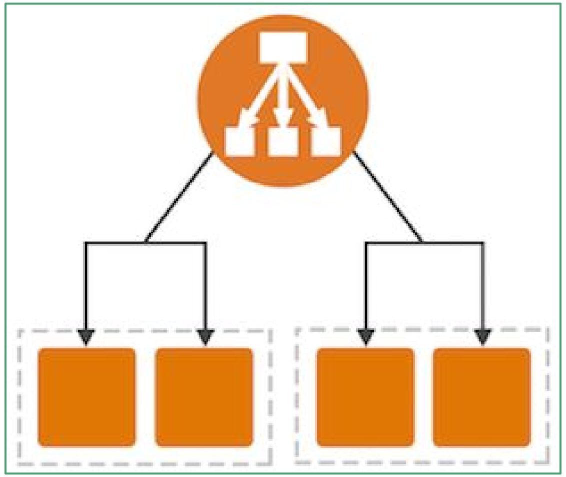
Amazon ELB

What Is Elastic Load Balancing?

- **Distributes** incoming application traffic across multiple EC2 instances, in multiple Availability Zones
- Serves as a **single point of contact** for clients
- **Add and remove instances dynamically** from your load balancer as your needs change, without disrupting the overall flow of requests to your application
- **Scales your load balancer** as traffic to your application changes over time



Manage your load balancers through:

- Console / CLI / SDK / Query API

Load Balancer Types

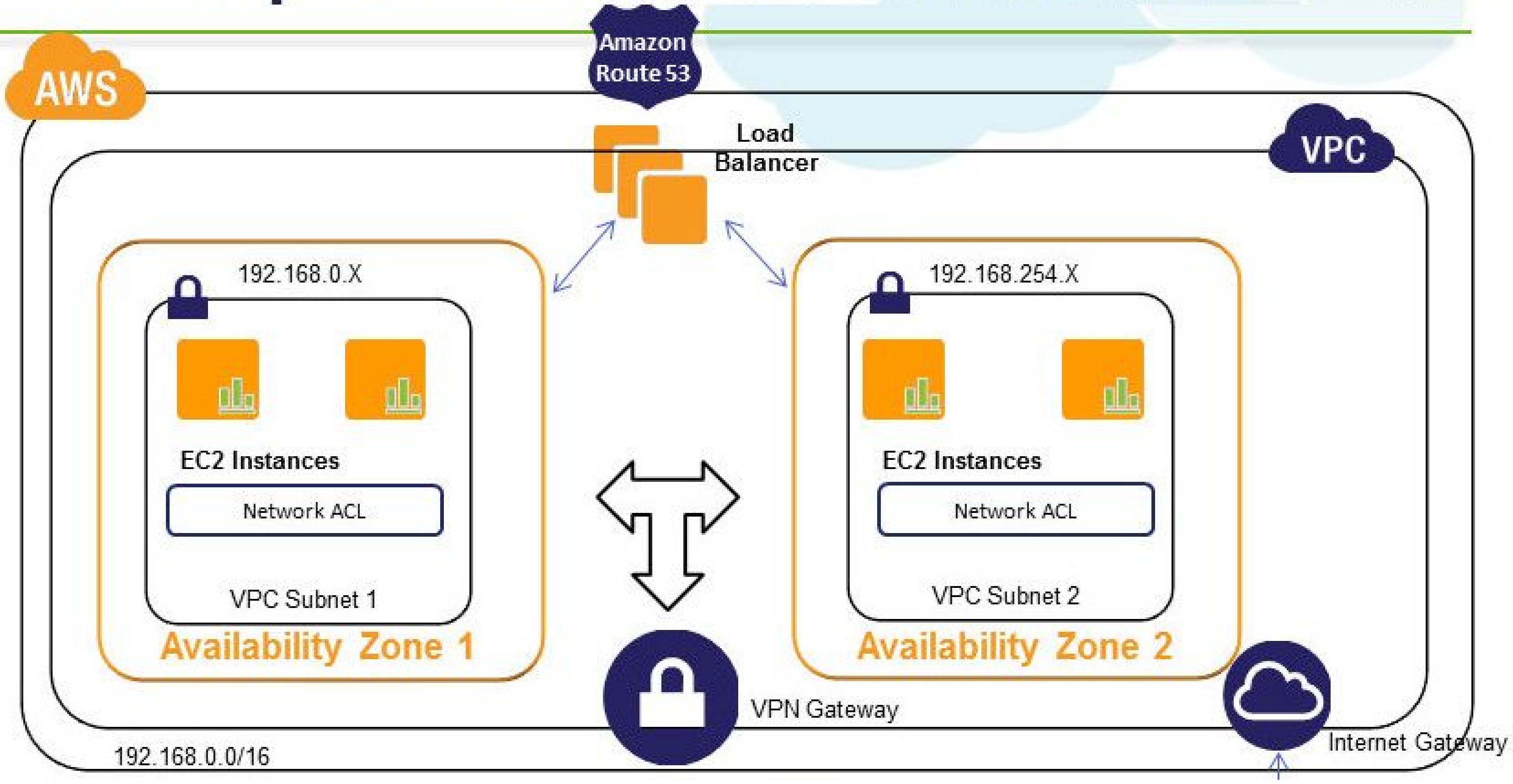
Two types of Load Balancer:

- Application Load Balancers
- Classic Load Balancers

Related Services

- **Amazon EC2**: Routes Traffic to EC2 Instances
- **Amazon ECS**: Routes traffic to your docker containers
- **Auto Scaling**: instances that are launched by Auto Scaling are automatically registered with the load balancer, and instances that are terminated by Auto Scaling are automatically de-registered from the load balancer
- **Amazon CloudWatch**: monitor your load balancer and take action as needed
- **Amazon Route 53**: assigns URLs to your resources, such as load balancers

Multiple AZ inside Amazon VPC



How Elastic Load Balancing Works

A load balancer **accepts incoming traffic from clients and routes requests to its registered EC2 instances in one or more Availability Zones**. The load balancer also **monitors the health of its registered instances** and ensures that it **routes traffic only to healthy instances**. When the load balancer detects an unhealthy instance, it **stops routing traffic to that unhealthy instance**, and then resumes routing traffic to that instance when it detects that the instance is healthy again.

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 and a protocol and port number for connections from the load balancer to the instances.

Kinds of Load Balancers

Elastic Load Balancing supports two types of load balancers:

- **Application Load Balancers**
- **Classic Load Balancers**

- With Classic Load Balancer, you register instances with the load balancer. With an Application Load Balancer, you register the instances as targets in a target group, and route traffic to a target group.
- Classic Load Balancer, we recommend that you enable multiple Availability Zones. With an Application Load Balancer, we require you to enable multiple Availability Zones
- Cross-zone load balancing is always enabled for an Application Load Balancer and is disabled by default for a Classic Load Balancer

Comparing the CLB and the ALB

- **High Availability:** Application Load Balancer **automatically supports cross-zone load balancing**, whereas you will need to enable it in the Classic Load Balancer, where it is disabled by default.
- **Health Checks:** Both load balancers have the ability to detect EC2 instance health. If the ELB detects an unhealthy instance, it will avoid sending traffic to the unhealthy instance or instances and instead send it to only healthy instances. In the case of **ALB, you can specify a range of HTTP response codes that constitute a healthy response**.
- **VPC:** Both load balancer offerings have VPC support, but the **Classic Load Balancer also supports EC2-Classic**. The Application Load Balancer does not support EC2-Classic.
- **Dynamic Port Mapping:** Classic Load Balancer only supports fixed mappings between listener and target hosts. The Application Load Balancer supports dynamic port mapping using the EC2 Container Service. (so that multiple tasks from the same service are allowed per container instance)
- **Protocols:** HTTP and HTTPS are supported by both ELBs. The Classic Load Balancer also supports TCP and SSL in addition to the above. Application Load Balancer supports HTTP/2 and WebSockets in addition to HTTP and HTTPS. TCP and SSL listeners are not currently supported by ALB.
- Backend Server Auth: Classic Load Balancer supports backend server auth. Application Load Balancer does not.

Comparing the CLB and the ALB

- **Backend Server Auth:** Classic Load Balancer supports backend server auth. Application Load Balancer does not.
- **Cloudwatch Metrics:** CLB supports Cloudwatch Metrics. **ALB supports an enhanced version, which provides for per port and per path monitoring of load balanced services**, where a range of acceptable HTTP response codes are permissible. In contrast, Classic Load Balancer only allows for monitoring of a single port, with the HTTP 200 response code. ALB includes three additional Cloudwatch metrics: connections per hour, active connections, and overall traffic volume.
- **Access Logs:** Classic Load Balancer supports ELB access logs. Application Load Balancer supports an enhanced version, that adds type of request (e.g. HTTP/HTTPS, HTTP/2 over SSL/TLS, WebSockets, and WebSockets over SSL/TLS), and the target Amazon Resource Name.
- **Path-based routing:** Application Load Balancers support path-based routing and priority rules (so that multiple services can use the same listener port on a single Application Load Balancer). This feature is not supported by CLB.
- **Deletion Protection:** ALB supports deletion protection, whereas CLB does not.

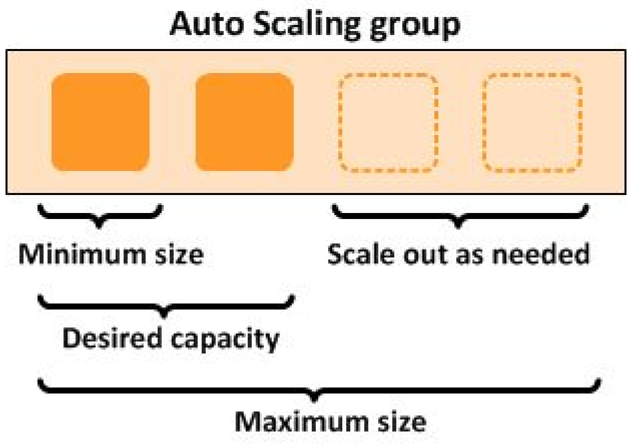
Configuring Elastic Load Balancing

- Idle Connection Timeout
  - idle timeout to 60 seconds
- Cross-Zone Load Balancing
  - ALB
- Connection Draining
  - stops sending requests to instances that are deregistering or unhealthy timeout before reporting the instance as deregistered can be set between 1 and 3,600 seconds (the default is 300 seconds).
- Proxy Protocol
  - Forwards requests to the back-end instances without modifying the request headers. If you enable Proxy Protocol, a human-readable header is added to the request header with connection information such as the source IP address, destination IP address, and port numbers.
- Sticky Sessions
  - enables the load balancer to bind a user’s session to a specific instance
- Health Checks
  - is a ping, a connection attempt, or a page that is checked periodically.. You can set the time interval between health checks and also the amount of time to wait to respond in case the health check page includes a computational aspect.

AWS Autoscaling

What Is Auto Scaling?

Auto Scaling is a web service designed to launch or terminate Amazon EC2 instances automatically based on user-defined policies, schedules, and health checks.



Auto Scaling Key Components

- Auto Scaling Groups
  - Your EC2 instances are organized into groups so that they can be treated as a logical unit for the purposes of scaling and management. When you create a group, you can specify its minimum, maximum, and, desired number of EC2 instances.
- Launch configurations
  - Your group uses a launch configuration as a template for its EC2 instances. When you create a launch configuration, you can specify information such as the AMI ID, instance type, key pair, security groups, and block device mapping for your instances.
- Scaling plans
  - A scaling plan tells Auto Scaling when and how to scale. For example, you can base a scaling plan on the occurrence of specified conditions (dynamic scaling) or on a schedule.

Accessing Auto Scaling

- AWS Management Console
- AWS Command Line Interface
- AWS Tools for Windows PowerShell
- Query API

Benefits of Auto Scaling

- **Better fault tolerance:** Auto Scaling **can detect when an instance is unhealthy, terminate it, and launch an instance** to replace it. You can also configure **Auto Scaling to use multiple Availability Zones**. If one Availability Zone becomes unavailable, Auto Scaling can launch instances in another one to compensate.
- **Better availability:** Auto Scaling can help you ensure that your application **always has the right amount of capacity** to handle the current traffic demands.
- **Better cost management:** Auto Scaling can dynamically increase and decrease capacity as needed. Because you **pay for the EC2 instances you use**, you save money by launching instances when they are actually needed and terminating them when they aren't needed.

Amazon CloudWatch

What is Amazon CloudWatch?

Amazon CloudWatch is a **service that you can use to monitor your AWS resources and your applications** in real time. With Amazon CloudWatch, you can collect and track metrics, create alarms that send notifications, and make changes to the resources being monitored based on rules you define.

Amazon CloudWatch offers either **basic or detailed monitoring** for supported AWS products. **Basic monitoring sends data points to Amazon CloudWatch every five minutes** for a limited number of pre selected metrics at no charge. **Detailed monitoring sends data points to Amazon CloudWatch every minute** and allows data aggregation for an additional charge. If you want to use detailed monitoring, you must enable it—basic is the default.

Amazon CloudWatch **supports monitoring and specific metrics for most AWS Cloud services**

Amazon CloudWatch has some limits that you should keep in mind when using the service. Each AWS account is limited to 5,000 alarms per AWS account, and metrics data is retained for two weeks by default. For longer you can move the Cloudwatch logs to S3 / Glacier.

Amazon RDS

What Is Amazon RDS?

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

RDS Benefits

- Scale CPU, memory, storage, and IOPS independently
- Manages backups, software patching, automatic failure detection, and recovery
- Does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges.
- Automated backups performed, create snapshots, restore process is reliable and efficient.
- High availability with a primary instance and a synchronous secondary instance that you can failover to when problems occur
- Can also use MySQL, MariaDB, or PostgreSQL Read Replicas to increase read scaling
- Available databases: MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server, and the new, MySQL-compatible Amazon Aurora DB engine
- Can control who can access your RDS databases by using AWS IAM to define users and permissions. You can also help protect your databases by putting them in a VPC

Amazon Redshift

What is Amazon Redshift?

Amazon Redshift is a **fast, powerful, fully managed, petabyte-scale data warehouse service in the cloud**. It s a **relational database designed for OLAP** and optimized for **high performance analysis and reporting of very large datasets**.

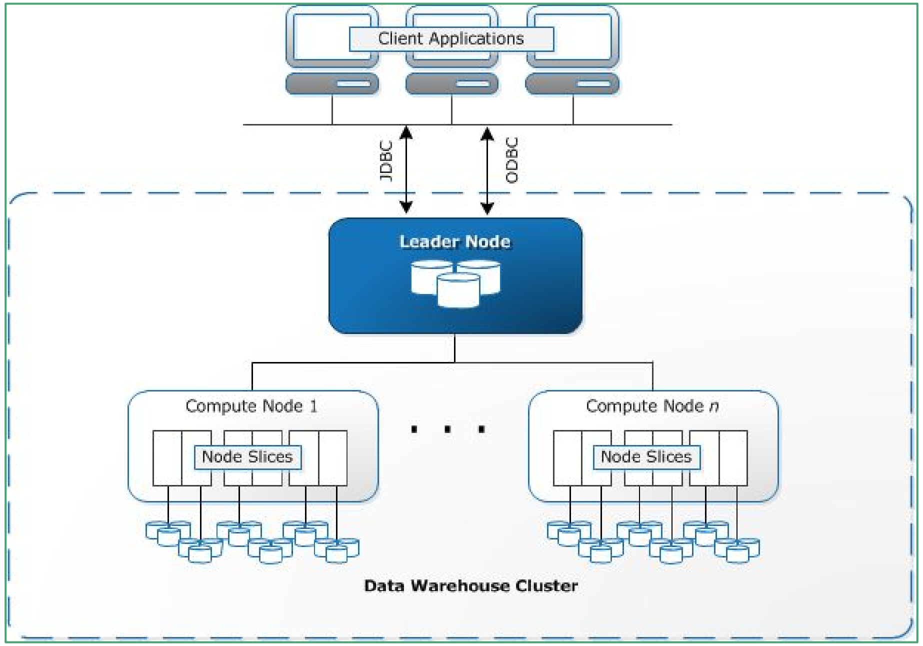
- Lower cost
- Easy analysis of data very quickly.
- Fast querying capabilities over structured data using SQL commands (SELECT/INSERT / UPDATE etc)
- Connectivity via ODBC or JDBC, Amazon Redshift integrates well with various data loading, reporting, data mining, and analytics tools.
- Based on industry-standard PostgreSQL
- Automatically monitors your nodes and drives to help you recover from failures.
- COPY command is much more efficient alternative than repeatedly calling INSERT, also supports multiple data sources
- Fastest way to load data into Amazon Redshift is doing bulk data loads from flat files stored in an Amazon Simple Storage Service (Amazon S3) bucket or from an Amazon DynamoDB table

Clusters and Nodes

-> Key component of an Amazon Redshift data warehouse is a cluster. A cluster is composed of a leader node and one or more compute nodes. Client application interacts directly only with the leader node.

-> Each cluster contains one or more databases. User data for each table is distributed across the compute nodes.

-> Amazon Redshift allows you to resize a cluster to add storage and compute capacity over time as your needs evolve



Redshift Distribution Strategy

When creating a table, you can choose between one of three distribution styles: EVEN, KEY, or ALL.

- EVEN distribution This is the **default option and results in the data being distributed across the slices in a uniform fashion** regardless of the data.
- KEY distribution With KEY distribution, **the rows are distributed according to the values in one column**. The leader node will store matching values close together and increase query performance for joins.
- ALL distribution With ALL, a **full copy of the entire table is distributed to every node**. This is useful for lookup tables and other large tables that are not updated frequently

Amazon DynamoDB

What is Amazon DynamoDB?

Amazon DynamoDB is a **fully managed NoSQL database service** that provides **fast and low latency performance** that scales with ease. Amazon DynamoDB lets you offload the administrative burdens of operating a distributed NoSQL database and focus on the application. Amazon DynamoDB significantly simplifies the hardware provisioning, setup and configuration, replication, software patching, and cluster scaling of NoSQL databases.

- Simplified database and cluster management
- Improve reliability with automatic replication
- Distributes the data and traffic for a table over multiple partitions
- All table data is stored on high performance SSD disk drives
- Automatic HA and durability by replicating across multiple AZ within Region
- 400KB limit of item size

```
{
    Id = 101
    ProductName = "Book 101 Title"
    ISBN = "123–1234567890"
    Authors = [ "Author 1", "Author 2" ]
    Price = 2.88
    Dimensions = "8.5 x 11.0 x 0.5"
    PageCount = 500
    InPublication = 1
    ProductCategory = "Book"
}
```

Primary Key

The primary key uniquely identifies each item in a table. The primary key can be simple (partition key) or composite (partition key and sort key).

The sort key allows for searching within a partition. For example, an Orders table with primary attribute CustomerId and sort attribute OrderTimestamp would allow for queries for all orders by a specific customer in a given date range.

Secondary Indexes
When you create a table with a partition and sort key (formerly known as a hash and range key), you can optionally define one or more secondary indexes on that table. A secondary index lets you query the data in the table using an alternate key, in addition to queries against the primary key.

Indexes Types

DynamoDB supports two different kinds of indexes:

- Global Secondary Index The global secondary index is an index with a partition and sort key(optional) that can be different from those on the table. You can **create or delete a global secondary index on a table at any time**. A global secondary index is considered "global" because queries on the index can span all of the data in a table, across all partitions
- Local Secondary Index The local secondary index is an index that has the same partition key attribute as the primary key of the table, but a different sort key. **You can only create a local secondary index when you create a table**. A local secondary index is "local" in the sense that every partition of a local secondary index is scoped to a table partition that has the same hash key.

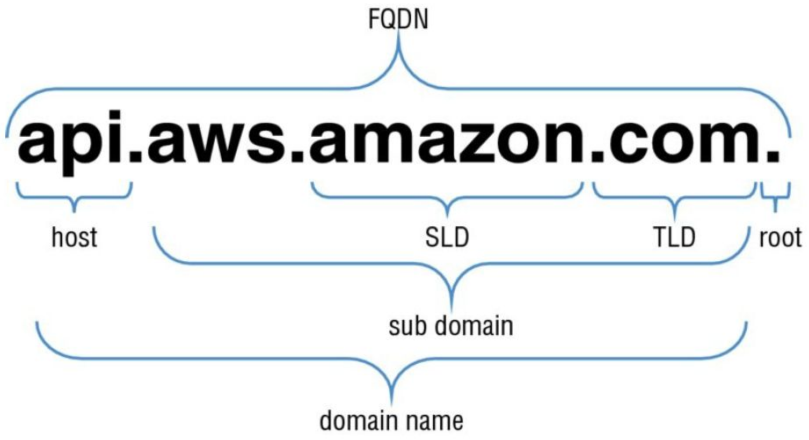
Test

Test

Amazon Route 53

Domain Name System (DNS) Concepts

- Top-Level Domains (TLDs)
- Domain Names
- IP Addresses
- Hosts
- Subdomains
- Fully Qualified Domain Name (FQDN)
- Name Servers
  - Computer designated to translate domains into IP
- Zone Files
  - Text file containing mapping between domain names and IP



Record Types

Each zone file contains records.

- Start of Authority (SOA) Record
  - Mandatory and identifies the base DNS information about the domain. Each zone contains a single SOA record.
- A and AAAA
  - Map a host to an IPv4/IPv6 IP address
- Canonical Name (CNAME)
  - Defines an alias for the CNAME for your server
- Mail Exchange (MX)
  - Defines the mail servers used for a domain and ensure that email messages are routed correctly.
  - Points to host defined by an A or AAAA record and not one defined by a CNAME
- Name Server (NS)
  - Used by TLD servers to direct traffic to the DNS server that contains the authoritative DNS records.
- Sender Policy Framework (SPF)
  - Used to combat spam. tells a mail server what IP addresses are authorized to send email from domain name.
- Text (TXT)
  - provides the ability to associate some arbitrary and unformatted text with a host or other name

What is Amazon Route 53?

Amazon Route 53 is a highly available and scalable cloud DNS web service that is designed to give developers and businesses an extremely reliable and cost-effective way to route end users to Internet applications.

Amazon Route 53 performs three main functions:

- **Domain registration:** Amazon Route 53 lets you register domain names, such as example.com.
- **DNS service:** Amazon Route 53 translates friendly domain names like www.example.com into IP addresses like 192.0.2.1. Amazon Route 53 responds to DNS queries using a global network of authoritative DNS servers, which reduces latency. To comply with DNS standards, responses sent over User Datagram Protocol (UDP) are limited to 512 bytes in size. Responses exceeding 512 bytes are truncated, and the resolver must re-issue the request over TCP.
- **Health checking:** Amazon Route 53 sends automated requests over the Internet to your application to verify that it's reachable, available, and functional.

Route 53 routing policies

- **Simple:** Most commonly used when you have a single resource that performs a given function for your domain
- **Weighted:** Used when you want to route a percentage of your traffic to one particular resource or resources
- **Latency-Based:** Used to route your traffic based on the lowest latency so that your users get the fastest response times
- **Failover:** Used for DR and to route your traffic from your resources in a primary location to a standby location
- **Geolocation:** Used to route your traffic based on your end user’s location

Amazon SES

What is Amazon SES?

Amazon SES is an email sending and receiving service that provides an easy, cost-effective way for you to send email.