Ubuntu Linux

- 設定主機
- 設定主機 2025 (Ubuntu24.04筆記)
- 基本指令
- SSL
- Docker
- clamav
- SSH登入免密碼
- bash
- VPS降速
- Ubuntu 22.04 VNC
- MySQL
- 使用 Fail2Ban 自動阻擋攻擊者
- 網頁版的 htop Glances
- 使用 Monit 監控資源
- 一些 log
- Ubuntu 24.04效能調校
- pm2移機
- 在 Windows 11安裝 WLS
- □□ Redis 快速檢查指令清單
- 用 Docker打造最佳效能的 Wordpress環境

設定主機

切換成 **root**角色

```
sudo su
```

> 有些 vps預設會直接用 root，就不用切換。我是習慣全用 root 去執行操作，主機只有自己會管理，有時遇到權限的問題真的蠻雷的 XD

修改時區

```
dpkg-reconfigure tzdata
```

修改語系

```
vi /etc/locale.gen
把 zh_TW.UTF-8 mark打開
locale-gen
```

更新程式並且重啟主機

```
apt update
apt dist-upgrade
reboot
```

重啟用，登入後再切到 **root** 角色

```
sudo su
```

安裝伺服器

```
apt install nginx php-fpm php-zip php-gd php-mbstring php-gd php-xml php-mysql mysql-server mysql-client certbot python3-certbot-nginx
```

> Drupal或WordPress裝完後可以再檢查看看有沒有漏掉哪個php程式，再補齊安裝即可

> 不要用apt-install php，這樣會安裝apache2

設定MySQL

設定mysql密碼

以root進入mysql

```
mysql -u root
```

更改密碼

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH caching_sha2_password BY 'yourpassword'; exit;
```

> yourpassword記得改成你自己的密碼

> 如果你的資料庫有設定外連，那可能就要避免直接用root，不然就是務必把防火牆的限定做好

調整conf檔

```
[client]
default-character-set = utf8mb4

[mysqld]
character-set-server = utf8mb4
max_connections = 600
innodb_read_io_threads = 24
innodb_write_io_threads = 24
expire_logs_days = 7
max_binlog_size = 1G
```

> 加入後後重啟server，binlog_size務必要設定，log檔可能會被你的空間佔滿

在 ~/.my.cnf 加入密碼

```
[client]
password = yourpassword
```

MySQL基本的管理指令

建立資料庫

```
mysqladmin -u root create site
```

刪除資料庫

```
mysqladmin -u root drop site
```

匯出資料庫

```
mysqldump -u root site > site.sql
```

匯入資料庫

```
mysql -u root site < site.sql
```

> 學會這幾招基本上就差不多可以不用裝 phpMyAdmin XD

啟用及設定伺服器防火牆

```
ufw enable
ufw allow from xxx.xxx.xxx.xxx (填上自己連入的 IP)
ufw allow 80/tcp
ufw allow 'Nginx HTTP'
ufw allow 'Nginx HTTPS'
```

加入不同版本的 **php**來源

```
add-apt-repository ppa:ondrej/php
aptitude update
```

設定主機 2025 (Ubuntu24.04 筆記)

切換成 **root**角色

```
sudo su
```

> 有些 vps預設會直接用 root，就不用切換。我是習慣全用 root 去執行操作，主機只有自己會管理，有時遇到權限的問題真的蠻雷的 XD

修改時區

```
dpkg-reconfigure tzdata
```

修改語系

```
vi /etc/locale.gen
把 zh_TW.UTF-8 mark打開
locale-gen
```

更新程式及更改主機名稱，然後重啟主機

```
apt update
apt dist-upgrade
vi /etc/hostname
reboot
```

重啟用，登入後再切到 **root** 角色

```
sudo su
```

安裝伺服器

```
apt install nginx php-fpm php-gd php-mbstring php-gd php-xml php-mysql mysql-server mysql-client certbot python3-certbot-nginx
```

> Drupal或WordPress裝完後可以再檢查看看有沒有漏掉哪個php程式，再補齊安裝即可

> 不要用apt-install php，這樣會安裝apache2

設定MySQL

設定mysql密碼

以root進入mysql

```
mysql -u root
```

更改密碼

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH caching_sha2_password BY 'yourpassword'; exit;
```

> yourpassword記得改成你自己的密碼

> 如果你的資料庫有設定外連，那可能就要避免直接用root，不然就是務必把防火牆的限定做好

調整conf檔

```
[client]
default-character-set = utf8mb4

[mysqld]
character-set-server = utf8mb4
max_connections = 600
innodb_read_io_threads = 24
innodb_write_io_threads = 24
expire_logs_days = 7
max_binlog_size = 1G
```

> 加入後後重啟server，binlog_size務必要設定，log檔可能會被你的空間佔滿

在~/.my.cnf加入密碼

```
[client]
password = yourpassword
```

MySQL基本的管理指令

建立資料庫

```
mysqladmin -u root create site
```

刪除資料庫

```
mysqladmin -u root drop site
```

匯出資料庫

```
mysqldump -u root site > site.sql
```

匯入資料庫

```
mysql -u root site < site.sql
```

> 學會這幾招基本上就差不多可以不用裝phpMyAdmin XD

啟用及設定伺服器防火牆

```
ufw enable
ufw allow from xxx.xxx.xxx.xxx (填上自己連入的IP)
ufw allow 'Nginx HTTP'
ufw allow 'Nginx HTTPS'
```

加入不同版本的**php**來源

```
add-apt-repository ppa:ondrej/php
aptitude update
```

```
apt install php7.4-fpm php7.4-gd php7.4-mbstring php7.4-gd php7.4-xml php7.4-mysql
apt install php5.6-fpm php5.6-gd php7.4-mbstring php5.6-gd php5.6-xml php5.6-mysql
```

□□ Docker

```
sudo apt install -y ca-certificates curl gnupg lsb-release
```

```
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

□□ Portainer

```
docker pull portainer/portainer-ce:latest
docker run -d -p 9000:9000 --restart always -v /var/run/docker.sock:/var/run/docker.sock portainer/portainer-
ce:latest
```

Nginx conf

```
server {
    listen 80;
    server_name docker.yoursite.com.tw;

    location / {
        proxy_pass http://127.0.0.1:9000;
        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-Proto $scheme;
```

```
    }
}
```

安裝舊版本的 Mysql

```
version: '3.8'
services:
  mysql:
    image: mysql:5.7
    container_name: my_mysql57
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: yourpassword    # 請換成你自己的密碼
      TZ: Asia/Taipei
    ports:
      - "43306:3306"
    volumes:
      - ./mysql57_data:/var/lib/mysql

  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: my_phpmyadmin
    restart: always
    ports:
      - "48081:80"
    environment:
      - PMA_ARBITRARY=1
      - PMA_HOST=mysql
      - PMA_USER=root
      - PMA_PASSWORD=yourpassword    # 要與上面一致
      - UPLOAD_LIMIT=128M    # 要與上面一致
    depends_on:
      - mysql
```

Nginx 設定檔 (phpmyadmin) (他會直接登入, 使用完記得移掉)

```
server {
    listen 80;
    server_name pma2025.yoursite.com.tw;
```

```
    client_max_body_size 128M;  # <<< 這行最重要, 數字請依需求調整

    location / {
        proxy_pass http://127.0.0.1:48081;
        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-Proto $scheme;
    }
}
```

Drupal 6 修改 port

```
$db_url = 'mysqli://root:yourpassword@127.0.0.1:43306/haccp';
$db_prefix = 'haccp_';
```

基本指令

```
ls
```

列出當前目錄的內容

```
ls -la
```

列出當前目錄的內容包含隱藏目錄

> Linux隱藏目錄或檔案的作法，是在檔名前面加 .

```
ls ~/
```

列出自己的家目錄

> 一般使用者的家目錄位於 /home/；最高權限管理者家目錄位於 /root/

```
mkdir test
```

建立目錄

```
mkdir -p test/a
```

於目錄下建立目錄

```
cp a.php b.php
```

複製檔案

> 一般最常用在當我需要再新增網站，copy nginx 設定檔 cp /etc/nginx/sites-enabled/site1 /etc/nginx/sites-enabled/site2

```
cp -r ~/abc /var/www/
```

移動目錄，把家目錄中的 abc 移到 /var/www/

> 分享一個小技巧，按鍵盤的tab，可以幫你索引檔案或目錄，例如輸入/var/w 按鍵盤 tab 就會自動跑出/var/www/

```
rm xxx.txt
```

刪除檔案

```
rm -r xxx
```

刪除目錄

> 在管理linux檔案管理有一件非常重要的事情要交待，linux沒有資源回收桶，檔案刪了就沒了

> 如果你不小心下了這個指令，整個系統就GG了 rm -rf /

```
mv ~/abc /var/www/
```

搬動檔案，相當於剪下貼上

```
tar zcf abc.tar.gz abc
```

壓縮檔案，把目錄abc壓縮成abc.tar.gz

```
tar zxvf abc.tar.gz
```

解壓縮檔案，把abc.tar.gz 解開

```
unzip abc.zip
```

如果要解開zip就用這個指令

```
cat /var/log/nginx/access.log
```

在螢幕列印出檔案的內容，一般也蠻常會運用在查詢log檔

```
cat /var/log/nginx/access.log | more
```

如果內容太多，可以在最後加上| more，就可以由上到下查看，查看模示按空白鍵會一直往下，按Ctrl + C，可退出

```
cat /var/log/nginx/access.log | less
```

與加上 more能達到大同小異的功能

```
zcat /var/log/nginx/access.log.10.gz
```

在查詢 log檔時，會蠻有機會用到 zcat去查看已被壓縮 .gz的文件

```
htop
```

查看主機的資訊，CPU、記憶體、程式運作情形

```
vi xxx.txt
nano xxx.txt
```

編輯檔案，我是習慣用 vi，nano比較易學

```
ssh name@ip
```

透過 ssh連線到主機

```
scp -r /var/www/ name@ip:/var/www/
```

透過 ssh複製檔案至另一台主機

```
rsync -av --delete /var/www/wp/wp-content/uploads/ root@ip:/var/www/wp/zfun/wp-content/uploads/
rsync -av --delete /var/www/wp/wp-content/plugins/ root@ip:/var/www/wp/zfun/wp-content/plugins/
```

透過 rsync同步檔案

> 免密碼登入遠端主機的方式如下
>
> 本機端執行 `ssh-keygen -t rsa`
>
> `cat .ssh/id_rsa.pub`
>
> 連線遠端主機 `.ssh/authorized_keys` 貼上 cat複製的內容
>
> 退出再重登一次，如果不用密碼即成功

```
crontab -e
```

設定排程，一般會用作在指定的時間幫你執行特定的程式

```
/etc/init.d/nginx reload
```

重新讀取 nginx設定檔，正常情況，當我有新增或修改 conf檔時，會先用 nginx -t 查看看有沒有問題

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

出現上情的內容確認沒問題後，再執行 reload

```
/etc/init.d/nginx restart
```

重新啟用服務

> 伺服器的服務管理，通常會針對他執行 reload, restart, stop, start 進行維護及管理
> Nginx: /etc/init.d/nginx
> PHP: /etc/init.d/php7.4-fpm
> MySQL: /etc/init.d/mysql
> 我會在網站的離峰時間用 cron執行 restart，以確保服務能順暢

一些 **shell script**筆記

> 以上最基本的指令及觀念務必要學會，其他的話 2023年 AI
> 開始普及使用後，在系統管理非常有幫助，建議大家可以多多利用

SSL

下面的筆記是把 DNS交給 Cloudflare, 然後再交由 Cloudflare設定 SSL

SSL/TLS 概觀頁面 選擇 完整嚴格
接著要 原始伺服器頁面 建立憑證 把 cert.pem 及 key.pem 存取在 /etc/ssl/ ; 勾選 認證的原點提取

```
 listen 443 ssl;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
    ssl_certificate         /etc/ssl/cert.pem;
    ssl_certificate_key     /etc/ssl/key.pem;
    ssl_client_certificate /etc/ssl/cloudflare.crt;
    ssl_verify_client on;
}
server {
    if ($host = site.com) {
        return 301 https://$host$request_uri;
    }



    server_name site.com;
    listen 80;
    return 404;
```

在原本的 conf檔的 server 的最後一個 "}" 前面貼上

在 Nginx安裝 SSL的方式可以參考:
https://www.digitalocean.com/community/tutorials/how-to-host-a-website-using-cloudflare-and-nginx-on-ubuntu-20-04

還有另一個方式是使用 Certbot

```
apt instll certbot
```

安裝

```
certbot -d site.com
```

新增網站

> 務必留意，要先在/etc/nginx/sites-enabled/建立site.com的設定檔後再執行，否則他會幫你寫在/etc/nginx/nginx.conf

```
certbot delete --cert-name site.com
```

刪除指定網站的SSL

> 經驗上Ubuntu 22.04之後的版本會自己執行寫入設定檔的動作，並且也會幫你定期更新憑證，不用自己寫cron跑certbot renew

Cloudflare想要在驗證SSL網站拿到A的方法到SSL/TLS邊緣憑證最低**TLS**版本選**1.2**；拿到A+的方法啟用**HTTP**強制安全傳輸**(HSTS)**

> 我後來用這個SSL的設定發生一件很奇怪的事情，某個subdomain的SSL怪怪的，Mac電腦無法連，但手機能連，就唯獨那個subdomain有問題，其他的沒有。我也有發現那段時間的流量減少，或許我在驗證沒做好?！我最後把Cloudflare代理全部移掉，改自己用的Certbot

Docker

```
apt install docker docker-compose
```

> Docker-compose 安裝也可以參考這裡用手安裝較新的版本
> https://hackmd.io/@jimmy801/docker_compose_install

安裝 docker主程式

```
docker pull portainer/portainer-ce:latest
docker run -d -p 9000:9000 --restart always -v /var/run/docker.sock:/var/run/docker.sock portainer/portainer-ce:latest
```

安裝 web版管理介面

```
server {
    server_name docker.site.com;

    location ~/ {
    # prevents 502 bad gateway error
    proxy_buffers 8 32k;
    proxy_buffer_size 64k;

    client_max_body_size 75M;

    # redirect all HTTP traffic to localhost:9000;
    proxy_pass http://localhost:9000;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    #proxy_set_header X-NginX-Proxy true;

    # enables WS support
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
```

```
    proxy_read_timeout 999999999;
}



}
```

> 一般而言，Docker要存取Docker建立出來的內容，都要透過IP + port 存取
> 由於我的VPS並無9000的port，因此必須透過Nginx Proxy 的方式存取Docker創建出來的本機port

安裝 WordPress

相同的原理，我如果想要創建WordPress的話，可以使用下面的Stacks

```
version: '3.1'

services:

  wordpress:
    image: wordpress:6.0.1-php7.4-apache
    restart: always
    ports:
      - 8001:80
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: exampleuser
      WORDPRESS_DB_PASSWORD: examplepass
      WORDPRESS_DB_NAME: exampledb
    volumes:
      - wordpress:/var/www/html

  db:
    image: mysql:5.7
    restart: always
    environment:
      MYSQL_DATABASE: exampledb
      MYSQL_USER: exampleuser
      MYSQL_PASSWORD: examplepass
      MYSQL_RANDOM_ROOT_PASSWORD: '1'
```

```
    volumes:
      - db:/var/lib/mysql

volumes:
  wordpress:
  db:
```

創建好後，再設定 Nginx設定檔

```
server {
    server_name demo1.site.com;

    location ~/ {
    # prevents 502 bad gateway error
    proxy_buffers 8 32k;
    proxy_buffer_size 64k;

    client_max_body_size 75M;

    # redirect all HTTP traffic to localhost:8088;
    proxy_pass http://localhost:8001;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    #proxy_set_header X-NginX-Proxy true;

    # enables WS support
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";

    proxy_read_timeout 999999999;
}



}
```

> 成功的話，輸入 https://demo1.site.com 就能連接到 Docker創建出來的網站

```
docker exec -i -t 8001-wordpress-1 bash
```

進入WordPress的容器

```
echo -e "file_uploads = On\nmemory_limit = 500M\nupload_max_filesize = 500M\npost_max_size = 500M\nmax_execution_time = 600" | tee -a /usr/local/etc/php/conf.d/uploads.ini;/etc/init.d/apache2 restart
```

本機的資料複製到docker容器目錄

```
docker cp /4TB/conoha2/www/wp/zfun/wp-content/uploads demo38-wordpress-1:/var/www/html/wp-content/
docker cp /4TB/conoha2/www/wp/site2/wp-content/themes demo38-wordpress-1:/var/www/html/wp-content/
docker cp /4TB/conoha2/www/wp/site2/wp-content/uploads demo38-wordpress-1:/var/www/html/wp-content/
```

Docker創建出來的WordPress預設由於有上傳限制，可以進入終端機模式，透過上面那一行解決
透過這個方式創建出來的WordPress會發現系統會判定是Apache2架出來的，還蠻有趣

每個WordPress網站都透過Docker的方式獨立區隔，有個好處就是如果中毒了只會影響到其中一個站XD
我曾經有個經驗，同事用了破解外掛被放毒，結果造成所有的WordPress架出來的全中獎，情況非常慘烈

clamav

安裝程式

```
apt install clamav clamav-daemon
```

定期執行: 更新病毒碼　　up_clamav.sh

```
service clamav-freshclam stop
freshclam
service clamav-freshclam start
```

定期執行　　：掃瞄指定目錄　　clamav_scan.sh

```
clamscan -r --bell -i /var/www/
```

SSH登入免密碼

本機端生成及複製金鑰

```
ssh-keygen -t rsa
```

複製金鑰

```
cat .ssh/id_rsa.pub
```

遠端主機 貼上複製的金鑰

```
vi .ssh/authorized_keys
```

成功的話在本地端輸入 ssh@ip 就不會問帳密

如果要登入 GCP主機，也是一樣要用類似的作法才能用本地端的 ssh登入

bash

備份所有的 mysql資料庫

```
#!/bin/bash

BACKUP_DIR="/root/backup/db"

if [ ! -d "$BACKUP_DIR" ]; then
    mkdir -p $BACKUP_DIR
fi

# 刪除目錄中的 .gz 檔案
rm -f $BACKUP_DIR/*.gz

DATABASES=$(mysql -u root -e "SHOW DATABASES;" | grep -Ev
"(Database|information_schema|performance_schema|mysql)")

for DB in $DATABASES; do
    BACKUP_FILE="$BACKUP_DIR/${DB}_backup_$(date +'%Y%m%d').sql"

    MYSQLDUMP_CMD="mysqldump -u root $DB > $BACKUP_FILE"

    echo "開始備份資料庫 $DB..."
    eval $MYSQLDUMP_CMD

    if [ $? -eq 0 ]; then
        echo "資料庫 $DB 備份成功 : $BACKUP_FILE"
        gzip $BACKUP_FILE
        echo "資料庫已壓縮 : $BACKUP_FILE.gz"
        # 顯示檔案大小, 單位 MB
        FILE_SIZE=$(du -m "$BACKUP_FILE.gz" | cut -f1)
        echo "壓縮後檔案大小 : $FILE_SIZE MB"
    else
        echo "資料庫 $DB 備份失敗 "
    fi
```

```
done
```

同步備份

```
#!/bin/bash
rsync -avz --delete --progress --exclude='cache/' --exclude='*demo*' root@ip:/etc/nginx/sites-enabled/
/4TB/conoha1/nginx/
```

備份指定網頁目錄

```
#!/bin/bash

# 來源目錄及目標目錄路徑
SRC_DIR="/var/www/"
DEST_DIR="/root/gcp/www/"

# 目錄列表
DIR_LIST=("web1" "web2" "web3")

# 備份及壓縮目錄語法
for DIR_NAME in "${DIR_LIST[@]}"; do
  FILE_NAME="${DEST_DIR}${DIR_NAME}.tar.gz"
  tar zcf "$FILE_NAME" -C "$SRC_DIR" "$DIR_NAME"
  echo "備份目錄 $DIR_NAME 為 $FILE_NAME 完成 "
done

echo "所有目錄備份完成 "
```

VPS降速

目前使用的VPS雖然標榜不限速，但他沒跟你說的就是會降速XD

由於降速也不會跟你講，你能做的就是知道他的限度在哪裡，不要去挑戰他

我發現只要我有大量用ssh傳送資料就很容易被鎖速度，因此要避免大量搬動檔案

Ubuntu 22.04 VNC

直接參考這裡

https://jackfrisht.medium.com/ubuntu-20-04-vnc%E5%AE%89%E8%A3%9D-b8ce6e7dec4a

先進行更新(可選)

```
sudo apt-get update
sudo apt-get upgrade
```

Ubuntu 22.04 默認使用 gdm3, 導致 VNC 工作異常(這邊案例就是設定檔設定完成沒有效果), 需切換到 lightdm

Ubuntu 16.04 默認使用 lightdm 因此一般不需要調整

```
sudo apt install lightdm  ## ubuntu 22.04 需用到
sudo dpkg-reconfigure lightdm   ## 將配置切換到 lightdm
```

安裝 x11VNC:

```
sudo apt-get install x11vnc -y
```

若未更新直接執行上面指令可能會出現 error, 解決方法:

```
sudo apt-get update
```

設定連接的 **VNC**密碼

```
sudo x11vnc -storepasswd /etc/x11vnc.pass
```

需手工設置一下權限, 默認設置的權限可能會導致其他用戶無法正常讀取

```
sudo chmod 755 /etc/x11vnc.pass
```

`rfbport` 參數指定監聽端口, `-forever` 參數指定客戶端斷開後不要停止服務而是繼續等待下一次的連接請求

```
sudo x11vnc -auth guess -rfbauth /etc/x11vnc.pass -rfbport 5900 -forever -display :0
```

設置開機自動啟動 **VNC** 功能

```
sudo gedit /etc/systemd/system/x11vnc.service[Unit]
Description=Start x11vnc at startup.
After=multi-user.target
```

```
[Service]
Type=simple
ExecStart=/usr/bin/x11vnc -auth guess -forever -loop -noxdamage -repeat -rfbauth /etc/x11vnc.pass -rfbport 5900
[Install]
WantedBy=multi-user.target
```

配置完 systemd 後啟動服務

```
sudo systemctl daemon-reload

sudo systemctl enable x11vnc

sudo systemctl start x11vnc
```

Step 2. 設定虛擬解析度

注意：需先裝上 Nvidia driver

再執行下列步驟，如果設定完重啟後導致無法開機，請進入命令行模式刪除該文件

Server 如果沒有外接螢幕顯示器， x-session 不能從外部獲取解析度，需要在 xorg.conf 中設置虛擬解析度。

ubuntu 默認已經沒有 /etc/X11/xorg.conf，也沒有必要用 Xorg -configure 創建，直接手動創建就行，並添加如下代碼 (用戶端螢幕解析度是 1920x1080，可以根據實際情況修改 Virtual 參數)

turn to /etc/X11

```
cd /etc/X11sudo gedit xorg.conf## VNC-virsual montior
Section "Device"
        Identifier "Configured Video Device"
EndSectionSection "Monitor"
        Identifier "Configured Monitor"
EndSectionSection "Screen"
        Identifier "Default Screen"
        Monitor "Configured Monitor"
        Device "Configured Video Device"
        SubSection "Display"
                Depth 24
                Virtual 1920 1080
        EndSubSection
EndSection
```

Restart System

```
sudo reboot
```

查看休眠是否開啟

```
systemctl status sleep.target  sleep.target - Sleep
```

關閉休眠模式

```
systemctl mask sleep.target suspend.target hibernate.target hybrid-sleep.target
```

確認休眠是否關閉

```
systemctl status sleep.target
```

MySQL

htop 看到　mysqld 佔 100%

```
mysql -u root
```

進入後執行

```
SHOW FULL PROCESSLIST;
```

會列出下面的內容，有問題的就會長很長

```
| 32892 | root | localhost | NULL | Query    |    0 | starting      | SHOW FULL PROCESSLIST
```

然後執行

```
KILL 32892;
```

使用 Fail2Ban 自動阻擋攻擊者

```
vi /etc/fail2ban/jail.local
```

加入

```
[nginx-http-auth]
enabled  = true
filter   = nginx-http-auth
action   = iptables[name=HTTP, port=http, protocol=tcp]
logpath  = /var/log/nginx/error.log
bantime  = 3600
findtime = 600
maxretry = 5
```

重新啟動服務

```
systemctl restart fail2ban
```

檢查當前被封的 IP

```
fail2ban-client status
```

額外再加入

```
[nginx-primary-script]
enabled = true
filter = nginx-primary-script
action = iptables[name=PrimaryScript, port=http, protocol=tcp]
logpath = /var/log/nginx/error.log
maxretry = 3
bantime = 3600
```

防止攻擊腳本

```
[Definition]
failregex = .*FastCGI sent in stderr: "Primary script unknown".*request: ".*installer\.php.*
            .*FastCGI sent in stderr: "Primary script unknown".*request: ".*WordPress/installer\.php.*
            .*FastCGI sent in stderr: "Primary script unknown".*request: ".*phpinfo\.php.*
```

```
        .*FastCGI sent in stderr: "Primary script unknown".*request: ".*info\.php.*

# 常見入侵腳本和工具
        .*GET /wp-admin.*                  # WordPress 管理面板攻擊
        .*GET /wp-login\.php.*             # WordPress 登錄攻擊
        .*GET /xmlrpc\.php.*               # WordPress XML-RPC 攻擊
        .*GET /phpMyAdmin/.*               # phpMyAdmin 攻擊
        .*GET /pma/.*                      # phpMyAdmin 縮寫攻擊
        .*GET /myadmin/.*                  # 其他管理工具探測
        .*GET /config\.php.*               # 嘗試訪問配置文件
        .*GET /setup\.php.*                # 安裝腳本
        .*GET /install\.php.*              # 安裝腳本
        .*GET /adminer.*                   # Adminer 攻擊

# 常見惡意腳本探測
        .*GET /shell\.php.*                # 後門探測
        .*GET /cmd\.php.*                  # 命令執行腳本
        .*GET /console\.php.*              # 惡意工具腳本
        .*GET /backdoor\.php.*             # 後門腳本探測
        .*GET /wp-content/uploads/shell.*  # WordPress 上傳漏洞
        .*GET /eval-stdin.*                # eval 利用腳本

# 文件探測攻擊
        .*GET /\.env.*                     # Laravel 或其他框架配置文件
        .*GET /\.git/config.*              # Git 配置文件
        .*GET /backup.*                    # 嘗試下載備份
        .*GET /dump.*                      # 嘗試訪問數據庫轉儲
        .*GET /debug.*                     # 調試工具
        .*GET /error_log.*                 # 日誌檔案

# 特定漏洞利用
        .*GET /HNAP1/.*                    # HNAP 協議漏洞 (常針對路由器)
        .*GET /boaform/admin/formLogin.*   # IOT 設備漏洞
        .*GET /invoker/JMXInvokerServlet.* # JBoss 漏洞
        .*GET /webdav.*                    # WebDAV 攻擊
        .*GET /manager/html.*              # Tomcat 管理界面探測
```

直接封某個 IP

```
fail2ban-client set nginx-primary-script banip xxx.xxx.xxx.xxx
```

如果封錯想解除

```
fail2ban-client set nginx-primary-script unbanip 192.168.1.100
```

白名單　　寫在　/etc/fail2ban/jail.local

```
[DEFAULT]
ignoreip = 127.0.0.1/8 ::1 192.168.1.100
```

做完任何設定都要記得重啟

```
systemctl restart fail2ban
```

網頁版的 htop Glances

安裝方式

```
pip install glances fastapi[all] uvicorn
```

執行 `glances -w` 即可

如果 which glances 顯示 /usr/local/bin/glances, 表示 Glances 已正確安裝在 /usr/local/bin 目錄，但系統可能無法直接執行它

如果沒有 /usr/local/bin, 執行以下命令將其臨時添加

```
export PATH=$PATH:/usr/local/bin
```

並將其永久添加到 .bashrc 或 .zshrc 文件中:

```
echo 'export PATH=$PATH:/usr/local/bin' >> ~/.bashrc
source ~/.bashrc
```

使用 Monit 監控資源

安裝

```
apt install monit
```

設定

```
cp /etc/monit/monitrc /etc/monit/monitrc.bak
```

vi /etc/monit/monitrc

```
set httpd port 2812
    use address 0.0.0.0  # 只允全部訪問
    allow localhost      # 允許    localhost 訪問
    allow yourip         # 允許    指定   IP 訪問
    allow admin:monit

# NGINX
check process nginx with pidfile /var/run/nginx.pid
  start program = "/bin/systemctl start nginx"
  stop program  = "/bin/systemctl stop nginx"
    if failed port 443 then restart
    if 5 restarts within 5 cycles then timeout

# PHP 7.4-FPM
check process php7.4-fpm with pidfile /run/php/php7.4-fpm.pid
    start program = "/etc/init.d/php7.4-fpm start"
    stop program = "/etc/init.d/php7.4-fpm stop"
    if failed unixsocket /var/run/php/php7.4-fpm.sock then restart
    if 5 restarts within 5 cycles then timeout

# MySQL
check process mysql with pidfile /var/lib/mysql/zfun-server.pid
    start program = "/etc/init.d/mysql start"
    stop program = "/etc/init.d/mysql stop"
    if failed host localhost port 3306 protocol mysql then restart
    if 5 restarts within 5 cycles then timeout
```

http://yourip:2812 就能進入監控

如果希望在服務異常時接收電子郵件通知 `vi /etc/monit/monitrc`

```
set mailserver smtp.your-email-provider.com port 587
    username "your-email@example.com" password "your-password"
    using tlsv1
    with timeout 30 seconds

set alert your-email@example.com
```

> 由於此功能的權限很大，可以做到關閉及啟用的功能，使用時務必要把防火牆等限制建置好

一些 log

查詢某個時段的流量，此完例為 2024年 11月 20日下午 2點

```
grep '20/Nov/2024:14:' /var/log/nginx/access.log | grep 'https://yoursite.com' | awk '{print $1}' | sort | uniq | wc -l
zgrep '20/Nov/2024:14:' /var/log/nginx/access.log.2.gz | grep 'https://yoursite.com' | awk '{print $1}' | sort | uniq | wc -l
```

Ubuntu 24.04效能調校

Opcache

/etc/php/8.3/fpm/conf.d/10-opcache.ini

```
; configuration for php opcache module
; priority=10
zend_extension=opcache.so
opcache.jit=off

; 強化穩定性（可選）
opcache.enable=1
opcache.enable_cli=0
opcache.memory_consumption=192
opcache.interned_strings_buffer=16
opcache.max_accelerated_files=10000
opcache.validate_timestamps=0
opcache.revalidate_freq=60
opcache.fast_shutdown=1
opcache.save_comments=1
opcache.enable_file_override=0
opcache.file_update_protection=2  ;可選
```

/etc/php/7.4/fpm/conf.d/10-opcache.ini

```
; configuration for php opcache module
; priority=10
zend_extension=opcache.so

; 其他建議參數
opcache.enable=1
opcache.enable_cli=0
opcache.memory_consumption=128
opcache.interned_strings_buffer=8
opcache.max_accelerated_files=10000
opcache.validate_timestamps=0
```

```
opcache.revalidate_freq=60
opcache.fast_shutdown=1
opcache.save_comments=1
opcache.enable_file_override=0
```

網站內容穩定、不常更新　　PHP 程式碼　　opcache.validate_timestamps=0

APcu

/etc/php/8.3/cli/conf.d/20-apcu.ini

```
extension=apcu.so
# chatgpt add
apc.enabled=1
apc.shm_size=64M
apc.ttl=3600
apc.user_ttl=7200
apc.gc_ttl=3600
apc.entries_hint=4096
apc.smart=1
apc.enable_cli=0
```

/etc/php/7.4/cli/conf.d/20-apcu.ini

```
extension=apcu.so
# chatgpt add
apc.enabled=1
apc.shm_size=64M
apc.ttl=3600
apc.user_ttl=7200
apc.gc_ttl=3600
apc.entries_hint=4096
apc.smart=1
apc.enable_cli=0
```

Nginx

/etc/nginx/nginx.conf

```
user www-data;
worker_processes auto;
worker_rlimit_nofile 100000;

events {
    worker_connections 10240;
    multi_accept on;
    use epoll;
}

gzip on;
gzip_vary on;
gzip_proxied any;
gzip_comp_level 6;
gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss
text/javascript;
```

pm2移機

更換主機並繼續使用 `pm2` 管理專案的過程，可以按照以下步驟進行：

---

1. 備份現有的專案和配置

(1) 備份專案文件

將原主機的專案目錄打包並下載到本地：

```
tar -czf projects_backup.tar.gz /path/to/your/project1 /path/to/your/project2
```

(2) 備份 `pm2` 配置

使用 `pm2` 導出當前的進程清單：

```
pm2 save
```

保存後的進程清單會存儲在 `~/.pm2/dump.pm2` 中，將此目錄也備份：

```
tar -czf pm2_backup.tar.gz ~/.pm2
```

(3) 備份相關環境（可選）

- 如果你的專案依賴 Node.js，建議記錄當前版本：`node -v`
- 如果專案使用 `.env` 文件等配置，確保這些文件也被備份。

---

2. 準備新主機環境

(1) 安裝 Node.js 和 npm

在新主機上安裝 Node.js，並確保版本與舊主機一致。可以使用以下方式安裝：

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install -y nodejs
```

（根據你的需求選擇合適的版本）

(2) 安裝 pm2

安裝 `pm2`:

```
sudo npm install -g pm2
```

---

3. 恢復備份

(1) 恢復專案文件

將之前的專案文件和目錄上傳到新主機，然後解壓:

```
scp projects_backup.tar.gz user@new-server:/path/to/
ssh user@new-server
tar -xzf projects_backup.tar.gz -C /desired/path/
```

(2) 恢復 `pm2` 配置

將 `pm2_backup.tar.gz` 上傳到新主機並解壓:

```
scp pm2_backup.tar.gz user@new-server:/path/to/
ssh user@new-server
tar -xzf pm2_backup.tar.gz -C ~/
```

然後加載進程清單:

```
pm2 resurrect
```

---

4. 檢查與測試

(1) 測試專案啟動

使用以下命令確認專案是否正常啟動:

```
pm2 list
```

若有任何專案未啟動，檢查日誌:

```
pm2 logs
```

(2) 配置自動啟動

確保 `pm2` 配置為開機自動啟動:

```
pm2 startup
pm2 save
```

---

5. 更新 **DNS** 或其他相關配置

若專案與域名相關，請更新 DNS 或修改其他相關配置，指向新主機的 IP。

這樣你就能順利地將專案從舊主機遷移到新主機，同時繼續使用 `pm2` 管理進程。

在 Windows 11安裝 WLS

安裝 WLS

開啟 PowerShell (以系統管理員身分)

```
wsl --install -d Ubuntu-24.04
```

※ wsl安裝成功後，有可能會發生在安裝 Ubuntu-24.04報錯誤沒安裝成功，另一台新機遇到的問題是 home 版會提示 vt沒開，我來是重開機後就好了

安裝後會詢問要使用什麼帳密，可直接用目前登入 windows用的帳密會比較好記

登入後是空空全新的 Ubuntu Linux環境，你可以針對需求進行安裝及設定

關聯 port

進入 UbuntuLinux後，會看到 IP是隨機的 net (外面不能連入)

假設我在 Ubuntu 裝好 nginx port 用 80，我在本機輸入隨機 IP 172.26.94.32，就能看到 Nginx的畫面

我想要讓外面的連入，要這樣做

使用 PowerShell (以系統管理員身分)

```
netsh interface portproxy add v4tov4 listenport=81 listenaddress=0.0.0.0 connectport=80
connectaddress=172.26.94.32
```

確認建立規則

```
netsh interface portproxy show all
```

刪除

```
netsh interface portproxy delete v4tov4 listenport=81 listenaddress=0.0.0.0
```

開通 81 PORT 防火牆

```
netsh advfirewall firewall add rule name="WSL2 Port 81" dir=in action=allow protocol=TCP localport=81
```

檢查是否生效

```
netsh advfirewall firewall show rule name="Allow WSL2 Port 81"
```

移除 WLS

使用 PowerShell (以系統管理員身分)

列出

```
wsl -l -v
```

```
  NAME              STATE           VERSION
* Ubuntu-24.04      Running         2
  docker-desktop    Running         2
```

移除

```
wsl --unregister Ubuntu-24.04
```

其他

自動更新 WSL2 IP 的 PowerShell 腳本

```
# ================================
# WSL2 PortProxy 自動更新腳本
# ================================

# 設定對外端口 (Windows listen port) 與內部端口 (WSL2 connect port)
$ListenPort  = 81
$ConnectPort = 80

# 抓取 WSL2 當前 IP
$wsl_ip = wsl hostname -I | ForEach-Object { $_.Trim() } | Select-Object -First 1

if (-not $wsl_ip) {
    Write-Host "□ □□□□ WSL2 IP, 請確認 WSL2 已啟動。 " -ForegroundColor Red
    exit
}

Write-Host "□ □□□ WSL2 IP: $wsl_ip" -ForegroundColor Green
```

```
# 移除舊規則（避免重複）
netsh interface portproxy delete v4tov4 listenport=$ListenPort listenaddress=0.0.0.0 > $null 2>&1

# 新增新的 PortProxy 規則
netsh interface portproxy add v4tov4 listenport=$ListenPort listenaddress=0.0.0.0 connectport=$ConnectPort
connectaddress=$wsl_ip

# 確保防火牆允許該 Port
netsh advfirewall firewall add rule name="WSL2 Port $ListenPort" dir=in action=allow protocol=TCP
localport=$ListenPort > $null 2>&1

Write-Host "□ PortProxy 已更新：0.0.0.0:$ListenPort → $wsl_ip:$ConnectPort" -ForegroundColor Cyan
```

執行

```
powershell -ExecutionPolicy Bypass -File C:\Users\你的帳號\update-wsl-port.ps1
```

開機自動執行

開啟 工作排程器 (taskschd.msc)

建立基本工作 → 觸發器選「當登入時」

動作選「啟動程式」，程式輸入：

```
powershell
```

參數輸入：

```
-ExecutionPolicy Bypass -File C:\Users\你的帳號\update-wsl-port.ps1
```

□ □□□□□□□ WSL2 IP 每次重開會變，永遠能透過 http://192.168.0.168:81/ 存取。

□□ Redis 快速檢查指令清單

1. 查看 Redis 整體記憶體使用

```
redis-cli info memory
```

重點看:

- `used_memory_human` → Redis 目前用多少
- `maxmemory` → 上限
- `mem_fragmentation_ratio` → 記憶體碎片率

---

2. 查看資料筆數

```
redis-cli dbsize
```

顯示 key 數量（Drupal 正常應該幾千~幾萬，不會幾百萬）。

---

3. 找出最大 key

```
redis-cli --bigkeys
```

會依 key type 顯示「最大 key」與統計資訊，幫助找出異常大的 cache。

例如看到:

```
Biggest string found 'cache:default:tags' has 200 MB
```

就代表 cache tag key 爆掉。

---

4. 檢查 key 記憶體大小

（需要 Redis 4.0+）

```
redis-cli memory usage <key>
```

例如:

`redis-cli memory usage cache:default:tags`

會輸出 bytes，用來判斷哪個 key 特別肥。

---

5. 抽樣列出 key

`redis-cli scan 0 match "cache:*" count 20`

可隨機抽樣看看 Drupal 快取 key 的型態。

---

6. 查看所有 key 大小排名（進階）

```
redis-cli --raw keys "*" | \
xargs -L1 redis-cli memory usage | \
sort -n | tail -20
```

這會顯示 最大 **20** 個 **key** 的記憶體用量。

---

7. 清理快取

- 清全部 Redis：

  `redis-cli flushall`
- 清 Drupal Cache：

  `drush cr`

□□ □□□□□

1. `redis-cli info memory` → 確認總用量
2. `redis-cli dbsize` → 確認 key 數量
3. `redis-cli --bigkeys` → 找最大 key
4. `redis-cli memory usage <key>` → 精準看佔用記憶體
5. 如果某個 cache tag 或 session 爆大 → 考慮調整 Drupal 的 Redis 設定或加上 `maxmemory` 控制

用 Docker打造最佳效能的Wordpress環境

要做的事: Docker → WordPress 可用流程

1. 建立 Docker 架構

- 撰寫 **docker-compose.yml**, 服務包含:
  - **nginx**: 前端 Web server (只負責容器內請求, 對外由主機 Proxy)
  - **php (php-fpm)**: 執行 WordPress/PHP
  - **mariadb**: 資料庫, 儲存 WordPress 資料
  - **redis**: Object Cache (提供效能優化)
  - **wp-cli**: 命令列管理工具
  - **cron**: 跑 WordPress 排程 (wp-cron)
- 設定:
  - Nginx 靜態檔案快取、購物流程繞過快取。
  - PHP FPM (memory_limit、 pm.max_children 等) 依你的 4C/4G RAM 調整。
  - MariaDB 用 volume 儲存資料、 utf8mb4 編碼。
  - Redis 用獨立 volume, 內建設定。

---

2. 主機層 Proxy (Ubuntu Nginx)

- 主機 **Nginx** 反向代理 → `127.0.0.1:8080` (容器內 nginx) 。
- 加入 **SSL (Let’s Encrypt Certbot)**, 伺服器區塊 `server_name mysite.com;` 。
- 解決 **502** 問題：確保容器內 nginx 配置正確、 port 綁在 127.0.0.1:8080。
- 調整 proxy header, 讓 WordPress 正確認定 HTTPS (解決 CSS/JS 不載入問題)：

```
proxy_set_header X-Forwarded-Proto https;
proxy_set_header X-Forwarded-Port  443;
```

3. WordPress 部署

- 使用 **wp-cli** 下載 WordPress 核心:

```
docker compose exec wp-cli wp core download --allow-root
```

- 產生 `wp-config.php`, 連線資訊:
    - DB 名稱: `wp`
    - 帳號: `wpuser`
    - 密碼: `change_me`
    - Host: `db:3306` ← 注意 Docker 要用服務名稱, 不是 localhost
- 透過安裝頁完成初始化（語言、站名、管理員帳密）。

---

4. 修正容器問題

- **Nginx**:
    - 修掉 `fastcgi_cache_path` 放錯區塊（要在 http 層級, 不是 server）。
    - 最小化設定檔 → 先跑起站再逐步加快取規則。
- **cron**:
    - 原本 Restarting → 新增 `cron/wp-cron.sh` 腳本, 正確執行排程。
- **wp-config.php**:
    - 修正 Redis 設定錯誤（`define('WP_REDIS_HOST', redis );` → `define('WP_REDIS_HOST', 'redis');`）
    - 加入 HTTPS 判斷（反向代理環境必須）:

```
if (isset($_SERVER['HTTP_X_FORWARDED_PROTO']) && $_SERVER['HTTP_X_FORWARDED_PROTO'] === 'https') {
    $_SERVER['HTTPS'] = 'on';
}
```

5. Redis 整合

- 確認 Redis 容器正常: `docker compose exec redis redis-cli ping → PONG`。
- 在 `wp-config.php` 正確寫入:

```
define( 'WP_REDIS_HOST', 'redis' );
define( 'WP_REDIS_PORT', 6379 );
define( 'WP_REDIS_DATABASE', 1 );
```

- 重啟 PHP 讓 opcache 重載: `docker compose restart php`。
- 使用 `wp-cli` 啟用 Redis drop-in, 確認狀態:

```
docker compose exec wp-cli wp redis enable --allow-root
docker compose exec wp-cli wp redis status --allow-root
```

---

6. WordPress 網站成功啟動

- 首頁、安裝頁能正常載入 CSS/JS（不再 502、不再純文字）。
- 安裝流程完成，能登入後台。
- Redis Object Cache 狀態正常，顯示連到 `redis:6379`。
- 主機 Proxy + 容器 Nginx + PHP-FPM + DB 都在健康狀態。

專案目錄結構

```
wordpress-perf/
├── docker-compose.yml
├── .env
├── nginx/
│   └── conf.d/
│       ├── wordpress.conf
│       └── 00-cache.conf
├── cron/
│   └── wp-cron.sh
└── README.md
```

檔案內容

1. `.env`

```
HTTP_PORT=8080

DB_NAME=wp
```

```
DB_USER=wpuser
DB_PASSWORD=change_me
DB_ROOT_PASSWORD=change_root

# PHP □□□  /□□□□□□  4C/4G RAM□
PHP_MEMORY_LIMIT=256M
PHP_MAX_CHILDREN=10
PHP_MAX_REQUESTS=400
```

---

2. docker-compose.yml

```
services:
  nginx:
    image: nginx:1.25-alpine
    restart: always
    ports:
      - "127.0.0.1:${HTTP_PORT:-8080}:80"
    volumes:
      - ./nginx/conf.d:/etc/nginx/conf.d:ro
      - wp_data:/var/www/html:ro
      - cache_data:/var/cache/nginx
    depends_on:
      - php

  php:
    build:
      context: .
      dockerfile_inline: |
        FROM wordpress:php8.3-fpm-alpine
        RUN docker-php-ext-install mysqli pdo pdo_mysql
        COPY --from=mlocati/php-extension-installer /usr/bin/install-php-extensions /usr/local/bin/
    restart: always
    environment:
      WORDPRESS_DB_NAME: ${DB_NAME}
      WORDPRESS_DB_USER: ${DB_USER}
      WORDPRESS_DB_PASSWORD: ${DB_PASSWORD}
      WORDPRESS_DB_HOST: db:3306
```

```
      PHP_MEMORY_LIMIT: ${PHP_MEMORY_LIMIT}
    volumes:
      - wp_data:/var/www/html

  db:
    image: mariadb:10.11
    restart: always
    environment:
      MYSQL_DATABASE: ${DB_NAME}
      MYSQL_USER: ${DB_USER}
      MYSQL_PASSWORD: ${DB_PASSWORD}
      MYSQL_ROOT_PASSWORD: ${DB_ROOT_PASSWORD}
    volumes:
      - db_data:/var/lib/mysql

  redis:
    image: redis:7-alpine
    restart: always
    volumes:
      - redis_data:/data

  wp-cli:
    image: wordpress:cli-php8.3
    depends_on:
      - php
      - db
    volumes:
      - wp_data:/var/www/html
    entrypoint: wp

  cron:
    image: wordpress:cli-php8.3
    depends_on:
      - php
    volumes:
      - wp_data:/var/www/html
      - ./cron:/cron
    entrypoint: /cron/wp-cron.sh

volumes:
```

```
  wp_data:
  db_data:
  redis_data:
  cache_data:
```

4. nginx/conf.d/wordpress.conf

```
server {
  listen 80 default_server;
  server_name _;
  root /var/www/html;
  index index.php index.html;

  # □□□□□□
  set $skip_cache 0;
  if ($request_method = POST) { set $skip_cache 1; }
  if ($query_string != "")   { set $skip_cache 1; }
  if ($http_cookie ~*
"wordpress_logged_in_|woocommerce_items_in_cart|woocommerce_cart_hash|wp_woocommerce_session_") {
set $skip_cache 1; }
  if ($request_uri ~* "^/(wp-admin/|wp-login\.php|cart|checkout|my-account|wc-api/|wp-json/)") { set
$skip_cache 1; }
  if ($request_uri ~* "add-to-cart=") { set $skip_cache 1; }
  if ($request_uri ~* "admin-ajax\.php") { set $skip_cache 1; }

  location / {
    try_files $uri $uri/ /index.php?$args;
  }

  location ~ \.php$ {
    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_pass php:9000;

    fastcgi_buffers 16 32k;
    fastcgi_buffer_size 32k;
    fastcgi_read_timeout 120s;

    fastcgi_cache_bypass $skip_cache;
```

```
    fastcgi_no_cache $skip_cache;
    fastcgi_cache PHP_MICRO;
    fastcgi_cache_valid 200 301 302 5s;

    add_header X-Cache $upstream_cache_status;
    fastcgi_param HTTPS on;
    fastcgi_param SERVER_PORT 443;
    fastcgi_param HTTP_X_FORWARDED_PROTO https;
  }

  location ~* \.(?:css|js|jpg|jpeg|gif|png|svg|webp|ico|ttf|otf|woff|woff2)$ {
    expires 30d;
    access_log off;
  }

  client_max_body_size 64m;
}
```

---

5. cron/wp-cron.sh

```
#!/bin/sh
set -e
# □  1 □□□□□         wp-cron
while true; do
  wp --path=/var/www/html --allow-root cron event run --due-now >/dev/null 2>&1 || true
  sleep 60
done
```

---

6. README.md□□□□□

```
# WordPress on Docker (WooCommerce Ready)

## Services
- Nginx 1.25 (reverse proxy inside container)
- PHP-FPM 8.3 (with mysqli, pdo_mysql)
```

- MariaDB 10.11
- Redis 7 (object cache)
- WP-CLI
- Cron container for wp-cron

Usage
```bash
docker compose up -d
```

WordPress setup

```
docker compose exec wp-cli wp core download --allow-root
docker compose exec wp-cli wp config create \
  --dbname=$DB_NAME --dbuser=$DB_USER --dbpass=$DB_PASSWORD \
  --dbhost=db:3306 --skip-check --allow-root
docker compose exec wp-cli wp core install \
  --url="https://mysite.com" \
  --title="MySite Shop" \
  --admin_user="admin" \
  --admin_password="ChangeMe123!" \
  --admin_email="you@example.com" \
  --allow-root
```

Redis setup

```
docker compose exec wp-cli wp config set WP_REDIS_HOST 'redis' --allow-root
docker compose exec wp-cli wp config set WP_REDIS_PORT 6379 --raw --allow-root
docker compose exec wp-cli wp config set WP_REDIS_DATABASE 1 --raw --allow-root
docker compose restart php
docker compose exec wp-cli wp redis enable --allow-root
```

Proxy setup (Host machine)

Nginx reverse proxy:

```
server {
    listen 443 ssl http2;
```

```
    server_name mysite.com;

    ssl_certificate     /etc/letsencrypt/live/mysite.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/mysite.com/privkey.pem;

    location / {
        proxy_pass http://127.0.0.1:8080;
        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-Proto https;
    }
}
```

Result

- WordPress □□□□ (`https://mysite.com`)
- Redis Object Cache □□
- Nginx + PHP-FPM + MariaDB + Redis □□□□