Network Administration/System Administration (NTU CSIE, Spring 2024) Homework #11 - Nginx

B12902110 呂承諺

May 13, 2024

Web Terminology

1.

	Apache	Nginx
Asynchronous, event-driven	Via event MPM	Native
Mail proxy support	No	Yes
License	Apache License 2.0	2-Clause BSD License

References

- Apache HTTP Server - Wikipedia
- Nginx - Wikipedia
- Boosting NGINX Performance 9x with Thread Pools
- event - Apache HTTP Server Version 2.4
- Apache vs Nginx: Practical Considerations | DigitalOcean

2. A **static web server** sends out files hosted on the server as-is.

   A **dynamic web server** constructs the response content during runtime with resources such as HTML templates, databases, or API calls.

   A static web server usually handle requests faster than a dynamic web server because no content has to be built by the server.

   References

   - What is a web server? - Learn web development | MDN
   - Dynamic web page - Wikipedia

3. **Client-side rendering (CSR)**: The server sends a minimal HTML page to the client, and the client then renders the main content dynamically using JavaScript, which may include additional HTTP fetches and API calls. A benefit of CSR is a smooth and interactive user experience, just like using an app.

   Server-side rendering (SSR): When the client requests a page, the server renders the complete page with data from databases or API calls, and then sends the response to the client. A benefit of SSR is better search engine optimization than CSR.

   Static-site generation (SSG): The whole website is rendered into static pages during build time. The main benefit of SSG excellent performance and fast loading times.

   Incremental static regeneration (ISR): ISR allows static pages to be built on a per-page basis. A page will be generated upon first request, and future requests within a certain amount of time will be served from cache. It offers better performance than SSR because only certain pages rather than the whole website has to be regenerated upon content change.

   References

   - [教學] SSR 與 CSR 深度解析：從渲染方式到效能優化 - Shubo 的程式開發筆記
   - Understanding CSR, SSR, SSG, and ISR: A Next.js Perspective | by Aditya Kumar Tiwari | Bootcamp
   - SSR vs CSR vs ISR vs SSG
   - Data Fetching: Incremental Static Regeneration (ISR) | Next.js

4. A **proxy** is an intermediate between the client and server. When the client wants to access a resource, the request goes to the proxy instead of directly to the server. The proxy performs the request to the server, receives the response, and then sends the response back to the client.

   Some benefits of a proxy are:

   - **Security**: Mask the IP address of the client.
   - **Filtering**: Filter out unwanted content.
   - **Caching**: Cache frequently accessed resources for performance.

   References

   - Proxy server - Wikipedia

5. A **reverse proxy** is a proxy server that acts to clients just like a normal web server, but actually forwards requests to and responses from one or more web servers behind it.

   Some benefits of a reverse proxy server are:

   - **Load balancing**: Distribute load among multiple servers.
   - **Security**: Only the reverse proxies have to be exposed to the public, while the actual web servers can be hidden behind a firewall.
   - **Caching**: Cache frequently accessed resources for performance.

   References

   - Reverse proxy - Wikipedia

Web Server Configurations

6. **Steps**

(1) Run the following commands to prepare the VM.

```
$ cp -r /tmp2/nasa-hw11 /tmp2/b1290110
$ cd /tmp2/b12902110/nasa-hw11
$ qemu-img create -f qcow2 disk0.qcow2 20G
```

(2) Create `/tmp2/b12902110/nasa-hw11/run_vm.sh` as the following.

```
#!/bin/bash

readonly MACHINE_IP="$(ip -4 a s net0.30 | grep -oP '(?<=inet\s)\d+(\.\d+){3}')"
qemu-system-x86_64 \
  -enable-kvm \
  -cpu host \
  -smp 8,sockets=1,cores=4,threads=2 \
  -m 8G \
  -nic user,hostfwd=tcp::11022-:22,hostfwd=tcp::11080-:80,hostfwd=tcp::11043-:443 \
  -monitor stdio \
  -vga virtio \
  -vnc ${MACHINE_IP}:0,to=10000,password=on \
  -drive file=disk0.qcow2 \
  -drive file=debian.iso,media=cdrom
```

(3) Boot up the VM, connect to it via QEMU's VNC, and follow the Debian installation guide. After the installation finished, reboot into the VM.

(4) Configure sudo as the root user.

```
$ su -
# apt install -y sudo
# usermod -aG sudo b12902110
```

(5) Re-login as user b12902110. Install the necessary package for our web server.

```
$ sudo apt install -y nginx
```

(6) Start the nginx service.

```
$ sudo systemctl start nginx.service
```

Result

```
b12902110@nasa-hw11:~$ systemctl status nginx.service
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-05-12 04:30:36 CST; 2min 14s ago
       Docs: man:nginx(8)
    Process: 962 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
    Process: 963 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 994 (nginx)
      Tasks: 9 (limit: 9472)
     Memory: 6.8M
        CPU: 80ms
     CGroup: /system.slice/nginx.service
             ├─ 994 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             ├─ 996 "nginx: worker process"
             ├─ 997 "nginx: worker process"
             ├─ 998 "nginx: worker process"
             ├─ 999 "nginx: worker process"
             ├─1000 "nginx: worker process"
             ├─1001 "nginx: worker process"
             ├─1002 "nginx: worker process"
             └─1003 "nginx: worker process"
b12902110@nasa-hw11:~$
```

References

- command usermod not found

7. **Steps**

(1) Install `ufw`.

```
$ sudo apt install -y ufw
```

(2) Configure firewall rules with the following commands.

```
$ sudo ufw default deny
$ sudo ufw allow 22
$ sudo ufw allow 80
$ sudo ufw allow 443
$ sudo ufw enable
```

Result

This part is done after the last problem, so we have more services than an HTTP and an SSH service.

In the QEMU console, add another port forwarding rule: 11088 on the host to 8888 on the VM.

```
(qemu) hostfwd_add tcp::11088-:8888
```

Therefore, we have 4 port forwarding rules.

Source	Destination
ws2.csie.ntu.edu.tw:11022	nasa-hw11:22
ws2.csie.ntu.edu.tw:11080	nasa-hw11:80
ws2.csie.ntu.edu.tw:11043	nasa-hw11:443
ws2.csie.ntu.edu.tw:11088	nasa-hw11:8888

All of the 4 ports has a service running on it.

```
b12902110@nasa-hw11:/var/www/hostB$ netstat -tl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:9999            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:http            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:https           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:8888            0.0.0.0:*               LISTEN
tcp6       0      0 [::]:9999               [::]:*                  LISTEN
tcp6       0      0 [::]:http               [::]:*                  LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
tcp6       0      0 [::]:https              [::]:*                  LISTEN
tcp6       0      0 [::]:8888               [::]:*                  LISTEN
b12902110@nasa-hw11:/var/www/hostB$
```

However, only ports 22, 80, and 443 are accessible from outside the VM.

```
b12902110@ws2: /tmp2/b12902110/nasa-hw11
$ ssh -p 11022 b12902110@127.0.0.1
The authenticity of host '[127.0.0.1]:11022 ([127.0.0.1]:11022)' can't be established.
ED25519 key fingerprint is SHA256:EJBy9bOgAc78EF9Fs+NGOvtWEneStJlGuYyFv6KuNog.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? ^C
[130]

b12902110@ws2: /tmp2/b12902110/nasa-hw11
$ curl http://localhost:11080
<!DOCTYPE html>
<html>
  <head>
    <title>Hello! My name is b12902110!</title>
  </head>
  <body>
    <h1>Hello! My name is b12902110!</h1>
  </body>
</html>

b12902110@ws2: /tmp2/b12902110/nasa-hw11
$ curl --insecure https://localhost:11043
<!DOCTYPE html>
<html>
  <head>
    <title>Hello! My name is b12902110!</title>
  </head>
  <body>
    <h1>Hello! My name is b12902110!</h1>
  </body>
</html>

b12902110@ws2: /tmp2/b12902110/nasa-hw11
$ curl http://localhost:11088
curl: (56) Recv failure: Connection reset by peer
[56]
```

References

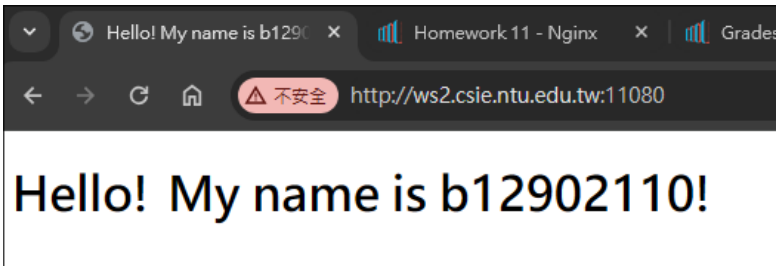
- How To Open a Port on Linux | DigitalOcean
- How to Set Up a Firewall with UFW on Ubuntu | DigitalOcean

8. **Steps**

Create `/var/www/html/index.html` as the following.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello! My name is b12902110!</title>
  </head>
  <body>
    <h1>Hello! My name is b12902110!</h1>
  </body>
</html>
```

Result



9. **Steps**

(1) Add the following `location` block into `/etc/nginx/sites-available/default`.

```
server {
  ...

  location ~ ^/~(.*?)/(.*) {
    alias /home/$1/htdocs/$2;
  }

  ...
}
```

(2) Run the following commands.

```
$ chmod 755 /home/b12902110
$ mkdir /home/b12902110/htdocs
```

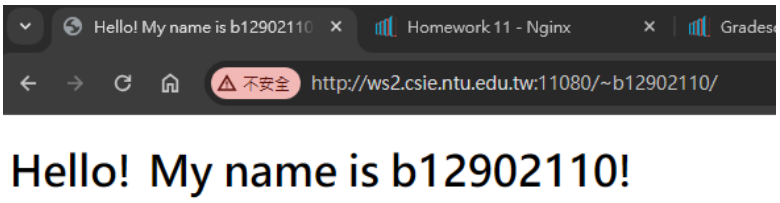
(3) Create `/home/b12902110/htdocs/index.html` as the following.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello! My name is b12902110!</title>
  </head>
  <body>
    <h1>Hello! My name is b12902110!</h1>
  </body>
</html>
```

(4) Reload the nginx service.

```
$ sudo systemctl reload nginx.service
```

Result



References

- nginx user public home without - Stack Overflow
- Beginner's Guide
- Module ngx_http_core_module

10. **Steps**

(1) Add the following `location` block into `/etc/nginx/sites-available/default`.

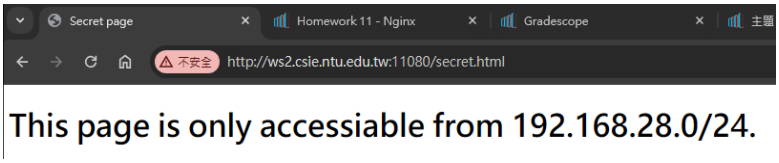
```
server {
  ...

  location = /secret.html {
    allow 192.168.28.0/24;
    deny all;
  }

  ...
}
```

(2) Reload the nginx service.

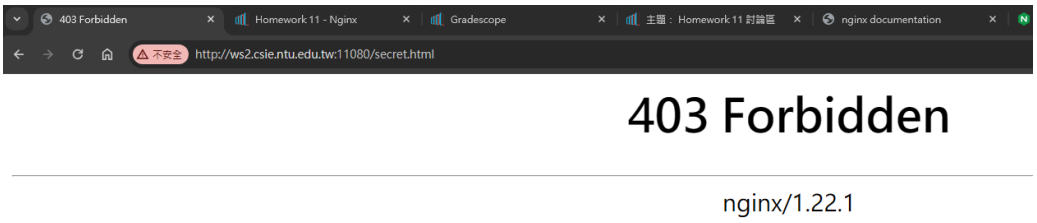
```
$ sudo systemctl reload nginx.service
```

Result

Before restriction:



After restriction:



References

- Module ngx_http_access_module

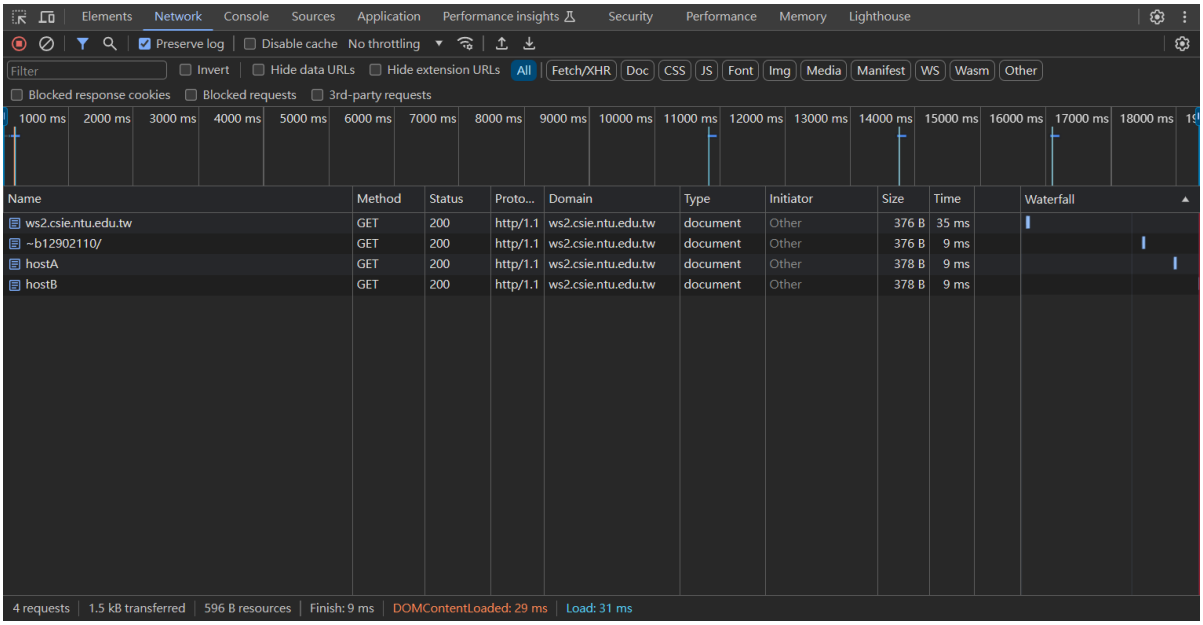
11. **Steps**

View the last few lines of `/var/log/nginx/access.log/`

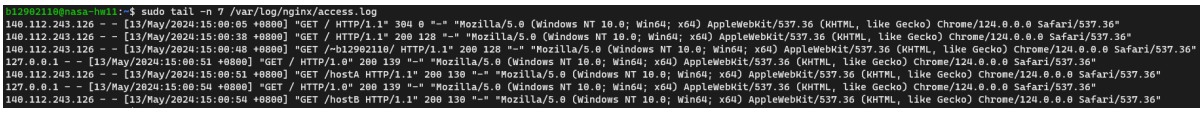
```
$ sudo tail /var/log/nginx/access.log
```

Result

Chrome DevTools:



`/var/log/nginx/access.log`:



References

- Configuring Logging | NGINX Documentation

12. (a) TLS uses PKI in the handshake process to authenticate identities of the communicating hosts.

    (b) A CA is a trusted third party that issues and signs digital certificates.

    References

    - Transport Layer Security - Wikipedia
    - Public key infrastructure - Wikipedia

    (c) **Steps (Server)**

    (1) Create an CA key and certificate.

```
$ openssl req -new -x509 -noenc -days 365000 -newkey rsa:2048 \
    -keyout ca-key.pem -out ca-cert.pem \
    -subj "/C=TW/O=NTU CSIE/OU=NASA/CN=b12902110 Root CA"
```

    (2) Create a server key and certificate request.

```
$ openssl req -new -noenc -newkey rsa:2048 \
    -keyout server-key.pem -out server-req.pem \
    -subj "/C=TW/O=NTU CSIE/OU=NASA/CN=nasa-hw11" \
    -addext "subjectAltName = DNS:*.csie.ntu.edu.tw"
```

    (3) Sign the server certificate.

```
$ openssl x509 -req -days 365000 -copy_extensions copyall \
    -in server-req.pem -out server-cert.pem \
    -CA ca-cert.pem -CAkey ca-key.pem
Certificate request self-signature ok
subject=C = TW, O = NTU CSIE, OU = NASA, CN = nasa-hw11
```

    (4) Install the certificate and key to `/etc/nginx`.

```
$ sudo cp server-key.pem server-cert.pem /etc/nginx
$ sudo chown www-data:www-data /etc/nginx/server-key.pem
```

    (5) Add the following directives in `/etc/nginx/sites-available/default`.

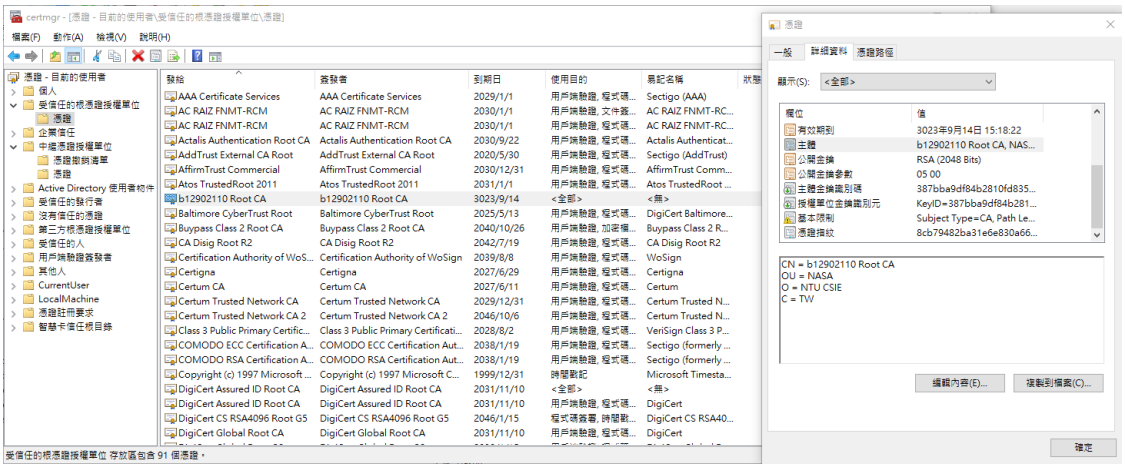
```
server {
  ...

  listen 443 ssl default_server;
  listen [::]:443 ssl default_server;
  ssl_certificate server-cert.pem;
  ssl_certificate_key server-key.pem;

  ...
}
```

    (6) Reload the nginx service.

```
$ sudo systemctl reload nginx.service
```

Steps (Windows Client)

Run `certmgr.msc`, and install `ca-cert.pem` to "Trusted Root Certification Authorities".



Result

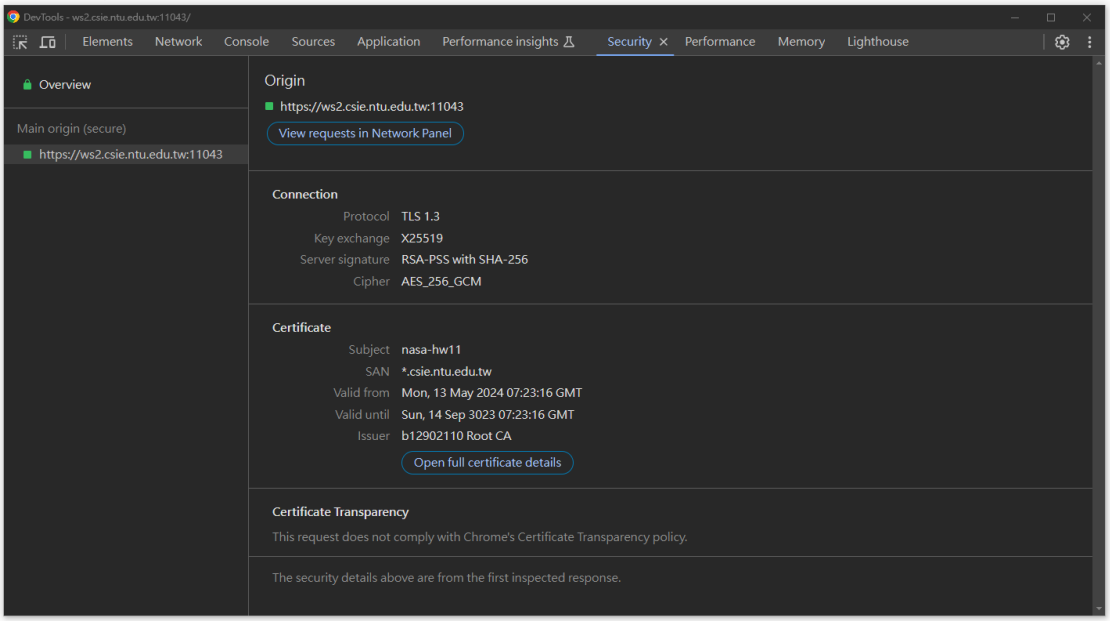
Certificates:

```
b12902110@nasa-hw11:~/certs$ openssl x509 -in ca-cert.pem -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            0b:20:e8:fd:64:1a:04:4b:80:a4:3b:0e:56:f9:cd:6c:89:9f:1c:6e
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = TW, O = NTU CSIE, OU = NASA, CN = b12902110 Root CA
        Validity
            Not Before: May 13 07:18:22 2024 GMT
            Not After : Sep 14 07:18:22 3023 GMT
        Subject: C = TW, O = NTU CSIE, OU = NASA, CN = b12902110 Root CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:ca:67:af:ae:97:52:68:0c:19:5f:73:e3:6f:e2:
                    de:f7:4f:23:6d:eb:61:5c:9d:25:ed:d0:a6:db:e2:
                    d1:02:d6:f9:bb:ea:53:08:af:14:05:4b:aa:ae:ab:
                    a8:a7:96:dc:cd:e0:2b:a2:24:e6:f1:c6:c1:ba:02:
                    3c:bd:8f:4f:37:b7:87:89:3a:fe:de:ba:2a:8b:5e:
                    d0:20:24:99:1d:05:cb:09:7c:44:ad:1b:a4:1f:a1:
                    df:1c:84:7f:8b:83:27:88:35:72:29:a6:0e:e1:07:
                    75:38:ae:e1:e4:e2:b5:86:79:a2:00:c1:5e:e8:d6:
                    ce:39:82:54:82:5a:af:8c:e1:50:a2:39:2c:39:50:
                    50:eb:73:e0:c1:c3:9c:b0:33:a4:bc:ac:1f:03:82:
                    2e:65:3b:40:67:cc:c1:26:3c:f5:67:6c:0e:21:85:
                    1f:c2:8b:e3:ed:a1:12:f7:8c:5d:4a:96:b1:43:c3:
                    cc:6e:6d:51:0c:69:d0:8a:a3:16:ce:6a:45:98:20:
                    03:bc:d7:b1:c6:47:97:75:c5:f5:20:16:6c:9b:a7:
                    43:2d:46:f9:66:a6:3f:7f:93:36:1e:83:bb:7c:29:
                    1b:bc:b2:91:dc:45:01:a6:a7:21:fa:3e:55:d9:32:
                    92:3a:ad:fe:aa:cf:98:04:38:18:45:90:1e:ea:f9:
                    a3:99
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                38:7B:BA:9D:F8:4B:28:10:FD:83:51:6D:43:6A:C5:0B:D1:16:F4:90
            X509v3 Authority Key Identifier:
                38:7B:BA:9D:F8:4B:28:10:FD:83:51:6D:43:6A:C5:0B:D1:16:F4:90
            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        4a:78:43:26:3d:1e:b4:9f:cb:8f:de:bb:ef:26:e6:b4:08:48:
        06:93:02:35:b2:0d:cd:94:37:89:18:a3:c2:99:88:06:ec:ec:
        f2:1d:27:a1:82:48:74:47:4c:84:02:ef:f8:a1:e3:39:a3:11:
        2b:fa:36:5a:2b:3d:8b:23:a7:43:54:1b:19:32:89:8b:29:05:
        93:b8:c5:ca:a3:93:65:56:f5:66:42:2b:e5:eb:9a:62:02:d6:
        e8:d7:e0:1e:e8:fb:53:8a:30:8a:09:b2:f6:d2:ab:9b:4e:2c:
        f6:db:b0:40:f5:30:b6:99:bc:1d:5b:e5:47:a9:ca:54:87:55:
        4f:ef:fa:ca:48:c9:a2:f4:03:9e:87:32:f8:af:49:40:09:30:
        14:da:e7:7b:87:0f:cc:2d:68:d3:36:4a:38:82:f0:be:2a:bf:
        6d:13:61:a0:70:d3:80:84:ec:5d:44:d8:e5:34:8b:1c:93:27:
        c8:84:25:60:d6:e2:0b:d1:ee:39:98:c2:4f:8d:41:fc:a7:1c:
        6c:fa:8a:e2:68:e5:8a:a2:52:a3:57:87:96:73:cb:de:e1:95:
        50:36:5d:5e:b8:07:18:24:33:77:61:a5:ca:1b:11:09:ac:e7:
        1b:9d:92:ac:56:a7:75:4a:2e:03:29:95:ac:f6:bd:99:a7:84:
        1f:c2:d5:bd
```

```
b12902110@nasa-hw11:~/certs$ openssl x509 -in server-cert.pem -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            2f:80:09:0a:6a:04:1c:4c:bf:bb:9c:dd:79:8b:52:9e:87:27:bc:ee
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = TW, O = NTU CSIE, OU = NASA, CN = b12902110 Root CA
        Validity
            Not Before: May 13 07:23:16 2024 GMT
            Not After : Sep 14 07:23:16 3023 GMT
        Subject: C = TW, O = NTU CSIE, OU = NASA, CN = nasa-hw11
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:ca:86:6e:c1:a0:63:ac:7c:15:41:1d:48:cd:f1:
                    60:b0:4f:c5:d3:d0:ba:67:15:4a:e3:51:bf:2a:39:
                    dc:8d:17:59:95:69:f1:c7:46:b9:f2:01:a0:9f:a7:
                    d1:4a:80:19:0e:95:17:b1:32:89:6e:17:39:69:e4:
                    a0:06:fa:4c:66:6c:4c:92:9e:e2:0a:bd:04:54:1c:
                    1d:98:ff:a3:e5:13:9a:15:3c:13:ab:17:48:ca:b1:
                    f8:7b:41:24:ca:39:6d:b4:c8:e5:67:b8:8d:53:aa:
                    72:21:9d:4a:8a:0e:ab:96:aa:d4:e0:fe:67:a7:06:
                    64:f8:68:af:8c:54:ff:12:e2:f2:c1:1a:f0:b0:d4:
                    d3:bd:2c:61:cf:84:6f:33:a1:90:89:aa:cd:c2:d8:
                    67:a0:53:94:28:cd:fa:6a:a7:67:ef:ba:3c:1d:41:
                    62:49:d2:4d:0d:b3:c7:d8:48:ae:1d:74:f8:66:59:
                    70:65:a0:e4:bd:14:63:a6:27:87:56:fa:22:a5:ab:
                    c6:4b:1a:d2:9a:7c:c9:86:bd:e1:0b:a2:f7:5a:ac:
                    05:65:c4:ce:2d:6c:96:e8:89:de:78:42:ae:6a:0f:
                    d2:10:8e:f3:da:df:03:76:38:1a:36:b0:ef:e1:fb:
                    2b:cf:4b:98:4b:34:dd:a0:89:4c:bd:94:24:23:a3:
                    5d:11
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:*.csie.ntu.edu.tw
            X509v3 Subject Key Identifier:
                1B:28:03:67:19:9B:67:0D:AB:F4:15:63:42:80:FC:2F:DD:DC:1D:F7
            X509v3 Authority Key Identifier:
                38:7B:BA:9D:F8:4B:28:10:FD:83:51:6D:43:6A:C5:0B:D1:16:F4:90
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        be:01:87:9e:13:11:c7:bf:77:f2:54:cd:94:54:41:dc:dd:20:
        a8:5b:f7:60:bf:0f:85:22:32:f0:2b:85:1e:b9:79:46:22:ad:
        11:d8:27:26:40:ef:0c:e2:37:88:41:19:3a:6c:da:b0:9a:a8:
        34:21:32:28:c5:e0:08:19:e6:f1:40:14:0a:b3:63:78:91:98:
        94:1b:d6:f5:34:e4:36:42:35:8b:ca:cf:11:a6:91:7d:a3:f0:
        01:f0:a5:7b:a8:a2:bd:7a:c4:71:c2:ee:a0:15:71:6e:45:66:
        a8:50:bb:b1:37:1d:23:82:d9:b4:0a:92:c1:5c:37:2e:2d:7c:
        ec:7c:a7:7f:58:a8:9b:3c:4d:9d:fd:49:e7:c7:c1:5a:11:b9:
        87:f8:9b:a3:ce:c1:f9:3e:9f:46:84:58:a3:e5:9d:fa:b2:0b:
        9c:24:91:43:5d:3b:e0:b6:e8:d8:76:1a:59:ee:48:a6:c0:e2:
        47:c1:9d:c3:cc:14:6e:8f:e5:f6:d9:19:e7:73:ed:1c:32:84:
        f8:7a:92:74:96:10:15:80:d9:a8:95:28:3d:c9:6d:cb:c2:cb:
        6e:29:4c:52:ea:45:d9:11:1b:06:65:5a:30:11:9e:c3:bd:23:
        e8:80:f6:bb:3e:b5:36:3b:0c:c5:28:bc:b8:2e:02:4c:cf:86:
        b9:a0:31:f1
```

Browser:

Hello! My name is b12902110!



References

- Create Self-Signed Certificates and Keys with OpenSSL —MariaDB Documentation
- /docs/man3.0/man1/openssl-req.html
- /docs/man3.0/man1/openssl-x509.html
- certificates - Provide subjectAltName to openssl directly on the command line - Information Security Stack Exchange

Reverse Proxy

13. **Steps**

(1) Create `/etc/nginx/sites-available/hostA` as the following.

```
server {
  listen 8888 default_server;
  listen [::]:8888 default_server;

  root /var/www/hostA;
  index index.html;

  server_name hostA;

  location / {
    try_files $uri $uri/ =404;
  }
}
```

(2) Create `/etc/nginx/sites-available/hostB` as the following.

```
server {
  listen 9999 default_server;
  listen [::]:9999 default_server;

  root /var/www/hostB;
  index index.html;

  server_name hostB;

  location / {
    try_files $uri $uri/ =404;
  }
}
```

(3) Enable both `hostA` and `hostB`.

```
$ sudo ln -s /etc/nginx/sites-available/hostA \
    /etc/nginx/sites-enabled/hostA
$ sudo ln -s /etc/nginx/sites-available/hostB \
    /etc/nginx/sites-enabled/hostB
```

(4) Create `/var/www/hostA/index.html` as the following.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Host A index.html</title>
  </head>
  <body>
    <h1>Hi! This is host A.</h1>
  </body>
</html>
```

(5) Create `/var/www/hostB/index.html` as the following.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Host B index.html</title>
  </head>
  <body>
    <h1>Hi! This is host B.</h1>
  </body>
</html>
```

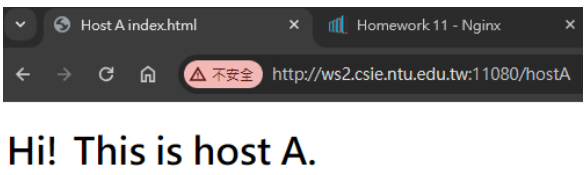
(6) Add `/hostA` and `/hostB` location blocks to `/etc/nginx/sites-available/default`.

```
server {
  ...

  location /hostA {
    proxy_pass http://127.0.0.1:8888/;
  }

  location /hostB {
    proxy_pass http://127.0.0.1:9999/;
  }

  ...
}
```

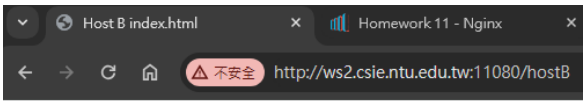
(7) Reload the nginx service.

```
$ sudo systemctl reload nginx.service
```

Result



Hi! This is host A.



Hi! This is host B.

References

- Beginner's Guide
- Module ngx_http_proxy_module