Elastic Load Balancer (ELB)

Load Balancing means distributing the work across multiple entities. In Networking, Load Balancer automatically distributes incoming traffic to applications running across multiple systems. In the case of Amazon cloud, ELB distributes the network traffic across multiple ec2 instances. With Amazon ELB one can:

- Configure multiple ec2 instances behind the ELB to distribute the work across them.
- Instances can be added or removed behind the load balancer as needed without disrupting the service. This improves availability of application
- If instance fails, ELB automatically reroutes the traffic to remaining ec2 instances
- All clients first connect to ELB. ELB keeps pool of connections with ec2 instances and route traffic to instance that has least amount of active connections
- You can have as many ec2 instances behind ELB as you like. Due to the Elastic nature of AWS, additional ELB will be added by AWS to handle additional workload.
- One can offload encryption/decryption to ELB. Thus ELB can act as HTTPS SSL (Secure Socket Layer) termination point and then route unencrypted traffic to ec2 instances behind it. SSL protocol is primarily used to encrypt confidential data over the insecure (Internet) network.
- One only needs to assign the DNS name of the company's Domain to ELB, for example, *mystore.cloudperf.net* to ELB using route53 API. Since ELB balances the load without exposing what is running behind it, ec2 instances (behind the ELB) do not need public IP or DNS name. ELB can connect to ec2 instances using private IP addresses.

AWS recommends:

- Assign CNAME to ELB, if you want to use a friendly name to access ELB. Don't assign "A" resource record to ELB considering ELB can change overtime.
- Spread ec2 instances across Availability zones (AZ). To make ELB spread the load across AZ, attach subnets associated with AZ to ELB. http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-manage-subnets.html
- Remember, By default, VPC public and private addresses are spanned across multiple AZ. Each AZ gets a unique subnet. To see what subnets are assigned to AZ in the VPC, run:
    - *$aws ec2 describe-subnets --region us-east-1*

```
SUBNETS  False  us-east-1c  4091  172.31.0.0/20   True  True  available  subnet-a241b6fb  vpc-eb69dd8e
...
SUBNETS  False  us-east-1d  4091  172.31.64.0/20  True  True  available  subnet-579ee432  vpc-eb69dd8e
SUBNETS  False  us-east-1a  4091  172.31.32.0/20  True  True  available  subnet-f07d6bd8  vpc-eb69dd8e
SUBNETS  False  us-east-1f  4091  172.31.80.0/20  True  True  available  subnet-5143d05d  vpc-eb69dd8e
```

ELB performs a health check of registered ec2 instances by periodically pinging or connecting to the service running on instances. Instance is marked **"OutOfService"** on failure of health check and taken out of the traffic routing. When ELB is configured, it uses default health checks as listed at URL:
http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-healthchecks.html

One can set or update the health checks of ELB using aws console or aws cli.

List load balancer:

$ aws elb describe-load-balancers --region us-east-1

List instance health behind load balancer

$ aws elb describe-instance-health --load-balancer-name <elb-name> --region us-east-1

LAB: Load balance the web traffic across multiple ec2 instances via ELB

For this exercise, we will create two instances in different Availability Zones (AZ) and place them behind ELB. Remember each AZ is a physical building or datacenter within a region. Some regions have more AZ than others. AZ helps service availability in case of a single AZ failure.

Find Availability Zones (AZ) names in us-east-1 region

$ aws ec2 describe-availability-zones --region us-east-1

AVAILABILITYZONE	*us-east-1a*	*available*	*us-east-1*
AVAILABILITYZONE	*us-east-1b*	*available*	*us-east-1*
AVAILABILITYZONE	*us-east-1c*	*available*	*us-east-1*
AVAILABILITYZONE	***us-east-1d***	***available***	***us-east-1***
AVAILABILITYZONE	***us-east-1e***	***available***	***us-east-1***

Start two ec2 instances, one in each AZ (us-east-1d, us-east-1e):

$ aws ec2 run-instances --key-name <> --security-groups <> --count 1 --instance-type t2.micro --region us-east-1 --placement AvailabilityZone=us-east-1d --image-id ami-033a3273401a27142

$aws ec2 run-instances --key-name cloudperf-netflix --security-groups UCSC --count 1 --instance-type t2.micro --region us-east-1 --placement AvailabilityZone=us-east-1d --image-id ami-033a3273401a27142

$aws ec2 run-instances --key-name cloudperf-netflix --security-groups UCSC --count 1 --instance-type t2.micro --region us-east-1 --placement AvailabilityZone=us-east-1e --image-id ami-033a3273401a27142

$ Create an HTTP ELB via aws cli tool. You need to find out about subnets created in your default VPC. To list all subnets type:

$ aws ec2 describe-subnets --region us-east-1

SUBNETS	*us-east-1e*	*4091*	*172.31.48.0/20*	*True*	*True*	*available*	*subnet-xxxx*	*vpc-36223150*
SUBNETS	*us-east-1a*	*4091*	*172.31.16.0/20*	*True*	*True*	*available*	*subnet-xxxx*	*vpc-36223150*
SUBNETS	*us-east-1d*	*4091*	*172.31.64.0/20*	*True*	*True*	*available*	*subnet-xxxx*	*vpc-36223150*
SUBNETS	*us-east-1b*	*4091*	*172.31.32.0/20*	*True*	*True*	*available*	*subnet-xxxx*	*vpc-36223150*
SUBNETS	*us-east-1c*	*4091*	*172.31.0.0/20*	*True*	*True*	*available*	*subnet-xxxx*	*vpc-36223150*

Create a load balancer to use HTTP protocol that can route traffic to subnets

$ aws elb create-load-balancer --load-balancer-name cloudstudents-elb --listeners Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80 --security-groups sg-75f1ec0a --subnet subnet-579ee432 subnet-44a2c07e --region us-east-1

NOTE: security-group should be id not name

If successful, you should see ELB DNS name:

Name: cloudstudents-elb, "DNSName": "cloudstudents-elb-1247310216.us-east-1.elb.amazonaws.com"

Register instances with ELB

$aws elb register-instances-with-load-balancer --load-balancer-name <ELB-NAME> --instances <instance list> --region us-east-1

NOTE: You should provide ELB name not DNS name

If successful, you should see a list of instance names added behind ELB. You can add or remove instances as desired using: *"aws unregister-instances-from-load-balancer …"*. ELB will redistribute the load across all remaining instances automatically.

Create a CNAME, if desired, in Route53 resource record so that you don't need to use randomly generated elb DNS names

To verify if elb is load balancing web traffic across two instances, you can use *"curl" to send a query to a graphite server that will return an instance private IP address* . Replace **myelb.mydomain.net** with your elb DNS name or CNAME.

Install jq package for processing json output.

$ sudo apt-get install jq -y

Since graphite server is running on port 80, we can send graphite query and process the JSON returned with private ip address of instance

while true; do curl -s -X GET cloudstudents-elb-<>.us-east-1.elb.amazonaws.com/metrics/find/format=text?query=carbon.agents.*|jq '.[] | .text'|head -1; sleep 2; done

In another terminal window, deregister and register different instances.

$ aws elb deregister-instances-from-load-balancer --load-balancer-name=<> --instances <instance-id> --region us-east-1

$ aws elb register-instances-with-load-balancer --load-balancer-name=<> --instances <instance-id> --region us-east-1

You should see that elb continues to service requests as long as there is one instance available.

References:

How to setup HTTPS ELB

Steps to create HTTPS (secure socket layer) ELB are the same as HTTP ELB except you need to create a self signed certificate or CA signed certificate and install it on ELB. ELB uses the certificate for SSL connection termination and then decrypts client requests before forwarding it to the instance behind the ELB. SSL protocol uses an X.509 certificate (SSL server certificate) to authenticate both the client and the back-end application. X.509 is a digital form of ID signed by certificate authority (CA) and contains: Validity period, a public key, a serial number, and the digital signature of the issuer.
Detail instructions are provided at URL:

https://aws.amazon.com/blogs/aws/elastic-load-balancer-support-for-ssl-termination/
http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-listener-config.html
https://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-create-https-ssl-load-balancer.html#configure_ciphers_clt
http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/ssl-server-cert.html

ELB shortcoming:

- Considering ELB is an extra hop that clients have to go through to reach ec2 instance, it may add latencies for high volume network traffic. For example, some SaaS applications aggregate logs or metrics from client systems, that are streaming tens of thousands of events per second, may hit latency issues by having ELB in the middle.

- ELB does not support forwarding UDP traffic. Accepted protocols are: HTTP, HTTPS, TCP, TCP over SSL and TCP port 25, 80, 443, 587, and 1024 - 65535. Recently, ELB starts supporting Proxy protocol
- Amazon provides limited visibility on ELB performance. Behind the scenes, ELBs are scaled up or down to deal with varying traffic patterns, AWS proprietary algorithms determine the size of ELB farms. AWS CloudWatch exposes a metric *"HTTPCode_ELB_500,* but it does not tell latency was due to ELB or the instance behind it. Best action you can take is to warm up the ELB by generating fake traffic so that they can scale up before real traffic starts.

Application ELB

https://aws.amazon.com/blogs/aws/new-aws-application-load-balancer/

Application elb (ALB) replaces standard elb (ELB). Unlike ELB, ALB operates at the application layer (OSI layer 7), as compared to ELB that operates at OSI layer 4 (network layer), and makes routing and load balancing decisions on application HTTP and HTTPS traffic. ELB does not look at the payload and thus is unaware of the specifics of HTTP and HTTPS. ALB, on the other hand, inspects packets and processes HTTP/HTTPS headers and thus is able to make more sophisticated content based routing decisions that are needed for supporting containers (Docker). It offers the same security, reliability, availability and scalability that you get from ELB. Additional features offered by ALB are:

- Content-based or Path-based Routing: You can define traffic routing rules based on the criteria such as the URL path. ALB can route traffic to multiple back-end services based on the url path accessed by the client.
    - For example, you can send requests that include **/api i**n the URL to one group of cloud instances (ALB calls it target groups) and **/mobile** URL requests to another set. This feature is useful when you build application using micro services (service oriented architecture)
    - Each ALB allows up to 10 URL-based rules to route requests to target groups.
- Container Support: Routing to multiple ports on a single EC2 instance. There can be multiple services running inside containers listening to multiple ports but hosted on a single EC2 instance. Amazon ECS (Elastic Container Service) supports dynamic ports and thus allows Docker to pick an unused port when a task is scheduled. Amazon ECS automatically adds the task to the load balancer using this port. ALB offers fine-grained port level mapping and health checks.
    - Health checks can specify a range of acceptable HTTP responses, and are accompanied by detailed error codes.
    - Due to support of content-based routing in ALB, one can collect metrics for each microservice running on an EC2 instance.
- Application Monitoring: Improved metrics and health check feature for greater visibility into application performance. CloudWatch and access log metrics are available for ALB. CloudWatch metrics include: overall traffic (in GB), number of active connections, and the connection rate per hour.

- AWS Integration: Integrate with other AWS services such as: Auto-scaling, ECS, Elastic Beanstalk, IAM..
- ALB is ideal for real-time streaming via HTTP/2 and Websocket workloads. ALB handles streaming workloads in a streaming fashion rather than buffering requests/responses. This helps reduce latency and improve application performance.
    - **WebSocket** offers long-standing TCP connections between peers. It is a better approach than HTTP heartbeat mechanism to keep the connection open. Ideal for mobile devices due to low power consumption and can deliver live content like: stock quotes, sport scores, other dynamic data. ALB has a native support for websocket via: ws:// and wss:// protocols
    - **HTTP/2:** major improvement to HTTP 1.1 protocols and offers features like multiplexed request across a single connection to reduce network traffic due to binary nature of the protocol.

When working with an ALB, you should know:

- **Rule**: Determines how to route requests, Each rule specifies a target group, a condition and a priority. An action is taken when the conditions on a rule are matched
- **Target**: An endpoint that be registered as a member of a target group. Valid target is EC2-instance. You can register the same instance multiple times using the different ports.
- **Target group:** a group of EC2-instances running the same service are logically represented as Target group. How to route requests to registered target group is the action for a rule. Each target group specifies a protocol and target port. You can define health checks on a per target group basis. You can route to multiple target groups from each ALB.
    - Idea is to run each of the microservices in its own target group (set of EC2 instances). This helps you to make better decision of scaling up/down the EC2 resources in response to load on individual micro service.

DNS round robin feature as an alternate to ELB

Amazon **Route53** service supports DNS **Round robin** feature and thus not only it converts user friendly domain names (myserver.cloudperf.net) into IP address, it performs load balancing similar to ELB. Some features of Route53 service:

Pros:

- In the case of Route53, client requests are sent directly to ec2 instances without going through ELB, and that reduces latencies for high volume traffic.
- Similar to ELB, Route53 service performs periodic health checks. Thus a failed instance is automatically taken out of the DNS rotation without impacting clients.
- Similar to ELB, Load balancing is supported across AZs (Amazon Availability Zone)
- Unlike ELB, one can specify weights to Route53 Round robin policy...allows traffic to be distributed across instances in proportions that one specifies (40%=instance1, 60%=instance2)

- Route53 also supports a number of other routing policies: Latency Routing, Failover Routing, Geolocation Routing. See URL: http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-policy.html

Cons:

- There is, however, an administrative overhead with using route53 service. It requires all ec2 instances to have public IP addresses, considering clients connect directly to ec2 instances instead of going through ELB. Such tasks can, however, be automated by baking a startup (rc) scripts into AMI or supplying a script via *user-data-script (used at instance launch time)* to self register the instance with the amazon route5*3* service, as we have done in route53 lab.