

How to Choose the Right Algorithm for Your Vultr Load Balancer

Learn how to select the optimal algorithm for your Vultr Load Balancer based on your application needs, traffic patterns, and performance requirements.

Contents

01	Introduction	3
02	1. The Round-Robin Algorithm	3
03	2. The Least Connections Algorithm	4
04	Conclusion	5

Introduction

When deploying the fully-managed Vultr load balancer in your application, there are different algorithms that you can implement. These are techniques for intelligently distributing traffic across multiple servers to provide horizontal scaling, security, and continuous availability.

The two available options include the **round-robin** and the **least connections** algorithms. In a round-robin setup, the Vultr load balancer sends requests to all servers in a rotational manner without considering the current load of each server. However, the least connections algorithm is more intelligent and only routes clients' requests to a server with the least number of active connections.

Both algorithms are useful and come with different pros and cons. In this tutorial, you'll learn how to choose the right option to ease the implementation process.

1. The Round-Robin Algorithm

The round-robin algorithm is the most widely used because it is easy to understand and implement. For instance, if you have a set of **3** backend servers in your cluster and you send **5** requests, the algorithm will select the servers in turns as illustrated below:

```
client request 1: Server 1
client request 2: Server 2
client request 3: Server 3
client request 4: Server 1
client request 5: Server 2
...
```

The above approach is quite simple. However, it won't perform optimally for a scenario where you've deployed servers with different specifications (for example, RAM and CPU). The method puts the same load on the different backend servers without considering their resources.

However, the round-robin algorithm is suitable in the following scenarios:

1. In a load balancing architecture where all your backend servers have the same capabilities.
2. In conducting a basic test to ensure that you've correctly configured a load balancer. For instance, if you have 10 backend servers and you throw in 10 requests, the load balancer should distribute traffic in a cyclic manner to each server in order for you to conclude that the load balancer is working.
3. Where you need an easily-implementable load balancing solution for horizontal scaling.

2. The Least Connections Algorithm

In real-life applications, the actual period of time that clients spend on the backend server varies depending on the complexity of the tasks they're executing. If the load balancing algorithm doesn't consider the active connections, some servers may be overloaded with processes that might take a very long time to process.

The least connections algorithm is a better fit in such situations. When a new request comes in, the method routes it to the server which is not serving any requests, and then other subsequent requests are forwarded to the servers with the least active connections in ascending order. For instance, assume you've 3 backend servers handling the following connections.

```
Server 1 - 15 active connections  
Server 2 - 0 active connections  
Server 3 - 10 connections
```

If you send 3 more requests to the servers, the least connections method will select the servers in the following order.

```
client request 1: Server 2  
client request 2: Server 3  
client request 3: Server 1
```

The least connections method is suitable in the following situations:

1. When you want the best performance for your servers. Remember, this method considers time-based session entries made between the clients and the backend service before making a routing decision to the most optimal server.
2. In case you need fairness and efficiency. The least connections method queues requests only to the backend server that guarantees minimum execution time. There are no chances of clients overcrowding in a single server.
3. Where you want all the backend servers to work at 100% capacity. The least connections method takes the clients' request processing time into consideration before choosing the best path. This eliminates any chances of idle servers during peak periods.

Conclusion

In this guide, you've seen the pros and cons of Vultr's load balancing algorithms. With the above tips in mind, you can choose the best method that works for your use case. The round-robin algorithm is simple to implement, but the least connections method is always better when you want to put equal loads on your backend services.

For more information about Vultr's load balancer, visit the following resources:

- [Vultr Load Balancer Quickstart Guide](#)
- [How to Configure a Vultr Load Balancer with Private Networking](#)
- [Use a Wildcard Let's Encrypt Certificate with Vultr Load Balancer](#)



VULTR

