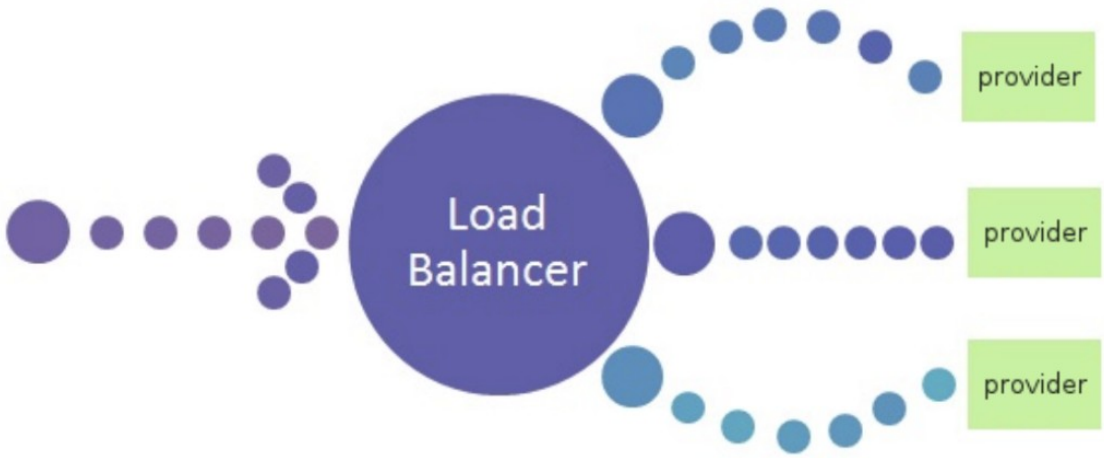
LOAD BALANCING

A **load balancer** is a component that, once invoked, it distributes incoming requests to a list of registered providers and return the value obtained from **one** of the registered **providers** to the original caller. For simplicity we will consider both the load balancer and the provider having a public method named get()



PLEASE NOTE: Every component described in the exercise is a piece of software of the same codebase. You don't need to build a "runnable" application, there is no need to create any real server or rest service, no need to build any real network-based interaction, there should be no framework within the codebase. Simulating real world scenario means however that it has to be working properly and effectively in all scenario that can happen in real life (eg. handling parallel requests, managing edge cases etc.)

iptiQ

Step 1 – Generate provider

Generate a Provider that, once invoked on his get() method, retrieve an unique identifier (string) of the provider **instance**

Step 2 – Register a list of providers

Register a list of provider instances to the Load Balancer - the maximum number of providers accepted from the load balancer is 10

Step 3 – Random invocation

Develop an algorithm that, when invoking multiple times the Load Balancer on its get() method, should cause the random invocation of the get() method of any registered provider instance.

Step 4 – Round Robin invocation

Develop an algorithm that, when invoking multiple times the Load Balancer on its get() method, should cause the round-robin (sequential) invocation of the get() method of the registered providers.

Step 5 – Manual node exclusion / inclusion

Develop the possibility to exclude / include a specific provider into the balancer

Step 6 – Heart beat checker

The load balancer should invoke every X seconds each of its registered providers on a special method called check() to discover if they are alive – if not, it should exclude the provider node from load balancing.

Step 7 – Improving Heart beat checker

If a node has been previously excluded from the balancing it should be re-included if it has successfully been “heartbeat checked” for 2 consecutive times

Step 8 – Cluster Capacity Limit

Assuming that each provider can handle a maximum number of *Y* parallel requests, the Balancer should not accept any further request when it has (*Y** aliveproviders) incoming requests running simultaneously