



NginX Nedir

NginX, ters proxy, yük dengeleyici, posta proxy'si ve HTTP önbelleği olarak da kullanılabilen bir web sunucusudur. Web sunucularının büyük bir kısmı NginX'i, genellikle bir yük dengeleyici olarak kullanır.^[1]

Detaylı Bilgi: <https://nginx.org/en/>

Yük Dengeleme (Load Balancing) Nedir

Yük dengeleme, işlemleri toplamda daha verimli hale getirmek amacıyla bir dizi görevi bir dizi kaynak (bilgi işlem birimleri) üzerinde dağıtma sürecini ifade eder. Yük dengeleme, yanıt süresini optimize edebilir ve diğer hesaplama düğümleri boşa kalırken bazı hesaplama düğümlerinin eşit olmayan şekilde aşırı yüklenmesini önleyebilir.^[2]

NginX Kurulumu

NginX'in açık kaynaklı (NginX Open Source) ve ticari (NginX Plus) sürümleri mevcuttur. Bu sürümler, farklı işletim sistemleri için aşağıdaki linklerde bulunan yönergeler doğrultusunda indirilip kurulabilir:

- Amazon Linux, CentOS, Debian, FreeBSD, Oracle Linux, Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), veya Ubuntu: <https://docs.nginx.com/nginx/admin-guide/installing-nginx>
- Windows: <https://www.javatpoint.com/how-to-install-nginx-on-windows>
- MacOS: <https://www.javatpoint.com/installing-nginx-on-mac>

NginX Konfigürasyonu

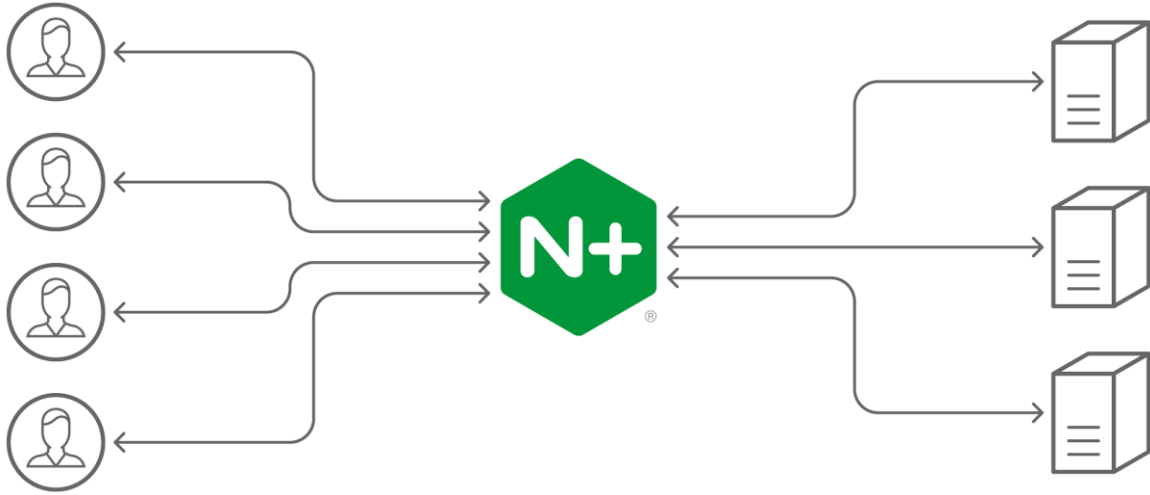
NginX'in kullanılabilmesi için öncelikle kullanım amacı doğrultusunda konfigürasyonu yapılmalıdır. NginX konfigürasyon dosyası NginX Plus için `/etc/nginx` altında bulunurken, NginX Open Source için ise kurulum esnasında kullanılan paket sistemine ve sahip olunan işletim sistemine göre `/usr/local/nginx/conf`, `/etc/nginx`, veya `/usr/local/etc/nginx` altında bulunabilir.^[3] Bu dosya, farklı ek konfigürasyonların yönetilebilmesi adına `/etc/nginx/conf.d/` altındaki `.conf` uzantılı dosyalar ile Ubuntu ve Debian'da `/etc/nginx/sites-enabled/` altında bulunan ve `/etc/nginx/sites-available` klasöründeki dosyalarına işaret eden sembolik linkleri `include` eder. NginX farklı uygulamalar için ve farklı amaçlar doğrultusunda yapılandırılırken oluşturulan konfigürasyon dosyaları bu klasörlerde konumlandırılmalıdır. `/etc/nginx/sites-`

available klasörüne konumlandırılan konfigürasyon dosyalarını aktifleştirmek için ayrıca /etc/nginx/sites-enabled/ altında birer sembolik link oluşturulmalıdır.

NginX konfigürasyon sözdizimi, direktiflerden ve parametrelerinden oluşur. Basit (tek satırlı) yönergelerin her biri noktalı virgülle biter. Diğer yönergeler, ilgili alt yönergeleri küme parantezleri içine alarak bir araya getiren “container’lar” olarak işlev görür. Bunlara genellikle “block (blok)” denir.

NginX ile Yük Dengeleme

NginX, diğer fonksiyonlarının yanı sıra, yazılım tabanlı bir yük dengeleyici olarak kullanılabilir. Bu şekilde kullanıldığında NginX, client’lar (istemciler) ve server’lar (sunucular) arasında bir katman olarak bulunur. İstemcilerden sunuculara gönderilen web request’leri önce NginX’e ulaşır. NginX, bu request’leri sunucular arasında paylaştırarak sunucuların eşit yükler altında çalışmasını sağlamayı hedefler. Bu sayede uzun süren bağlantılar bazı sunucularda kümelenirken diğer sunucuların boş kalması önlenerek genel toplamda zaman optimizasyonu sağlanabilir.



Görsel 1: NginX yük dengeleyici çalışma prensibi şeması

(Kaynak: <https://www.nginx.com/products/nginx/load-balancing/>)

NginX ile yük dengeleme yapabilmek için önce yük dengeleme konfigürasyonunu yapmak gerekir. Bu bağlamda, bir *upstream* bloğu içerisinde web sunucusunun çalışacağı gerçek adres ve portlar listelenmeli; ardından gelecek bir *server* bloğunda ise NginX’in dinleyeceği port ve domain adı tanımlanmalı ve bir *location* alt bloğu içerisinde, gelen request’leri yönlendireceği *upstream* bloğu belirtilmelidir. Aşağıdaki konfigürasyon, en basit NginX yük dengeleme konfigürasyonuna bir örnektir.

nginx.conf

```
http {
    upstream myproject {
        server 127.0.0.1:8000 weight=3;
        server 127.0.0.1:8001;
        server 127.0.0.1:8002;
        server 127.0.0.1:8003;
    }

    server {
        listen 80;
        server_name www.domain.com;
        location / {
            proxy_pass http://myproject;
        }
    }
}
```

Görsel 2: Basit bir NginX yük dengeleme konfigürasyonu

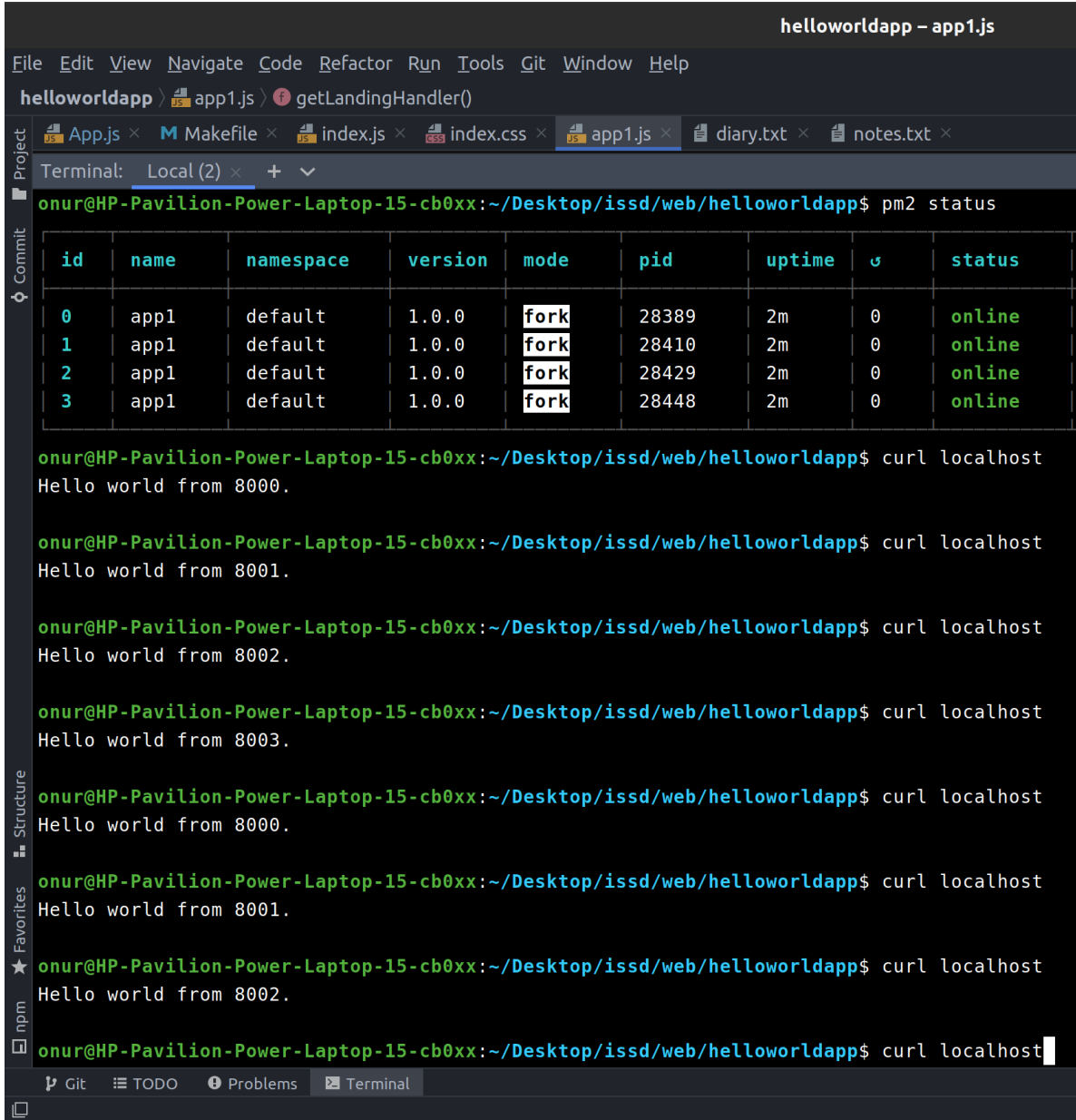
(Kaynak: <https://www.nginx.com/resources/wiki/start/topics/examples/loadbalanceexample/>)

Görseldeki basit konfigürasyonu inceleyecek olursak:

- *upstream* bloğu, “myproject” ismi verilerek tanımlanmış.
 - *server* direktifi yardımıyla 4 adet sunucunun asıl adresi ve portu listelenmiş.
 - 8000 portundaki sunucunun ağırlığı *weight* parametresi yardımıyla 3 olarak belirlenmiş (bu gibi detaylı parametre ve direktiflere sonraki bölümlerde daha detaylıca değinilecek).
- *server* bloğu tanımlanmış.
 - *listen* direktifi ile NginX, port 80’i (0.0.0.0:80) dinleyecek şekilde ayarlanmış.
 - *server_name* direktifi yardımıyla, *listen* direktifinin yeterince spesifik olmadığı durumlarda request’leri işleyecek *server* bloğunu seçebilmek için domain adı “www.domain.com” olarak belirlenmiş.
 - *location* bloğu aracılığıyla, “/” ile başlayan tüm URI’lar için proxy’nin gönderileceği *upstream* bloğu (“myproject”) belirtilmiş.

Not: Eğer NginX tarafından sağlanan *nginx.conf* dosyası değiştirilmeden bu dosyanın *include* path’lerinde ek konfigürasyon dosyaları oluşturulmak isteniyorsa, *upstream* ve *server* blokları *http* bloğu içerisine sarılmamalıdır. Bunun sebebi, hazırda bulunan *nginx.conf* dosyasında bu *http* bloğunun zaten tanımlanmış olmasıdır.

Bu konfigürasyon ile yapılandırıldığında NginX, port 80’e gönderilen request’leri 127.0.0.1 adresindeki 8000, 8001, 8002 ve 8003 portları arasında dağıtır (*upstream* bloğunda hiçbir yük dengeleme metodu belirtilmediği için round-robin metodu kullanılır). Böylece gönderilen her bir request sırayla bir sonraki sunucuya yönlendirilmiş olur.



The screenshot shows a VS Code editor window titled "helloworldapp - app1.js". The terminal window is open, showing the command "pm2 status" and its output, which is a table of PM2 processes. Below the table, several "curl localhost" commands are executed, each returning a "Hello world from" message followed by a port number (8000, 8001, 8002, 8003). The terminal output is as follows:

```
onur@HP-Pavilion-Power-Laptop-15-cb0xx:~/Desktop/issd/web/helloworldapp$ pm2 status
```

id	name	namespace	version	mode	pid	uptime	⌵	status
0	app1	default	1.0.0	fork	28389	2m	0	online
1	app1	default	1.0.0	fork	28410	2m	0	online
2	app1	default	1.0.0	fork	28429	2m	0	online
3	app1	default	1.0.0	fork	28448	2m	0	online

```
onur@HP-Pavilion-Power-Laptop-15-cb0xx:~/Desktop/issd/web/helloworldapp$ curl localhost
Hello world from 8000.

onur@HP-Pavilion-Power-Laptop-15-cb0xx:~/Desktop/issd/web/helloworldapp$ curl localhost
Hello world from 8001.

onur@HP-Pavilion-Power-Laptop-15-cb0xx:~/Desktop/issd/web/helloworldapp$ curl localhost
Hello world from 8002.

onur@HP-Pavilion-Power-Laptop-15-cb0xx:~/Desktop/issd/web/helloworldapp$ curl localhost
Hello world from 8003.

onur@HP-Pavilion-Power-Laptop-15-cb0xx:~/Desktop/issd/web/helloworldapp$ curl localhost
Hello world from 8000.

onur@HP-Pavilion-Power-Laptop-15-cb0xx:~/Desktop/issd/web/helloworldapp$ curl localhost
Hello world from 8001.

onur@HP-Pavilion-Power-Laptop-15-cb0xx:~/Desktop/issd/web/helloworldapp$ curl localhost
Hello world from 8002.

onur@HP-Pavilion-Power-Laptop-15-cb0xx:~/Desktop/issd/web/helloworldapp$ curl localhost
```

Görsel 3: Örnek konfigürasyonun basit bir “Hello world from <PORT#>.” programıyla çıktısı

Yukarıdaki görsel, request atıldığında “Hello world from <PORT#>.” formatında çıktı veren, “app1” adında basit bir sunucu uygulamasının 4 farklı örneğinin sırasıyla port 8000, 8001, 8002 ve 8003’te çalıştırılması ve NginX’in örnek konfigürasyonda olduğu gibi bu portlar üzerinde yük dengelemesi uygulayacak şekilde yapılandırılması sonucu ortaya çıkan durumu göstermektedir. Görselde görüldüğü gibi localhost’a (127.0.0.1) varsayılan port (80) üzerinden gönderilen request’ler, NginX tarafından farklı portlardaki sunucular arasında sırasıyla paylaşılır.

Detaylı Bilgi: Temel NginX yük dengeleme direktifleri ve parametreleri ile ilgili daha fazla bilgi edinmek için; bu yazıdan daha düzgün bir terminoloji ile yazılmış olup daha teknik, daha sistematik ve daha detaylı bilgi içeren aşağıdaki üçüncü parti rehber yazılar incelenebilir:

- <https://www.plesk.com/blog/various/nginx-configuration-guide/>
- <https://www.digitalocean.com/community/tutorials/understanding-nginx-server-and-location-block-selection-algorithms>

Gelişmiş Yük Dengeleme Konfigürasyonları

Not 1: Bu bölüm büyük çoğunlukla, aşağıda linkleri verilen dokümantasyon sayfalarında bulunan bilgilerin derlenip özetlenerek çevrilmesiyle oluşturulmuştur. Olası kafa karışıklıklarında bu linklere atıfta bulunulması önerilir:

- http://nginx.org/en/docs/http/nginx_http_upstream_module.html
- <https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/>
- <https://docs.nginx.com/nginx/admin-guide/load-balancer/http-health-check/>

Not 2: Gelişmiş direktif ve parametrelerin bir kısmı NginX Open Source tarafından desteklenirken bir kısmı yalnızca ticari sürüm olan NginX Plus tarafından desteklenmektedir. Aşağıda bulunan listelerde “(N+)” ile işaretlenmiş direktif ve parametreler yalnızca NginX Plus tarafından desteklenenler olup işaret barındırmayanlar ise her iki sürüm tarafından desteklenenlerdir.

Yük Dengeleme Metotları

NginX, kendisine gönderilen request’lerin sunuculara hangi metot ile dağıtılacağını seçilmesine olanak tanır. İstenen yük dengeleme metodu *upstream* bloğunda ilgili direktif kullanılarak belirtilir. Herhangi bir metot belirtilmemesi durumunda varsayılan olarak round-robin metodu uygulanır.

- **least_conn:** Gelen request o anda en az sayıda bağlantıya sahip olan sunucuya iletilir.
- **ip_hash:** İstemciler IP adreslerine göre hash’lenerek karşılık gelen sunucuya atanırlar. Bu metot kullanıldığında bir istemciye, aktif olduğu sürece aynı sunucu yanıt verir.
- **hash:** İstemciler sahip oldukları belli bir anahtar değerin hash’ine göre sunuculara map’lenirler. “Generic hash” adı verilen bu metotta da mümkün olduğu sürece aynı istemciye aynı sunucu yanıt verir.
 - **consistent:** *hash* direktifi bu parametre ile kullanıldığında “ketama consistent hashing^[5]” metodu uygulanır. Bu metotta, listeye yapılan sunucu ekleme/çıkarma işlemlerinde daha az sayıda anahtar yeni sunuculara map’lenir ve bu sayede sonraki işlemlerdeki cache miss sayısı azaltılmış olur.
- **random:** Gelen request, rastgele seçilen bir sunucuya iletilir.
 - **two:** Sunucu ağırlıkları hesaba katılarak rastgele iki sunucu seçilir ve belirtilen bir metoda göre ikisi arasında seçim yapılır.
 - **least_conn:** O anda en az bağlantıya sahip olan sunucu (varsayılan metot).
 - **least_time^(N+):** Minimum ortalama response süresine ve minimum sayıda aktif bağlantıya sahip olan sunucu.

- **least_time^(N+)**: Gelen request'in yönlendirileceği sunucu seçilirken, sunucuların ortalama yanıt süreleri ve o anki aktif bağlantıları, sunucu ağırlıklarıyla birlikte hesaba katılır.

Sağlık Kontrolü (Health Check)

NginX, sunucuların kullanılabilirlik durumlarını denetleyebilir ve bazı sunucuların kullanılmadığı durumlarda önlemler alabilir. Sunucuların uygunluk durumlarının bu şekilde kontrol altında tutulmasına sağlık kontrolü (health check) adı verilir. NginX, sunucuların sağlık kontrollerini pasif ve aktif olmak üzere iki şekilde gerçekleştirebilir.

Pasif sağlık kontrolünde NginX, işlemleri izler ve başarısız bağlantıları yeniden sürdürmeye çalışır. İşlemin belirlenen süre boyunca ve belirtildiği kez yeniden sürdürülemediği durumlarda NginX söz konusu sunucuyu geçici süreyle servis dışı olarak işaretler ve bu sürede bu sunucuya request yönlendirmeyi durdurur. Pasif sağlık kontrolleri, *upstream* bloğundaki sunuculara ilgili parametreler ilâştirilerek etkinleştirilir.

- **max_timeout**: Sunucunun servis dışı sayılması için başarısız bağlantı girişimlerinin içerisinde gerçekleşmesi gereken süre. Ayrıca sunucunun servis dışı olarak işaretlenme süresi.
- **max_fails**: Sunucunun servis dışı sayılabilmesi için *max_timeout* ile belirtilen süre içerisinde gerçekleşmesi gereken başarısız girişim sayısı (varsayılan 1 girişim).
- **slow_start^(N+)**: Sunucu yeniden hazır olduğunda, belirtilen süre boyunca sunucunun ağırlığının 0'dan başlayarak kademeli olarak asıl düzeyine ulaşmasını sağlar. Bu sayede yeniden devreye giren sunucunun gelen request'lerle boğulması önlenmiş olur.

Aktif sağlık kontrollerinde ise NginX, sunuculara periyodik olarak özel "sağlık kontrolü" request'leri göndererek sunucuların yanıtlarına göre kullanılabilirlik durumlarını belirler. Aktif sağlık kontrolleri, *upstream* bloğunda bir paylaşımlı bellek alanı tanımlanıp *server* bloğu altındaki *location* bloğunda ilgili direktifler eklenerek etkinleştirilir. Eğer *match* bloğu kullanılacaksa o da *server* bloğunun dışına konumlandırılır.

- **health_check^(N+)**: Aktif sağlık kontrollerini etkinleştirir.
 - **port^(N+)**: Aktif sağlık kontrolleri için opsiyonel olarak belirlenen yeni port.
 - **interval^(N+)**: Aktif sağlık kontrolleri arasındaki periyot (varsayılan 5 saniye).
 - **fails^(N+)**: Sunucunun sağlıklı sayılabilmesi için gerçekleşmesi gereken başarısız girişim sayısı (varsayılan 1 girişim).
 - **passes^(N+)**: Sağlıksız olarak işaretlenmiş bir sunucunun yeniden sağlıklı sayılabilmesi için gerçekleşmesi gereken başarılı girişim sayısı (varsayılan 1 girişim).
 - **uri^(N+)**: Aktif sağlık kontrolü request'ine URI eklemeye olanak tanır. Bu URI, aktif sağlık kontrolü esnasında *upstream* bloğundaki sunuculara ilâştirilir ve sağlık kontrolü request'i bu URI'a gönderilir.
 - **mandatory^(N+)**: *upstream* grubuna yeni katılan sunucuların da sağlık kontrolünden geçmesini sağlar.

- ***persistent*^(N+)**: Konfigürasyon yeniden yüklenirken sunucuların önceki sağlık durumunun hatırlanmasını sağlar.
- ***match*^(N+)**: Sunucunun sağlık kontrolünü başarıyla geçmesi için eşleşmesi gereken *match* bloğunu belirtir (bkz. *match* bloğu).
- ***match*^(N+)**: Bir response'un eşleşmesi için status code (durum kodu), header ve body tanımlar. *health_check* direktifine eklenen *match* parametresinin işaret ettiği *match* bloğu, sunucuların sağlık kontrolü request'lerine verdikleri response'larla karşılaştırılır. Söz konusu response'un durum kodunun belli bir aralıkta olup olmadığı, header ve body içeriğinin sağlıklı olarak nitelendirilen durumda olup olmadığı gibi özellikleri kontrol edilerek sağlık kontrolünden bu karşılaştırmaya göre bir sonuç çıkması sağlanır.

NginX Open Source yalnızca pasif sağlık kontrollerini desteklerken NginX Plus aktif sağlık kontrollerini de desteklemektedir.

Diğer Sunucu Parametreleri

Aşağıdaki parametreler sunuculara iliştilerilerek kullanılır.

- ***down***: Sunucuyu servis dışı olarak işaretler. Ona yönlendirilecek olan request'ler bir sonraki sunucuya yönlendirilir.
- ***backup***: Sunucuyu bir yedek sunucu olarak işaretler. Bu sunucu ancak *upstream* grubundaki diğer tüm sunucular kullanım dışı olduğunda etkinleşir.
- ***weight***: Sunucuya belirtilen ağırlık atanır (varsayılan 1 ağırlık). Request'ler sunuculara, ağırlıklarıyla doğru orantılı olacak oranda gönderilir.
- ***max_conns*^(N+)**: Sunucunun aynı anda sahip olabileceği aktif bağlantı sayısını sınırlandırır.

Diğer Çeşitli Direktifler

- ***zone***: Belirtilen ad ve boyut doğrultusunda, *upstream* grubunun konfigürasyonu ve runtime state'ini tutan, çalışan sunucular arasında paylaşılan bir paylaşımlı bellek alanı oluşturur.
- ***queue*^(N+)**: *upstream* bloğunda, gelen bir request için hemen bir sunucunun seçilememesi durumunda request'in sıraya alınabileceği bir alan oluşturur. *queue* direktifi ile ifade edilen sıranın da dolması, istemcinin 502 (Bad Gateway) hatası almasıyla sonuçlanır.
- ***state*^(N+)**: Dinamik olarak yapılandırılmış *upstream* grubunun state'ini tutan bir dosya belirler.
- ***keepalive***: *upstream* sunucuları için ön belleği etkinleştirir ve ön bellekte tutulacak maksimum idle (başıboş) bağlantı sayısını belirler.
- ***keepalive_requests***: Bir *keepalive* bağlantısı üzerinden işlenebilecek maksimum request sayısı. Bu sayıya ulaşıldığında *keepalive* bağlantısı kapatılır.
- ***keepalive_time***: Bir *keepalive* bağlantısının request işleyebileceği maksimum süre. Bu süreye ulaşıldığında *keepalive* bağlantısı kapatılır.

- **keepalive_timeout**: Bir *upstream* sunucuyla olan başıboş bir *keepalive* bağlantısının açık kalma süresi.
- **ntlm^(N+)**: Request'lerin "NTLM Authentication^[6]" ile proxy'lenebilmesine izin verir.

Bu Dokümanda İşlenmeyen Direktifler ve Parametreler

- **resolver^(N+), resolver_timeout^(N+), sticky^(N+), sticky_cookie_insert^(obsolete)**
- **resolve, route, service, drain**

Linkler

Kaynakça

1. <https://en.wikipedia.org/wiki/Nginx>
 2. [https://en.wikipedia.org/wiki/Load_balancing_\(computing\)](https://en.wikipedia.org/wiki/Load_balancing_(computing))
 3. <https://docs.nginx.com/nginx/admin-guide/basic-functionality/managing-configuration-files/>
 4. <https://docs.nginx.com/nginx/admin-guide/basic-functionality/managing-configuration-files#directives>
 5. <https://www.metabrew.com/article/libketama-consistent-hashing-algo-memcached-clients>
 6. https://en.wikipedia.org/wiki/Integrated_Windows_Authentication
- <https://nginx.org/en/>
 - <https://www.plesk.com/blog/various/nginx-configuration-guide/>
 - <https://www.digitalocean.com/community/tutorials/understanding-nginx-server-and-location-block-selection-algorithms>
 - http://nginx.org/en/docs/http/nginx_http_upstream_module.html
 - <https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/>
 - <https://docs.nginx.com/nginx/admin-guide/load-balancer/http-health-check/>

Görseller

1. <https://www.nginx.com/products/nginx/load-balancing/>
2. <https://www.nginx.com/resources/wiki/start/topics/examples/loadbalanceexample/>

NginX Kurulum Linkleri

- <https://docs.nginx.com/nginx/admin-guide/installing-nginx>
- <https://www.javatpoint.com/how-to-install-nginx-on-windows>
- <https://www.javatpoint.com/installing-nginx-on-mac>