

# Developer Operations

---

## Load Balancing

# Load Balancing

---

- Most performance problems are related to competition for shared resources (processor, memory, disk, network capacity, etc.)
- Load balancing allows applications to proceed concurrently without all competing for the same resources
- Simple example: distribute application load between server and browser
  - e.g. JavaScript to validate user form input in browser to avoid repeated validation round-trips to server

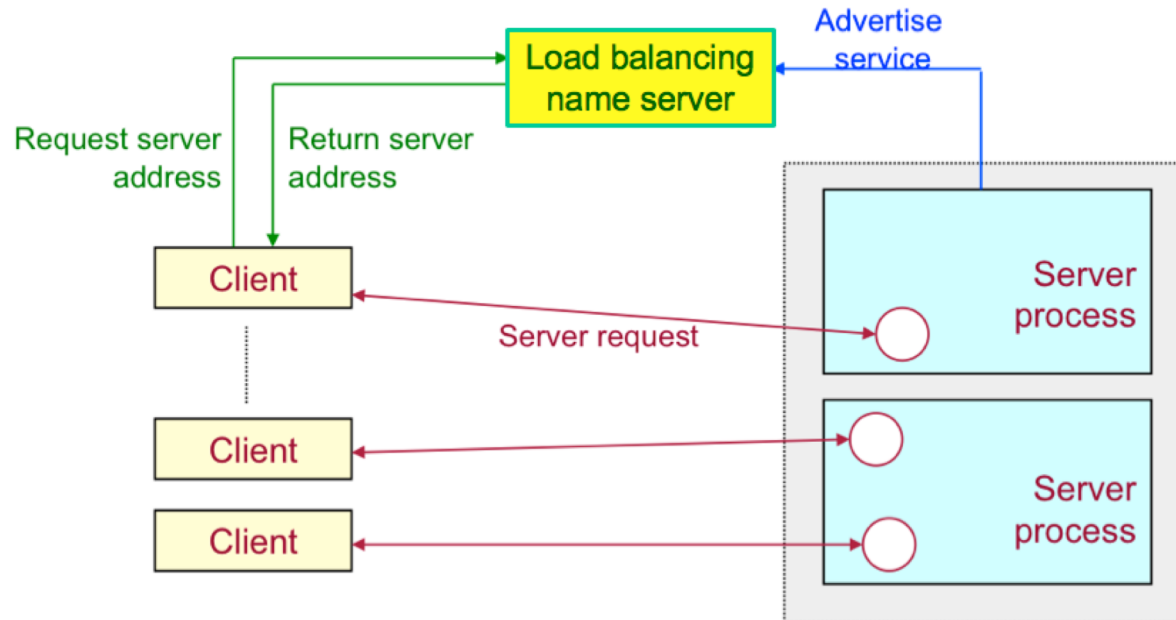
# Server load balancing architecture

---

- Imagine a distributed environment containing:
  - Several clients
  - Multiple replicated servers that can serve client requests
- How are client requests balanced across the servers?
  - => This is the essential task of load balancing

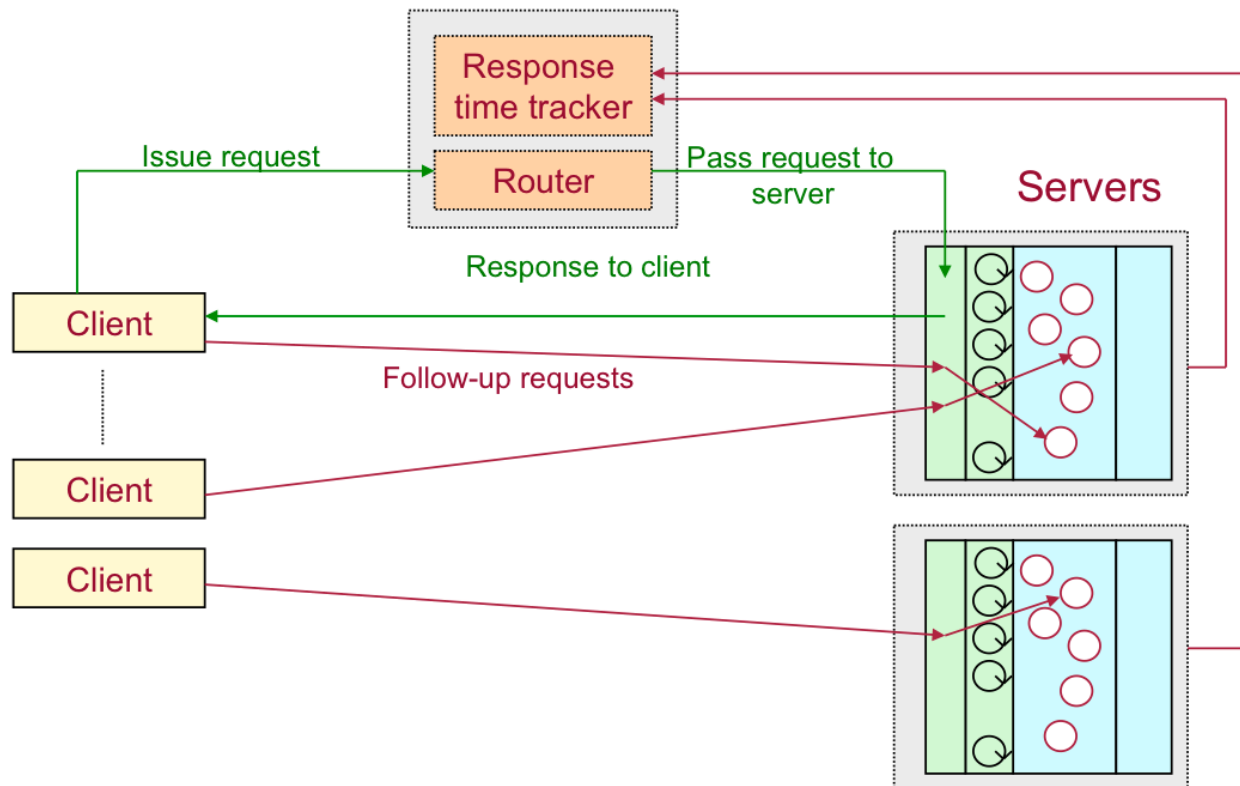
# Server load balancing architecture – example 1

- Client calls on name server to find the location of a suitable server
- Name server can spread client objects across multiple servers
  - Often 'round robin'
- Client is bound to server until it decides to request new address from name server



# Server load balancing architecture – example 2

- Client calls load balancing router when issuing request
- Router passes request to chosen server
- Follow up dialogue directly from client to same server



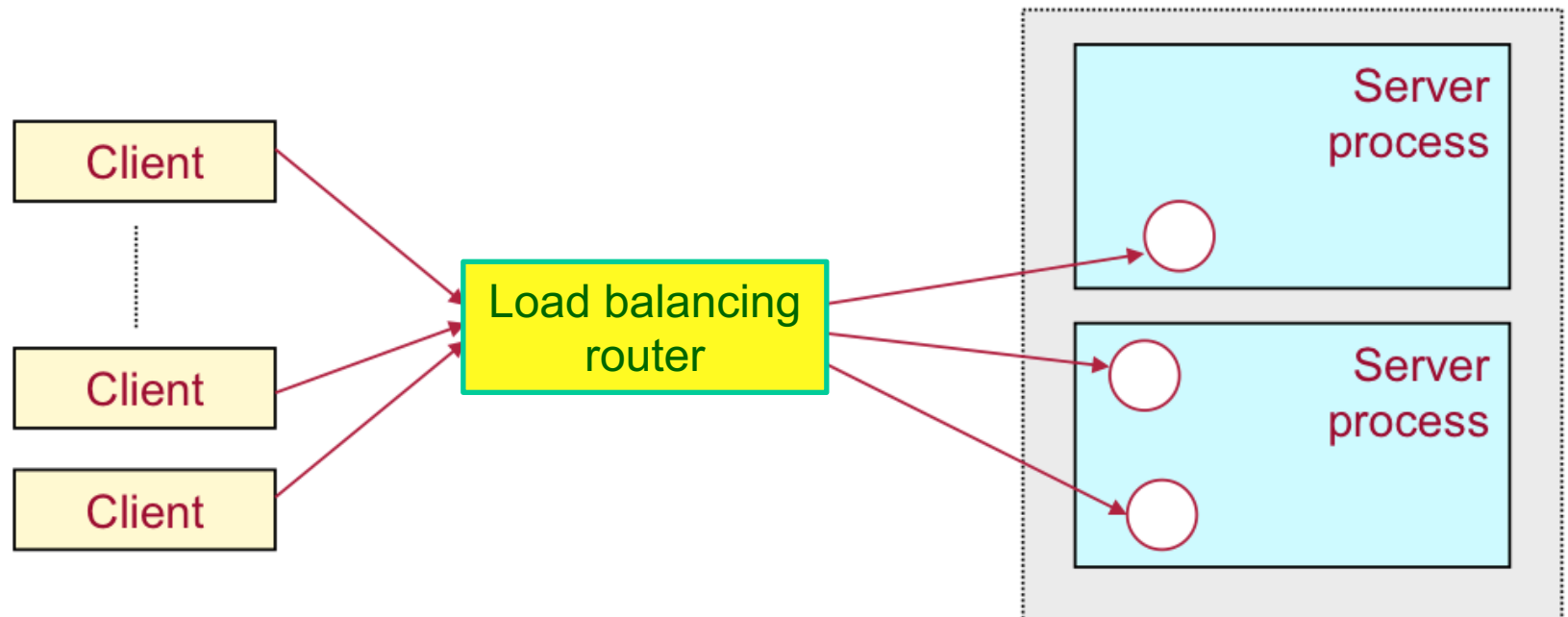
# Server load balancing architecture – example 2

---

- Scales well ...
  - Router only involved when request is issued
  - Follow-up dialogue directly from client to same server
    - Good for handling state
  - Initial server allocation based on response time
    - Client allocated to least-loaded server
- But ...
  - No support for dynamic re-balancing as server load changes

# Server load balancing architecture – example 3

- Some systems involve the router in every browser request
  - Request goes to router who then passes it on to a server process
  - Router can be a bottleneck here
    - Need to consider router scalability and fault tolerance as well!



# Scheduling algorithms – **server** selection

---

- Several approaches, including:
  - Round robin
    - Apply each successive request to each server in turn
  - Highest response time
    - Based on monitoring server performance
  - Lowest load
    - Based on monitoring server resource utilisation
  - Match request size to server performance
  - Match request priority to server performance/availability
  - Combination of the above



# Scheduling algorithms – **request** selection

---

- Several approaches, including:
  - First come first served
  - Priority queueing
    - Maintain a number of request queues with different priorities
    - e.g. a service like Tickemaster might like to complete work-in-progress bookings before admitting new ones
  - Shortest job first
    - Prioritise request with lower resource requirements
    - Risks request *starvation* – long jobs never processed
  - Shortest remaining time
    - Prioritise request where overall session/transaction is closest to completion

# AWS Elastic Load Balancing

---

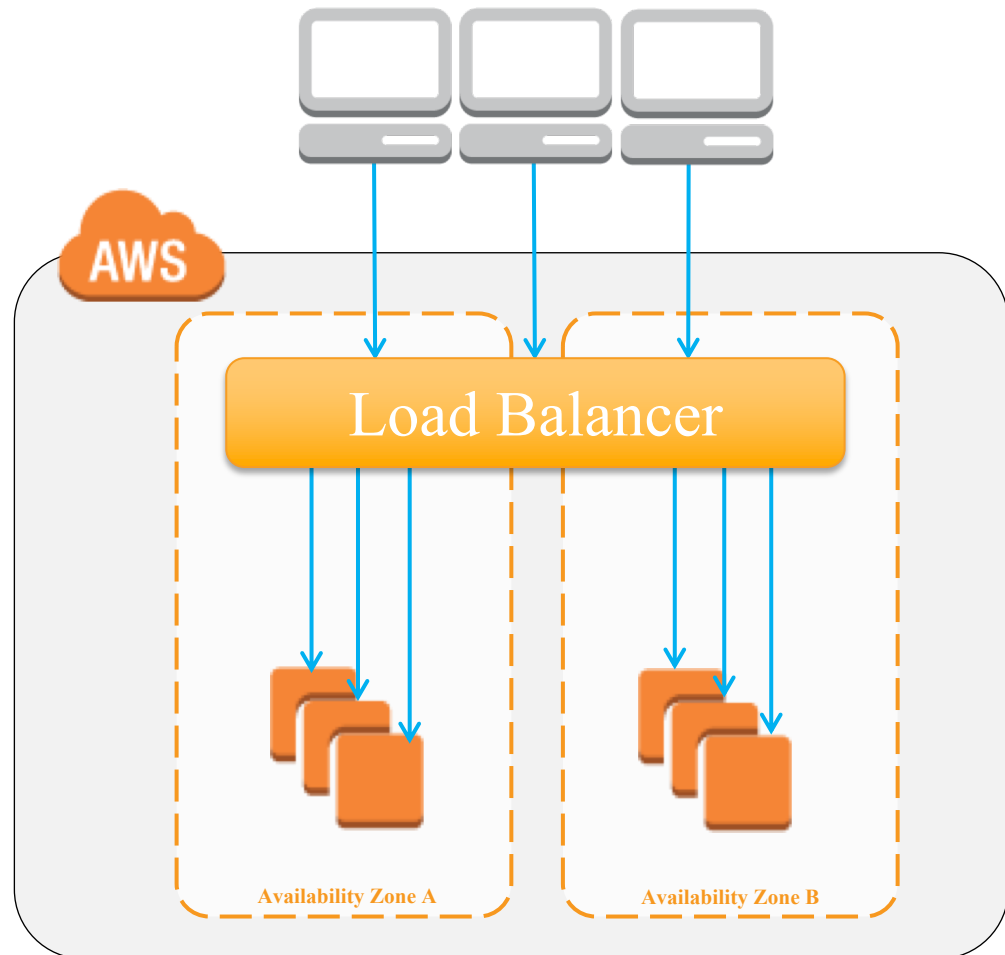


Elastic Load  
Balancing

- Distributes traffic across multiple EC2 instances, in multiple Availability Zones (AZs)
- Supports health checks to detect unhealthy Amazon EC2 instances
  - Traffic no longer routed to instances that are not responding adequately
- Routes and load balances HTTP, HTTPS and TCP traffic to EC2 instances
- Region-specific but not dependent on specific AZ
  - A "highly available" AWS-managed service

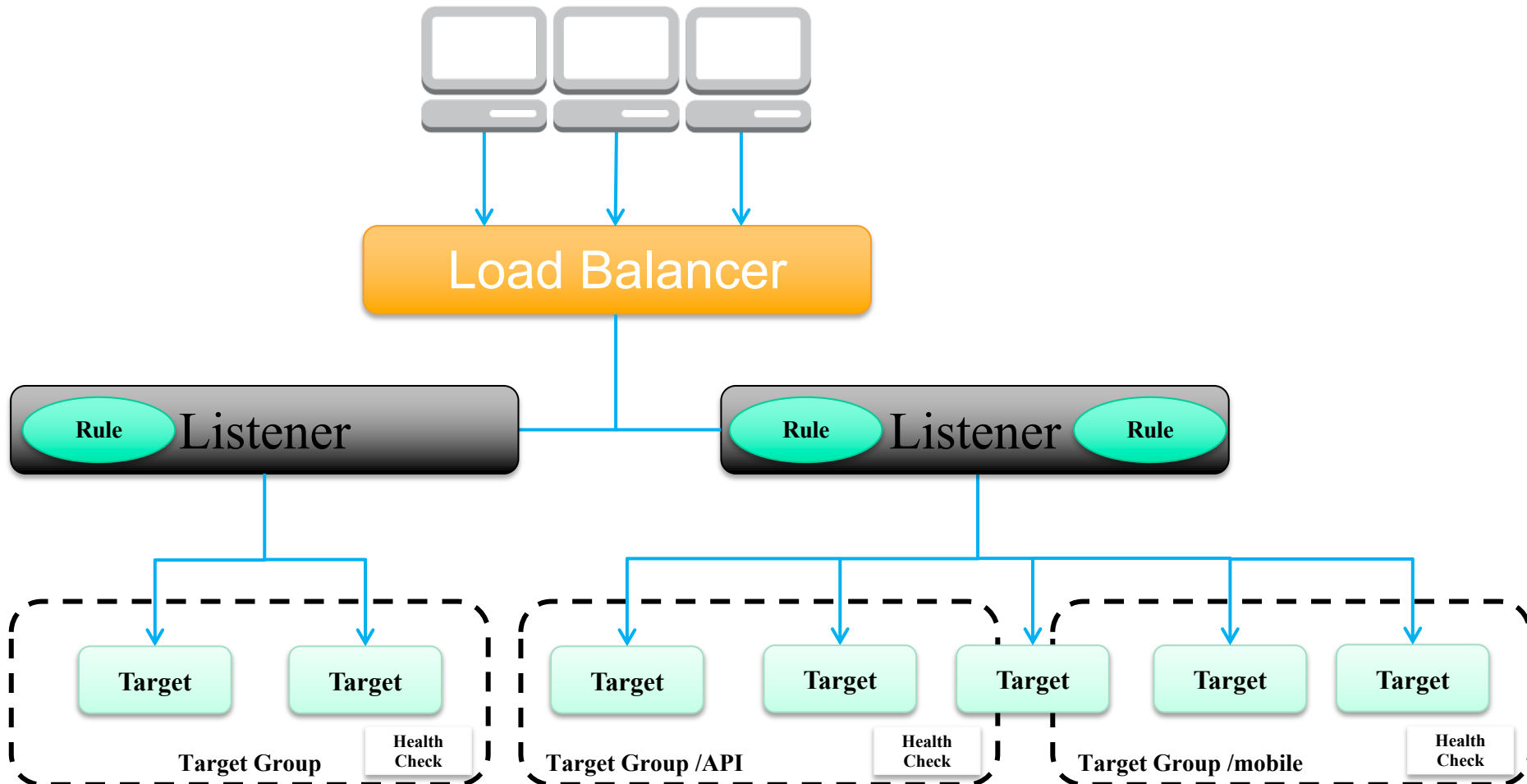
# AWS Classic Load Balancer

Register  
instances with  
your load  
balancer



# AWS Application Load Balancer

Register instances as targets in a **target group**, and route traffic to a target group.



# Next: Create an Application Load Balancer

EC2 Management Console

Secure | <https://eu-west-1.console.aws.amazon.com/ec2/v2/home?region=eu-west-1#V2CreateELBWizard:type=app...>

Services Resource Groups

1. Configure Load Balancer 2. Configure Security Settings 3. Configure Security Groups 4. Configure Routing 5. Register Targets 6. Review

## Step 1: Configure Load Balancer

### Basic Configuration

To configure your load balancer, provide a name, select a scheme, specify one or more listeners, and select a network. The default configuration is an Internet-facing load balancer in the selected network with a listener that receives HTTP traffic on port 80.

**Name** ⓘ My-Test-LB

**Scheme** ⓘ ☒ internet-facing ☐ internal

**IP address type** ⓘ ipv4

### Listeners

A listener is a process that checks for connection requests, using the protocol and port that you configured.

Load Balancer Protocol	Load Balancer Port
HTTP	80

Add listener

### Availability Zones

Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. You can specify only one subnet per Availability Zone. You must specify subnets from at least two Availability Zones to increase the availability of your load balancer.

**VPC** ⓘ vpc-71602a1a (172.31.0.0/16) (default)

Availability Zone	Subnet ID	Subnet IPv4 CIDR	Name
<input checked="" type="checkbox"/> eu-west-1a	subnet-70602a1b	172.31.32.0/20	XXX
<input checked="" type="checkbox"/> eu-west-1b	subnet-76602a1d	172.31.0.0/20	XXX
<input checked="" type="checkbox"/> eu-west-1c	subnet-77602a1c	172.31.16.0/20	XXX