Dibin Joseph

Dibin.Joseph@in.ey.com

Nginx

More than a web server

Index

1. Introduction
2. Usage

2.a Web Server

2.b Reverse Proxy

2.c Load Balancer

1. Nginx with our application.

3.a Reverse Proxy

3.b Load Balancer

1. A sample configuration file



**Introduction**

NGINX is a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server. NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption.

NGINX is one of a handful of servers written to address the [C10K problem](http://www.kegel.com/c10k.html). Unlike traditional servers, NGINX doesn’t rely on threads to handle requests. Instead it uses a much more scalable event-driven (asynchronous) architecture. This architecture uses small, but more importantly, predictable amounts of memory under load. Even if you don’t expect to handle thousands of simultaneous requests, you can still benefit from NGINX’s high-performance and small memory footprint. NGINX scales in all directions: from the smallest VPS all the way up to large clusters of servers.

**Usage**

**Nginx can be used as a**

1. Web Server
2. Reverse Proxy
3. Load Balancer

**1. WebServer**

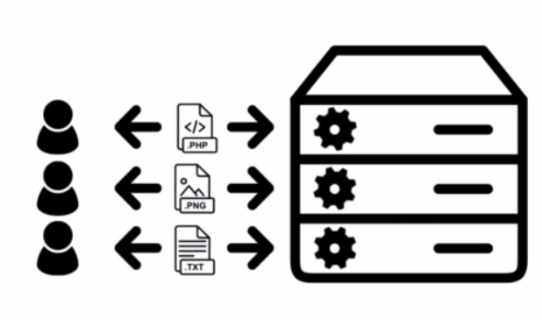
A **web server** is **server** software, or hardware dedicated to running said software, that can satisfy client requests. A **web server** processes incoming network requests over HTTP and several other related protocols. The primary function of a **web server** is to store, process and deliver **web** pages to clients.

Nginx is primarily used as a web server

**Nginx vs Apache (***a very commonly used web server***)**

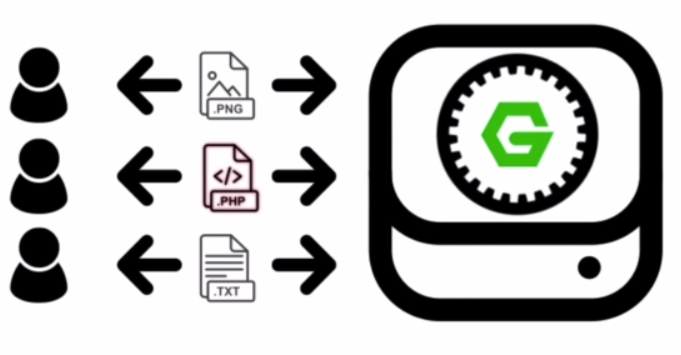
Difference in Basic Architecture

Apache by default runs on pre-fork mode. That is it spawns a set number of processes each of which can server a single request at a time.

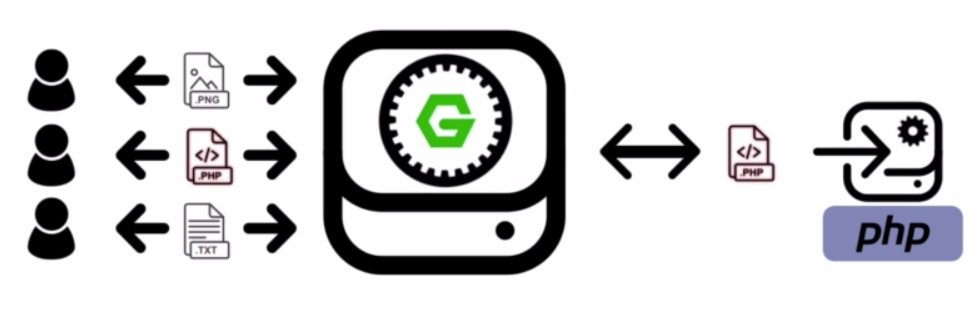


Apache Server using multiple processes each of which server one client request

Nginx manages request asynchronously meaning a single nginx can server multiple request concurrently. (Unlike Apache there’s no division of processes within Nginx)



And it can’t embed server-side programming languages to its process. Meaning all request for a dynamic content needs to be dealt with completely different process and the content resides outside of nginx



Example php content would be reverse proxied back to the client via nginx

Which makes it a server with

***High Performance***

***High Concurrency***

***Low Resource Usage***

**2. Reverse Proxy Server**

A **reverse proxy server** is a type of **proxy server** that typically sits behind the firewall in a private network and directs client requests to the appropriate backend **server**. A **reverse proxy** provides an additional level of abstraction and control to ensure the smooth flow of network traffic between clients and **servers**.

**3. Load Balancer**

**Nginx** can be configured as a simple yet powerful **load balancer** to improve your servers resource availability and efficiency. **Nginx** acts as a single-entry point to a distributed web application working on multiple separate servers.

**Nginx with our application**

1. Reverse Proxy Server
2. Load Balancer

**1.Reverse Proxy Server**

We placed nginx between our web application and our server code viz. any request from client (web application) to the server (Node Application) would be proxied through nginx



Benefits of using Nginx as a reverse proxy

**Security**

1. **Masking Server Identity** - The server details are masked i.e the client requesting response would never know the server details which serves them with data but would only receive responses as if it were being served by the Nginx itself

2. **Rate Limiting** – Nginx can be configured to limit the request rate thus protecting the server from a potential attack (example DOS attack).

Request can be limited on

* Server (Server Based) – We can put an upper limit on the rate of request hitting the nginx server
* Binary Remote Address (Client Based) – We can put an upper limit on rate of request originating from a client (IP).
* Request URI (URI/API based) – We can also put upper limit on the rate of request hitting a specific API for a response.

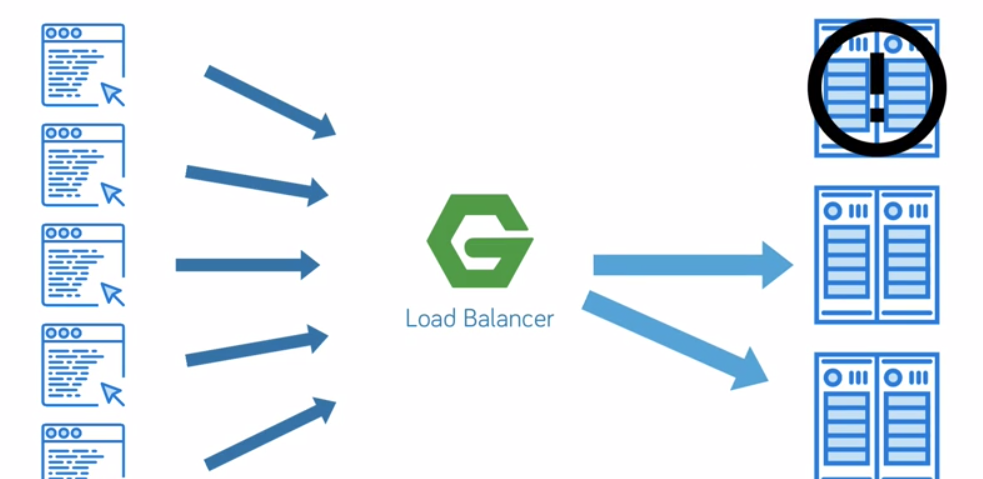
**Performance**

1. **Caching –** The static response from the actual server, in our case NodeJS , could be cached on the nginx server and sent as response directly with greatly reduces the response time .And also the nginx with NodeJS can be configured to also cache dynamic server response and serve the content from cache when there’s no change to the response.

**2.Load Balancer**

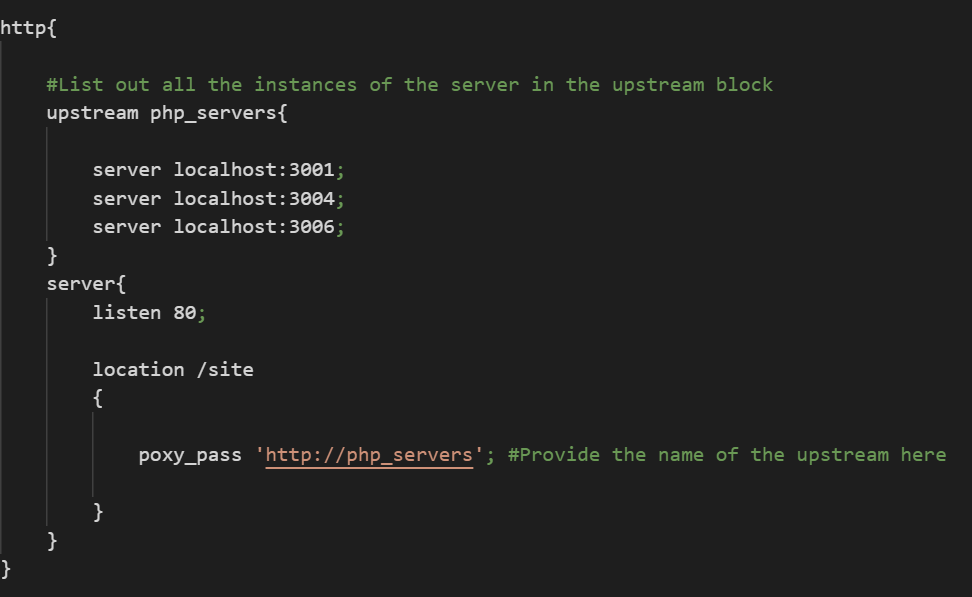
We’ve also used Nginx as a load balancer to manage the requests from the client (web app) to various different node server which nginx achieves with considerable ease.

Nginx acts as a single-entry point to a distributed web application working on multiple separate servers.



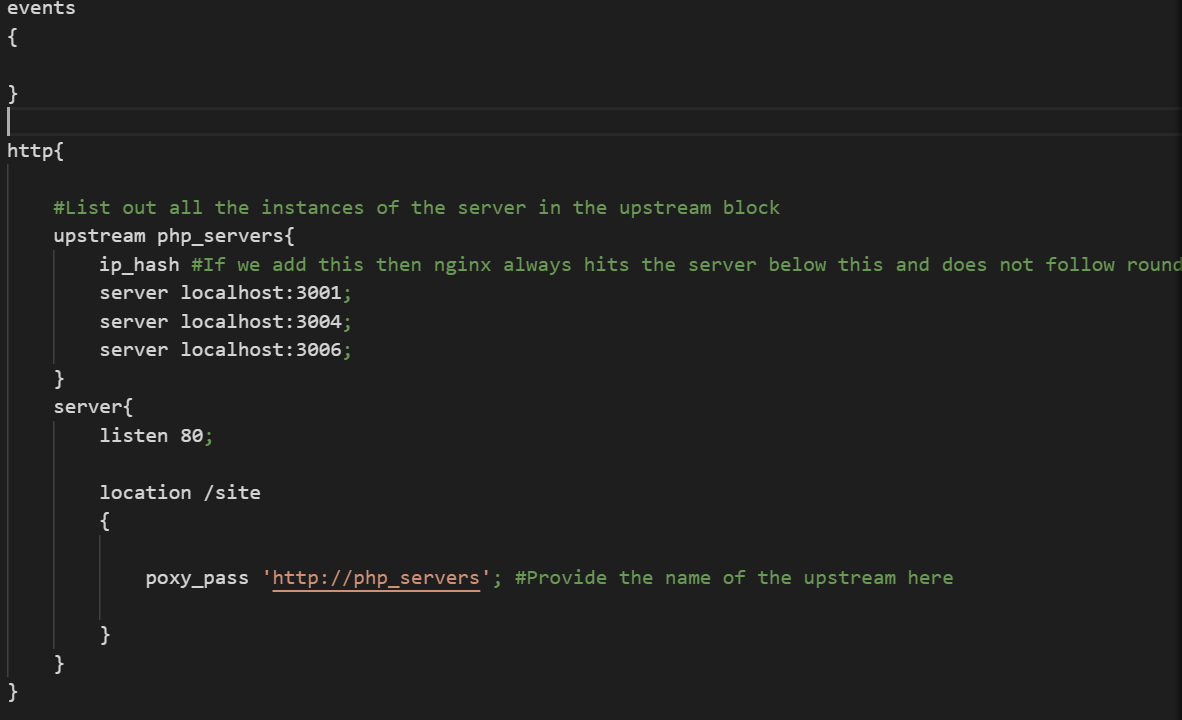
Nginx routes the client request to the multiple server in round robin fashion which is each of the (active)server serves request consecutively.

Below is a sample nginx configuration file (nginx.conf) serving request from servers in a round robin fashion



But can also be configured in such a way that the requests are always served from a server up until it becomes inactive and then nginx moves on to next server in sequence

Below is the sample configuration file for the same.



**A sample configuration file**

