AUTOSCALING - OVERVIEW

- There are two main ways that an application can scale:
  - **Vertical scaling**, also called scaling up and down, means changing the capacity of a resource. For example, you could move an application to a larger VM size. Vertical scaling often requires making the system temporarily unavailable while it is being redeployed. Therefore, it's less common to automate vertical scaling.
  - **Horizontal scaling**, also called scaling out and in, means adding or removing instances of a resource. The application continues running without interruption as new resources are provisioned. When the provisioning process is complete, the solution is deployed on these additional resources. If demand drops, the additional resources can be shut down cleanly and deallocated.
- All 3 cloud platforms support horizontal scaling

AUTO SCALING - AWS

- At the time of initial launch, AWS did not offer autoscaling, but the ability to programmatically create and terminate instances gave developers the flexibility to write their own code for autoscaling
- Third-party autoscaling software for AWS began appearing around April 2008. These included tools by Scalr and RightScale
- On May 18, 2009, Amazon launched its own autoscaling feature along with Elastic Load Balancing
- On-demand video provider Netflix documented their use of autoscaling with Amazon Web Services to meet their highly variable consumer needs

AUTOSCALING - AWS

- Various best practice guides for AWS use suggest using its autoscaling feature even in cases where the load is not variable
- That is because autoscaling offers two other advantages:
  - automatic replacement of any instances that become unhealthy for any reason (such as hardware failure, network failure, or application error)
  - automatic replacement of spot instances that get interrupted for price or capacity reasons, making it more feasible to use spot instances for production purposes

AUTOSCALING - AZURE

- On June 27, 2013, Microsoft announced that it was adding autoscaling support to its Windows Azure cloud computing platform
- Conditions can be set for a cloud service worker role that trigger a scale in or out operation
- The conditions for the role can be based on the CPU, disk, or network load of the role
- You can scale an application only within the limit of cores for your subscription
- To enable high availability of your application, you should ensure that it is deployed with two or more role instances
- Auto Scale only happens when all the roles are in **Ready** state

AUTOSCALING - AZURE

- You can configure scale settings for a role with either two modes **MANUAL** or **AUTOMATIC**
- Manual is as you would expect, you set the absolute count of instances
- Automatic however allows you to set rules that govern how and by how much you should scale
- You can scale
  - Always
  - Recurrence (set of days)
  - Fixed date

AUTOSCALING - GCP

- On November 17, 2014, the Google Compute Engine announced a public beta of its autoscaling feature for use in Google Cloud Platform applications
- Autoscaling is a feature of managed instance groups. A managed instance group is a pool of homogeneous instances, created from a common instance template
- Scaling using the following policies:
  - Average CPU utilization
  - HTTP load balancing serving capacity, which can be based on either utilization or requests per second
  - Stackdriver Monitoring metrics

Serverless Computing - Overview

- **Serverless computing** is a cloud **computing** execution model in which the cloud provider dynamically manages the allocation of machine resources.
- AWS offers the following products within it's serverless platform; AWS Lambda, Amazon API Gateway, Amazon Simple Storage Service, Amazon DynamoDB, Amazon SNS, Amazon SQS, AWS Step Functions, Amazon Kinesis.
- Azure offers the following products within it's serverless platform; Azure Functions, Azure Storage, Azure Cosmos DB, Azure Active Directory, Event Grid, Service Bus, Logic Apps, Azure Stream Analytics, Azure Bot Service, Cognitive Services
- GCP offers the following products within it's serverless platform; App Engine, Cloud Functions, Cloud Datastore, Cloud Storage, Cloud Pub/Sub, Apigee, Endpoints, Cloud Dataflow, BigQuery, Cloud ML Engine

AWS LAMBDA FEATURES

- Supported programming languages - Java, Node.js, C#, and Python code, with support for other languages coming in the future.
- Automatic Scaling - AWS lambda automatically scales to support the rate of incoming requests without requiring you to configure anything.
- Integration with other services - You can configure other resources such as S3 buckets, DynamoDB table, or Kinesis stream to trigger lambda functions.
- Fault Tolerance - AWS Lambda maintains compute capacity across multiple Availability Zones in each region to help protect your code against individual machine or data center facility failures.
- Pricing model - Billing is metered in increments of 100 milliseconds, making it cost-effective and easy to scale automatically from a few requests per day to thousands per second.
- Continuous Integration - To automate the deployment process, you can use the following services, CodePipeline, CodeBuild, CodeDeploy, AWS CloudFormation.

Azure Functions

- Supported programming languages - C#, F#, or Javascript.
- Automatic Scaling - Azure Functions uses a component called the *scale controller* to monitor the rate of events and determine whether to scale out or scale in.
- Integration with other services - The following service integrations are supported. Azure Cosmos DB, Azure Event Hubs, Azure Event Grid, Azure Mobile Apps, Azure Notification Hubs, Azure Service Bus (queues and topics), Azure Storage (blob, queues, and tables), GitHub (webhooks), On-premises (using Service Bus), Twilio (SMS messages).
- Pricing model - **Consumption plan :** you only pay for the time that your code runs, **App Service plan: If** you are already using App Service for your other applications, you can use the same plan.
- Continuous integration - Code your functions right in the portal or set up continuous integration and deploy your code through GitHub, Visual Studio Team Services, and other services.

GCP – Cloud Functions

- Supported programming languages – Go, Java, .NET, Node.js, PHP, Python, Ruby.
- Automatic Scaling - Cloud Functions automatically manages and scales underlying infrastructure with the size of workload.
- Integration with other services - Cloud Functions allows you to trigger your code from Google Cloud Platform, Firebase, and Google Assistant, or call it directly from any web, mobile, or backend application via HTTP.
- Pricing model - Pay only while your function is executing, metered to the nearest 100 milliseconds, and pay nothing after your function finishes.
- Continuous integration - You can configure a continuous integration and deployment (CI/CD) platform such as Cloud Container Builder to run your existing Cloud Functions tests on an ongoing basis.
- Local Emulator - The Cloud Functions Emulator allows you to **deploy**, **run**, and **debug** your Cloud Functions on your local machine before deploying them to the production Cloud Functions service.

WHAT IS LOAD BALANCING

- A load balancer is a device that acts as a reverse proxy and distributes network or application traffic across a number of servers. Load balancers are used to increase capacity (concurrent users) and reliability of applications.
- Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any single resource
- Using multiple components with load balancing instead of a single component may increase reliability and availability through redundancy.
- Load balancing usually involves dedicated software or hardware, such as a multilayer switch or a Domain Name System server process.

AWS - Elastic Load Balancing

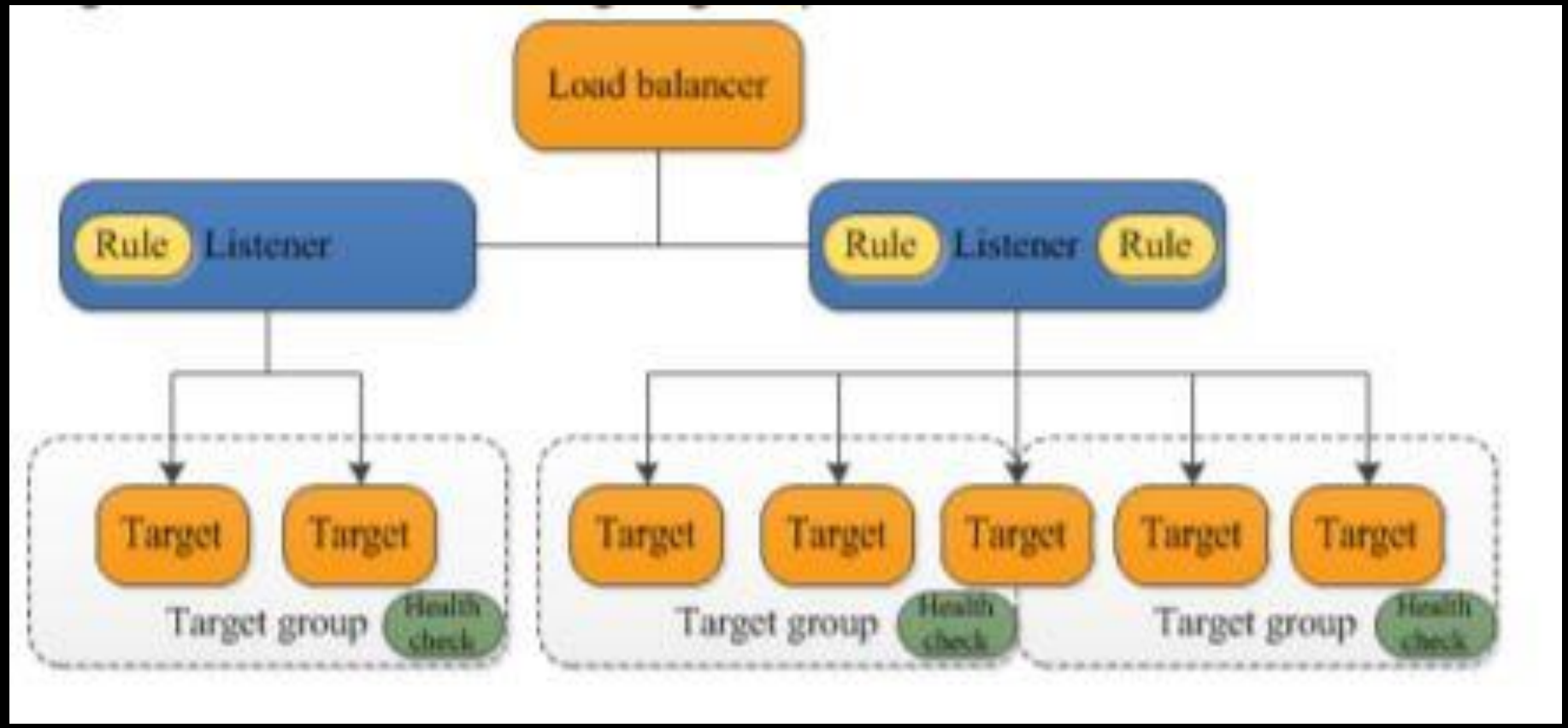
- Elastic Load Balancing distributes incoming application traffic across multiple EC2 instances, in multiple Availability Zones. This increases the fault tolerance of your applications.
- The load balancer serves as a single point of contact for clients, which increases the availability of your application. You can add and remove instances from your load balancer as your needs change, without disrupting the overall flow of requests to your application. Elastic Load Balancing scales your load balancer as traffic to your application changes over time, and can scale to the vast majority of workloads automatically.
- 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

Types of Load Balancers

Elastic Load Balancing supports three types of load balancers:

- Application Load Balancers
- Network Load Balancers
- Classic Load Balancers

You can select a load balancer based on your application needs



LOAD BALANCER FLOW CHART

Application & Classic Load Balancer Overview

- An Application Load Balancer functions at the application layer, the seventh layer of the Open Systems Interconnection (OSI) model.
- After the load balancer receives a request, it evaluates the listener rules in priority order to determine which rule to apply, and then selects a target from the target group for the rule action using the round robin routing algorithm.
- We can configure listener rules to route requests to different target groups based on the content of the application traffic. You can add one or more listeners to your load balancer.
- You can configure *health checks*, which are used to monitor the health of the registered instances so that the load balancer only sends requests to the healthy instances.

Network Load Balancer Overview

- A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. It can handle millions of requests per second.
- After the load balancer receives a connection request, it selects a target from the target group for the default rule.
- It attempts to open a TCP connection to the selected target on the port specified in the listener configuration.

Azure Load Balancer

- Azure Load Balancer allows you to scale your applications and create high availability for your services. Load Balancer supports inbound as well as outbound scenarios, and provides low latency, high throughput, and scales up to millions of flows for all TCP and UDP applications
- Load Balancer will distribute new inbound flows arriving on the load balancer's frontend to backend pool instances according to rules and health probes.
- Additionally, a public Load Balancer can also provide outbound connections for virtual machines inside your virtual network by translating their private IP addresses to public IP addresses.
- Azure Load Balancer is available in two different SKUs: Basic and Standard

STANDARD SKU VS BASIC SKU

	Standard SKU	Basic SKU
Backend pool size	up to 1000 instances	up to 100 instances
Backend pool endpoints	any virtual machine in a single virtual network, including blend of virtual machines, availability sets, virtual machine scale sets.	virtual machines in a single availability set or virtual machine scale set
Availability Zones	zone-redundant and zonal frontends for inbound and outbound, outbound flows mappings survive zone failure, cross-zone load balancing	/
Diagnostics	Azure Monitor, multi-dimensional metrics including byte and packet counters, health probe status, connection attempts (TCP SYN), outbound connection health (SNAT successful and failed flows), active data plane measurements	Azure Log Analytics for public Load Balancer only, SNAT exhaustion alert, backend pool health count
HA Ports	internal Load Balancer	/
Secure by default	default closed for public IP and Load Balancer endpoints and a network security group must be used to explicitly whitelist for traffic to flow	default open, network security group optional

STANDARD SKU VS BASIC SKU CONT....

Outbound connections	Multiple frontends with per rule opt-out. An outbound scenario *must* be explicitly created for the virtual machine to be able to use outbound connectivity. VNet Service Endpoints can be reached without outbound connectivity and do not count towards data processed. Any public IP addresses, including Azure PaaS services not available as VNet Service Endpoints, must be reached via outbound connectivity and count towards data processed. When only an internal Load Balancer is serving a virtual machine, outbound connections via default SNAT are not available. Outbound SNAT programming is transport protocol specific based on protocol of the inbound load balancing rule.	Single frontend, selected at random when multiple frontends are present. When only internal Load Balancer is serving a virtual machine, default SNAT is used.
Multiple frontends	Inbound and outbound	Inbound only
Management Operations	Most operations < 30 seconds	60-90+ seconds typical
SLA	99.99% for data path with two healthy virtual machines	Implicit in VM SLA
Pricing	Charged based on number of rules, data processed inbound or outbound associated with resource	No charge

GCP LOAD BALANCER

Types of load balancing:

1. Global external load balancing
   - HTTP(S) load balancing
   - SSL Proxy load balancing
   - TCP Proxy load balancing
2. Regional external load balancing
   - Network load balancing
3. Regional internal load balancing
   - Internal load balancing

HTTP(S) LOAD BALANCING

- HTTP(S) load balancing can balance HTTP and HTTPS traffic across multiple backend instances, across multiple regions.
- Your entire app is available via a single global IP address, resulting in a simplified DNS setup.
- HTTP(S) load balancing is scalable, fault-tolerant, requires no pre-warming, and enables content-based load balancing. For HTTPS traffic, it provides SSL termination and load balancing.
- HTTP(S) load balancing supports both IPv4 and IPv6 addresses for client traffic. Client IPv6 requests are terminated at the global load balancing layer, then proxied over IPv4 to your backends.
- HTTP requests can be load balanced based on port 80 or port 8080. HTTPS requests can be load balanced on port 443

SSL LOAD BALANCING

- With SSL (TLS) proxying for your SSL traffic, you can terminate SSL sessions at the global load balancing layer, then forward the traffic to your virtual machine instances using SSL (recommended) or TCP.
- SSL proxy is a global load balancing service. You can deploy your instances in multiple regions, and the load balancer automatically directs traffic to the closest region that has capacity. If the closest region is at capacity, the load balancer automatically directs new connections to another region with available capacity. Existing user connections remain in the current region.
- SSL proxy provides SSL termination for your non-HTTPS traffic with load balancing.
- With SSL (TLS) proxying for your SSL traffic, you can terminate SSL sessions at the global load balancing layer, then forward the traffic to your virtual machine instances using SSL (recommended) or TCP

TCP Proxy Load Balancing

- TCP load balancing can spread TCP traffic over a pool of instances within a Compute Engine region.
- Google Cloud Platform (GCP) TCP Proxy Load Balancing allows you to use a single IP address for all users around the world. GCP TCP proxy load balancing automatically routes traffic to the instances that are closest to the user.
- Cloud TCP Proxy Load Balancing is intended for non-HTTP traffic. For HTTP traffic, HTTP(S) Load Balancing is recommended instead. For proxied SSL traffic, use SSL Proxy Load Balancing.
- TCP Proxy Load Balancing supports both IPv4 and IPv6 addresses for client traffic. Client IPv6 requests are terminated at the global load balancing layer, then proxied over IPv4 to your backends.
- It is scalable, does not require pre-warming, and health checks help ensure only healthy instances receive traffic.

Network load balancing

- Network load balancing distributes traffic among a pool of instances within a region. Network load balancing can balance any kind of TCP/UDP traffic
- Network load balancing allows you to balance load of your systems based on incoming IP protocol data, such as address, port, and protocol type.
- Network load balancing uses forwarding rules that point to target pools, which list the instances available for load balancing and define which type of health check that should be performed on these instances. See the Network load balancing example for more information.
- Network load balancing is a regional, non-proxied load balancer. You can use it to load balance UDP traffic, and TCP and SSL traffic on ports that are not supported by the SSL proxy and TCP proxy load balancers.
- A Network load balancer is a pass-through load balancer. It does not proxy connections from clients

Internal load balancing

- Internal load balancing distributes traffic from Google Cloud Platform virtual machine instances to a group of instances in the same region.
- Internal Load Balancing enables you to run and scale your services behind a private load balancing IP address which is accessible only to instances internal to your Virtual Private Cloud (VPC).
- Internal Load Balancing works with auto mode VPC networks, custom mode VPC networks, and legacy networks.
- Internal Load Balancing can also be implemented with regional managed instance groups. This allows you to autoscale across a region, making your service immune to zonal failures.