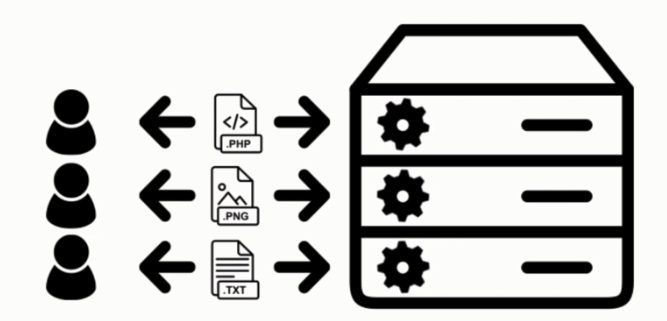
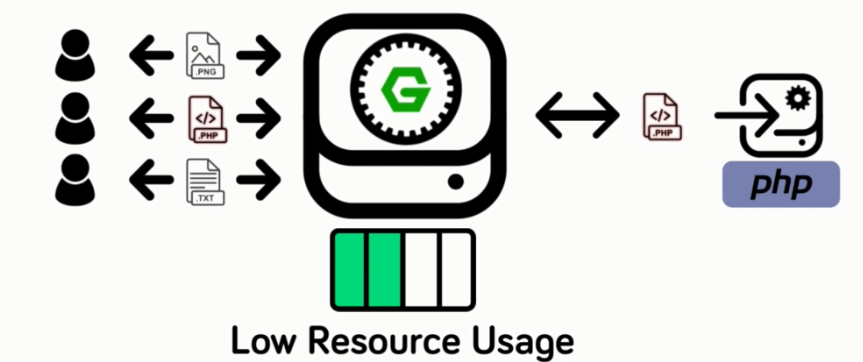
**Nginx vs Apache**

Apache server single request at a time



Whereas ngnix serves asynchronously means a single nginx process serve concurrently

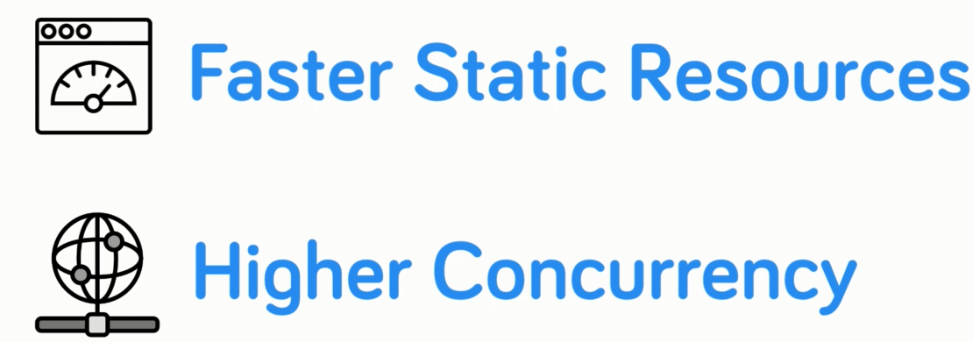
So all dynamic connect serve by different process (php) and reverse proxy via nginx to the client



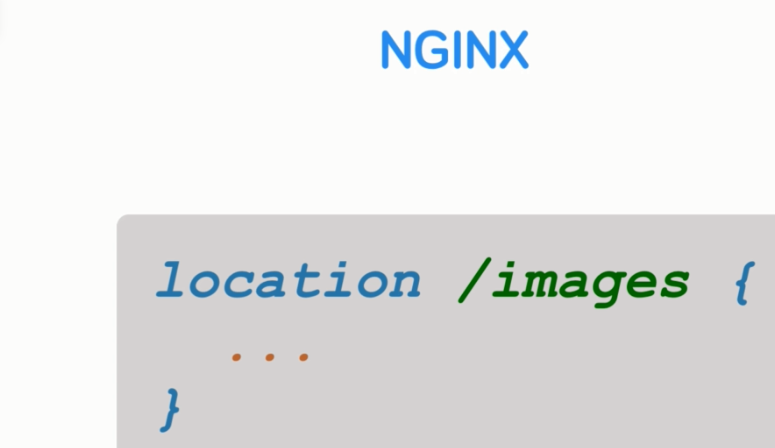


Ngnix serve static content/request without server side languages

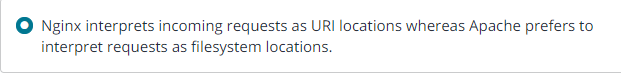
-------------------------------------------------------------------------------------------------------



**Configuration Approach**



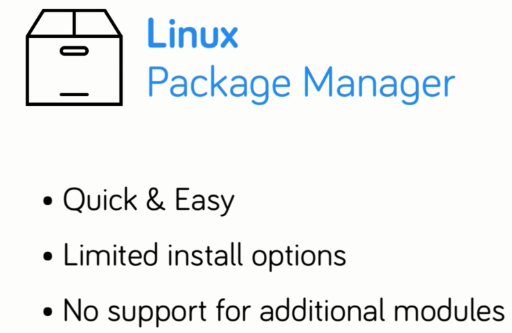


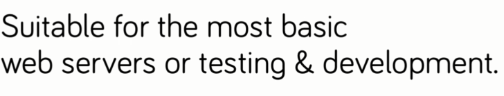


Section 2

Installing with a Package manager

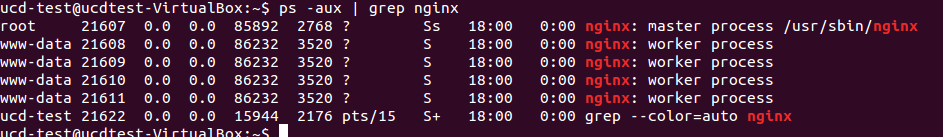
And the first installation method will cover is via the operating systems built in backage manager.



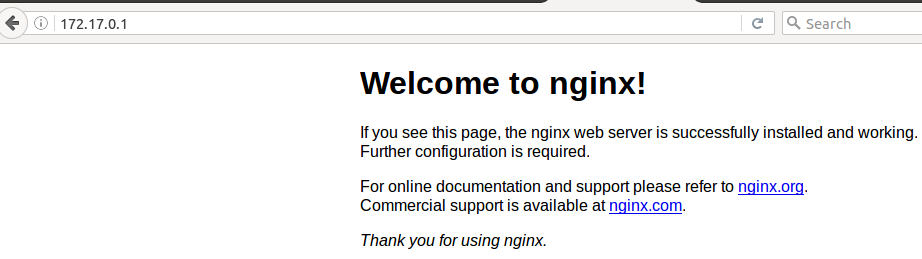


apt-get update

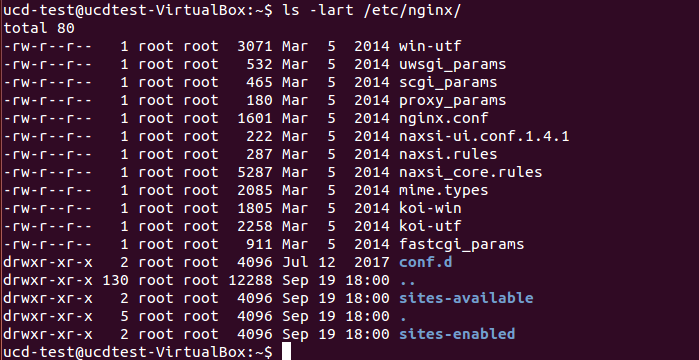
apt-get add nginx



Ifconfig



Config file are in /etc/config



Similarly for centos use. YUM command

Lect 3, Section 2



<http://nginx.org/en/docs/windows.html>

**nginx for Windows**

**docker run -d --name docker-nginx1 -p 80:80 nginx**

**Config file(s) location "C:\tools\nginx\conf.d"**

**To start service:**

net start nginx-service

or

nssm start nginx-service

**To stop service**:

net stop nginx-service

or

nssm stop nginx-service

**To restart service do both:**

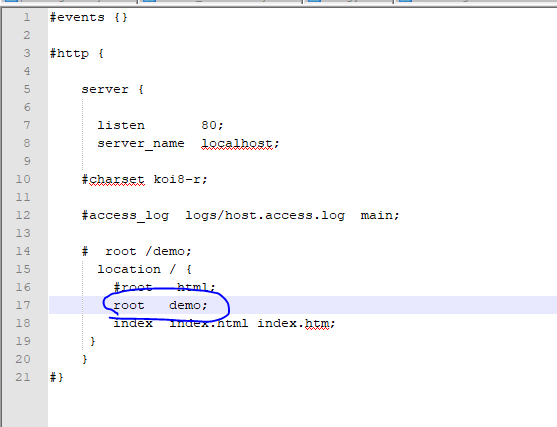
net stop nginx-service

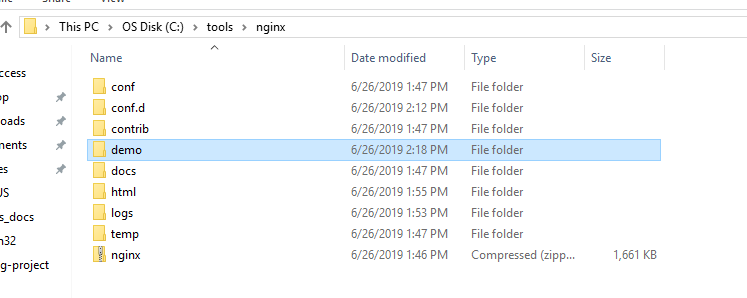
net start nginx-service

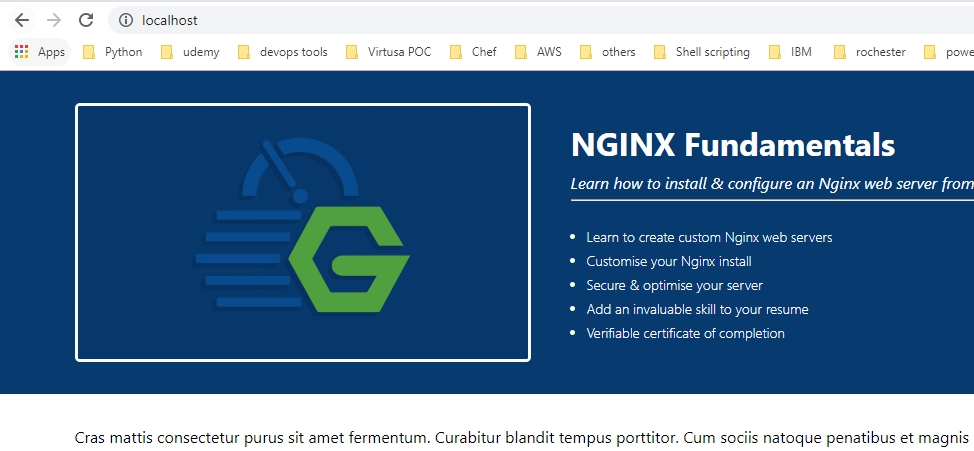
or

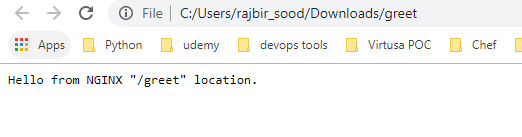
nssm stop nginx-service

nssm start nginx-service



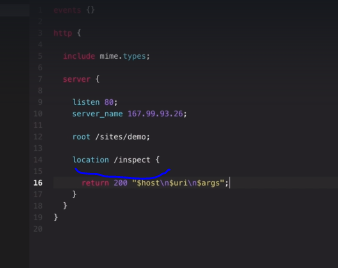


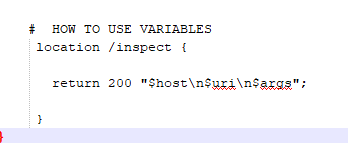


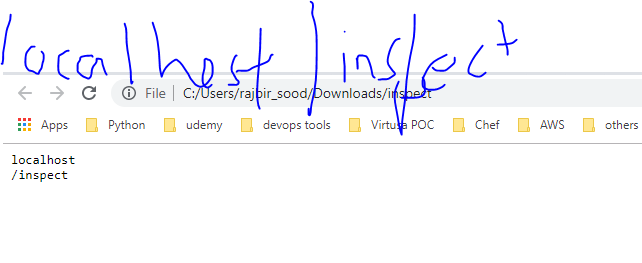


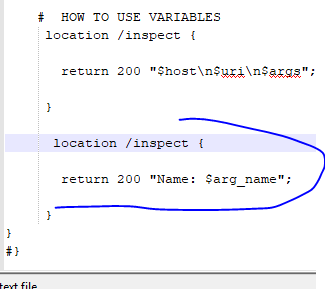
Localhost/greet

**Variable:**

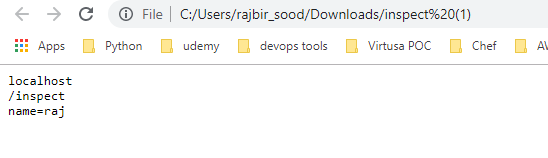




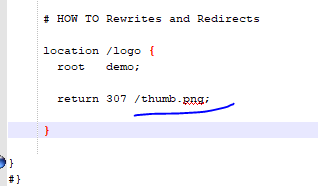


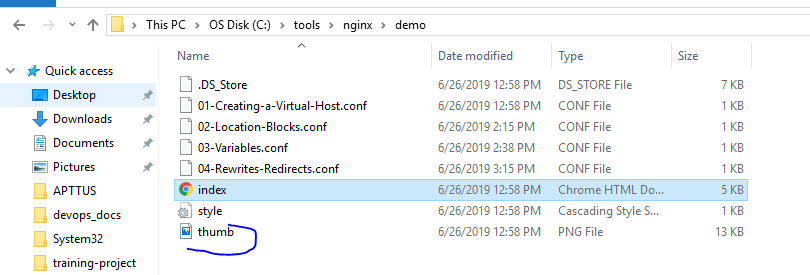


**localhost/inspect?name=raj**

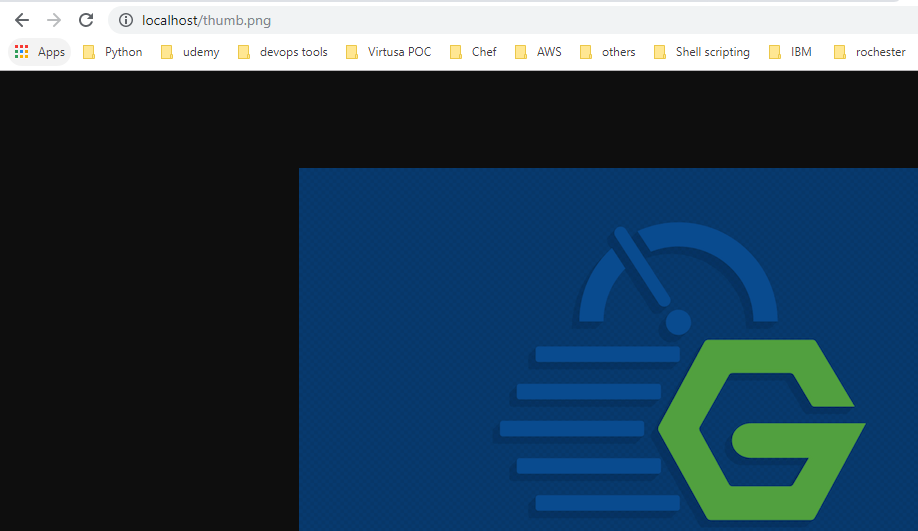


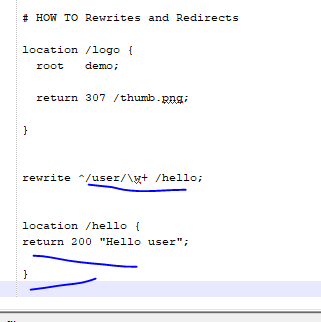
**Rewrites and Redirects**



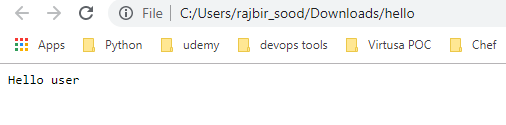


<http://localhost/logo>

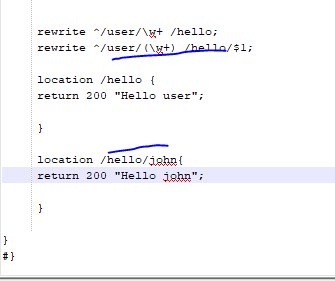




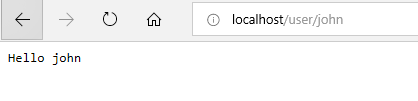
**Localhost/hello/john**



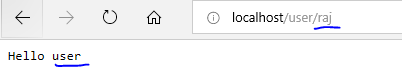
**Lets catch username now**



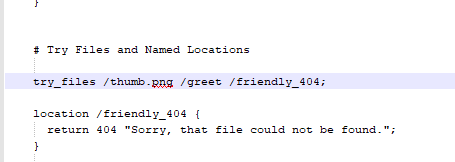
**/user/john redirected to /hello/john**





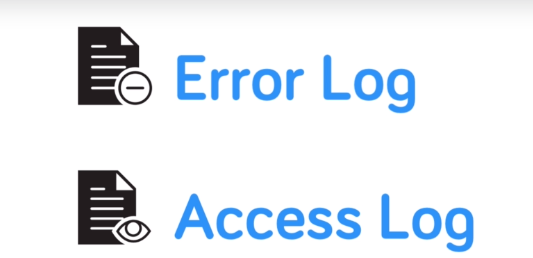


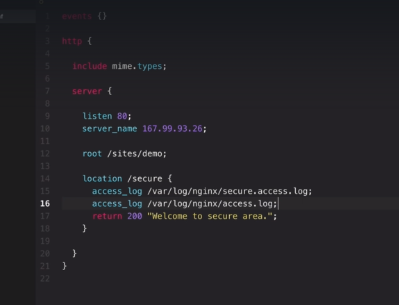
**Try Files and Named Locations**



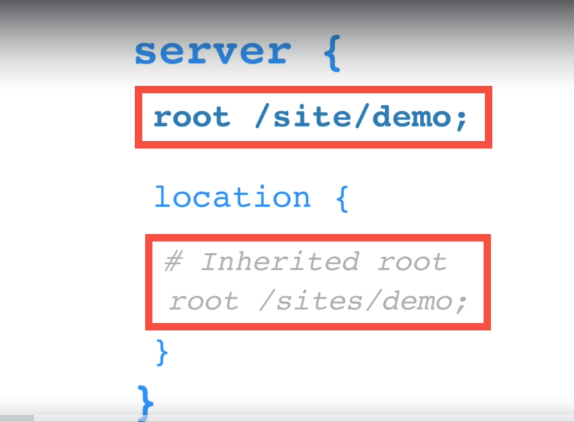
**Not worked**

**Logging**

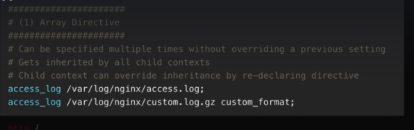


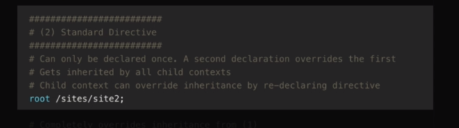


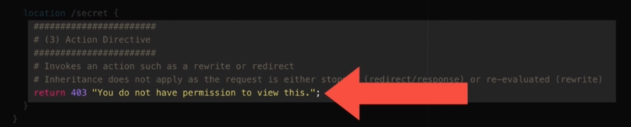
**Inheritance and Directive Types**











**Security (Section 5)**

Install openssl to create self signed certificate

<https://chocolatey.org/packages/OpenSSL.Light>

choco install openssl.light

choco upgrade openssl.light

<https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-in-ubuntu-16-04>

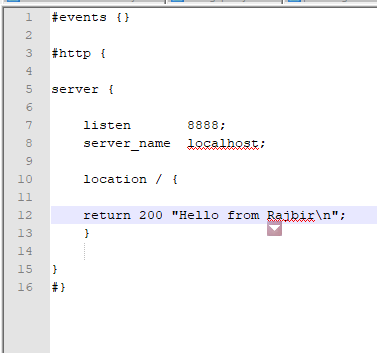
**Reverse proxy**

So what exactly is reverse proxy in.

Simply put a reverse proxy act as an intermediary between a client for example a browser and the resource

itself.

$ php -S 127.0.0.1:8000



rajbir\_sood@PTL06625 MINGW64 /

$ curl localhost:8888/

% Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

100 18 100 18 0 0 73 0 --:--:-- --:--:-- --:--:-- **73Hello from Rajbir**

To test our nginx in a reverse proxy and loadbalancing, we will use simple php server

rajbir\_sood@PTL06625 MINGW64 /

$ cd "C:\tools\nginx"

rajbir\_sood@PTL06625 MINGW64 /c/tools/nginx

$ pwd

/c/tools/nginx

rajbir\_sood@PTL06625 MINGW64 /c/tools/nginx

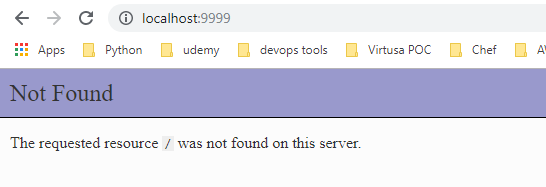
$ php -S localhost:9999

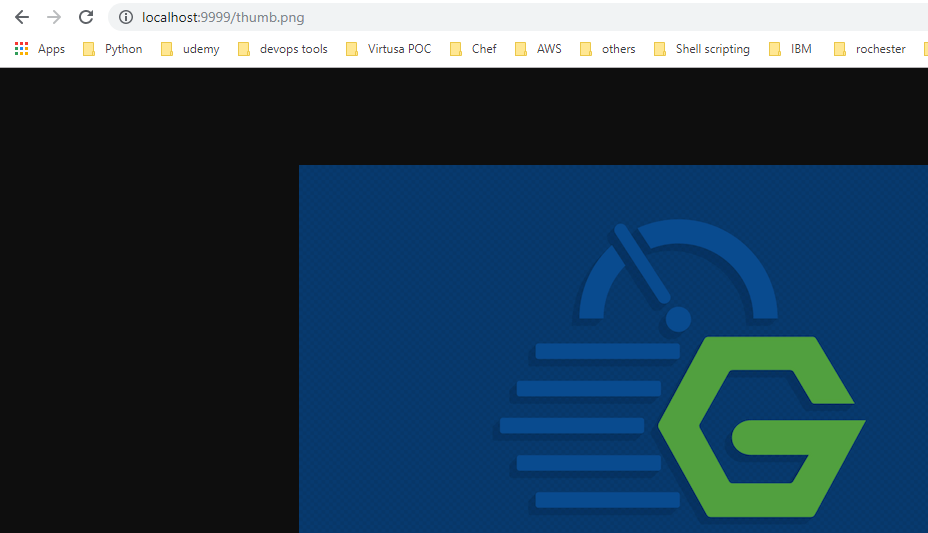
PHP 7.3.6 Development Server started at Thu Jun 27 16:12:01 2019

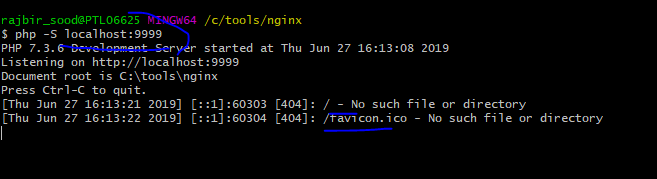
Listening on http://localhost:9999

Document root is C:\tools\nginx

Press Ctrl-C to quit.

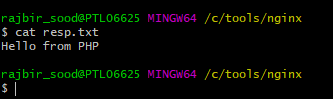


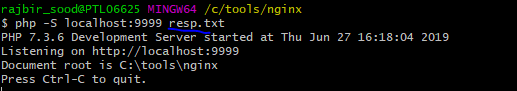


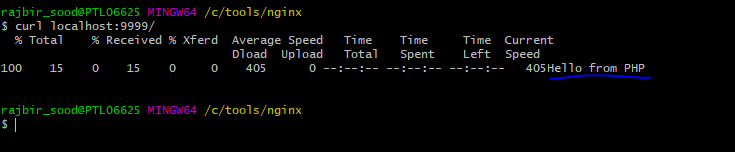




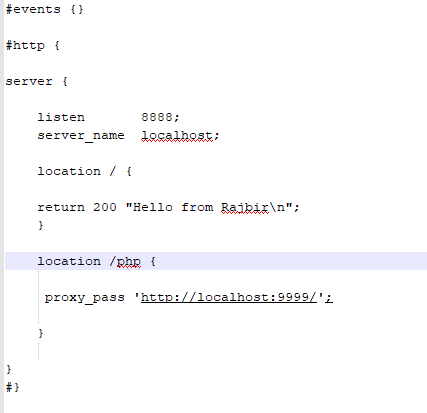
Alternately pass the txt file to run the PHP server

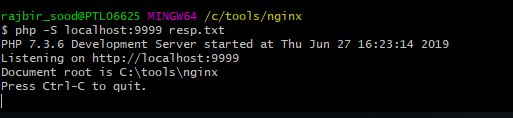


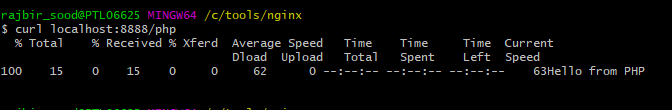




Now use reverse proxy





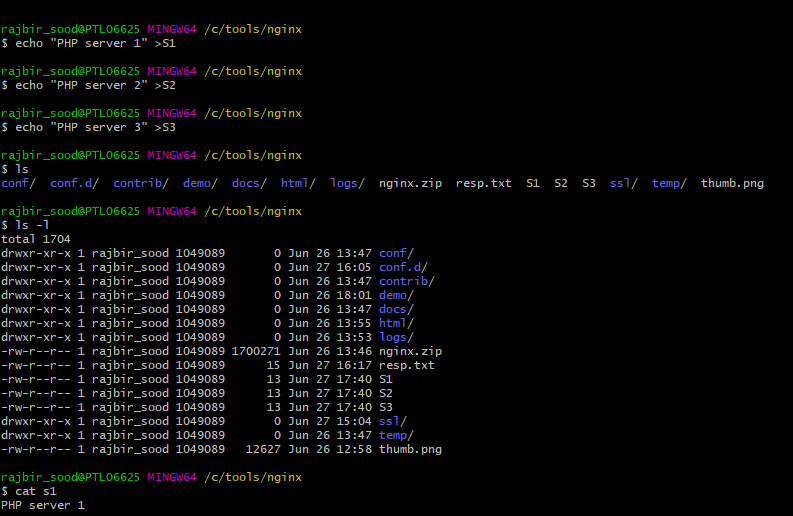


Request on our Nginx server.

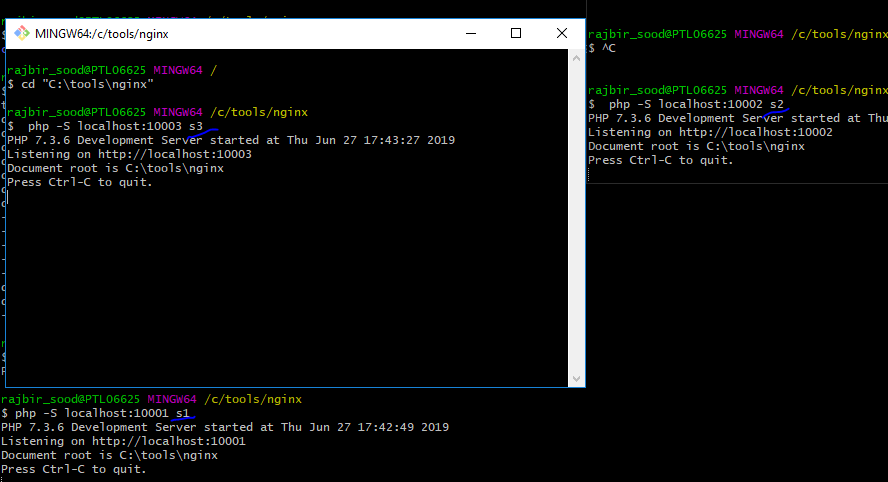
Hello from PHP server.

So a response clearly from I appear to be server but accessed and reverse proxied via a were engine

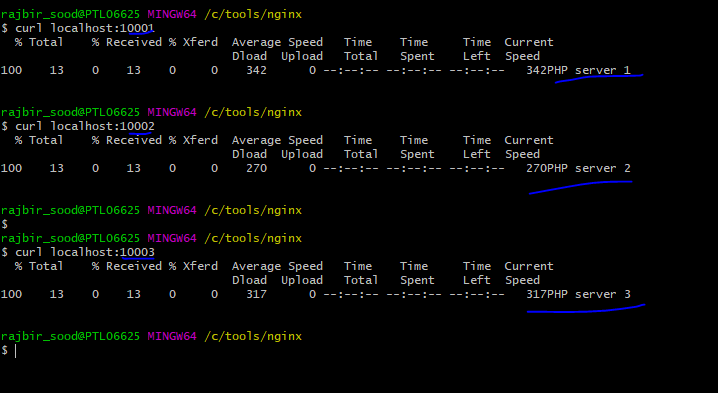
**LoadBalancer**



**Create 3 terminal windows and run**

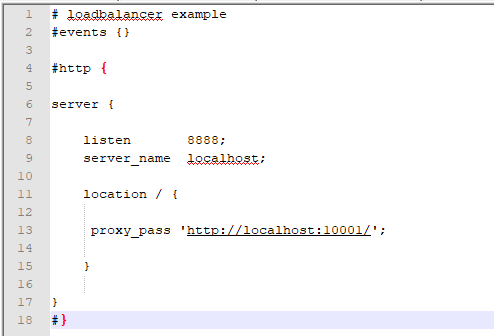


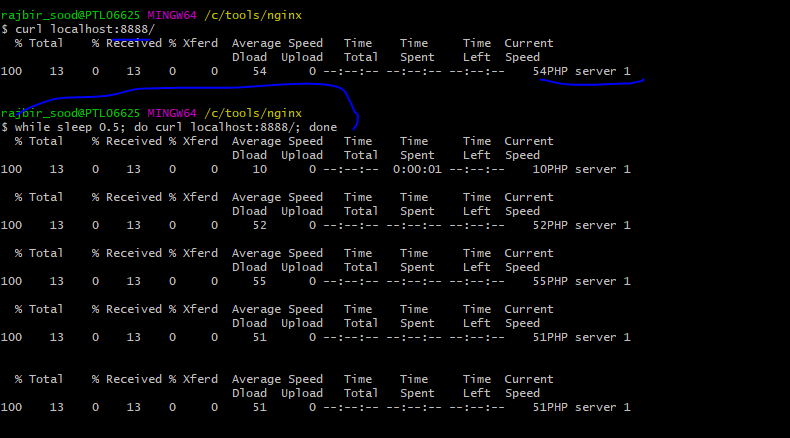
**Test it**



**Create new conf file load-balancer.conf**

**First test for one server**





**Now config actual load-balancer**

**First create upstream**

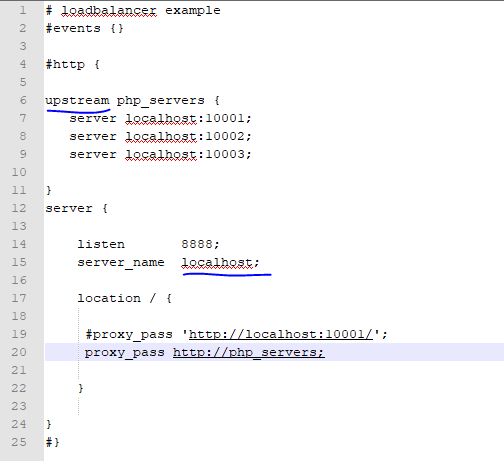
Let's implement the actual load balancing now.

First we have to create what's called an upstream.

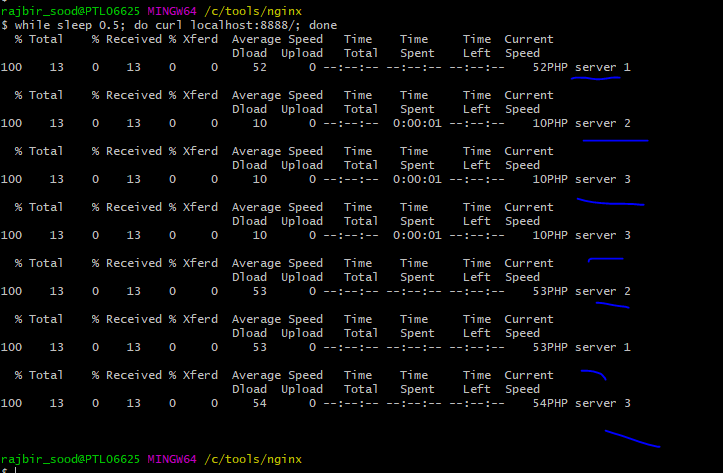
This is a context or block in nginx that group several servers with the ability to add some options

to the Upstream.

Think of it as a named collection of servers that share somewhere the commonality.



Test it



**Loadbalancer options**

**Sticky sesson**

**If one server gets down, we switched to other servers**

