute Engine

This solution shows how to use Consul by HashiCorp , a DNS-based service-discovery product, as part of an HAProxy-based, internal load-balancing system on Google Cloud Platform.

> ★ **Note:** Compute Engine offers managed internal load balancing for your TCP/UDP-based traffic. Using the provided internal load balancer can save you time and effort over implementing your own solution. For information about this feature, see Setting Up Internal Load Balancing.

An internal load balancer distributes network traffic to servers on a private network. Neither the internal load balancer nor the servers it distributes traffic to are exposed to the Internet. The Internal Load Balancing using HAProxy on Google Compute Engine tutorial provides a good explanation of the basics of internal load balancing and walks you through configuring an internal load balancer using HAProxy and Google Compute Engine.


Internet
Frontend
Server
(External IP)
HAProxy
Load Balancer
(Internal IP)
Backend
Server
(Internal IP)
Backend
Server
(Internal IP)
Backend
Server
(Internal IP)

SEND FEEDBACK

Autoscaling Groups of Instances

Managed instance groups offer autoscaling capabilities that allow you to automatically add or remove instances from a managed instance group based on increases or decreases in load. Autoscaling helps your applications gracefully handle increases in traffic and reduces cost when the need for resources is lower. You just define the autoscaling policy and the autoscaler performs automatic scaling based on the measured load.

Autoscaling works by scaling up or down your instance group. That is, it adds more instances to your instance group when there is more load (upscaling), and removes instances when the need for instances is lowered (downscaling).

Specifications

- Autoscaling only works with managed instance groups. Unmanaged instance groups are not supported.
- Do not use Compute Engine autoscaling with managed instance groups that are owned by Google Kubernetes Engine. For Google Kubernetes Engine groups, use Cluster Autoscaling instead.

  If you are not sure if your group is part of a Google Kubernetes Engine cluster, look for the `gke` prefix in the managed instance group name. For example, `gke-test-1-3-default-pool-eadji9ah`.
- An autoscaler can make scaling decisions based on multiple metrics, but it can handle only one policy per metric type except in the case of Stackdriver monitoring metrics; an autoscaler can handle up to five policies based on Stackdriver monitoring metrics. The autoscaler calculates the recommended number of virtual machines for each policy and then scale based on the policy that provides the largest number of virtual machines in the group.

Microsoft Azure

Auto scaling

Video:

https://azure.microsoft.com/en-us/resources/videos/auto-scaling-azure-web-sites/

Serverless computing

What Is the AWS Serverless Application Repository?

The AWS Serverless Application Repository makes it easy for developers and enterprises to quickly find, deploy, and publish serverless applications in the AWS Cloud. For more information about serverless applications, see Serverless Computing and Applications on the AWS website.

The AWS Serverless Application Repository is deeply integrated with the AWS Lambda console, so that developers of all levels can get started with serverless computing without needing to learn anything new. You can use category keywords to browse for applications such as web and mobile backends, data processing applications, or chatbots. You can also search for applications by name, publisher, or event source. To use an application, you simply choose it, configure any required fields, and deploy it with a few clicks.

Google Cloud

Serverless computing

What is Serverless?

Serverless is a new paradigm of computing that abstracts away the complexity associated with managing servers for mobile and API backends, ETL, data processing jobs, databases, and more.

No upfront provisioning - Just provide your code and data, and Google dynamically provisions resources as needed.

No management of servers - Get out of the repetitive and error-prone task of managing or automating server management like scaling your cluster, OS security patches, etc.

Pay-for-what-you-use - Because of the dynamic provisioning and automatic scaling, you only pay for what you use.

Microsoft Azure

Serverless computing

What is serverless computing?

Serverless computing is the abstraction of servers, infrastructure, and operating systems. When you build serverless apps you don't need to provision and manage any servers, so you can take your mind off infrastructure concerns. Serverless computing is driven by the reaction to events and triggers happening in near-real-time —in the cloud. As a fully managed service, server management and capacity planning are invisible to the developer and billing is based just on resources consumed or the actual time your code is running.

Why build serverless applications?

Benefit from a fully managed service

Spare your teams the burden of managing servers. By utilizing fully managed services, you focus on your business logic and avoid administrative tasks. With serverless architecture you simply deploy your code, and it runs with high availability.

Scale flexibly

Serverless compute scales from nothing to handle tens of thousands of concurrent functions almost instantly (within seconds), to match any workload, and without requiring scale configuration—it reacts to events and triggers in near-real time.

Only pay for resources you use

With serverless architecture, you only pay for the time your code is running. Serverless computing is event-driven, and resources are allocated as soon as they're triggered by an event. You're only charged for the time and resources it takes to execute your code—through sub-second billing.