AI 雲端部署
feat. LINE Bot

Enos Chou

2022/07/13, 07/15

Enos' Steps

2022/06/10(五) 13:30~16:30	專題分享 by Enos
2022/06/17(五) 09:00~16:30	專題訂定 by Teams feat. Enos
2022/07/13(三) 09:00~16:30	AI 雲端部署(上) by Enos
2022/07/15(五) 09:00~16:30	AI 雲端部署(下) by Enos
2022/08/05(五) 09:00~16:30	專題預報 by Teams feat. Enos

Focus

- LINE Bot AI 部署
- LINE Bot 部署 + AI 部署
- TensorFlow.js AI 部署

課前準備

1. New GCP account
   a. 以三個月內新申請的帳號完成登入 https://console.cloud.google.com
   b. 申請時須信用卡，預刷 40 元不會請款

2. Ready Azure account
   a. 以班導師提供的序號儲值或新申請帳號並完成登入
   b. 序號兌換流程 https://www.microsoftazurepass.com/Home/HowTo
   c. 完成 Azure 登入 https://portal.azure.com

課前準備

3. MobaXterm Home Edition (Windows Only)
   a. 下載並安裝 MobaXterm Home Edition 最新版
   b. https://mobaxterm.mobatek.net/ > Download > Home Edition Download now

4. LINE Messaging API settings
   a. 備妥一組測試用 LINE Bot 並完成 LINE Messaging API 相關設定

5. (Optional) TensorFlow 開發環境
   a. 備妥 Python 虛擬環境，包含 TensorFlow 2.4.4 以上

LINE Bot AI 部署

LINE Bot AI 部署重點

1. 長期可用且廉價的硬體環境 ...

Why Not Cloud ?

雲端 vs 地端

To Tech

Scalability

To Biz

CAPEX vs OPEX

Which Cloud ?

Magic Quadrant for Cloud

1. Magic Quadrant for Cloud Infrastructure and Platform Services (2021/07)

2. Magic Quadrant for Cloud AI Developer Services (2021/02)

Cloud Market Share

IaaS + PaaS Market Share (2021/Q4)

LINE Bot AI 部署重點

1. 長期可用且廉價的硬體環境 ... Cloud
2. Flask as Web Server 的替代方案 ...
3. 長期可用且廉價的 SSL 網域方案 ...

Warning from Flask

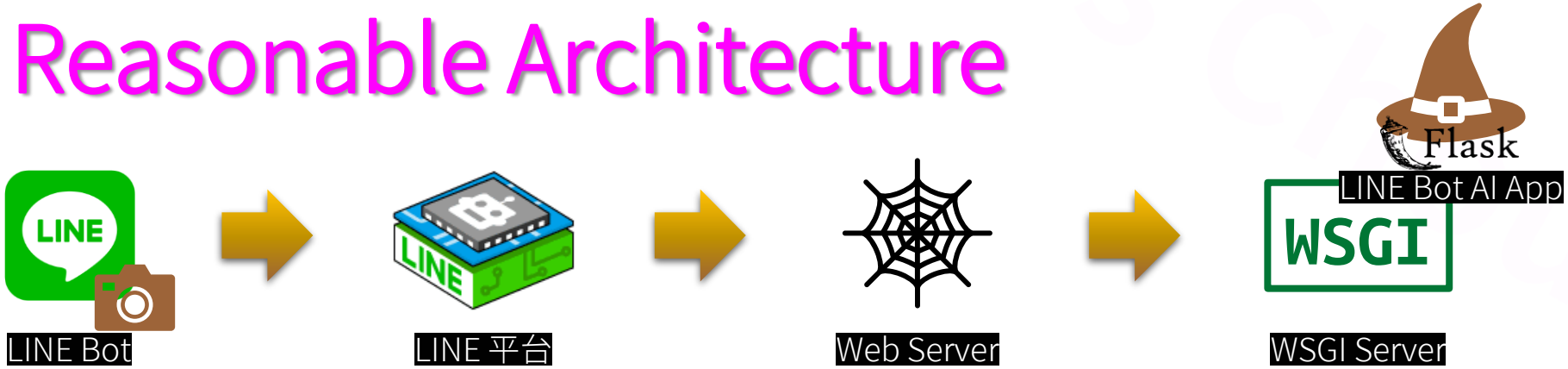
```
python3 your_module.py (app.run())
```

```
 * Serving Flask app 'somebot' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Architecture

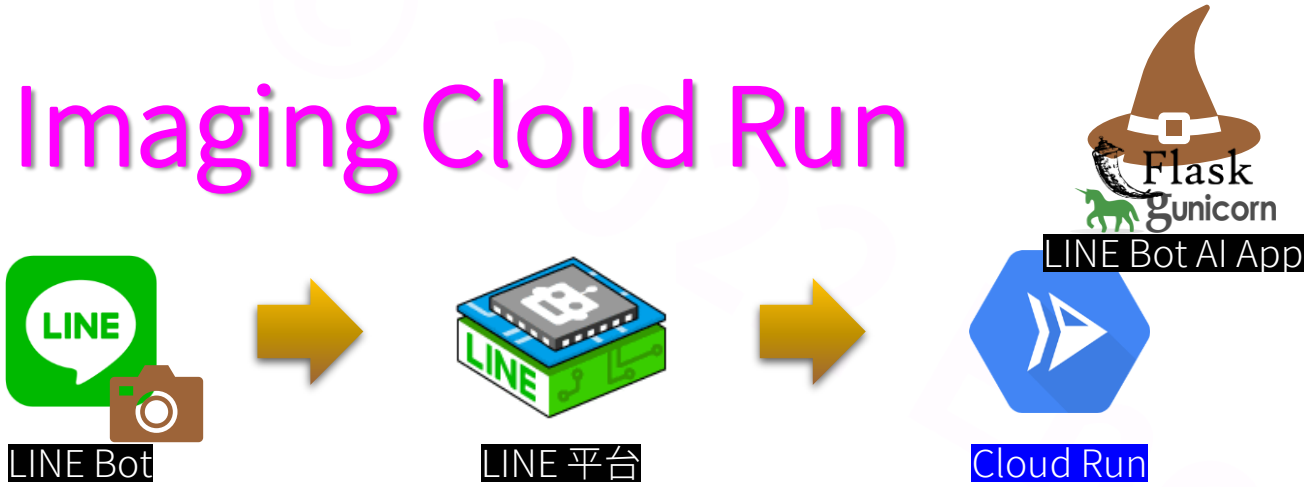
Warning from Flask

Reasonable Architecture

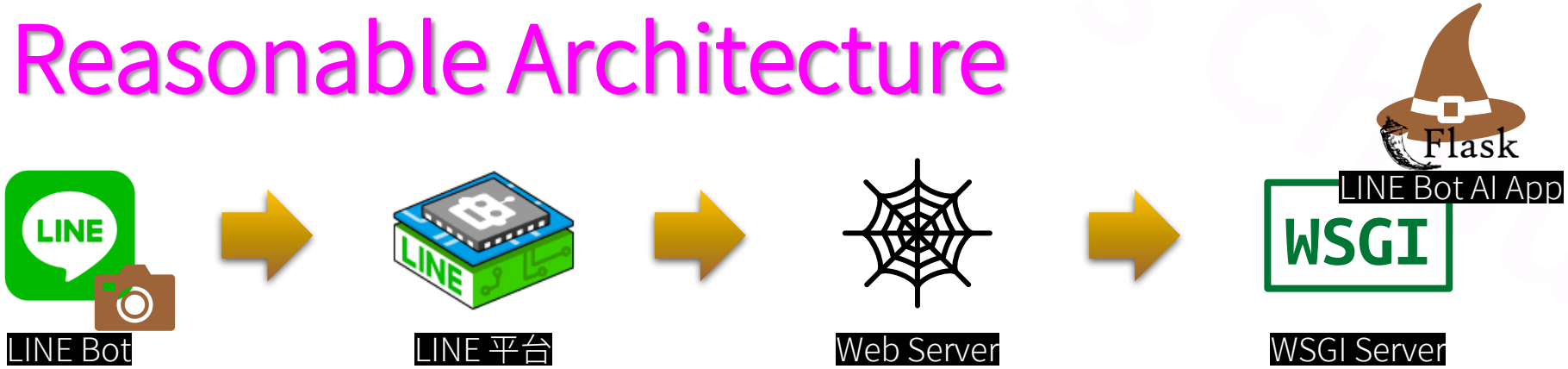


Architecture

Imaging Cloud Run

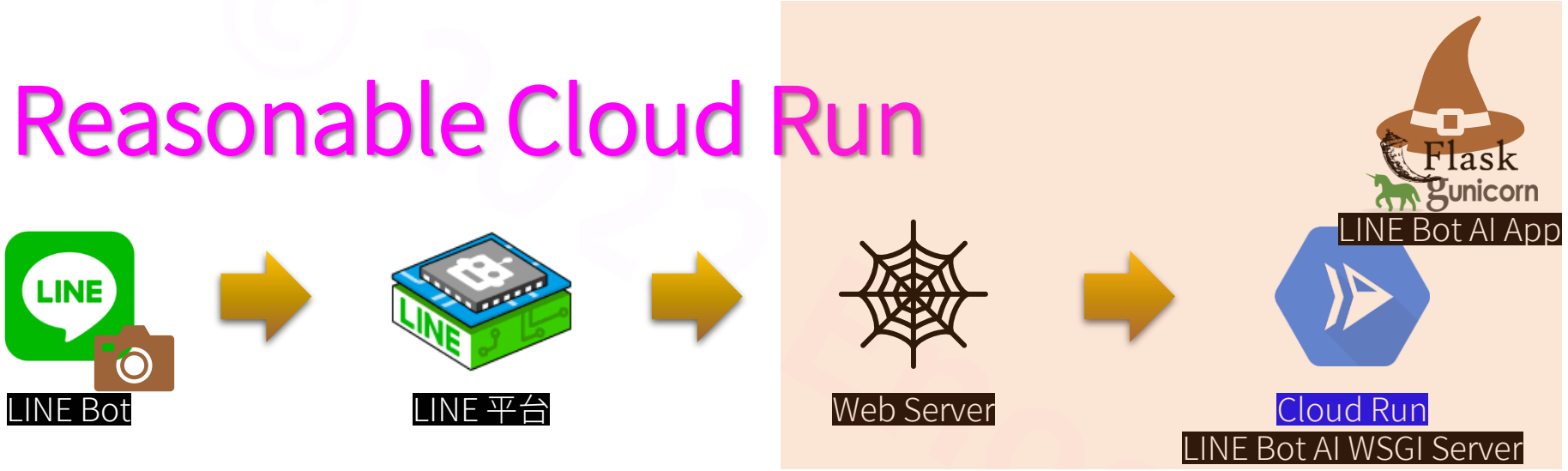


Reasonable Architecture

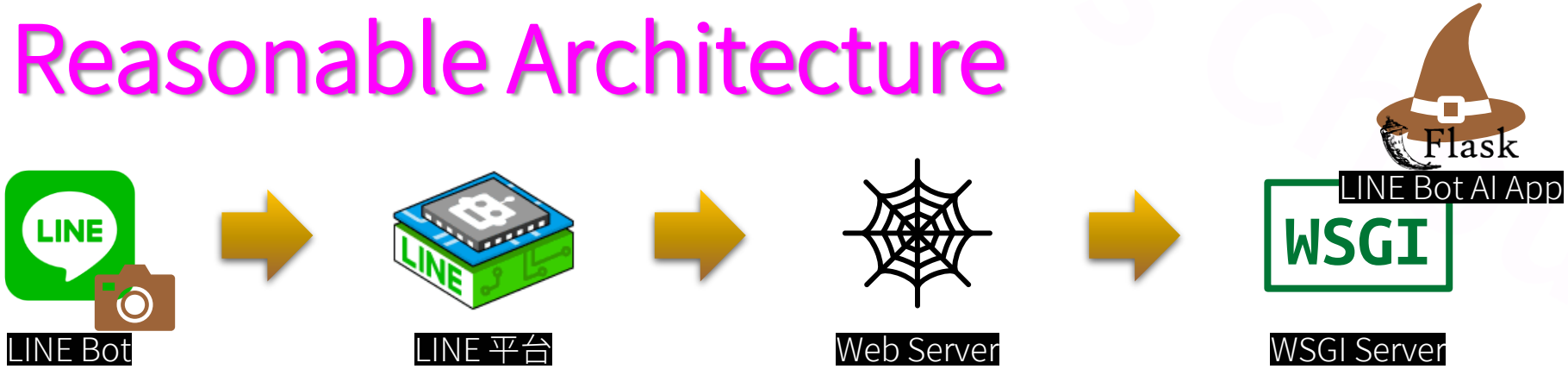


Architecture

Reasonable Cloud Run

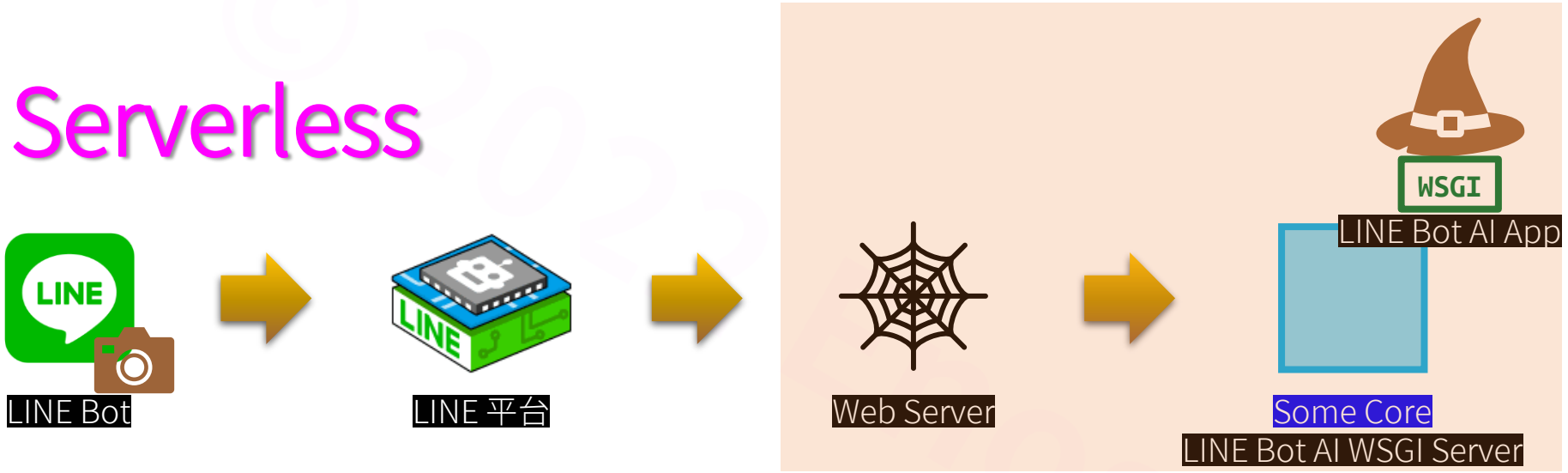


Reasonable Architecture

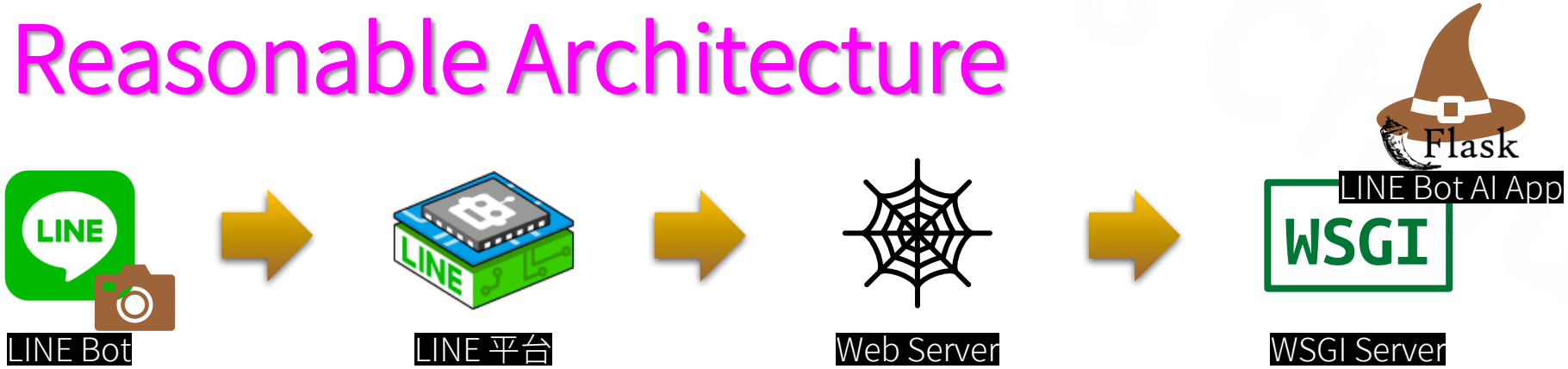


Architecture

Serverless

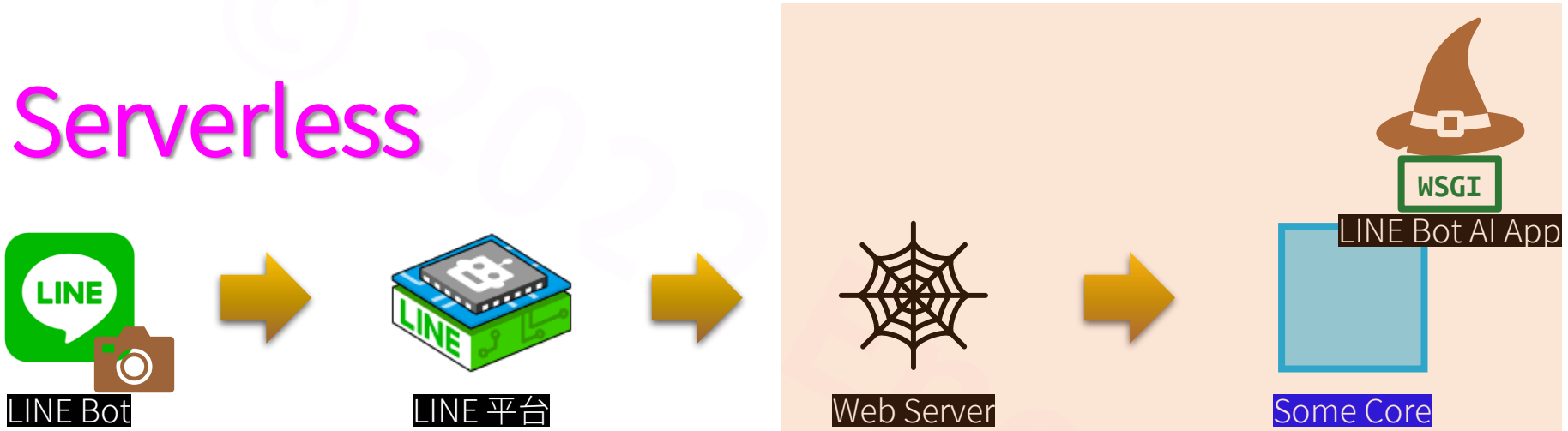


Reasonable Architecture

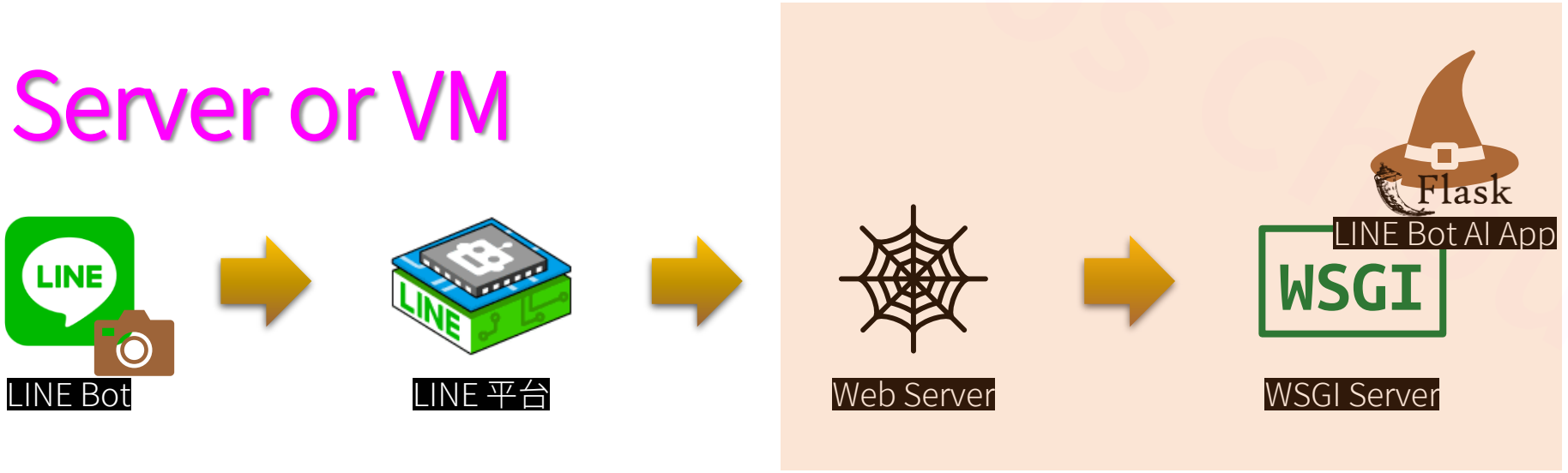


Architecture

Serverless

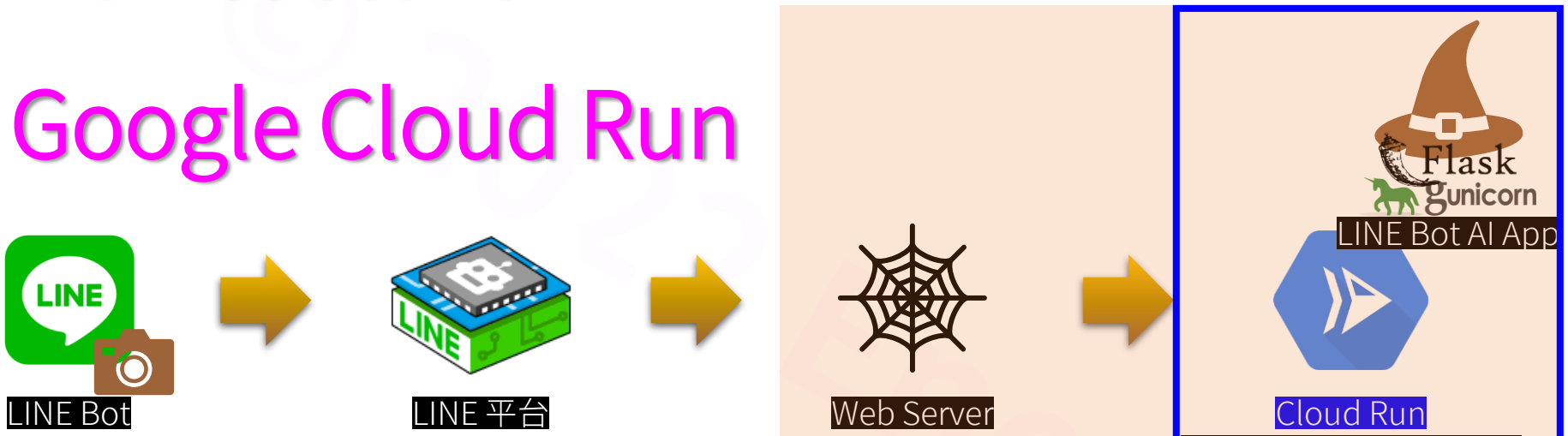


Server or VM

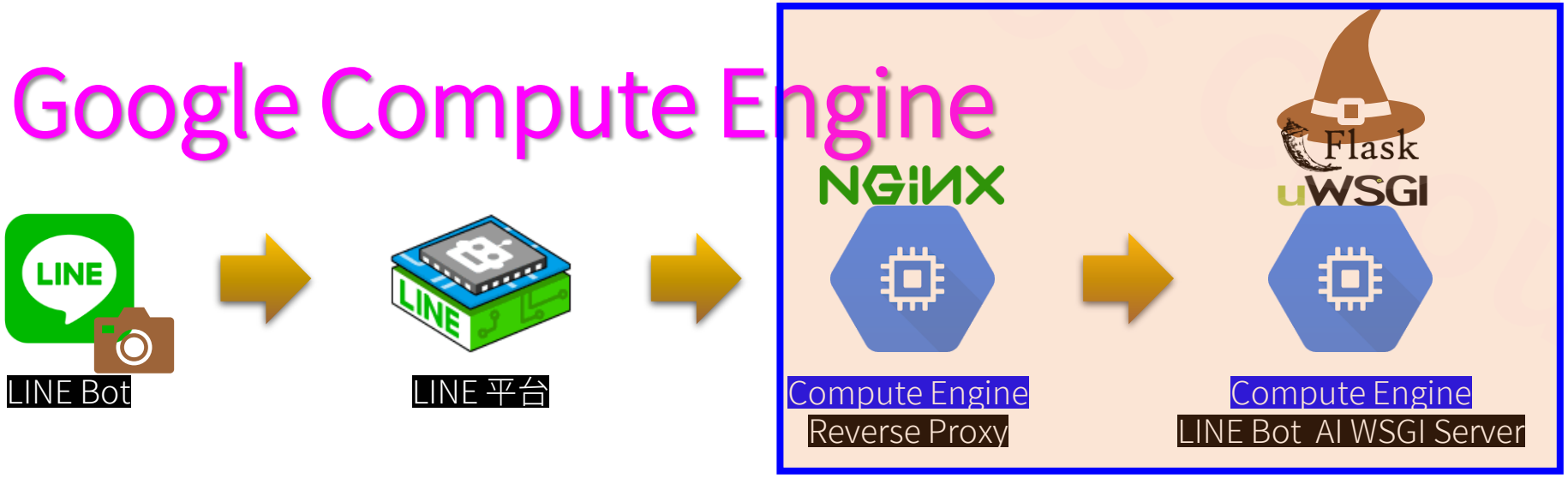


Architecture

Google Cloud Run



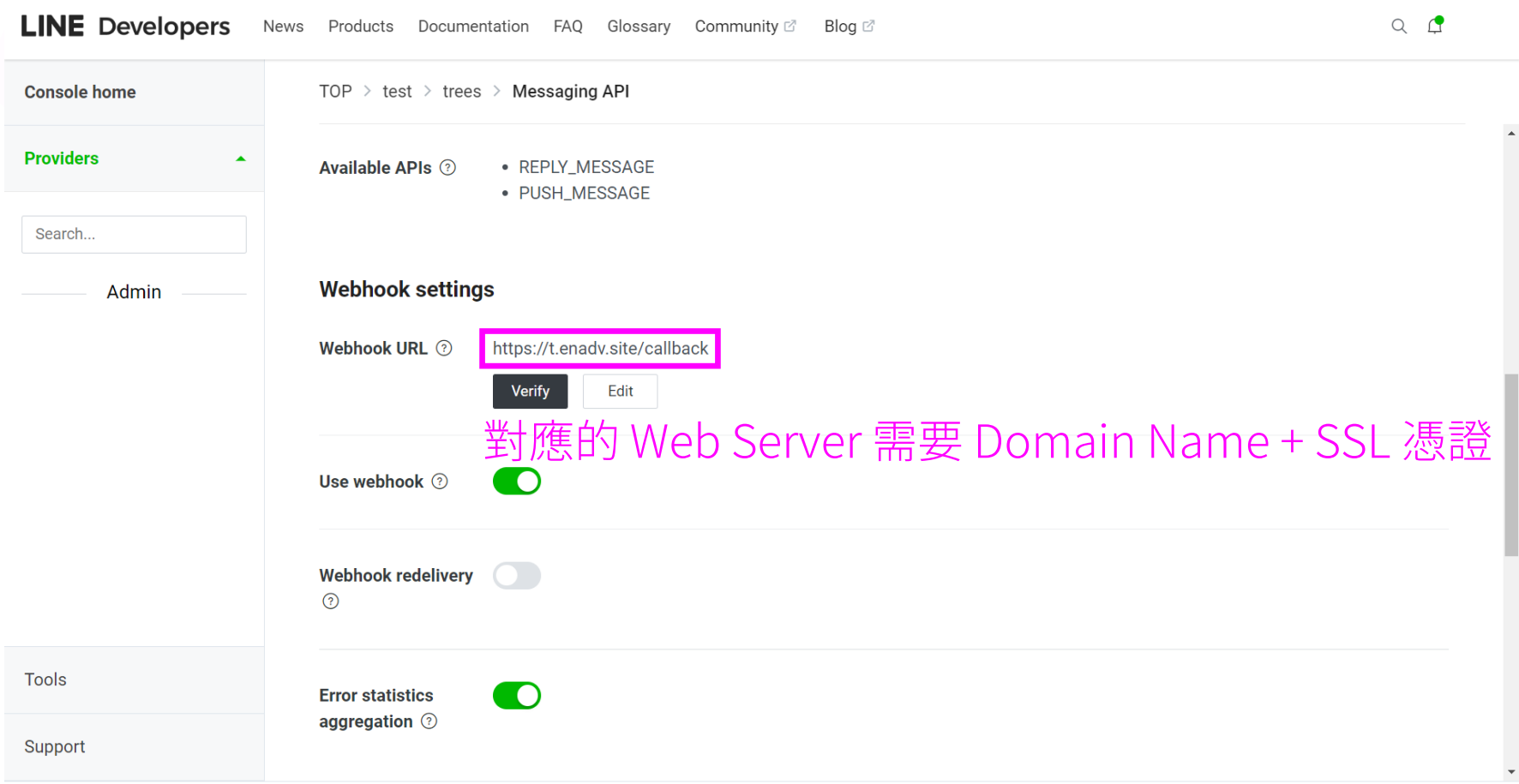
Google Compute Engine



LINE Bot AI 部署重點

1. 長期可用且廉價的硬體環境 ... Cloud
2. Flask as Web Server 的替代方案 ... Got it
3. 長期可用且廉價的 SSL 網域方案 ...

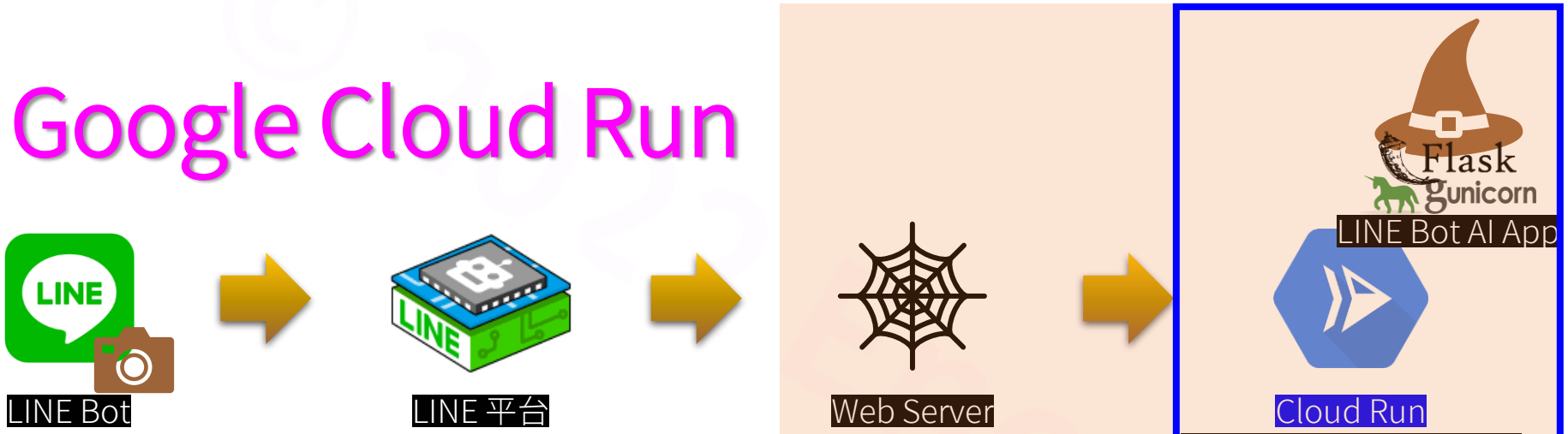
LINE Messaging API 的 SSL 需求



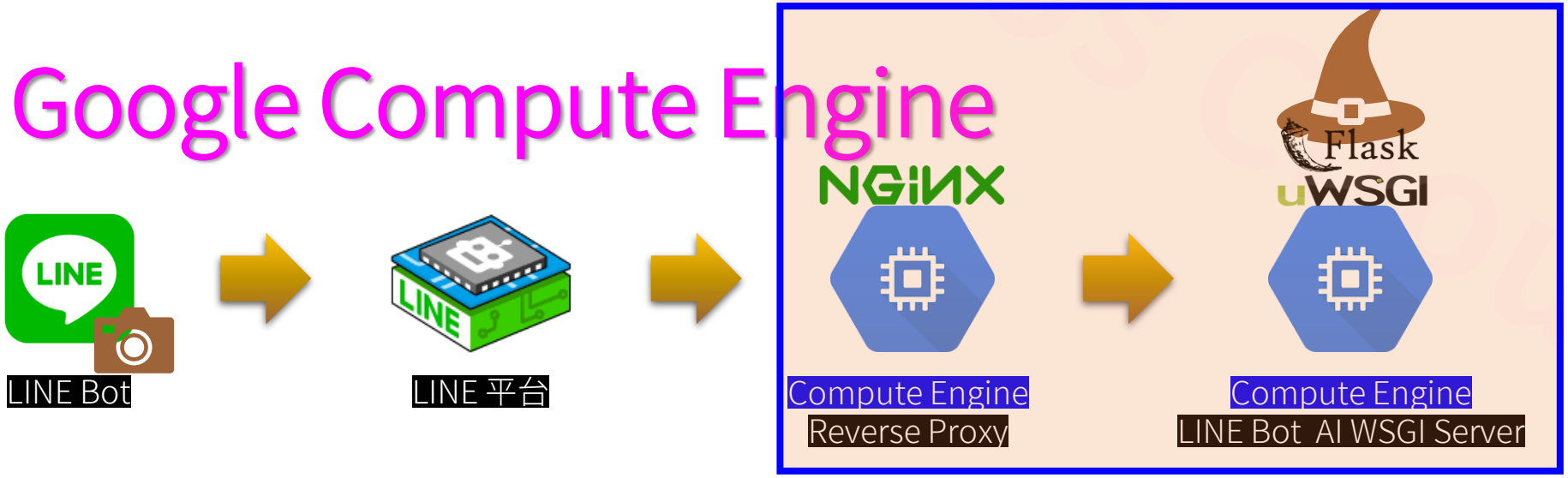
SSL: Security Socket Layer ➔ TLS: Transport Layer Security

Architecture

Google Cloud Run



Google Compute Engine



一線品牌憑證

DigiCert

- https://digicert.com/ > TLS/SSL Certificates > Basic TLS/SSL Certificates
- US$ 250/ year

免費憑證

Let’s Encrypt

- https://letsencrypt.org/zh-tw/
- 免費憑證
- 多家機構合資成立，包含 Cisco、IBM、NGINX、Heroku、LINE … 等
- 效期 3 個月
- Line Messaging API 接受

LINE Bot AI 部署重點

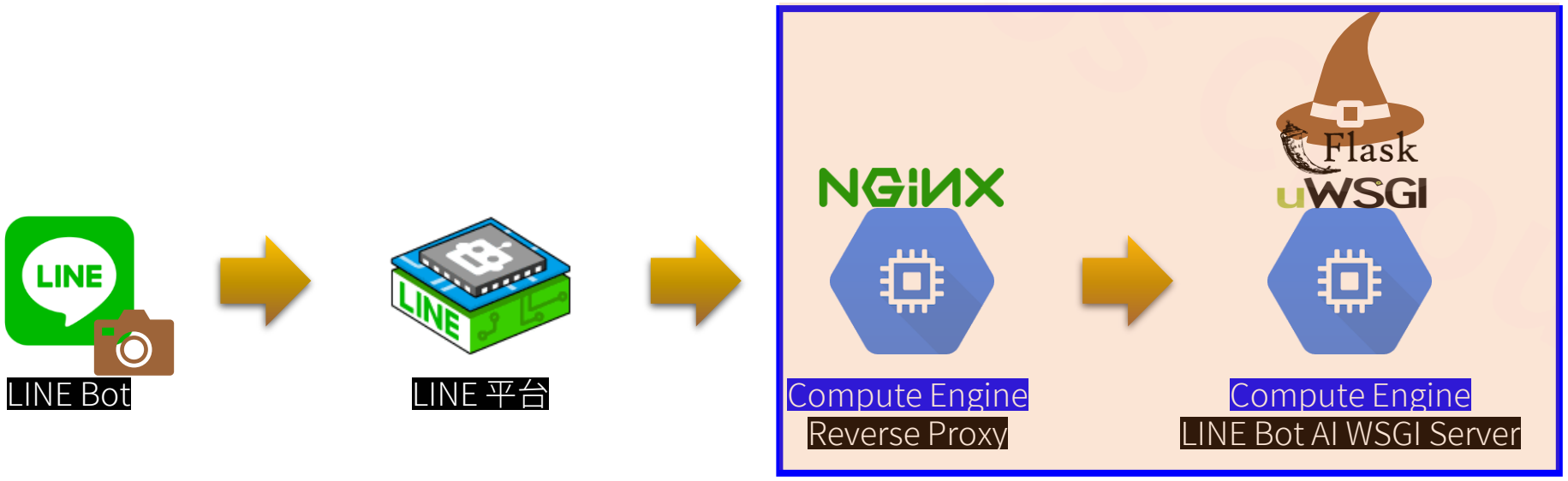
1. 長期可用且廉價的硬體環境 ... Cloud
2. Flask as Web Server 的替代方案 ... Got it
3. 長期可用且廉價的 SSL 網域方案 ... Got it

LINE Bot AI 雲端部署選項

Cloud	Server (VM)	Serverless	Note
Heroku	N/A	Heroku	http://heroku.com/ > Pricing
Google	Compute Engine	Cloud Run	*GCP 需要每3個月換帳號才能免費執行
Azure	VM	Web App	*Azure 需要 Coupon 才能免費執行
AWS	EC2	Fargate	?

Solution 1 - VM 部署

1. 長期可用且廉價的硬體環境 ... Google Compute Engine/ Azure VM
2. Flask as Web Server 的替代方案 ... NGINX + Domain + uWSGI
3. 長期可用且廉價的 SSL 網域方案 ... Let's Encrypt + Certbot

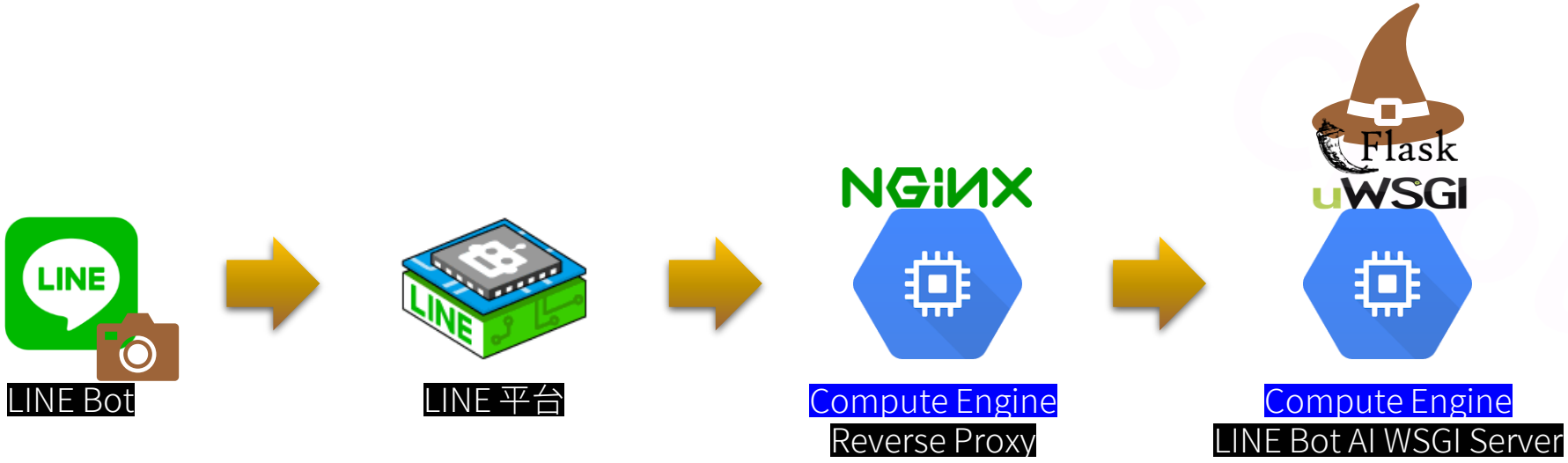


任務

1. 於 GCP 與 Azure 建立 VM
2. 以 MobaXterm 連線 VM 進行部署
3. 於 VM 架設 LINE Bot & uWSGI
4. 申購與設定 Domain
5. 於 VM 架設 NGINX
6. 於 VM 運用 Cetbot 設定憑證

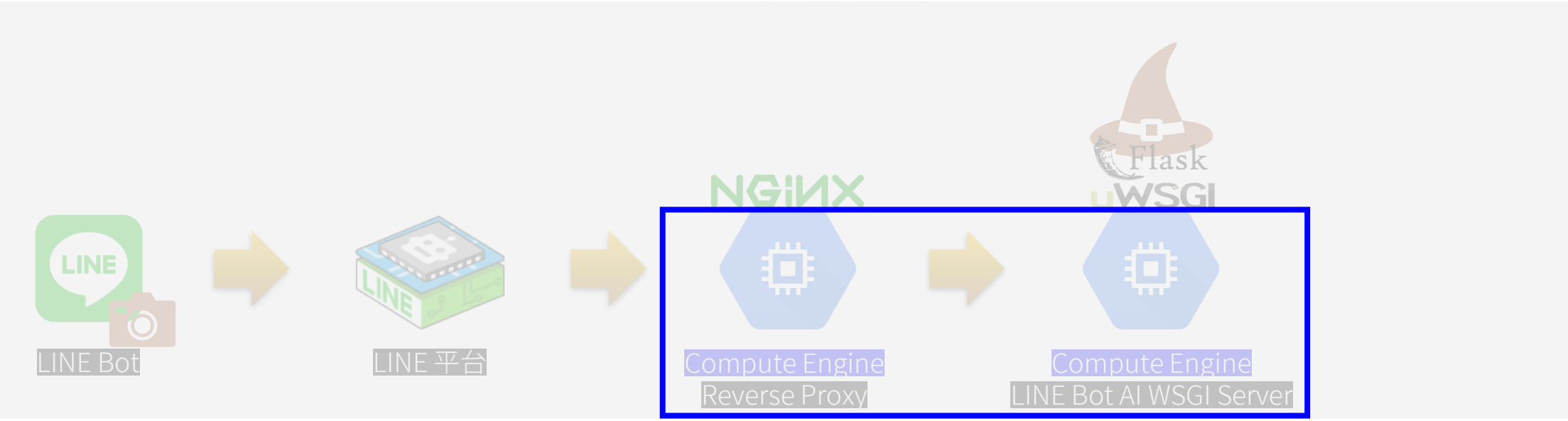
流程

架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging



LINE Bot → LINE 平台 → Compute Engine Reverse Proxy → Compute Engine LINE Bot AI WSGI Server

架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging



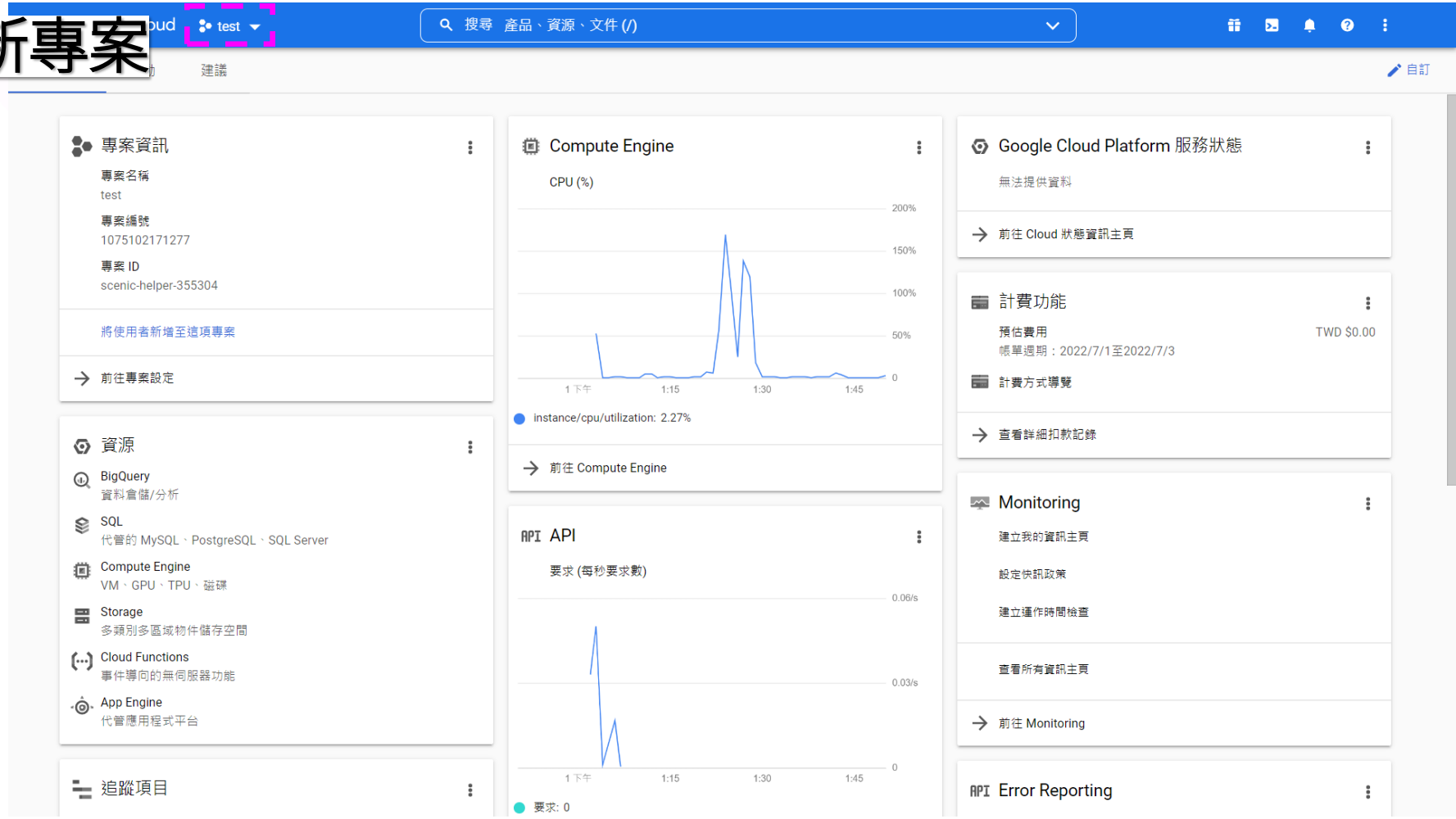
架設 VM - GCP

1. 登入

   https://console.cloud.google.com

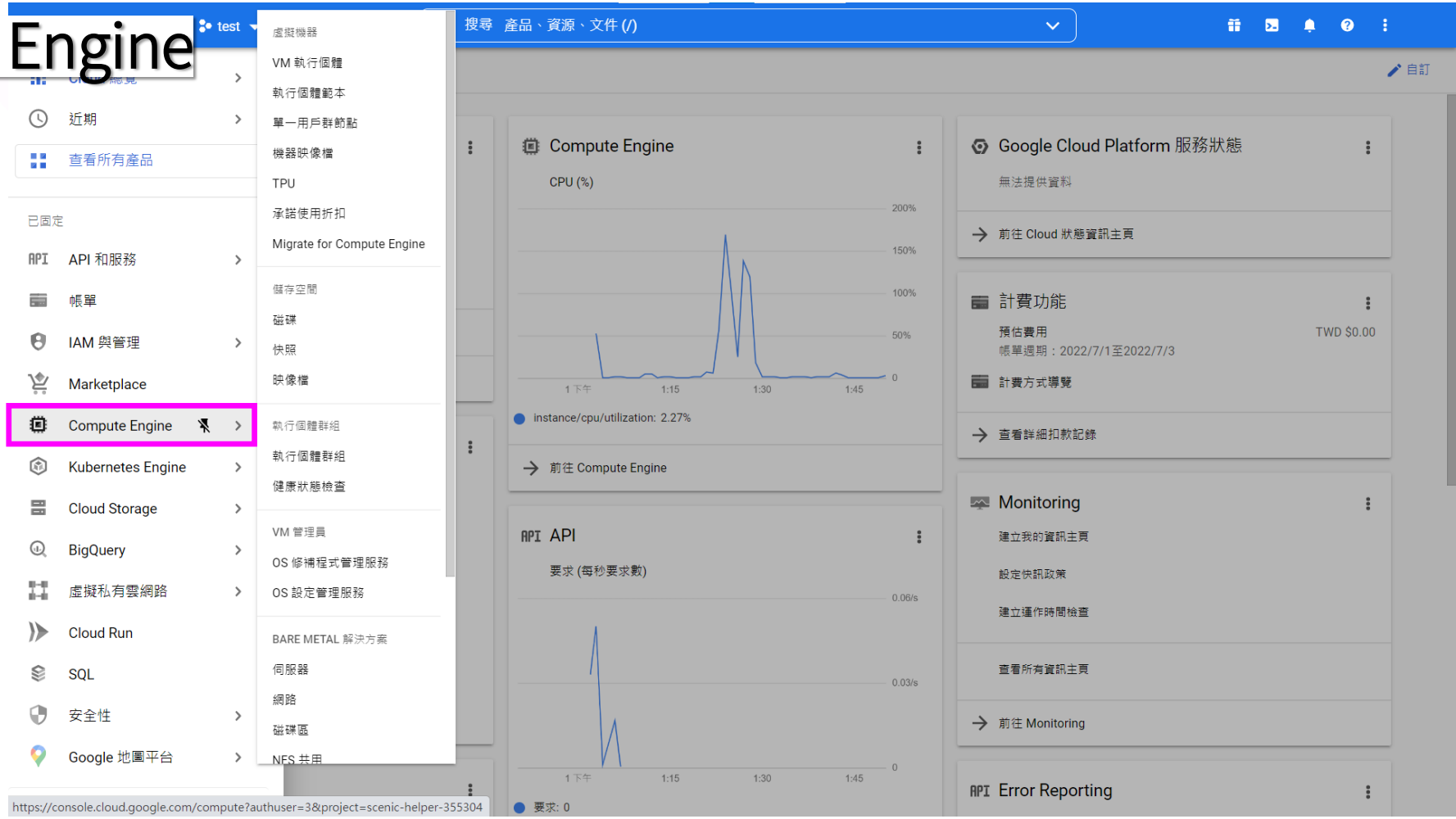
架設 VM - GCP

2. 建立並切換至新專案



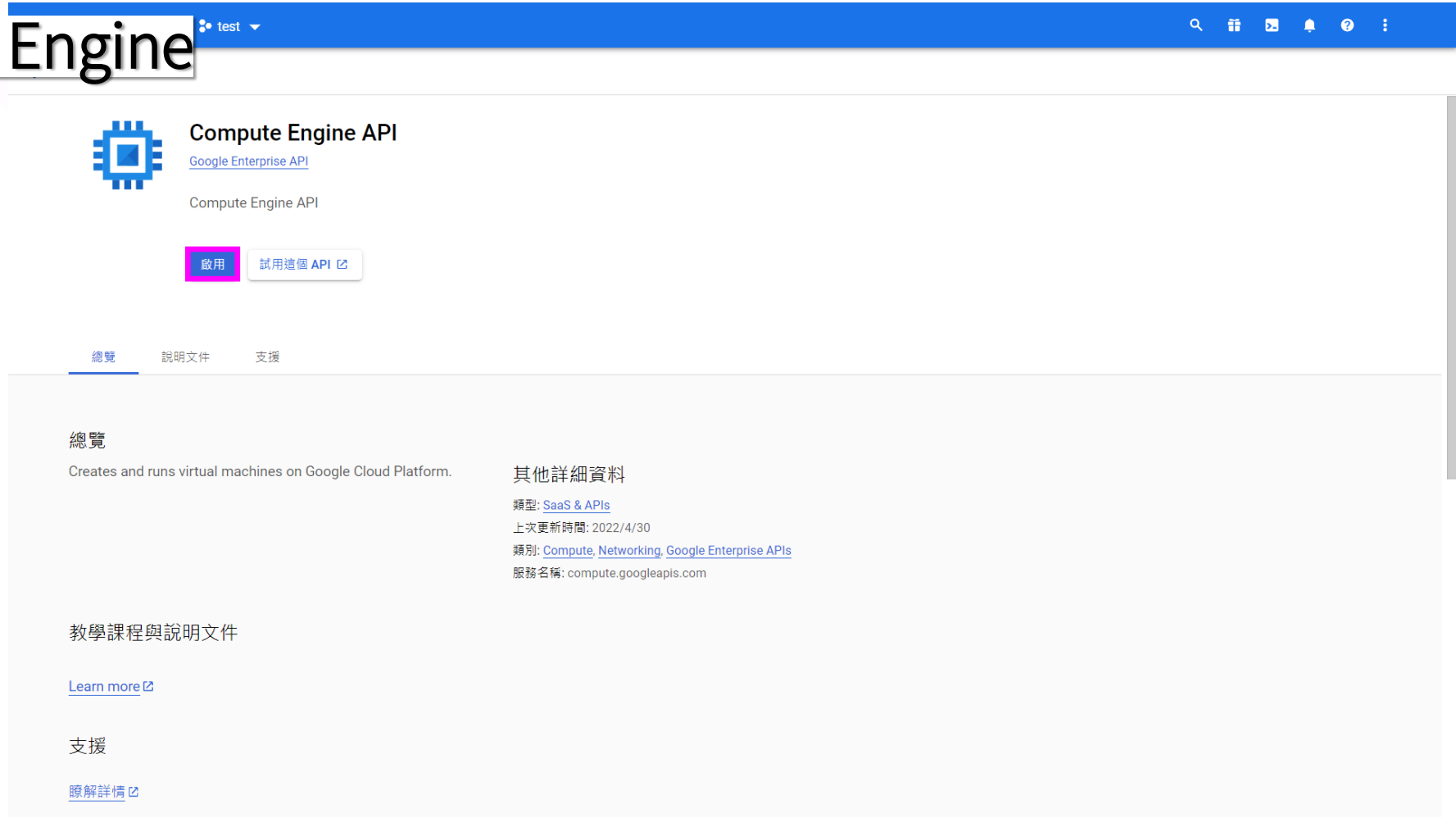
架設 VM - GCP

3. 啟用 Compute Engine



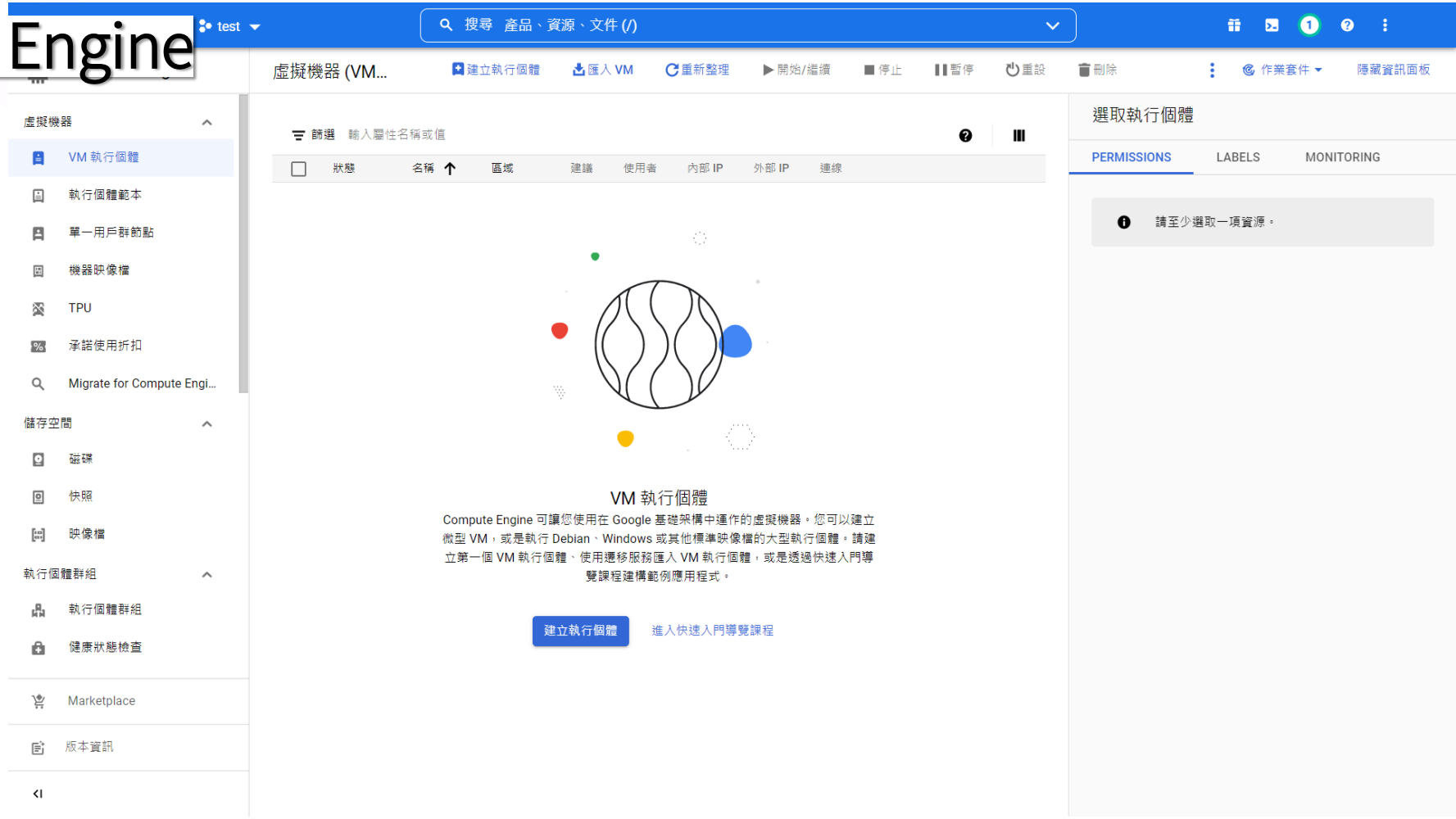
架設 VM - GCP

3. 啟用 Compute Engine



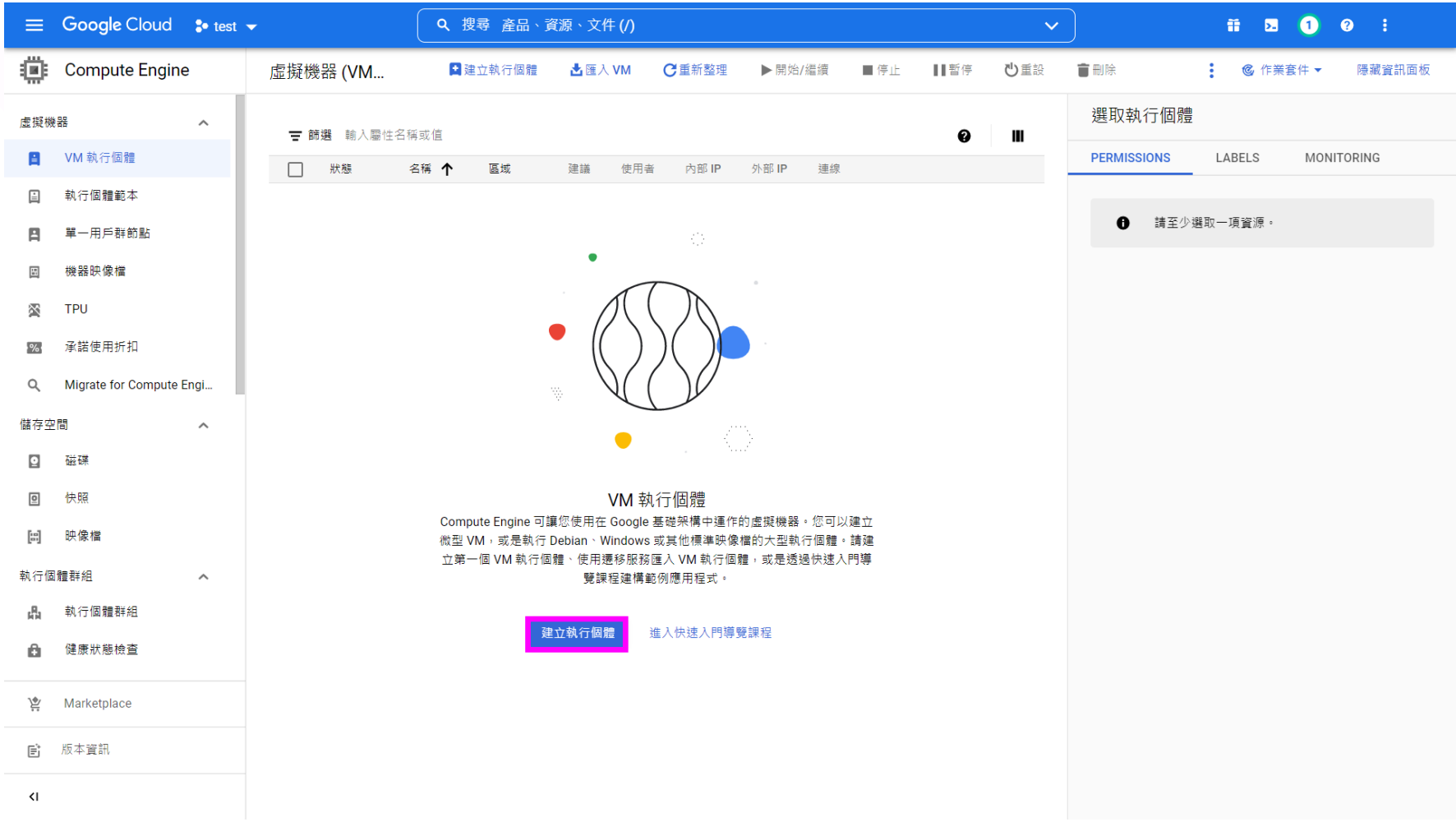
架設 VM - GCP

3. 啟用 Compute Engine



架設 VM - GCP

4. 建立 VM



架設 VM - GCP

4. 建立 VM

GCP 一律免費方案 (Free Tier)

https://cloud.google.com/ > GCP 定價 > 20 項以上的產品

Compute Engine

每月 1 個 e2-micro 執行個體

- 2 個 vCPU
- 1GB 記憶體
- 地區限制：奧勒岡州： us-west1、愛荷華州：us-central1、南卡羅來納州：us-east1

每月 30GB 標準永久磁碟

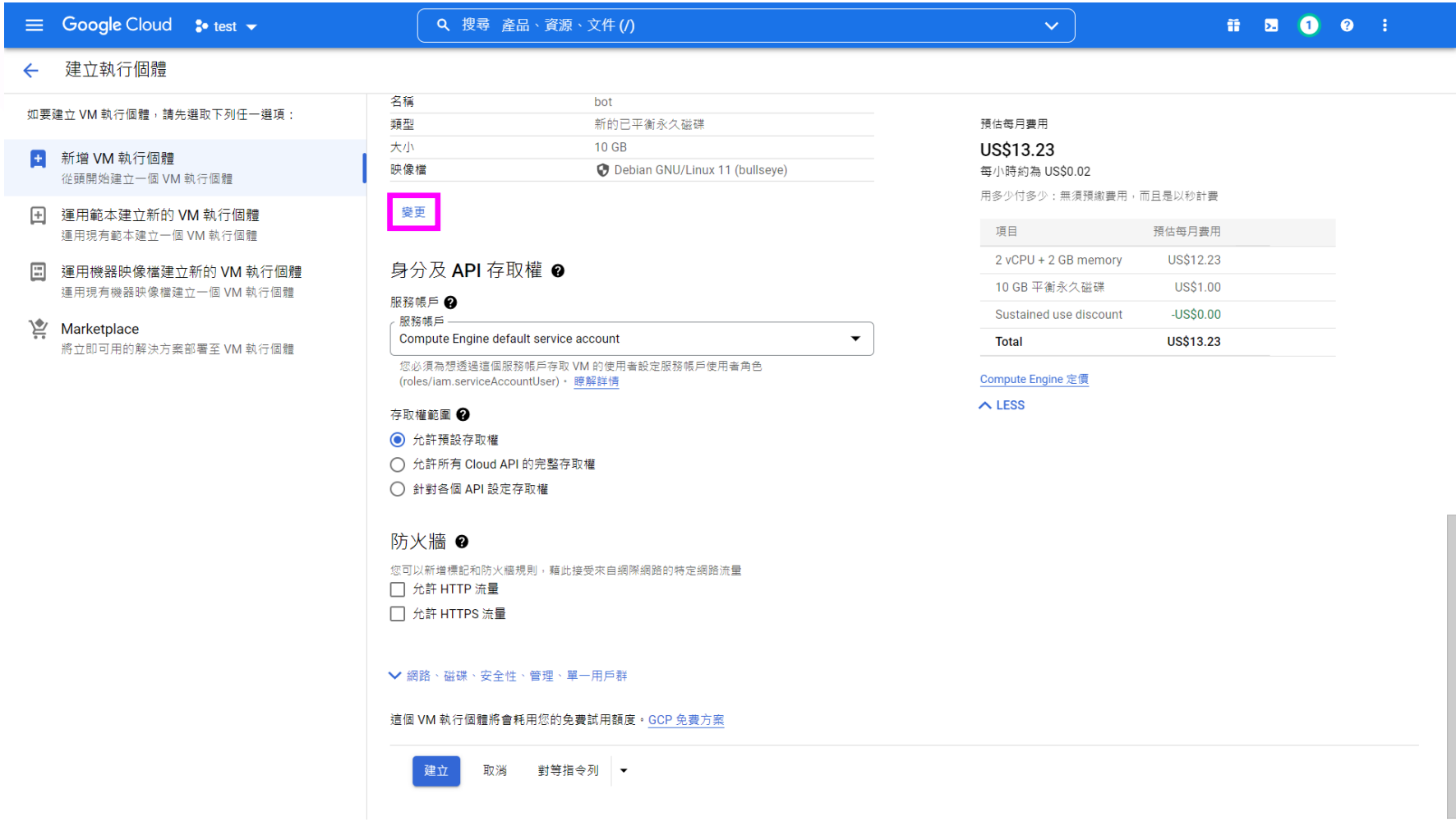
每月 1GB 網路輸出流量

注意：預設為**已平衡的永久磁碟**。移除 VM 後需個別移除磁碟

注意：VM 停止後，VM 不收費但磁碟與外部 IP 仍會收費

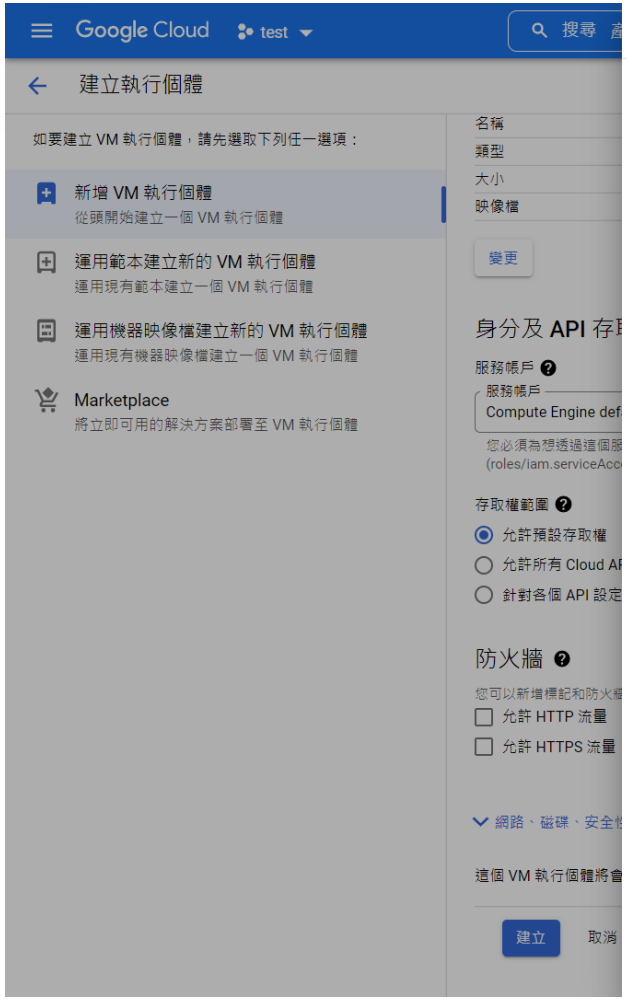
架設 VM - GCP

4. 建立 VM



架設 VM - GCP

4. 建立 VM



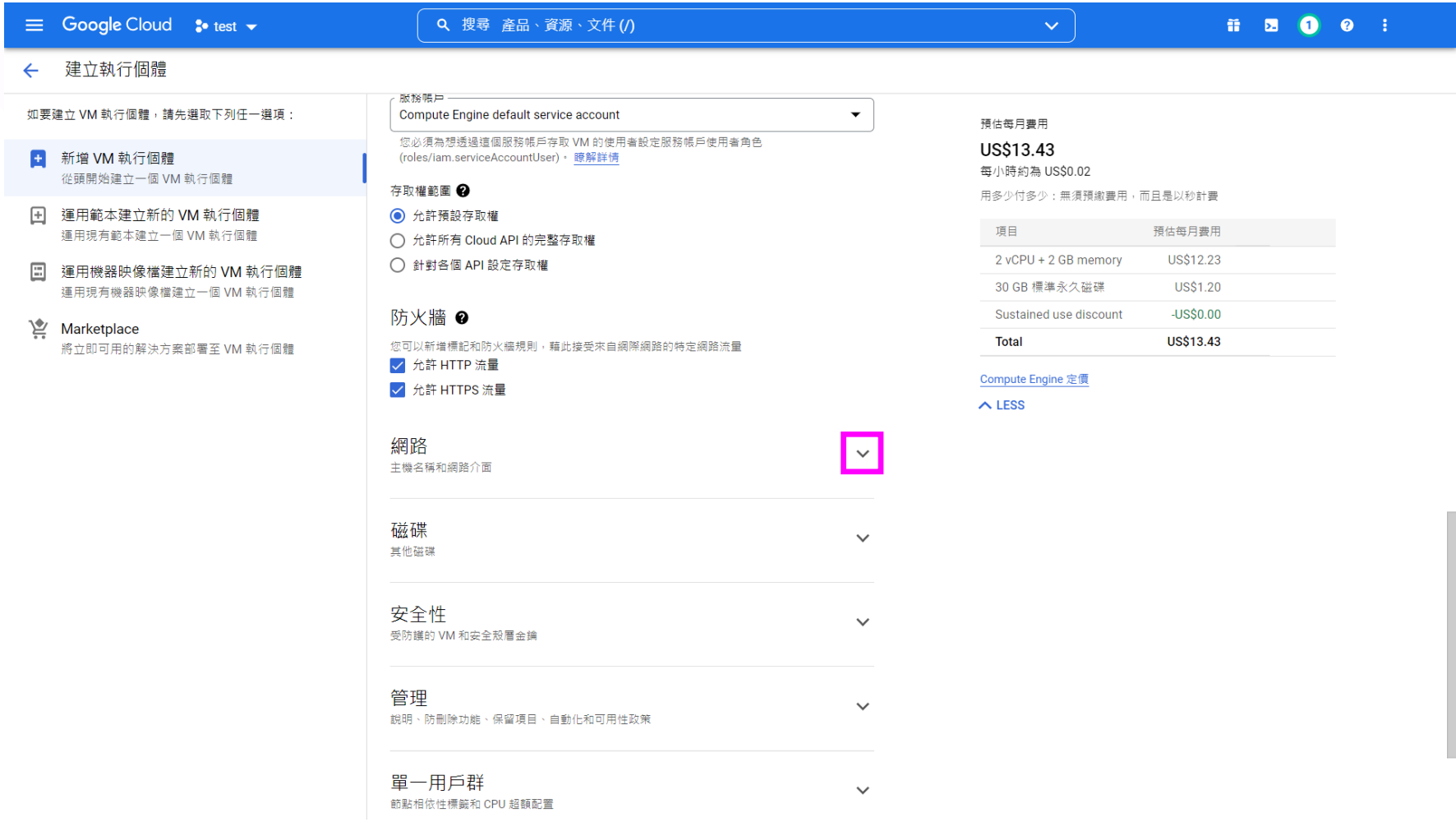
1. 硬碟空間視需求設定，一般應用可設為 30G
2. 預設的已平衡的永久硬碟非永久免費，為防未來若轉為正式帳號導致誤出帳可改為標準永久磁碟

架設 VM - GCP

4. 建立 VM

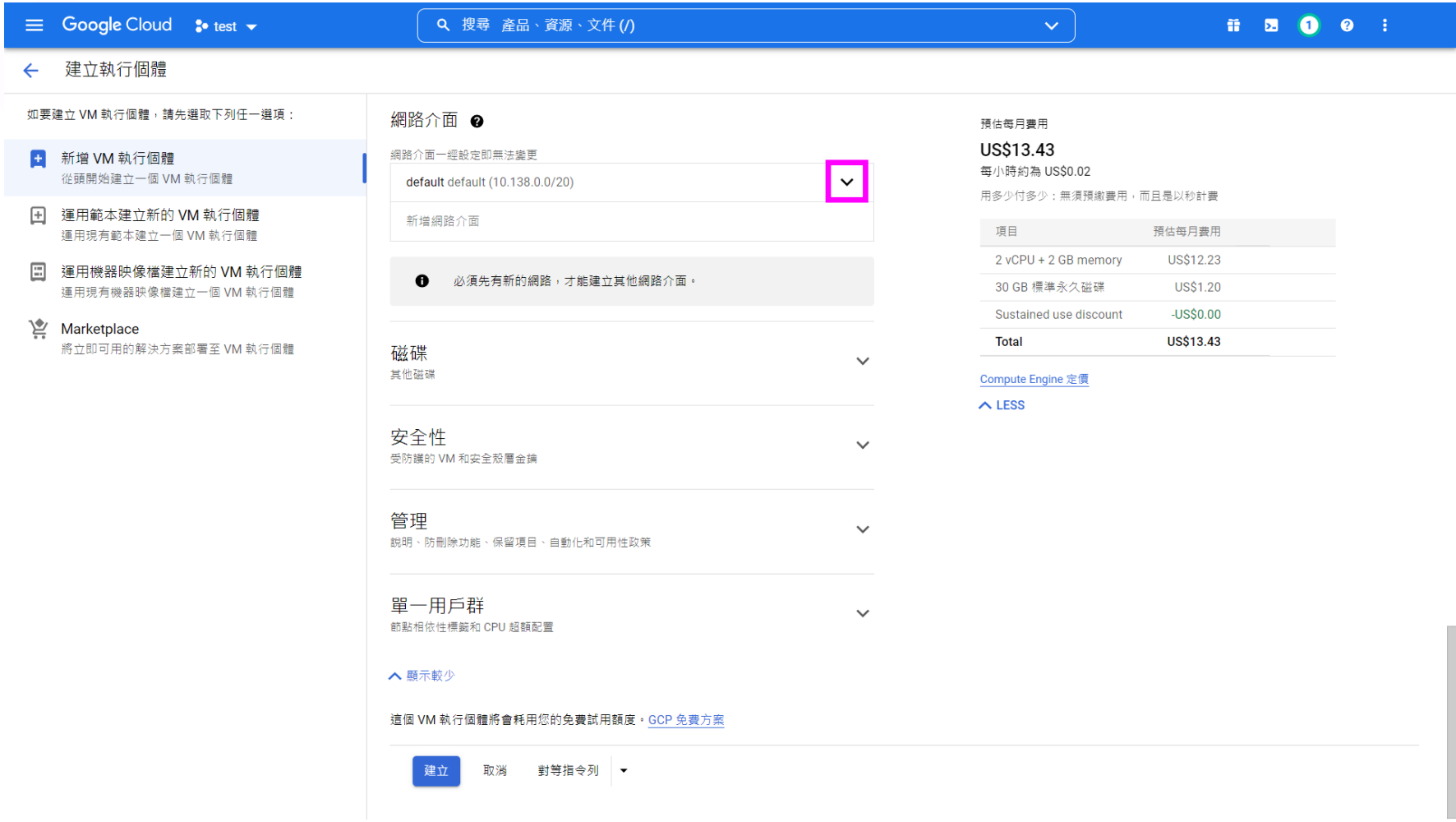
架設 VM - GCP

4. 建立 VM



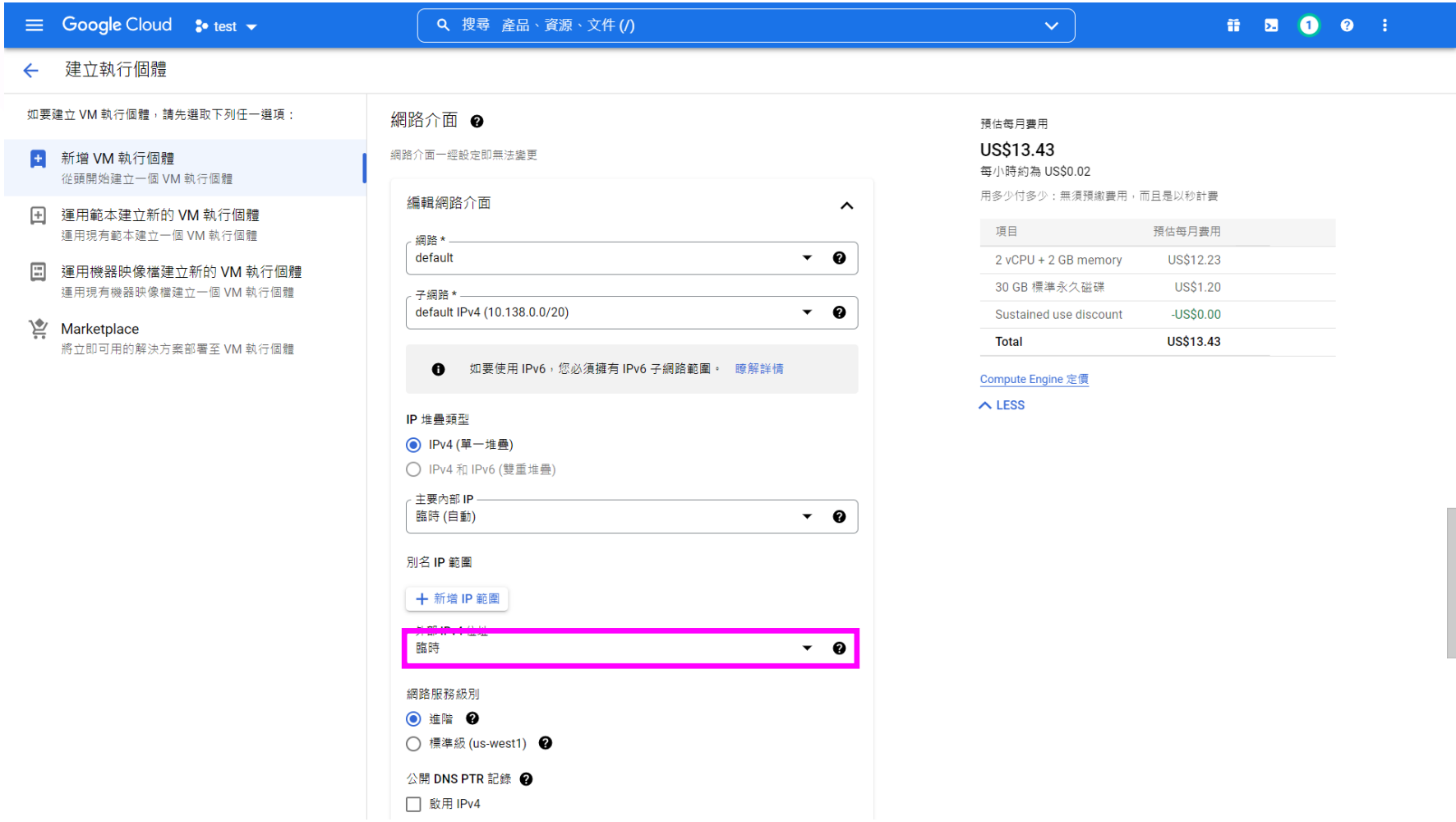
架設 VM - GCP

4. 建立 VM



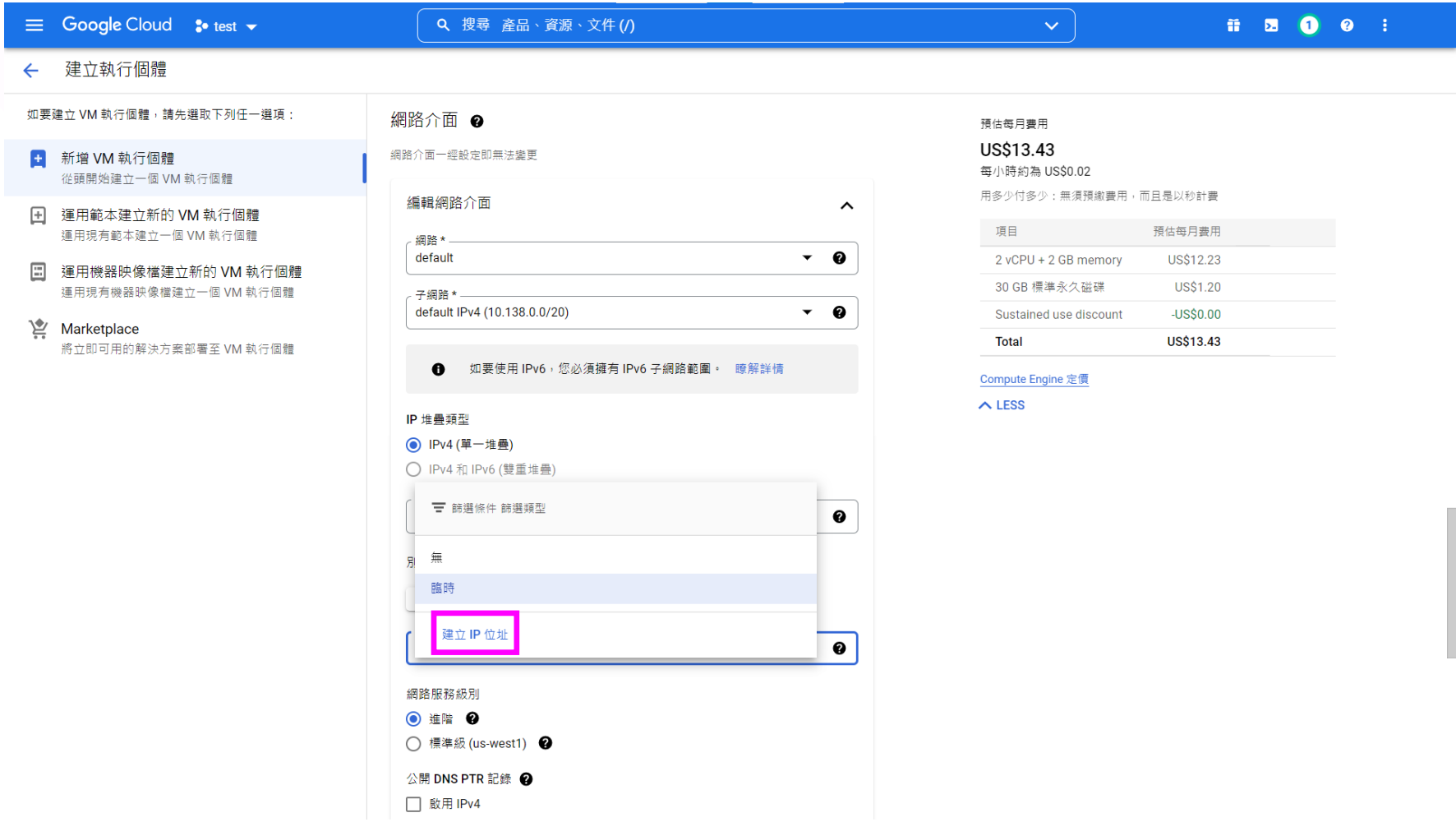
架設 VM - GCP

4. 建立 VM



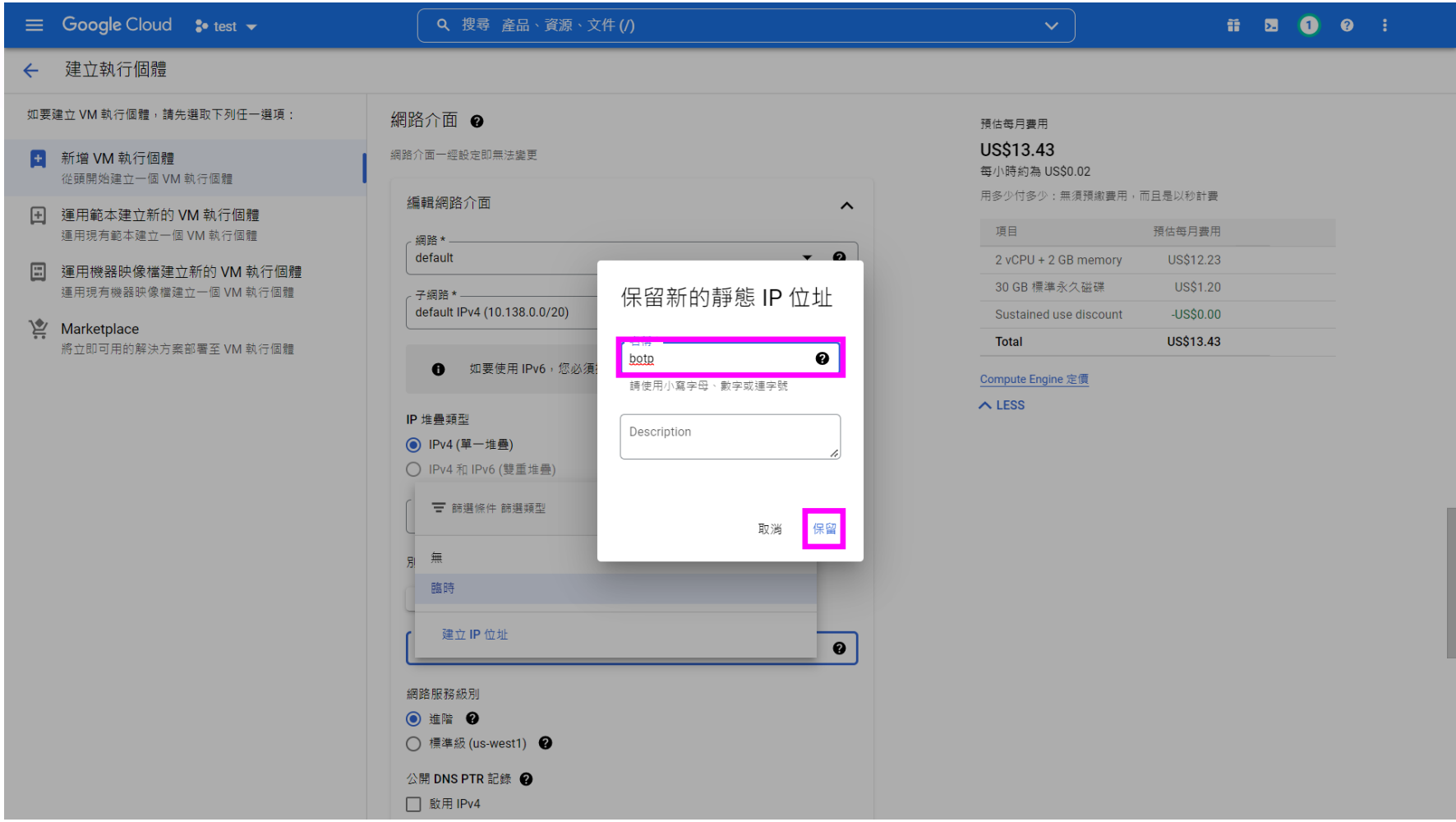
架設 VM - GCP

4. 建立 VM



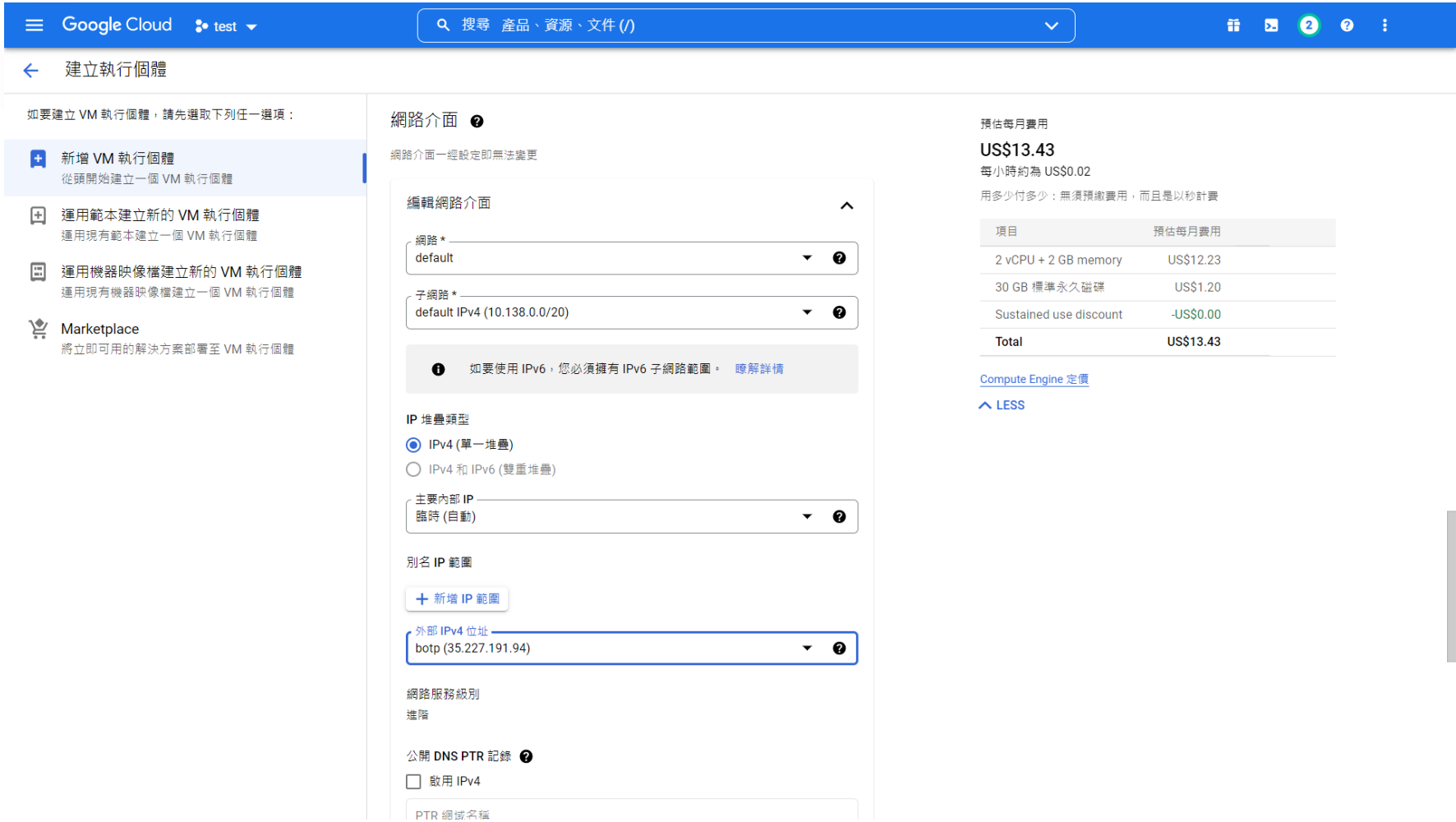
架設 VM - GCP

4. 建立 VM



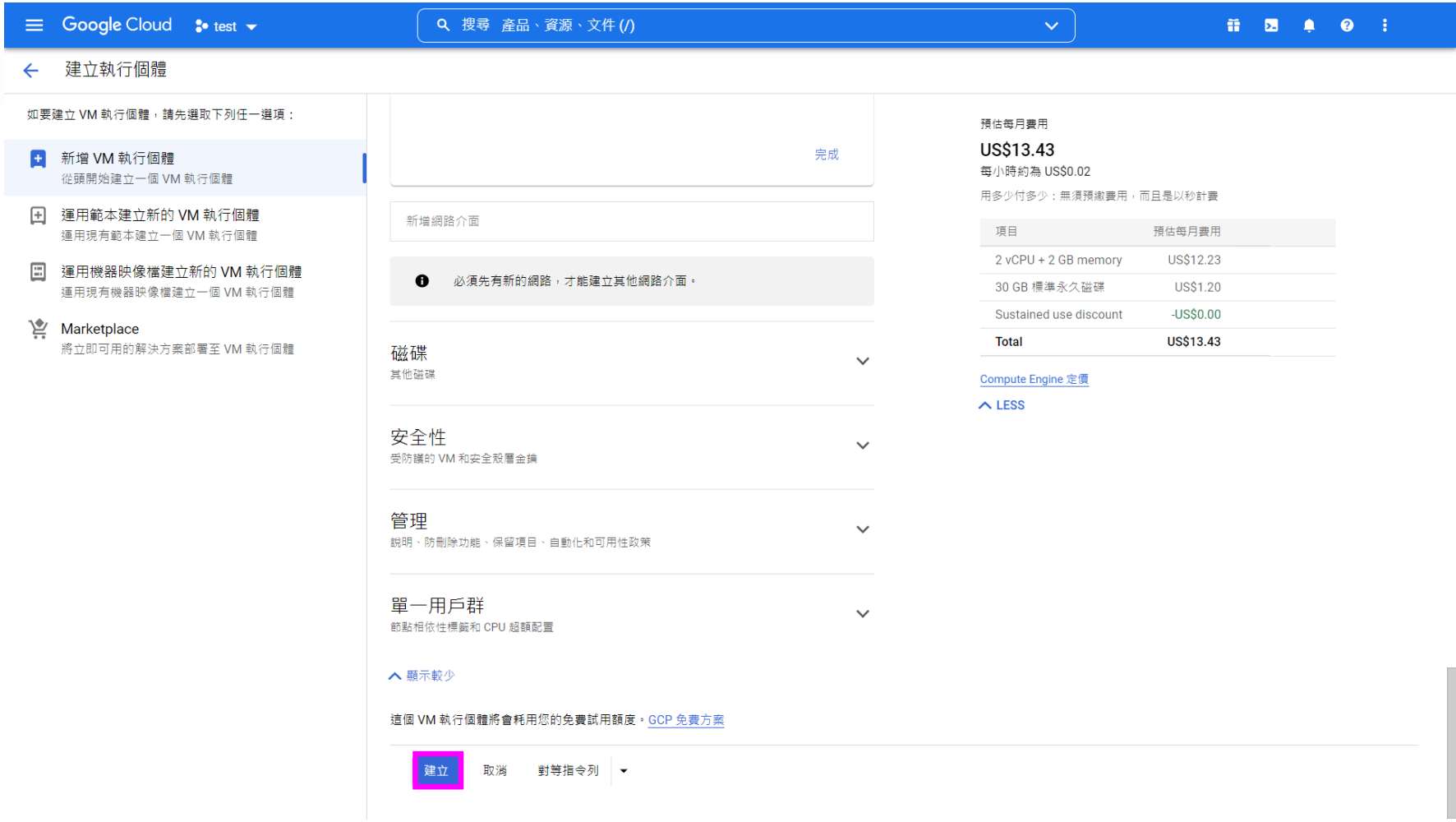
架設 VM - GCP

4. 建立 VM



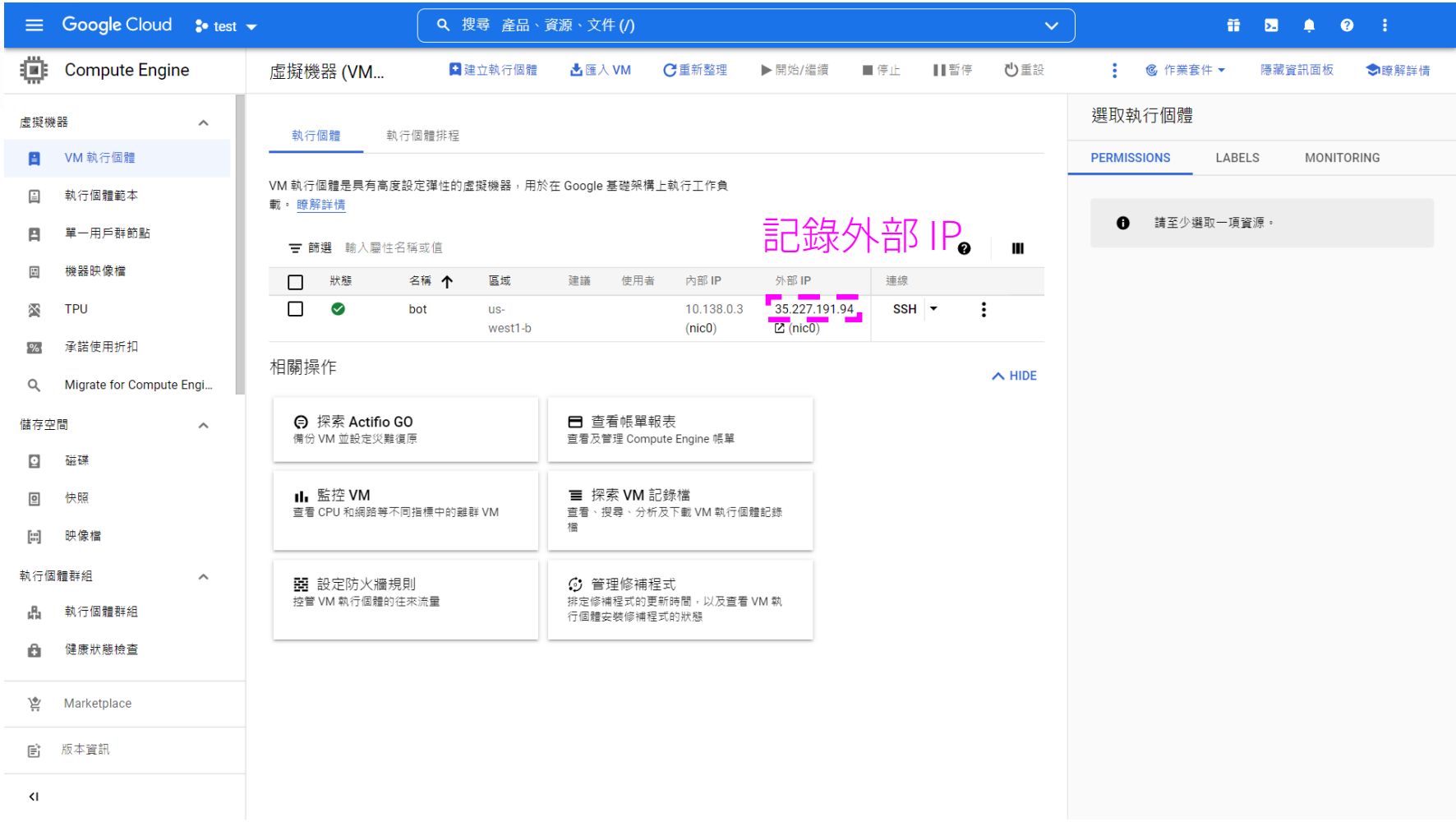
架設 VM - GCP

4. 建立 VM



架設 VM - GCP

4. 建立 VM



架設 VM - GCP

5. 準備開發環境

建立金鑰

https://cloud.google.com/ > 文件 > 運算 > Compute Engine > Guides: Managing SSH Keys in metadata > Windows

設定用戶端

https://cloud.google.com/ > 文件 > 運算 > Compute Engine > Guides: Connecting to instances > Connecting to Linux VMs using advanced methods > Windows(PuTTY)

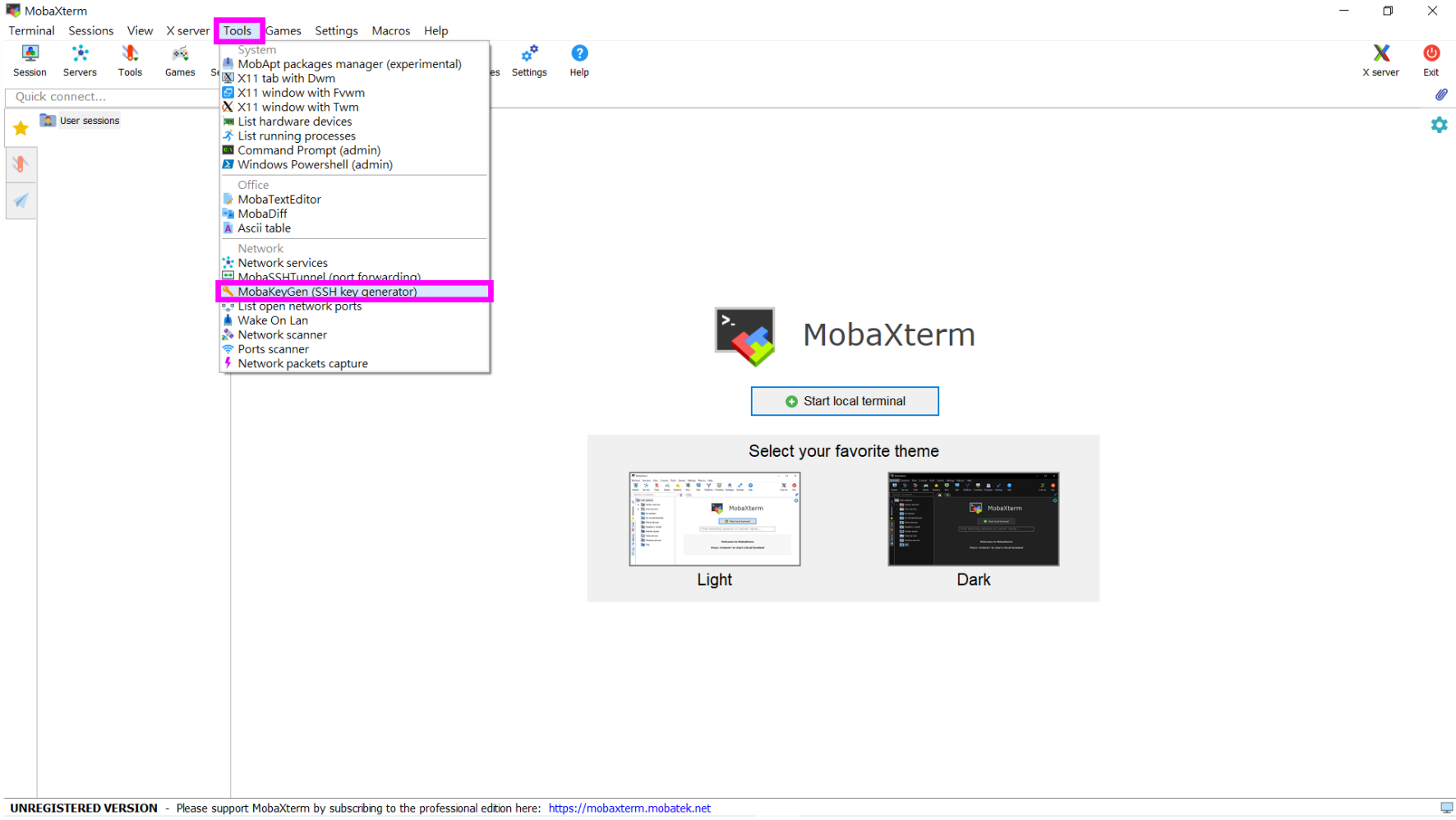
架設 VM - GCP

5. 準備開發環境

啟動 MobaXterm

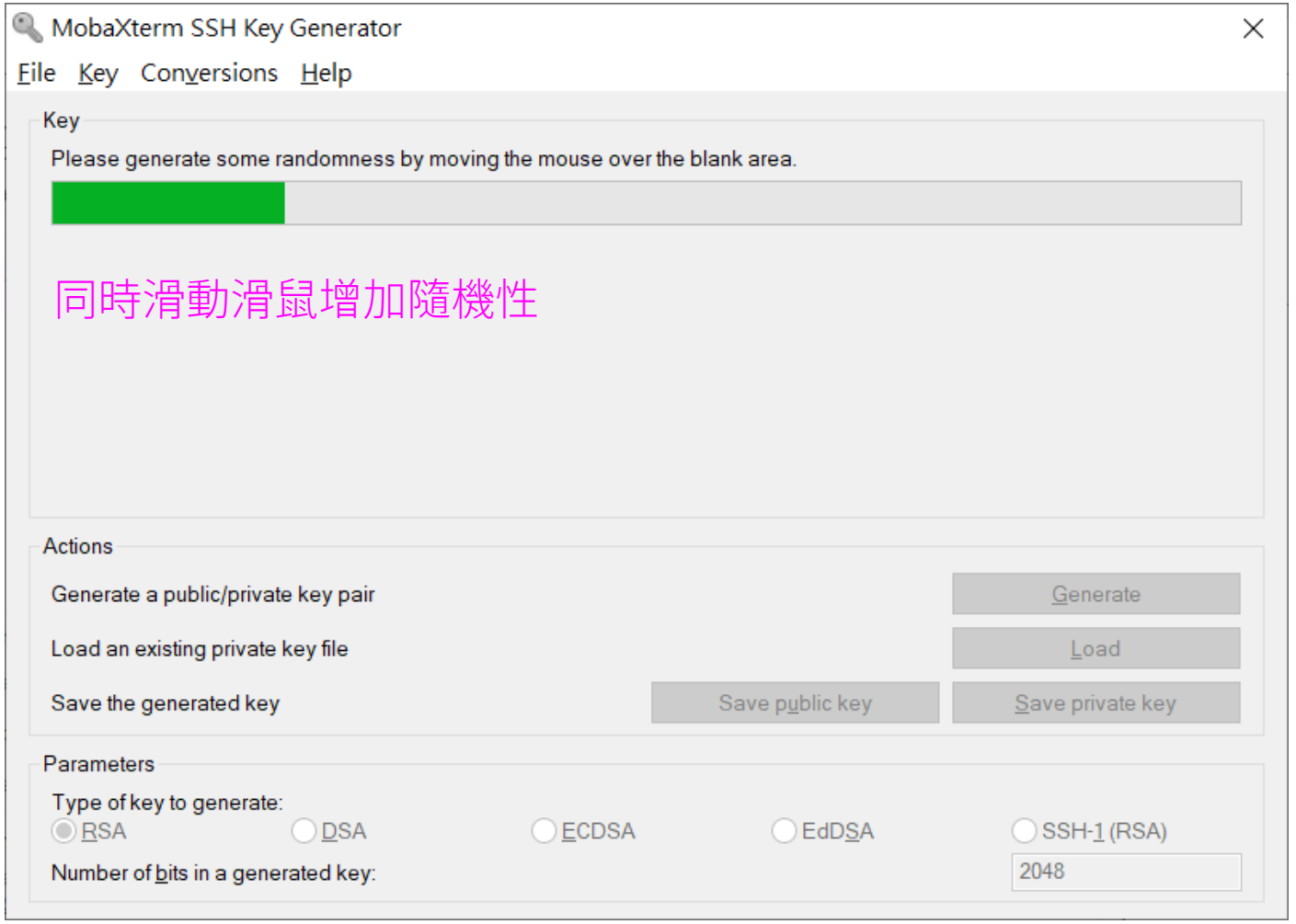
架設 VM - GCP

5. 準備開發環境



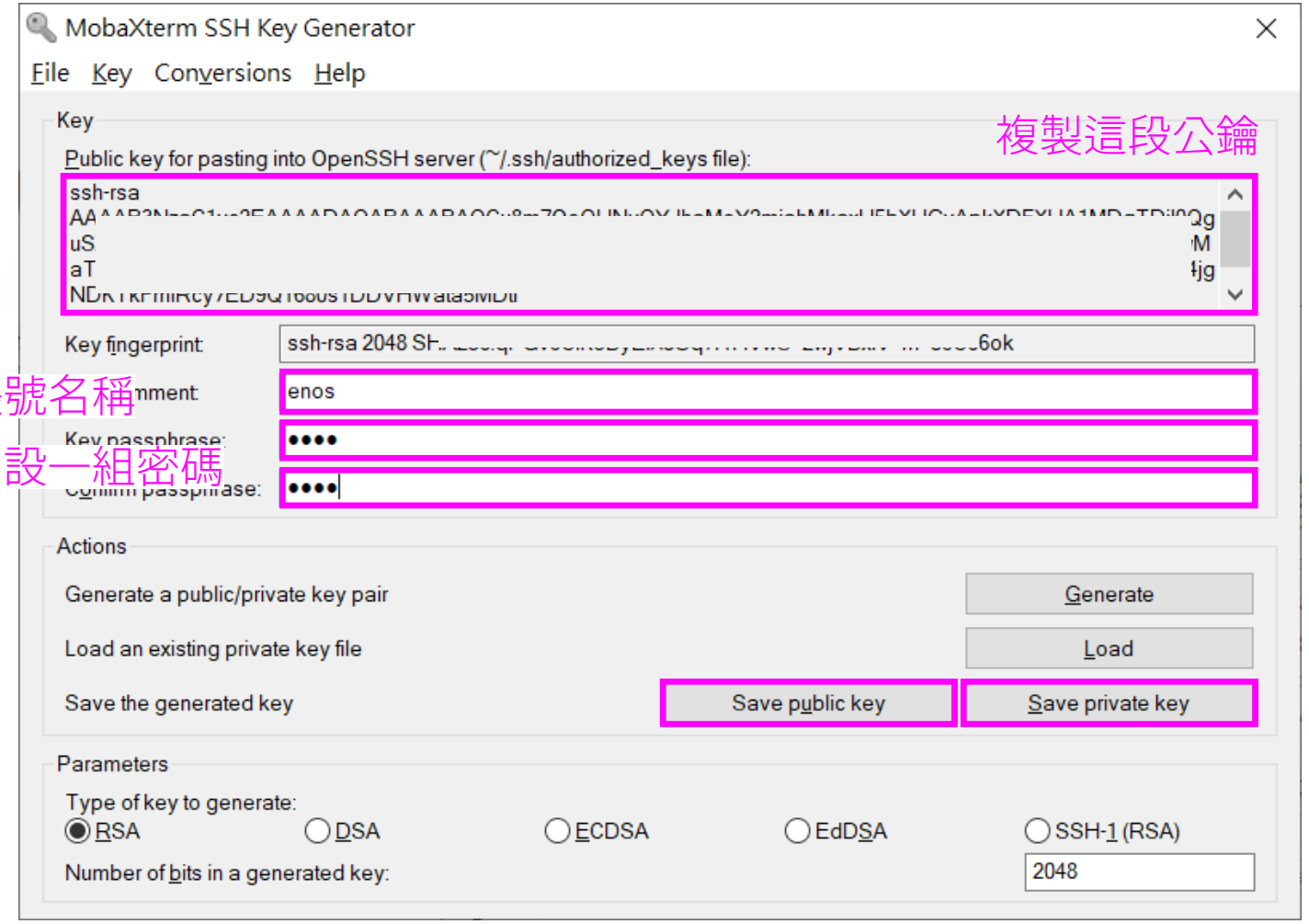
架設 VM - GCP

5. 準備開發環境



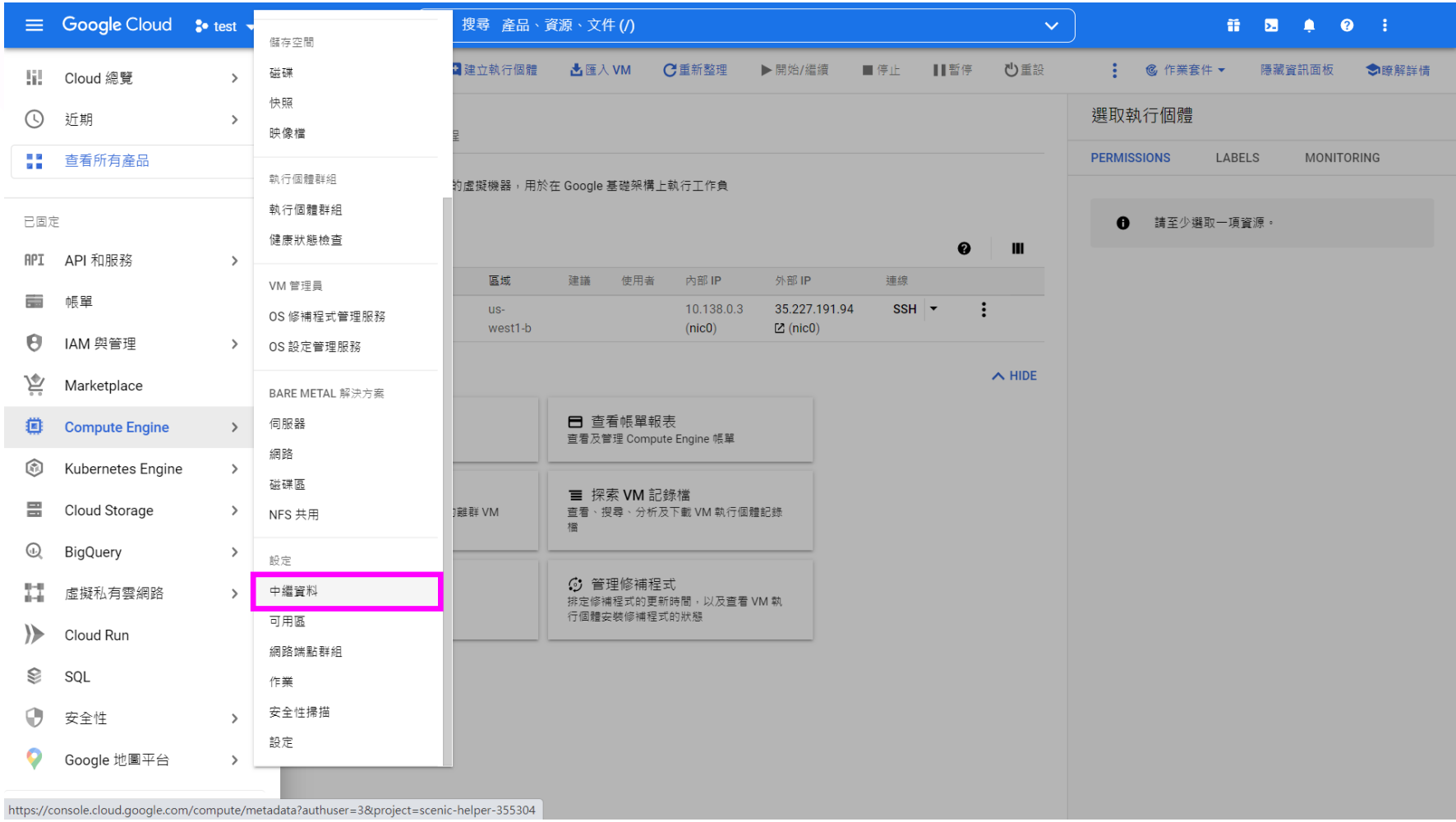
架設 VM - GCP

5. 準備開發環境



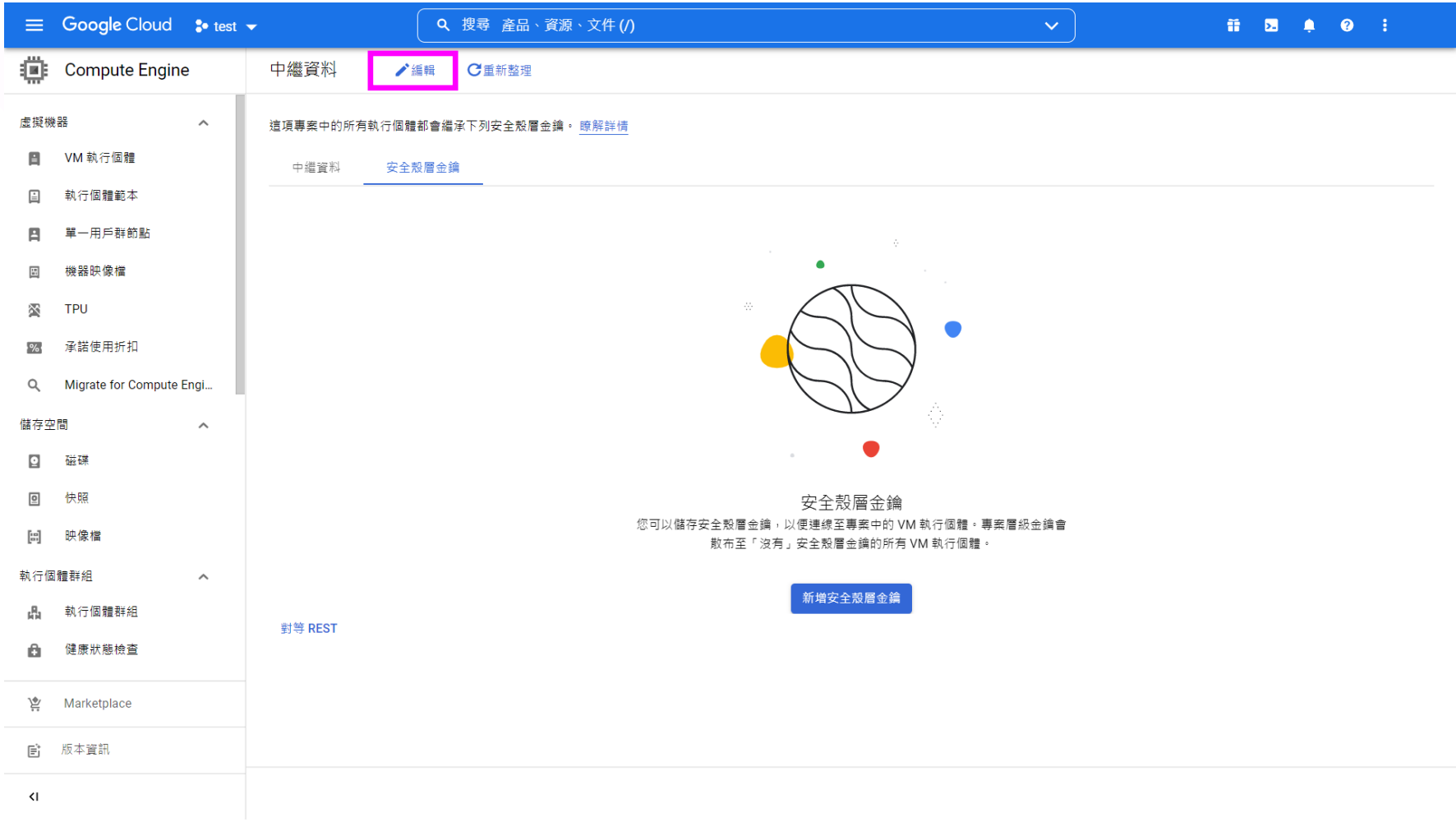
架設 VM - GCP

5. 準備開發環境



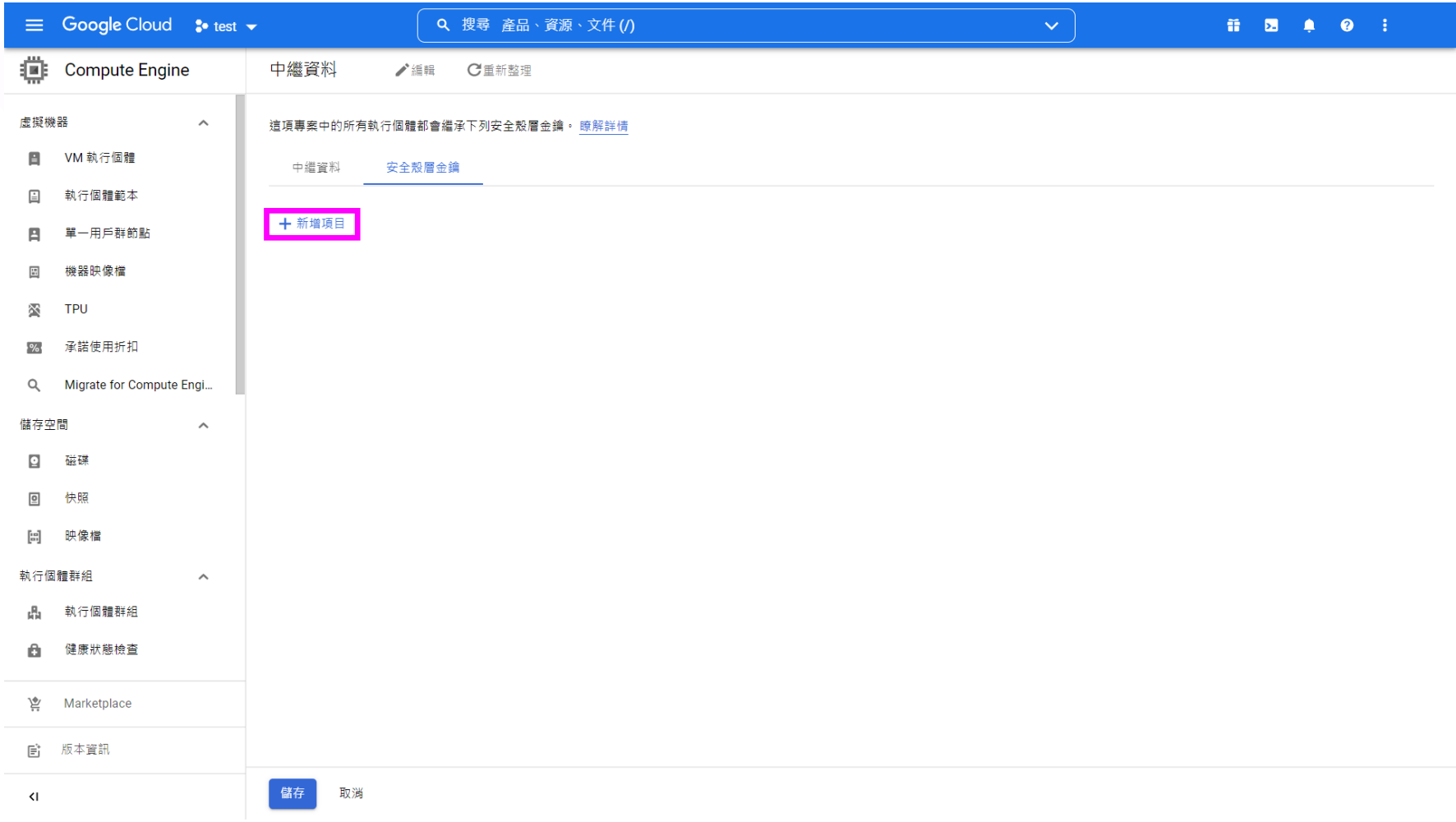
架設 VM - GCP

5. 準備開發環境



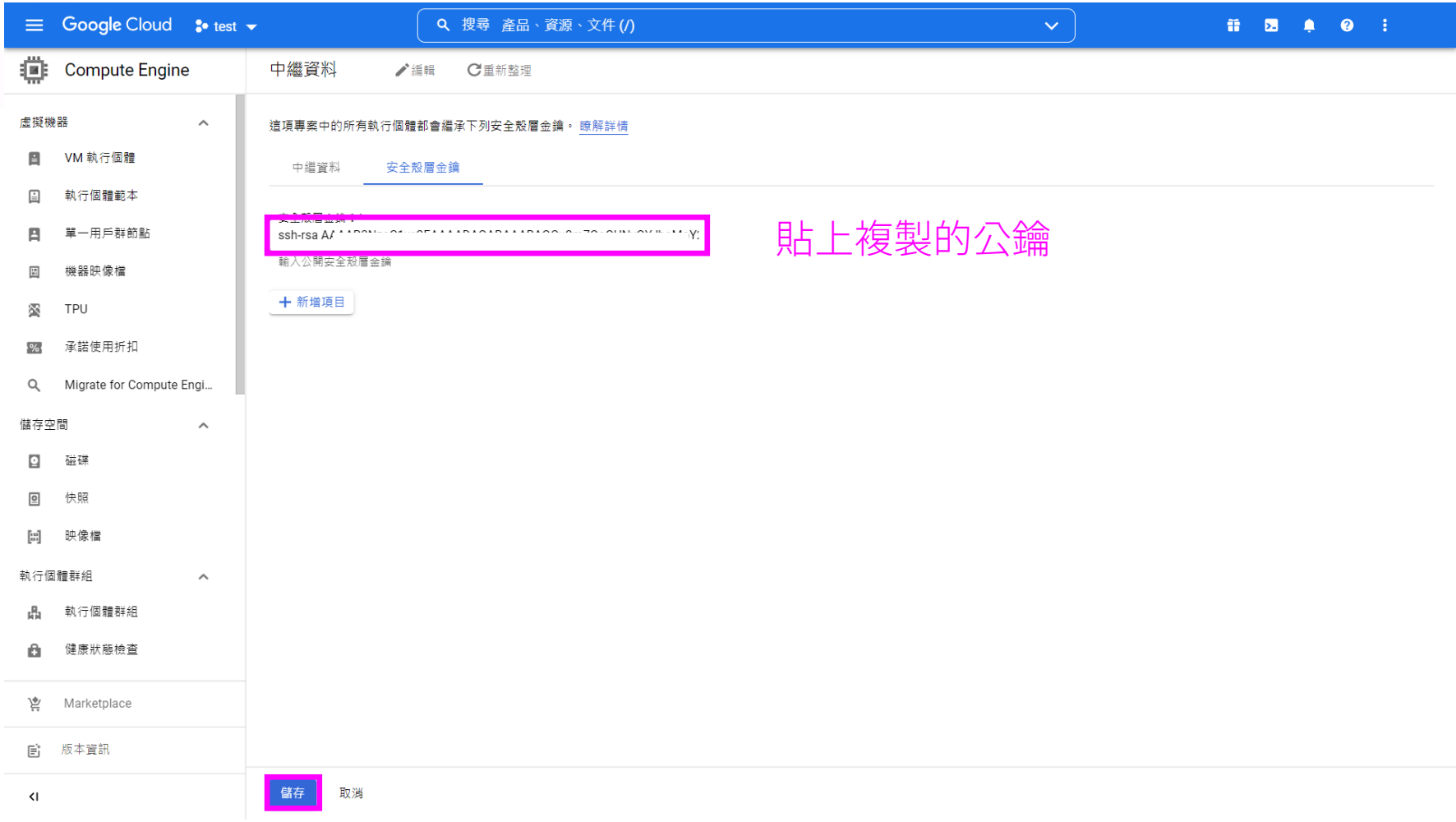
架設 VM - GCP

5. 準備開發環境



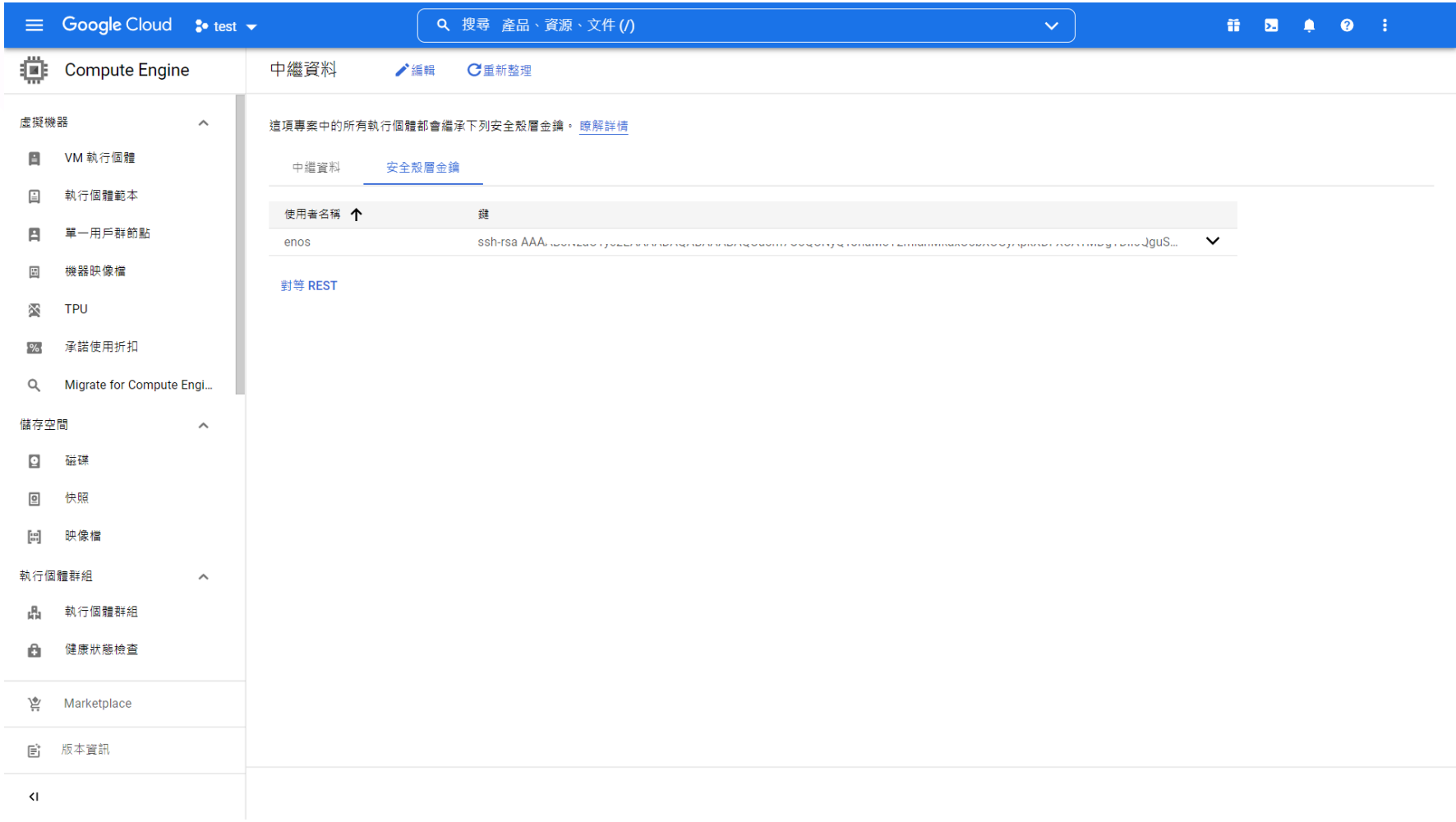
架設 VM - GCP

5. 準備開發環境



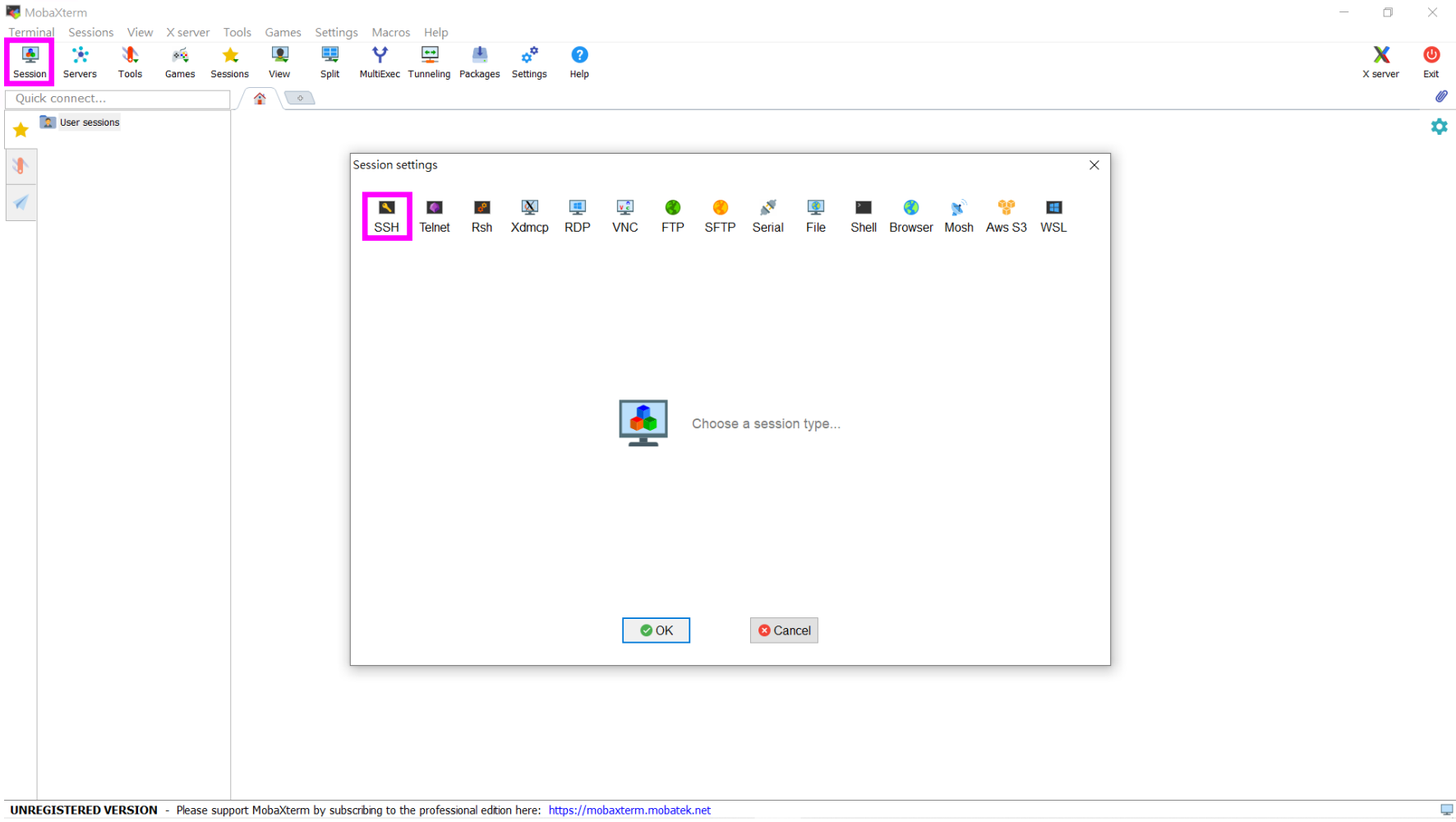
架設 VM - GCP

5. 準備開發環境



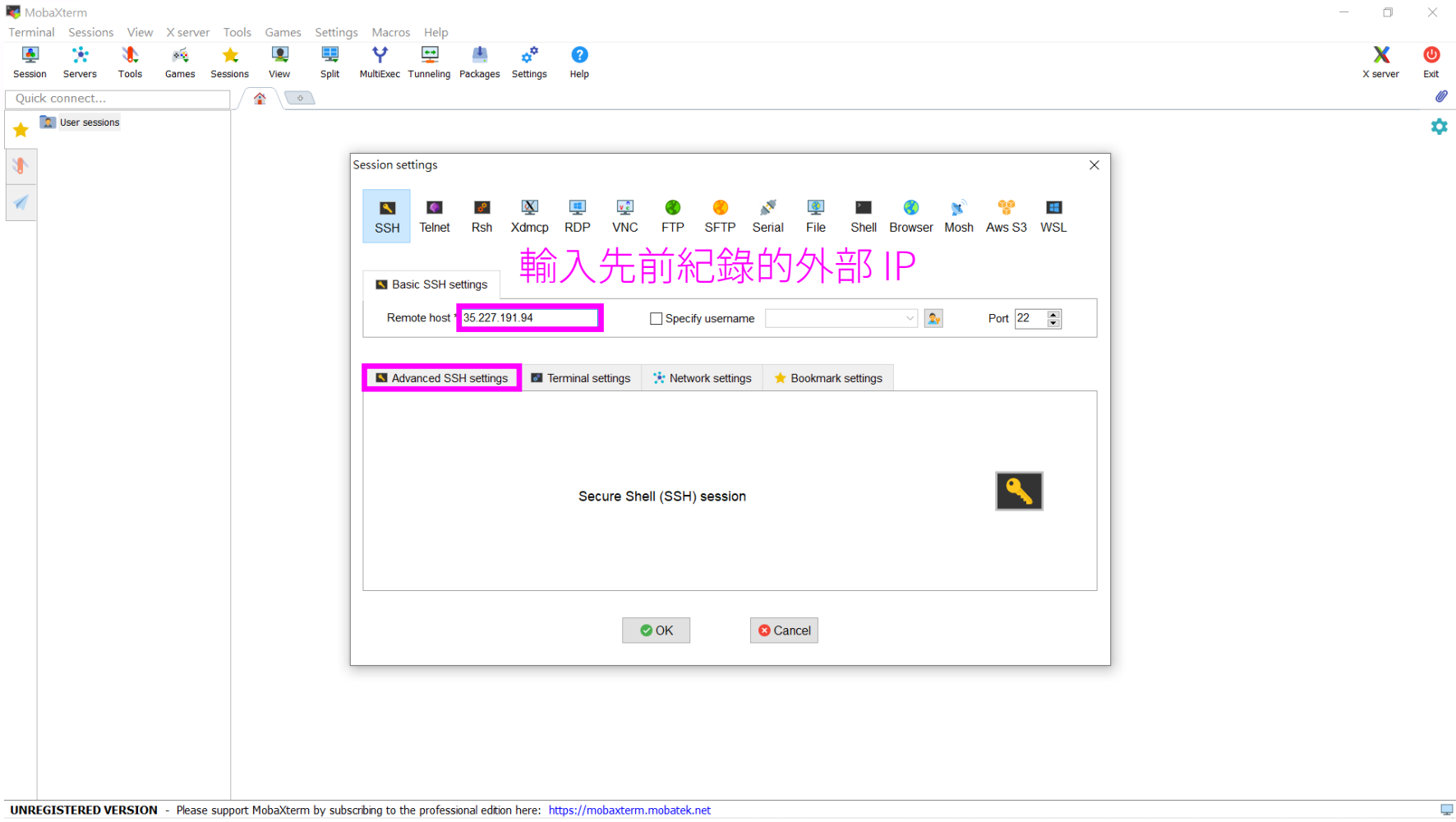
架設 VM - GCP

5. 準備開發環境



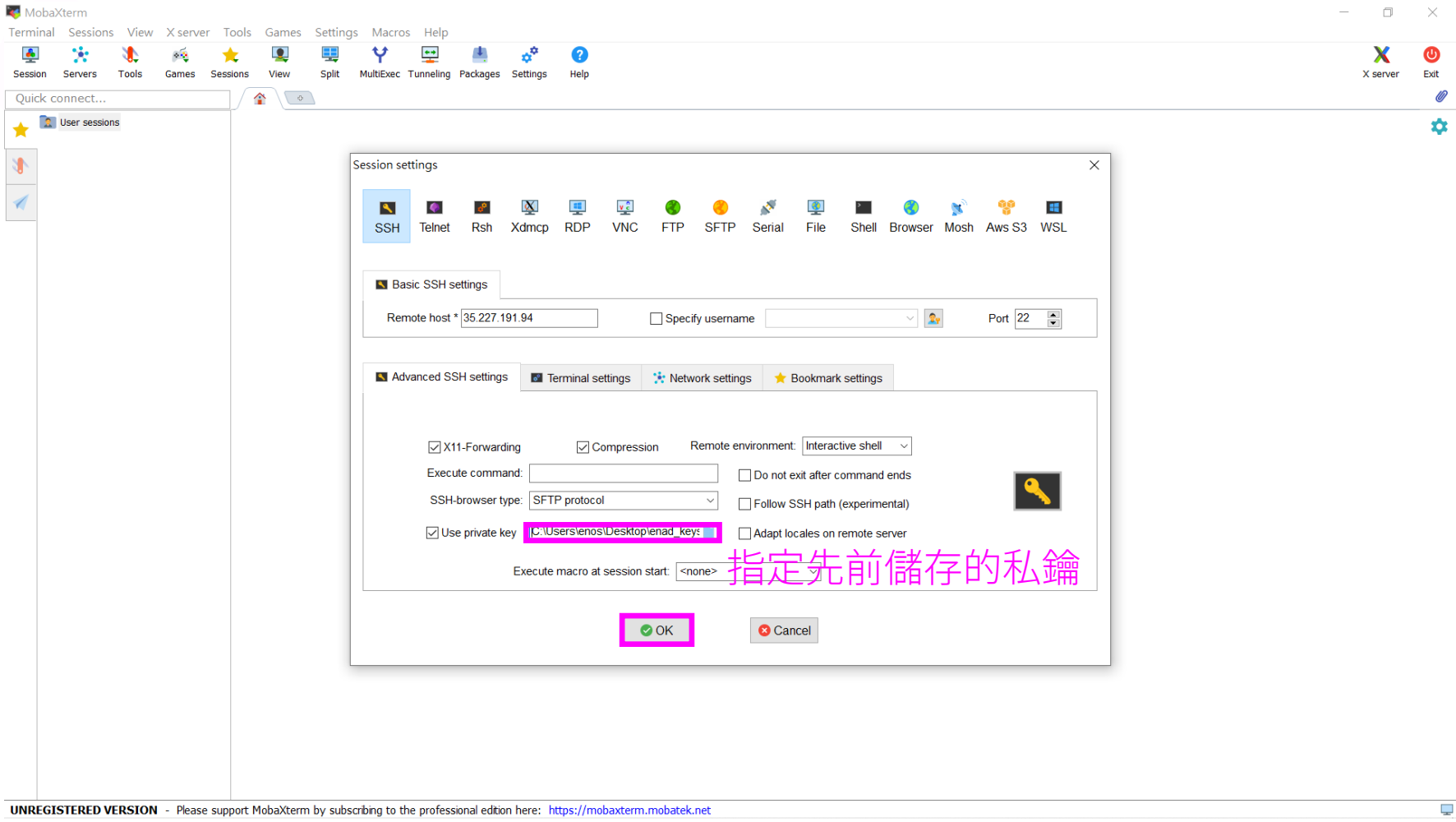
架設 VM - GCP

5. 準備開發環境



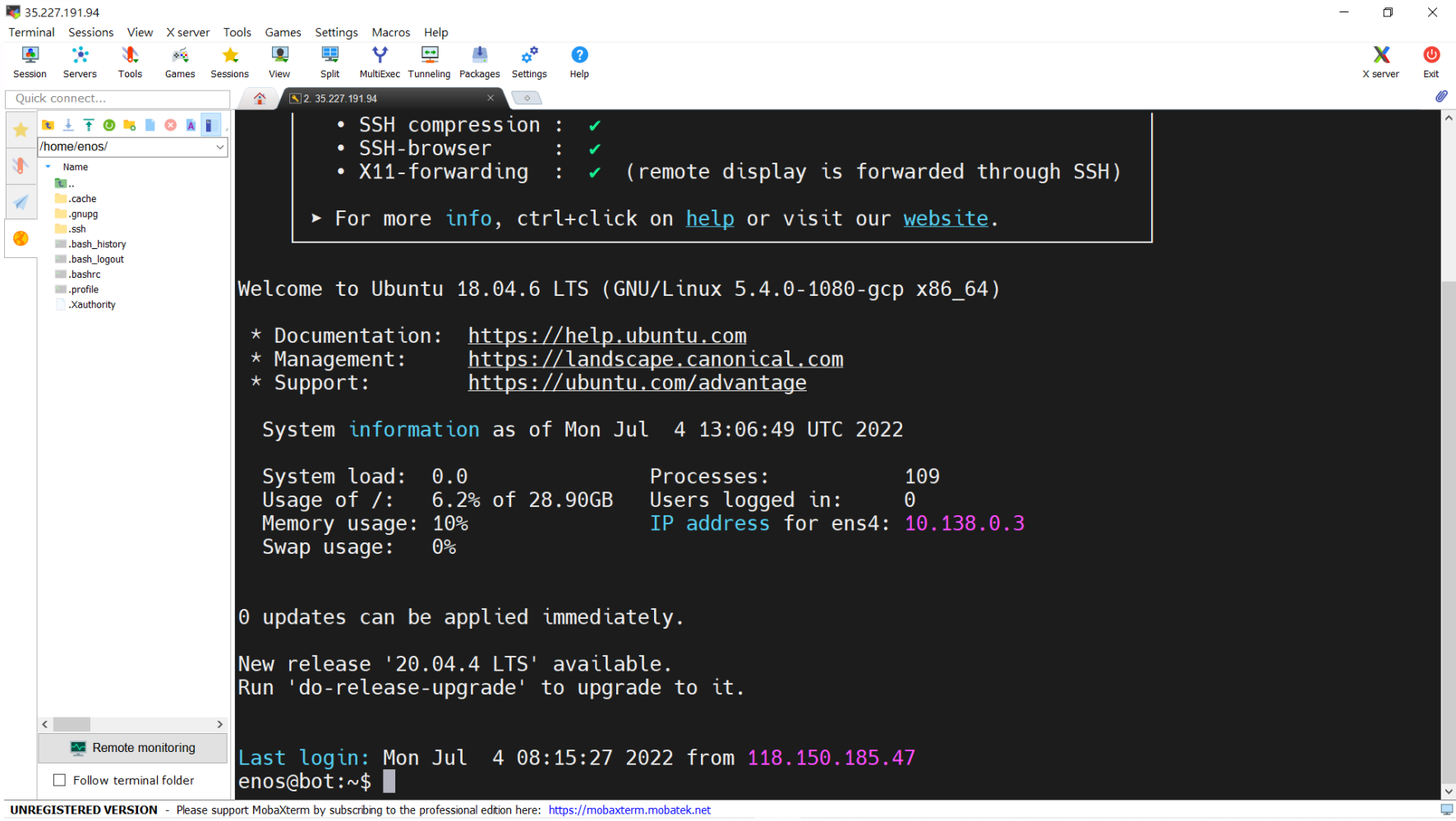
架設 VM - GCP

5. 準備開發環境



架設 VM - GCP

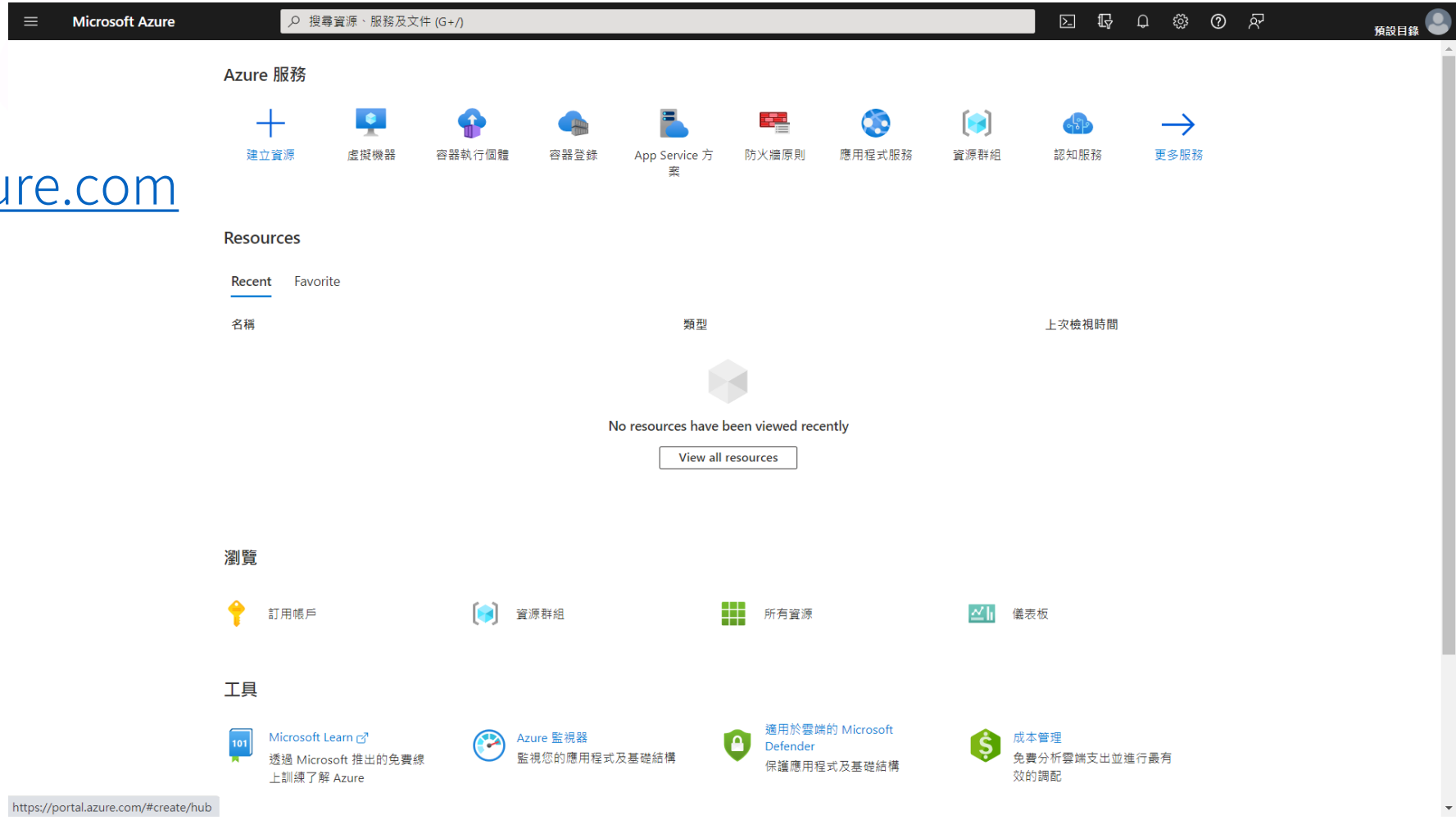
5. 準備開發環境



架設 VM - Azure

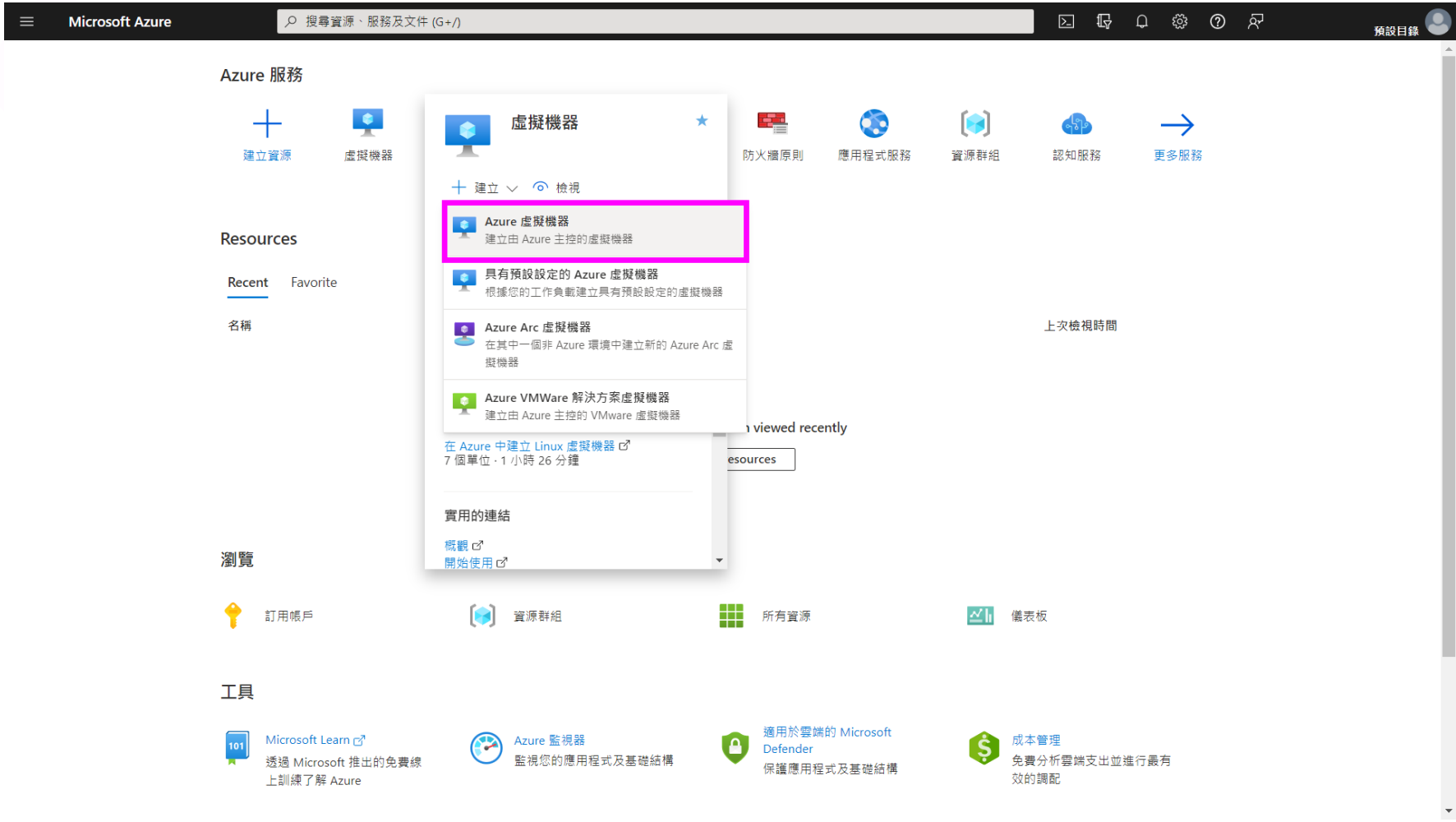
1. 登入

https://portal.azure.com



架設 VM - Azure

2. 建立 VM



架設 VM - Azure

2. 建立 VM

架設 VM - Azure

2. 建立 VM

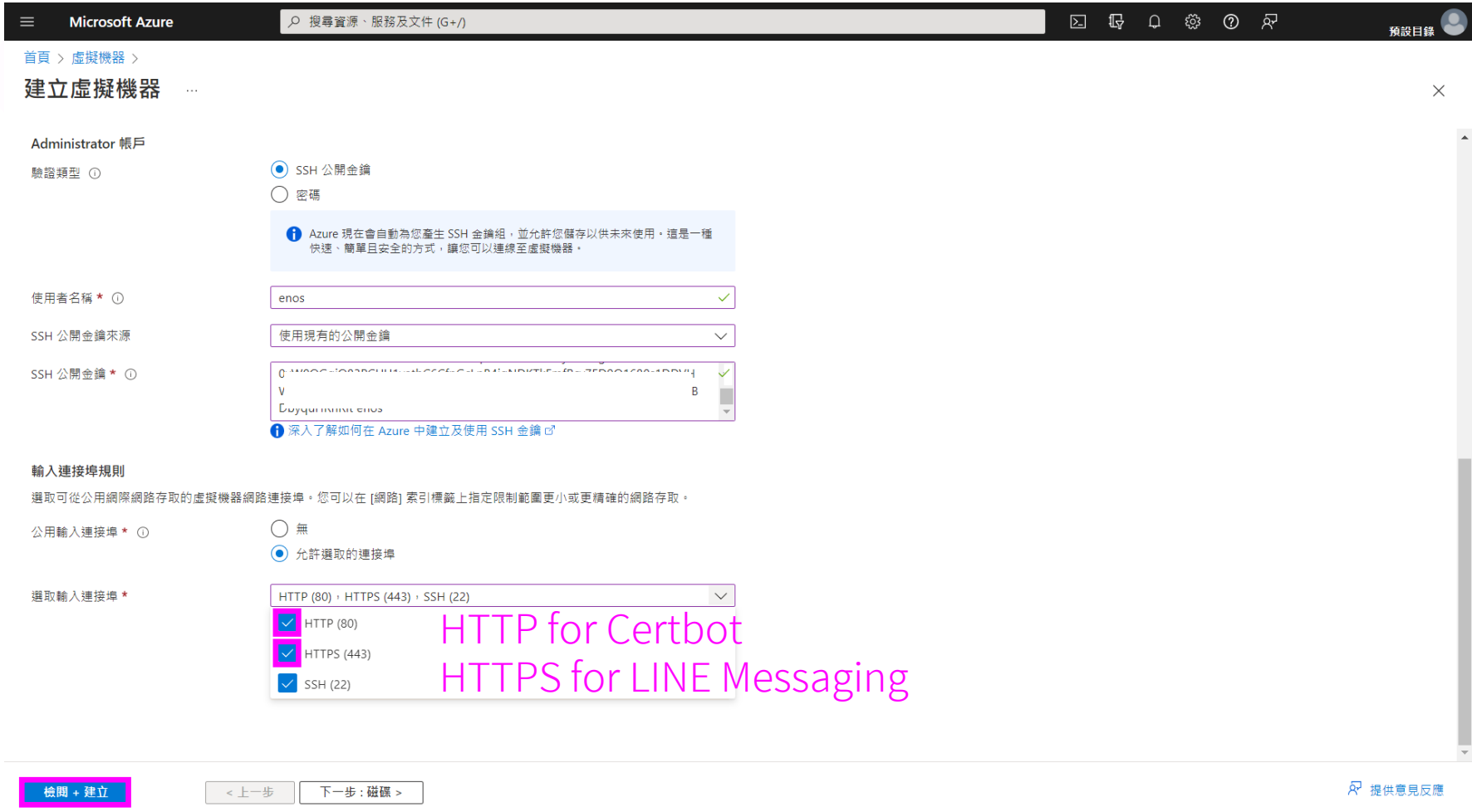
架設 VM - Azure

2. 建立 VM

自訂帳號
金鑰建立方式，選擇現有
貼上公鑰

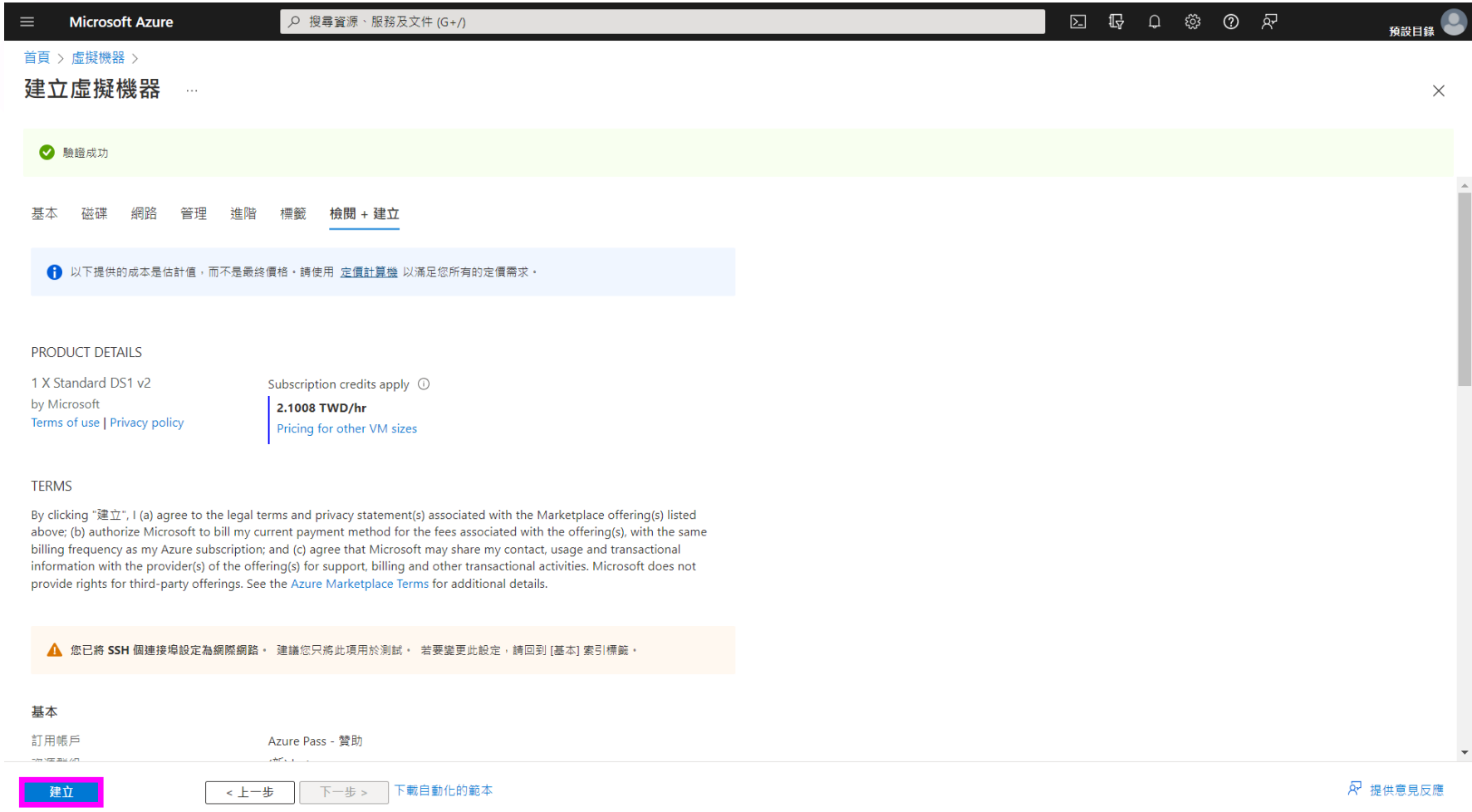
架設 VM - Azure

2. 建立 VM



架設 VM - Azure

2. 建立 VM

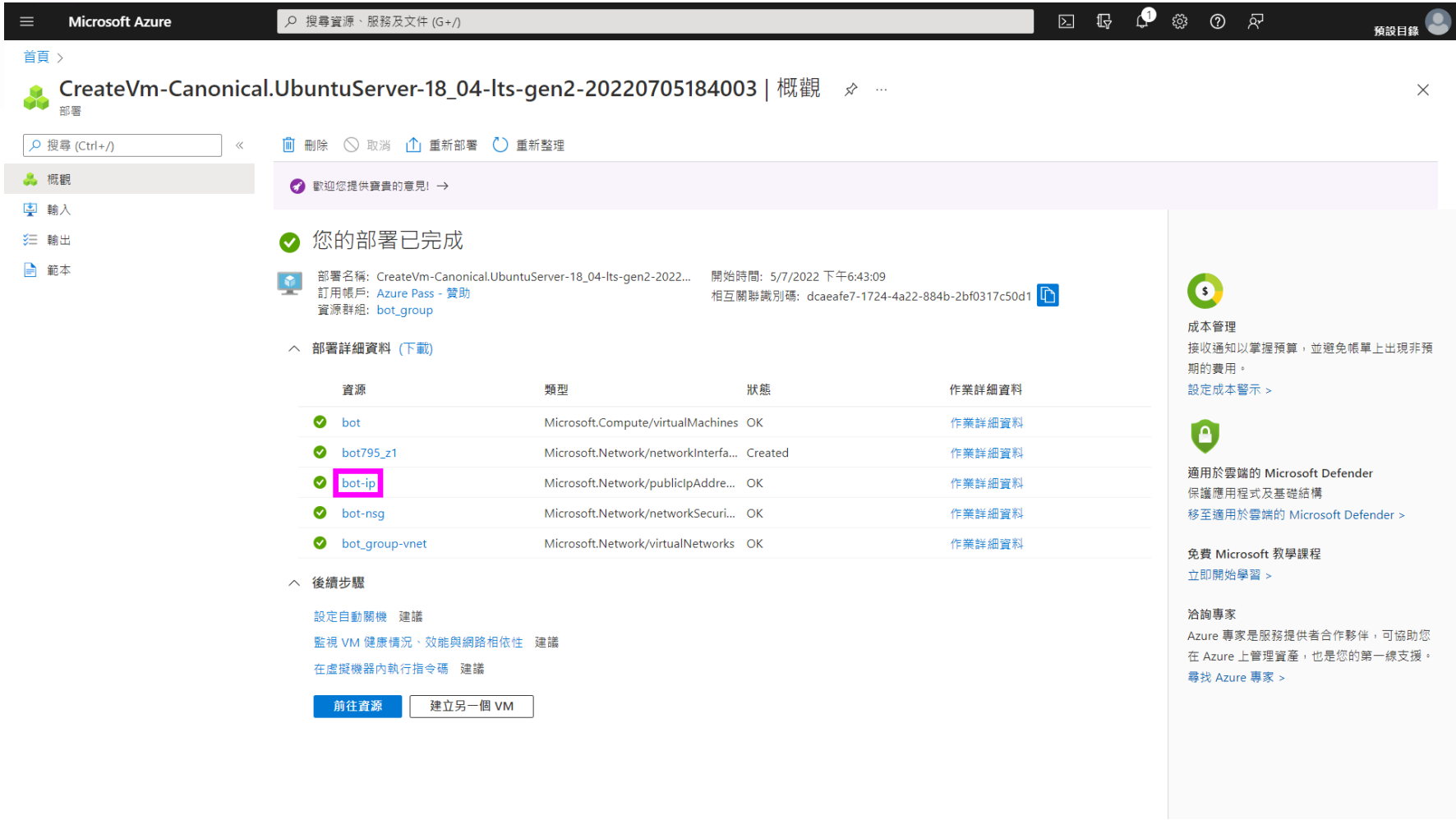


架設 VM - Azure

2. 建立 VM

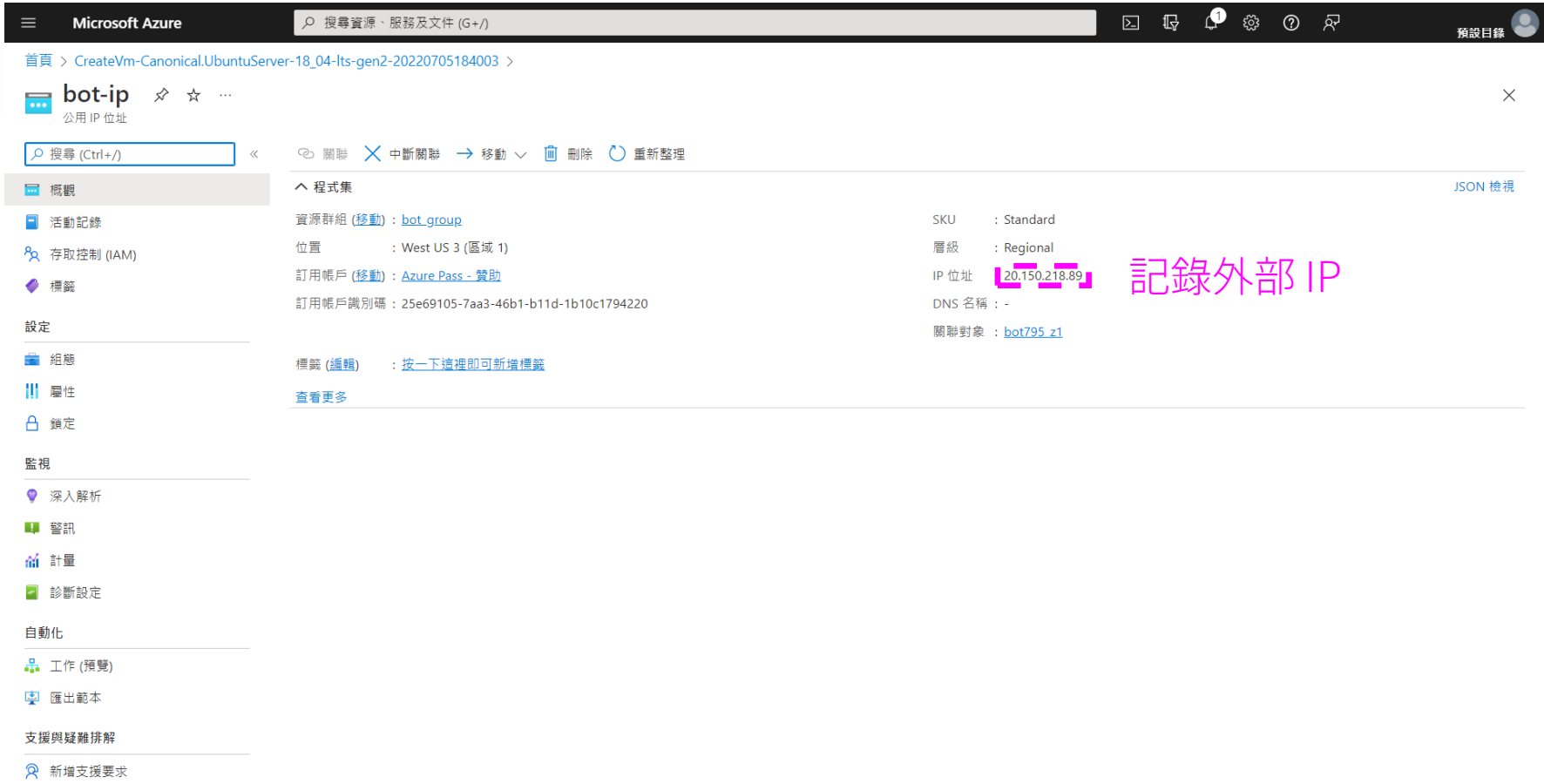
架設 VM - Azure

2. 建立 VM



架設 VM - Azure

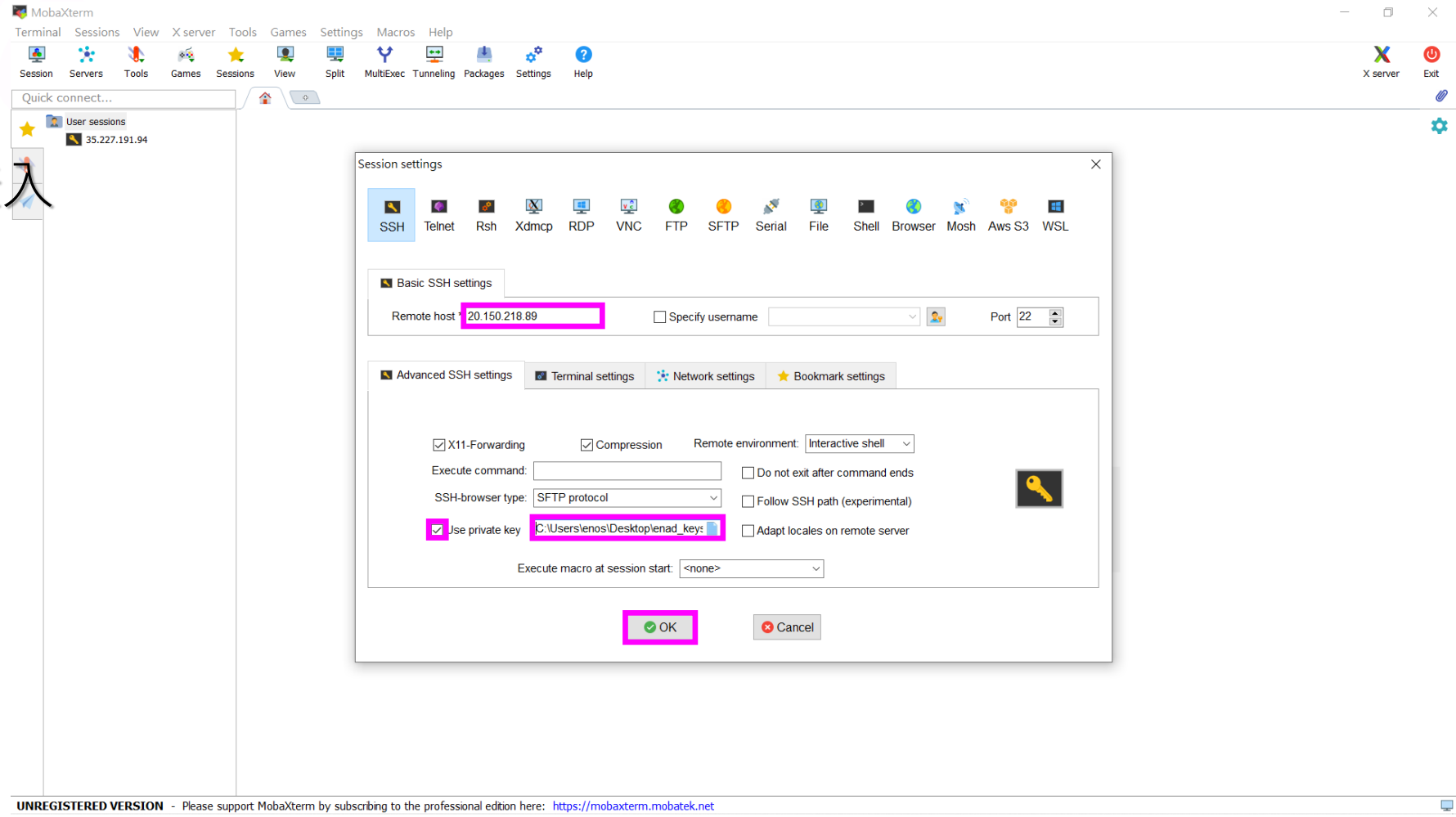
2. 建立 VM



架設 VM - Azure

3. 準備開發環境

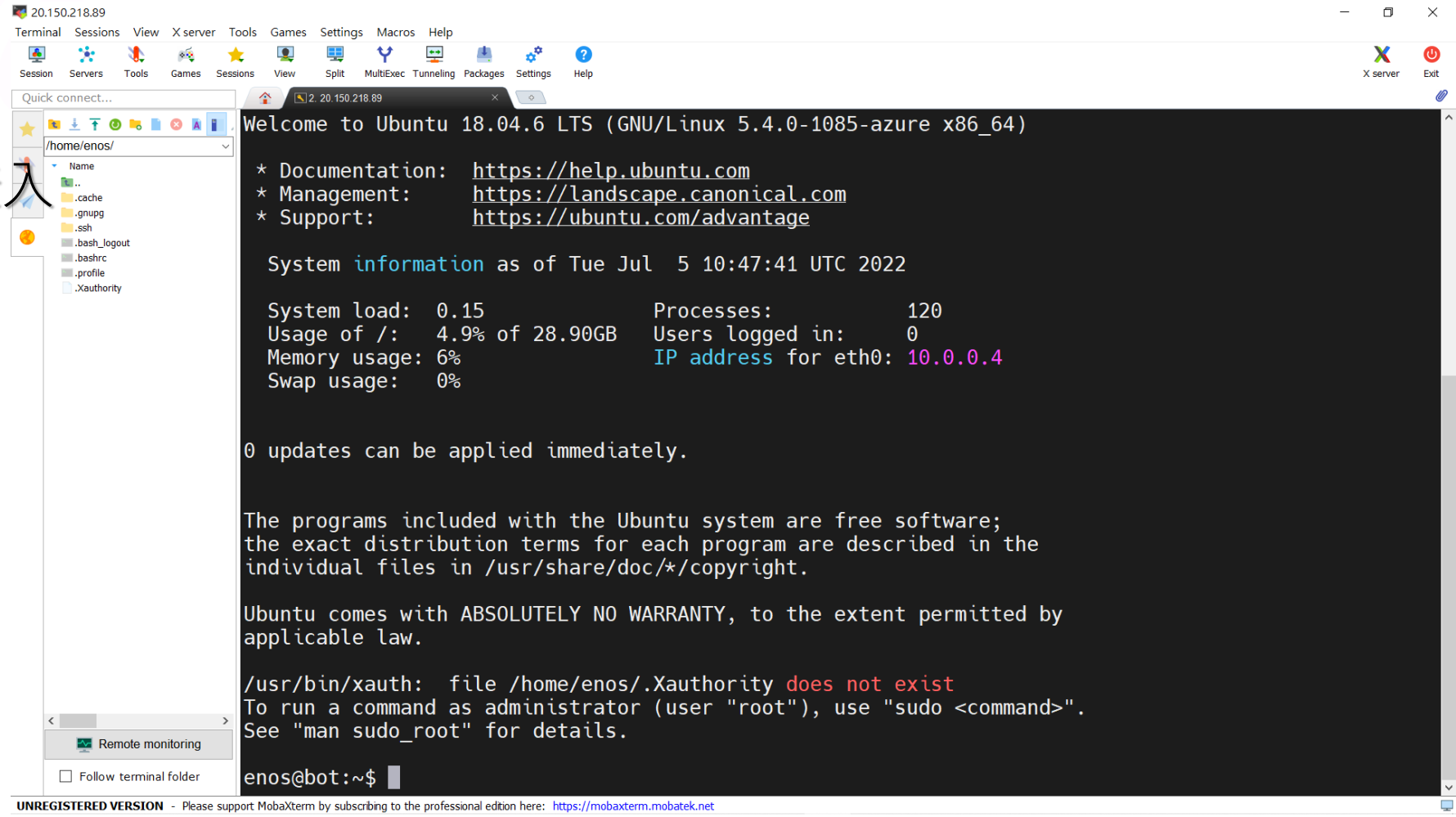
以 MobaXterm 登入



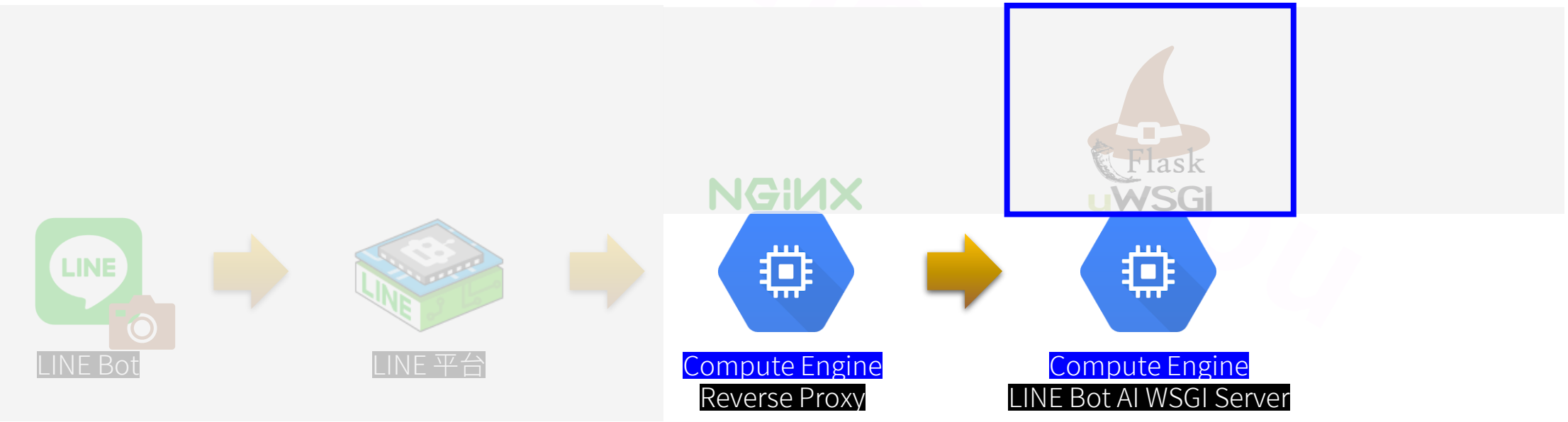
架設 VM - Azure

3. 準備開發環境

以 MobaXterm 登入



架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging



LINE Bot & WSGI

1. 準備程式碼

a. 下載範例程式並調整

① `trees17bot.py  # secret, token, model, labels`

② `trees17V1.h5`

③ `treeset_labels.txt`

④ `other tree samples`

LINE Bot & WSGI

1. 準備程式碼

b. 製作 requirements.txt

```
line-bot-sdk
flask
pillow
tensorflow==2.4.4
uwsgi
```

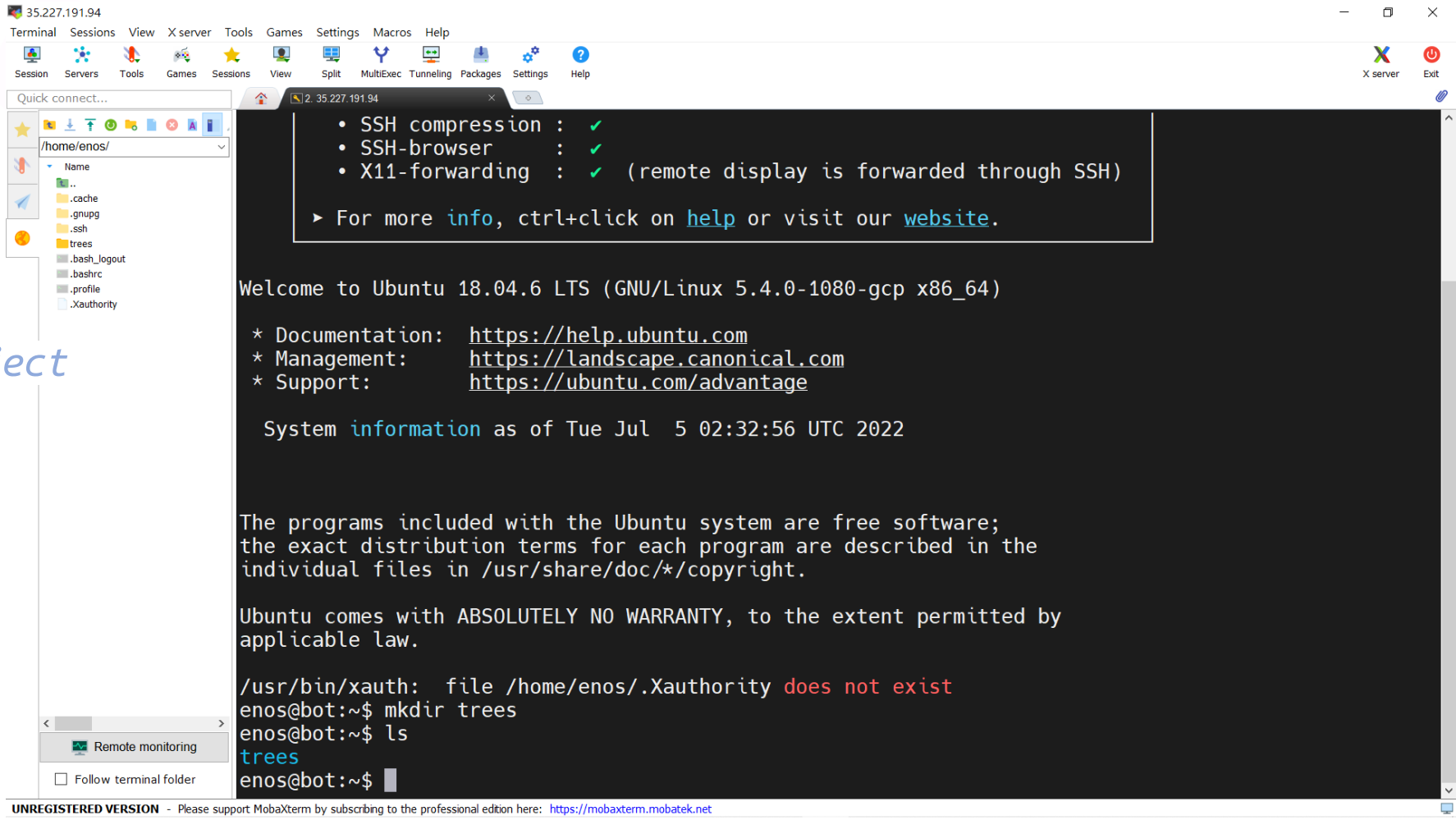
LINE Bot & WSGI

1. 準備程式碼

c. 於 VM 建立專案目錄

```
mkdir your_project
mkdir trees

ls
```



LINE Bot & WSGI

1. 準備程式碼

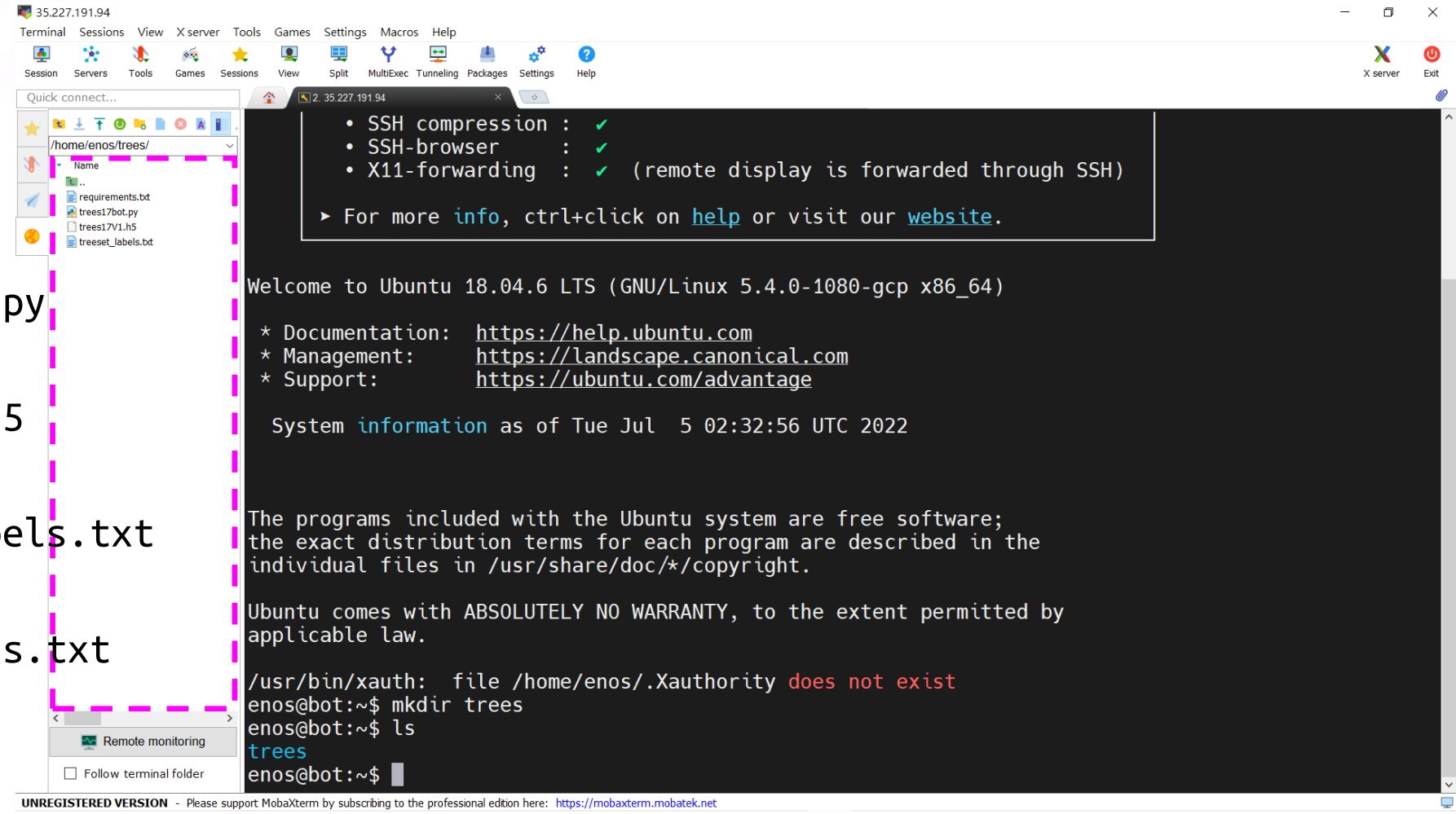
d. 上傳上述檔案至專案目錄

① trees17bot.py

② trees17V1.h5

③ treeset_labels.txt

④ requirements.txt



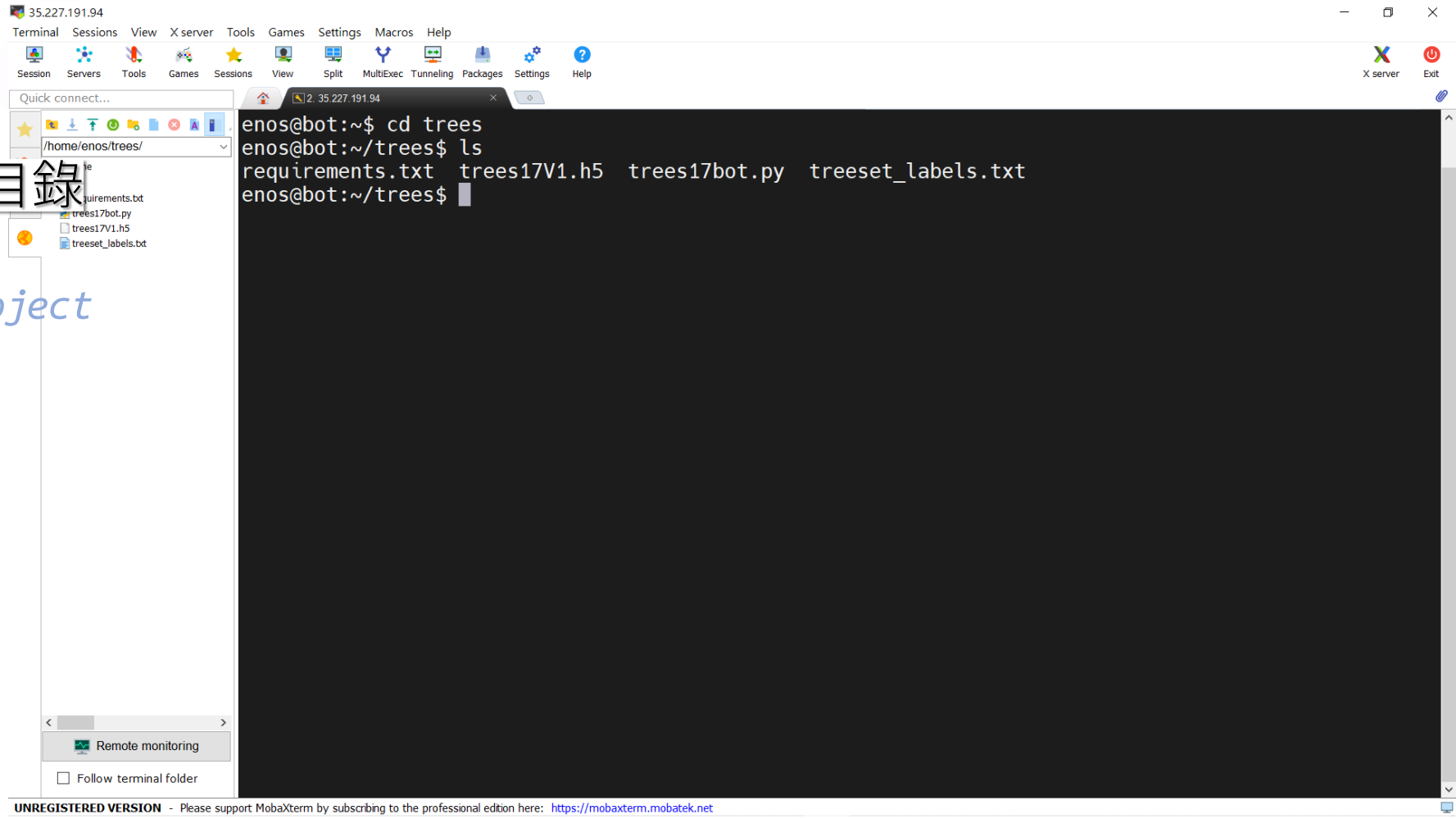
LINE Bot & WSGI

2. 部署服務

a. 進入 VM 專案目錄

```
cd; cd your_project
cd; cd trees

ls
```



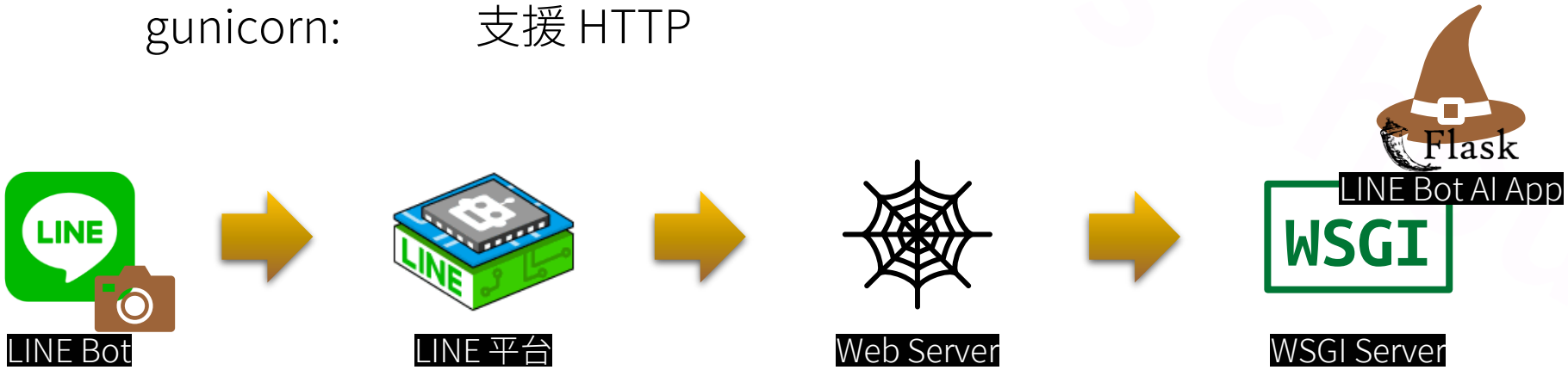
LINE Bot & WSGI

WSGI Server 的選擇

uWSGI vs gunicorn

uWSGI: 支援 HTTP，亦可搭配 NGINX 支援 uwsgi protocol

gunicorn: 支援 HTTP



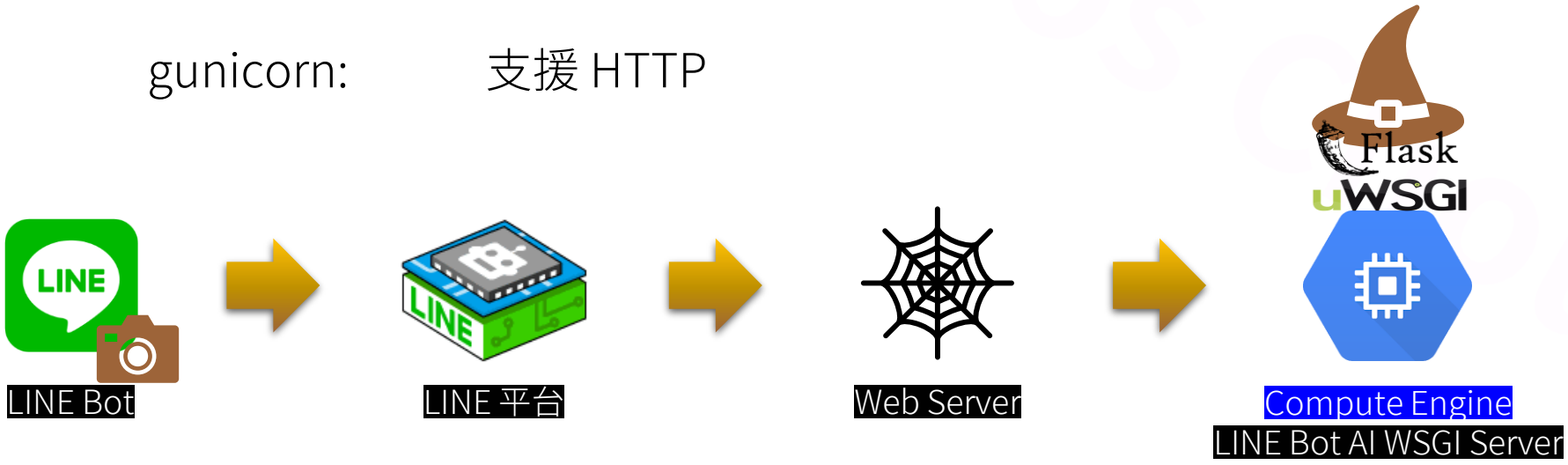
LINE Bot & WSGI

WSGI Server 的選擇

uWSGI vs gunicorn

- uWSGI: 支援 HTTP，亦可搭配 NGINX 支援 uwsgi protocol

  gunicorn: 支援 HTTP



LINE Bot & WSGI

2. 部署服務

b. 更新系統同時安裝 LINE Bot 與 WSGI Server (< 7 分)

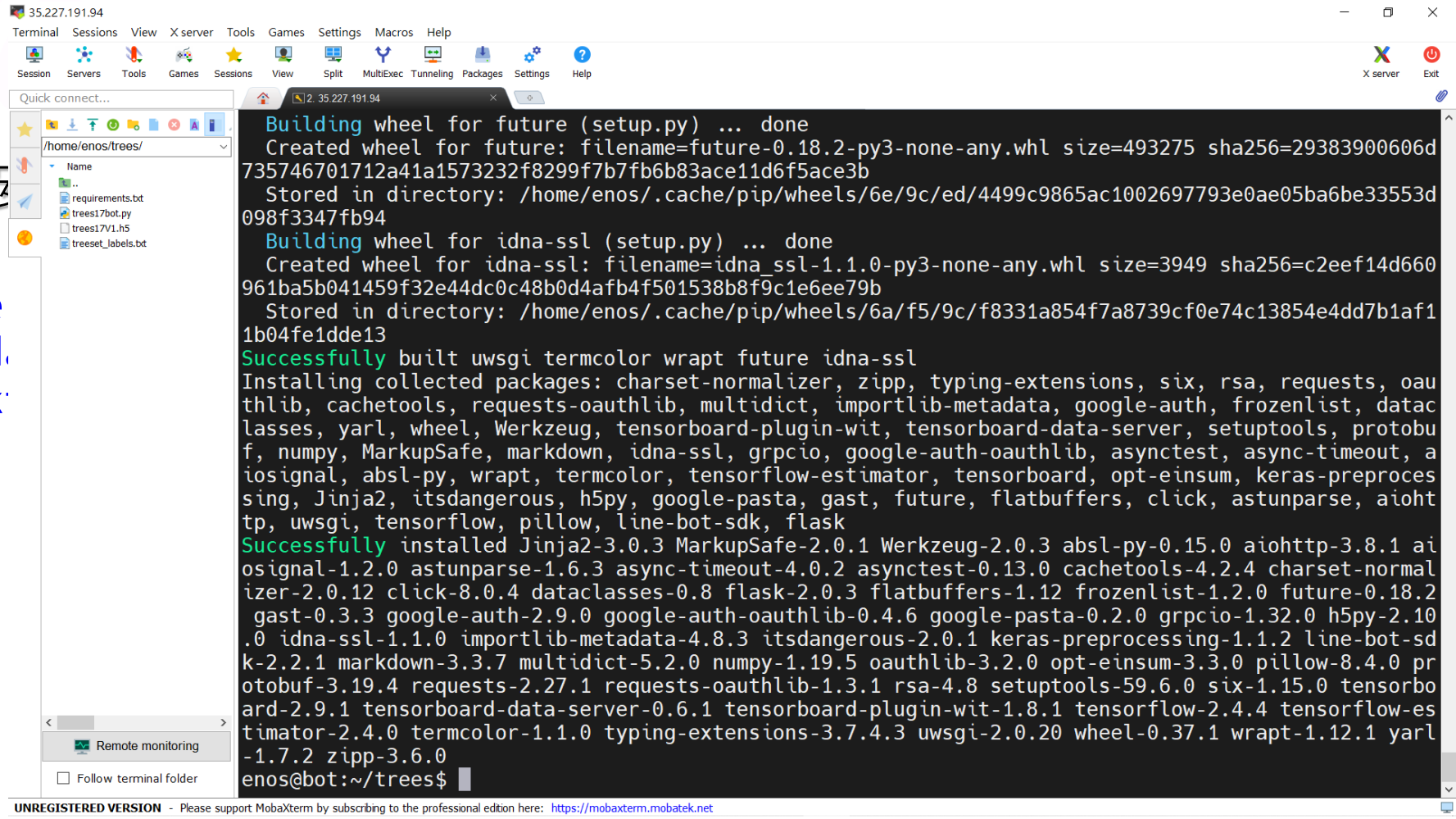
```
sudo apt update; sudo apt install -y python3-pip; pip3 install --upgrade
pip; sudo timedatectl set-timezone Asia/Taipei; python3 -m pip install -r
requirements.txt --no-warn-script-location; source ../.profile
```

LINE Bot & WSGI

2. 部署服務

b. 更新系統同時安

```
sudo apt update
pip; sudo timed
requirements.tx
```



LINE Bot & WSGI

3. 啟動服務

a. 進入 VM 專案目錄

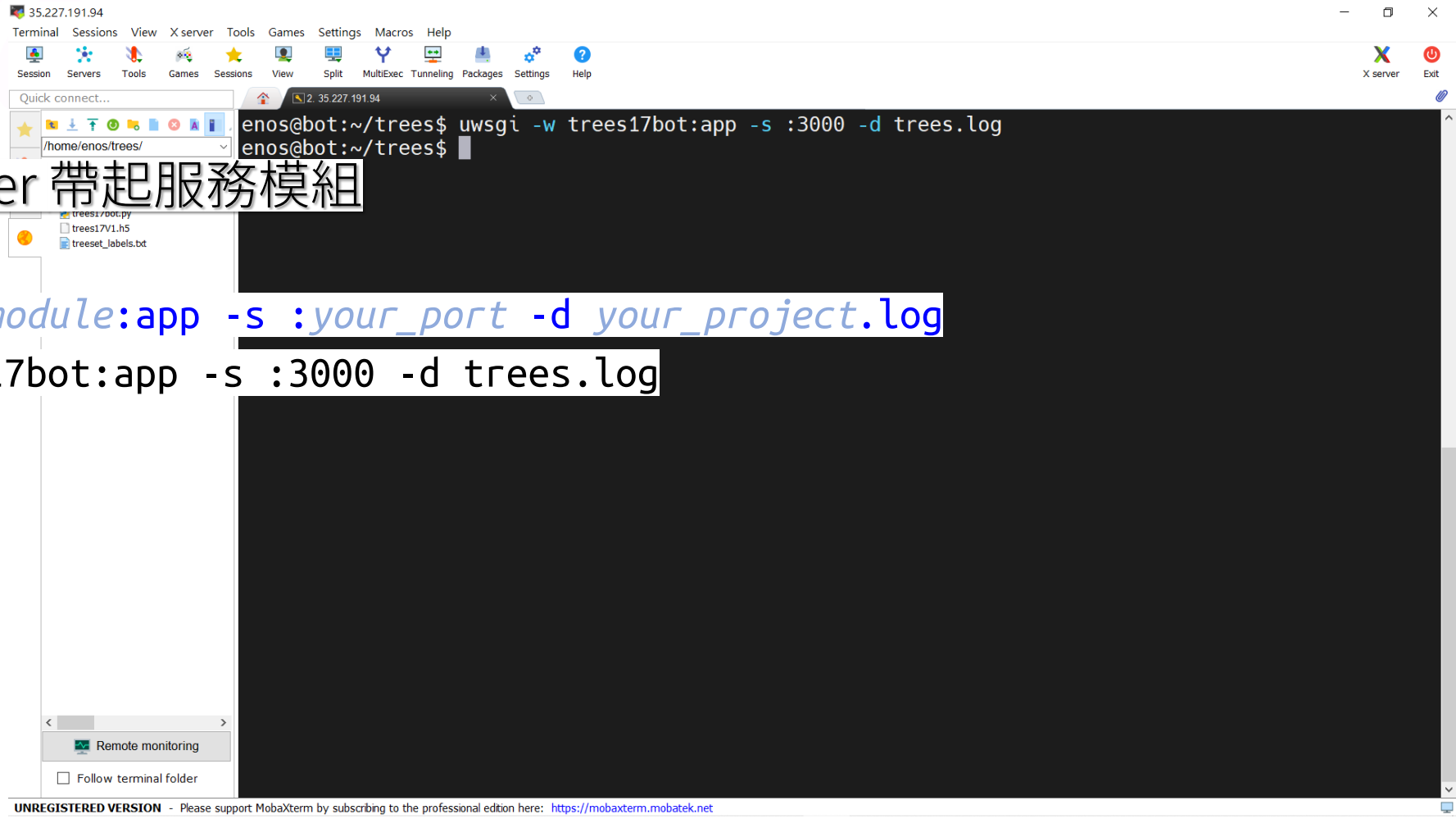
```
cd; cd your_project
cd; cd trees
```

LINE Bot & WSGI

3. 啟動服務

b. 以 WSGI Server 帶起服務模組

```
uwsgi -w your_module:app -s :your_port -d your_project.log
uwsgi -w trees17bot:app -s :3000 -d trees.log
```

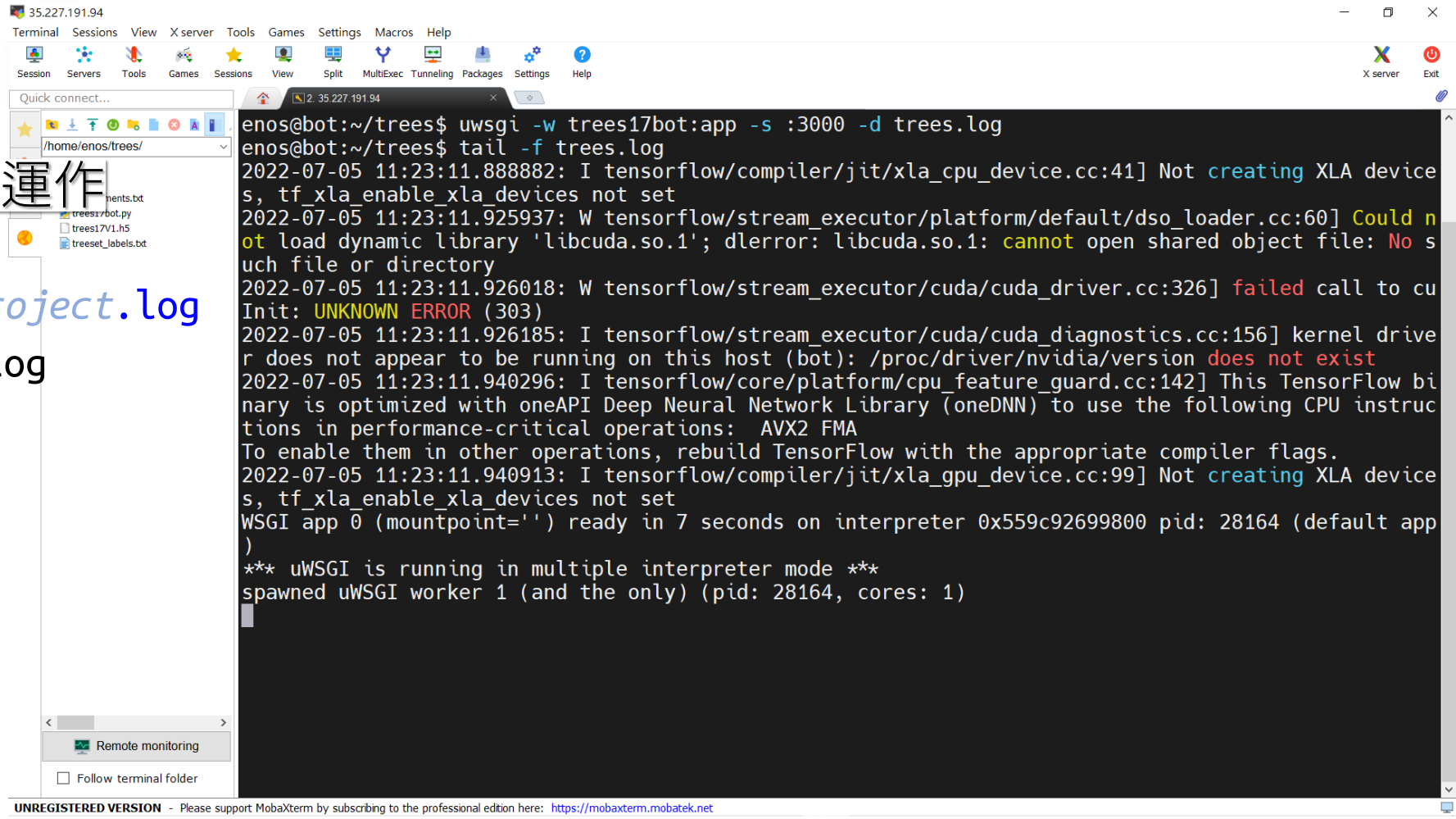


LINE Bot & WSGI

3. 啟動服務

c. 確認服務正常運作

```
tail -f your_project.log
tail -f trees.log
```



LINE Bot & WSGI

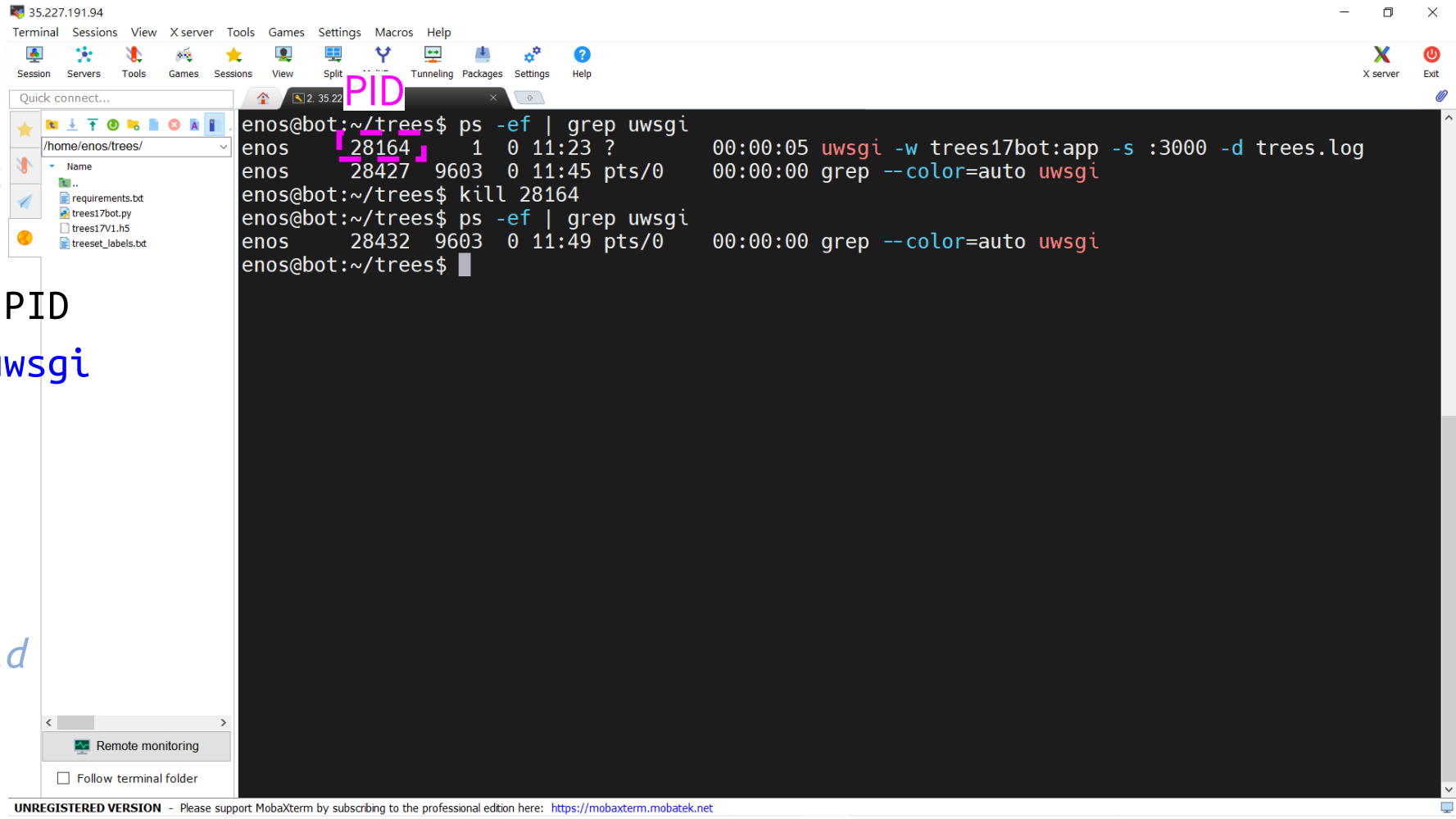
3. 啟動服務

d. 日常上下服務

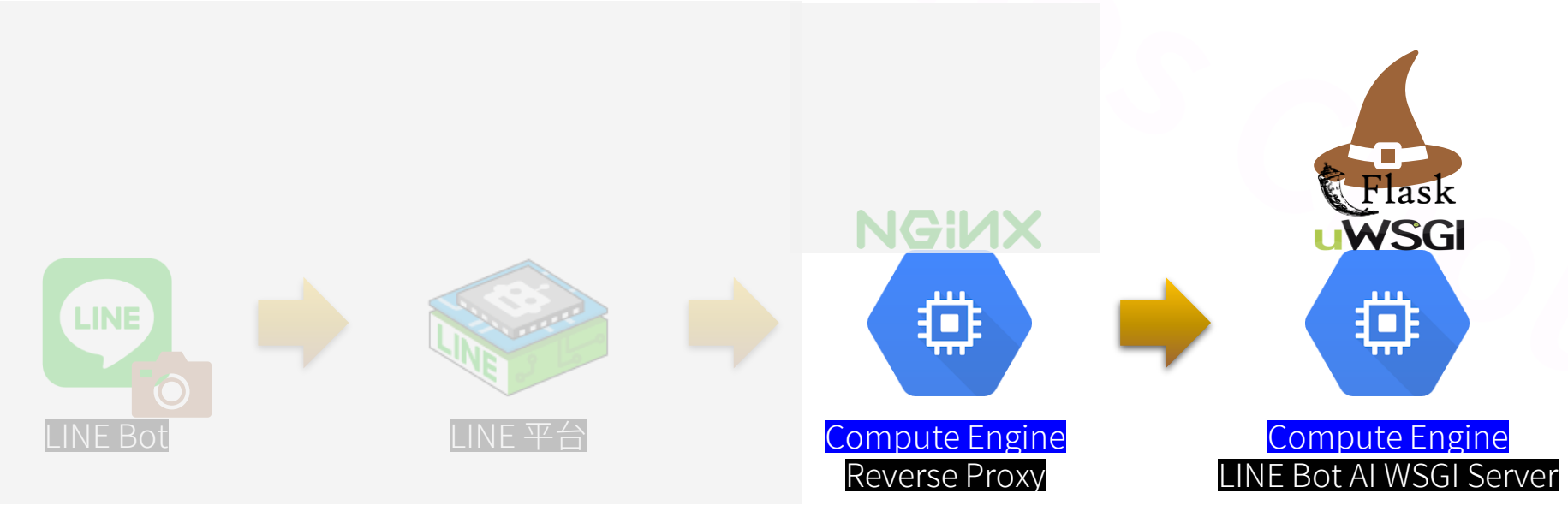
```
# 取得背景程序 PID
ps -ef | grep uwsgi

# 移除背景程序
kill your_pid

kill -9 your_pid
```



架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging



申購並設定網域

何謂網域？

a. 網域為可讀且符合格式的字串，對應特定 IP Address
b. 主網擁用有者可設定多組子網域

中央氣象局

	IP Address	俗稱
cwb.gov.tw		主網域
www.cwb.gov.tw	210.61.218.11	子網域
opendata.cwb.gov.tw	61.56.14.16	子網域
opendata.cwb.gov.tw/dataset		子目錄

申購並設定網域

何謂 CAA？

RFC 8659

DNS Certification Authority Authorization (CAA) Resource Record

RFC: Request For Comments

申購並設定網域

已知支援 CAA 的網域供應商

⭕ 1. GoDaddy
⭕ 2. PC home 買網址
🔺 3. Cloudflare (感謝台北 AI 班 4 期推薦，待實證)

已知不支援 CAA 的網域供應商

❌ 1. Seednet
❌ 2. Hinet

申購並設定網域

GoDaddy

https://tw.godaddy.com/

申購並設定網域

GoDaddy

a. 申購網域

申購並設定網域

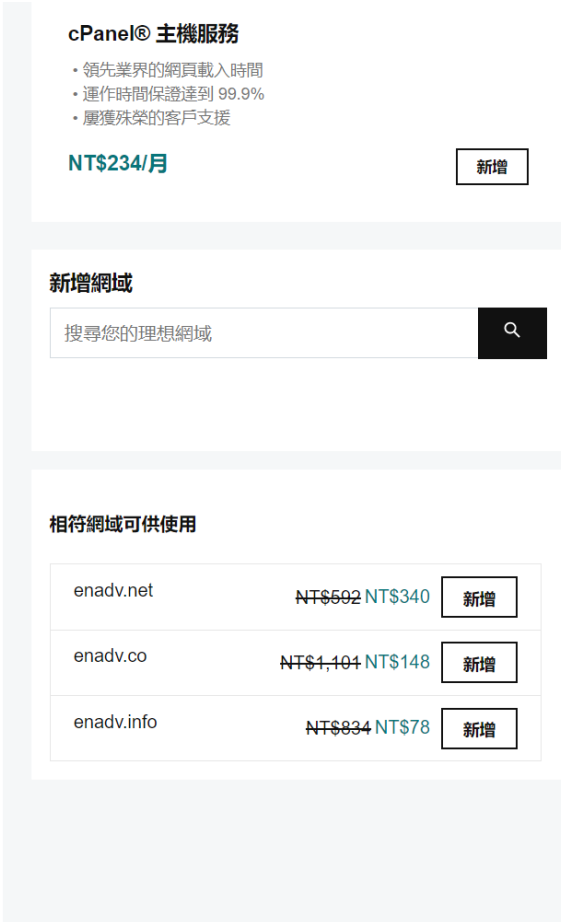
GoDaddy

a. 申購網域

申購並設定網域

GoDaddy

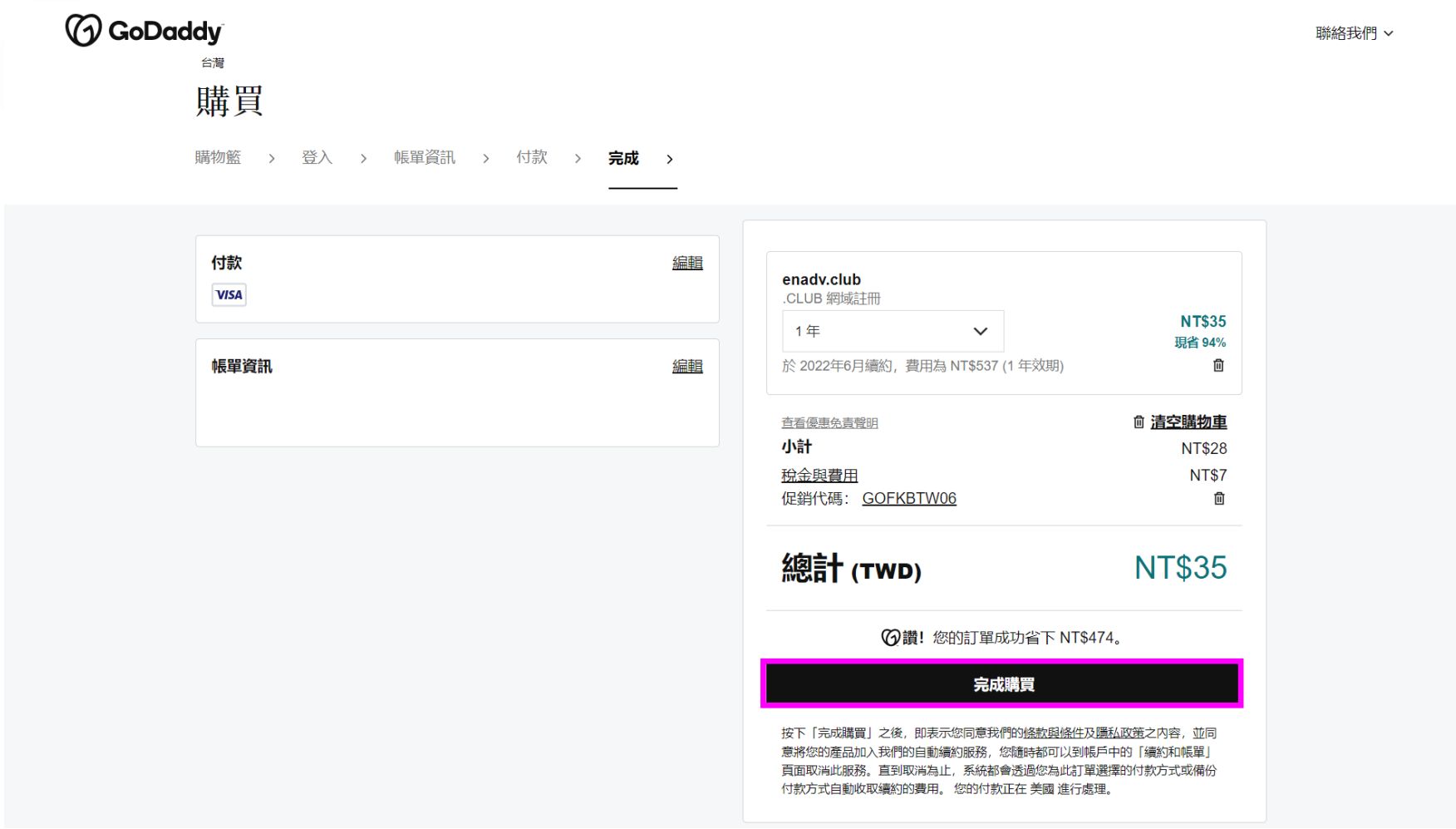
a. 申購網域



申購並設定網域

GoDaddy

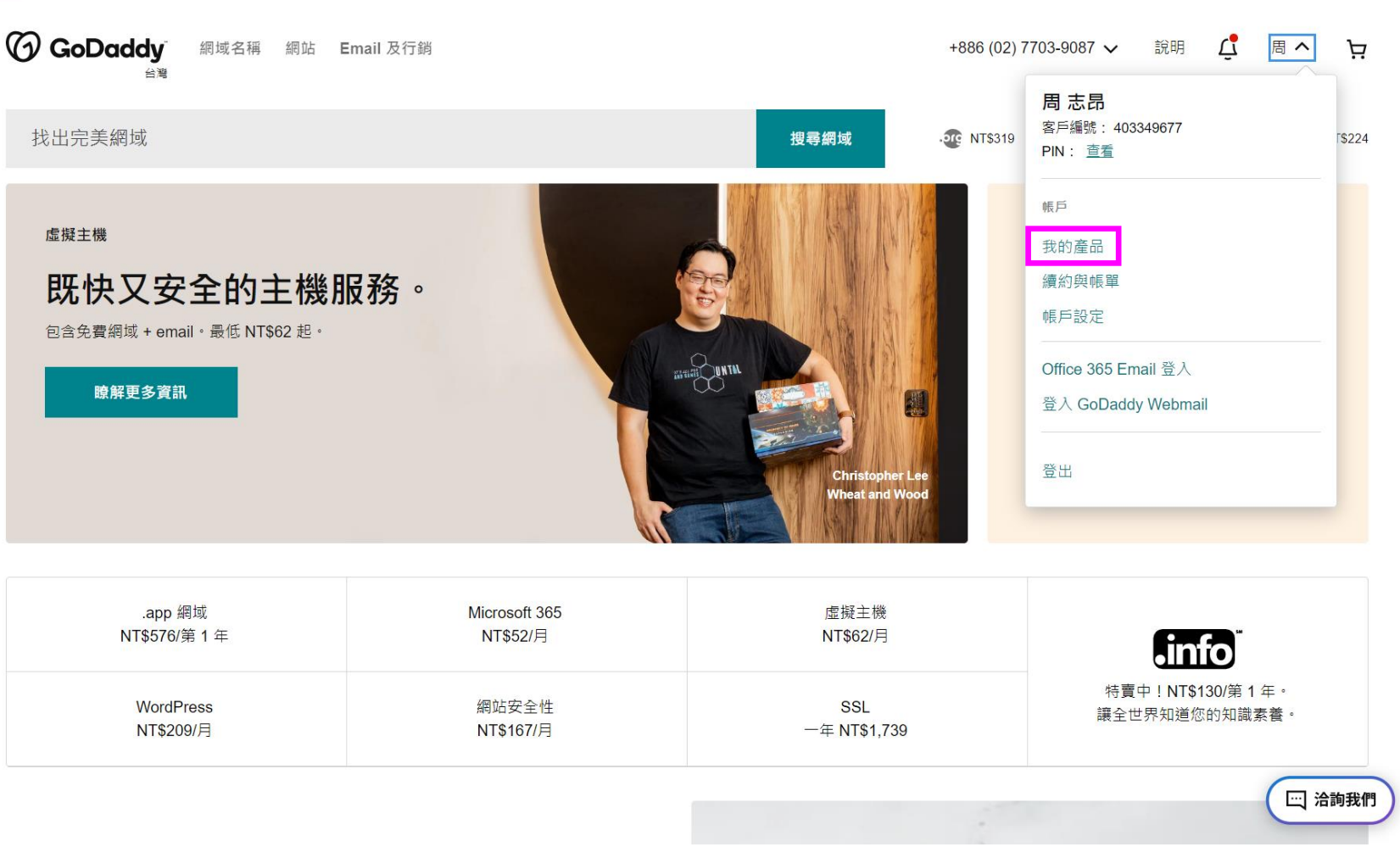
a. 申購網域



申購並設定網域

GoDaddy

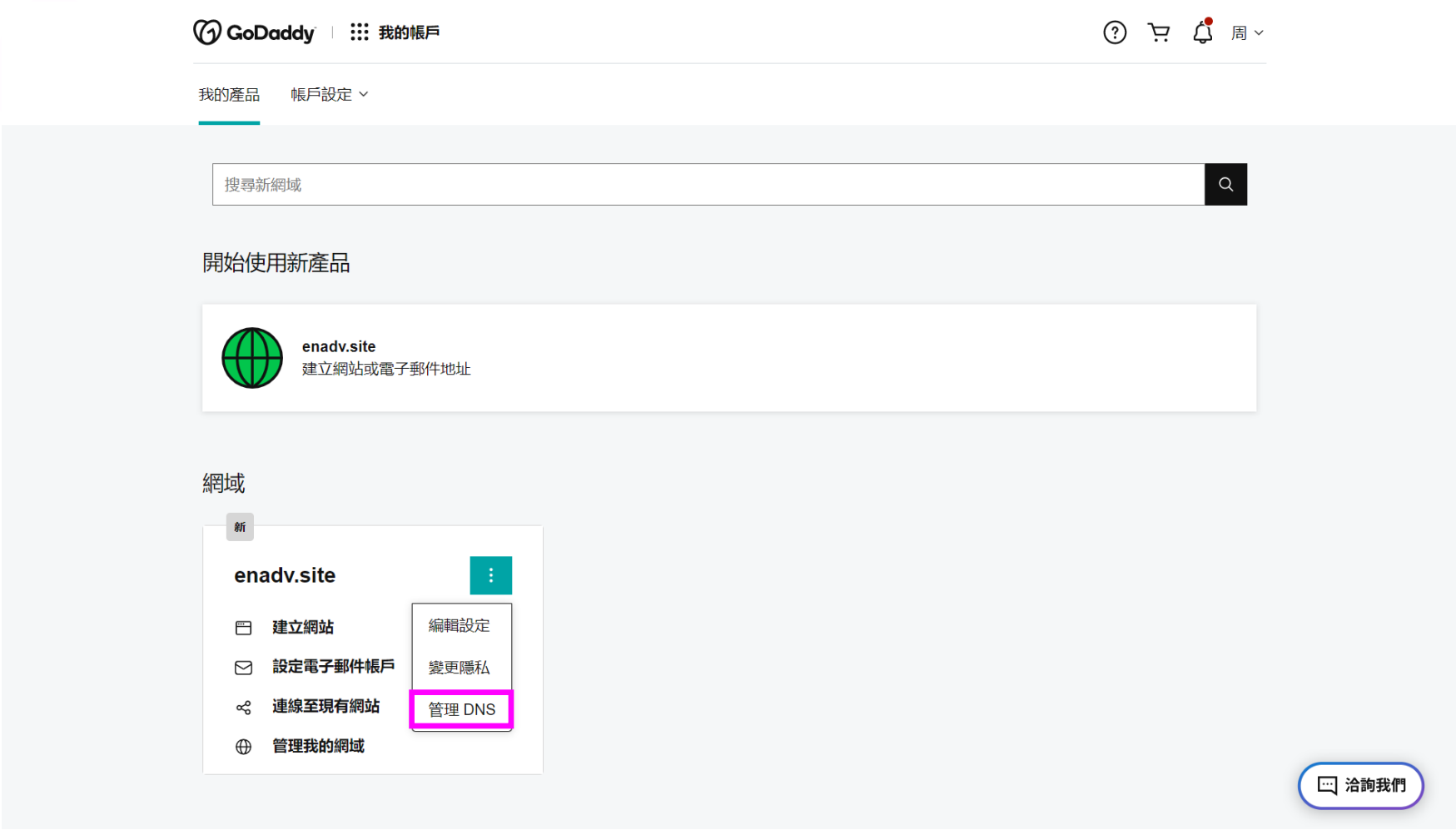
b. 設定網域



申購並設定網域

GoDaddy

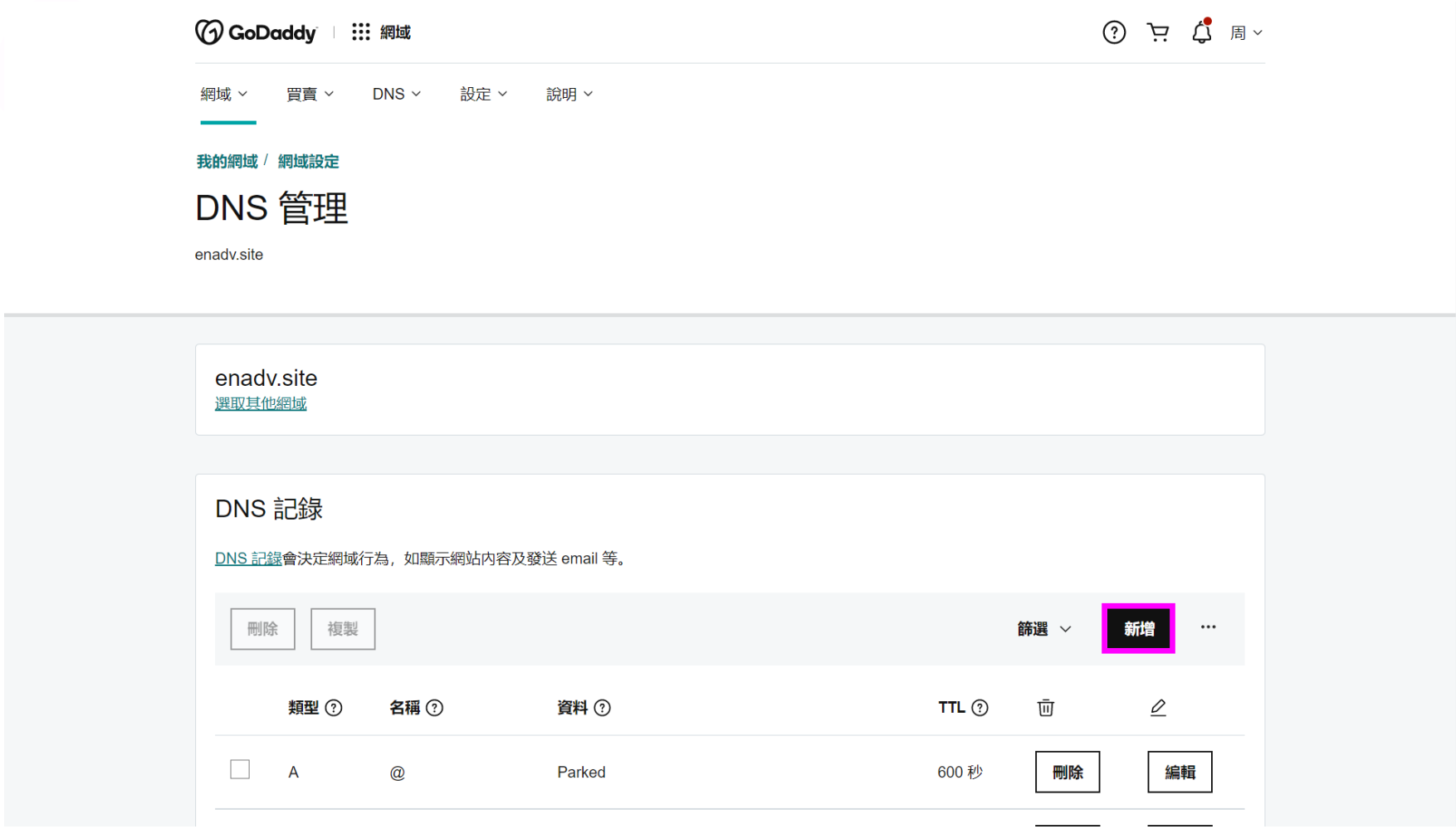
b. 設定網域



申購並設定網域

GoDaddy

b. 設定網域



申購並設定網域

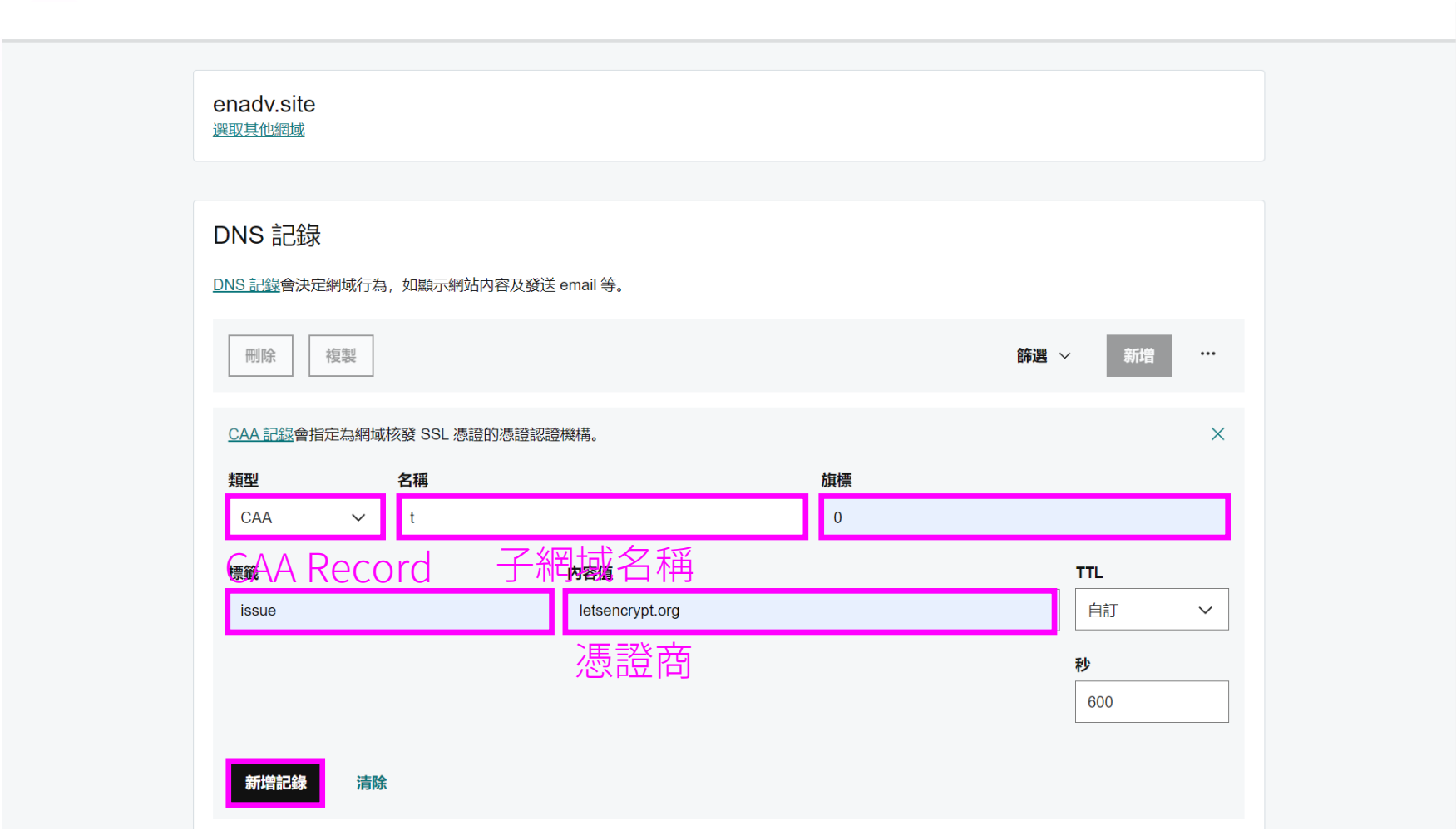
GoDaddy

b. 設定網域

申購並設定網域

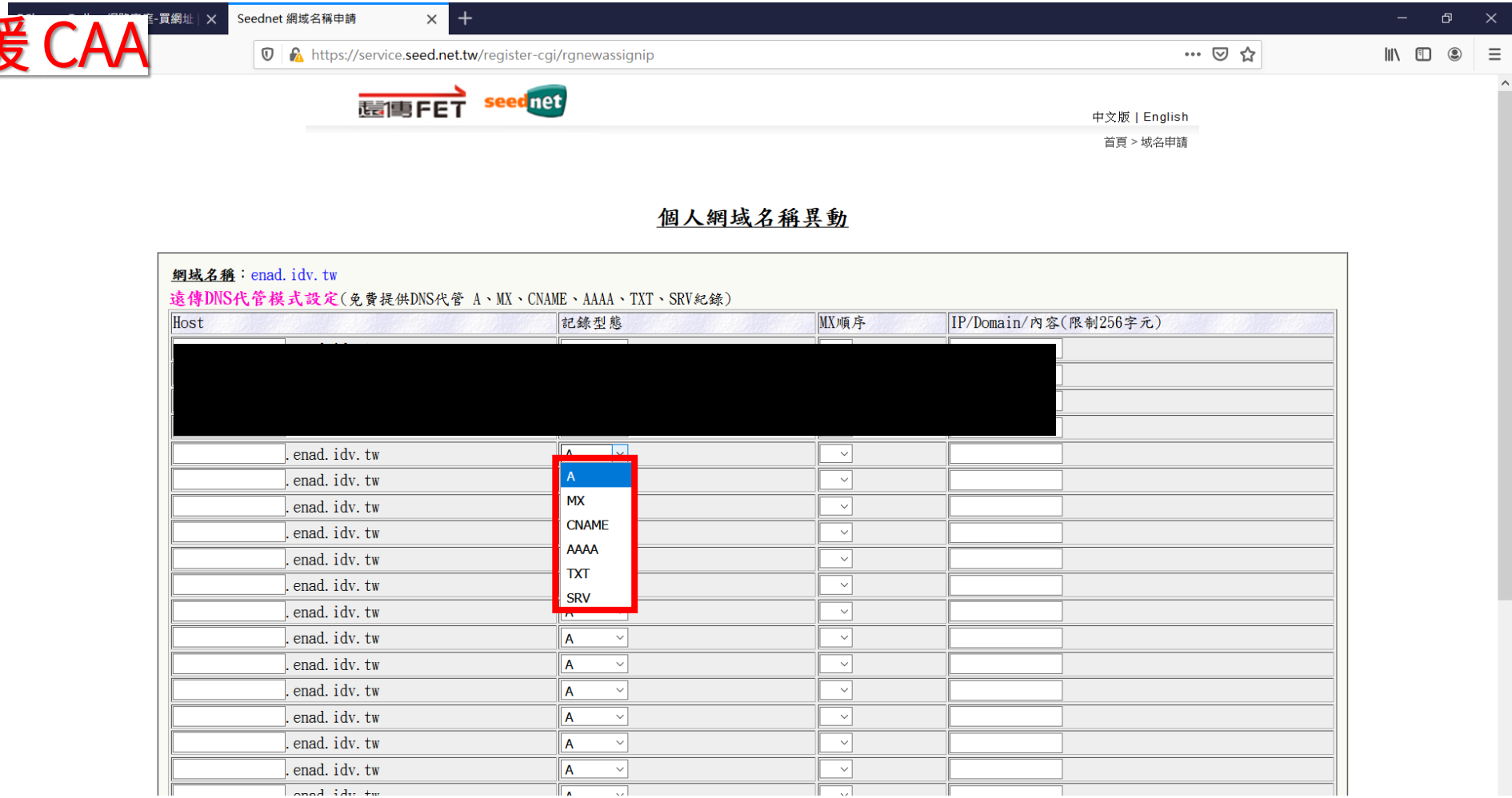
GoDaddy

b. 設定網域



申購並設定網域

Seednet 不支援 CAA



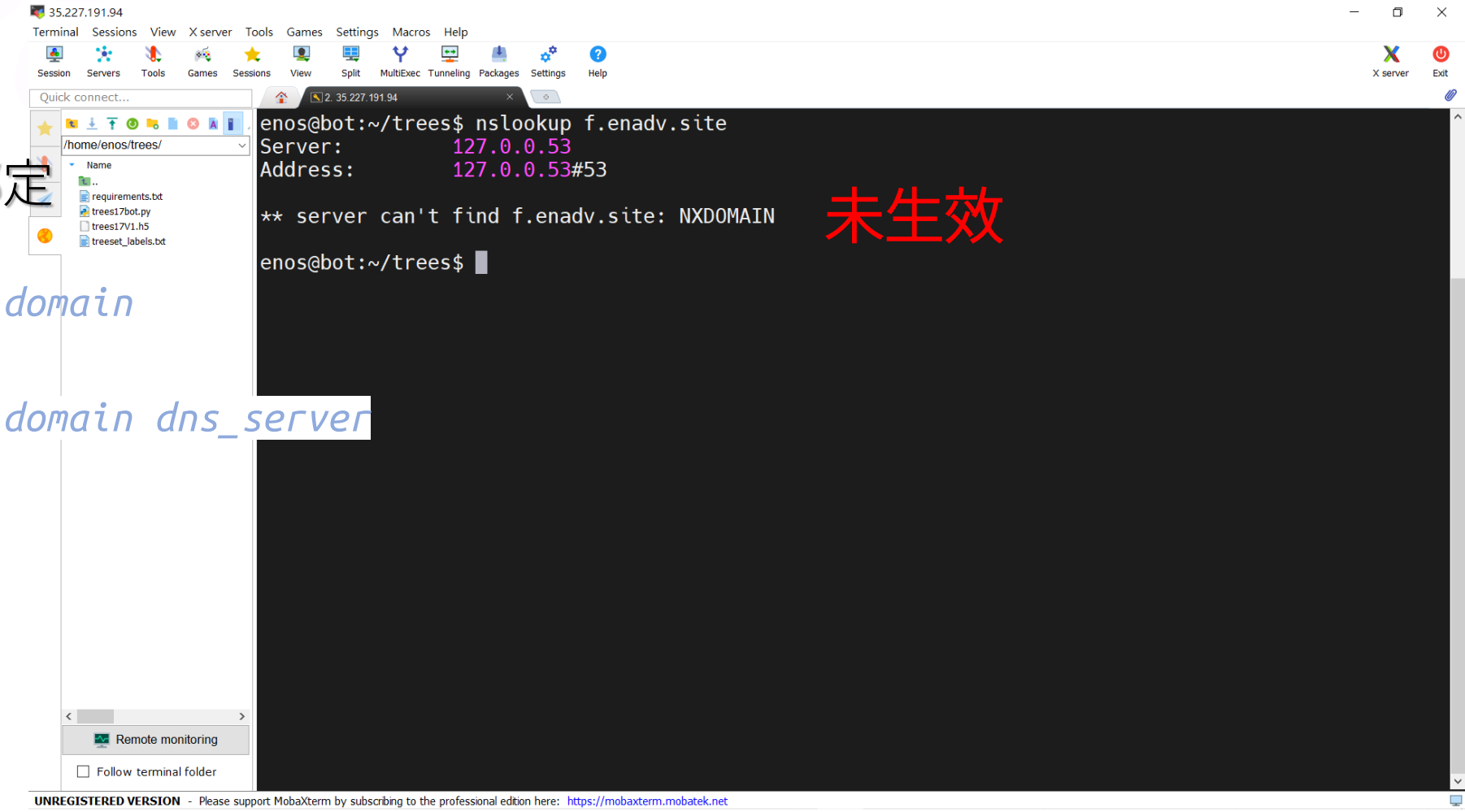
申購並設定網域

驗證網域

確認網域與 IP 綁定

```
nslookup your_domain
```

```
nslookup your_domain dns_server
```



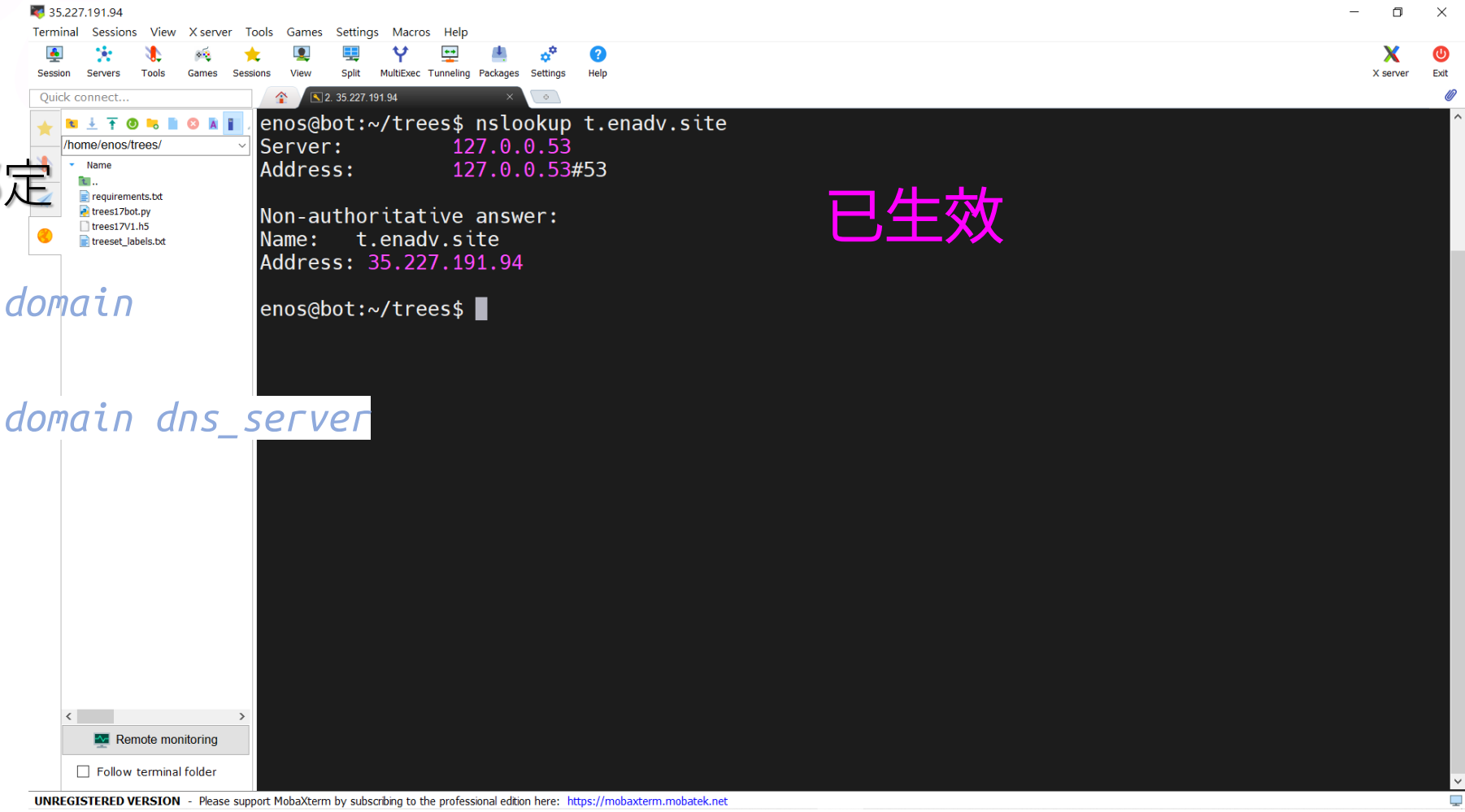
申購並設定網域

驗證網域

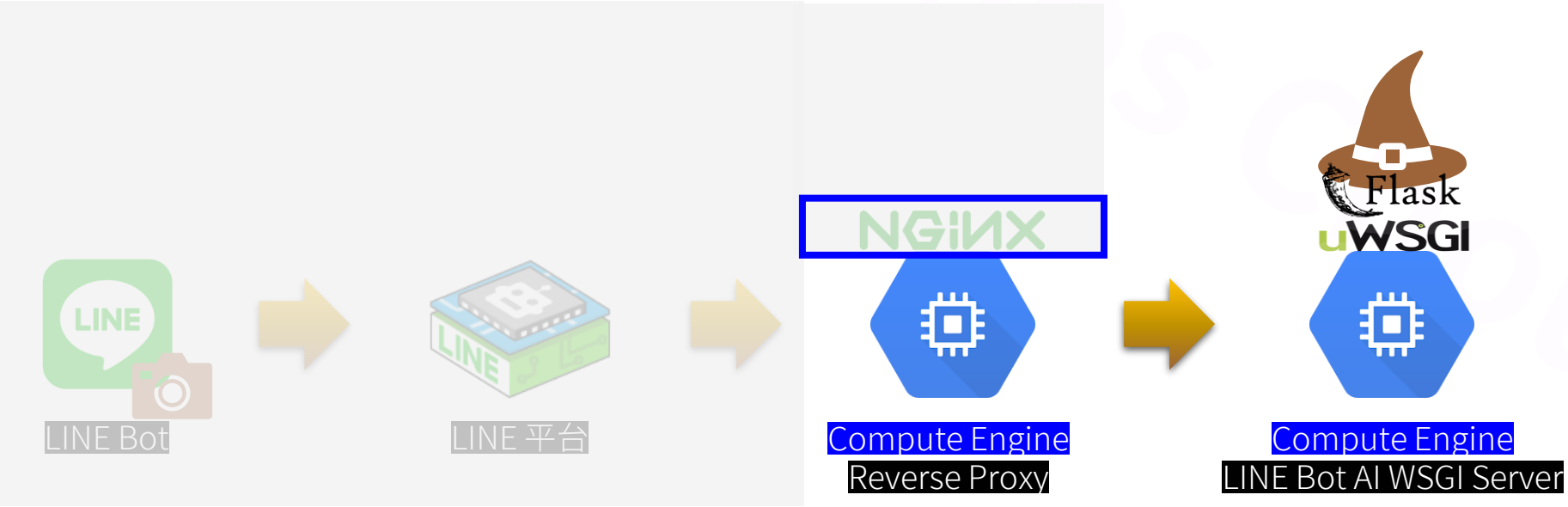
確認網域與 IP 綁定

```
nslookup your_domain
```

```
nslookup your_domain dns_server
```



架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging



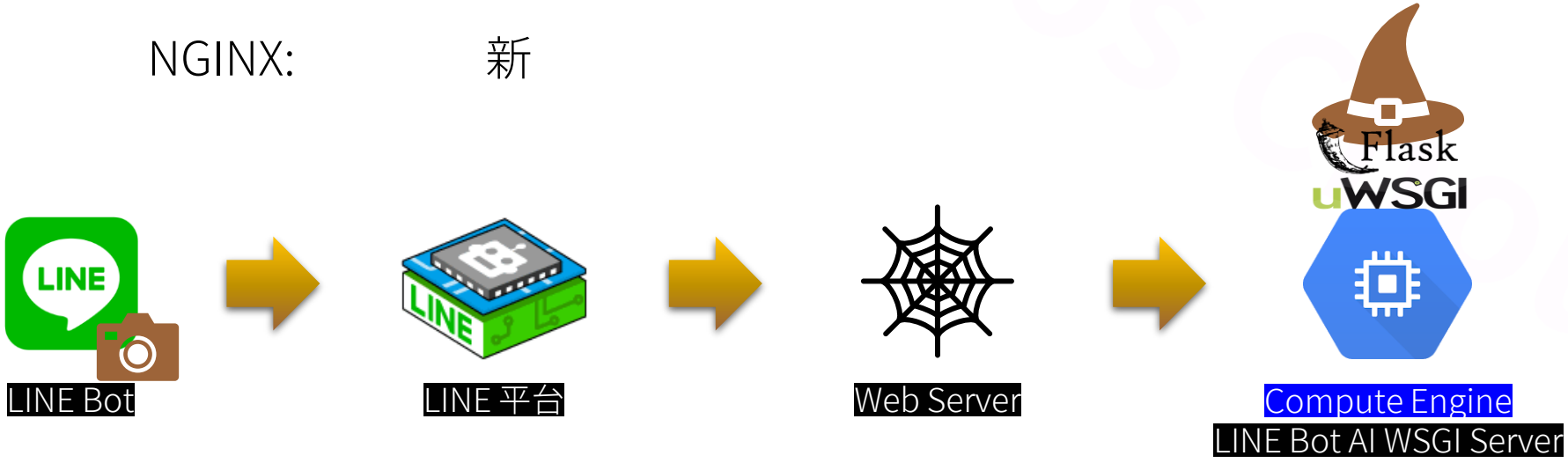
網站與憑證

Web Server 的選擇

Apache vs NGINX

Apache: 經典

NGINX: 新



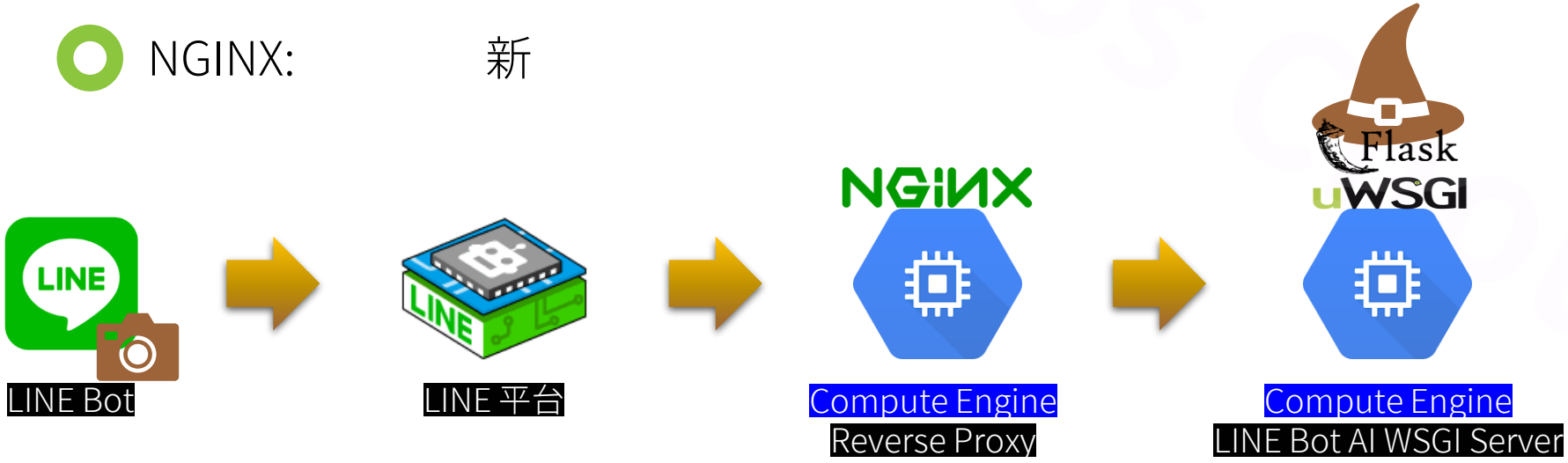
網站與憑證

Web Server 的選擇

Apache vs NGINX

Apache: 經典

NGINX: 新

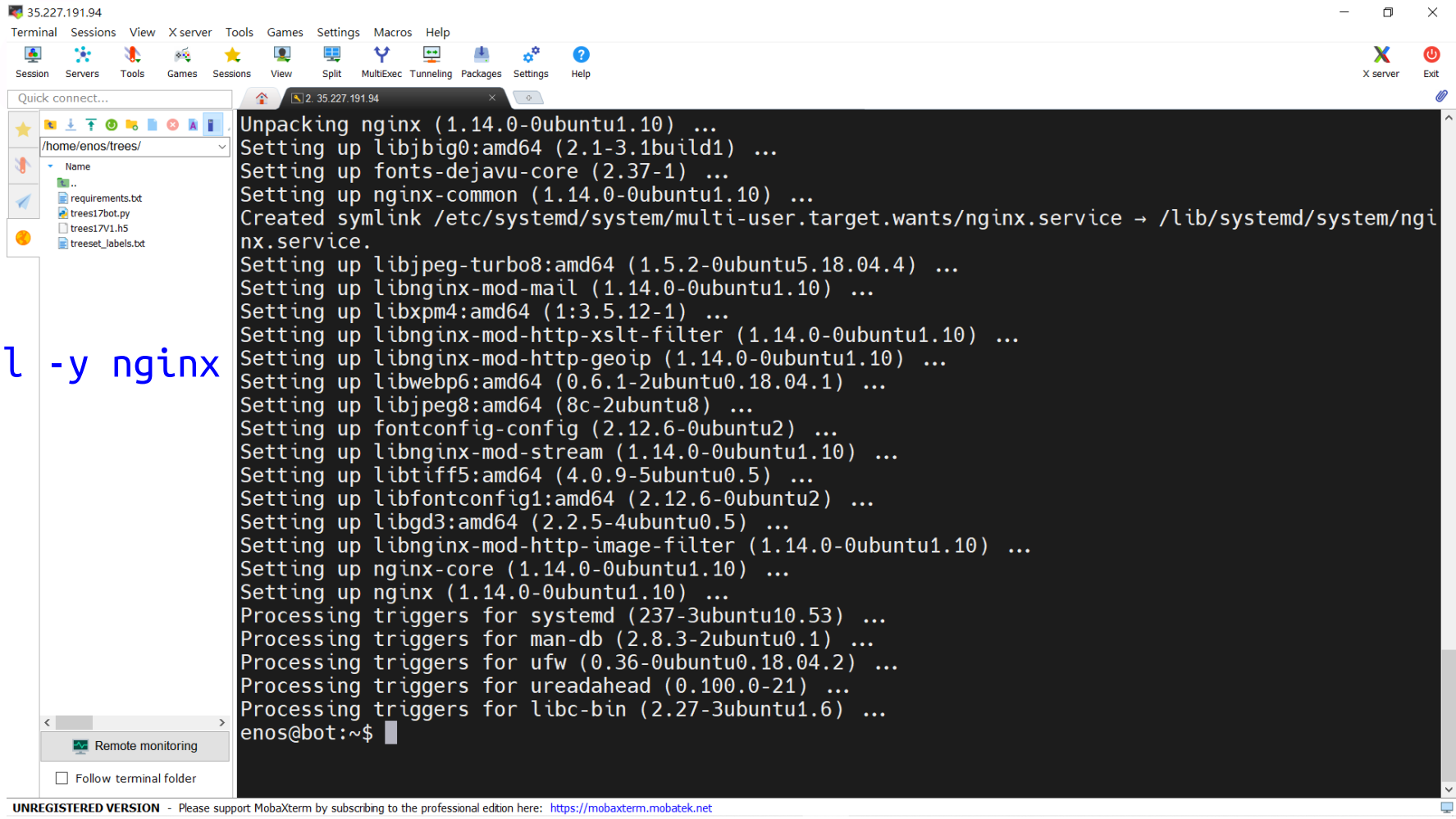


網站與憑證

1. 架設 NGINX

a. 安裝 NGINX

```
# 登入 VM
sudo apt install -y nginx
```

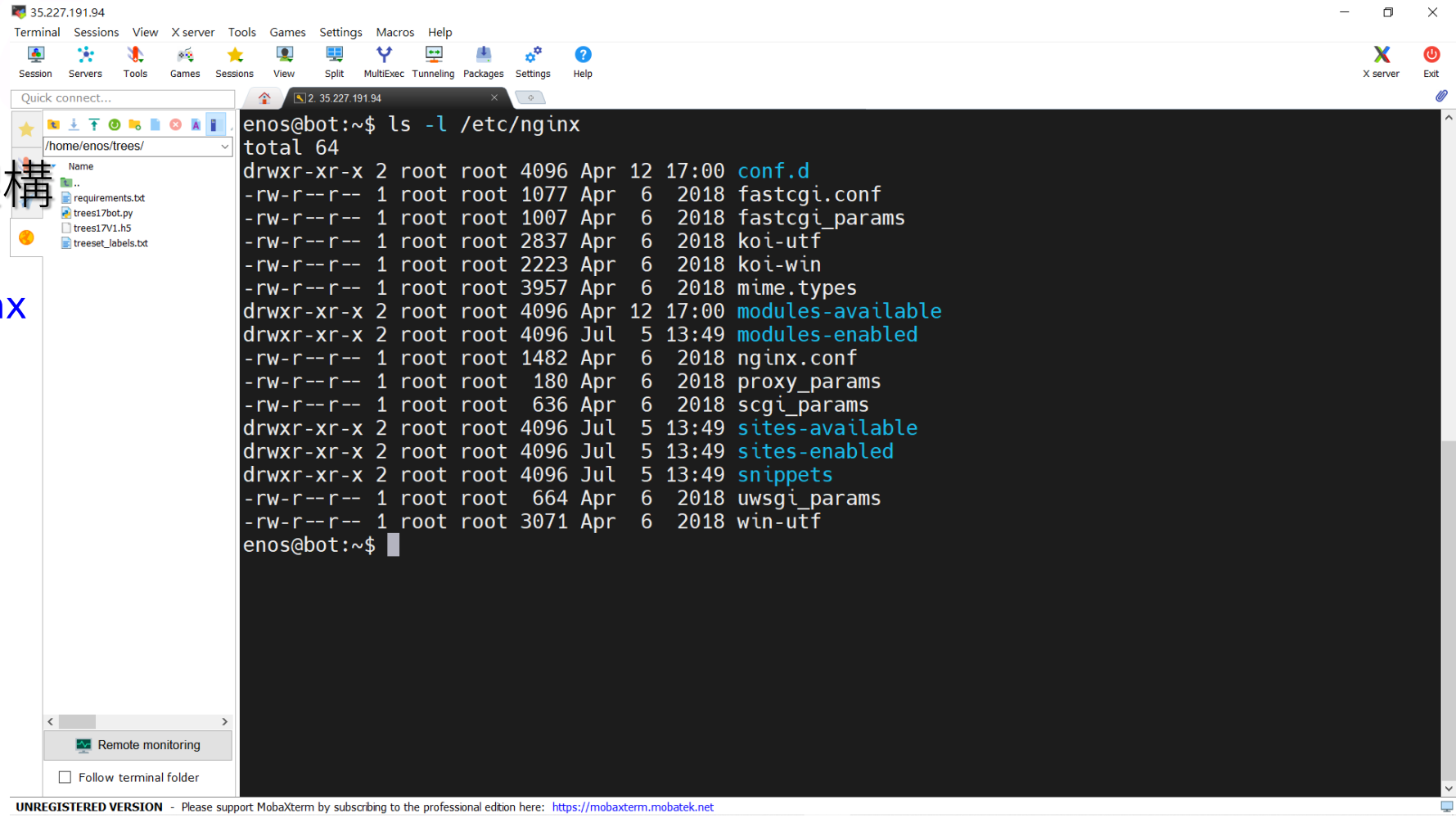


網站與憑證

1. 架設 NGINX

   b. 觀察組態檔架構

```
ls -l /etc/nginx
```



網站與憑證

1. 架設 NGINX

b. 觀察組態檔架構

```
ls -l /etc/nginx
```

nginx.conf

NGINX 啟動時預設讀取的參數檔，內容包含掃描 sites-enabled 中所有參數檔

sites-available 目錄

儲存各種啟動參數檔，但未必會被啟動。欲啟動的參數檔應以 symbolic link 方式置於 sites-enabled 目錄

sites-enabled 目錄

實際欲啟動的參數檔

網站與憑證

1. 架設 NGINX

c. 建立組態檔

```
sudo nano /etc/nginx/sites-available/your_project.conf
sudo nano /etc/nginx/sites-available/trees.conf
```

```
server {
    server_name your_domain;
    location / {
        include uwsgi_params;
        uwsgi_pass your_ip:your_port;
    }
}
```

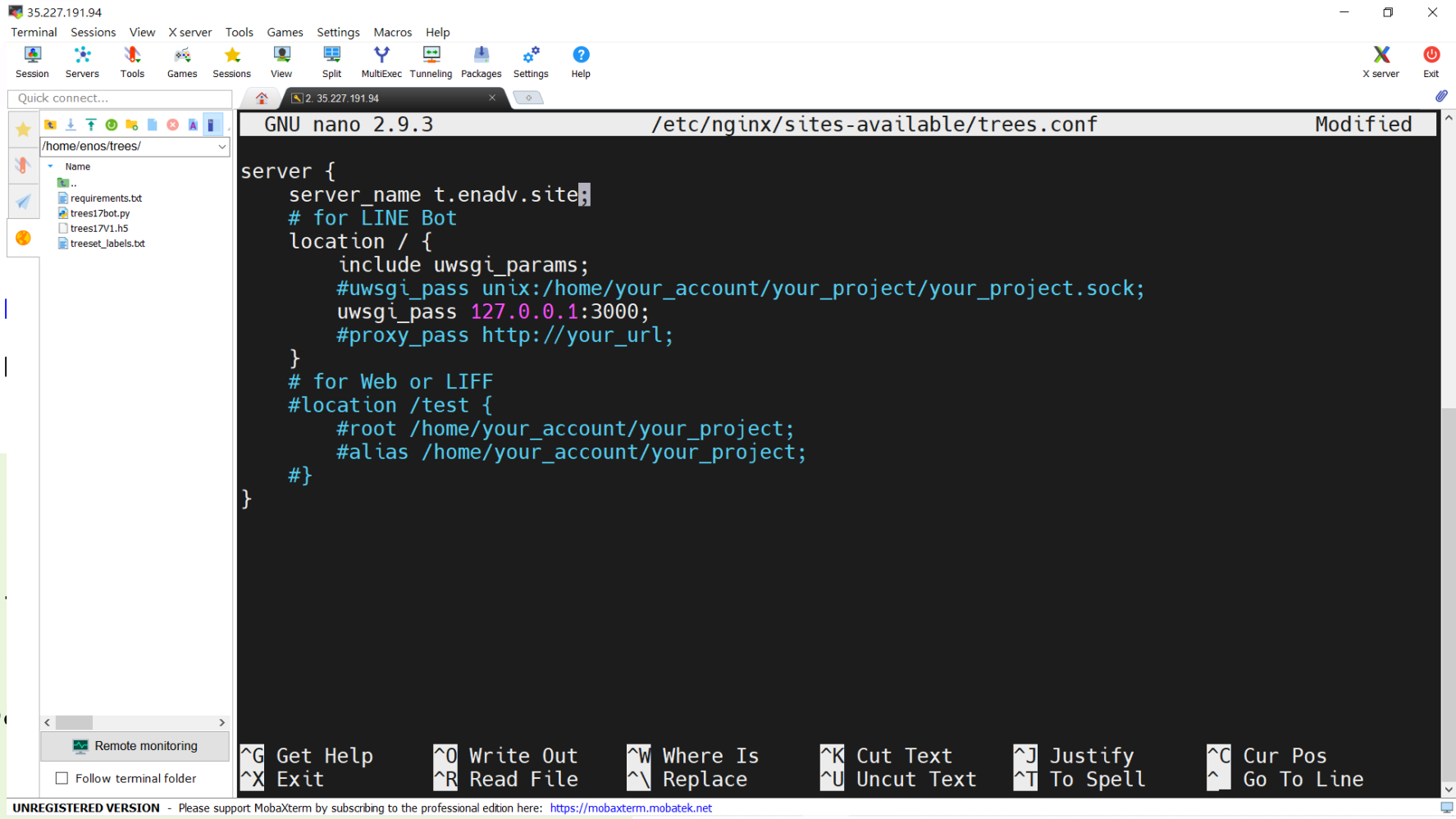
網站與憑證

1. 架設 NGINX

c. 建立組態檔

```
sudo nano /etc/
sudo nano /etc/
```

```
server {
    server_name
    location / 
        include
        uwsgi_p
    }
}
```



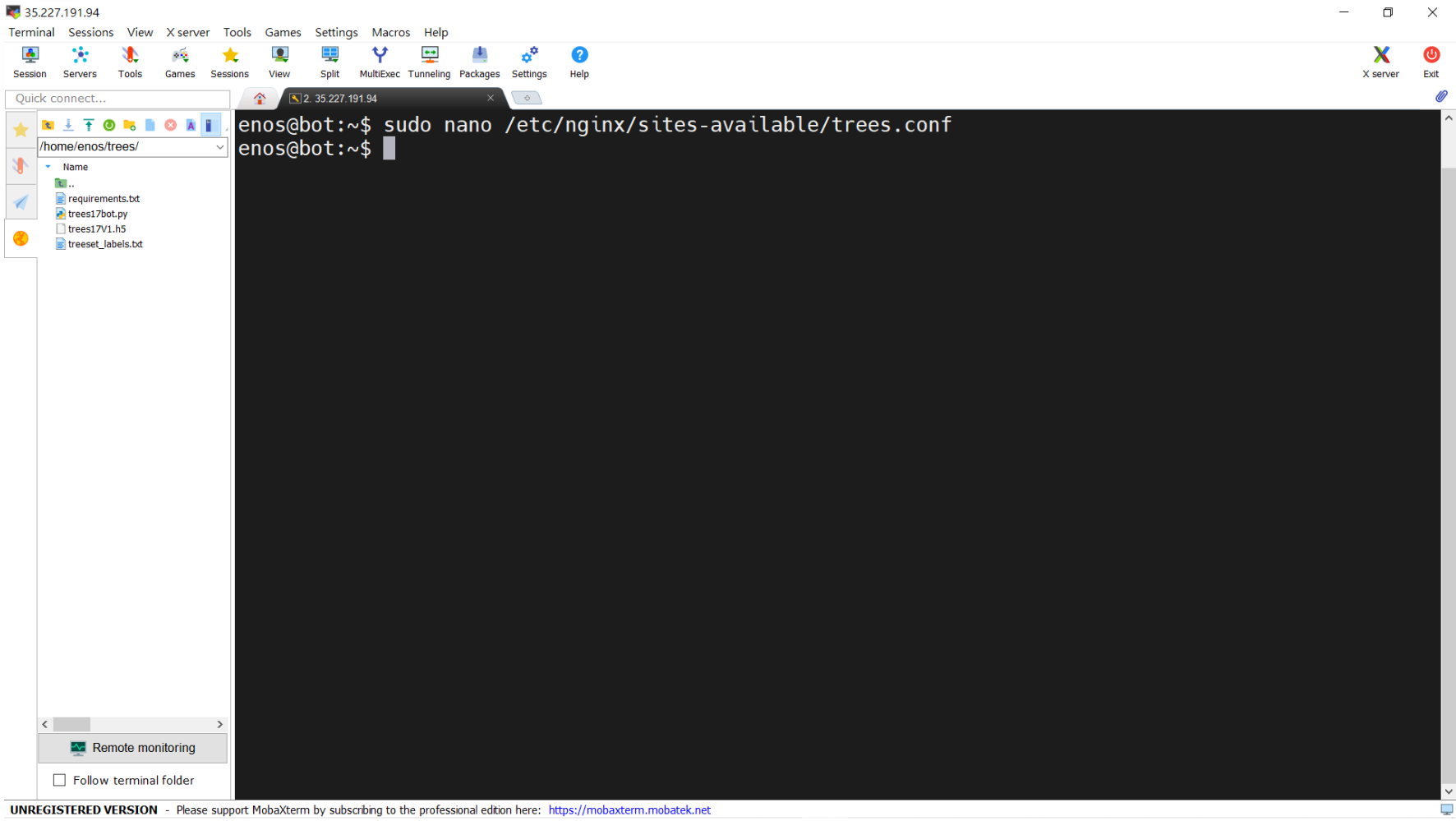
網站與憑證

1. 架設 NGINX

c. 建立組態檔

儲存
Ctrl + O

退出
Ctrl + X

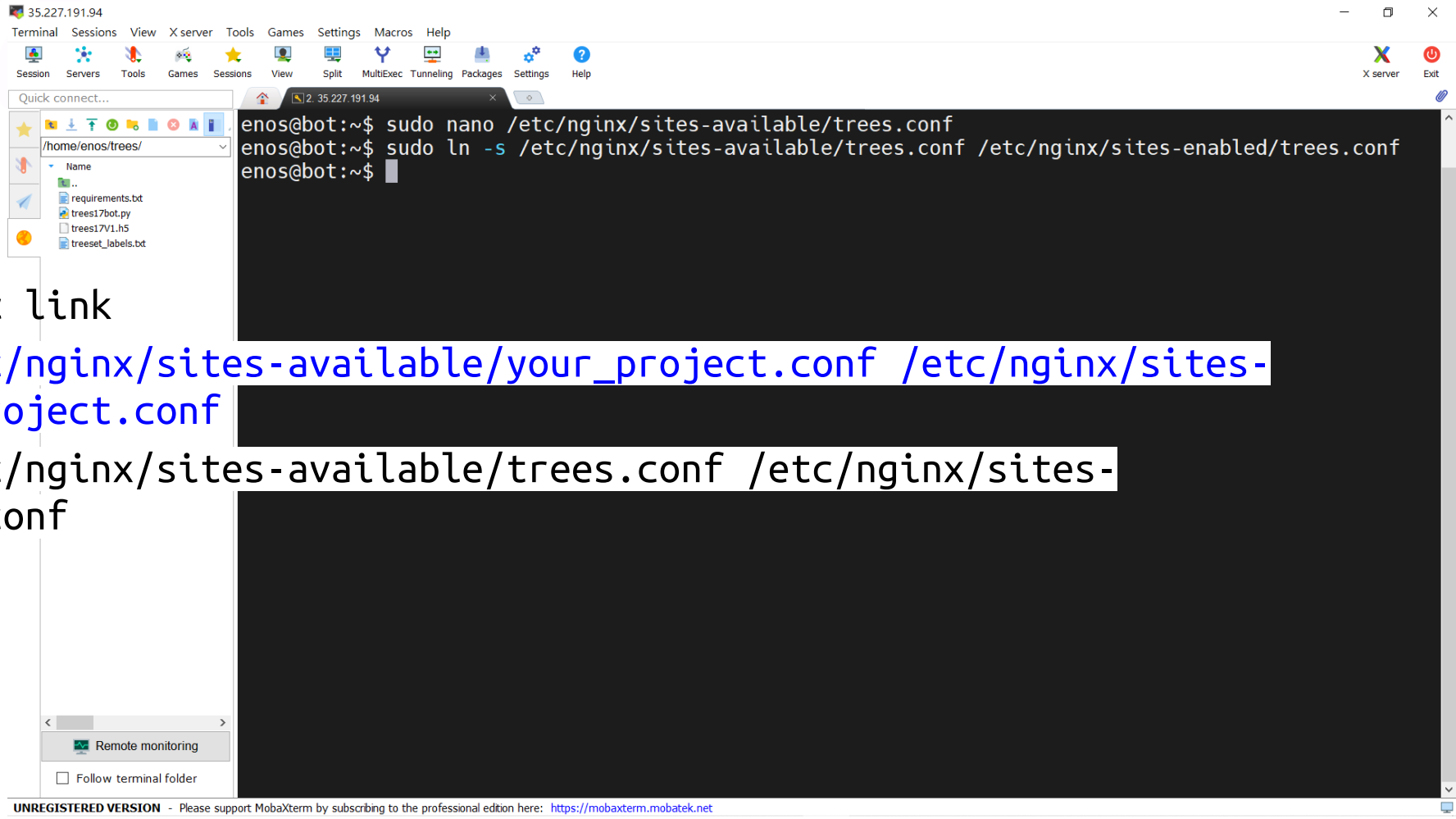


網站與憑證

1. 架設 NGINX

c. 建立組態檔

```
# 建立 symbolic link
sudo ln -s /etc/nginx/sites-available/your_project.conf /etc/nginx/sites-enabled/your_project.conf
sudo ln -s /etc/nginx/sites-available/trees.conf /etc/nginx/sites-enabled/trees.conf
```



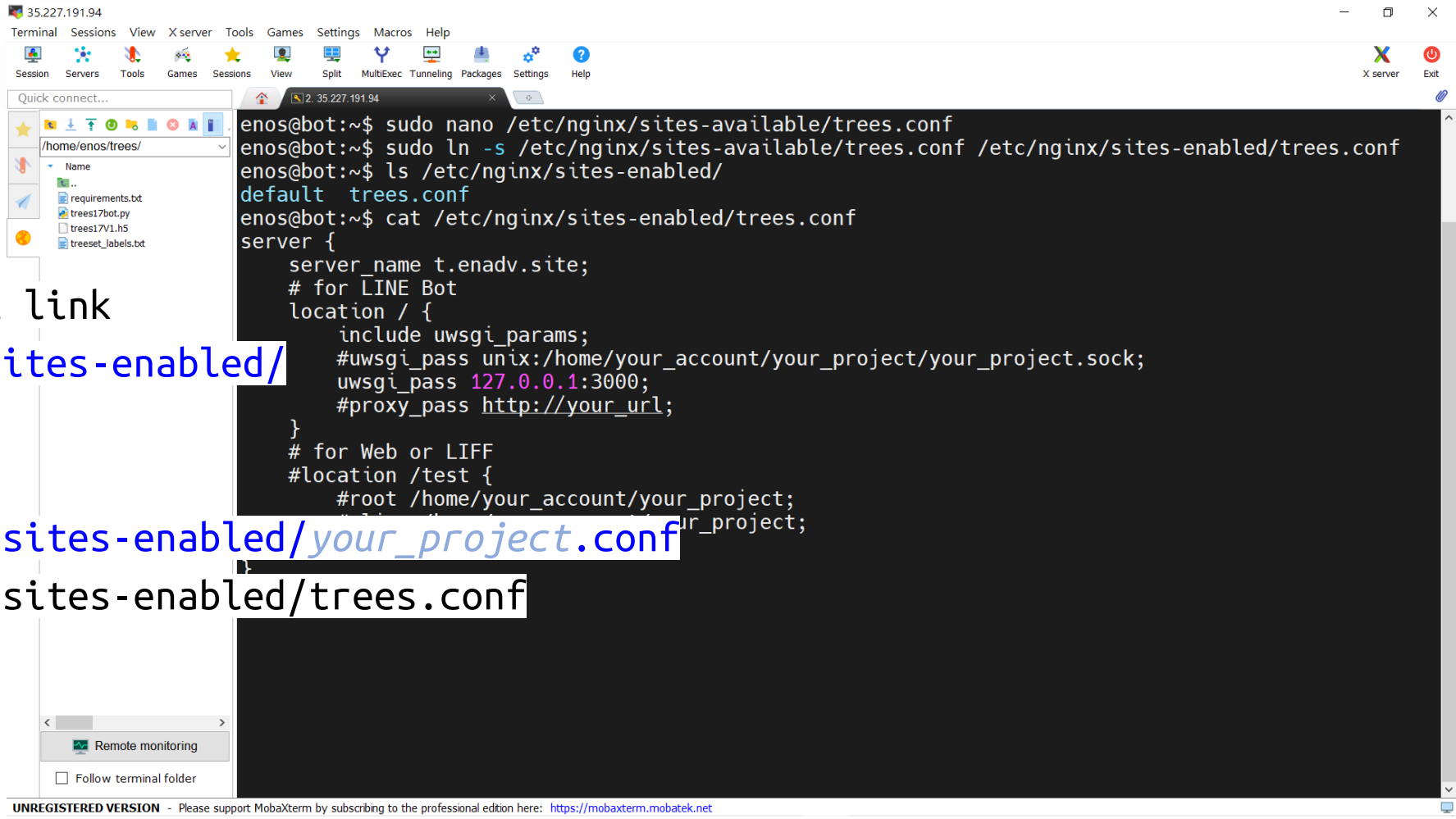
網站與憑證

1. 架設 NGINX

c. 建立組態檔

```
# 確認 symbolic link
ls /etc/nginx/sites-enabled/

# 確認組態檔
cat /etc/nginx/sites-enabled/your_project.conf
cat /etc/nginx/sites-enabled/trees.conf
```

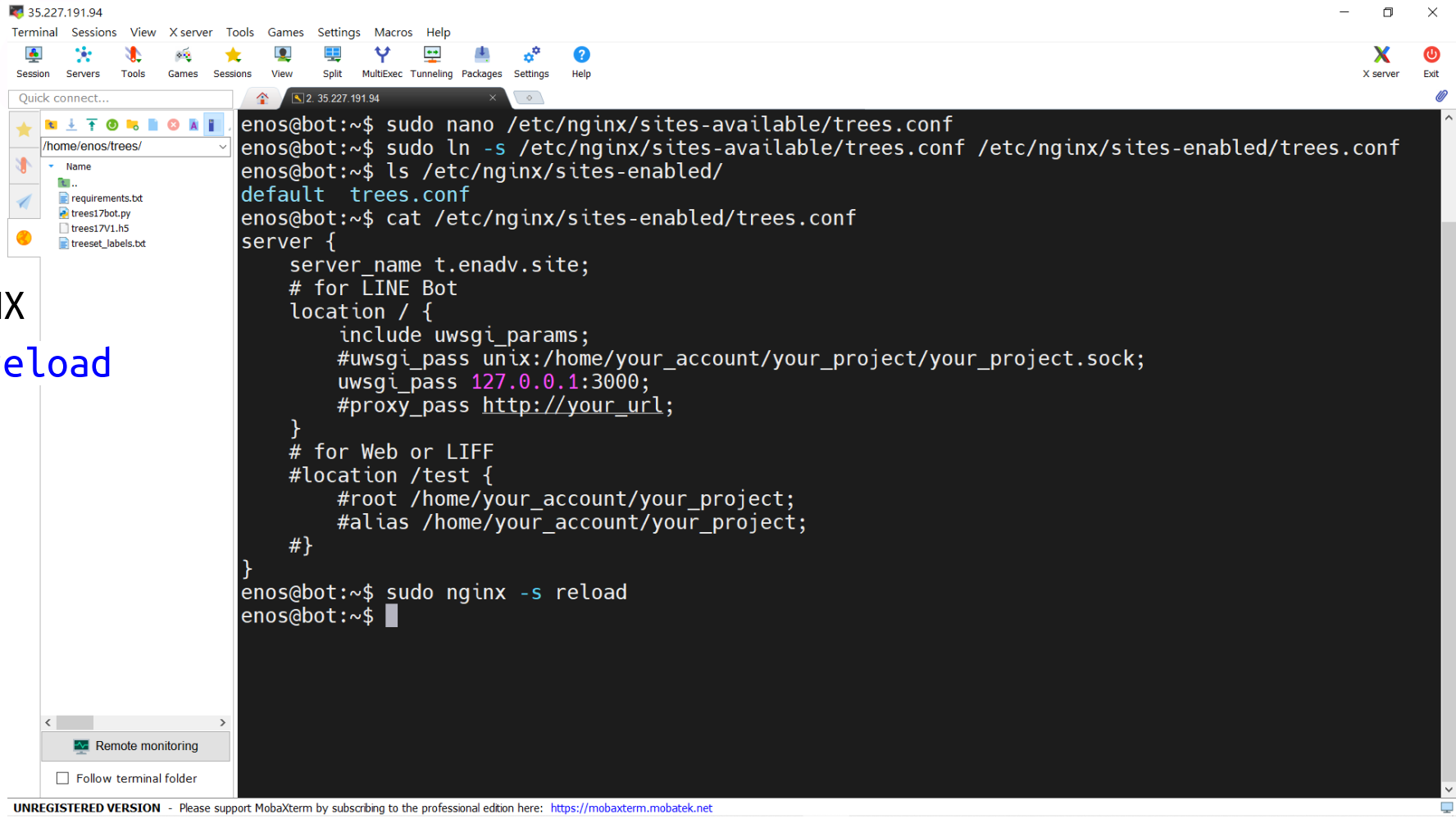


網站與憑證

1. 架設 NGINX

d. 載入新組態

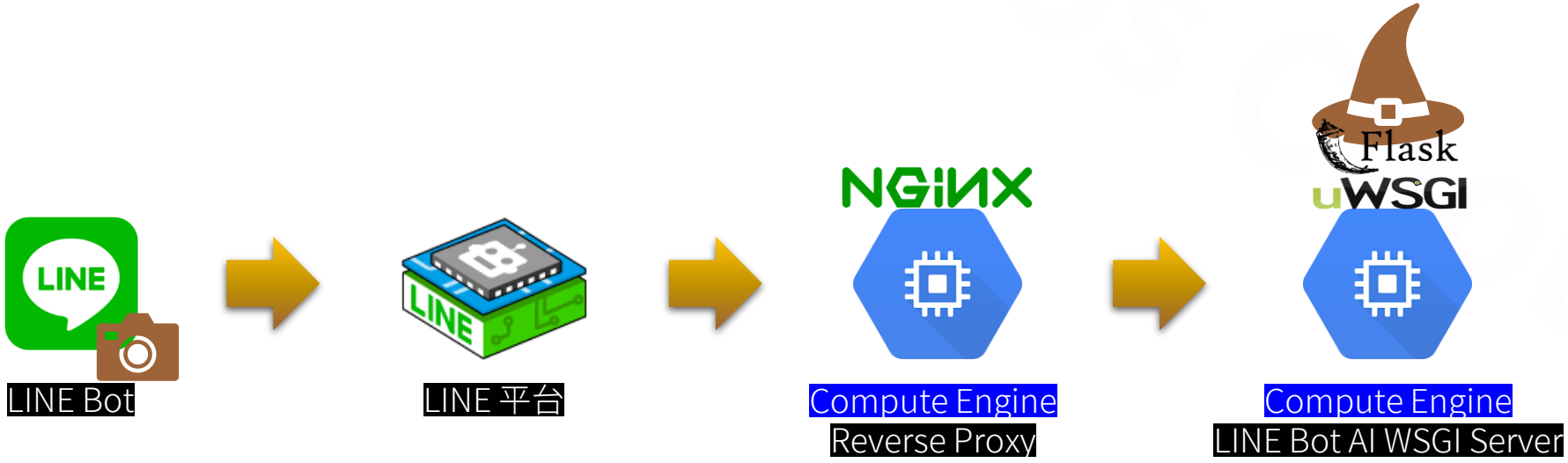
```
# 重新啟動 NGINX
sudo nginx -s reload
```



網站與憑證

免費憑證機器人 Certbot 特色

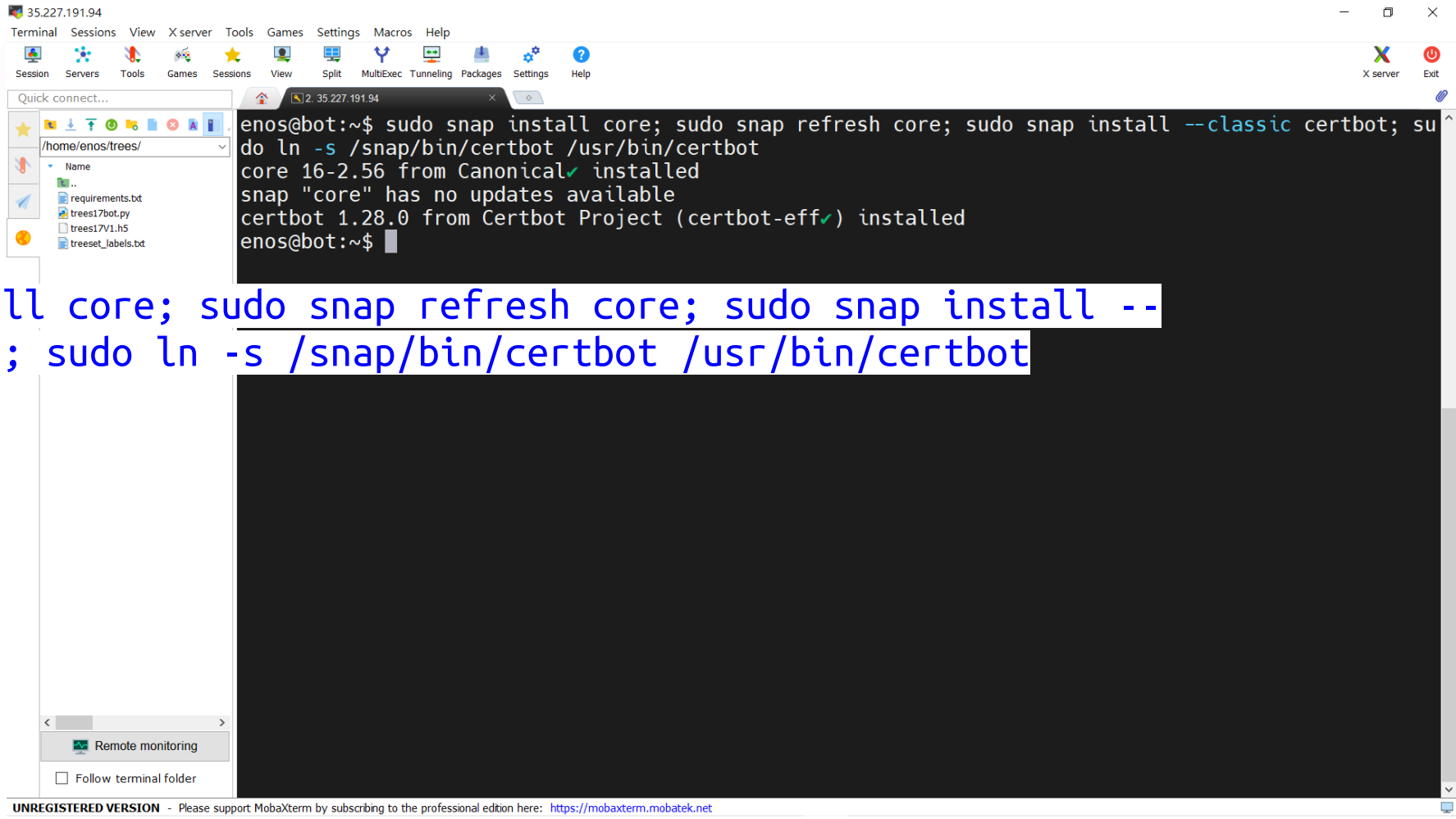
- https://certbot.eff.org/
- 採用 Let's Encrypt 免費憑證
- 自動安裝且自動更新憑證



網站與憑證

2. 架設 Certbot

a. 安裝 Certbot



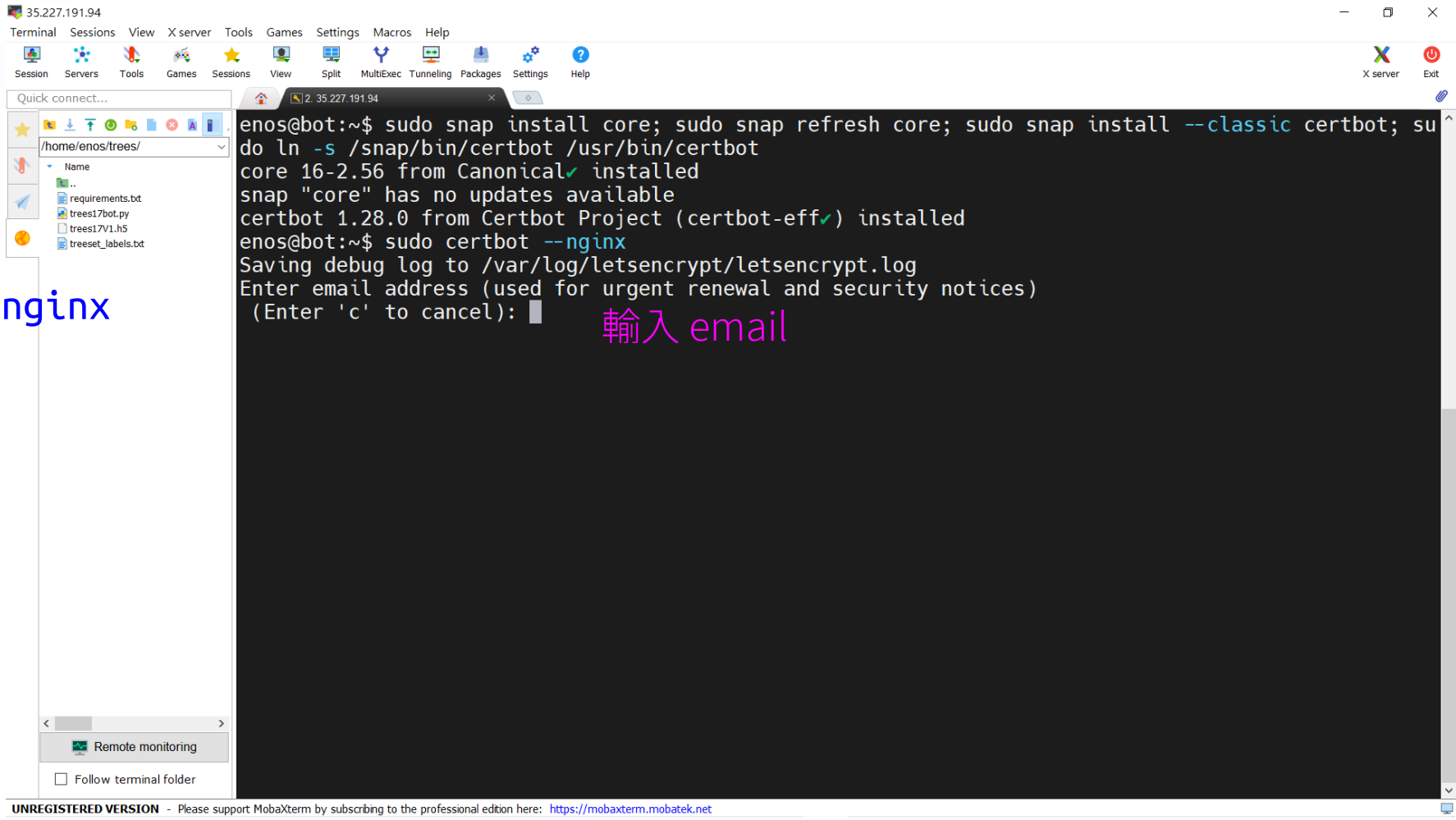

```
sudo snap install core; sudo snap refresh core; sudo snap install --
classic certbot; sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

網站與憑證

2. 架設 Certbot

b. 申請憑證

```
sudo certbot --nginx
```

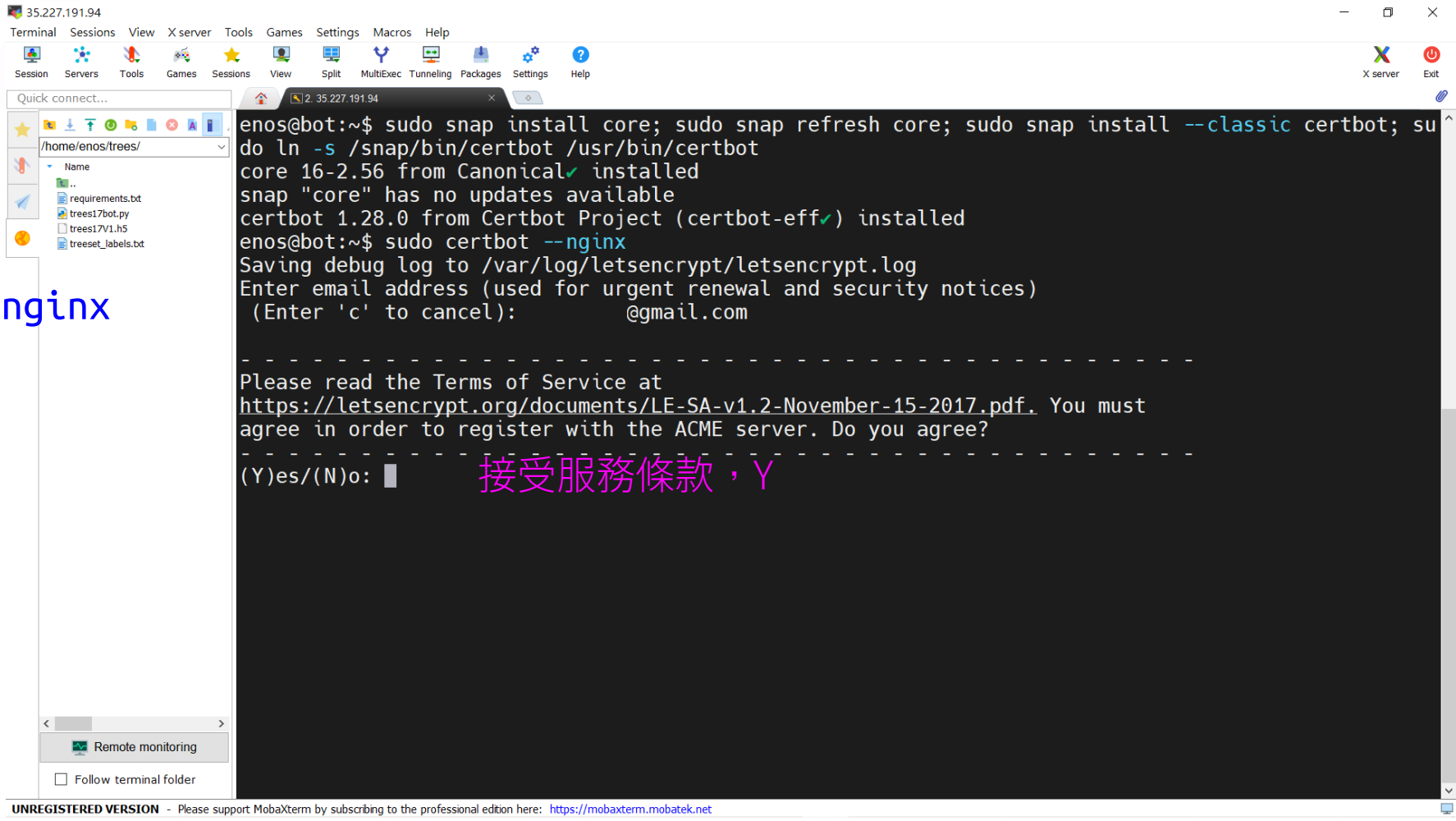


網站與憑證

2. 架設 Certbot

b. 申請憑證

```
sudo certbot --nginx
```

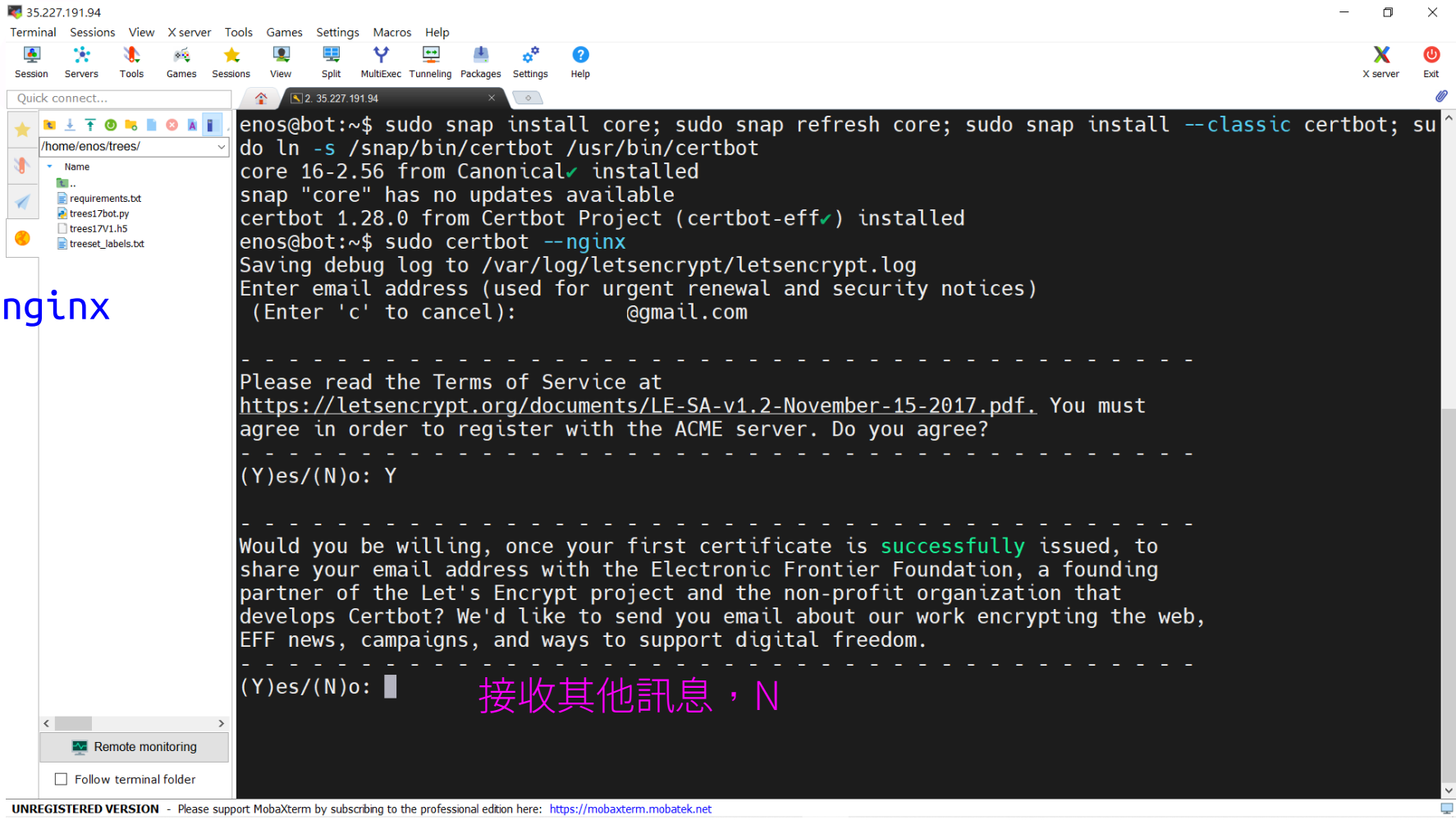


網站與憑證

2. 架設 Certbot

b. 申請憑證

```
sudo certbot --nginx
```

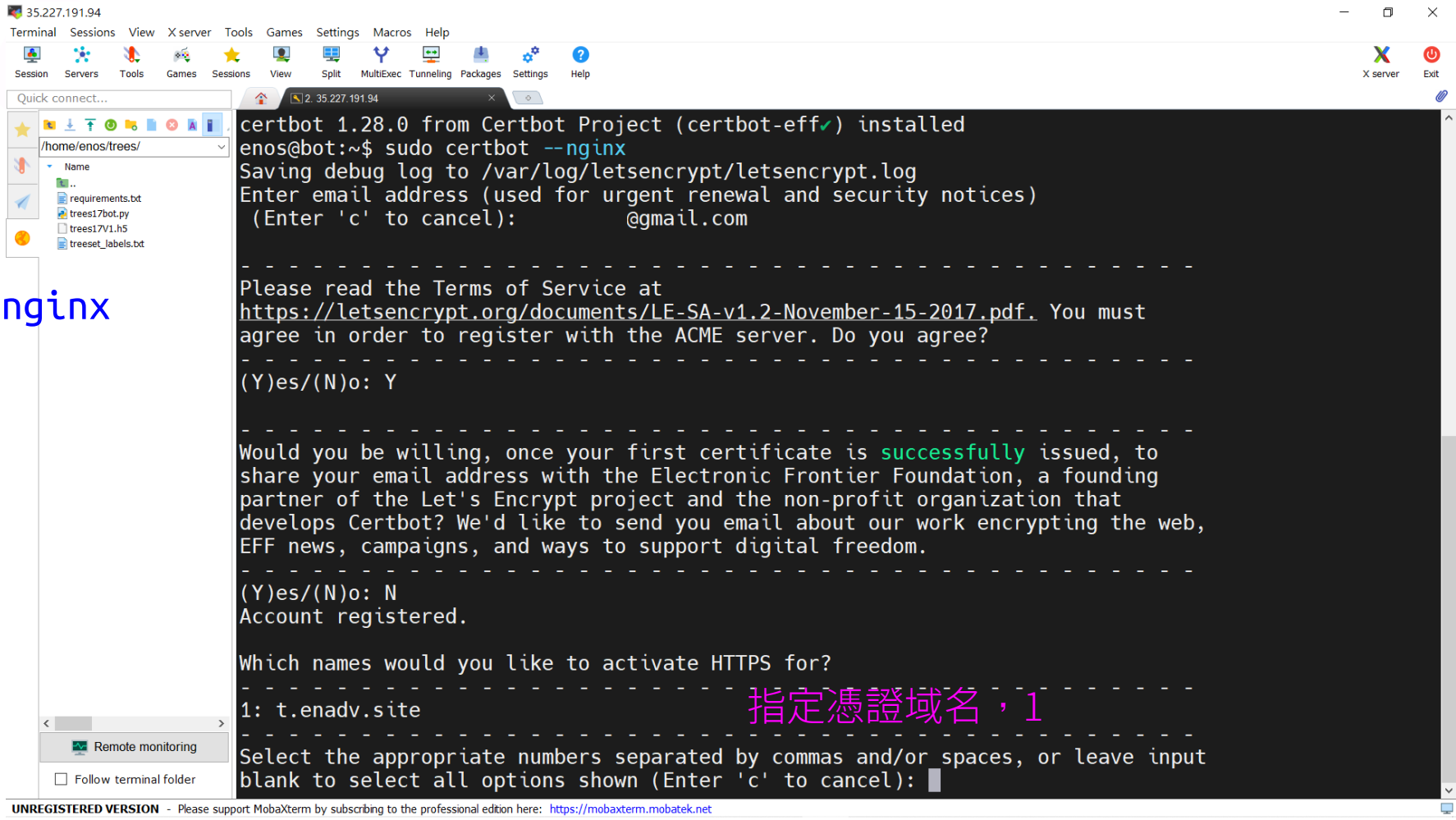


網站與憑證

2. 架設 Certbot

b. 申請憑證

```
sudo certbot --nginx
```

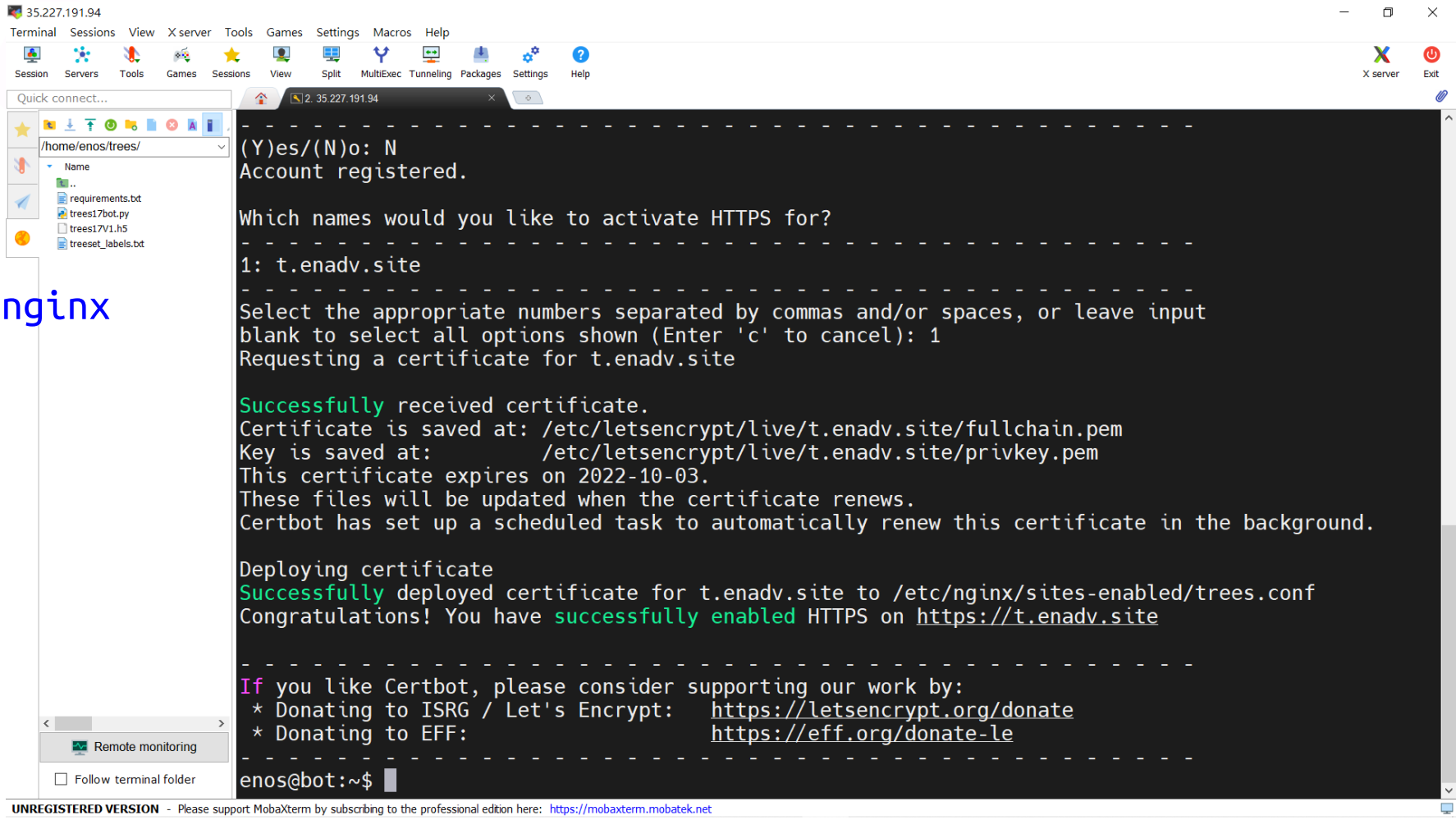


網站與憑證

2. 架設 Certbot

b. 申請憑證

```
sudo certbot --nginx
```

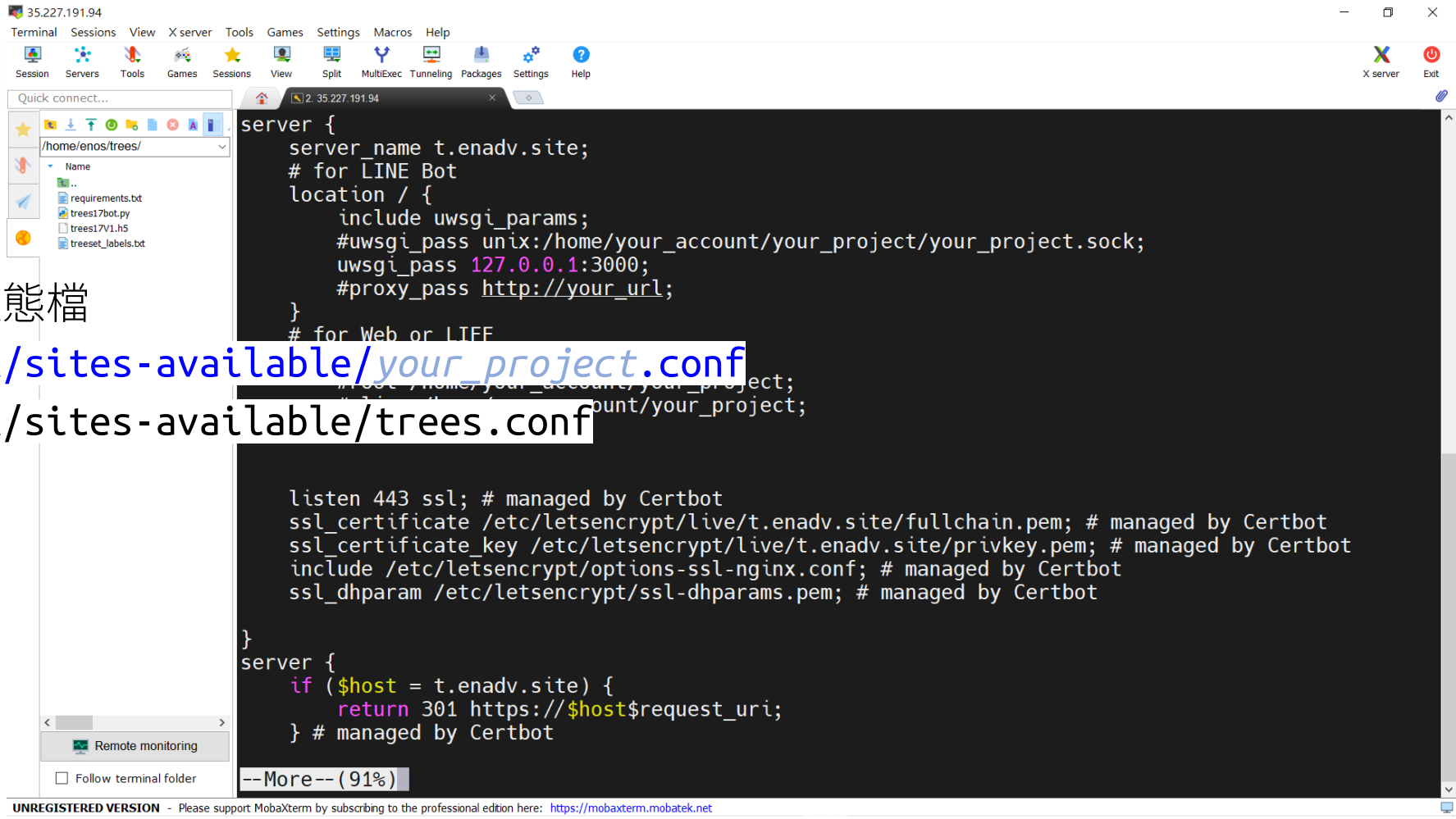


網站與憑證

2. 架設 Certbot

b. 申請憑證

```
# 觀察 NGINX 組態檔
more /etc/nginx/sites-available/your_project.conf
more /etc/nginx/sites-available/trees.conf
```



網站與憑證

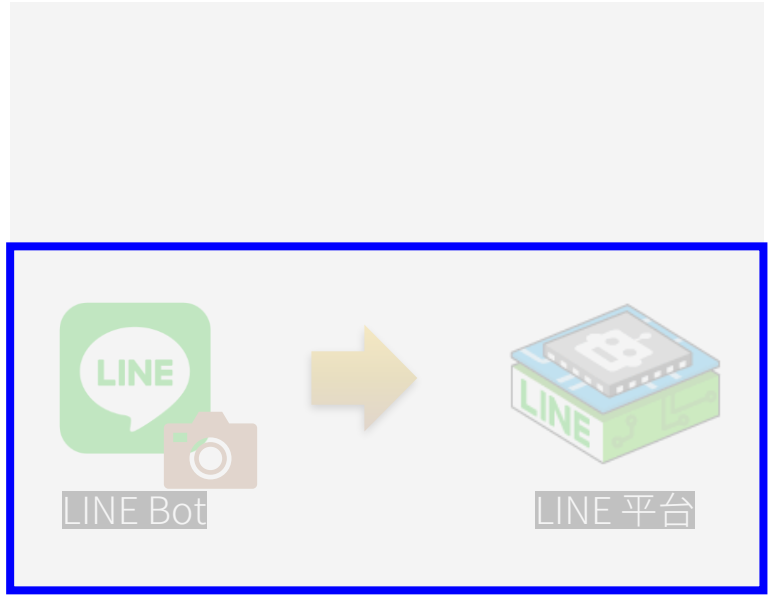
2. 架設 Certbot

c. (移除憑證)

① `sudo certbot delete --cert-name your_domain`

② 手動移除 `/etc/nginx/sites-available/your_project.conf` 中 Certbot 增加的設定

架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging

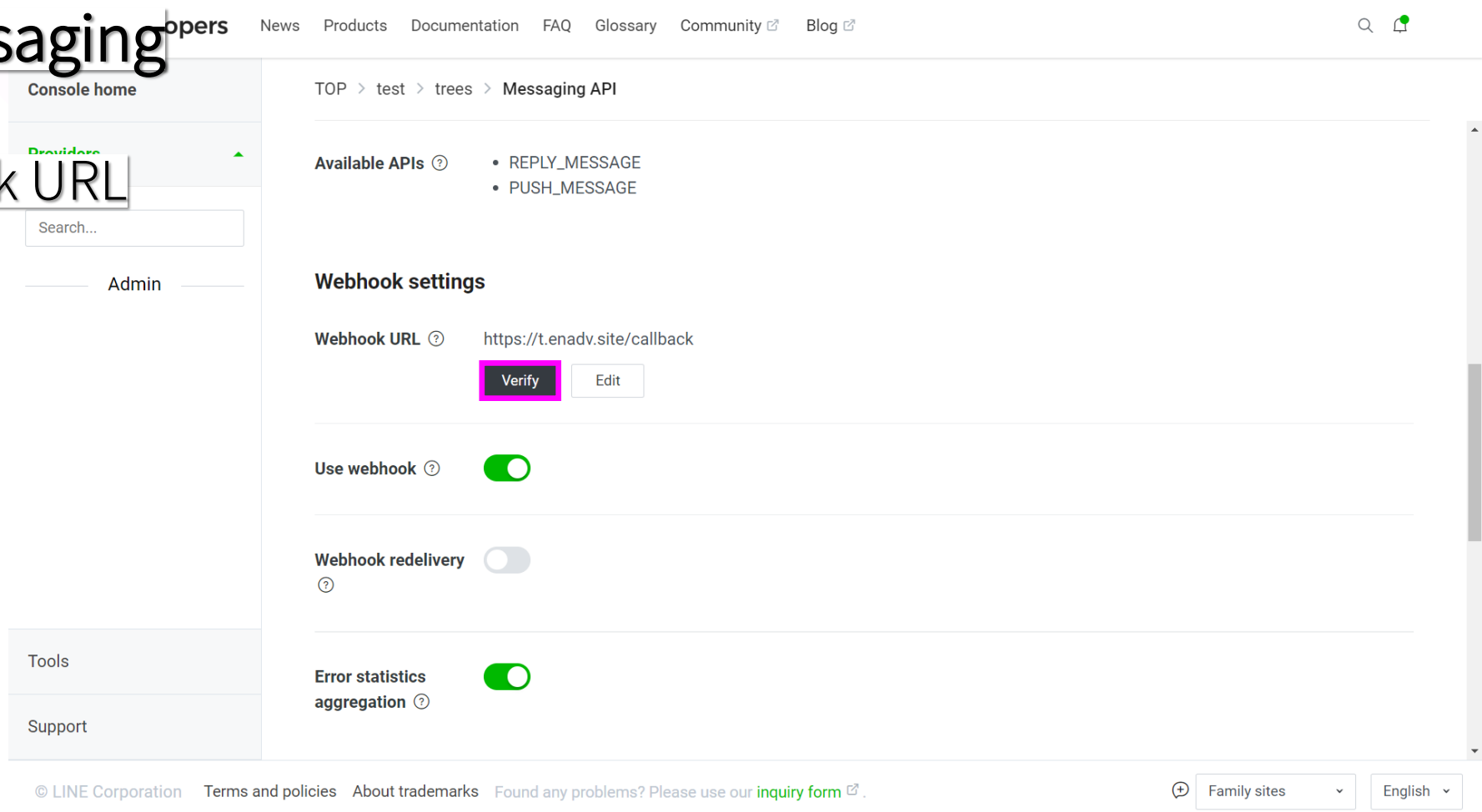


Compute Engine
Reverse Proxy

Compute Engine
LINE Bot AI WSGI Server

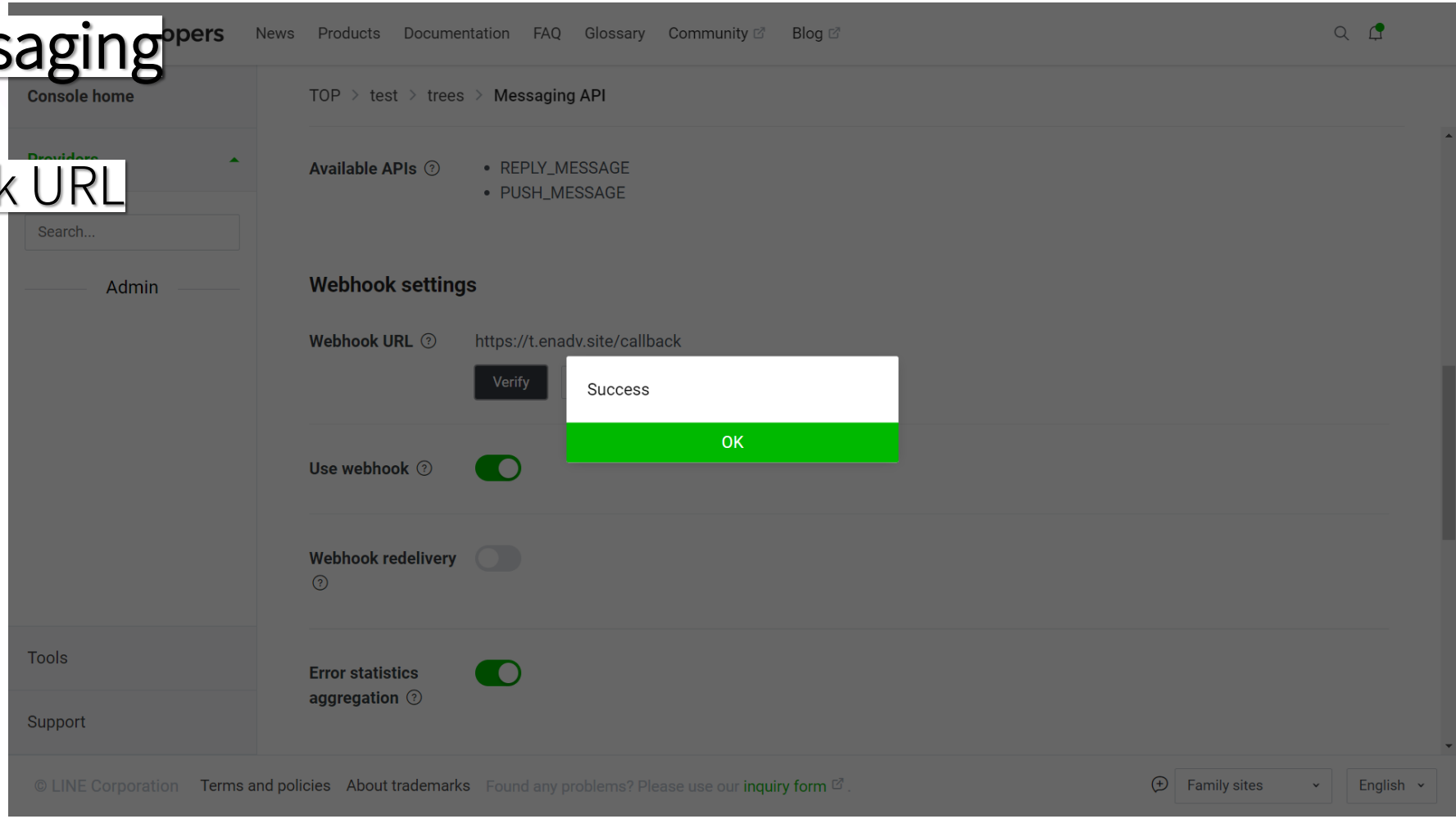
設定 LINE Messaging

1. 調整 LINE Messaging
   a. 調整 Webhook URL



設定 LINE Messaging

1. 調整 LINE Messaging
    a. 調整 Webhook URL

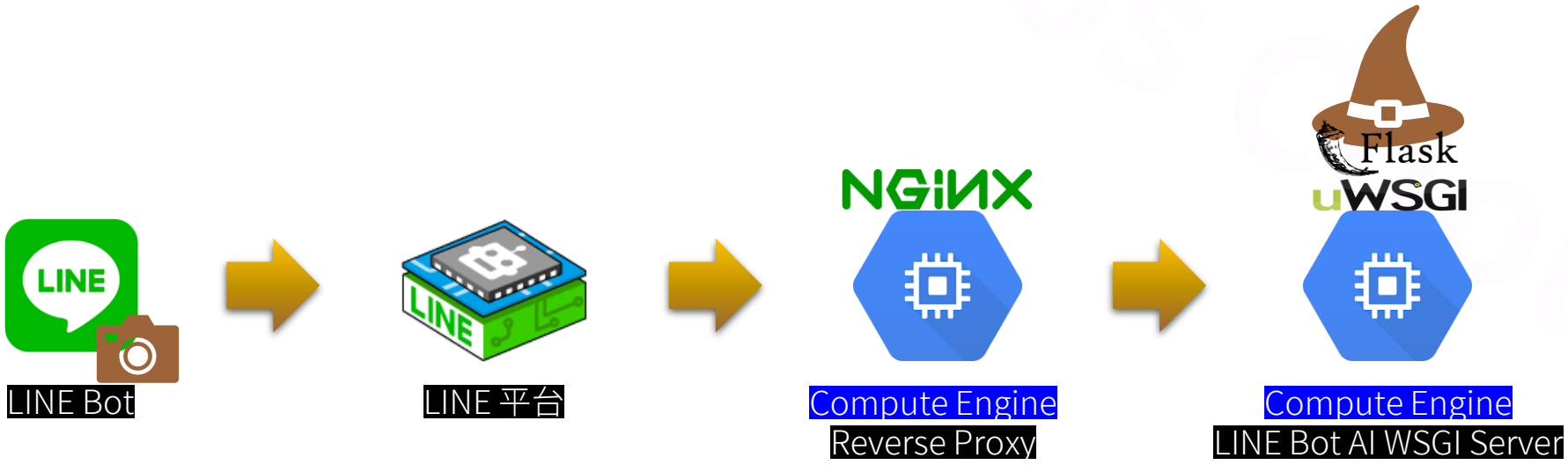


設定 LINE Messaging

1. 調整 LINE Messaging

   b. LINE 測試

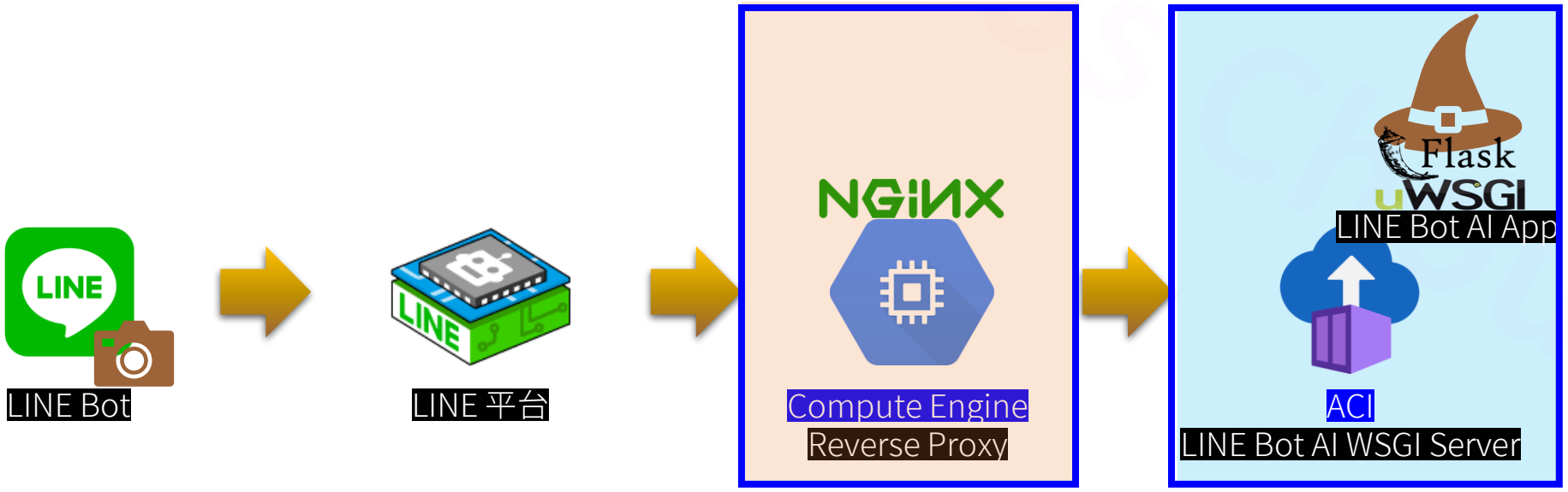
架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging



LINE Bot → LINE 平台 → Compute Engine Reverse Proxy → Compute Engine LINE Bot AI WSGI Server

Solution 2 - Mixed 部署

1. 長期可用且廉價的硬體環境 ... Google Compute Engine + Azure ACI
2. Flask as Web Server 的替代方案 ... NGINX + Domain + uWSGI
3. 長期可用且廉價的 SSL 網域方案 ... Let's Encrypt + Certbot



任務

1. 上傳程式至 Cloud Shell
2. 建立 Container 並推送至容器登錄
3. 部署 Container 至容器執行個體
4. 調整 GCE NGINX 轉導設定

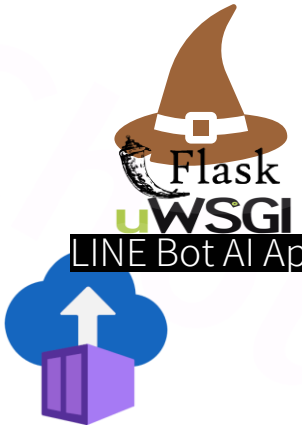
流程

架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging

LINE Bot

LINE 平台

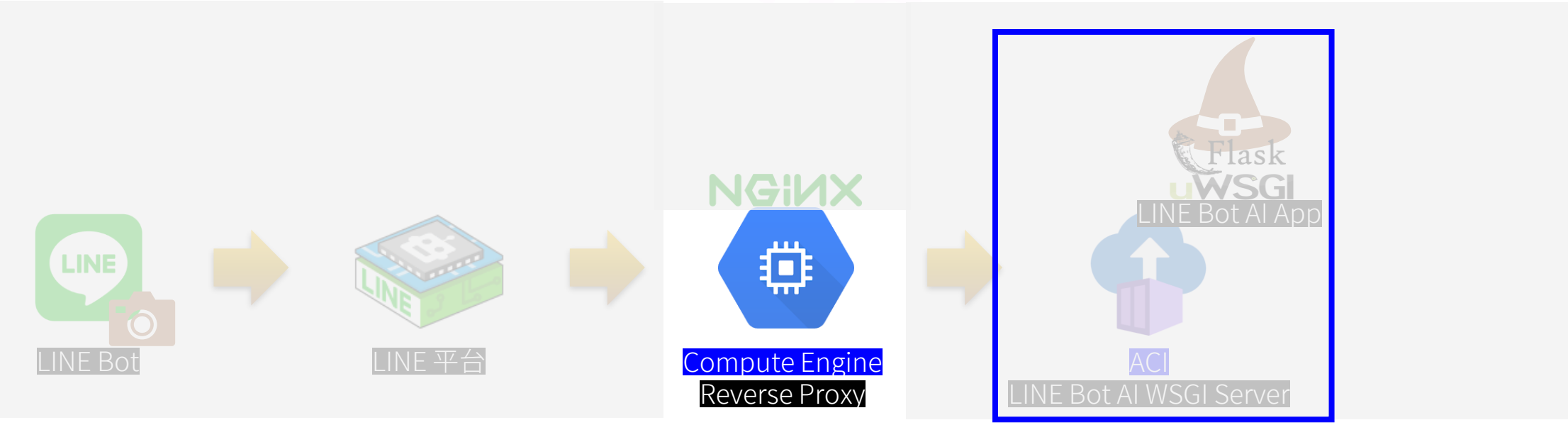
Compute Engine
Reverse Proxy



LINE Bot AI App

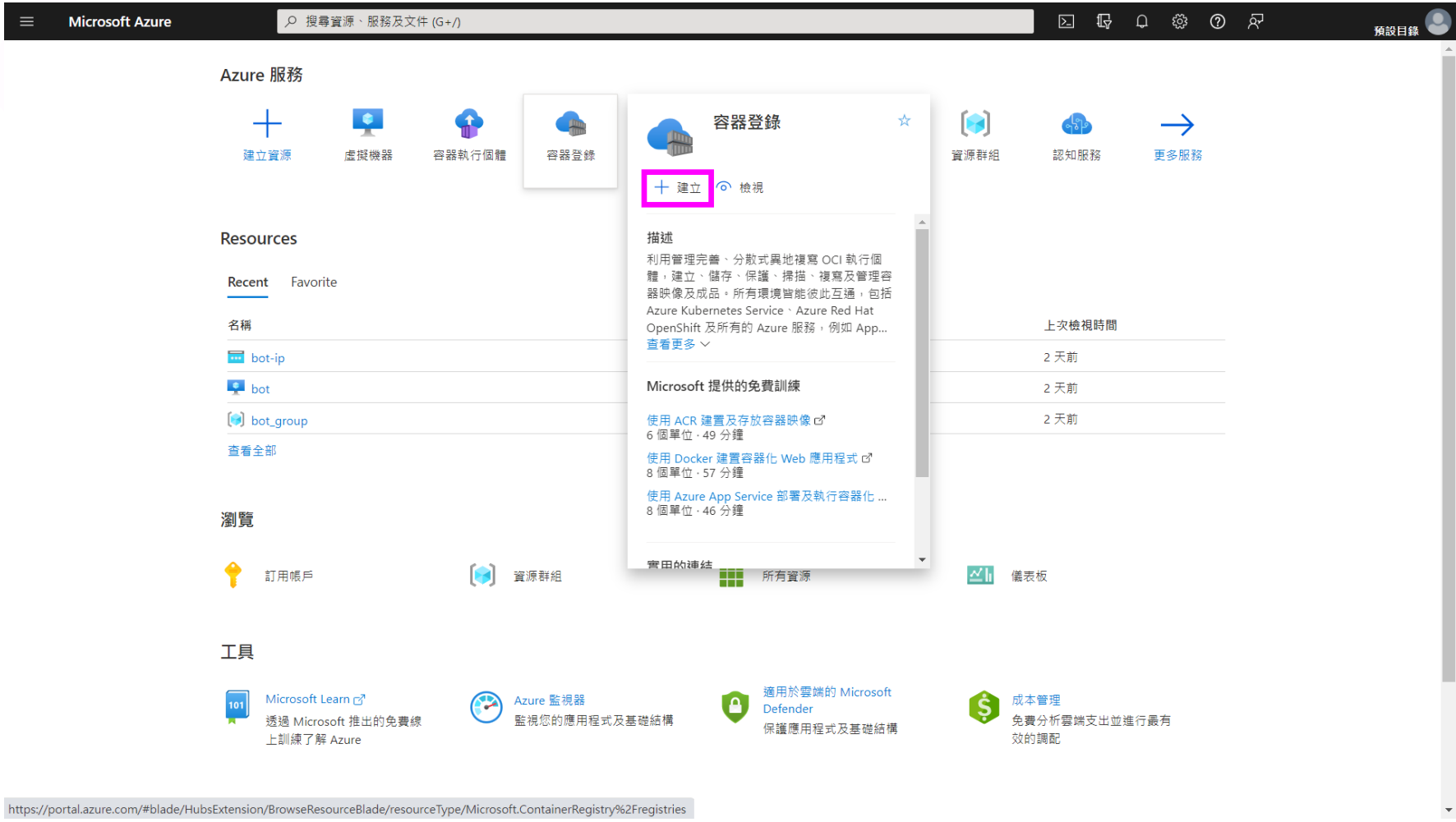
ACI
LINE Bot AI WSGI Server

架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging



LINE Bot & WSGI

1. 啟動容器登入



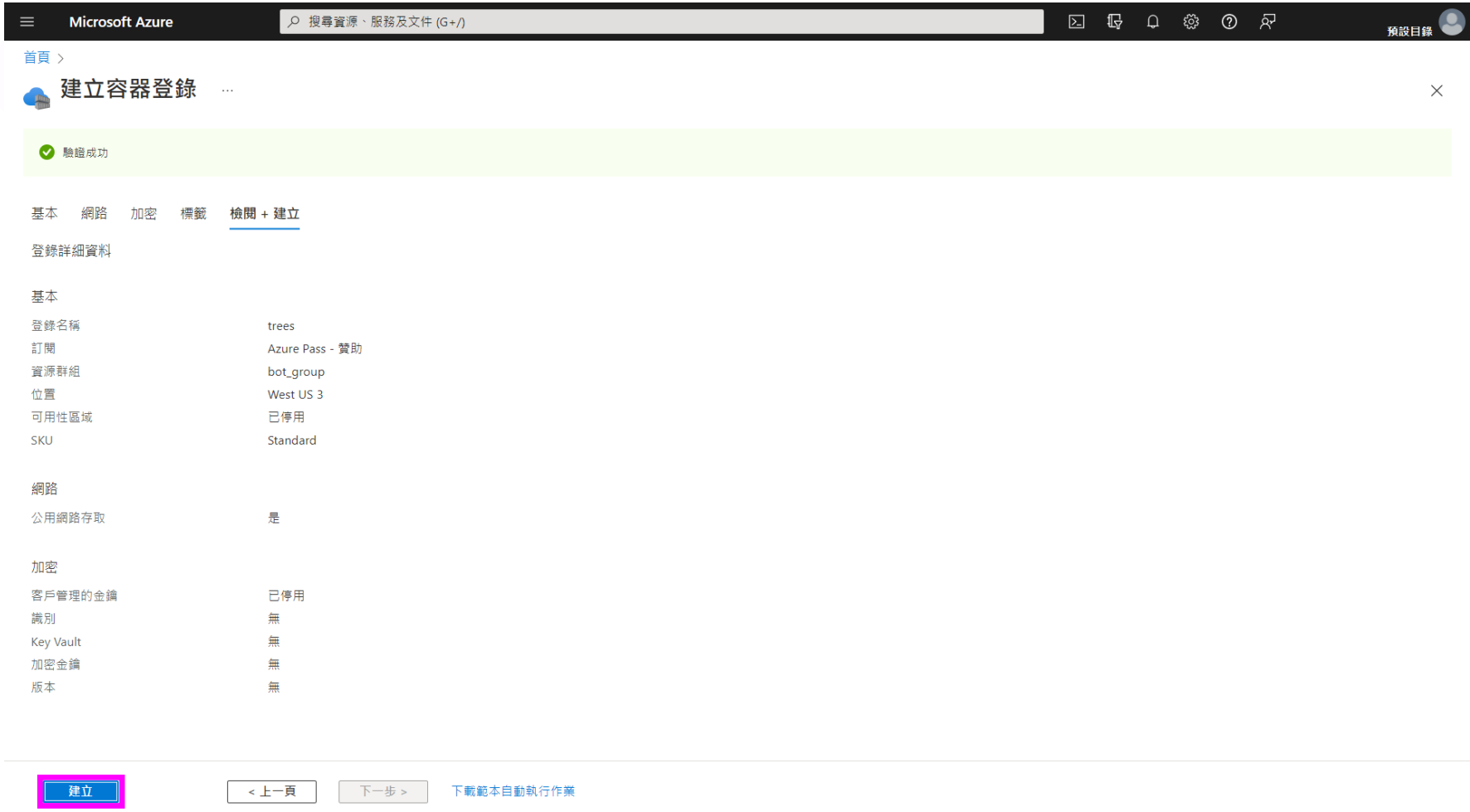
LINE Bot & WSGI

1. 啟動容器登入


```
.azurecr.io    # global unique
```

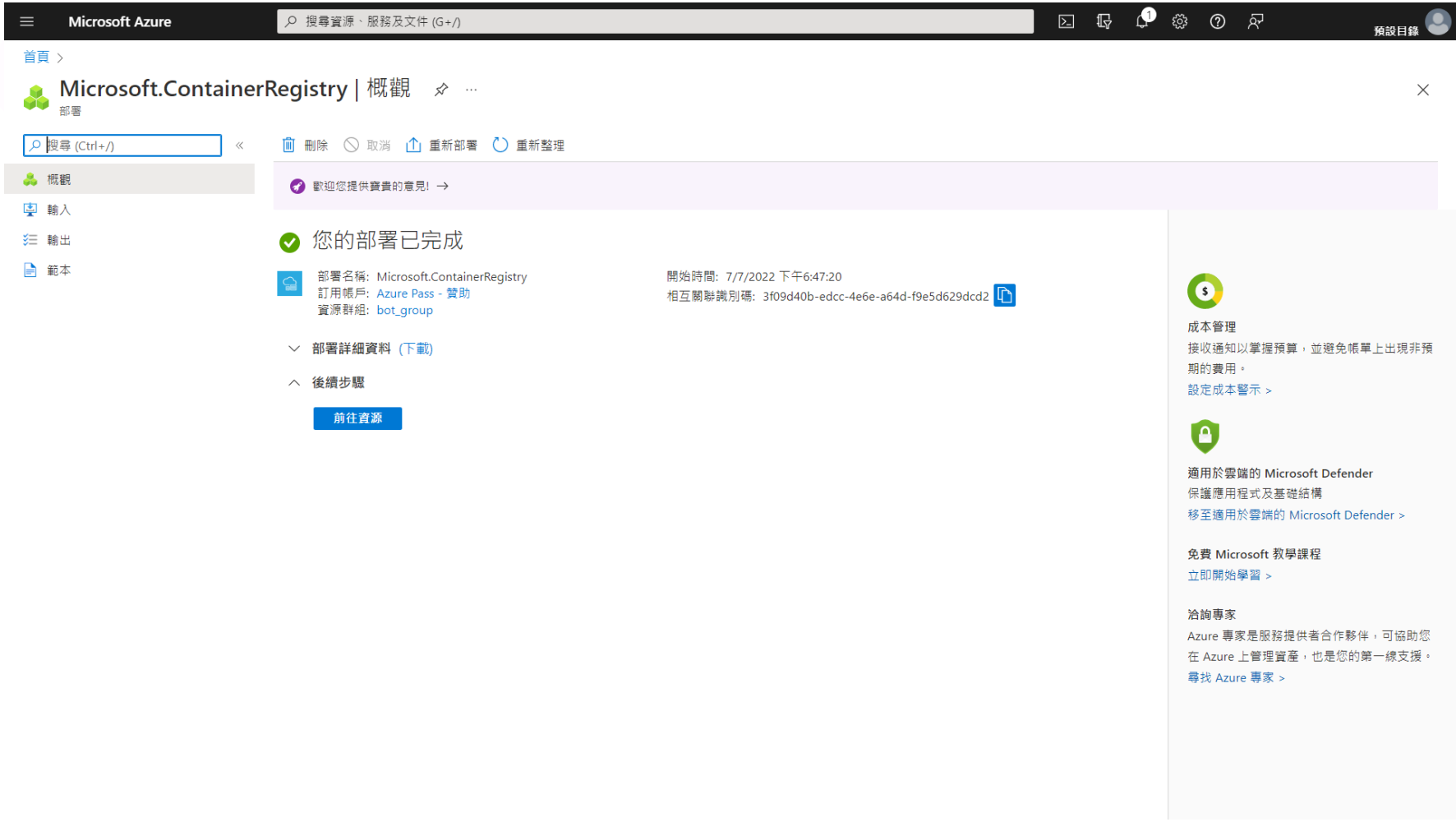
LINE Bot & WSGI

1. 啟動容器登入



LINE Bot & WSGI

1. 啟動容器登入



LINE Bot & WSGI

2. 準備程式碼

a. 下載範例程式並調整

① `trees17bot.py  # secret, token, model, labels`

② `trees17V1.h5`

③ `treeset_labels.txt`

④ `other tree samples`

LINE Bot & WSGI

2. 準備程式碼

b. 製作 requirements.txt

```
line-bot-sdk
flask
pillow
tensorflow==2.4.4
uwsgi
```

LINE Bot & WSGI

2. 準備程式碼

c. 製作 Dockerfile

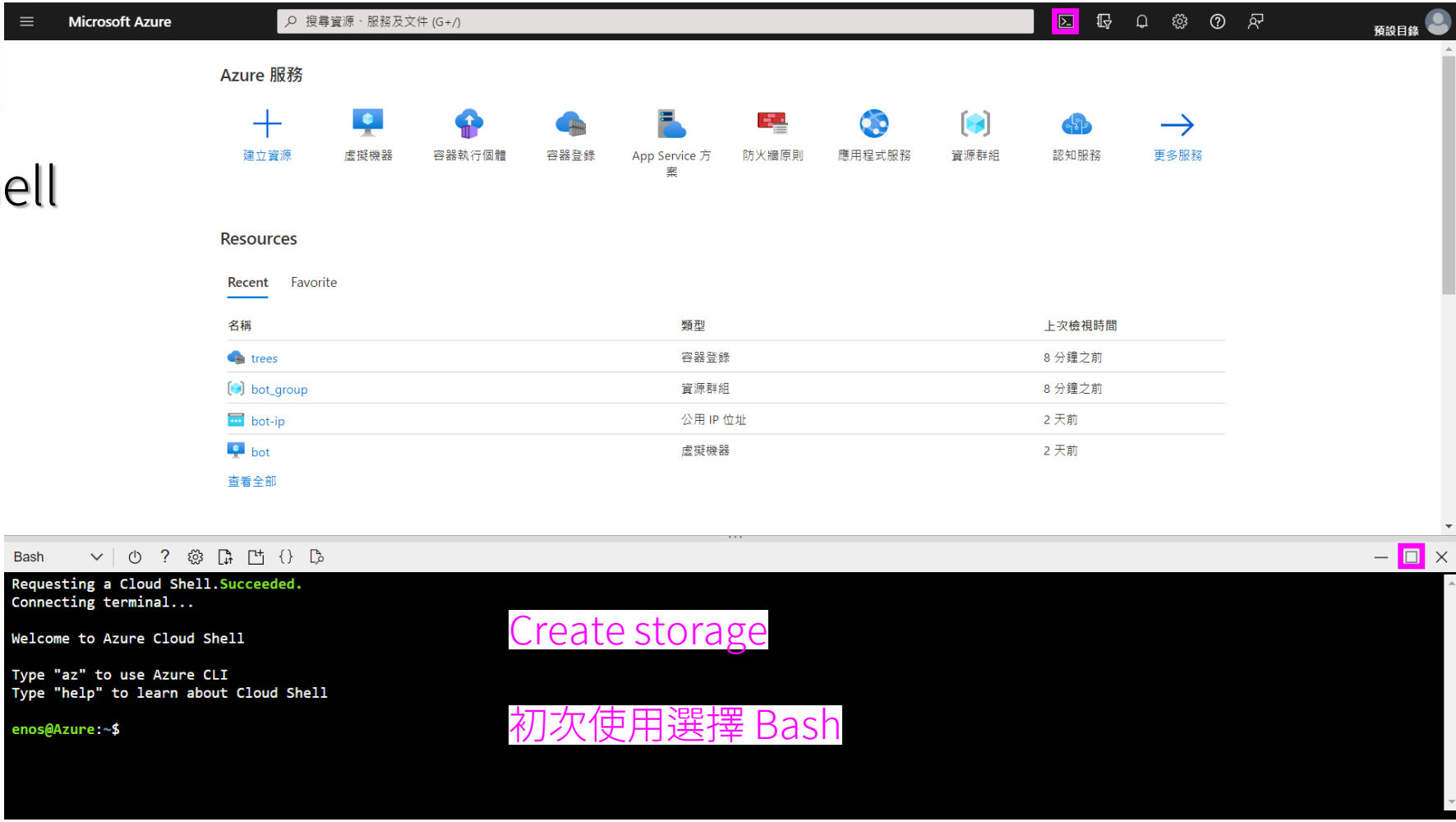
Note
調整 *your_module*

```
FROM ubuntu:18.04
WORKDIR /your_module
COPY * ./
ENV TZ=Asia/Taipei
RUN mkdir var && \
    ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && \
    echo $TZ > /etc/timezone && \
    apt-get update && \
    apt-get install -y python3-pip tzdata && \
    dpkg-reconfigure -f noninteractive tzdata && \
    python3 -m pip install --upgrade pip && \
    python3 -m pip install -r requirements.txt
CMD uwsgi -w your_module:app --http-socket :$PORT
```

LINE Bot & WSGI

2. 準備程式碼

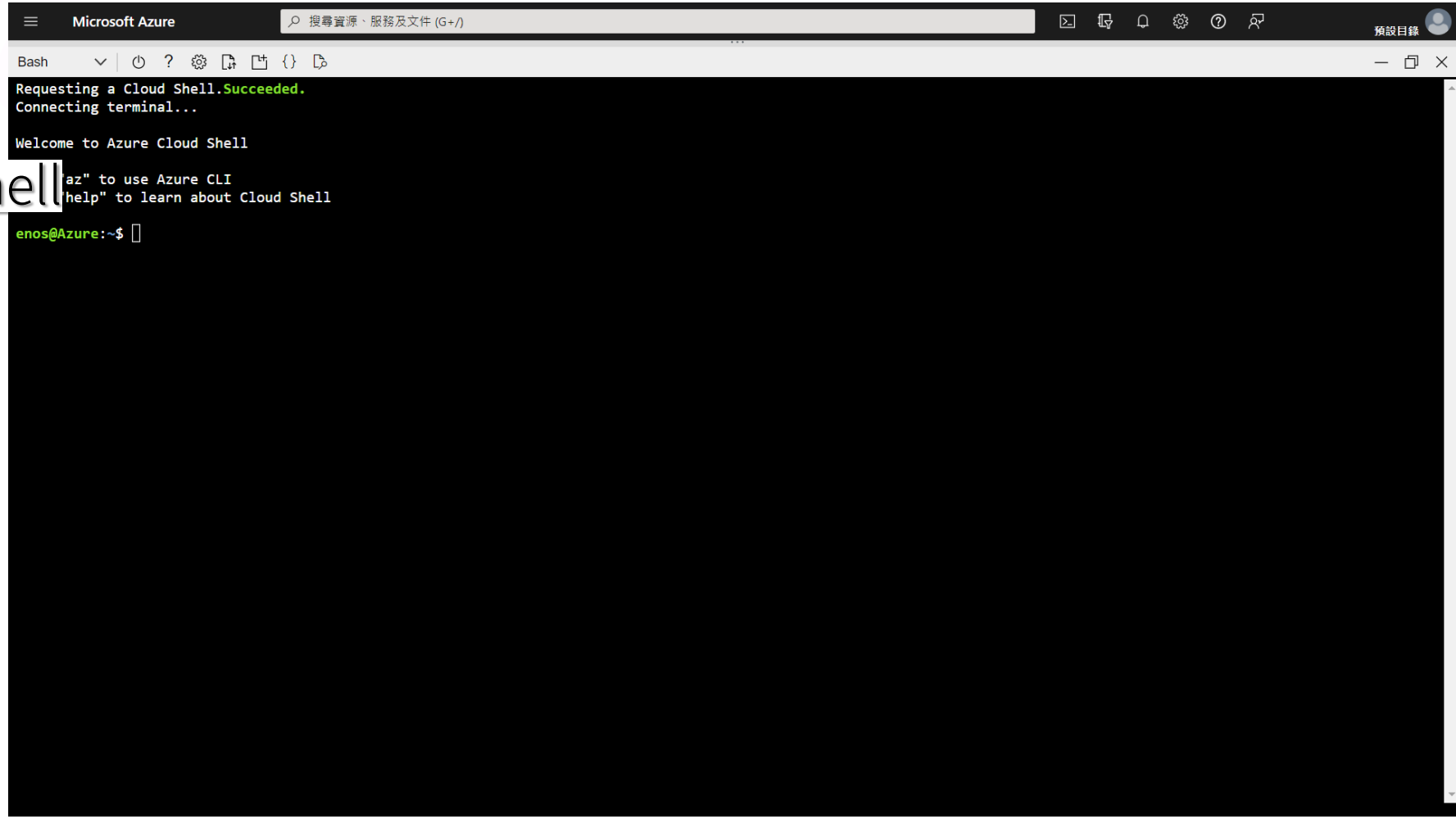
d. 啟動 Cloud Shell



LINE Bot & WSGI

2. 準備程式碼

d. 啟動 Cloud Shell



LINE Bot & WSGI

2. 準備程式碼

e. 上傳下列程式程式並置於專案目錄

① trees17bot.py

② trees17V1.py

③ treeset_labels.txt

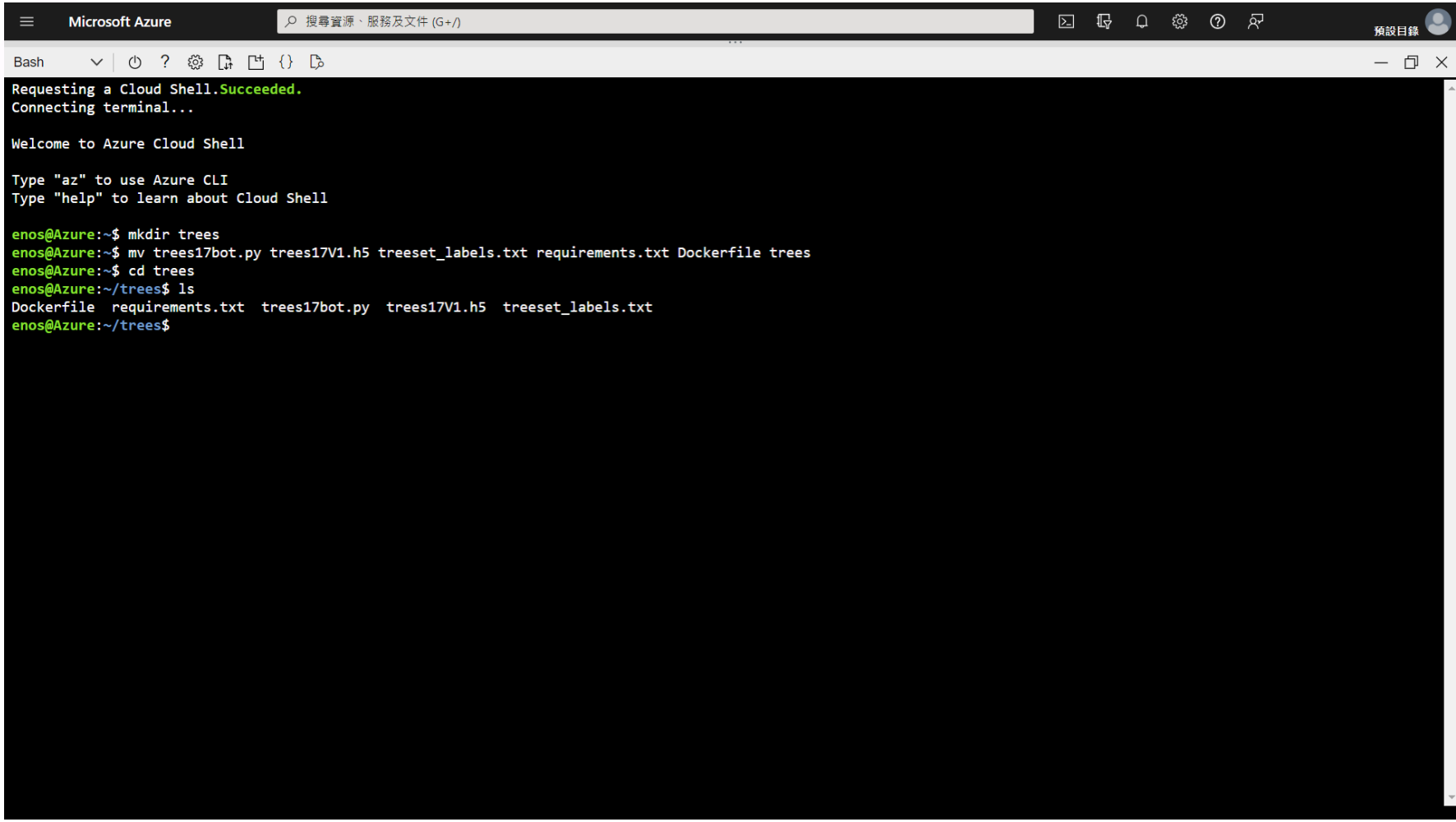
④ Dockerfile

⑤ requirements.txt

LINE Bot & WSGI

2. 準備程式碼

f. 移至專案目錄



LINE Bot & WSGI

3. 建立 Container

a. 製作並推送 Container 至容器登錄

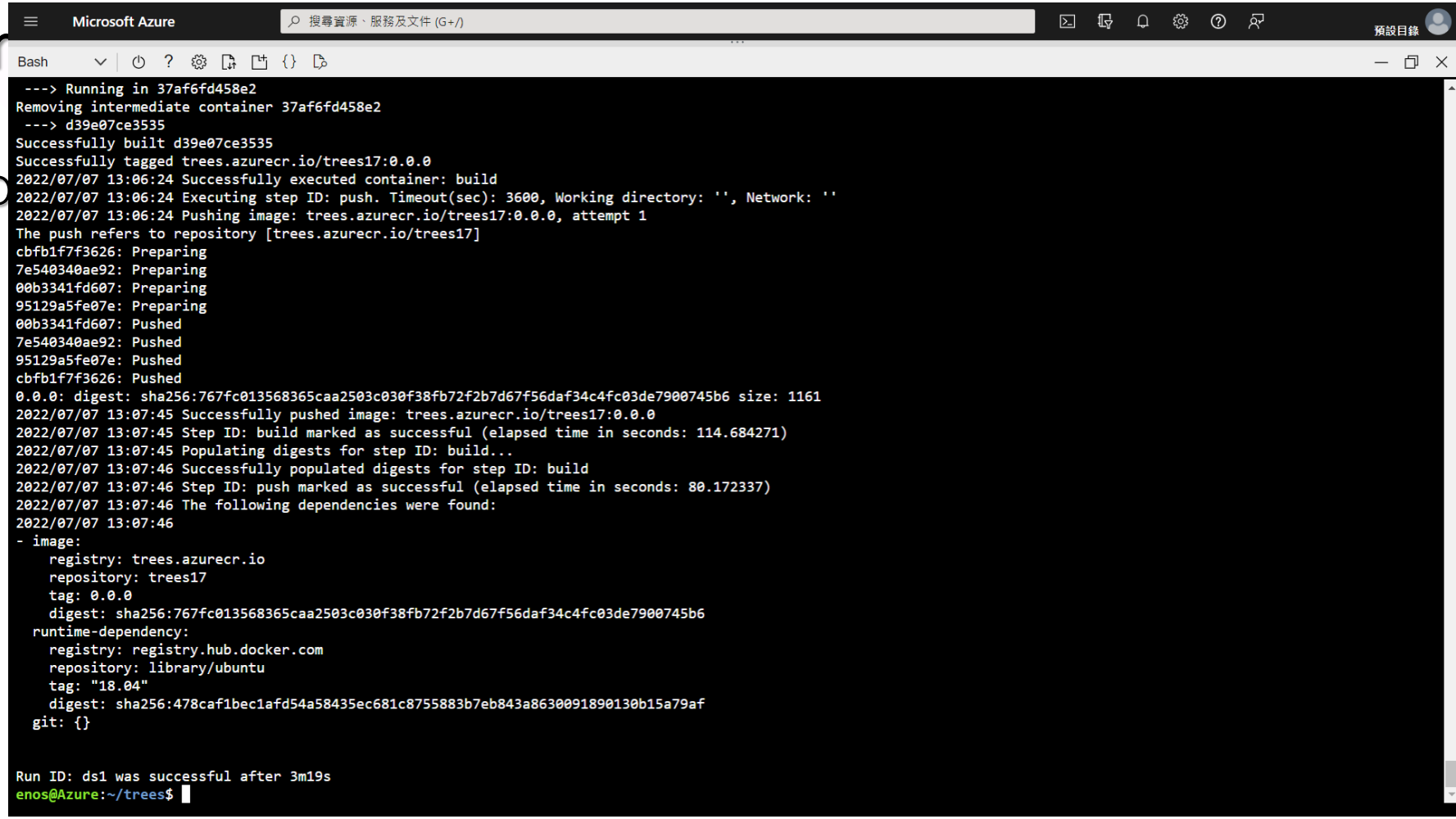
```
az acr build --registry your_registry --image your_container:your_tag .
az acr build --registry trees --image trees17:0.0.0 .
```

LINE Bot & WSGI

3. 建立 Container

a. 製作並推送 Co

```
az acr build --
az acr build --
```

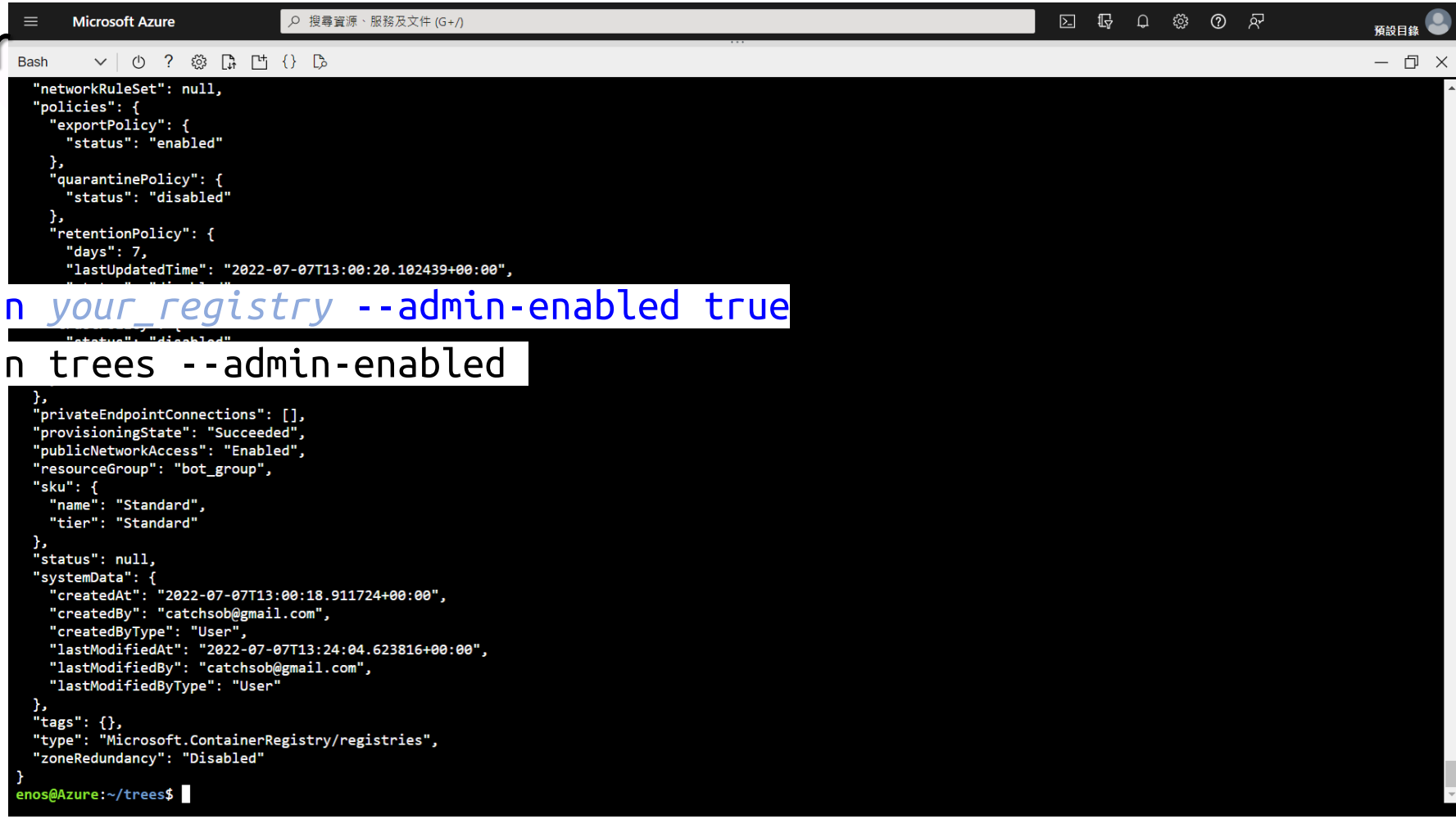


LINE Bot & WSGI

3. 建立 Container

b. 授權容器使用

```
az acr update -n your_registry --admin-enabled true
az acr update -n trees --admin-enabled
```



LINE Bot & WSGI

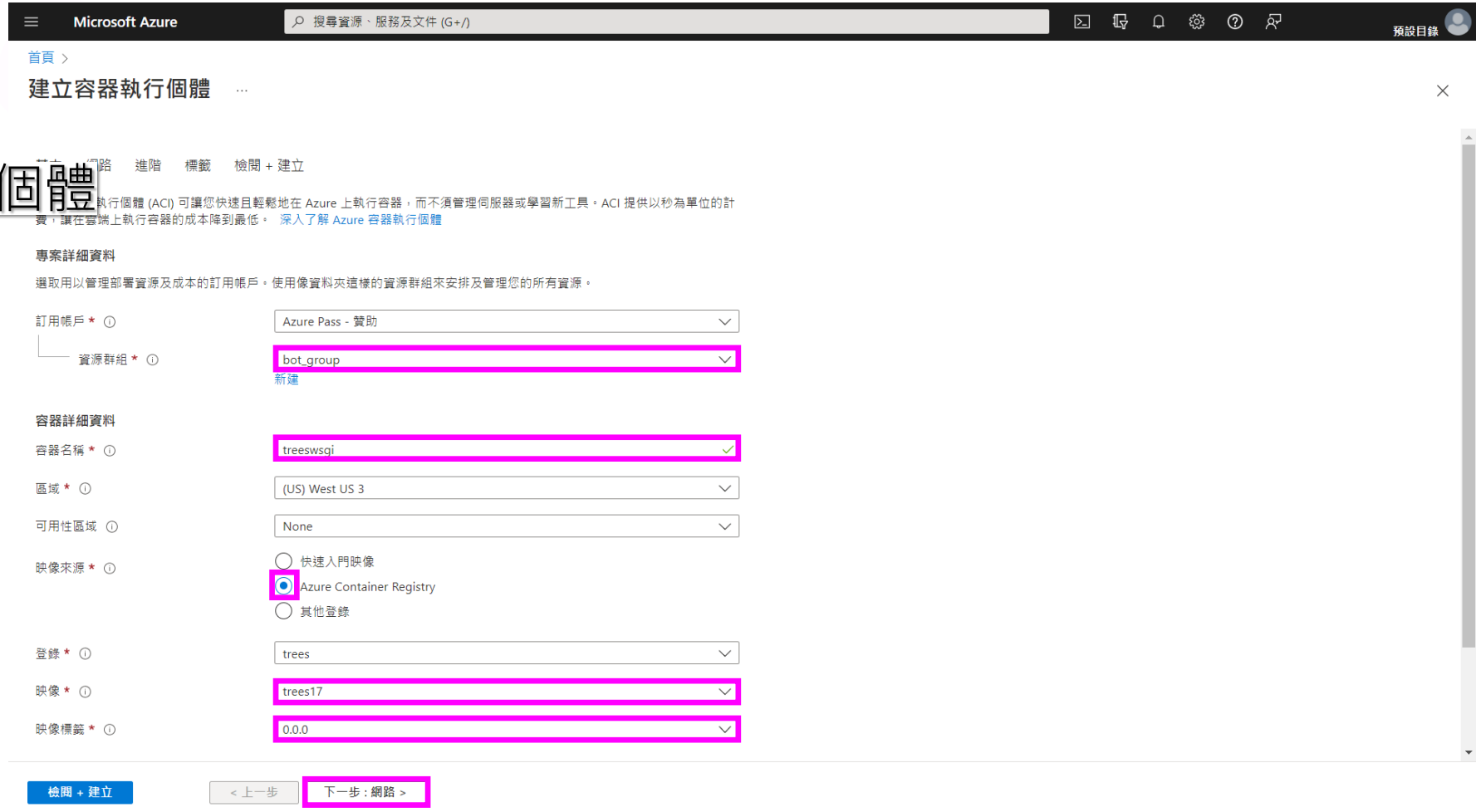
4. 部署 ACI

a. 建立容器執行個體

LINE Bot & WSGI

4. 部署 ACI

a. 建立容器執行個體



LINE Bot & WSGI

4. 部署 ACI

a. 建立容器執行個體

此 PORT 不須另外開通防火牆

LINE Bot & WSGI

4. 部署 ACI

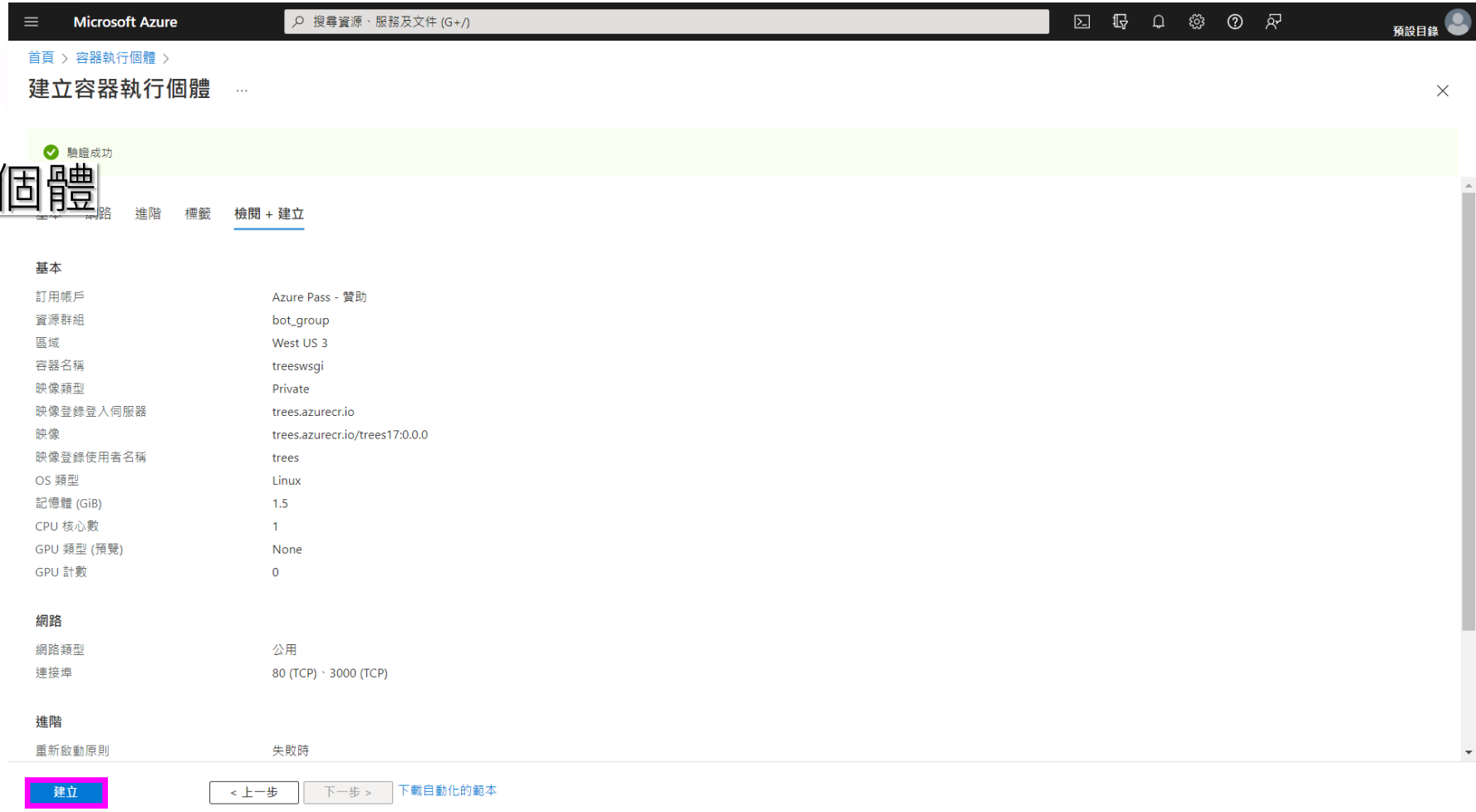
a. 建立容器執行個體

Container 使用的 PORT 以環境變數指定

LINE Bot & WSGI

4. 部署 ACI

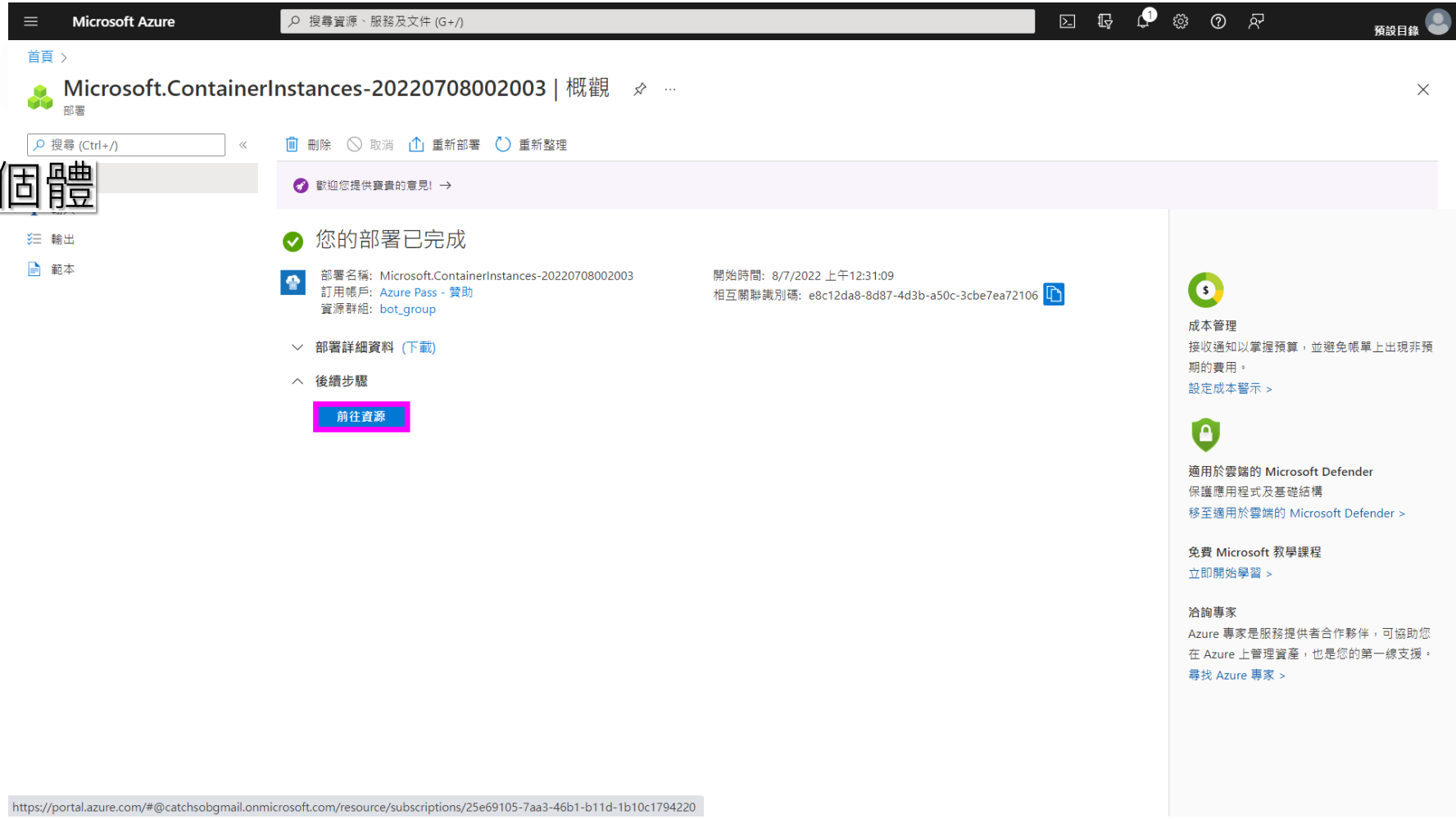
a. 建立容器執行個體



LINE Bot & WSGI

4. 部署 ACI

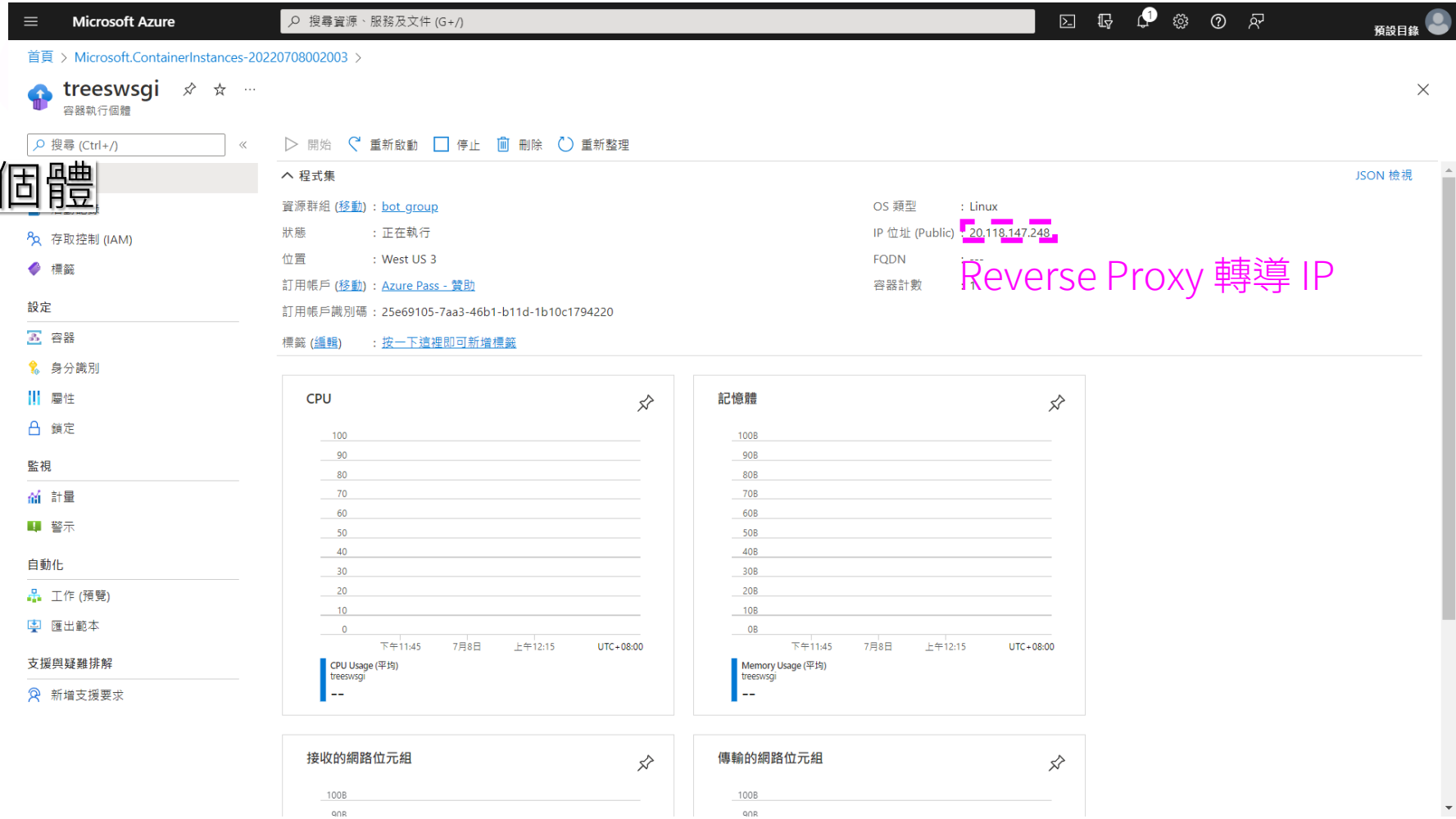
a. 建立容器執行個體



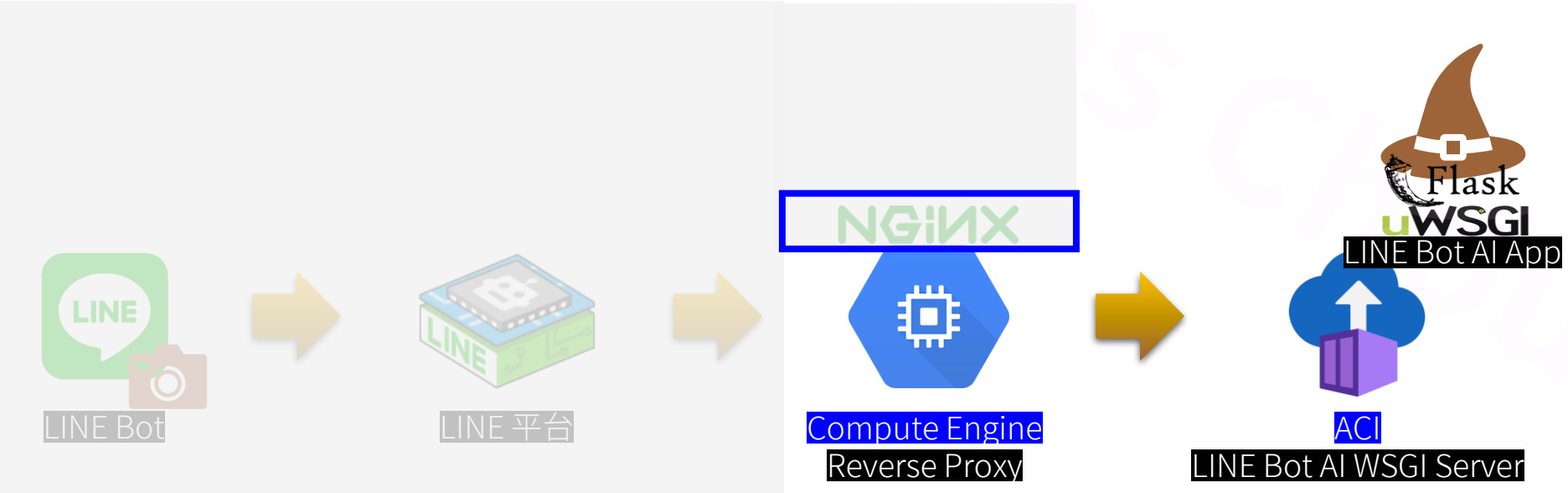
LINE Bot & WSGI

4. 部署 ACI

a. 建立容器執行個體

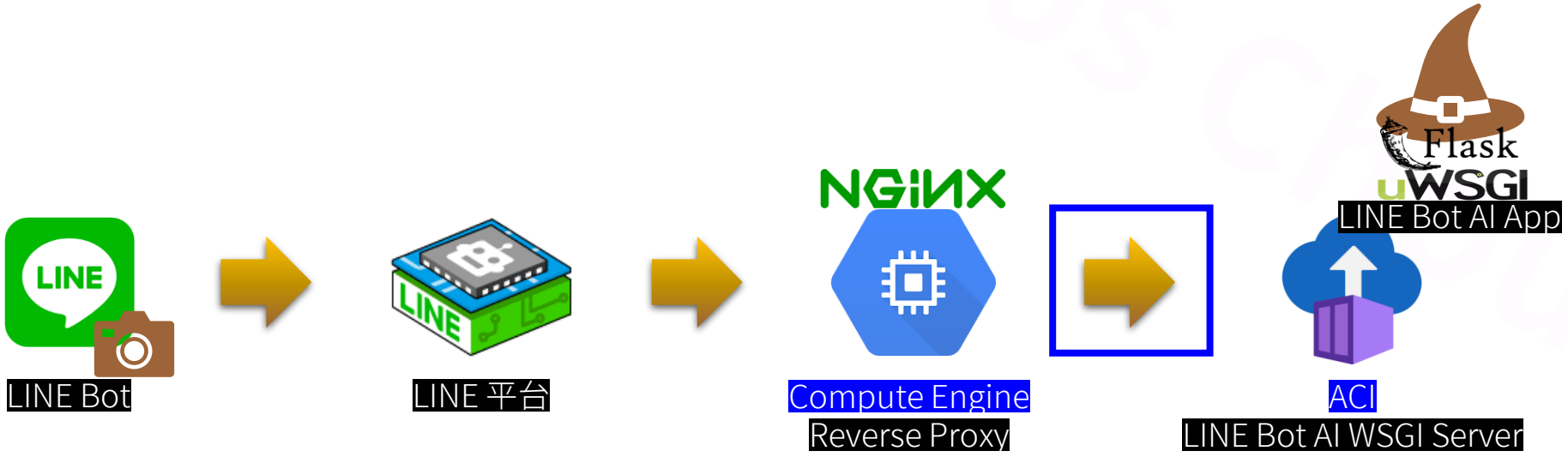


架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging

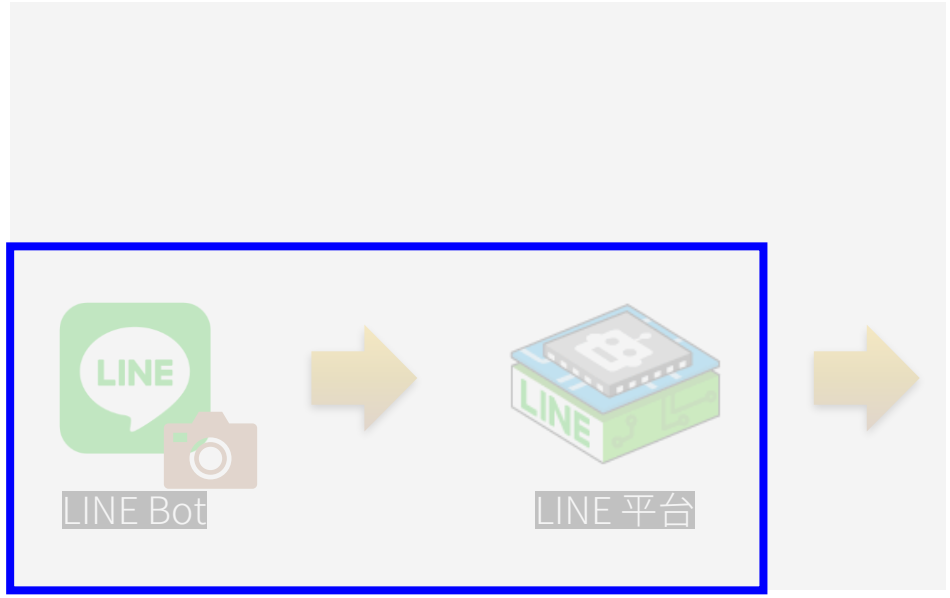


網站與憑證 - Your Turn

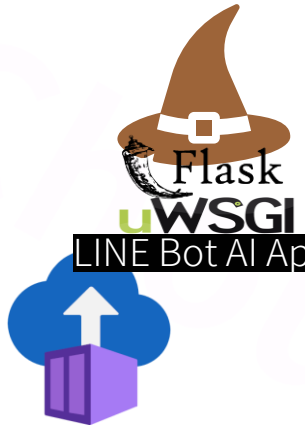
嘗試由 NGINX @VM 轉導 LINE Message 至 LINE Bot + WSGI @ACI



架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging



Compute Engine
Reverse Proxy



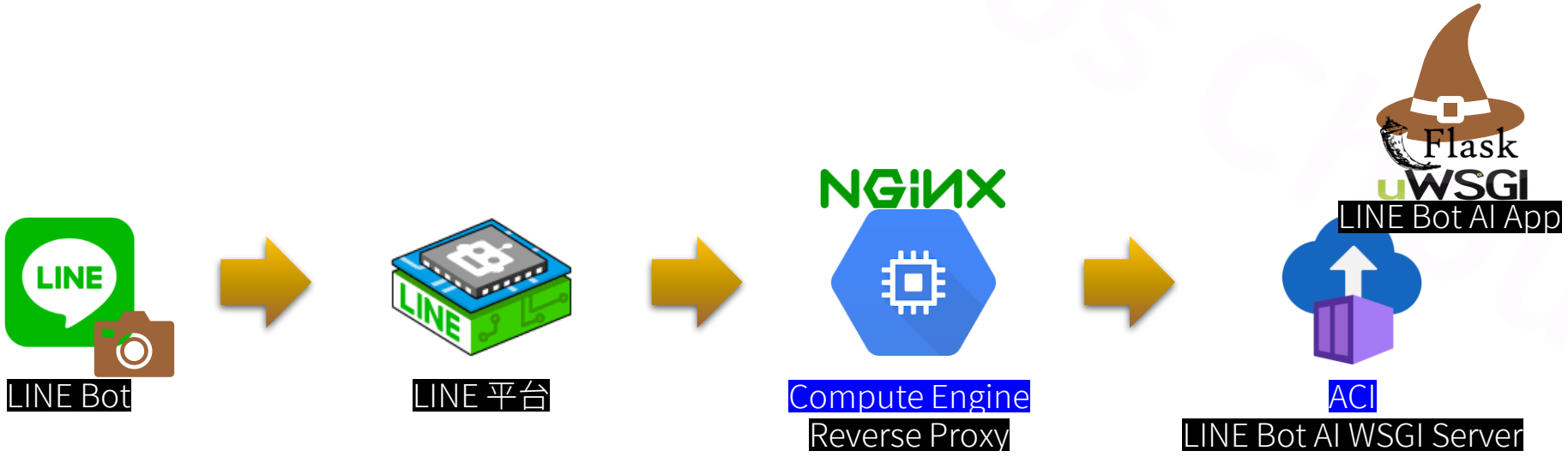
LINE Bot AI App

ACI
LINE Bot AI WSGI Server

設定 LINE Messaging

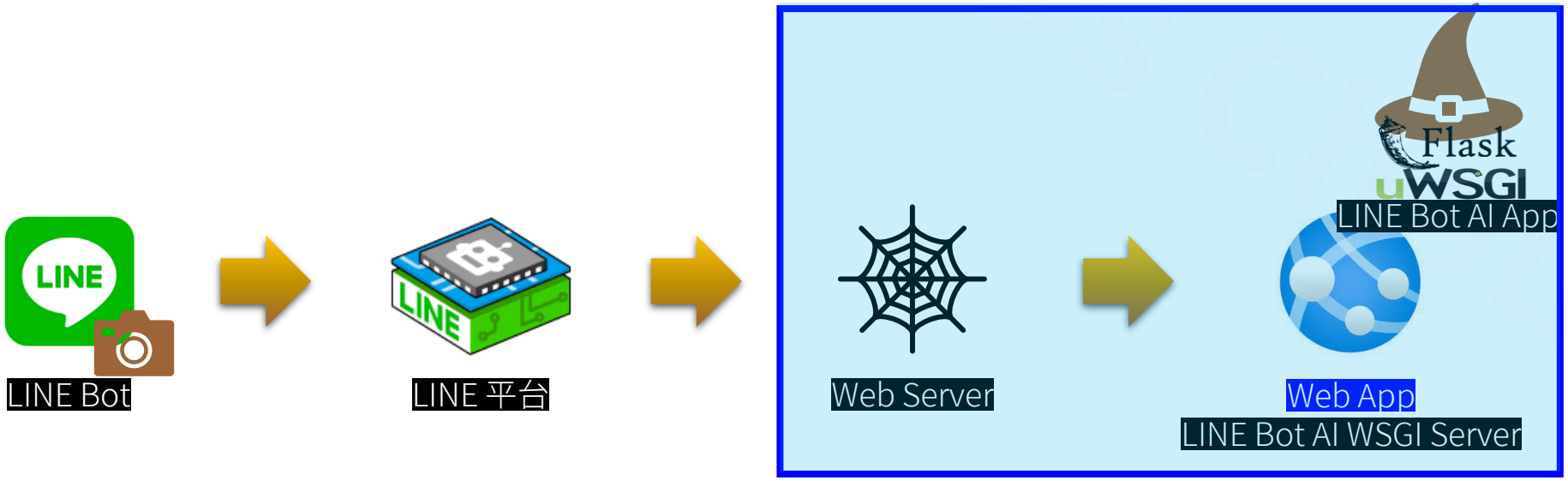
1. 調整 LINE Messaging
    a. 調整 Webhook URL
    b. LINE 測試

架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging



Solution 3 - Serverless 部署

1. 長期可用且廉價的硬體環境 ... Azure Web App
2. Flask as Web Server 的替代方案 ... Azure Web App
3. 長期可用且廉價的 SSL 網域方案 ... Azure Web App

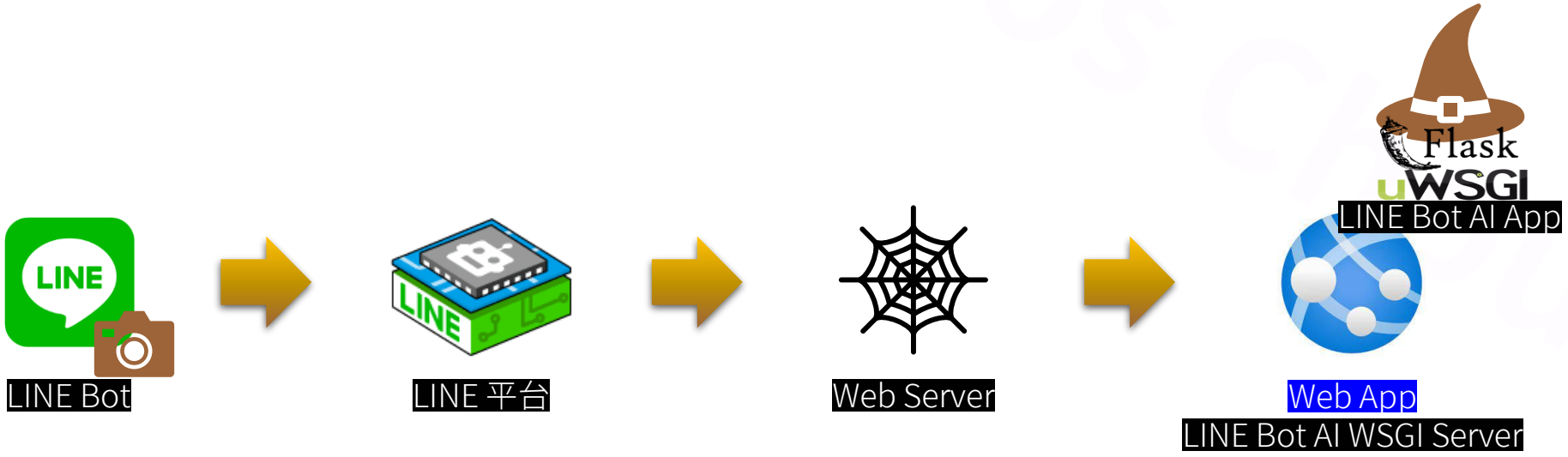


任務

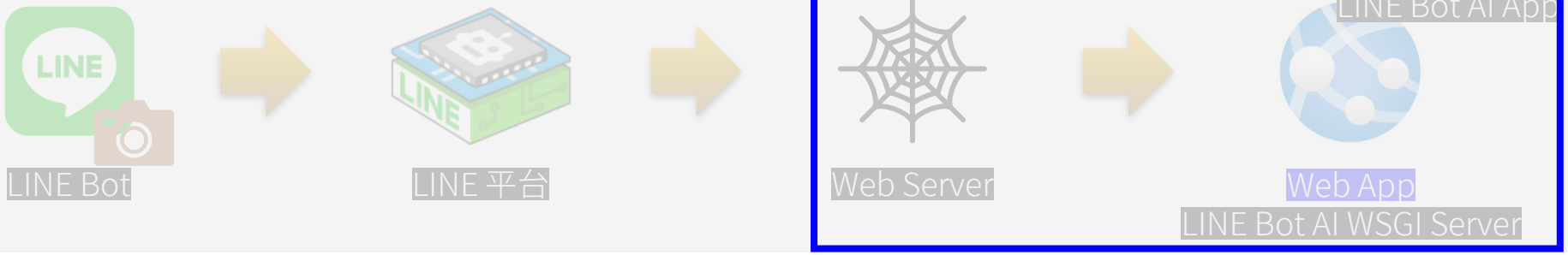
1. 上傳程式至 Cloud Shell
2. 建立 Container 並推送至容器登錄
3. 部署 Container 至應用程式服務

流程

架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging



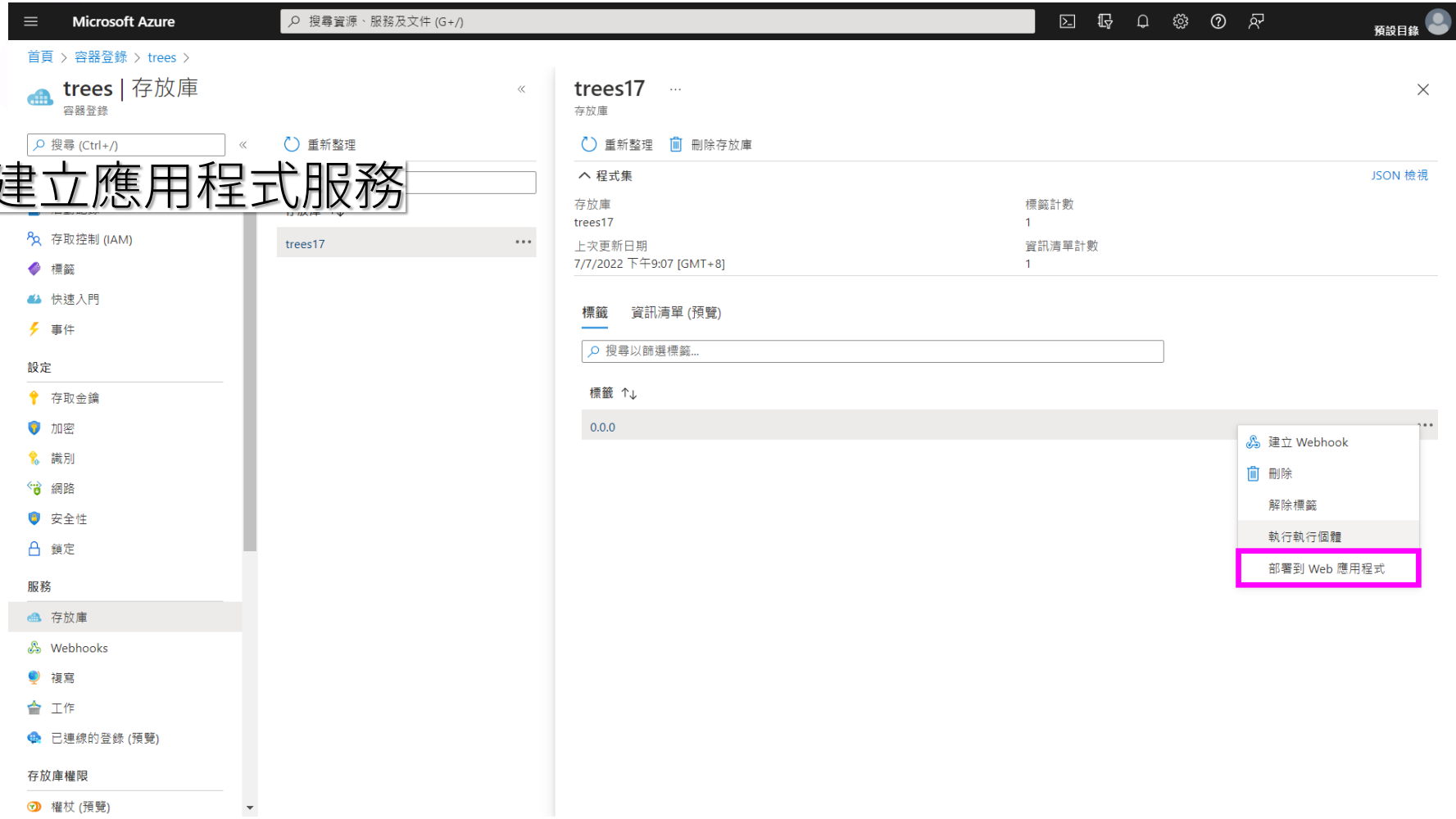
架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging



LINE Bot & WSGI

1. 部署 Web App

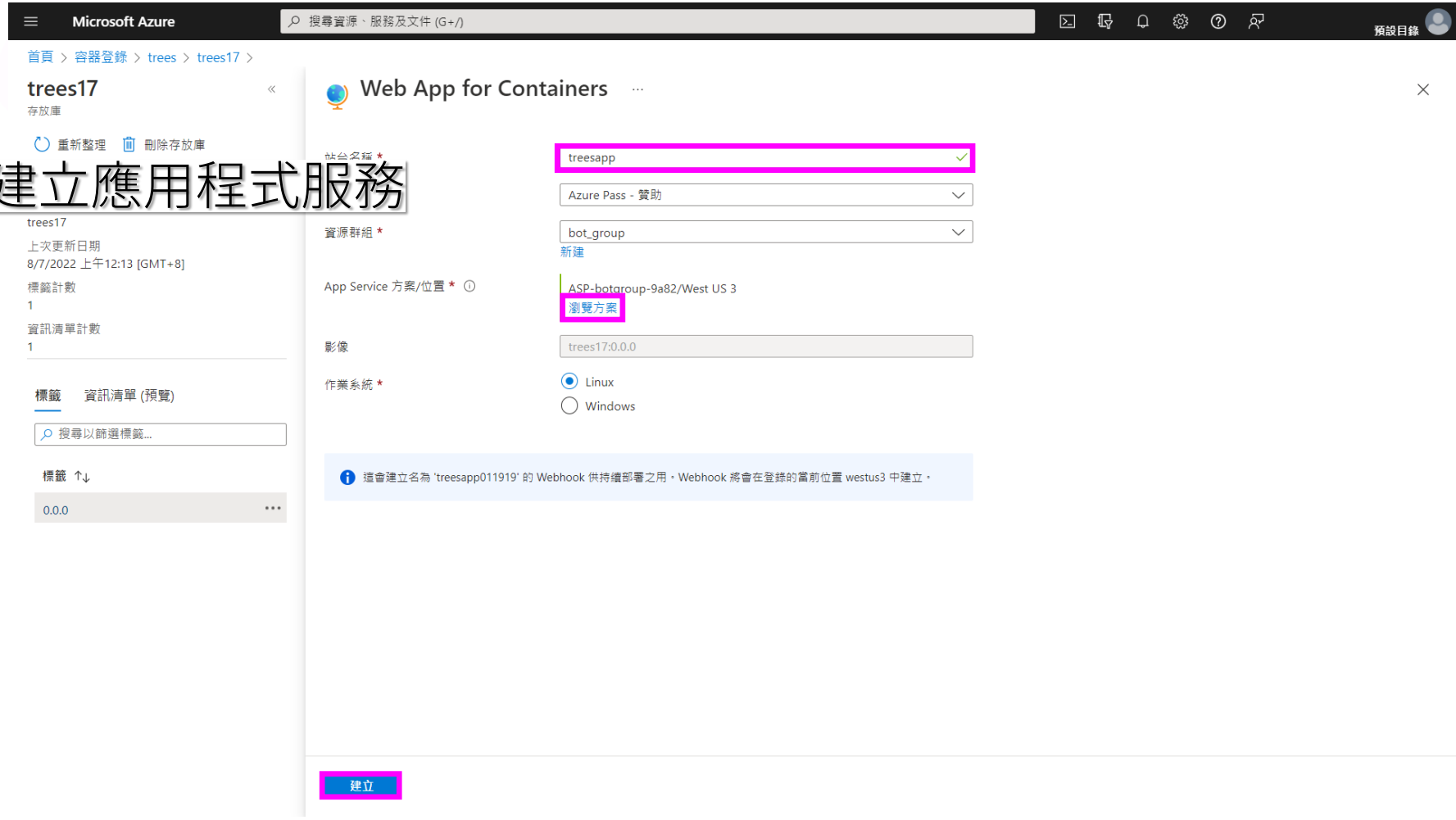
   a. 由 Container 建立應用程式服務



LINE Bot & WSGI

1. 部署 Web App

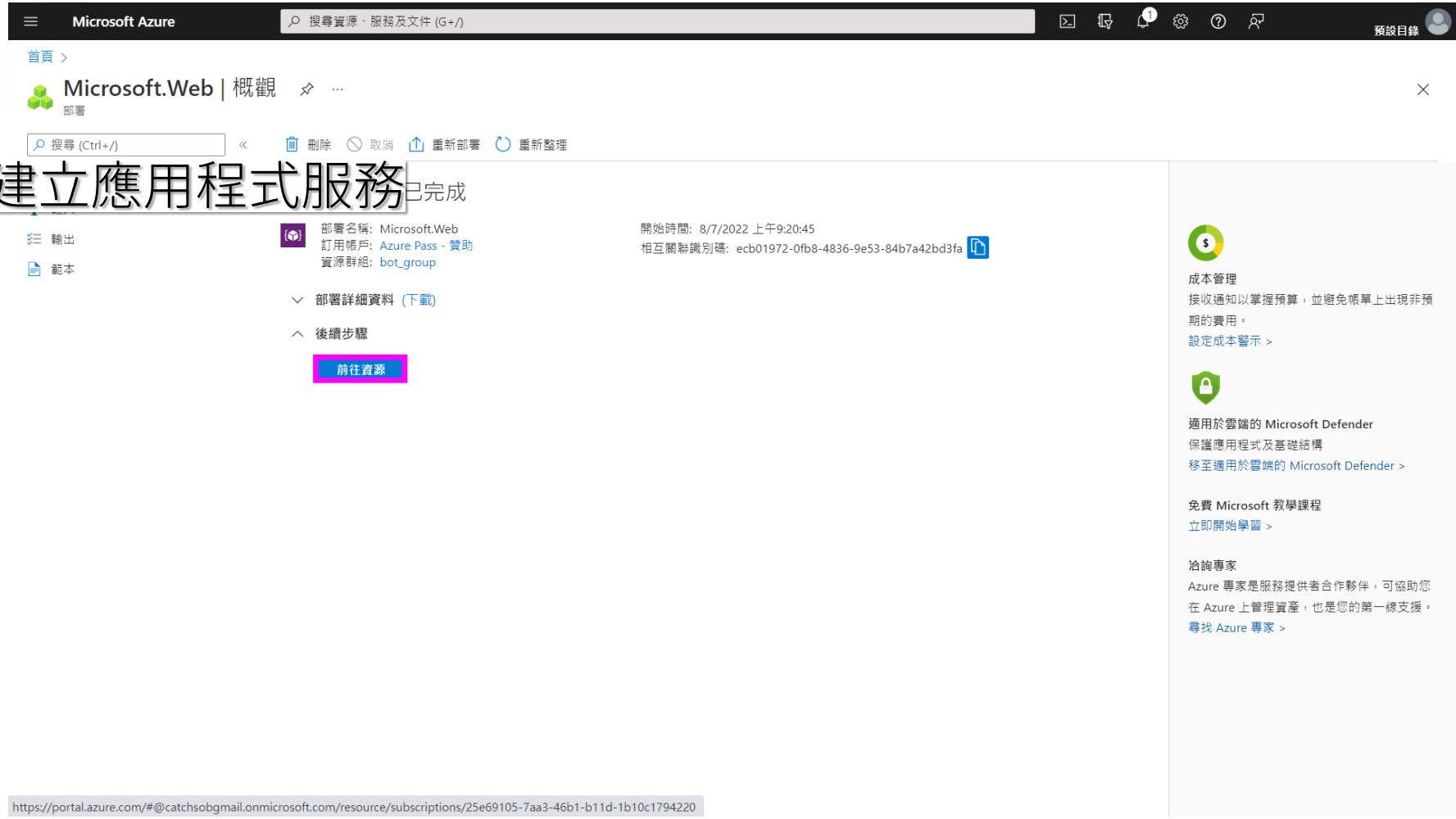
   a. 由 Container 建立應用程式服務



LINE Bot & WSGI

1. 部署 Web App

   a. 由 Container 建立應用程式服務

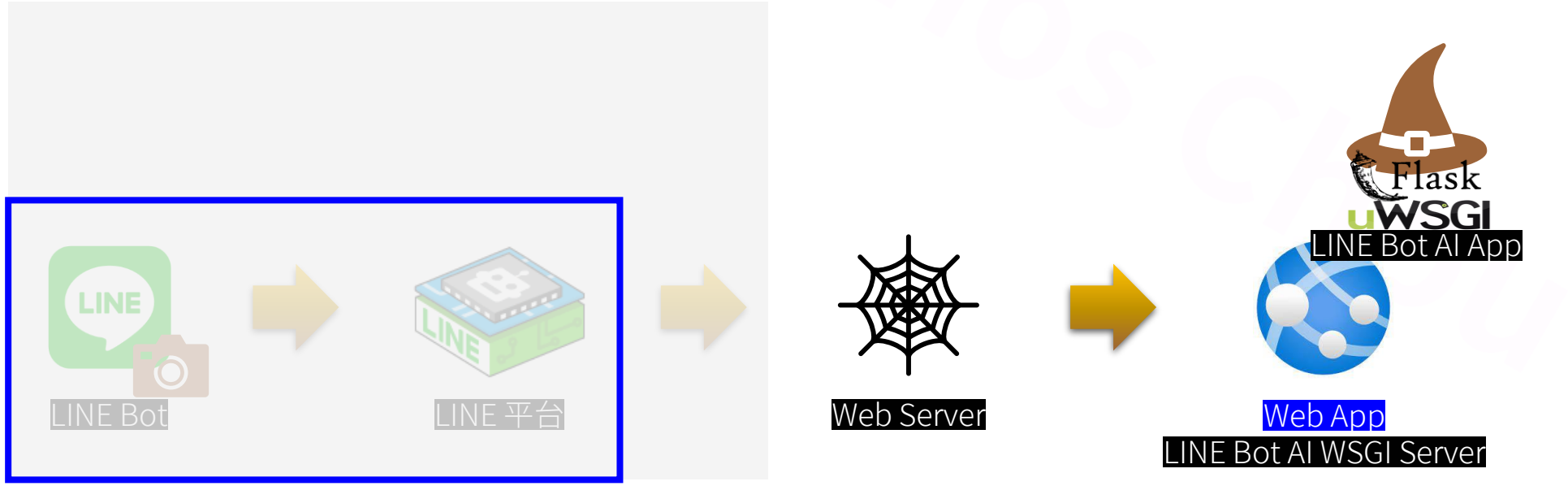


LINE Bot & WSGI

1. 部署 Web App

   a. 由 Container 建立應用程式服務

架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging



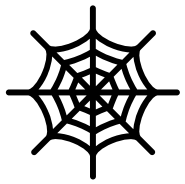
設定 LINE Messaging

1. 調整 LINE Messaging
    a. 調整 Webhook URL
    b. LINE 測試

架設 VM → LINE Bot & WSGI → 申購並設定網域 → 網站與憑證 → 設定 LINE Messaging

LINE Bot

LINE 平台



Web Server

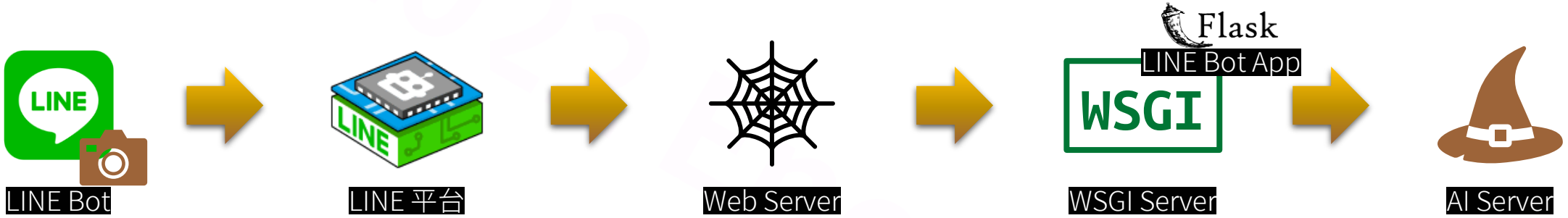
LINE AI Bot App

Web App
LINE Bot AI WSGI Server

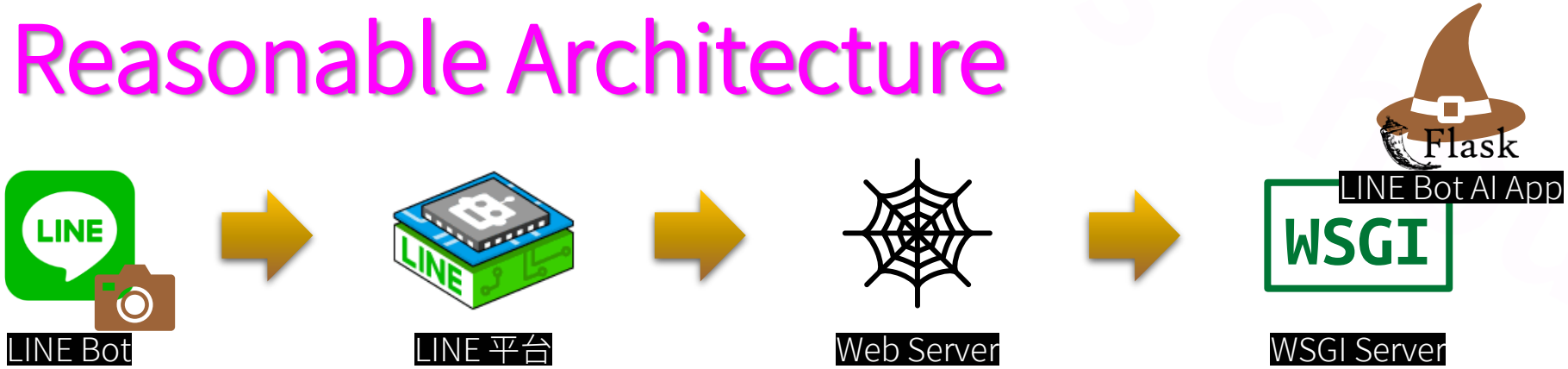
LINE Bot 部署 + AI 部署

Architecture

Smoother Architecture

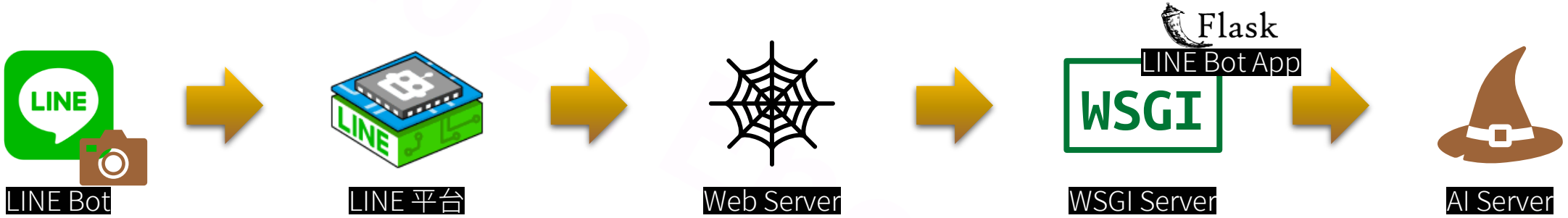


Reasonable Architecture

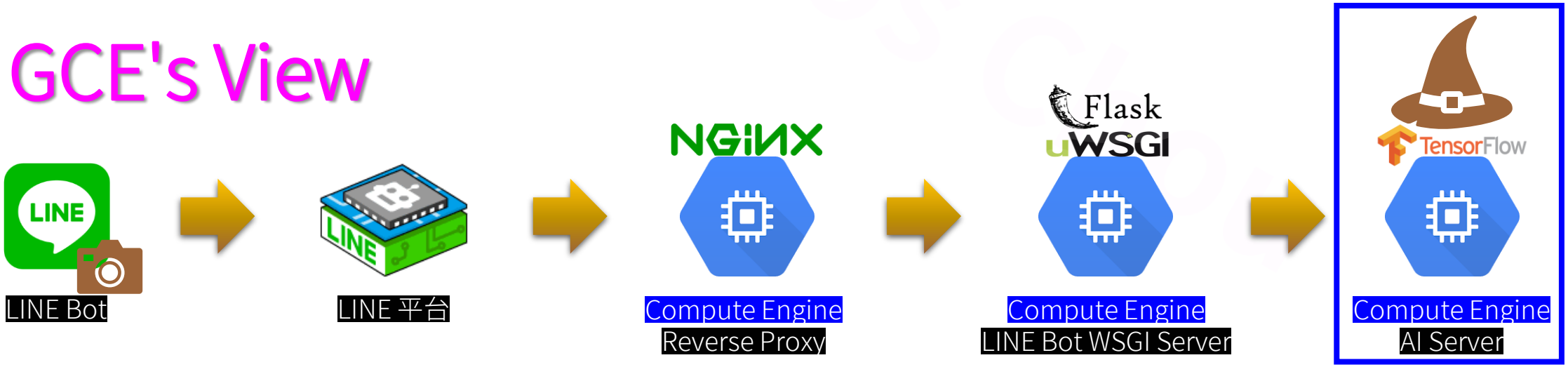


部署 TensorFlow Serving - VM

Smoother Architecture

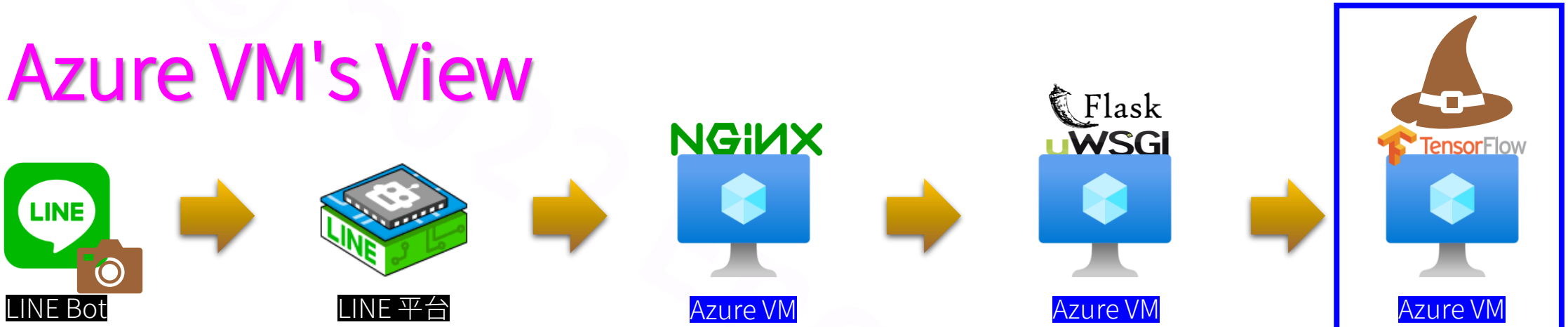


GCE's View

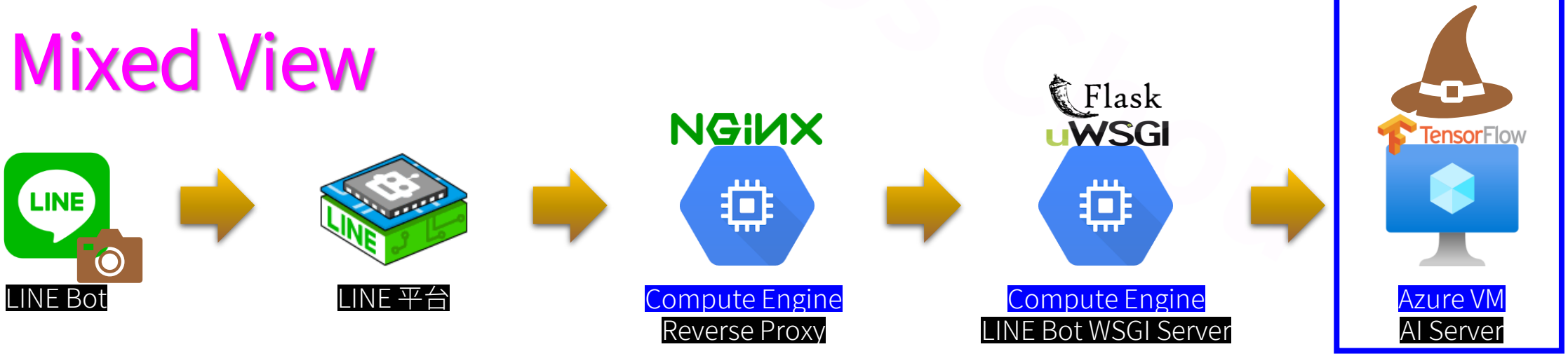


部署 TensorFlow Serving - VM

Azure VM's View



Mixed View



部署 TensorFlow Serving - VM

1. 建立 VM

- OS: Ubuntu 20.04
- RAM: 1GB
- HD: 10GB
- PORT: 8500, 8501 TCP

① Google Compute Engine 防火牆開通

導覽選單 > 虛擬私有雲網路 > 防火牆 > 建立防火牆規則 >
名稱："**自行命名**"；目標："**網路中的所有執行個體**"；來源 IPv4 範圍："**0.0.0.0/0**"；
指定的通訊協定和埠: "**TCP**" "**8500, 8501**" > 建立

② Azure VM 防火牆開通

虛擬機器 > 進入指定 VM > 網路 > 新增輸入連接埠規則 >
目的地連接埠範圍："**TCP**" "**8500, 8501**"；名稱："**自行命名**" > 新增

部署 TensorFlow Serving - VM

2. 準備模型

a. 轉換模型為 TensorFlow SavedModel 格式

① 準備 TensorFlow 2.4.4 以上環境

```
pip install tensorflow==2.4.4
```

② 轉換模型

```
hdf5_to_savedmodel.py your_hdf5.h5 your_savedmodel
hdf5_to_savedmodel.py trees17V1.h5 1
```

部署 TensorFlow Serving - VM

2. 準備模型

b. 上傳 SavedModel 模型至 VM

① 登入 `VM`

② 建立模型目錄

③ 上傳 `SavedModel` 至模型目錄

部署 TensorFlow Serving - VM

2. 準備模型

b. 上傳 SavedModel 模型至 VM

```
# 路徑範例
/home/your_account/your_model/1/assets
/home/your_account/your_model/1/saved_model.pb
/home/your_account/your_model/1/variables/variables.data-00000-of-00001
/home/your_account/your_model/1/variables/variables.index
```

Note

① 模型放在 /home/someone/*your_model*/2 ... N/ 之下，系統會選用數字最大的模型

② 切勿忽略 variables 目錄下檔案

部署 TensorFlow Serving - VM

3. 安裝 TensorFlow Serving

安裝最新版本 TensorFlow Server (v2.9.0)

```
echo "deb [arch=amd64] http://storage.googleapis.com/tensorflow-serving-apt stable tensorflow-model-server tensorflow-model-server-universal" | sudo tee /etc/apt/sources.list.d/tensorflow-serving.list; curl https://storage.googleapis.com/tensorflow-serving-apt/tensorflow-serving.release.pub.gpg | sudo apt-key add -; sudo apt-get update; sudo apt-get install tensorflow-model-server
```

Note
系統不須安裝其他 TensorFlow 套件

部署 TensorFlow Serving - VM

4. 啟動 TensorFlow Serving

啟動 TensorFlow Serving for RETful API & gRPC

```
nohup tensorflow_model_server --model_base_path=your_model_path --
model_name=your_model_name --rest_api_port=your_rest_port --
port=your_grpc_port &
```

```
nohup tensorflow_model_server --model_base_path=/home/enos/trees --
model_name=trees --rest_api_port=8501 --port=8500 &
```

Note

① `--rest_api_port` for RESTful API

② `--port` for gRPC

③ `nohup ... &` for 背景執行

部署 TensorFlow Serving - VM

5. 測試 TensorFlow Serving

a. 測試 RESTful API

```
python tfserving_test.py --help
usage: tfserving_test.py [-h] [--protocol {gRPC,REST}] [--ssl] [--host HOST]
                         [--port PORT] [--labels LABELS] [--input INPUT]
                         [--output OUTPUT]
                         MODEL PIC


python tfserving_test.py --host 34.168.152.9 --port 8501 --protocol REST --
labels treeset_labels.txt trees 鳳凰木.jpg
```

部署 TensorFlow Serving - VM

5. 測試 TensorFlow Serving

b. 測試 gRPC

```
# 確認 SavedModel I/O 名稱
python model_io.py your_savedmodel
python model_io.py 1
```

Note
gRPC 需要 model I/O 名稱作為參數

部署 TensorFlow Serving - VM

5. 測試 TensorFlow Serving

b. 測試 gRPC

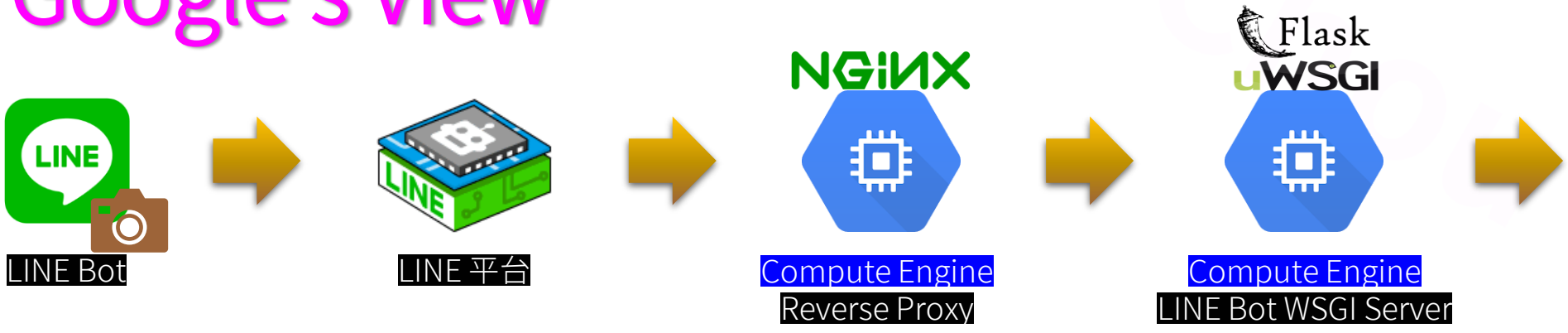
```
python tfserving_test.py --help
usage: tfserving_test.py [-h] [--protocol {gRPC,REST}] [--ssl] [--host HOST]
                         [--port PORT] [--labels LABELS] [--input INPUT]
                         [--output OUTPUT]
                         MODEL PIC


python tfserving_test.py --host 34.168.152.9 --port 8500 --protocol gRPC --
input input_4 --labels treeset_labels.txt trees 鳳凰木.jpg
```

部署 TensorFlow Serving - Cloud Run

Mixed View

Google's View



部署 TensorFlow Serving - Cloud Run

1. 準備模型

a. 轉換模型為 TensorFlow SavedModel 格式

```
# 準備 TensorFlow 2.4.4 以上環境
pip install tensorflow==2.4.4

# 轉換模型
hdf5_to_savedmodel.py your_hdf5.h5 your_savedmodel
hdf5_to_savedmodel.py trees17V1.h5 1
```

部署 TensorFlow Serving - Cloud Run

1. 準備模型

b. 將 SavedModel 置於模型目錄

① 建立模型目錄

② 上傳或複製 `SavedModel` 至模型目錄

部署 TensorFlow Serving - Cloud Run

1. 準備模型

b. 將 SavedModel 置於模型目錄

```
# 路徑範例
your_model/1/assets
your_model/1/saved_model.pb
your_model/1/variables/variables.data-00000-of-00001
your_model/1/variables/variables.index
```

Note

① 模型放在 `your_model/2 ... N/` 之下，系統會選用數字最大的模型

② 切勿忽略 `variables` 目錄下檔案

部署 TensorFlow Serving - Cloud Run

2. 製作 Container

a. 製作 Dockerfile

```
# Dockerfile 範例
FROM tensorflow/serving:2.8.0
COPY your_model/1/ /models/your_model/1
ENTRYPOINT tensorflow_model_server --port=$PORT --model_name=your_model --
model_base_path=/models/your_model
```

Note
此範例採用 gRPC，可自行調整為 RESTful API

部署 TensorFlow Serving - Cloud Run

2. 製作 Container

b. 將 Dockerfile 置於模型目錄同層級

```
# 路徑範例
Dockerfile
your_model/1/assets
your_model/1/saved_model.pb
your_model/1/variables/variables.data-00000-of-00001
your_model/1/variables/variables.index
```

部署 TensorFlow Serving - Cloud Run

3. 部署

a. 以 Google Cloud Shell 編輯器部署

① 上傳 trees 目錄與 Dockerfile 至 GCP 專案帳號之 Cloud Shell
② 開啟 Cloud Shell 編輯器，指定 Workspace 為 Dockerfile 所在目錄，包含 trees 目錄
③ 以 Cloud Shell 編輯器部署 Cloud Run

```
Cloud Shell 編輯器 > Coud Code - Cloud Run: Cloud Run >
(Enable Cloud Run API) > (Login to Google Cloud SDK) >
(授權 Cloud Shell： 授權) > (Select a project) > Deploy to Cloud Run >
Create a service > Service name："自行命名"；Region："自行指定" >
Show Advanced Settings >  Memory allocated："512" MiB > Deploy
```

部署 TensorFlow Serving - Cloud Run

3. 部署

b. 以 Google Cloud Shell + Cloud Run 部署

① 上傳 trees 目錄與 Dockerfile 至 GCP 專案帳號之 Cloud Shell

② `docker build -t gcr.io/your_project/your_container:your_tag .`

③ `docker push gcr.io/your_project/your_container:your_tag`

④ 由 Cloud Run 部署

`Cloud Run` > 建立服務 > 容器映像檔網址："**選擇先前推送的 `Container`**"；服務名稱："**自行命名**"；地區："**自行指定**"；驗證："**允許未經驗證的叫用**" > 建立

Note

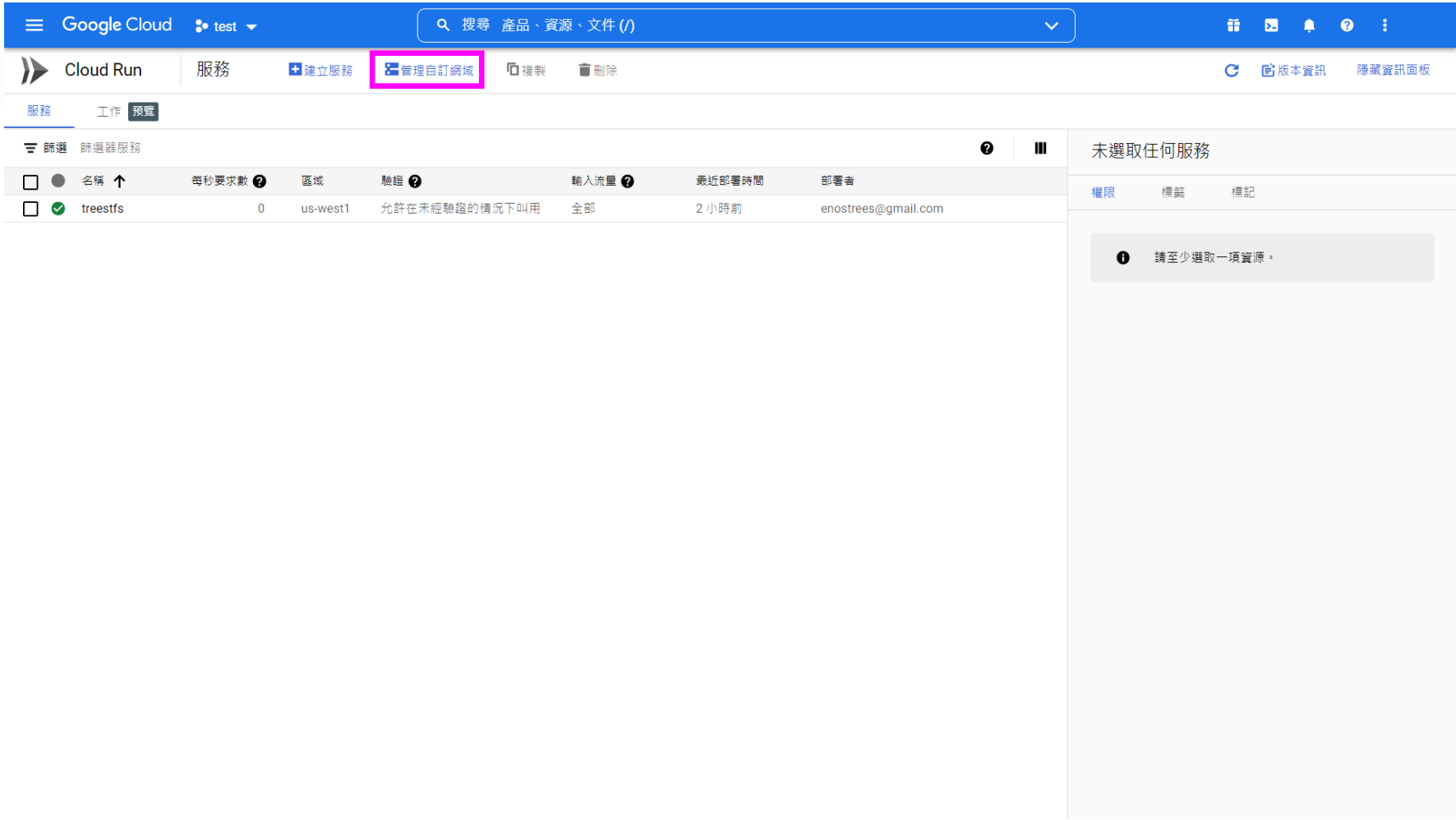
① Cloud Run 記憶體需 512M 以上；建議採用 HTTP/2

② Client 連結時，HOST 須移除 "**https://**"；PORT 為 443，採用 SSL

部署 TensorFlow Serving - Cloud Run

4. (調整 Domain)

   a. 驗證網域



部署 TensorFlow Serving - Cloud Run

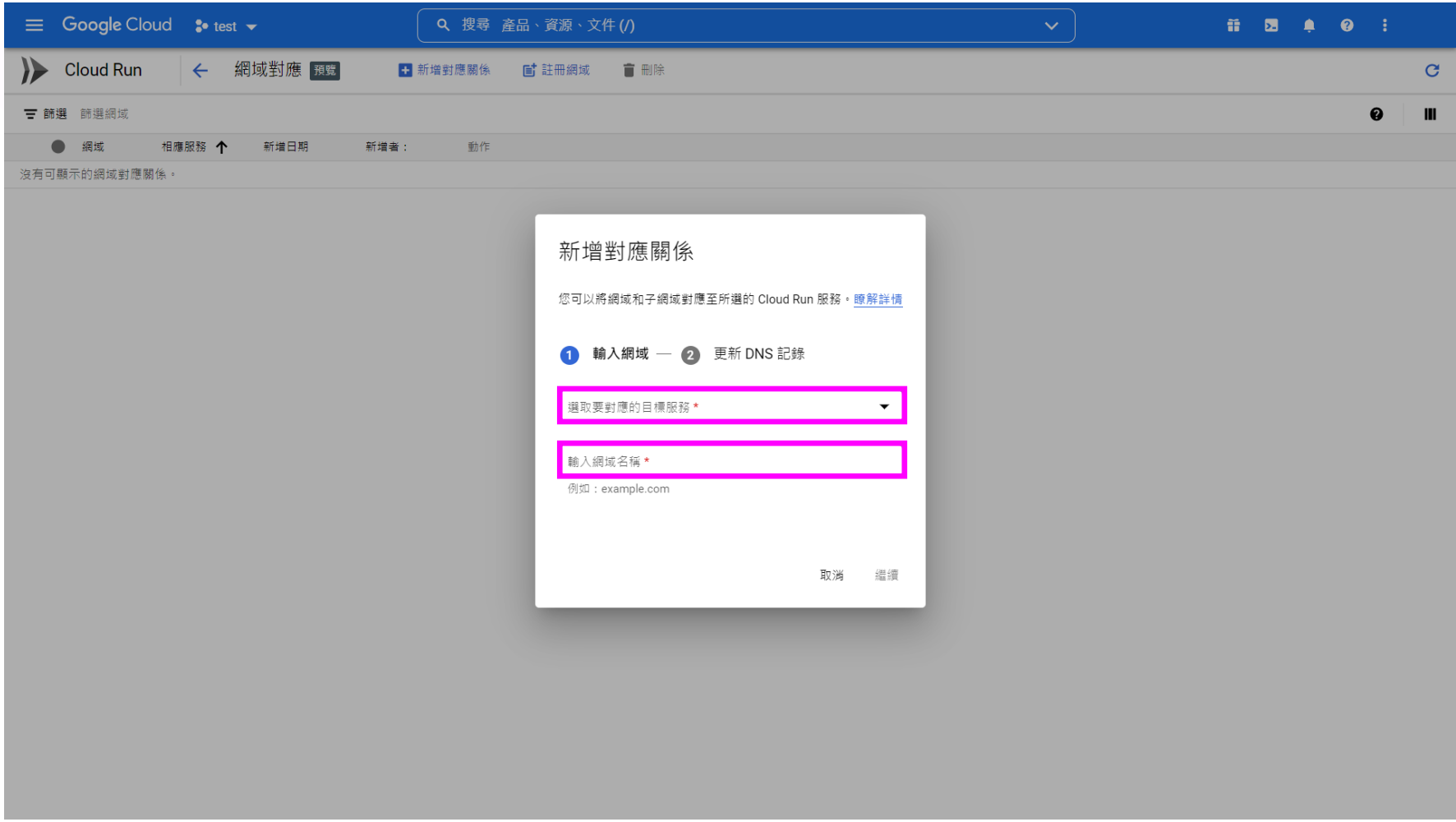
4. (調整 Domain)

   a. 驗證網域

部署 TensorFlow Serving - Cloud Run

4. (調整 Domain)

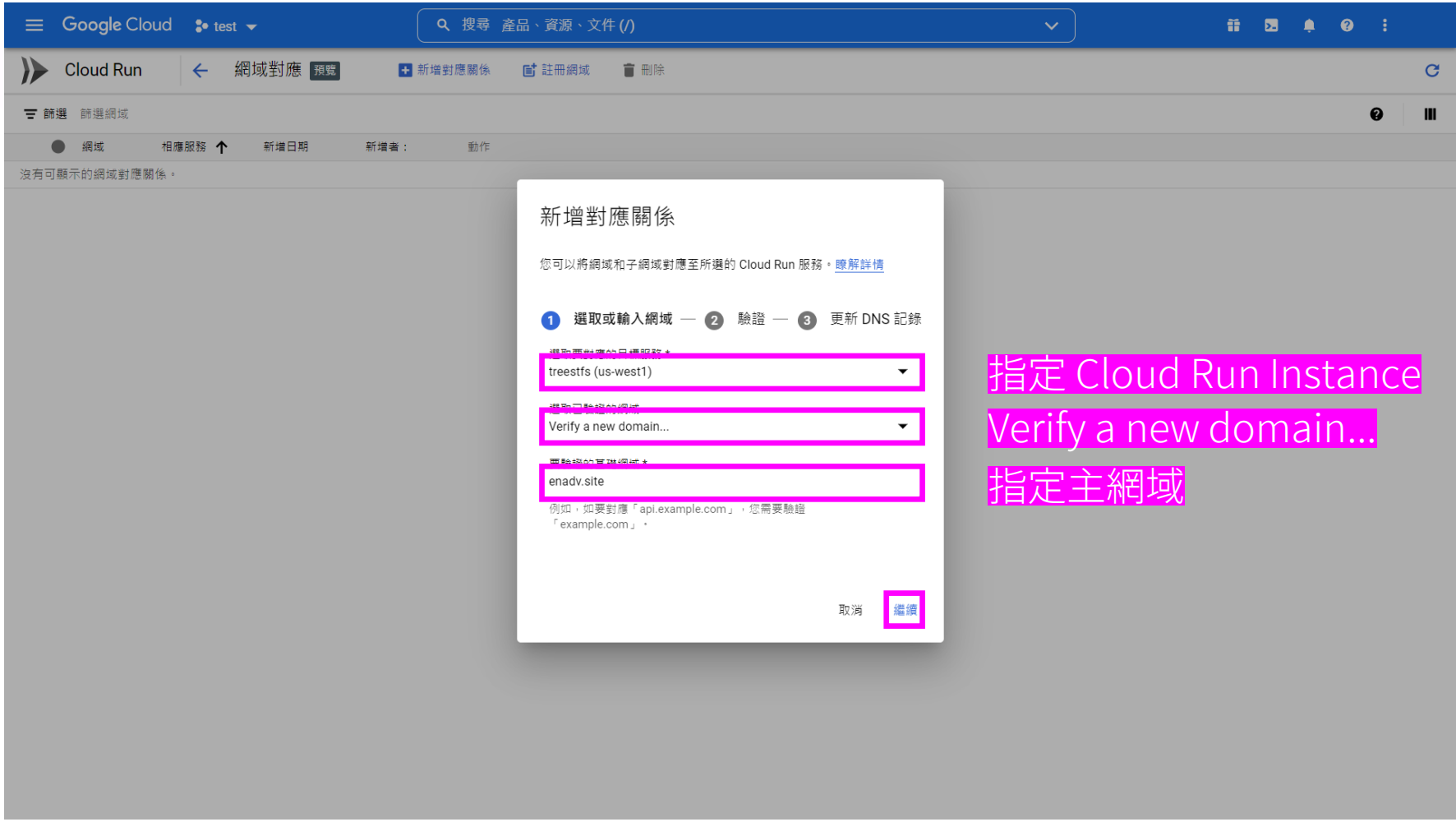
   a. 驗證網域



部署 TensorFlow Serving - Cloud Run

4. (調整 Domain)

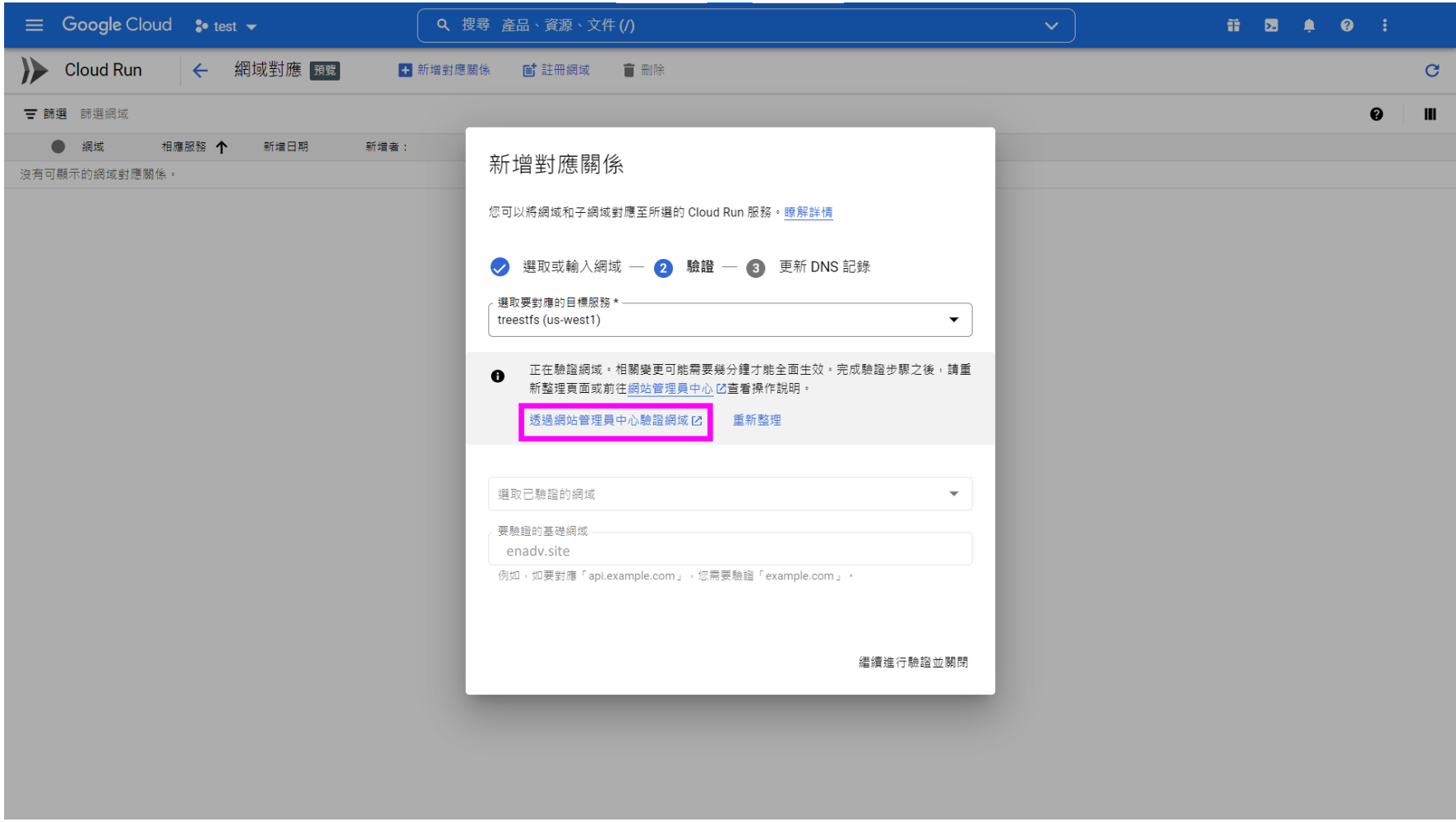
   a. 驗證網域



部署 TensorFlow Serving - Cloud Run

4. (調整 Domain)

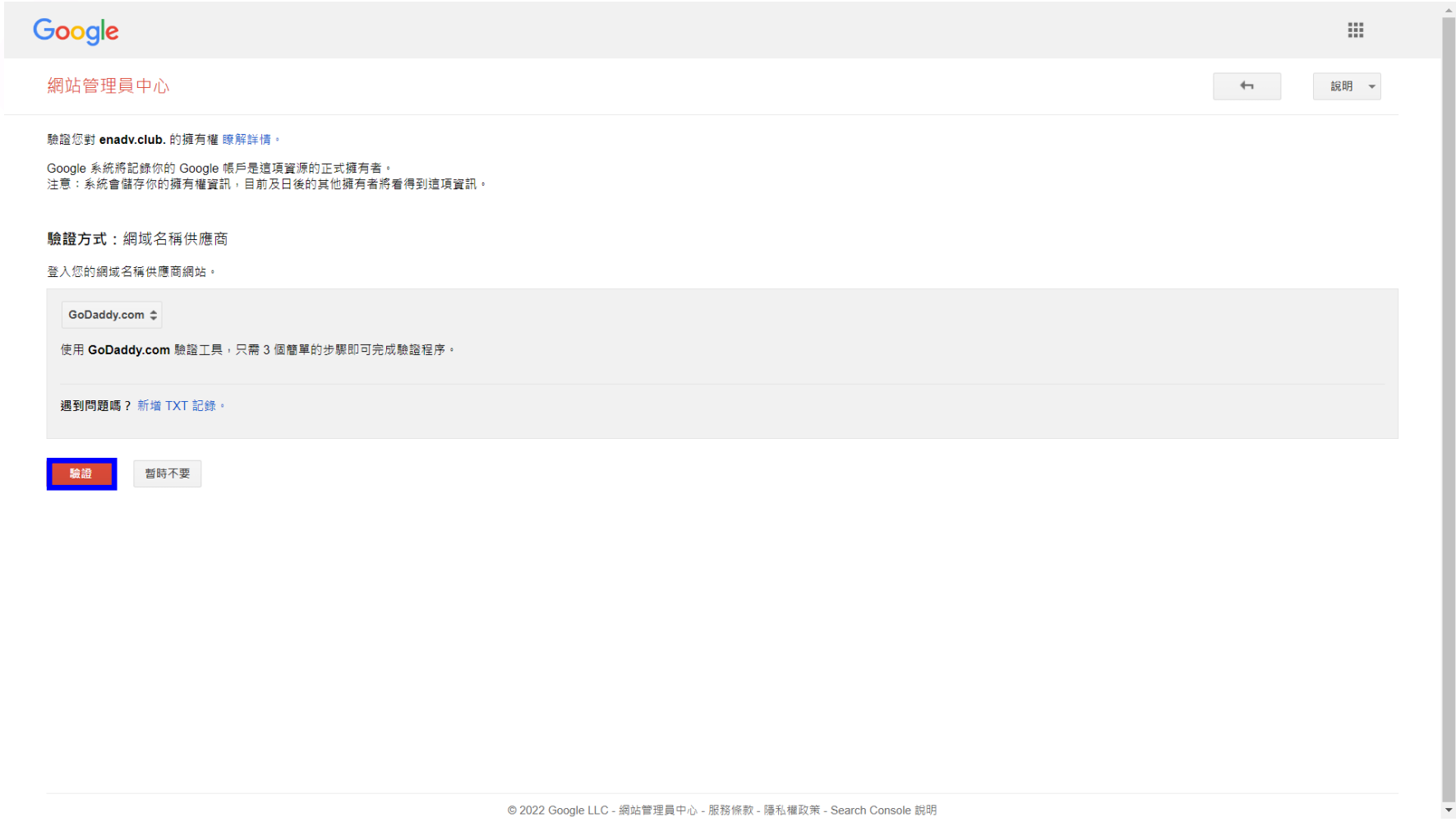
   a. 驗證網域



部署 TensorFlow Serving - Cloud Run

4. (調整 Domain)

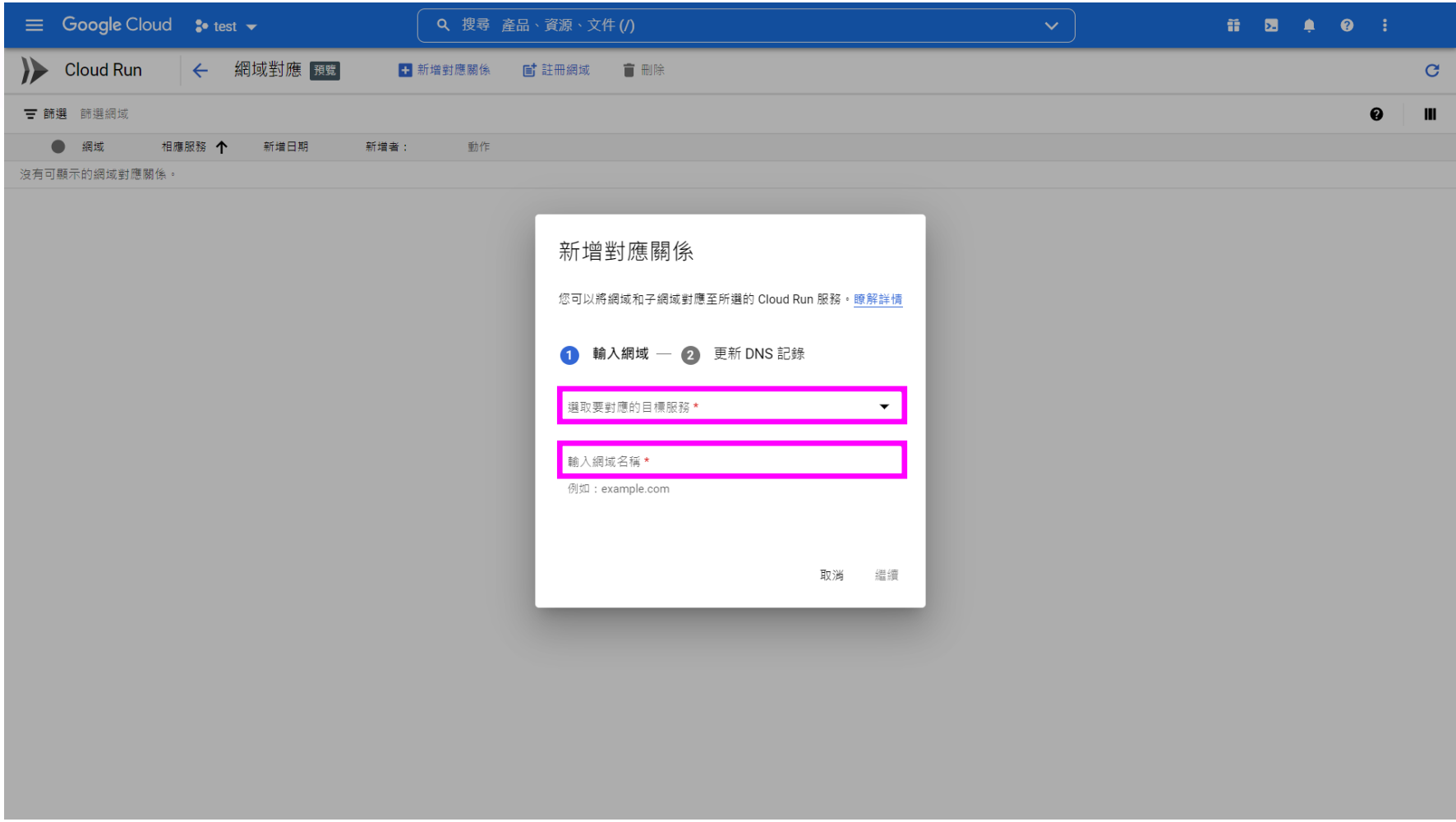
   a. 驗證網域



部署 TensorFlow Serving - Cloud Run

4. (調整 Domain)

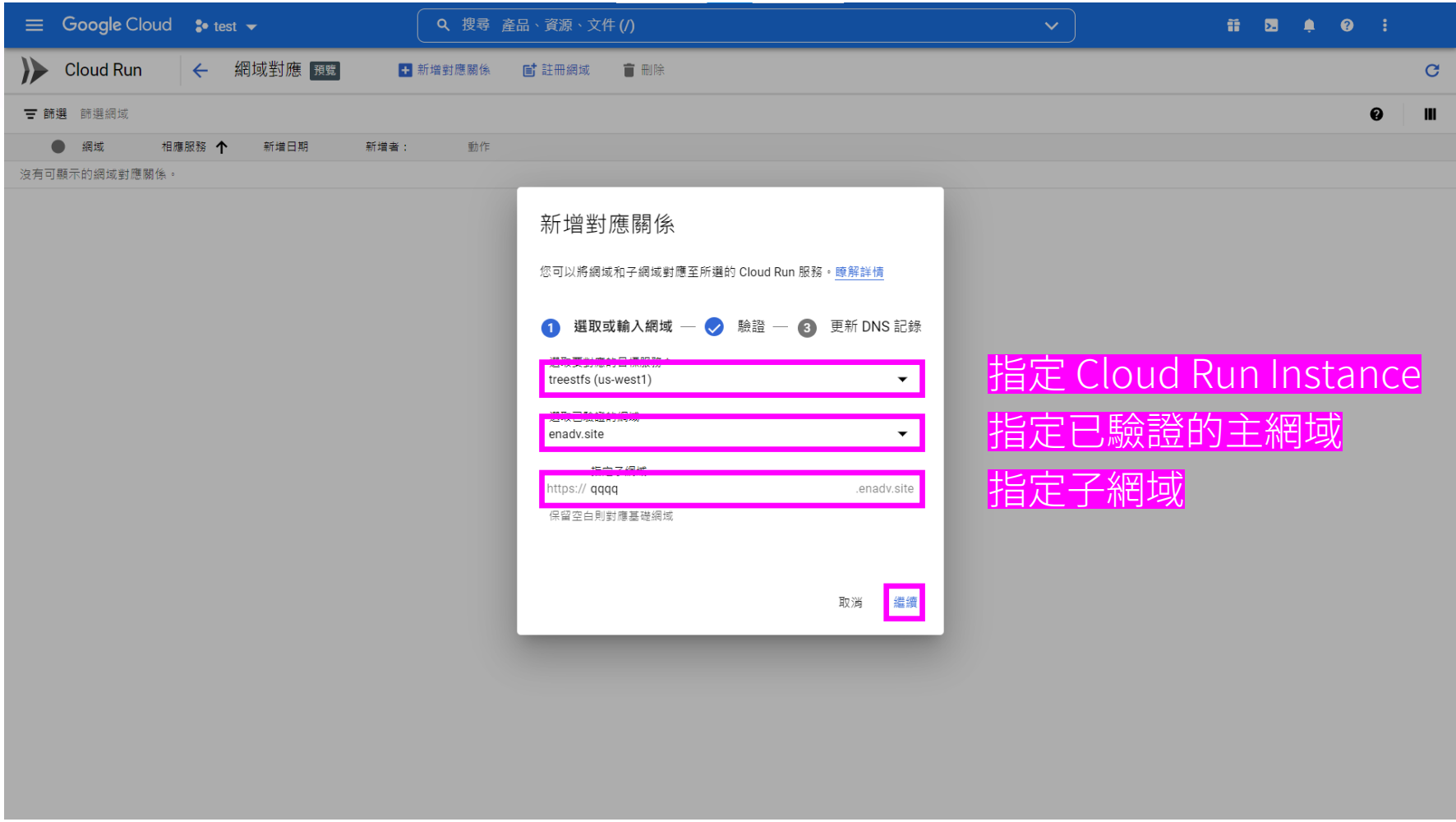
   b. 建立網址對應



部署 TensorFlow Serving - Cloud Run

4. (調整 Domain)

   b. 建立網址對應



部署 TensorFlow Serving - Cloud Run

4. (調整 Domain)

   b. 建立網址對應

部署 TensorFlow Serving - Cloud Run

4. (調整 Domain)

   b. 建立網址對應

部署 TensorFlow Serving - Cloud Run

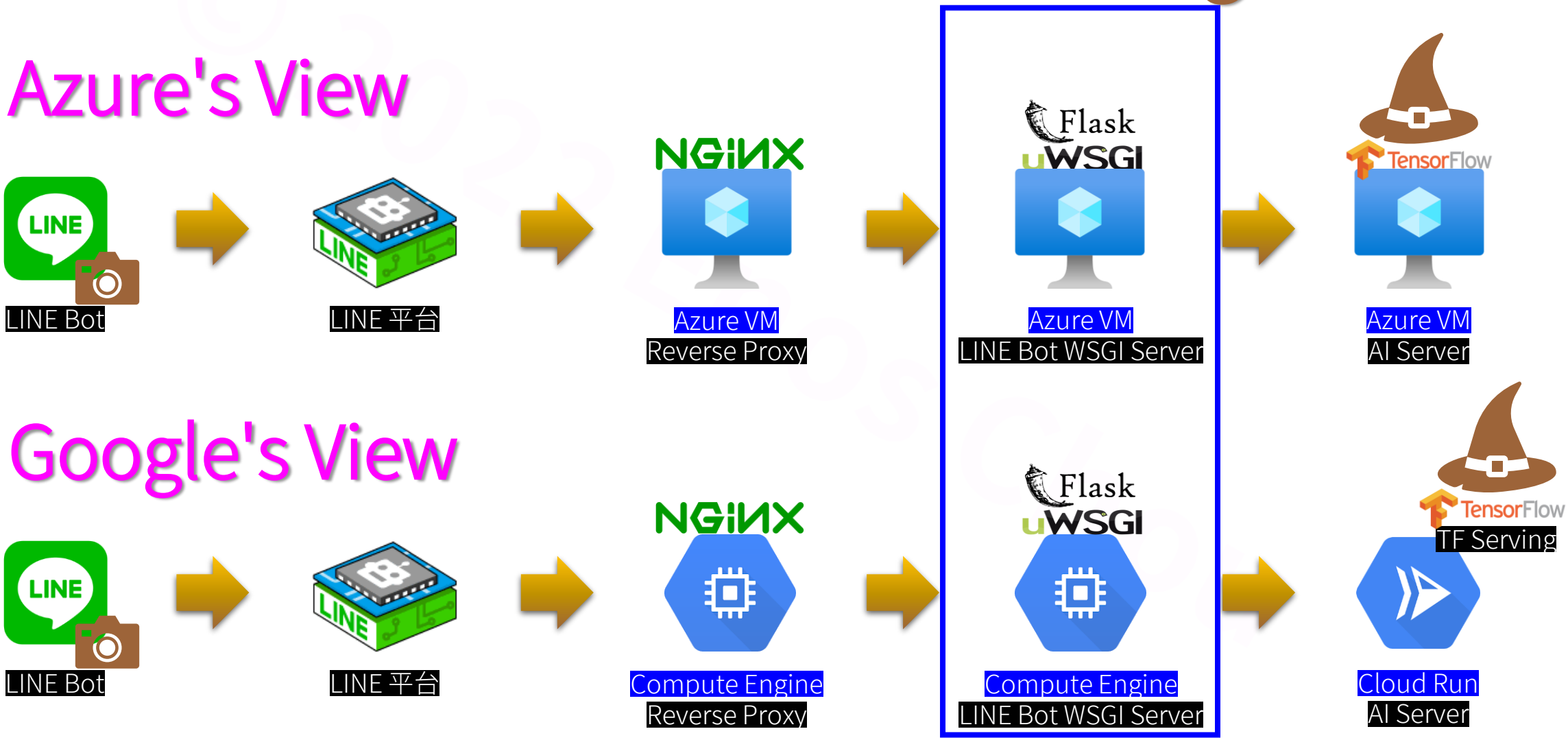
4. (調整 Domain)

b. 建立網址對應

部署 TensorFlow Serving - Cloud Run

5. 測試 TensorFlow Serving

部署 LINE Bot + TensorFlow Serving - REST



部署 LINE Bot + TensorFlow Serving - REST

1. 調整 LINE Bot

RESTful API 叫用 TensorFlow Serving 範例

```
rest = 'http://YOUR_HOST:YOUR_PORT/v1/models/YOUR_MODEL:predict'
headers = {"content-type": "application/json"}
data = json.dumps({"instances": img.tolist()})
r = requests.post(rest, headers=headers, data=data)
p = np.argmax(r.json()['predictions'])
```

部署 LINE Bot + TensorFlow Serving - REST

2. 重新部署 LINE Bot

a. 上傳新版 LINE Bot

```
# 須先行調整參數
trees17bot_rest.py
```

部署 LINE Bot + TensorFlow Serving - REST

2. 重新部署 LINE Bot

b. 更新套件

```
line-bot-sdk
flask
pillow
~~tensorflow==2.4.4~~
requests
numpy
uwsgi
```

部署 LINE Bot + TensorFlow Serving - REST

2. 重新部署 LINE Bot

c. 以 uWSGI 重新啟動 LINE Bot

```
# 取得舊程序 PID
ps -ef | grep uwsgi

# 刪除舊程序
kill your_pid

# 重起服務
uwsgi -w your_module:app -s :your_port -d your_project.log
uwsgi -w trees17bot_rest:app -s :3000 -d trees.log
```

部署 LINE Bot + TensorFlow Serving - REST

2. 重新部署 LINE Bot

d. LINE 測試

部署 LINE Bot + TensorFlow Serving - gRPC

1. 調整 LINE Bot

a. gRPC 叫用 TensorFlow Serving 範例

```
grpcurl = 'YOUR_HOST:YOUR_PORT'
ssl = False
model = 'YOUR_MODEL'
modelin = 'YOUR_MODELIN'
modelout = 'YOUR_MODELOUT'
```

部署 LINE Bot + TensorFlow Serving - gRPC

1. 調整 LINE Bot

a. gRPC 叫用 TensorFlow Serving 範例

```
if ssl:
    channel = grpc.secure_channel(grpcurl, grpc.ssl_channel_credentials())
else:
    channel = grpc.insecure_channel(grpcurl)
stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
req = predict_pb2.PredictRequest()
req.model_spec.name = model
req.inputs[modelin].CopyFrom(make_tensor_proto(img))
r = stub.Predict(req, 13.0)
p = np.argmax(r.outputs[modelout].float_val)
```

部署 LINE Bot + TensorFlow Serving - gRPC

1. 調整 LINE Bot

b. requirements.txt

```
line-bot-sdk
flask
pillow
tensorflow==2.4.4
tensorflow-serving-api
~~requests~~
numpy
uwsgi
```

Note

```
pip3 install -r requirements.txt

pip3 install --no-deps tensorflow-serving-api
```

部署 LINE Bot + TensorFlow Serving - gRPC

2. 重新部署 LINE Bot

a. 上傳新版 LINE Bot

```
# 上傳以下檔案至 VM 專案目錄
①  trees17bot_grpc.py  # 須先行調整參數

②  treeset_labels.txt

③  requirements.txt  # 須先行調整套件
```

部署 LINE Bot + TensorFlow Serving - gRPC

2. 重新部署 LINE Bot

b. 安裝或視需要更新套件

```
cd; cd your_project

pip3 install -r requirements.txt

pip3 install --no-deps tensorflow-serving-api
```

Note

① 若以既有平台部署可於部署前執行 `pip3 list` 確認需求套件是否已存在；安裝缺乏的套件即可

② `tensorflow-serving-api` 安裝時會一併安裝 `tensorflow`，以 `--no-deps` 避免

部署 LINE Bot + TensorFlow Serving - gRPC

2. 重新部署 LINE Bot

c. 以 uWSGI 重新啟動 LINE Bot

```
# 取得舊程序 PID
ps -ef | grep uwsgi

# 刪除舊程序
kill your_pid

# 重起服務
uwsgi -w your_module:app -s :your_port -d your_project.log
uwsgi -w trees17bot_grpc:app -s :3000 -d trees.log
```

部署 LINE Bot + TensorFlow Serving - gRPC

2. 重新部署 LINE Bot

d. LINE 測試

部署 LINE Bot + TensorFlow Serving - gRPC

3. 進階調整 LINE Bot

Issues

① Channel 建立成本高
建立 channel 耗時 ... 重複使用 Channel

② TensorFlow Serving API 叫用 TensorFlow 函式
導致一併安裝 TensorFlow 套件，浪費空間 (> 300MB) ... 去化 TensorFlow

部署 LINE Bot + TensorFlow Serving - gRPC

3. 進階調整 LINE Bot

a. 重複使用 Channel

略

部署 LINE Bot + TensorFlow Serving - gRPC

3. 進階調整 LINE Bot

b. 去化 TensorFlow

Note
① 需要套件 `grpcio-tools`
② 建議在開發環境執行，產出可保存並上傳於部署環境

① 自行建立 proto

- `git clone https://github.com/tensorflow/tensorflow`
- `mkdir p; cd tensorflow`
- `python3 -m grpc.tools.protoc ./tensorflow/core/framework/*.proto --python_out=../p --grpc_python_out=../p --proto_path=.`
- `python3 -m grpc.tools.protoc ./tensorflow/core/example/*.proto --python_out=../p --grpc_python_out=../p --proto_path=.`
- `python3 -m grpc.tools.protoc ./tensorflow/core/protobuf/*.proto --python_out=../p --grpc_python_out=../p --proto_path=.`
- `cd ..; move p/tensorflow to project directory`

部署 LINE Bot + TensorFlow Serving - gRPC

3. 進階調整 LINE Bot

b. 去化 TensorFlow

② 取代 request.inputs[modelin].CopyFrom(make_tensor_proto(img))

```
from tensorflow.core.framework import types_pb2
from tensorflow.core.framework.tensor_shape_pb2 import TensorShapeProto
from tensorflow.core.framework.tensor_pb2 import TensorProto

d = [TensorShapeProto.Dim(size=x) for x in img.shape]
t_p = TensorProto(dtype=types_pb2.DT_FLOAT, tensor_shape=TensorShapeProto(dim=d),
                  tensor_content=img.tobytes())
req.inputs[modelin].CopyFrom(t_p)
```

部署 LINE Bot + TensorFlow Serving - gRPC

3. 進階調整 LINE Bot

b. 去化 TensorFlow

③ 無依賴安裝 tensorflow-serving-api

```
# requirements.txt
line-bot-sdk
flask
pillow
numpy
grpcio
protobuf
uwsgi
```

Note

```
pip3 install -r requirements.txt

pip3 install --no-deps tensorflow-serving-api
```

部署 LINE Bot + TensorFlow Serving - gRPC

4. 重新部署進階 LINE Bot

a. 上傳新版 LINE Bot

```
# 上傳以下檔案至 VM 專案目錄
①  trees17bot_adv.py  # 須先行調整參數

②  treeset_labels.txt

③  requirements.txt  # 須先行調整套件
```

部署 LINE Bot + TensorFlow Serving - gRPC

4. 重新部署進階 LINE Bot

b. 安裝或視需要更新套件

```
cd; cd your_project

pip3 install -r requirements.txt

pip3 install --no-deps tensorflow-serving-api
```

Note

① 若以既有平台部署可於部署前執行 `pip3 list` 確認需求套件是否已存在；安裝缺乏的套件即可

② `tensorflow-serving-api` 安裝時會一併安裝 `tensorflow`，以 `--no-deps` 避免

部署 LINE Bot + TensorFlow Serving - gRPC

4. 重新部署進階 LINE Bot

c. 以 uWSGI 重新啟動 LINE Bot

```
# 取得舊程序 PID
ps -ef | grep uwsgi

# 刪除舊程序
kill your_pid

# 重起服務
uwsgi -w your_module:app -s :your_port -d your_project.log
uwsgi -w trees17bot_adv:app -s :3000 -d trees.log
```

部署 LINE Bot + TensorFlow Serving - gRPC

4. 重新部署進階 LINE Bot

d. LINE 測試

gcloud CLI

專案

初始化

```
gcloud init
```

登入

```
gcloud auth login
```

全部登出

```
gcloud auth revoke --all
```

查詢預設專案

```
gcloud config list project
```

設定預設專案代號

```
gcloud config set project your_project_id
gcloud config set project cryptic-bolt-319014
```

App Engine

部署

```
gcloud app deploy
```

觀察 log

```
gcloud app logs tail -s default
```

gcloud CLI

Compute Engine

SSH 登入

```
gcloud compute ssh your_id@your_vm
```

SCP 複製

```
gcloud compute scp your_file your_id@your_vm:
gcloud compute scp --recurse your_dir your_id@your_vm:
```

TensorFlow.js AI 部署

部署 TensorFlow.js Inference

1. 準備模型

a. 備妥 SavedModel 模型

① 準備 TensorFlow 2.4.4 以上環境

```
pip install tensorflow==2.4.4
```

② 轉換模型

```
hdf5_to_savedmodel.py your_hdf5.h5 your_savedmodel
hdf5_to_savedmodel.py trees17V1.h5 1
```

部署 TensorFlow.js Inference

1. 準備模型

b. 轉換 SavedModel 為 TensorFlow.js Model

① 安裝套件

```
pip install tensorflowjs # 會自動安裝 TensorFlow
```

② 轉換為 TensorFlow.js 格式

```
tensorflowjs_converter --input_format tf_saved_model your_savedmodel your_output
tensorflowjs_converter --input_format tf_saved_model 1 js
```

部署 TensorFlow.js Inference

2. 開發測試

a. 開發 TensorFlow.js Inference

https://www.tensorflow.org/js/

部署 TensorFlow.js Inference

2. 開發測試

b. 架設測試 HTTP Server

```
python -m http.server your_port
python -m http.server 8000
```

部署 TensorFlow.js Inference

2. 開發測試

c. 網頁預覽

```
http://localhost:your_port/your_html
```

部署 TensorFlow.js Inference

3. 部署

a. 建立 VM

部署 TensorFlow.js Inference

3. 部署

b. 上傳程式與模型

① 登入 VM

② 於 /home/*your_account* 建立專案目錄

③ *.html, labels, TensorFlow.js 模型目錄移至專案目錄

部署 TensorFlow.js Inference

3. 部署

c. 架設 NGINX

```
# .conf
server {
    server_name your_domain;

    location /your_path {
        alias /home/your_account/your_project;
        index your_html;
    }
}
```

部署 TensorFlow.js Inference

4. Browser 測試

a. 以瀏覽器開啟網址

`https://your_domain/your_project/your_path`

`https://your_domain/your_project/your_path/your_html`

The End