
# DEVELOPING A CUSTOM LOAD BALANCER USING GO & ENVOY

SANDEEP BHAT

**SANDEEP BHAT**

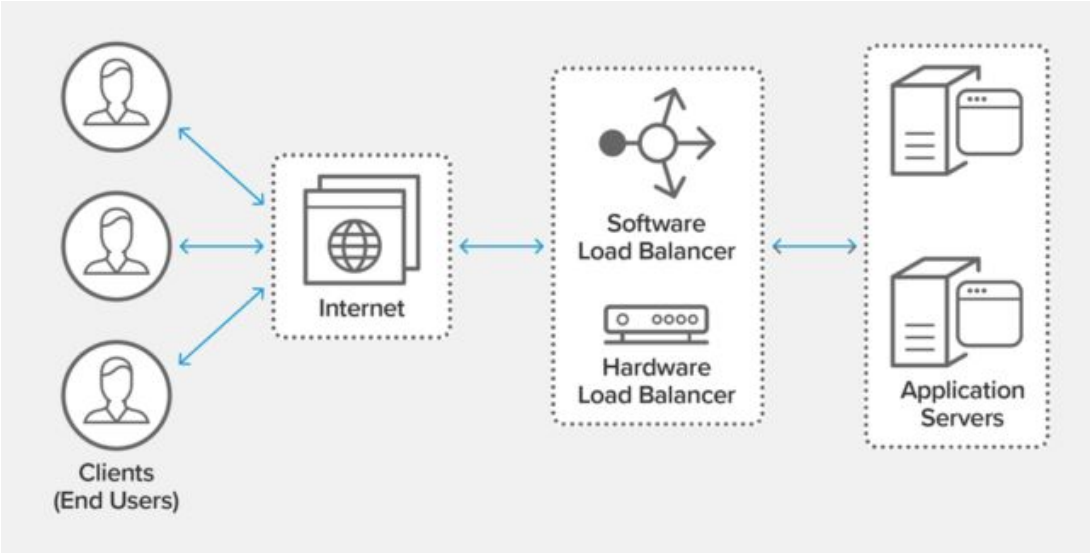
**Staff Software Engineer, Harness**

harness

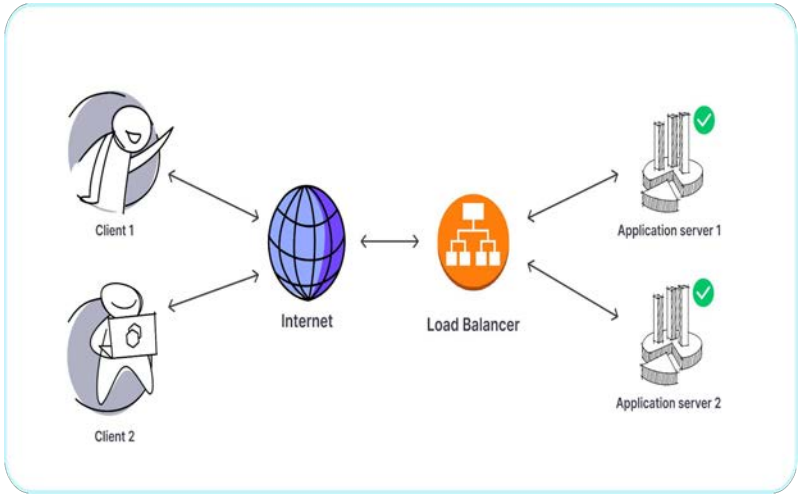
- 8+ years of experience
- Worked at HPE, Cisco and Walmart.
- Working as a Staff Software Engineer at Harness.
- Experience across multiple cloud providers like AWS, GCP and Azure.

# WHAT IS LOAD BALANCING

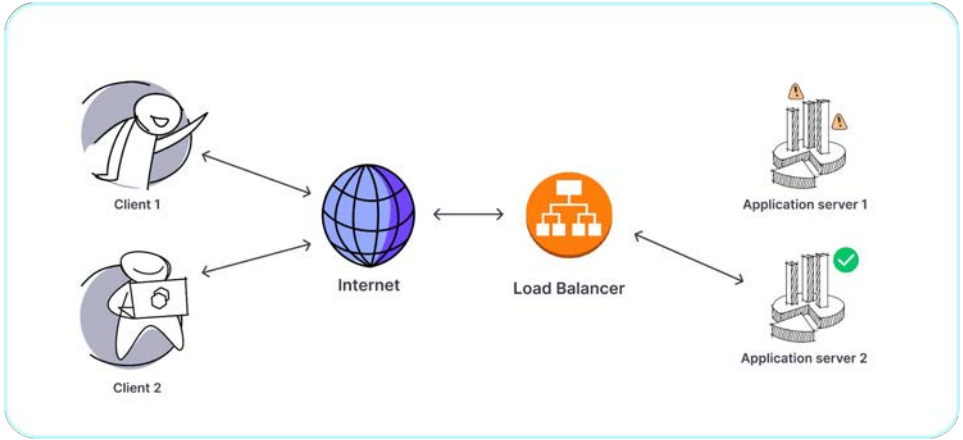
- Distribute incoming network traffic across a group of backend servers
- High availability and reliability
- Flexibility to scale
- Improve performance of application



# What is Load Balancing



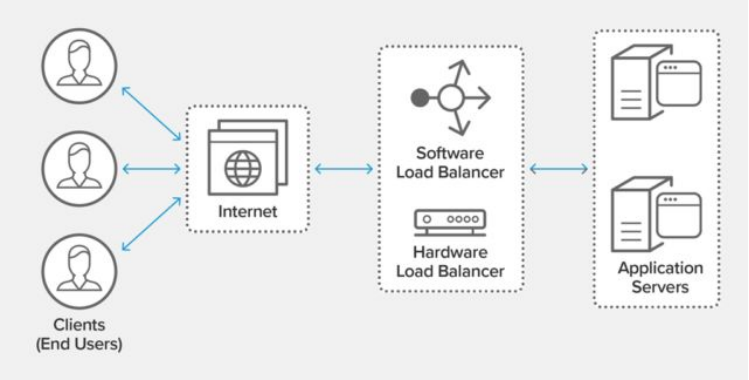

Before




After

# CLOUD NATIVE OPTIONS

- AWS ALB
- AZURE APPGATEWAY
- GCP



# ENVOY

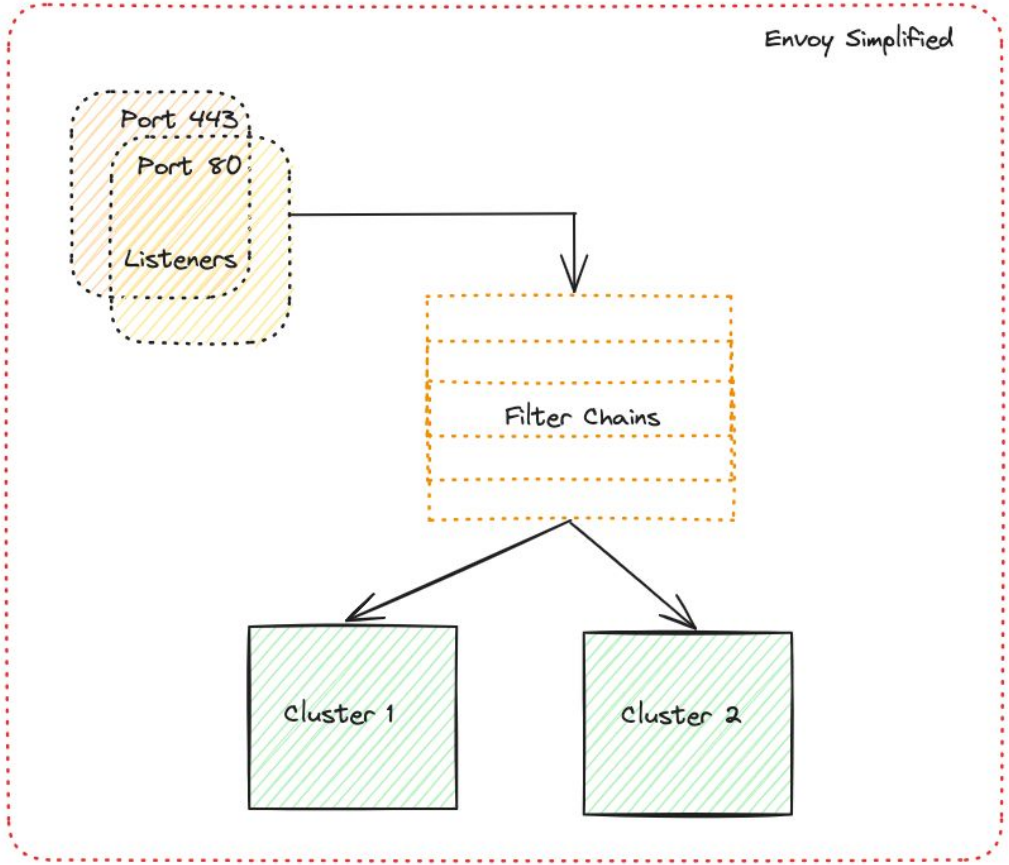
- CNCF graduated project
- Evolved out of Lyft
- Written using C++
- Filter based mechanism
- 19k commits
- 22k Stars

# Features of Envoy

- Service discovery
- Load balancing
- Health checking
- Security
- Observability
- Rate limiting
- Extensible

# Key Components of Envoy

- Listener
- Filters
- Clusters
- Secrets
- Upstream



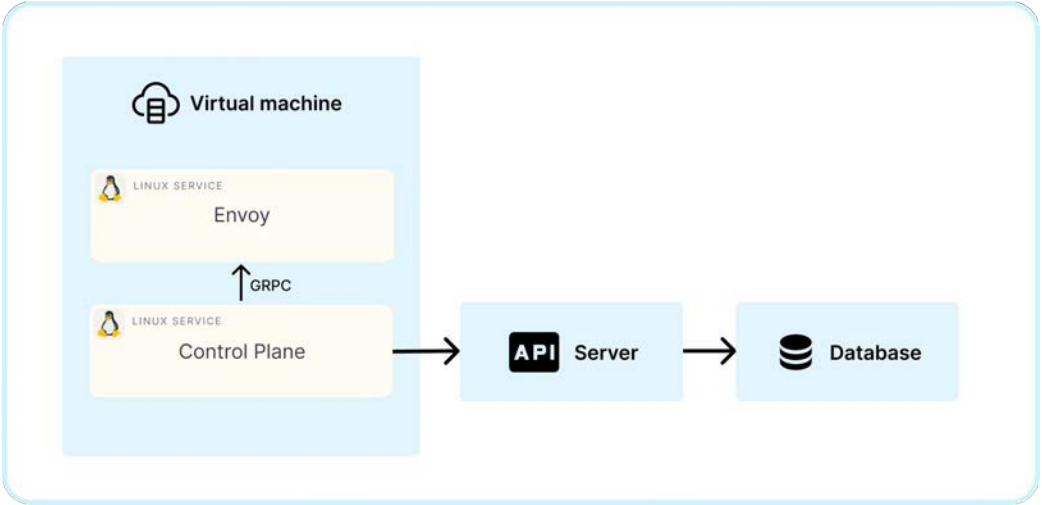
```yaml
static_resources:
  listeners:
  - name: listener_0
    address:
      socket_address: { address: 127.0.0.1, port_value: 80 }
    filter_chains:
    - filters:
      - name: envoy.filters.network.http_connection_manager
        typed_config:
          "@type": type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionManager
          stat_prefix: ingress_http
          codec_type: AUTO
          route_config:
            name: local_route
            virtual_hosts:
            - name: local_service
              domains: ["*"]
              routes:
              - match: { prefix: "/" }
                route: { cluster: some_service }
          http_filters:
          - name: envoy.filters.http.router
  clusters:
  - name: some_service
    connect_timeout: 0.25s
    type: STATIC
    lb_policy: ROUND_ROBIN
    health_checks:
    - timeout: 2s
      interval: 5s
      unhealthy_threshold: 2
      healthy_threshold: 2
      http_health_check:
        path: "/"

    load_assignment:
      cluster_name: some_service
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: 35.238.9.13
                port_value: 80
```

# Requirements of Custom Load Balancer

- Distribute traffic among multiple backend targets
- Support multiple domains
- Health checking
- Cloud agnostic
- Scalability customization

# Components of Envoy based Load Balancer

- Envoy
- Control Plane
- API Server
- Cloud Init
- Server



# Envoy - Service

```
[Unit]
Description=Service to run Envoy in a Linux machine
[Install]
WantedBy=multi-user.target
After=network.target
[Service]
Type=simple
ExecStart=/usr/bin/envoy -c /var/custom_lb/envoy_startup_config.yaml
WorkingDirectory=/var/custom_lb/
Restart=always
RestartSec=5
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=%n
```

```yaml
node:
  cluster: test-cluster
  id: envoy-manager
dynamic_resources:
  lds_config:
    resource_api_version: V3
    api_config_source:
      api_type: GRPC
      transport_api_version: V3
      grpc_services:
        - envoy_grpc:
            cluster_name: xds_cluster
  cds_config:
    resource_api_version: V3
    api_config_source:
      api_type: GRPC
      transport_api_version: V3
      grpc_services:
        - envoy_grpc:
            cluster_name: xds_cluster
static_resources:
  clusters:
  - connect_timeout: 1s
    http2_protocol_options: {}
    name: xds_cluster
    type: STATIC
    load_assignment:
      cluster_name: xds_cluster
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: 127.0.0.1
                port_value: 18000
```

# Snippet - Controller

```
func startController() {

	cache := cache.NewSnapshotCache(false, cache.IDHash{}, logrus.WithFields(logrus.Fields{}))

	go runEnvoyServer(cache)

	go syncServer(time.Duration(SyncInterval()*int(time.Second)), cache)

	c := make(chan os.Signal, 1)
	signal.Notify(c, os.Interrupt, syscall.SIGTERM)
	<-c
}
```

```go
// RunServer starts an xDS server at the given port.
func RunServer(ctx context.Context, srv server.Server, port uint) {
	var grpcOptions []grpc.ServerOption
	grpcOptions = append(grpcOptions, grpc.MaxConcurrentStreams(grpcMaxConcurrentStreams),
grpc.KeepaliveEnforcementPolicy(keepalive.EnforcementPolicy{MinTime: 10 * time.Second, PermitWithoutStream: true}))
	grpcServer := grpc.NewServer(grpcOptions...)

	lis, err := net.Listen("tcp", fmt.Sprintf(":%d", port))
	if err != nil {
		log.Fatal(err)
	}

	registerServer(grpcServer, srv)

	log.Printf("management server listening on %d\n", port)
	if err = grpcServer.Serve(lis); err != nil {
		log.Println(err)
	}
}
```

# Snippet - GRPC Server

```
func syncServer(interval time.Duration, cache cache.SnapshotCache) {
    timer := time.NewTicker(interval)
    for { // nolint
        select {
        case <-timer.C:
            // Perform sync operation by fetching configuration
from DB
            _ = syncConfiguration(cache)
        }
    }
}
```

# Snippet - Sync Configuration

```
{
    "host": "sandeepbhat.co.in",
    "targets": [
        "X.X.X.X",
        "X.Y.X.Y"
    ],
    "incoming_port": 80,
    "outgoing_port": 80,
    "Health_checks": [
    {
      "path": "/",
      "port": 80,
      "timeout": 5,
      "interval": 10
    }]
}
```

Snippet - Database

# Packaging - Cloud Init

- Initialization system
- Package installation
- Custom Scripts
- SSH Key Setup
- Cloud-Independent
- Customization and Extension

# Sample - Cloud Init

```
Content-Type: multipart/mixed; boundary="//"
MIME-Version: 1.0

--//
Content-Type: text/cloud-config; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="cloud-config.txt"

#cloud-config
cloud_final_modules:
 - [scripts-user, always]
--//
Content-Type: text/x-shellscript; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="userdata.txt"

#!/bin/bash
# Script goes here
--//--
```

# Cost Comparison

## AWS ALB

- Hourly pricing per ALB
- Cost per Load Balancer Capacity Units (LCU) per hour
    - New Connections / sec
    - Bytes Processed
    - Active Connections
    - Rules processed
- Example - Ohio (US East)
    - ALB - $0.0225/hr
    - Total - 0.0225 * 24 * 30 ~ $16.2/month

## CUSTOM LB

- Hourly pricing per Instance
- Additional cost of traffic
- Flexibility of instance types
- Example - Ohio (US East)
    - t2.micro - $0.0116/hr
    - Total - 0.0116 * 24 * 30 ~ $8.3/month

# Demo