Nginx

1-Installation

2-nginx start serving from (or entry point) /etc/nginx/nginx.conf

🡪It contains all configurations

3-include /usr/share/nginx/modules/\*.conf;

🡪It contains additional configuration file.

4-/usr/share/nginx/html -🡪centos and ubuntu 🡪/var/www/html/index.nginx-debian.html

🡪It contains default index.html page

5-Now configure default nginx.conf

@@@ include /etc/nginx/sites-enabled/\*;

Create new file within below location --------

/etc/nginx/sites-enabled/anyname.conf

Vi naveen.conf

Here define block

server {

listen 80 default\_server;

root /var/www/naveen; #index file location

# /var/www/🡪here we clone real code for website include index file then it will work as server

server\_name \_; #hre we define specific server name.(naveen.com)

index index.html index.htm;

#If I want to serve the (prefix match,it will return for / )

location /naveen {

return 200 ‘Hello from nginx “/naveen” location. ’; #restart nginx and getting response

}

#for exact match, it will return only /naveen

location = /naveen {

return 200 ‘Hello from nginx “/naveen” location. ’; #restart nginx and getting response

}

#regular match,REGEX now it will return /naveen1 or any no. between 0-9(case sensitive)

location ~ /naveen [0-9]{

return 200 ‘Hello from nginx “/naveen” location. ’; #restart nginx and getting response

}

#regular match,REGEX now it will return on /naveen1 , Naveen1 or any no. between 0-9(case insensitive)

location ~\* /naveen [0-9]{

return 200 ‘Hello from nginx “/naveen” location. ’; #restart nginx and getting response

}

==============================================================================

How to return variable

server{

listen 80;  
 server\_name ip/hostname;

root /path of index file;

location /inspect {

return 200 “$host\n$uri\n$args”; #ip/inspect it will return->host name/ip and /inspect i.e.uri and for arg ip/inspect?name=naveen it will return host name/ip and /inspect name=naveen

}

}

#REDIRECTS and REWRITE

Redirect rules are used for old paths that you'd like to redirect to new ones. AND Rewrite rule does not change the original URL; it simply serves the content of the rule destination at the original path

server{

listen 80;  
 server\_name ip/hostname;

root /path of index file;

rewrite ^/user/\w+ /great; #w+ more than 1 char

location /great {

return 200 “hello user”; #like ip/user/Naveen it will return hello user it means return from /great

}

}

#TRY FILES

server{

listen 80;  
 server\_name ip/hostname;

root /path of index file;

try-files /abc.png /great; #it will firstly serve abc.png if not found then serve /great and return hello suer

location /great {

return 200 “hello user”

}

location / { #uri is that eg.ip:80/app here app is uri if app is not found then $uri/ =404;

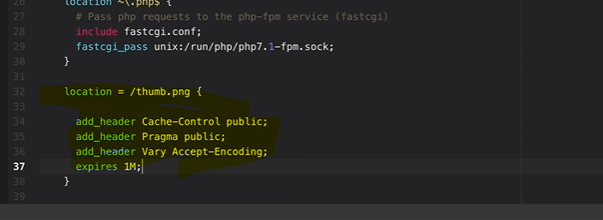
try\_files $uri $uri/ =404;

}

}

# Headers and Expires

Browsers explicitly know to keep a copy parts of your website, you can set content-expiry headers within nginx. This means that subsequent page loads or revisits within the expiry time,



Example->

location ~\* \.(?:ico|css|js|gif|jpe?g|png)$ {

expires 30d;

add\_header Pragma public;

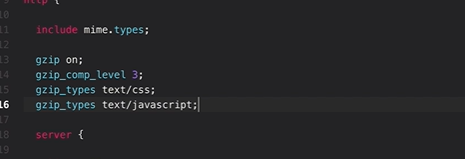
add\_header Cache-Control "public";

}

compressed resores with gzip

Enabling gzip compression allows the server to compress HTTP responses before sending them to clients. This reduces the size of data transferred over the network, resulting in faster page load times and reduced bandwidth usage

gzip on; #It will applied on entire server block.



server {

listen 80 default\_server;

}

HTTP2

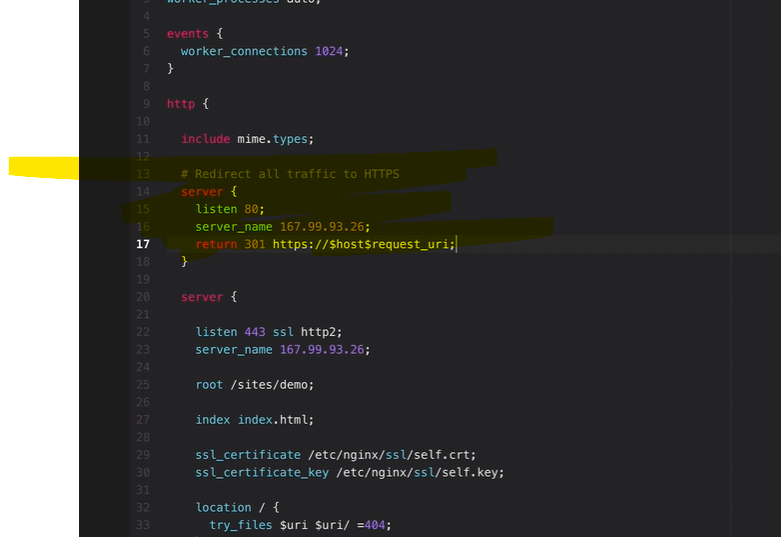
Improving the speed and security of client-server interactions across the Internet

All types of files like css, js,html convert into single

| **Feature** | **HTTP/1.1** | **HTTP/2** |
| --- | --- | --- |
| Multiplexing | No | Yes |
| Header Compression | No | Yes |
| Binary Protocol | No (Textual) | Yes (Binary) |
| Server Push | No | Yes |
| Stream Prioritization | No | Yes |
| TLS Encryption | Optional (HTTPS) | Highly Recommended (HTTPS) |

HTTPS

Route the all http traffic to https so that all the traffic serve by https



DHPARAM

This is a method of securely exchanging cryptographic keys over a public channel and is a fundamental part of setting up HTTPS connections for secure communications over the internet.

Note-- Vi naveen.conf

We can create multiple .conf file with multiple subdomains of main domain and every subdomain contains different index page with code

NOW WE WANT TO AUTHENTIACTE FOR LOGIN🡪

server {

listen 80 default\_server;

root /var/www/naveen; #index file location

/var/www/ -🡪here we clone real code for website include index file then it will work as server

# server\_name \_; #hre we define specific server name.(naveen.com)

server\_name kumar.naveen.com;

auth\_basic "under kumar.naveen.com ";

auth\_basic\_user\_file /etc/nginx/.htpasswd;

##for this create user and password and stored in/etc/nginx/.htpasswd;

#sudo sh -c "echo -n 'kumar:' >>/etc/nginx/.htpasswd"

#sudo sh -c "openssl passwd -arp1 >>/etc/nginx/.htpasswd"

#sh used for write in file

index index.html index.htm;

location / { #uri is that eg.ip:80/app here app is uri if app is not found then $uri/ =404;

try\_files $uri $uri/ =404;

}

}

NOW WE WANT naveen.com serve without password and naveen.com/app serve with password

server {

listen 80 default\_server;

root /var/www/naveen; #index file location

/var/www/ -🡪here we clone real code for website include index file then it will work as server

# server\_name \_; #hre we define specific server name.(naveen.com)

server\_name kumar.naveen.com;

auth\_basic "under kumar.naveen.com ";

auth\_basic\_user\_file /etc/nginx/.htpasswd;

index index.html index.htm;

location / {

auth\_basic off;

try\_files $uri $uri/ =404;

#uri is that eg.ip:80/app here app is uri if app is not found then $uri/ =404;

}

location /admin {

auth\_basic off;

try\_files $uri $uri/ =404;

} #for /admin , this directory must present in /var/www/naveen with index file

}

PROBLEMS:-

security

--open only specific ports

--need to access using port 80/443

TLS

data transfer between application to client need to encrypt

scalability

when need to scale then port issue can to use same port

load balancing

TLS configuration required again

We used reverse proxy

U🡪NR🡪App/servers

https/ip/ 🡪 NGINX-----------http-🡪app/server

TLS configuration on NGINX and tls termination

# Define an upstream block with the backend servers

upstream backend {

server localhost:8080; #replaced with self-domain (backend1.example.com:8080;)

}

# Nginx server block

server {

listen 80;

root /var/www/naveen; #index file location

server\_name api.naveen.com;

# **api.naveen.com first** created at /etc/nginx/conf.d like api.naveen.com.conf

location / {

proxy\_pass http:// backend;

}

}

LOAD BALANCING BY NGINX REVERSE PROXY

**NODE JS APPLICATION**

---🡪

# Define an upstream block with the backend servers

upstream backend {

server localhost:8000; #ec2-1 #replaced with selfdomain (backend1.example.com)

server localhost:8001; #ec2-2#following round robin algorithm for LB DEFAULT

}

# Nginx server block

server {

listen 80;

root /var/www/naveen; #index file location

server\_name api.naveen.com;

# **api.naveen.com first** created at /etc/nginx/conf.d like api.naveen.com.conf

location / {

proxy\_pass http:// backend;

}

}

**We want distribute traffic based om weighted i.e.canary deployment**

# Define an upstream block with the backend servers

upstream backend {

server localhost:8000 weight=3;

#used weight modifier for same weight=3 it means first server serve 3 request then 1 second server

#ec2-1 #replaced with selfdomain (backend1.example.com)

server localhost:8001;

#ec2-2#following round robin algorithm for LB DEFAULT

}

# Nginx server block

server {

listen 80;

root /var/www/naveen; #index file location

server\_name api.naveen.com;

# **api.naveen.com first** created at /etc/nginx/conf.d like api.naveen.com.conf

location / {

proxy\_pass http:// backend;

}

}

* **Sticky sessions** focus on maintaining session consistency by ensuring a user's requests are always handled by the same server.
* **Least connections** aim to improve load distribution based on the current workload of each server, without necessarily considering session persistence.

**BACKUP SERVER REQUIREMENT**

# Define an upstream block with the backend servers

upstream backend {

server localhost:8000;

server localhost:8001 backup; #used backup modifier for same.

}

# Nginx server block

server {

listen 80;

root /var/www/naveen; #index file location

server\_name api.naveen.com;

# **api.naveen.com first** created at /etc/nginx/conf.d like api.naveen.com.conf

location / {

proxy\_pass http:// backend;

}

}

**NOW SETUP SSL CERTIFICATE**

**Need certificate and certificate key**

listen 443 ssl;

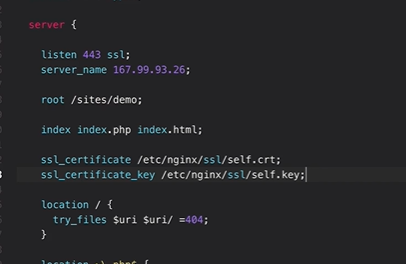
server\_name api.naveen.com;

root /var/www/html

# SSL/TLS configuration (replace with your SSL certificate details)

ssl\_certificate /path/to/your/certificate.crt;

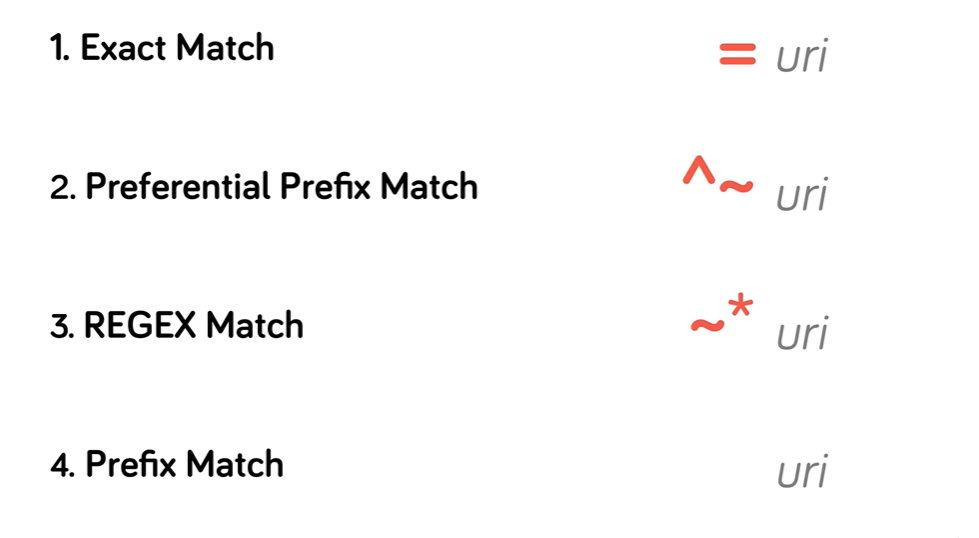
ssl\_certificate\_key /path/to/your/private-key.key;

****

Use cert bot for certification, if we created certificate with certbot then it will automatically update the nginx configuration.

**->**we can use cronjob for renew the certificate.

**LAST**

**Priority for location block  
**

**URI (Uniform Resource Identifier) is a generic identifier for resources. URL (Uniform Resource Locator) specifies the resource's location**

Variable defined with if condition

