How To Configure DNS Round-Robin Load-Balancing For High-Availability

Posted February 20, 2014 248k DNS LOAD BALANCING HIGH AVAILABILITY

Introduction

Albeit one of the more controversial techniques, a good method to geographically distribute your application by taking advantage of your provider's global presence is to use and manage DNS responses (i.e. list of IP addresses returned). Unless you are willing to spend a little fortune on hardware and infrastructure costs, working with DNS to achieve high-availability is probably an excellent way to go.

In this article, we will see how to exploit some of the truly excellent and unique possibilities offered by DigitalOcean's global cloud server / data-centre infrastructure to have a geographically-distributed, highly-available application set-up for minimal downtime (and thus dataloss) by managing DNS responses.

Glossary

1. Traditional Application Deployment Structure

2. High-Availability

1. Highly-Available Application Deployment Structure
2. How To Achieve High-Availability Using DNS
3. Summary

3. How To Deploy Highly-Available Applications

1. Setting Up Load-Balancers / Reverse-Proxy
2. Setting Up DNS Records
3. Setting Up Application Servers
4. Setting Up Databases

Traditional Application Deployment Structure

Traditional and most common application deployments depend on setups with all related components being located at the same place due to several reasons, such as:

- Providers' lack of means;
- High costs, and/or;
- Complicated engineering work.

Even if an application is served from multiple machines sitting behind load-balancers (or reverse-proxies), and even if the database is also set up in a way to offer reliability and prevent data-loss, these kind of arrangements are prone to different levels of errors, causing you, at times, downtime.

In order to prevent this, one must rely on and use a more dependable system architecture. One whereby data and servers are globally distributed at different areas (e.g. San Francisco and New York).

High-Availability

If your application is your business, you need to keep it accessible 24/7, if possible, almost without any interruption. Unfortunately, scaling horizontally over many servers at one location is not always the solution because of unexpected data-centre problems.

Globally-distributing your virtual servers across different geographical centres, however, can provide you the stability you require -- thus, keeping applications' up-time level as high as possible.

In terms of IT system-design, this kind of structure is referred to as *high-availability*.

Thanks to DigitalOcean's presence in two continents, at five different locations, you can also spread your application stack globally.

You can connect a Floating IP, which is a publicly-accessible static IP address that can be mapped to one of your Droplets, to your redundant infrastructure and launch your site or service with a single public IP. This Floating IP can be instantly remapped to a new droplet, to allow for flexibility and responsiveness in your infrastructure. Read more about this new feature here.

Highly-Available Application Deployment Structure

Simply put, highly available application deployment structure, as we have just covered, depends on delivery and response to clients from different datacentres.

Although there are a number of possible ways to obtain this kind of structure, probably the simplest and the most affordable one is to take advantage of how DNS works.

A basic example set up can be considered as the following:

```
                     ________________
                    |                |
                    |     CLIENT     |
                    |  WEB BROWSER   |
                    |________________|
                            ||
                            ||
                     _______\/_______
                    |                |
                    |   DNS SERVER   |
                    |________________|
                            ||
                            ||
                    _______/  \_______
                   /                  \
                  /                    \
 ____________________                ____________________
|                    |              |                    |
|   SAN FRANCISCO    |              |   SAN FRANCISCO    |
|____________________|              |____________________|
|   ______________   |              |   ______________   |
|  |              |  |              |  |              |  |
|  |  WEB SERVER  |  |              |  |  WEB SERVER  |  |
|  | LOAD BALANCE |  |              |  | LOAD BALANCE |  |
|  |     PROXY    |  |              |  |     PROXY    |  |
|  |_____    _____|  |              |  |_____    _____|  |
|________|  |________|              |________|  |________|
          ||      __                  __      ||
          ||<=====||==================||=====>||
          \/      \/                  \/      \/
 ____________________                ____________________
|                    |              |                    |
|    SAN FRANCISCO   |              |       NEW YORK     |
|____________________|              |____________________|
|   ______________   |              |   ______________   |
|  |              |  |              |  |              |  |
|  |  APP SERVER  |  |              |  |  APP SERVER  |  |
|  |______  ______|  |              |  |______  ______|  |
|   ______||______   |              |   ______||______   |
|  |              |  |              |  |              |  |
|  |   DATABASE   |<==================>|   DATABASE   |  |
|  |______________|  |              |  |______________|  |
|____________________|              |____________________|
```

How To Achieve High-Availability Using DNS

When a user types the domain name of a website, through a set of defined rules (i.e. a protocol), the web-browser dials name-servers and asks them the address of the machines hosting the said web-site. Once it receives the IP address, then it sends the request to that computer, along with some additional data, and renders the response.

Since DNS allow multiple records to be kept (even the same kind), it becomes possible to list multiple hosts as the server.

Therefore, as demonstrated on the above schema, if you list the IP address of 2 load-balancers / reverse-proxies, located at two different locations, each set to balance the load between application servers, again located at at least two different data-centres, if one of the data-centres becomes unreachable, the client's web-browser will try the next IP address records returned by the DNS server and repeat the process to get the web-site.

This kind of load-balancing is called Round Robin DNS load-balancing.

Summary

Things might look a little bit complex at a first glance. Let's summarise them using step-by-step instructions:

1. DNS can hold multiple records for the same domain name.
2. DNS can return the list of IP addresses for the same domain name.
3. When a web-browser requests a web-site, it will try these IP addresses one-by-one, until it gets a response.
4. These IP addresses should point at *not* application servers but at load-balancers / reverse-proxies.
5. These reverse-proxies need to be balancing the load between multiple servers at multiple locations.
6. If a data-centre is down and the web-browser is unable to get a response from an IP address (i.e. a load-balancer), it will try to reach the address of the other.
7. Since it is very unlikely for both data-centres to be unreachable at the same time, the second load-balancer will return a response.
8. Web-application servers should be stateless to make the job of load-balancers easier.
9. Database servers should be set up in a replicated way.

How To Deploy Highly-Available Applications

Note: This tutorial is programming language or web-server type agnostic. Following these instructions, you can achieve high-availability regardless of your choice of frameworks, web or HTTP servers.

Setting Up Load-Balancers / Reverse-Proxy

The first step to high-availability is to set up two or more load-balancing reverse proxies which are going to communicate between your application servers.

1. **Instantiate two cloud servers at two locations:**

Create two DigitalOcean droplets.

e.g. Article: How To Create A DO Cloud Server

1. **Set up a load-balancer / reverse-proxy on each droplet:**

Install and configure Nginx, Apache or HAProxy.

e.g. Article: Nginx as a Front End Proxy, HAProxy Load-balancing on Ubuntu

1. **Get the IP Addresses of your load balancers:**

Type `/sbin/ifconfig` and find out your droplets' IP addresses.

e.g. `inet addr:107.170.40.112`

Setting Up DNS Records

DNS A Records translate domain names (e.g. `www.digitalocean.com`) to machine-reachable IP addresses.

Once you are done configuring two droplets with a load-balancing reverse-proxy on each, the next step consists of adding 2 A records through DigitalOcean's DNS service to point your domain name to the IP address.

1. **Log-in to your DigitalOcean control panel:**

Click `DNS` on the left-hand menu and add a new domain name pointing to a load-balancer droplet from the previous step.

1. **Add a new A Records:**

Once you are on the next step, click "Add Record" on the upper-hand side and create a new A record, with the IP address of the other load-balancer droplet.

Setting Up Application Servers

Next step is setting up the application servers.

For global distribution to work, just like the first load-balancing servers you have created, you need two new droplets to host your application servers.

Note: You can also run each of your application servers on the same machine as the load-balancers; however, this would not be recommended.

Deploy or duplicate your application server droplet on two locations. For example:

- In **NY 1** and **NY 2**;
- In **AMS 1** and **AMS 2**;
- In **SF 1** and **NY 2** etc.

Go back to the first step and following the load-balancer setting up articles, configure them to proxy incoming connections to these two application-serving droplets.

Setting Up Databases

It is hard to imagine a web-application without a database. The hardest part of distributing applications over multiple servers is probably dealing with databases.

epending on your choice of database server, create a duplicated configuration but across multiple ocations.

See:

- **For MySQL Master/Slave Replication:**

How To Set Up Master Slave Replication in MySQL

- **For MySQL Master/Master Replication:**

How To Set Up MySQL Master-Master Replication

- **For PostgreSQL Master/Slave Replication:**

How To Set Up Master Slave Replication on PostgreSQL

Once you complete creating a replicated database structure, point your applications to use their addresses, as interacted in tutorials as the DB server.

Submitted by: O.S. Tezer

Upvote (26) Subscribe Share

Related Tutorials

Configuration Drift: Phoenix Server vs Snowflake Server Comic

Solution Deep Dive: Building a Highly Available Web Application with Web Processing and Stor...

An Introduction to Load Balancers Comic

How to Use ProxySQL as a Load Balancer for MySQL on Ubuntu 16.04

How To Create a High Availability Setup with Heartbeat and Floating IPs on Ubuntu 16.04

16 Comments

Leave a comment...

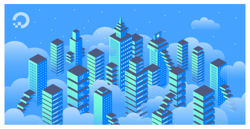
Log In to Comment

1
support1 *March 14, 2014*

Thank you for this tutorial.
Would you be kind enough to also add the setup for replication of files please?
many thanks

1
kamaln7 MOD *March 15, 2014*

@Vincent: Check out https://www.digitalocean.com/community/articles/how-to-create-a-redundant-storage-pool-using-glusterfs-on-ubuntu-servers.



How To Create a Redundant Storage Pool Using GlusterFS ...
by Justin Ellingwood

GlusterFS is a technology that allows you to create pools of storage that are accessible from the network. Using this software, in this article we will discuss how to create redundant

0
nospam *April 1, 2014*

Good article, but I am confused by the diagram.

1) I need a load balancer droplet for each of my application droplets?

2) Is the diagram using a master to master database?

I want to load balance my application, and my application should write to the master db and read from a load balanced slave. Can you point me in the right direction for that ?

I must be wrong because it looks to me like I need the following 9 servers to make it happen.

Server 1: Application load balancer 1
Server 2: Application load balancer 2
Server 3: Application
Server 4: Application
Server 5: MySQL master to handle writes from Server 3 and 4
Server 6: MySQL Slave Load Balancer 1
Server 7: MySQL Slave Load Balancer 2
Server 8: MySQL Slave
Server 9: MySQL Slave

0
ahmedtawfikis *January 28, 2015*

Only four droplets are needed: Two load balancers and two application/database servers. Each application server will have a database as well. This is a minimal effort configuration. If you want to expand to the setup you mentioned, it's up to you, depending on the needs of your application.

0
CD2Team *December 8, 2016*

You can have as many app servers per DC as you need, but everything has a minimum of two.

I'm pretty sure it's

DNS:

- A record and AAA record for DC1
- A record and AAAA record for DC2

DC1:

- Load balancer[s] (1 minimum)
- Shared-state server[s] (1 minimum)
- Application server[s] (I'd say 2 minimum)
- DB server[s] (1 minimum as master with master-master or master-slave replication and master promotion upon fail-over)

DC2:

- Load balancer[s] (1 minimum)
- Shared-state server[s] (1 minimum)
- Application server[s] (I'd say 2 minimum)
- DB server[s] (1 minimum as master with master-master or master-slave replication and master promotion upon fail-over)

There are services that offer to take the complexity of this away and for anyone without time and resources to dedicate I'd advise sticking with those.

This is 10 servers, but I've added 2 for shared state off the DB so yours is not totally off ($80-$100 min cost)

One other thing not mentioned here is the benefit of partitioning your servers like this for performance. As long as you don't keep visitors on one PC with client pinning you'll have more cores and ram per-machine to think.

If you use glusterFS I'd also advise you to have a minimum of 2CPU cores (Ideally 4, which brings cost to $360/pcm @ *84GB optionally + 22*GB). You'd get a lot of mileage out of that setup as well. Probably at least your first year for a high-growth private app, maybe less for the next big thing, but costs of running one of those is huge!

0

mallikarjuna.hello *April 5, 2014*

Probably we need when users grow exponentially

3

mateusz *January 16, 2015*

One sentence in this article is a huge lie and makes proposed solution unusable:

"When a web-browser requests a web-site, it will try these IP addresses one-by-one, until it gets a response."

No, it doesn't! Browser selects the random IP address and sticks to it no matter if the host is up or not! DNS round-robin is not a failover solution! What you say is only true for MX records and mail servers.

If you want to have a spare load-balancer, you need to implement a solution that will make it capture original IP address of failed one. I guess there is no way to use DO droplets for this. So - if any of your load balancers will be down - half of your visitors will see error message.

To proof my words:
I setup a dns entry test1.pr1v.net which has 3 records:
``test1.pr1v.net has address 1.2.3.4
test1.pr1v.net has address 1.2.3.5
test1.pr1v.net has address 78.47.31.86

~~~
Two of them (1.2.3.*) are unused addresses, and the third one is my host IP. Please see if

```$ curl -v http://test1.pr1v.net
* Rebuilt URL to: http://test1.pr1v.net/
* Hostname was NOT found in DNS cache
*   Trying 1.2.3.5...
~~~

ahmedtawfikis *January 28, 2015*

1

well, I tried my own droplets, and it took *curl* some time until the timeout of the turned off droplet was reached and then it switched to the second droplet's IP and obtained the response. I was following this tutorial literally and used the same setup mentioned.

---

faizullas3e *January 29, 2015*

0

@mateusz : Same is happening with me . DNS load Balancing does not work on the digital ocean side. Randomly it sticks to one IP and keep on calling the same, no matter what the situation is(doesn't even change for failover) . :(

---

kamaln7 MOD *January 29, 2015*

0

What's your domain name? The DNS nameservers serve all A records at once and one of them is selected by the software that you're using or your local DNS resolver if you're running one.

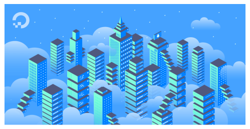
---

faizullas3e *January 30, 2015*

0

I have configured the DNS records according to the DNS load balancing section of this https://www.digitalocean.com/community/tutorials/dns-tips-and-tricks article . I have two droplets and both of then has nginx working as a load balancer for backend servers. I have added A records for my domain which points to the IP of these droplets.

Thanks



DNS Tips and Tricks

by Etel Sverdlov

This tutorial covers several tips that help a user when setting up DNS. It covers confirming if your DNS records are working with the "whois" and "dig" commands, setting

---

stephenchengonl *July 5, 2015*

1

This looks a very conventional solution. Whether it works or not purely depends on if digitalocean's DNS system does failover or not. can you confirm that?

Without this ability, DO's dns can have still have chance to send back dead reverse proxy's ip back to user, hence resulting in a no-responding/freezing web site or app.

Please, can someone from DO confirm that?

---

paulprijs *July 22, 2015*

0

If i don not need load balancing (because each of the two servers can easily handle all my traffic) but i want to have a fall-over-server in case one of the servers is unreachable; can i then skip the load balancer proxies and just put the application servers in the dns?

---

jamesclouser *October 18, 2015*

0

I too would like to know if DigitalOcean DNS zoning provides health checks for server endpoints.

Currently, I'm using Amazon Route 53 for this purpose...

CD2Team *December 8, 2016*

0

Essentially this misses shared state servers (most DB's are too slow for this)

---

vij *March 14, 2017*

With latest release Nginx support UDP/DNS load balancing http://www.techietown.info/2017/03/dns-loadbalancing-using-nginx/