**SWE40006 – Deployment Activity 3**

**Student Details**

* **Full name:** Dang Khoa Le
* **Student ID:** 103844421
* **Due date:** 14/09/2025
* **Submission date:** 14/09/2025

**Unit Details**

* **Unit code & name:** SWE40006 – Software Deployment and Evolution
* **Semester:** Semester 2, Year 2025

**Declaration of Task Levels Attempted**

* **Task 3.1 (Pass)** – Create Azure account and deploy existing Python app to Azure.
* **Task 3.2 (Credit)** – Develop C# app and deploy to Azure cloud. Deactivate the app.
* **Task 3.3 (High Distinction)** – Install PHP, develop a webapp and deploy to Azure.

**Prerequisites**

* **VS Code** (preferred IDE for MacOS)
* **Azure CLI**
* **Python 3.11** (for Task 3.1)
* **.NET 8 SDK** (for Task 3.2)
* **PHP 8.x** (for Task 3.3)

### **Public URLs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Azure App | URL (public) | Runtime | Region |
| 3.1 Python | *swe40006-python-calculator-2* | https://*swe40006-python-calculator-2.azurewebsites.net* | Python 3.11 (Linux) | Australia East |
| 3.2 C# | *swe40006-*csharp*-calculator* | https://*swe40006-*csharp*-calculator*.azurewebsites.net | .NET 8 (Linux) | Australia East |
| 3.3 PHP | *swe40006-php-webapp* | https://*swe40006-php-webapp*.azurewebsites.net | PHP 8.2 (Linux) | Australia East |

**3.1 - Pass: Deploy Existing Python App to Azure**

**3.1.0 Installations**

* **Azure CLI:** Install Azure CLI to MacOS using command:
* **With Brew:**

brew update && brew install azure-cli

* **With pip** (pip3 since we are using Python3.x on MacOS)**:**

pip3 install azure-cli

Then we will need to edit our  shell's PATH environment variable .zshrc to ensure az command is in my system's PATH. Since we installed with pip and currently have Python 3.11 built-in, I will export this path to my .zshrc file.

A close-up of a computer

Description automatically generated

Figure 1. Edit system path file .zshrc

We can verify Azure CLI installation and version with az –version command.

A screenshot of a computer

Description automatically generated

Figure 2. Check Azure CLI version and installation

* **IDE**: As a Mac user, I decided to progress with this task onward using VS Code.

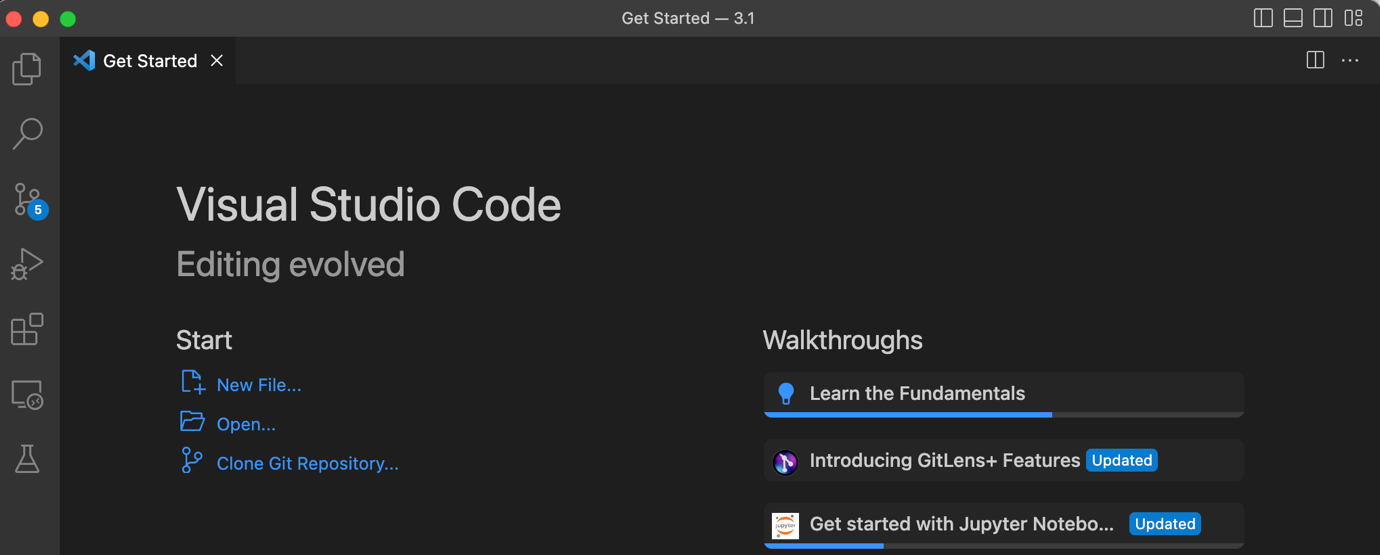


Figure 3. VS Code as IDE preference

**Azure Account**

I previously already created an Azure account from my student email, thus I only need to login with this account on Azure portal.

A screenshot of a computer

Description automatically generated

Figure 4. Microsoft Azure website dashboard

**Sign in and Create resource group**

I will login (verify account) and create resource group (named *rg-swe40006* with region/location at Australia East) from my device terminal with CLI commands following:

az login

az account show

az group create -n rg-swe40006 -l australiaeast

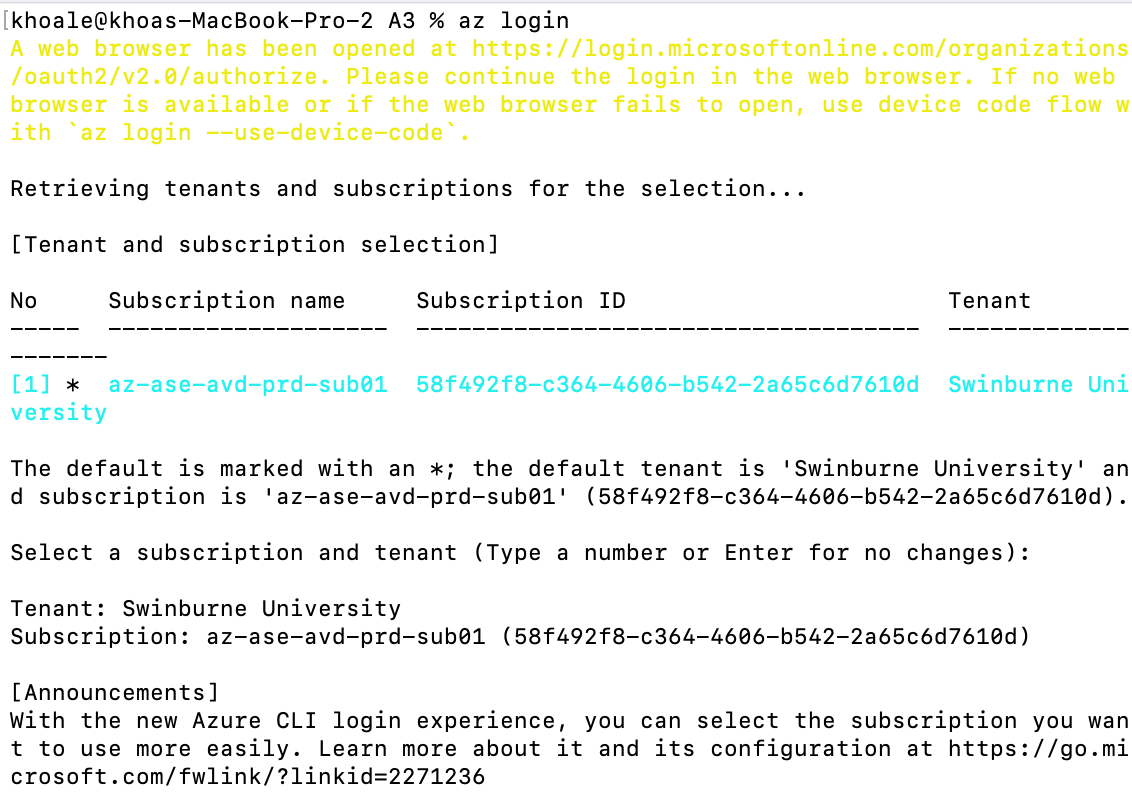


Figure 5. Azure CLI login

A screenshot of a computer code

Description automatically generated

Figure 6. Azure CLI verify account

I need to verify my current permitted/assigned roles (on subscription) for this student email account, which doesn’t exist any role:

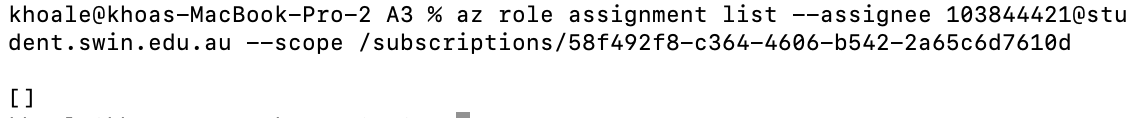
****

Figure 7. No available role on Azure CLI

I returned to Azure dashboard website and on Subscriptions, I choose Azure for Students option. Which then prompt me to sign up again for academic verification. Upon proceed successfully, I will have an Education account with $100 free credit.

A screenshot of a computer

Description automatically generated

Figure 8. Azure website Subscriptions page

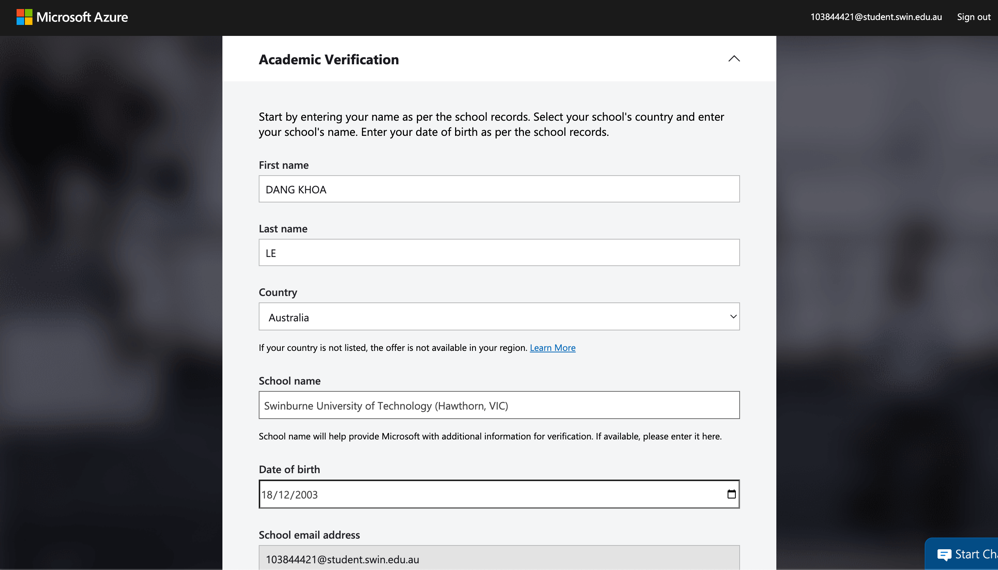


Figure 9. Azure website Academic Verification page

**A screenshot of a computer

Description automatically generated**

Figure 10. Azure website Education account page

Then I proceed to Subscription again, which show my ‘Azure for Students’ section displayed. I then check for its ID and set that subscription ID to my terminal CLI by running:

az account set –subscription id

A screenshot of a computer

Description automatically generated

Figure 11. Azure subscription ID

Once set the subscription, we will proceed with the resource group creation (named *rg-swe40006* with region/location at Australia East).

A screenshot of a computer

Description automatically generated

Figure 12. Azure resource group details

**3.1.1 Create Azure App Service**

**3.1.1.1 Create an App Service plan (free-tier)**

We need to create an App Service plan (named *plan-swe40006*) for our resource group. First, we will need to register Microsoft.Web resource provider, which is required to create App Services (Web Apps, App Service plans, etc):

az provider register –namespace Microsoft.Web

Then, we will configure a free Linux plan for our resource group with:

az appservice plan create -g rg-swe40006 -n plan-swe40006 --sku F1 --is-linux



Figure 13. App Service plan details

**3.1.1.2 Create the Python web app**

For this Azure Linux App Service’s Web app creation, the current supported Python runtime stack (locally) is Python 3.11, so we will explicitly use **3.11** in the --runtime flag. We create the webapp service named *swe40006-python-calculator* that is attached to our resource group and plan as previously configured. Note that the app name must be globally unique. This CLI command was executed:

az webapp create -g rg-swe40006 -p plan-swe40006 -n swe40006-python-calculator --runtime "PYTHON:3.11"

A screenshot of a computer program

Description automatically generated

Figure 14. swe40006-python-calculator details

**3.1.2 Implement FastAPI Calculator**

**3.1.2.1 Project layout**

The root project named “3.1” include main.py (Python script with main functionalities) and requirements.txt (for Python libraries and dependencies version control consistency when creating a project environment).

A screenshot of a computer

Description automatically generated

Figure 15. Calculator Python app setups

**3.1.2.2 Scripts**

**main.py:** A FastAPI app with routes including basic POST request for add, subtract, multiply and divide computations. It also support GET request to check for application health status.

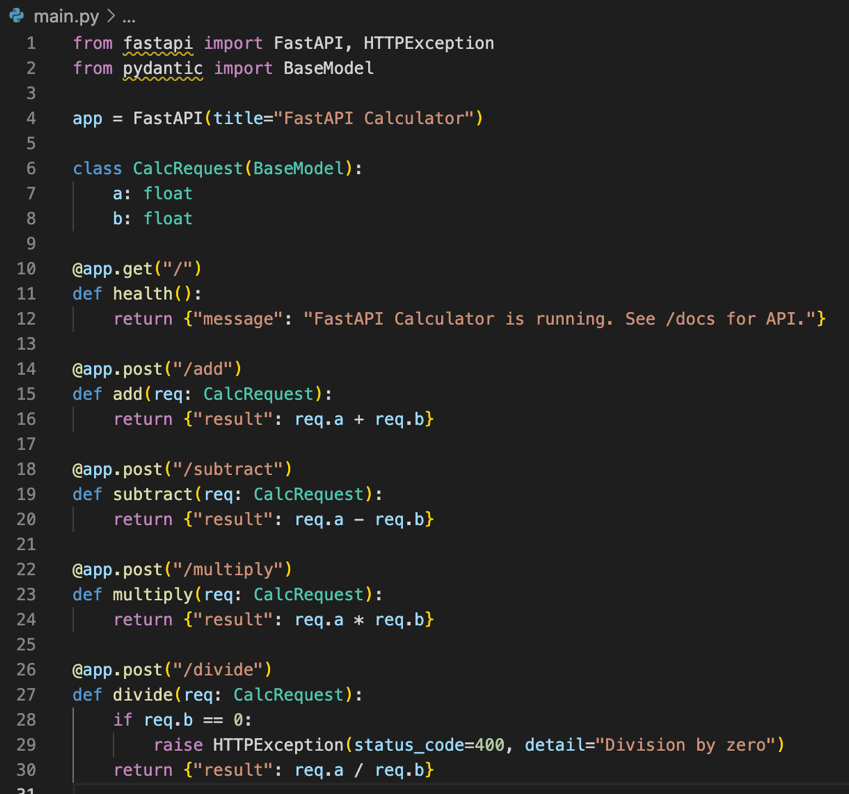
****

Figure 16. main.py

**requirements.txt:** This file storing all required dependencies and modules for our Python FastAPI, with version specifications that support our Python 3.11 runtime.

A screenshot of a computer

Description automatically generated

Figure 17. requirements.txt

**3.1.2.3 Local Testing**

At project directory *3.1*, we create a virtual environment (.venv) and building Python wheels with existing dependencies specified at *requirements.txt* (so this way our current Python environment setup locally will not be impacted from these version changes). Then we run the FastAPI app with unvicorn, following these CLI commands:

python -m venv .venv

pip install -r requirements.txt

uvicorn main:app –reload

Upon launching, the app will be available locally at port 8000 (default Python app startup port)

**A screenshot of a computer

Description automatically generated**

Figure 18. App local test

We will try to execute a few examples, editing value schema (JSON object), then trigger Execute button, which will simulate a CURL request to our backend which execute mathematical computation.

A screenshot of a computer

Description automatically generated

Figure 19. Example test with Add route

A screenshot of a computer

Description automatically generated

Figure 20. Accurate addition output

Other mathematical computation routes are tested also, which all show accurate outputs.

**3.1.3 Deploy from VS Code (Zip Deploy)**

**3.1.3.1 VS Code Azure setup**

* Install Azure App Service extension for deployment.

A screenshot of a computer

Description automatically generated

Figure 21. VS Code extension Azure App Service

A screenshot of a computer

Description automatically generated

Figure 22. Azure App Service extension details

* In Explorer, right-click *3.1* → Deploy to Web App → select *swe40006-python-calculator* app → Confirm.
* Or otherwise, we can use CLI directly:
  + Step 1: Prepare dependency environment (requirements.txt) and zip this project

A screenshot of a computer

Description automatically generated

Figure 23.Prepare environment and zip project

* + Step 2: Deploy using *az webapp deploy*. I sent the zip file over the pre-configured webapp by pre-configured resource group *rg-swe40006*.

az webapp deploy \

--resource-group rg-swe40006 \

--name swe40006-python-calculator \

--src-path swe40006-python-calculator.zip \

--type zip

A screenshot of a computer

Description automatically generated

Figure 24. Deploy zipped app

**3.1.3.2 Configure startup command (FastAPI on App Service)**

* Azure Portal → *swe40006-python-calculator* → Configuration → General Settings → Startup Command:

gunicorn -w 1 -k uvicorn.workers.UvicornWorker main:app

* Or using CLI. I configured main.py as my startup file that is rendered by *gunicorn* (with 1 uvicorn worker):

az webapp config set \

--resource-group rg-swe40006 \

--name swe40006-python-calculator \

--startup-file "gunicorn -w 1 -k uvicorn.workers.UvicornWorker main:app"

A screenshot of a computer program

Description automatically generated

Figure 25. Configured app startup

**3.1.4 Validate live app**

Initially, the app isn’t accessible from the public URL (<https://swe40006-python-calculator.azurewebsites.net>), I rechecked the app building status, which it persisted at starting state, then I checked with *az webapp show* command, which state that our plan (registered for this webapp) has “QuotaExceeded” status. This could be a result of over-testing from myself while using a free-tier account.



Figure 26. Webapp QuotaExceeded status

So then, I recreate another alternative plan (or I will have to wait until the next day to try on this plan again), using the same resource group, with this CLI command:

az appservice plan create \

--resource-group rg-swe40006 \

--name plan-swe40006-alt \

--sku F1 --is-linux \

--location southeastasia

And recreate another app with this new plan named *swe40006-python-calculator-2*:

az webapp create \

--resource-group rg-swe40006 \

--plan plan-swe40006-alt \

--name swe40006-python-calculator-2 \

--runtime "PYTHON:3.11"

Similarly, I re-setup the same startup command for this app:

az webapp config set \

--resource-group rg-swe40006 \

--name swe40006-python-calculator-2 \

--startup-file "gunicorn -w 1 -k uvicorn.workers.UvicornWorker main:app"

And redeploy the zip file to this app:

az webapp deploy \

--resource-group rg-swe40006 \

--name swe40006-python-calculator-2 \

--src-path swe40006-python-calculator.zip \

--type zip

The deployment is successful.

A screenshot of a computer program

Description automatically generated

Figure 27. Successful webapp deployment

We can also check the log with CLI command:

az webapp log tail -g rg-swe40006 -n swe40006-python-calculator-2

And to restart the web app with CLI command:

az webapp restart -g rg-swe40006 -n swe40006-python-calculator-2

**3.1.5 Inspect app from public URL**

We can inspect the app from: <https://swe40006-python-calculator-2.azurewebsites.net/docs>

A screenshot of a computer

Description automatically generated

Figure 28. Web app demo on public URL

**3.2 - Credit: Deploy C# App to Azure**

**3.1.0 Installations**