NASA HW6

B11901164 陳秉緯

1 Short Answer

1. (a) ref: 1 (https://www.wix.com/blog/static-vs-dynamic-website)
    - Static web server 是指提供靜態內容（例　HTML、CSS、圖片等）的伺服器，使用者請求什麼就直接回傳什麼，不經過額外處理；Dynamic web server 則會根據使用者的請求動態產生內容，通常會與資料庫互動或執行 server-side 程式（例　PHP、Node.js）。
    - 差異：
        - Static web server 回傳固定內容，Dynamic web server 回傳經處理後的內容
        - Static web server 效能較高、延遲較低，Dynamic web server 負擔較重
        - Static web server 比較適合簡單網站，Dynamic web server 適合需要互動或資料處理的網站

    (b) ref: 1 (https://www.cloudflare.com/zh-tw/learning/cdn/glossary/reverse-proxy/), 2 (https://medium.com/starbugs/web-server-nginx-1-cf5188459108)
    - Forward proxy 是位在使用者與網路之間的代理伺服器，使用者透過 proxy 存取外部網站；Reverse proxy 則是位在網路與伺服器之間，用來處理來自使用者的請求，再轉給內部伺服器。
    - 優點：
        - 可做快取，減少伺服器負擔與延遲
        - 可統一管理請求流量，支援負載平衡或流量控管
        - 提高安全性，隱藏內部伺服器 IP，防止攻擊

    (c) ref: 1 (https://www.cloudflare.com/learning/performance/what-is-load-balancing/), 2 (https://www.vmware.com/topics/round-robin-load-balancing)
    - Load balancing 是一種分散 requests 流量的技術，將多個使用者的 requests 分配給多台伺服器，以提升整體系統效能與降低單一伺服器的負

載。

- Round Robin：requests 輪流分配給後端伺服器。例　有三台伺服器，第一個 request 給 server1，第二個給 server2，第三個給 server3，接下來又回到 server1，　此循環分配。

2. (a) ref: 1 (https://medium.com/@daniel.doody/web-servers-a-high-level-overview-of-nginx-vs-apache-ba87f923c024), 2 (https://mohitmishra786.github.io/chessman/2024/12/29/Understanding-NGINX-Worker-Architecture.html)

   - Event-driven architecture 是用非同步、單一 thread 的方式處理多個 client，透過事件迴圈（event loop）來管理所有連線，不需要為每個 client 開一個新的 process，這樣可以有效支援大量連線，降低資源消耗。
   - Process-driven architecture 是每個 client request 都會對應一個 process 或 thread，也就是來一個 client 就開一個新的處理單位，這在連線數少的時候沒問題，但在大量連線像 C10k 時，process/thread 的 overhead 很高，導致效能下降。

   (b) ref: 1 (https://docs.nginx.com/nginx/admin-guide/basic-functionality/runtime-control/)

   - Master process 負責初始化設定、讀取設定檔、管理 Worker processes。
   - Worker process 負責實際處理 client 的 request，包括接收連線、回應內容、處理反向代理、負載平衡等功能。

   (c) ref: 1 (https://mohitmishra786.github.io/chessman/2024/12/29/Understanding-NGINX-Worker-Architecture.html)

   - 預設執行 Master process 的 user 是 root，因為它需要綁定低於 1024 的 port（像是 80）以及執行一些需要高權限的初始化工作。
   - Worker process 的預設 user 根據不同的 distro 而有所不同，通常是 nginx 或 www-data，因為實際處理請求的部分不需要 root 權限，這樣做可以提高安全性，避免某個 Worker 被攻擊時直接取得 root 權限。

3. (a) ref: 1 (https://zh.wikipedia.org/zh-tw/%E5%85%AC%E9%96%8B%E9%87%91%E9%91%B0%E5%9F%BA%E7%A4%8E%E5%BB%BA%E8%A8%AD), 2 (https://www.ssl.com/zh-TW/article/what-is-public-key-infrastructure-pki/)

   - PKI（Public Key Infrastructure）是一套管理公鑰和私鑰的系統，主要用於

建立、管理、分發、使用、儲存和撤銷數位證書。PKI 的核心組件包括憑證授權中心（CA）、註冊機構（RA）、憑證儲存庫和憑證撤銷清單（CRL）等。

- TLS 透過 PKI 提供的數位證書，TLS 能夠驗證通信雙方的身份，並建立加密的通信通道，確保資料的機密性和完整性。

(b) ref: 1 (https://zh.wikipedia.org/zh-tw/%E8%87%AA%E5%8B%95%E6%86%91%E8%AD%89%E6%9B%B4%E6%96%B0%E7%92%B0%E5%A2%83)

- ACME（Automatic Certificate Management Environment）是一種自動化協議，旨在簡化 SSL/TLS 憑證的申請、驗證、發放和續期過程，減少人工干預。
- 流程：
  1. 生成密鑰對：用戶端生成一對新的公私鑰。
  2. 註冊帳戶：用戶端使用生成的密鑰對向 CA 註冊一個新的 ACME 帳戶，並簽署註冊請求。
  3. 提交憑證簽名請求（CSR）：用戶端創建一個 CSR，包含所需的域名等資訊，並提交給 CA。
  4. 驗證域名所有權：CA 通過 ACME 協議執行驗證挑戰（Validation Challenge），例　HTTP-01 或 DNS-01 挑戰，以確認用戶端對所申請的域名具有控制權。
  5. 頒發憑證：驗證成功後，CA 向用戶端頒發憑證。
  6. 部署憑證：用戶端接收並將憑證部署到伺服器上。
  7. 憑證續期：在憑證即將到期時，用戶端自動執行續期流程。

(c)

- 以網域名稱發放憑證的好處：
  - 網域名稱可以指向不同的 IP 地址，允許伺服器變更 IP 而不用重新申請憑證。
  - 使用網域名稱可確保用戶訪問的是預期的網站，即使其 IP 地址發生變化。
  - 基於網域的憑證管理更簡單，特別是在使用內容分發網路（CDN）或負載平衡等技術時。

- 若以 IP 地址發放憑證會有什麼缺點：
  - IP 地址變更時，需要重新申請和部署新的憑證。
  - 對於共享主機環境或使用虛擬主機的情況，基於 IP 的憑證難以管理。

(d) ref: 1 (https://docs.nginx.com/nginx/admin-guide/security-controls/terminating-ssl-http/)

- SSL termination 是指在負載均衡器或反向代理伺服器（ Nginx）處理 SSL/TLS 加密和解密的過程。客戶端與 Nginx 之間的通信是加密的，而 Nginx 與後端伺服器之間的通信則是未加密的 HTTP。這樣可以減輕後端伺服器的負擔，因為它們不需要處理加解密操作。
- 在 Nginx 經過 SSL termination 後，為了讓後端伺服器識別原始請求是否為 HTTPS，可以透過以下方式：
  - Nginx 可以在轉發請求時，添加特定的 HTTP 標頭，例 `X-Forwarded-Proto: https`，讓後端伺服器知道原始請求使用的是 HTTPS。
  - 在 Nginx 配置中，使用 `proxy_set_header` 指令設置相關標頭。

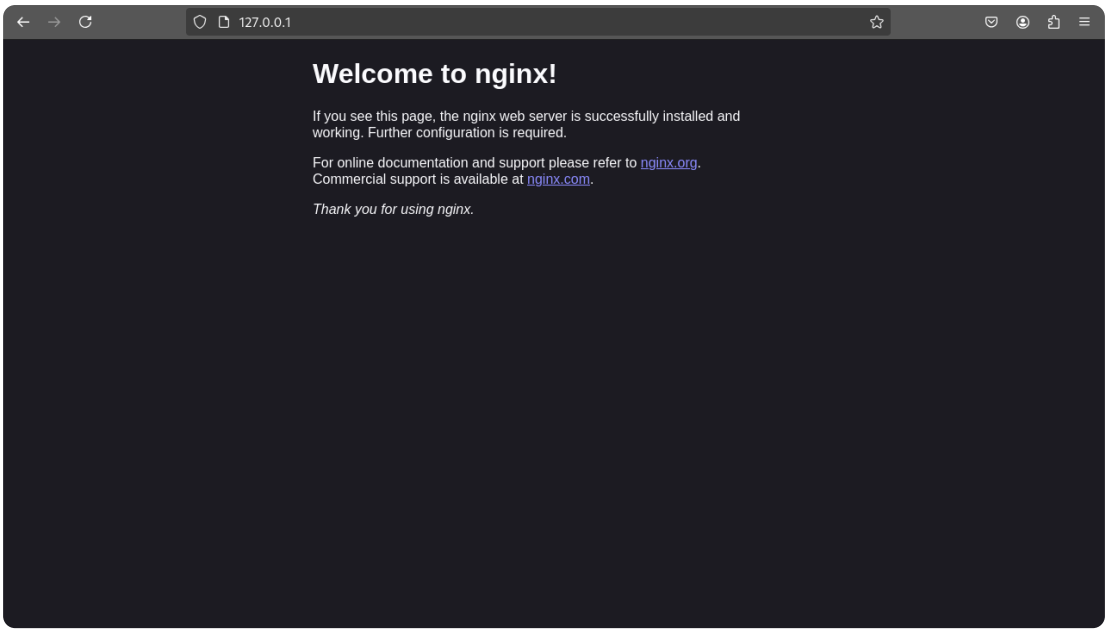
2 Web Server Configurations

ref:
1 (https://stackoverflow.com/questions/10674867/nginx-default-public-www-location), 2 (https://wshs0713.github.io/posts/6a171975/), 3 (https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/), 4 (https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/)

1. 1. `sudo apt update`
   2. `sudo apt install nginx -y` 安裝 nginx
   3. `sudo systemctl start nginx && sudo systemctl enable nginx`
   4. 設定防火牆，只允許 port 22 與 80：

```
sudo apt install ufw -y
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow 22
sudo ufw allow 80
sudo ufw enable
```

- 瀏覽器截圖：



- 測試防火牆是有工作的：

```
vboxuser@vbox:~$ nmap -p- 127.0.0.1
Starting Nmap 7.93 ( https://nmap.org ) at 2025-04-06 17:27 CST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000061s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 1.96 seconds
```

2. 1. `sudo nano /var/www/html/index.html` 加入：

```
<!DOCTYPE html>
<html>
  <head>
    <title>index</title>
  </head>
  <body>
    <h1>B11901164: Default</h1>
  </body>
</html>
```

- 瀏覽器截圖：

B11901164: Default

3. 1. `mkdir /var/www/html/forbidden` 創 forbidden 資料夾
   2. `sudo chmod 000 /var/www/html/forbidden` 讓 nginx 無法讀取 forbidden 資料夾
   3. `sudo nano /var/www/html/403.html` 加入：

```
<!DOCTYPE html>
<html>
  <head>
    <title>403 Forbidden</title>
  </head>
  <body>
    <h1>B11901164: 403 Forbidden</h1>
  </body>
</html>
```

   4. `sudo nano /etc/nginx/sites-available/default` 修改 nginx 的設定檔來指定 403 錯誤顯示自訂頁面，把以下加在 `server {...}` 內：

```
error_page 403 /403.html;
location = /403.html {
    root /var/www/html;
    internal;
}
```

   5. `sudo systemctl reload nginx` reload nginx

   ○ 瀏覽器截圖：

B11901164: 403 Forbidden

- 測試： `curl -i http://127.0.0.1/forbidden`

```
vboxuser@vbox:/var/www/html$ curl -i http://127.0.0.1/forbidden
HTTP/1.1 403 Forbidden
Server: nginx/1.22.1
Date: Sun, 06 Apr 2025 09:42:20 GMT
Content-Type: text/html
Content-Length: 140
Connection: keep-alive
ETag: "67f24b8e-8c"

<!DOCTYPE html>
<html>
  <head>
    <title>403 Forbidden</title>
  </head>
  <body>
    <h1>B11901164: 403 Forbidden</h1>
  </body>
</html>
```

4. 1. `sudo nano /etc/nginx/sites-available/default` 加入：

```
location ~ ^/~([^/]+)(/.*)?$ {
    alias /home/$1/htdocs$2;
    autoindex on;
    index index.html;
}
```

2. `sudo systemctl reload nginx` reload nginx
3. `sudo mkdir /etc/skel/htdocs` 讓所有新建立的使用者家目錄都會包含 htdocs 資料夾

4. `sudo adduser user1` 新增使用者 user1
5. `echo "<h1>user1: UserDir</h1>" | sudo tee /home/user1/htdocs/index.html` 在 /home/user1/htdocs 下建立 index.html
6. 更改權限，讓 nginx 可讀：

```
sudo chmod o+x /home/user1
sudo chmod -R o+r /home/user1/htdocs
```

- 瀏覽器截圖：

user1: UserDir

5. 1. 使用 NAT + Host-only network 在 ServerMain，ServerA，與 ServerB
   2. 在 ServerMain： `sudo nano /etc/network/interfaces` 加入：

```
auto enp0s8
iface enp0s8 inet static
    address 192.168.56.10
    netmask 255.255.255.0
```

再 `sudo reboot`

3. 在 ServerA：

```
sudo apt update
sudo apt install nginx -y
echo "I am Server A" | sudo tee /var/www/html/index.html
sudo systemctl restart nginx
```

`sudo nano /etc/network/interfaces` 加入：

```
auto enp0s8
iface enp0s8 inet static
    address 192.168.56.11
    netmask 255.255.255.0
```

然後再 `sudo reboot`

4. 在 ServerB：

```
sudo apt update
sudo apt install nginx -y
echo "I am Server B" | sudo tee /var/www/html/index.html
sudo systemctl restart nginx
```

`sudo nano /etc/network/interfaces` 加入：

```
auto enp0s8
iface enp0s8 inet static
    address 192.168.56.12
    netmask 255.255.255.0
```

然後再 `sudo reboot`

5. 在 ServerMain： `sudo nano /etc/nginx/sites-available/default` 在 `server{...}` 加入：

```
location /serverA/ {
    proxy_pass http://192.168.56.12/;
    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;
}

location /serverB/ {
    proxy_pass http://192.168.56.13/;
    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;
}
```

然後再 `sudo systemc reload nginx` reload nginx

- 瀏覽器截圖：

I am Server A

I am Server B

6. 1. 照 2-5. 架 Server C 或 clone 一臺改 IP: 192.168.56.14
   2. 在 ServerMain： `sudo nano /etc/nginx/sites-available/default` 在 `server {...}` 加入:

```
location /serverC/ {
    proxy_pass http://192.168.56.14/;
    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;
}

location /balance/ {
    proxy_pass http://backend/;
    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;
}
```

3. 在 ServerMain： `sudo nano /etc/nginx/nginx.conf` 在 `http {...}` 加入：

```
upstream backend {
    server 192.168.56.12 max_fails=3 fail_timeout=30s;
    server 192.168.56.13 max_fails=3 fail_timeout=30s;
    server 192.168.56.14 backup;
}
```

然後再 `sudo systemctl reload nginx` reload nginx

- 瀏覽器截圖：

I am Server A

I am Server B

- 成功連線至 ServerC 的截圖：

I am Server C