# NGINX Installation

1. Update your system first before you do anything else. Run the following commands:

|  |
| --- |
| sudo apt-get update && apt-get upgrade |

1. Install the required package if you don’t have it already installed

|  |
| --- |
| sudo apt-get install software-properties-common |

1. Ubuntu comes with Nginx packages by default, so use the below command to install it directly.

|  |
| --- |
| sudo apt-get install nginx -y |

1. After Installation.just check the status of the nginx service is activated or not using below commands.

|  |
| --- |
| Ps -aux | grep nginx |

# Simple HTTP Configuration

By default, Nginx created a configuration file called nginx.conf for http redirection under /etc/nginx directory. We can confirm and modify the configuration if required. This will also refer to the default.conf file under /etc/nginx/conf.d directory for http redirection.

|  |
| --- |
| http {  ##  # Basic Settings  ##  sendfile on;  tcp\_nopush on;  tcp\_nodelay on;  keepalive\_timeout 65;  types\_hash\_max\_size 2048;  # server\_tokens off;  # server\_names\_hash\_bucket\_size 64;  # server\_name\_in\_redirect off;  include /etc/nginx/mime.types;  default\_type application/octet-stream;  ##  # SSL Settings  ##  ssl\_protocols TLSv1 TLSv1.1 TLSv1.2; # Dropping SSLv3, ref: POODLE  ssl\_prefer\_server\_ciphers on;  ##  # Logging Settings  ##  access\_log /var/log/nginx/access.log;  error\_log /var/log/nginx/error.log;  ##  # Gzip Settings  ##  gzip on;  # gzip\_vary on;  # gzip\_proxied any;  # gzip\_comp\_level 6;  # gzip\_buffers 16 8k;  # gzip\_http\_version 1.1;  # gzip\_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/javascript;  ##  # Virtual Host Configs  ##  include /etc/nginx/conf.d/\*.conf;  include /etc/nginx/sites-enabled/\*;  } |

# HTTP & HTTPS Configuration

Create a config file named default.conf under /etc/nginx/conf.d directory and paste the below contents over there.

|  |
| --- |
| server {  listen 80;  server\_name <domain-name>;  return 301 https://$server\_name$request\_uri;  }  server {  listen 443 ssl;  server\_name  **<domain-name>**;  ssl\_certificate /etc/ssl/**<pem file name>**;  ssl\_certificate\_key /etc/ssl/**<Key file name>**;  location / {  proxy\_pass http:**//<App-server-ip:port>**/;  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  #proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_set\_header X-Forwarded-For $remote\_addr;  }  location /**<prefix**>/ {  proxy\_pass http://**<App-server-ip:port>**/;  proxy\_set\_header Host $host;  proxy\_set\_header X-Forwarded-For $remote\_addr;  }  } |

Once done, restart the nginx to see the changes.

|  |
| --- |
| sudo systemctl restart nginx |

# Multiple Backend Applications

For multiple backend applications, we should have to specify the backend application domain name or URL in the default.conf file as per below contents.

Server{} block is the place where we need to configure all the redirection details. We can create multiple server blocks in each file.

|  |
| --- |
| server {  server\_name <App1-client.domain.com>;  proxy\_set\_header X-Real-IP $remote\_addr; proxy\_set\_header Host $host; proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_pass http://<App1-IP>:80;  }  server {  server\_name <App2-client.domain.com>;  proxy\_set\_header X-Real-IP $remote\_addr; proxy\_set\_header Host $host; proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_pass http://<App2-IP>:80;  } |

# URL Rewrite to handle the redirects

Nginx used Rewrite and Return directives to handle the URL redirects. Rewrite or Return collect the client requested URL and redirect the same request to the user specified URL (Kind of load balancing).

It can be written and Server and Location block as per our app requirement.

|  |
| --- |
| **Return**  server {  listen 80;  server\_name <domain-which-client-request>;  return 301 $scheme://<domain-which-proxy-redirect>$request\_uri;  }  **Rewrite**  server {  ...  ...  location = /<path-prefix>  {  rewrite ^<path-prefix>?$ /somePage.html break;  }  ...  ...  } |

We can add multiple path references in the rewrite to proxy to redirect the correct path.

# Access Log Configuration

NGINX writes information about client requests in the access log right after the request is processed.

By default, log path configuration has been configured automatically by Nginx inside nginx.conf file under /etc/nginx directory. Refer this file to get the access log path information and modify the path location if needed.

Default path is access\_log /var/log/nginx/access.log;

|  |
| --- |
| http {  log\_format compression '$remote\_addr - $remote\_user [$time\_local] '  '"$request" $status $body\_bytes\_sent '  '"$http\_referer" "$http\_user\_agent" "$gzip\_ratio"';  access\_log <path-to-log-file> compression;  ...    } |

# Rate Limit

Nginx used Rate limit to limit no of requests and connections hitting on the proxy.

Use the **limit\_conn** directive to apply the limit within the location {}, server {}, or http {} context.

|  |
| --- |
| Rate-Connection  http {  limit\_conn\_zone $server\_name zone=servers:10m;  server {  limit\_conn servers 1000;  }  }  Rate-Request  http {  limit\_req\_zone $binary\_remote\_addr zone=one:10m rate=1r/s;  } |

# IP Whitelist

IP can be whitelisted in our default.conf configuration file. Refer the same below.

We can allow mu;tiple IP’s as well, If we directly specify the IP in our file, then it would only allow that IP request for our domain rest all will be denied.

|  |
| --- |
| Main-Domain  http{  allow <IP>;  deny all;  }  server{  allow <IP>;  deny all;  }  location / {  allow <IP>;  deny all;  }  Sub-Domain  server {  server <anything.example.com>;  allow <IP>;  deny all;  } |

# Load Balancing

Load Balancing can be easily configured in default.conf file under /etc/nginx/conf.d directory.

Proxy\_pass is the flag which is simply used to redirect the request.

|  |
| --- |
| http {  upstream <project-name> {  server <IP1:port> weight=3;  server <IP2:port>;  server <IP3:port>;  server <IP4:port>;  }  server {  listen 80;  server\_name <domain-name>;  location / {  **proxy\_pass http://<project-name>;**  }  }  } |

# Serve Static Pages

Static website serving has been configured in our default.conf file as the same as below contents.

|  |
| --- |
| server {  listen 80 default\_server;  root /var/www/<domain-name>;  index index.html;  server\_name <domain-name>;  location / {  try\_files $uri $uri/;  }  } |

# SSL for multiple domains

If we want to enforce SSL for multiple domains, then we can configure ssl certificates into the server block for each domain as per below contents.

|  |
| --- |
| server {  listen \*:443 ssl;  server\_name domain1.com;  ssl\_certificate /path/to/domain1.crt;  ssl\_certificate\_key /path/to/domain1.key;  return 301 https://www.domain1.com$request\_uri;  }  server {  listen \*:443 ssl;  server\_name domain2.com www.domain2.com;  ssl\_certificate /path/to/domain2.crt;  ssl\_certificate\_key /path/to/domain2.key;  return 301 https://www.domain1.com$request\_uri;  } |