# Nginx task

# 1-redirect abc.com ->xyz.com

server {

listen 80;

server\_name abc.com;

return 301 https://xyz.com$request\_uri;

}

OR

server {

listen 80;

server\_name abc.com;

location / {

proxy\_pass http://xyz.com$request\_uri;

}

}

OR

server {

listen 80;

server\_name abc.com;

rewrite ^ https://xyz.com$request\_uri permanent;

}

# 2-abc.com/page1(only for page1 otherwise on abc) -->xyz.com/page5

server {

listen 80;

server\_name abc.com;

location = /page1 {

rewrite ^ https://xyz.com/page5 permanent;

}

}

OR

server {

listen 80;

server\_name abc.com;

if ($uri = "/page1") {

rewrite ^ https://xyz.com/page5 permanent;

}

}

# 3- https://hirakranibyebye.com/en/index.html -we don’t want access

# index.html

server {

listen 80;

server\_name your\_domain.com;

location = /index.html {

deny all;

return 403;

}

}

server {

listen 80;

server\_name your\_domain.com;

root /path/to/your/website/root;

location / {

try\_files $uri $uri/ /index.html;

}

error\_page 403 404 = /error.html;

location = /error.html {

root /path/to/error/page/directory;

internal;

}

}

# 4-cache enable and disable server side

http {

# Define cache path and settings

proxy\_cache\_path /path/to/cache levels=1:2 keys\_zone=my\_cache:10m max\_size=10g inactive=60m use\_temp\_path=off;

server {

listen 80;

server\_name example.com;

# Enable caching for specific locations

location /cached {

proxy\_cache my\_cache;

proxy\_cache\_valid 200 302 10m;

proxy\_cache\_valid 404 1m;

proxy\_pass http://backend\_server;

}

# Disable caching for specific locations

location /uncached {

proxy\_cache\_bypass $http\_cache\_control;

proxy\_no\_cache $http\_pragma $http\_authorization;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header Host $http\_host;

proxy\_redirect off;

proxy\_pass http://backend\_server;

}

}

}

# 5-abc.com on access want to display frontend and abc.com/v1 ,v2,v3 -->access backend (v1,v2,v3 are backbends)

server {

listen 80;

server\_name abc.com;

root /path/to/frontend;

index index.html;

# Frontend

location / {

try\_files $uri $uri/ /index.html;

}

# Backend v1

location /v1/ {

proxy\_pass http://backend\_server\_v1/;

}

# Backend v2

location /v2/ {

proxy\_pass http://backend\_server\_v2/;

}

# Backend v3

location /v3/ {

proxy\_pass http://backend\_server\_v3/;

}

}

upstream backend\_server\_v1 {

server localhost:3001;

}

upstream backend\_server\_v2 {

server localhost:3002;

}

upstream backend\_server\_v3 {

server localhost:3003;

}

# 6-only one backend but have 3 domains and 3 different types of frontend

# Define the upstream backend server

upstream backend {

server backend-server-ip:port;

}

# Configuration for domain1.com

server {

listen 80;

server\_name domain1.com www.domain1.com;

location / {

proxy\_pass http://backend;

}

}

# Configuration for domain2.com

server {

listen 80;

server\_name domain2.com www.domain2.com;

location / {

proxy\_pass http://backend;

}

}

# Configuration for domain3.com

server {

listen 80;

server\_name domain3.com www.domain3.com;

location / {

proxy\_pass http://backend;

}

}

===========================frontend===============

server {

listen 80;

server\_name domain3.com www.domain3.com;

# Define the root directory for frontend type 3

root /path/to/frontend3;

location / {

try\_files $uri $uri/ @backend;

}

# Proxy requests not found in the frontend to the backend

location @backend {

proxy\_pass http://backend;

}

}

# 7-want to serve only files inside frontend otherwise serve error page

server {

listen 80;

server\_name your\_domain.com;

root /path/to/your/frontend;

index index.html;

location / {

try\_files $uri $uri/ /error.html;

}

error\_page 404 /error.html;

location = /error.html {

root /path/to/your/frontend;

}

}

# 8-if we want to run app on ip and domain and both, 3 cases

===============================ip only===========================

server {

listen your\_ip\_address:80; only port no.

server\_name \_;

root /path/to/your/app;

index index.html;

location / {

try\_files $uri $uri/ /index.html;

}

}

====================================domain only================

server {

listen 80;

server\_name your\_domain.com www.your\_domain.com;

root /path/to/your/app;

index index.html;

location / {

try\_files $uri $uri/ /index.html;

}

}

================================both ip and domain=================

server {

listen your\_ip\_address:80;

server\_name your\_domain.com www.your\_domain.com;

root /path/to/your/app;

index index.html;

location / {

try\_files $uri $uri/ /index.html;

}

}

# 9-cors enabled and disabled(Cross-Origin Resource Sharing)

By default disabled the cors

server {

listen 80;

server\_name example.com;

location / {

# Allow cross-origin requests from any origin

add\_header 'Access-Control-Allow-Origin' '\*' always;

# Allow specific methods (e.g., GET, POST) and headers

add\_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS' always;

add\_header 'Access-Control-Allow-Headers' 'Authorization, Content-Type' always;

}

}

**====================================================================================================================================================**

**deferred** is used to handle requests only when necessary to prevent overriding more specific server blocks, while **default\_server** is used to catch requests that do not match any other server block.

server {

listen 80 deferred;

server\_name example.com;

}

# **Preventing Overrides**: Using **deferred** helps prevent less specific server blocks from overriding more specific ones, ensuring that requests are routed according to the intended configuration.

server {

listen 80 default\_server;

server\_name \_;

...

}

# default server to handle requests for a specific port when no other server block matches the request. If a request comes in and none of the other server blocks on the same port match the requested hostname, the default server block will handle the request.

# The **\_** symbol in **server\_name** acts as a wildcard, matching any hostname.

## **deferred** is used to prioritize more specific server blocks over less specific ones, ensuring that requests are handled appropriately without overriding. On the other hand, **default\_server** is used as a fallback option to catch requests that do not match any specific server block, ensuring that no requests are left unhandled