<u>Steps to Create Dynamic Server Allocation as Fail Safe Mechanism</u>

<u>To Create Your First DigitalOcean Droplet Virtual Server</u>

DigitalOcean calls its virtual private servers Droplets; each Droplet that you spin up is a new VPS for your personal use.

Step One — Log In

To create your first Droplet go to the DigitalOcean Control Panel and log in with your email and password. The create button will be right there on the first page: click on "Create Droplet":

Step Two — Select Droplet Image

You can create your Droplet image from 4 possible categories.

Step Three — Select Your Droplet's Size

Depending on your needs and budget, you can select the Droplet option that works best for you.

Step Four — Select Your Droplet Region

You may choose the most effective region for your Droplet location. Although equally powerful, the best region to choose is the one nearest to you and your customers or other possible users. Selecting a more distant server location may increase your server latency without serving any practical purpose.

Step Five — Select Additional Options

The Select additional options section allows you to select which features you would like your

Step Six — Select SSH Keys (Optional)

Optional: Select which SSH keys you would like to add to your new Droplet.
It is recommended that you set up SSH keys to authenticate to your Droplets because it provides better security than a basic password.

Step Seven — Select the Number and Names of the Droplets to Create

Next, you can choose the number and names of the Droplets you wish to create. Depending on the number of Droplets currently in your account, you can create up to five Droplets that will use the configuration that you have selected. By default, a single Droplet is set to be created. You can adjust the number of Droplets to create be clicking the plus or minus buttons.

Step Eight — Create Your Droplet

Once you have selected all of your preferred options, click on "Create".After your Droplet is created, its root password will arrive in your email inbox and the Droplet will be set up. If you included an SSH key in the previous steps, you will not be emailed a root password — use your SSH private key to authenticate as the root user instead.

With that, your server is ready!

<u>Initial Server Setup with Ubuntu 14.04</u>

When you first create a new Ubuntu 14.04 server, there are a few configuration steps that you should take early on as part of the basic setup. This will increase the security and usability of your server and will give you a solid foundation for subsequent actions.

Step One — Root Login

To log into your server, you will need to know your server's public IP address and the password for the "root" user's account. If you have not already logged into your server, you may want to follow the first tutorial in this series, How to Connect to Your Droplet with SSH, which covers this process in detail.

If you are not already connected to your server, go ahead and log in as the root user using the following command (substitute the highlighted word with your server's public IP address):

```
local $ssh root@SERVER_IP_ADDRESS
```

Step Two — Create a New User

Once you are logged in as root, we're prepared to add the new user account that we will use to log in from now on. #adduser demo

Step Three — Root Privileges

Now, we have a new user account with regular account privileges. However, we may sometimes need to do administrative tasks.:

# gpasswd -a demo sudo

Step Four — Add Public Key Authentication (Recommended)

The next step in securing your server is to set up public key authentication for your new user. Setting this up will increase the security of your server by requiring a private SSH key to log in.

Generate a Key Pair

To generate a new key pair, enter the following command at the terminal of your local machine (ie. your computer):

local $ssh-keygen

Step Five — Configure SSH Daemon

Begin by opening the configuration file with your text editor as root:

# nano /etc/ssh/sshd_config.

Step Six -- Reload SSH

Now that we have made our change, we need to restart the SSH service so that it will use our new configuration.

Type this to restart SSH:

# service ssh start.

For the server that we showed you how to configure above, you would connect using this command. Substitute your own user name and server IP address where appropriate:

ssh demo@SERVER_IP_ADDRESS

At this point, you have a solid foundation for your server. You can install any of the software you need on your server now.

To Set Up Highly Available Web Servers with Keepalived and Floating IPs on Ubuntu 14.04

Install and Configure Nginx

Start off by updating the local package index on each of your servers. We can then install Nginx:

```
$ sudo apt-get update
$ sudo apt-get install nginx
```

In most cases, for a highly available setup, you would want both servers to serve exactly the same content. However, for the sake of clarity, in this guide we will use Nginx to indicate which of the two servers is serving our requests at any given time. To do this, we will change the default index.html page on each of our hosts. Open the file now:

```
$ sudo nano /usr/share/nginx/html/index.html
```

On your first server, replace the contents of the file with this:

Primary server's /usr/share/nginx/html/index.html

```
<h1>Primary</h1>
```

On your second server, replace the contents of the file with this:

Secondary server's /usr/share/nginx/html/index.html

```
<h1>Secondary</h1>
```

Build and Install Keepalived

Next, we will install the keepalived daemon on our servers. There is a version of keepalived in Ubuntu's default repositories, but it is outdated and suffers from a few bugs that prevent our configuration from working. Instead, we will install the latest version of keepalived from source.

```
$ sudo apt-get install build-essential libssl-dev
```

Once the dependencies are in place, we can download the tarball for keepalived. Visit this page to find the latest version of the software. Right-click on the latest version and copy the link address. Back on your servers, move to your home directory and use wget to grab the link you copied:

```
$ cd ~
$ wget http://www.keepalived.org/software/keepalived-1.2.19.tar.gz
```

Use the tar command to expand the archive and then move into the resulting directory:

```
$ tar xzvf keepalived*
$ cd keepalived*
```

Build and install the daemon by typing:

```
$ ./configure
$ make
$ sudo make install
```

The daemon should now be installed on the system.

Create a Keepalived Upstart Script

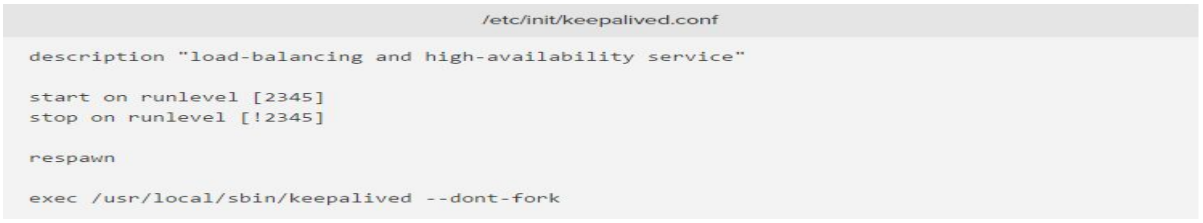
We can create a very simple Upstart script that can handle our keepalived service. Open a file called keepalived.conf within the /etc/init directory to get started:

```
$ sudo nano /etc/init/keepalived.conf
```

/etc/init/keepalived.conf

```
description "load-balancing and high-availability service"

start on runlevel [2345]
stop on runlevel [!2345]
```

Because this service is integral to ensuring our web service remains available, we want to restart this service in the event of a failure. We can then specify the actual exec line that will start the service. We need to add the --dont-fork option so that Upstart can track the pid correctly:

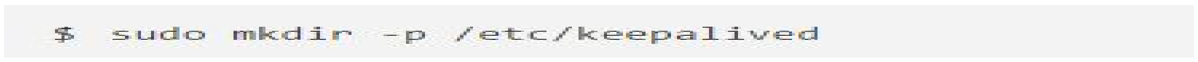


/etc/init/keepalived.conf

```
description "load-balancing and high-availability service"

start on runlevel [2345]
stop on runlevel [!2345]

respawn

exec /usr/local/sbin/keepalived --dont-fork
```

Create the Keepalived Configuration File

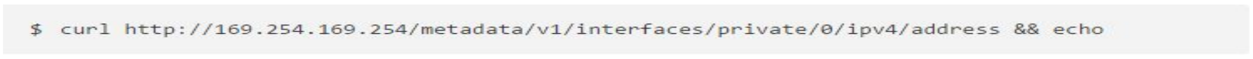
With our Upstart file in place, we can now move on to configuring keepalived.The service looks for its configuration files in the /etc/keepalived directory. Create that directory now on both of your servers:



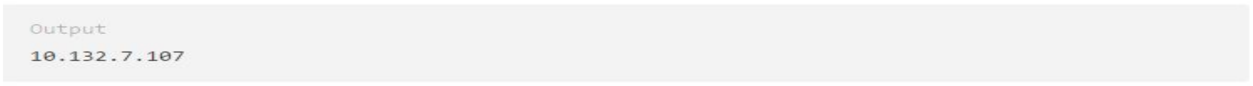
```
$ sudo mkdir -p /etc/keepalived
```

Collecting the Private IP addresses of your Servers

Before we create the configuration file, we need to find the private IP addresses of both of our servers. On DigitalOcean servers, you can get our private IP address through the metadata service by typing:



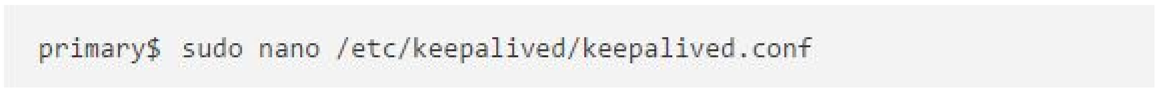
```
$ curl http://169.254.169.254/metadata/v1/interfaces/private/0/ipv4/address && echo
```



```
Output
10.132.7.107
```

Creating the Primary Server's Configuration

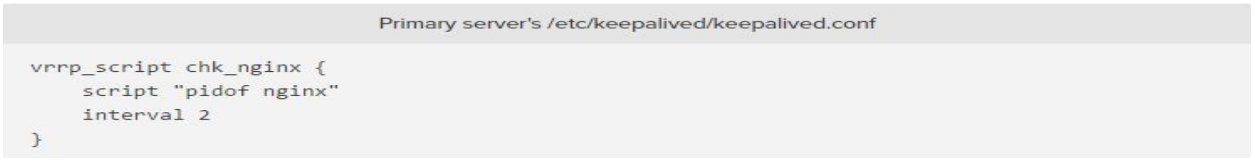
Next, on your primary server, create the main keepalived configuration file. The daemon looks for a file called keepalived.conf inside of the /etc/keepalived directory:



```
primary$ sudo nano /etc/keepalived/keepalived.conf
```

Inside, we will start by defining a health check for our Nginx service by opening up a vrrp_script block. This will allow keepalived to monitor our web server for failures so that it can signal that the process is down and begin recover measures.

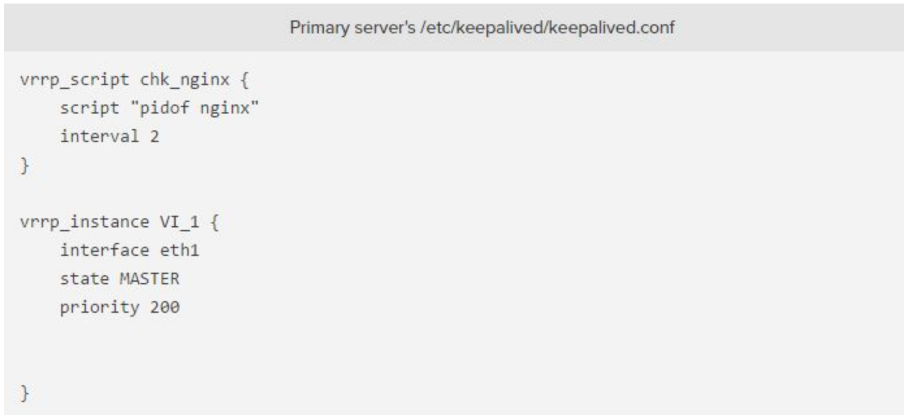
Our check will be very simple. Every two seconds, we will check that a process called nginx is still claiming a pid:



Primary server's /etc/keepalived/keepalived.conf

```
vrrp_script chk_nginx {
    script "pidof nginx"
    interval 2
}
```

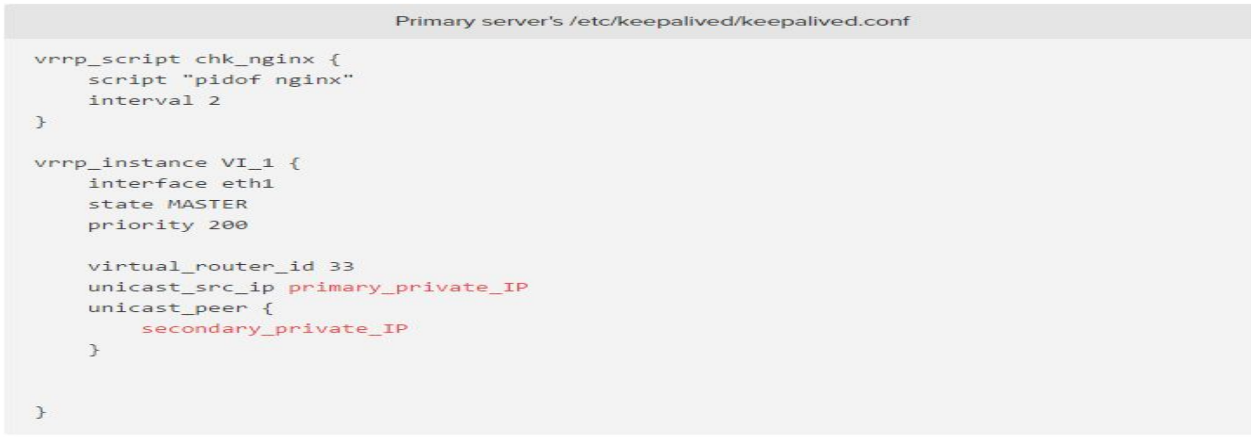
Next, we will open a block called vrrp_instance. This is the main configuration section that defines the way that keepalived will implement high availability.



Primary server's /etc/keepalived/keepalived.conf

```
vrrp_script chk_nginx {
    script "pidof nginx"
    interval 2
}

vrrp_instance VI_1 {
    interface eth1
    state MASTER
    priority 200


}
```

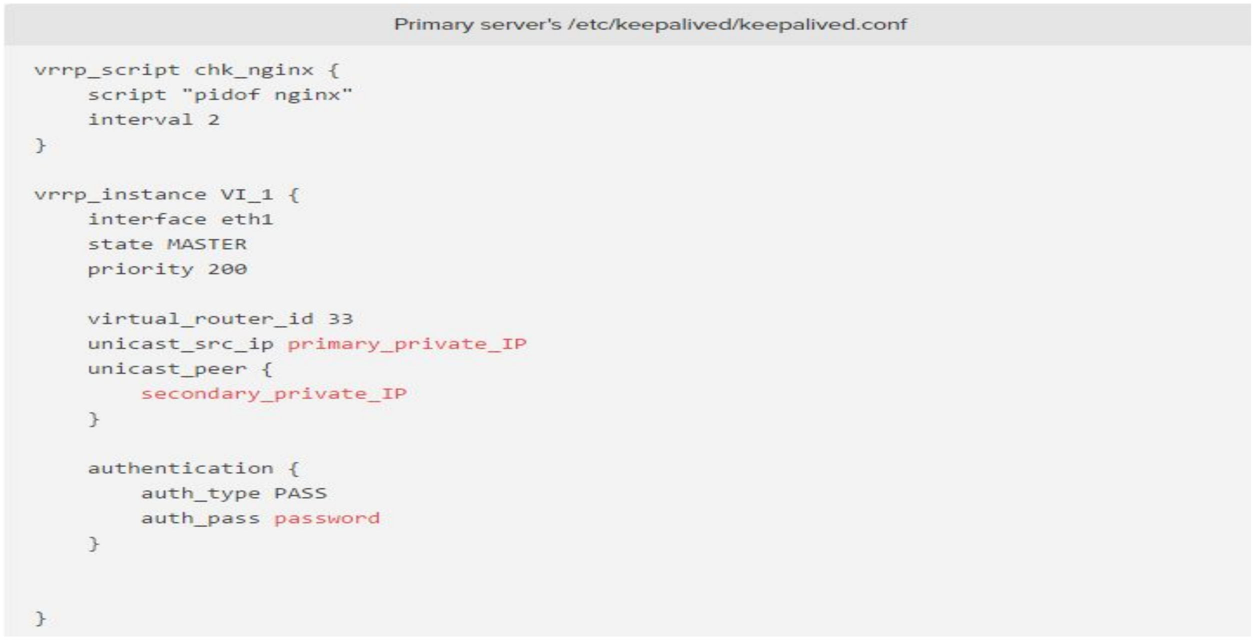
Next, we will assign an ID for this cluster group that will be shared by both nodes. We will use "33" for this example. We need to set unicast_src_ip to our primary server's private IP address that we retrieved earlier. We will set unicast_peer to our secondary server's private IP address:



Primary server's /etc/keepalived/keepalived.conf

```
vrrp_script chk_nginx {
    script "pidof nginx"
    interval 2
}

vrrp_instance VI_1 {
    interface eth1
    state MASTER
    priority 200

    virtual_router_id 33
    unicast_src_ip primary_private_IP
    unicast_peer {
        secondary_private_IP
    }


}
```

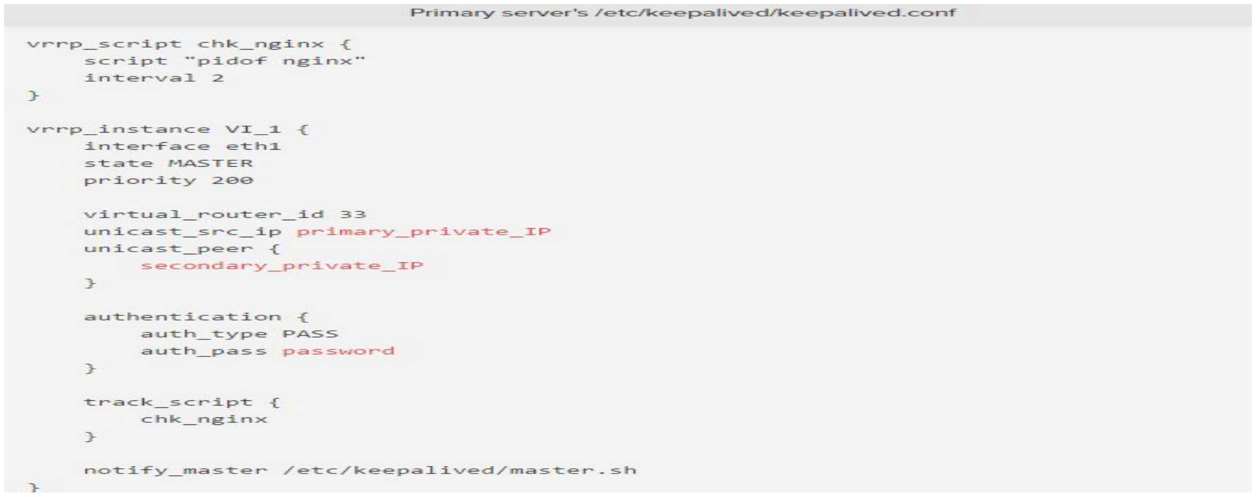
Next, we can set up some simple authentication for our keepalived daemons to communicate with one another. This is just a basic measure to ensure that the servers in question are legitimate. Create an authentication sub-block. Inside, specify password authentication by setting the auth_type. For the auth_pass parameter, set a shared secret that will be used by both nodes. Unfortunately, only the first eight characters are significant:



Primary server's /etc/keepalived/keepalived.conf

```
vrrp_script chk_nginx {
    script "pidof nginx"
    interval 2
}

vrrp_instance VI_1 {
    interface eth1
    state MASTER
    priority 200

    virtual_router_id 33
    unicast_src_ip primary_private_IP
    unicast_peer {
        secondary_private_IP
    }

    authentication {
        auth_type PASS
        auth_pass password
    }


}
```

Next, we will tell keepalived to use the routine we created at the top of the file, labeled chk_nginx, to determine the health of the local system. Finally, we will set a notify_master script, which is executed whenever this node becomes the "master" of the pair. This script will be responsible for triggering the floating IP address reassignment. We will create this script momentarily:



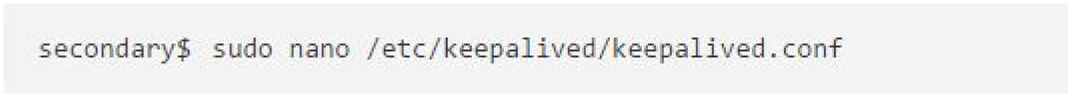
Primary server's /etc/keepalived/keepalived.conf

```
vrrp_script chk_nginx {
    script "pidof nginx"
    interval 2
}

vrrp_instance VI_1 {
    interface eth1
    state MASTER
    priority 200

    virtual_router_id 33
    unicast_src_ip primary_private_IP
    unicast_peer {
        secondary_private_IP
    }

    authentication {
        auth_type PASS
        auth_pass password
    }

    track_script {
        chk_nginx
    }

    notify_master /etc/keepalived/master.sh
}
```

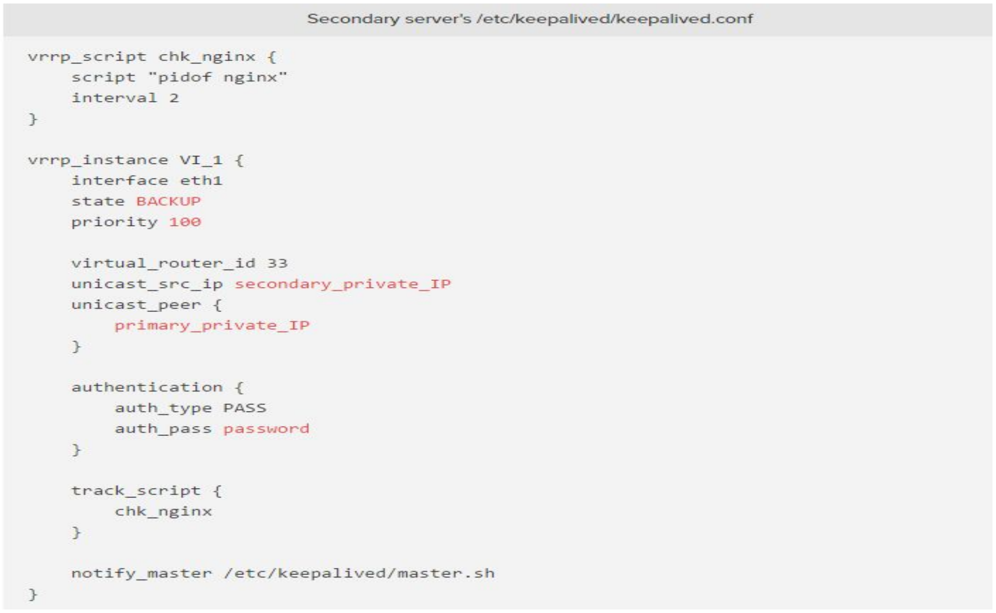
Creating the Secondary Server's Configuration

Next, we will create the companion script on our secondary server. Open a file at

/etc/keepalived/keepalived.conf on your secondary server:



```
secondary$ sudo nano /etc/keepalived/keepalived.conf
```



Secondary server's /etc/keepalived/keepalived.conf

```
vrrp_script chk_nginx {
    script "pidof nginx"
    interval 2
}

vrrp_instance VI_1 {
    interface eth1
    state BACKUP
    priority 100

    virtual_router_id 33
    unicast_src_ip secondary_private_IP
    unicast_peer {
        primary_private_IP
    }

    authentication {
        auth_type PASS
        auth_pass password
    }

    track_script {
        chk_nginx
    }

    notify_master /etc/keepalived/master.sh
}
```

Create the Floating IP Transition Scripts

Next, we will need to create a pair of scripts that we can use to reassign the floating IP address to the current Droplet whenever the local keepalived instance becomes the master server.

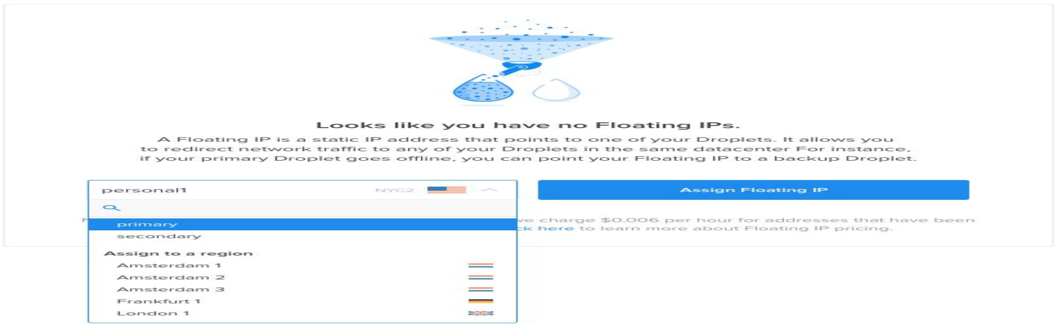
Create a DigitalOcean API Token

In order to use the script above, we will need to create a DigitalOcean API token in our account.In the control panel, click on the "API" link at the top. On the right-hand side of the API page, click "Generate new token":

On the next page, select a name for your token and click on the "Generate Token" button:

Configure a Floating IP for your Infrastructure

Next, we will create and assign a floating IP address to use for our servers.In the DigitalOcean control panel, click on the "Networking" tab and select the "Floating IPs" navigation item. Select the Droplet from the list that you assigned as your "primary" server:

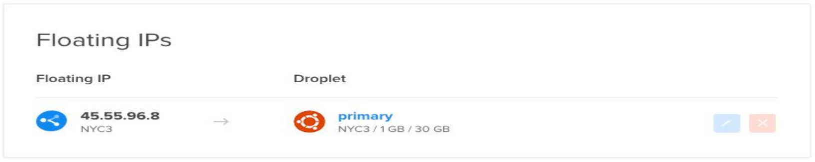


Looks like you have no Floating IPs.

A Floating IP is a static IP address that points to one of your Droplets. It allows you to redirect network traffic to any of your Droplets in the same datacenter For instance, if your primary Droplet goes offline, you can point your Floating IP to a backup Droplet.

ve charge $0.006 per hour for addresses that have been
ck here to learn more about Floating IP pricing.

A new floating IP address will be created in your account and assigned to the Droplet specified:



Floating IPs

Floating IP	Droplet
45.55.96.8 NYC3	primary NYC3 / 1 GB / 30 GB

Primary

Copy the floating IP address down. You will need this value in the script below.

Create the Wrapper Script

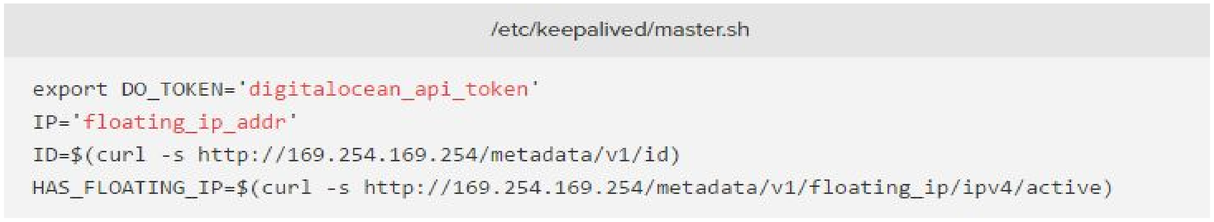
Create the file now on both servers by typing:

```
$ sudo nano /etc/keepalived/master.sh
```

Inside, start by assigning and exporting a variable called DO_TOKEN that holds the API token you just created. Below that, we can assign a variable called IP that holds your floating IP address:

```
/etc/keepalived/master.sh

export DO_TOKEN='digitalocean_api_token'
IP='floating_ip_addr'
```

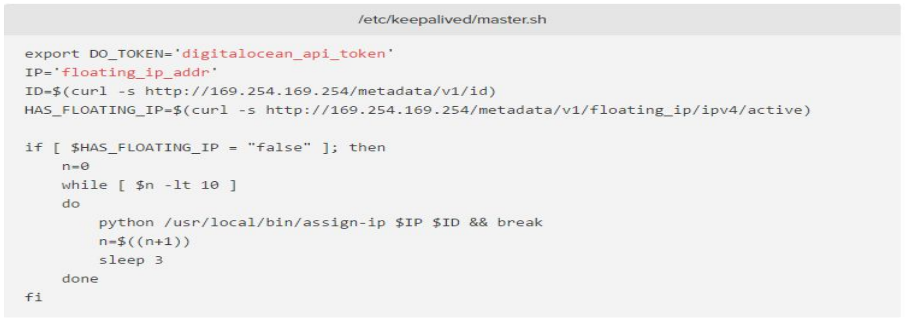
Next, we will use curl to ask the metadata service for the Droplet ID of the server we're currently on. This will be assigned to a variable called ID. We will also ask whether this Droplet currently has the floating IP address assigned to it. We will store the results of that request in a variable calledHAS_FLOATING_IP:



/etc/keepalived/master.sh

```
export DO_TOKEN='digitalocean_api_token'
IP='floating_ip_addr'
ID=$(curl -s http://169.254.169.254/metadata/v1/id)
HAS_FLOATING_IP=$(curl -s http://169.254.169.254/metadata/v1/floating_ip/ipv4/active)
```

To handle cases where the floating IP already has an event in progress, we will retry the assign-ip script a few times. Below, we attempt to run the script 10 times, with a 3 second interval between each call. The loop will end immediately if the floating IP move is successful:

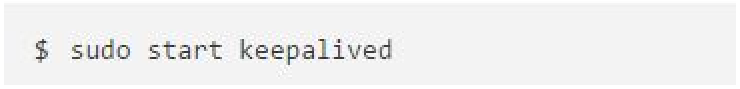


/etc/keepalived/master.sh

```
export DO_TOKEN='digitalocean_api_token'
IP='floating_ip_addr'
ID=$(curl -s http://169.254.169.254/metadata/v1/id)
HAS_FLOATING_IP=$(curl -s http://169.254.169.254/metadata/v1/floating_ip/ipv4/active)

if [ $HAS_FLOATING_IP = "false" ]; then
    n=0
    while [ $n -lt 10 ]
    do
        python /usr/local/bin/assign-ip $IP $ID && break
        n=$((n+1))
        sleep 3
    done
fi
```

Start Up the Keepalived Service

The keepalived daemon and all of its companion scripts should now be completely configured. We can start the service on both of our machines by typing:



```
$ sudo start keepalived
```

When both servers are healthy, if you visit your floating IP in your web browser, you should be taken to the primary server's Nginx page.