Universiti Kuala Lumpur
Where Knowledge is Applied

ASSESSMENT COVERSHEET

(STUDENTS: Fill in all sections)

Attach this coversheet as the cover for your submission. All sections need to be completed.

For online submission, attached this document as pdf or in MS Words.

Section A: Submission Details

Programme	:	BACHELOR IN INFORMATION TECHNOLOGY (HONS) (INTERNET OF THINGS)
Course Code & Name	:	IIB43203 - CLOUD COMPUTING
Course Lecturer(s)	:	Megat Norulazmi Megat Mohamed Noor
Type of Submission	:	**GROUP PROJECT**
Penalties	:	• 5% will be deducted per day to a maximum of four (4) working days, after which the submission will **not** be accepted. • Plagiarised work is an Academic Offence in University Rules & Regulations and will be penalised accordingly.

Section B: Academic Integrity

Tick (√) each box below if you agree:

√	I/You have read and understood the UniKLs' policy on Plagiarism in University Rules & Regulations.
√	This assignment is own work, unless indicated with proper referencing.
√	This assignment not submitted and not published previously.
√	This submission follows the requirements stated in the course.

Section C: Submission Receipt

Office Receipt of Submission

Date & Time of Submission (by student)	Student Name(s) (by student)	Student ID(s) (by student)
JUNE 2025	SHAIFUL HAZIQ HIDZMI BIN SAIPOL AZMI	52224123153
	SHARIFAH ARLEENA BARAKBAH BINTI SYED ASWAD	52224123403
	NUR A'QILAH BINTI SUHAIMI	52224123216
	NIK NUR HUSNA BINTI NIK AHMED RAZANI	52224123376

CLOUD COMPUTING GROUP PROJECT

Youtube link: https://youtu.be/ZtHtAwZtbT0

Introduction

This lab exercise explores the deployment and scaling of a web application using Microsoft Azure. The objective is to gain hands-on experience in creating web apps, configuring deployment using Git, managing deployment slots, enabling autoswap, and performing performance tests. The exercise demonstrates best practices in application lifecycle management and cloud performance optimization.

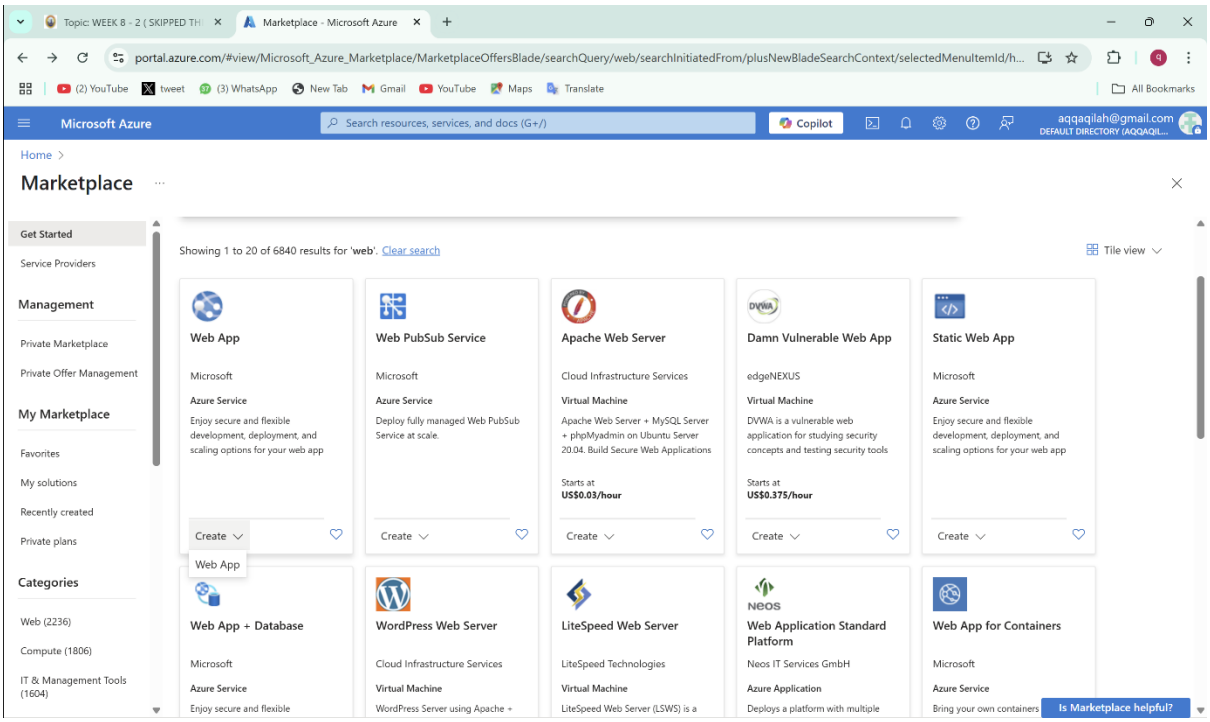
1. Create deployment slots and deploy a web app by using deployment slots

Procedure

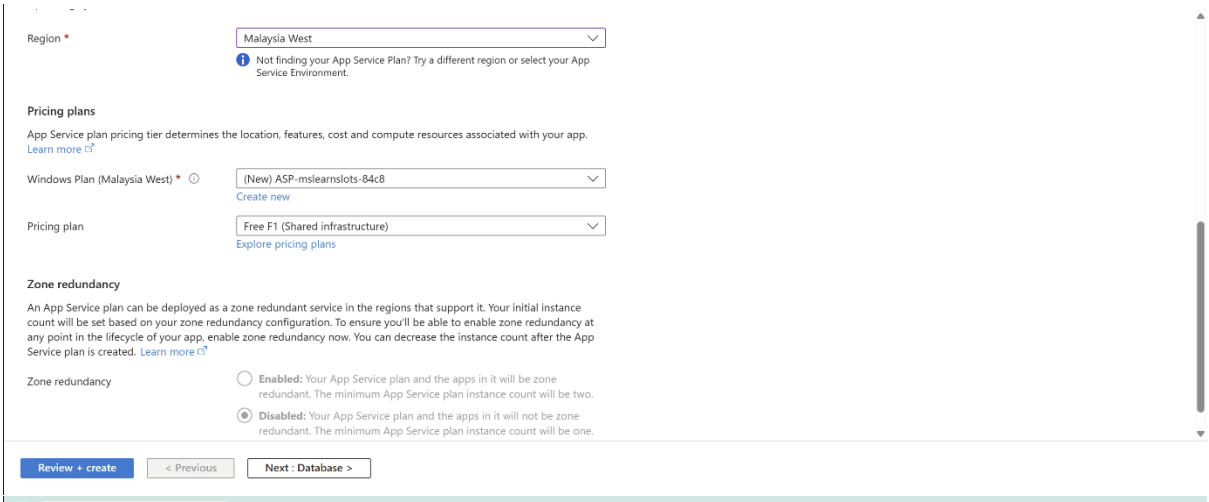
Step 1: Create a web app

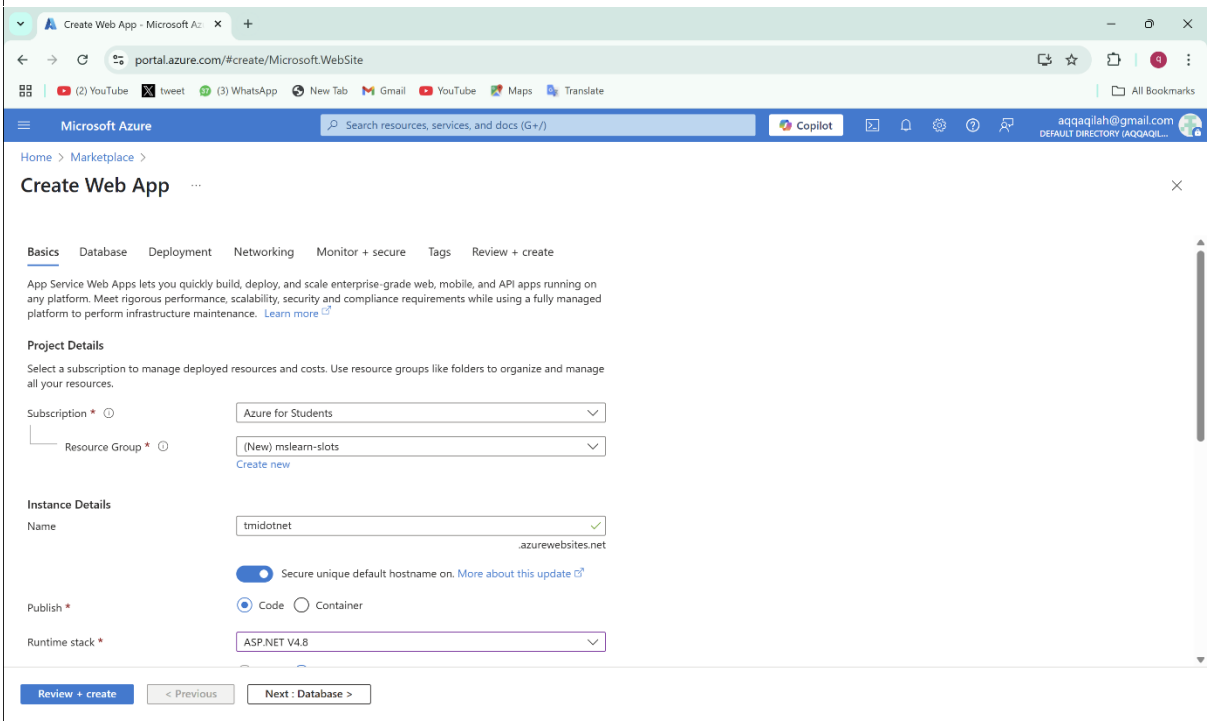
Start by creating a new web app resource in the Azure portal.

1. Sign in to the Azure portal.
2. On the Azure portal menu or from the **Home** page, select **Create a resource**. The **Create a resource** pane appears.
3. In the left menu, select **Web**, and then search for and select **Web App**. The **Create Web App** pane appears.



4. These are what on basic tab

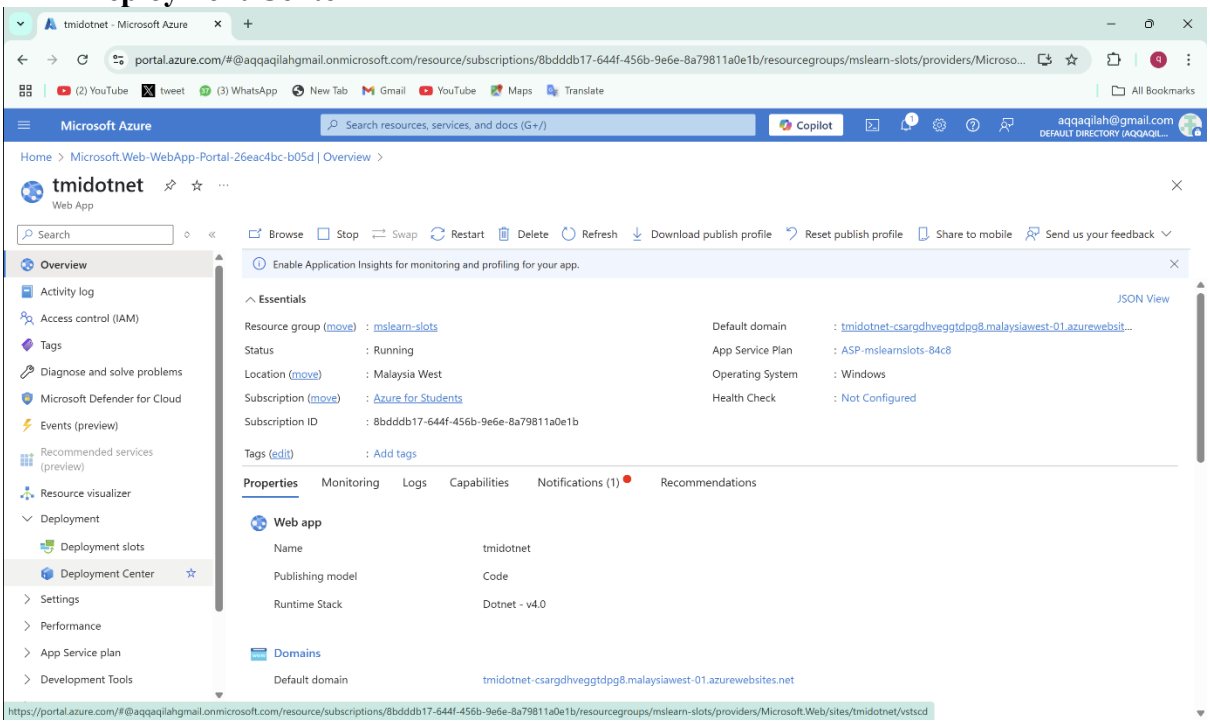




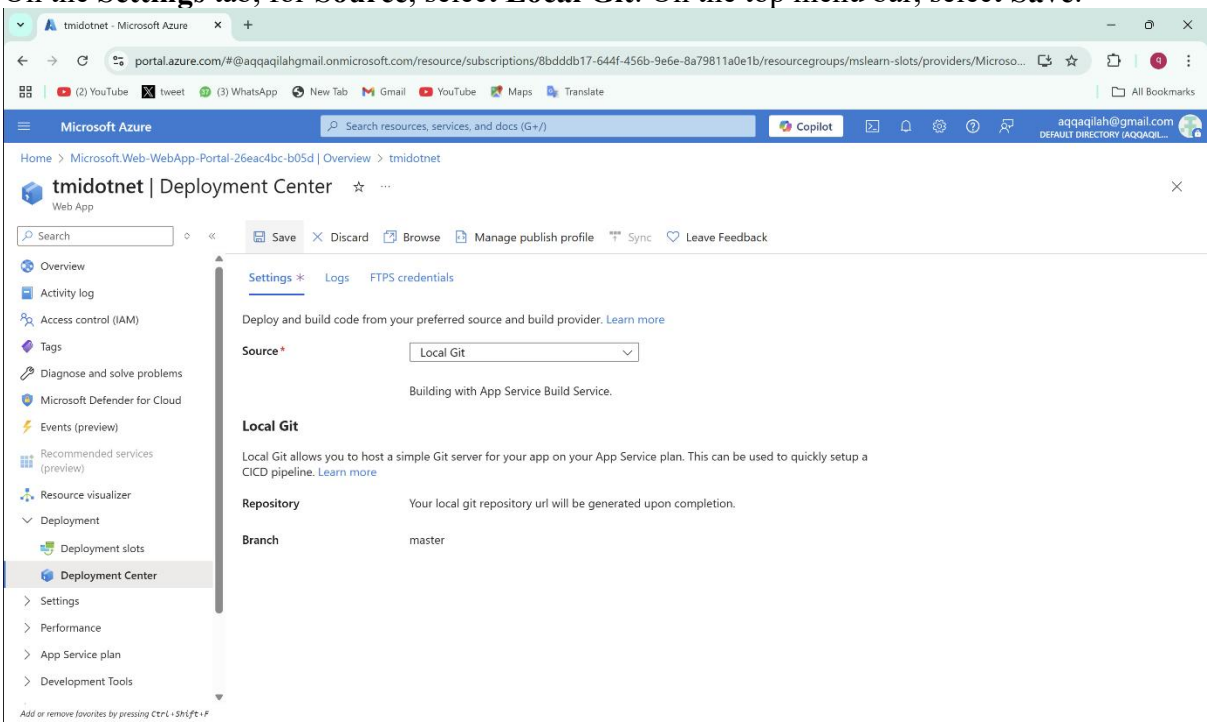
5. Select Review + create and when the content is validated, select Create. Wait while Azure creates the web app.The pricing plan was later changed from F1 to S1

Step 2: Configure git deployment

1. On your web app App Service page, in the left menu, under **Deployment**, select **Deployment Center**.

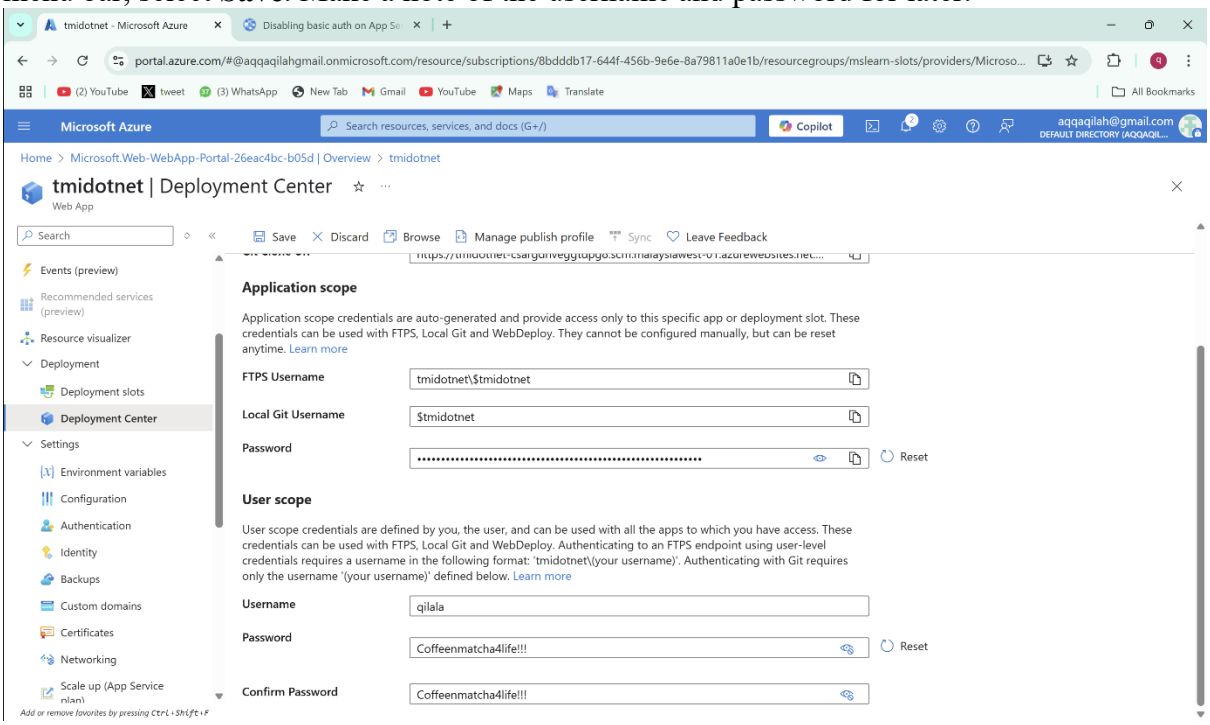


2. On the **Settings** tab, for **Source**, select **Local Git**. On the top menu bar, select **Save**.



3. On the resulting **Deployment Center** pane, select the **Local Git/FTPS credentials** tab.

4. Under **User scope**, enter a username and password of your choice, and in the top menu bar, select **Save**. Make a note of the username and password for later.



Step 3: Configure the git client and clone the web app source code

1. In the Cloud Shell tool bar, ensure that Bash is selected.
2. Copy and paste your edited code into the Cloud Shell and run it.

   git config --global user.name "your-username"

   git config --global user.email "your-email-address"

3. Create a folder for the source code. Run the following commands.

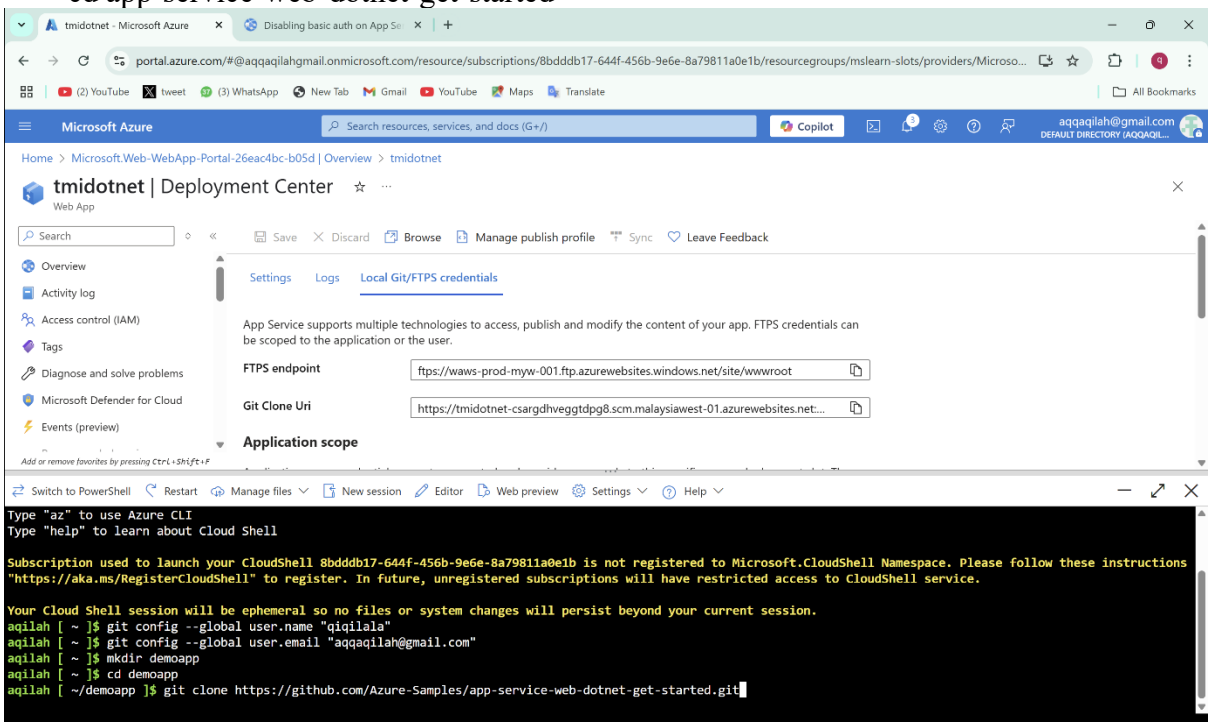
   mkdir demoapp

   cd demoapp

4. Clone the source for the web app. Run the following commands.

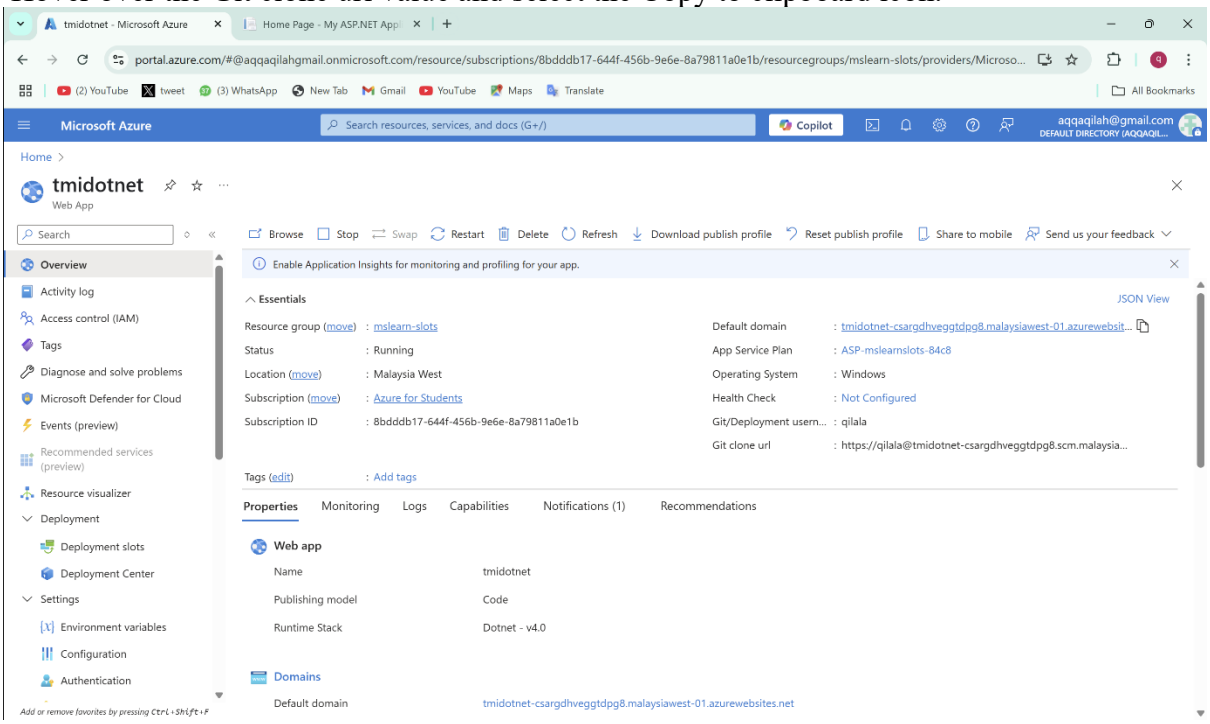
   git clone https://github.com/Azure-Samples/app-service-web-dotnet-get-started.git

   cd app-service-web-dotnet-get-started

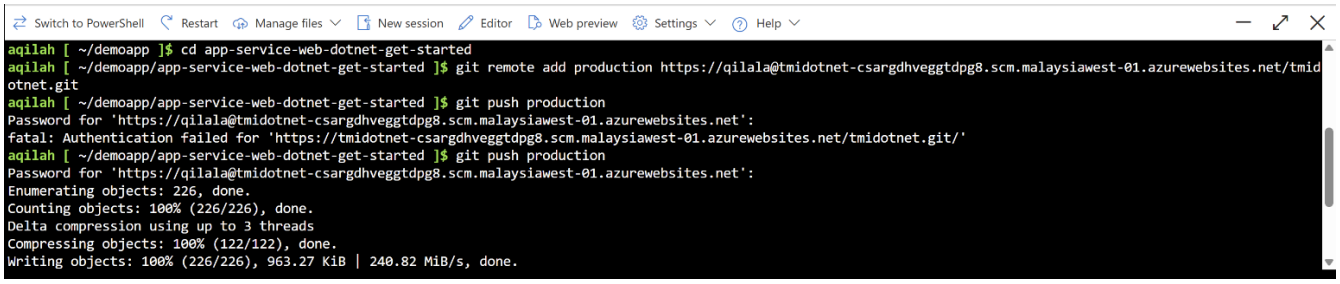


Step 4: Configure a git remote to deploy the app to production

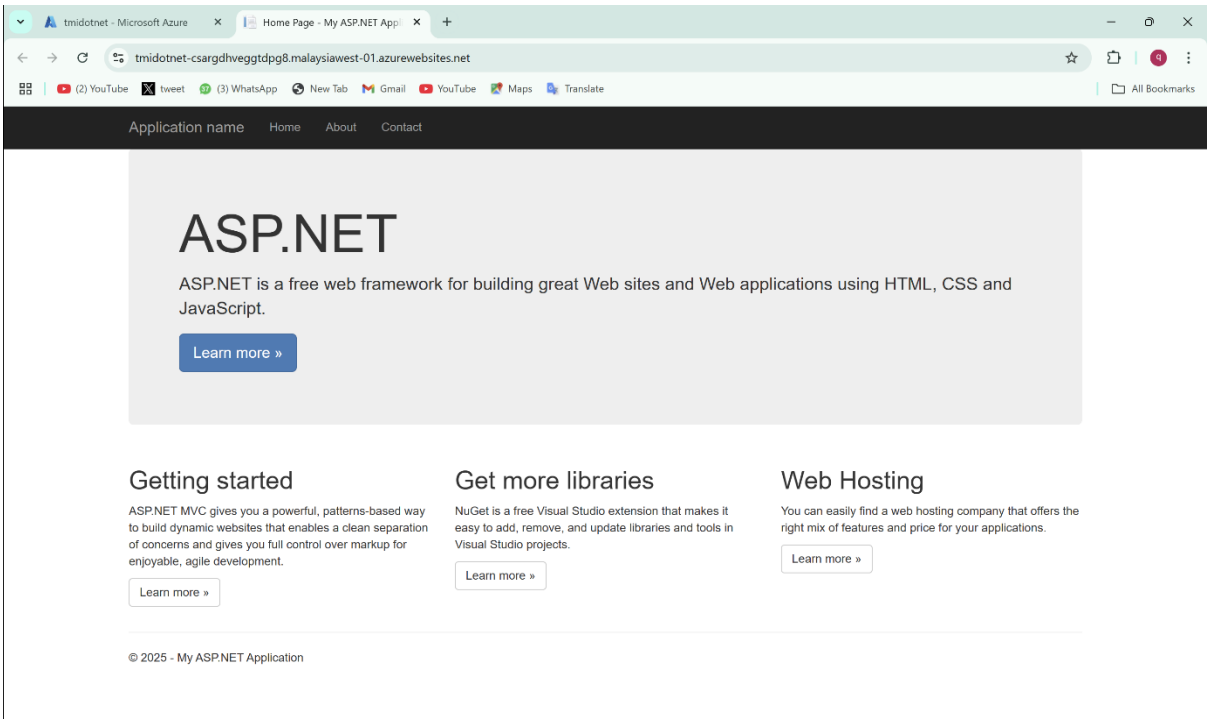
1. In the Azure portal, your web app should be active. In the left menu, select **Overview**.
2. In the Overview pane for your web app note that the **Essentials** section has a URL and a Git clone url. Hover over the **URL** and select the *Copy to clipboard* icon. Note that the URL contains your deployment name for the web app.
3. Hover over the Git clone url value and select the Copy to clipboard icon.



4. In Cloud Shell, run the following command to configure the git remote with a name "production". Replace git-clone-url with the content you copied to the clipboard from the previous step.
5. To deploy the web app to the production slot, run the following command. When you're prompted for the password, enter your deployment password you created previously.



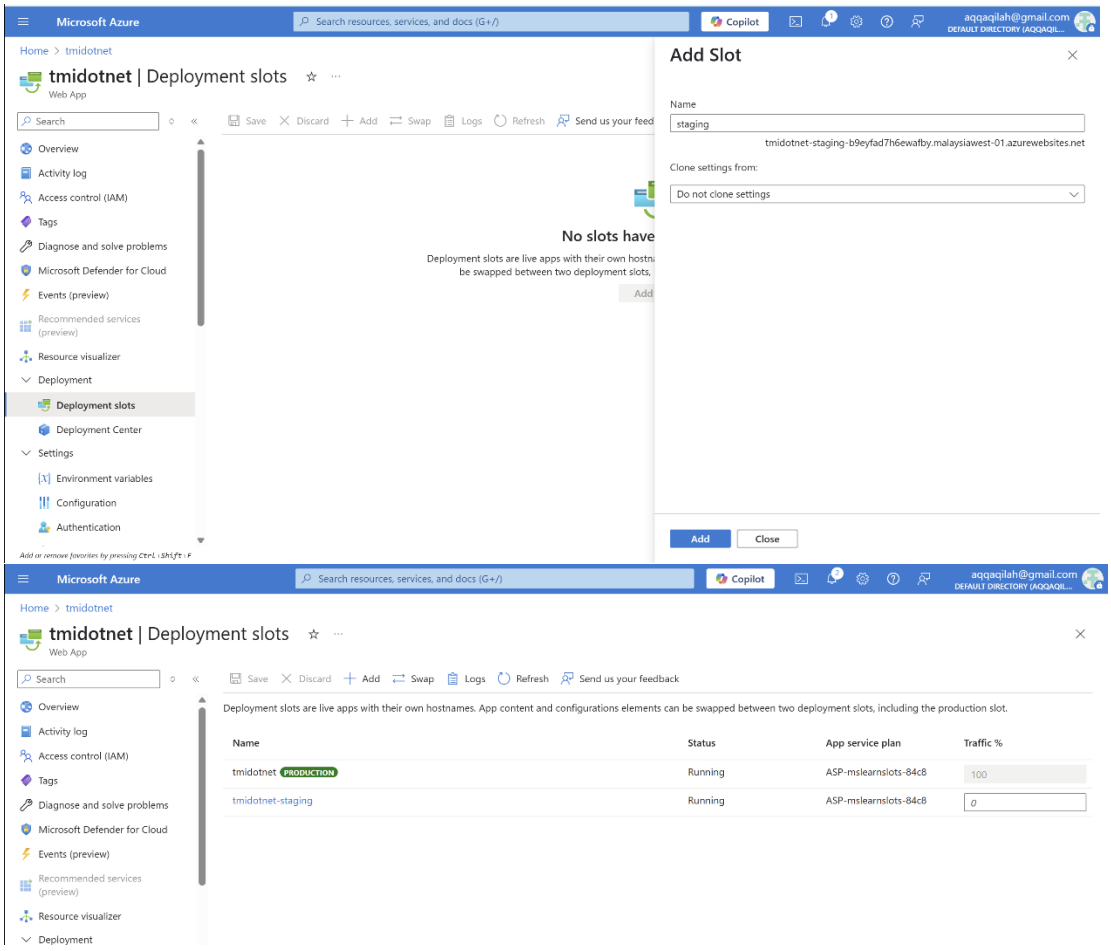
6. When the deployment finishes, in the Azure portal, go to the web app's Overview page, and then select URL. You can paste it into a browser or double-click to open the URL in a new tab.



Step 5: Create a new staging slot

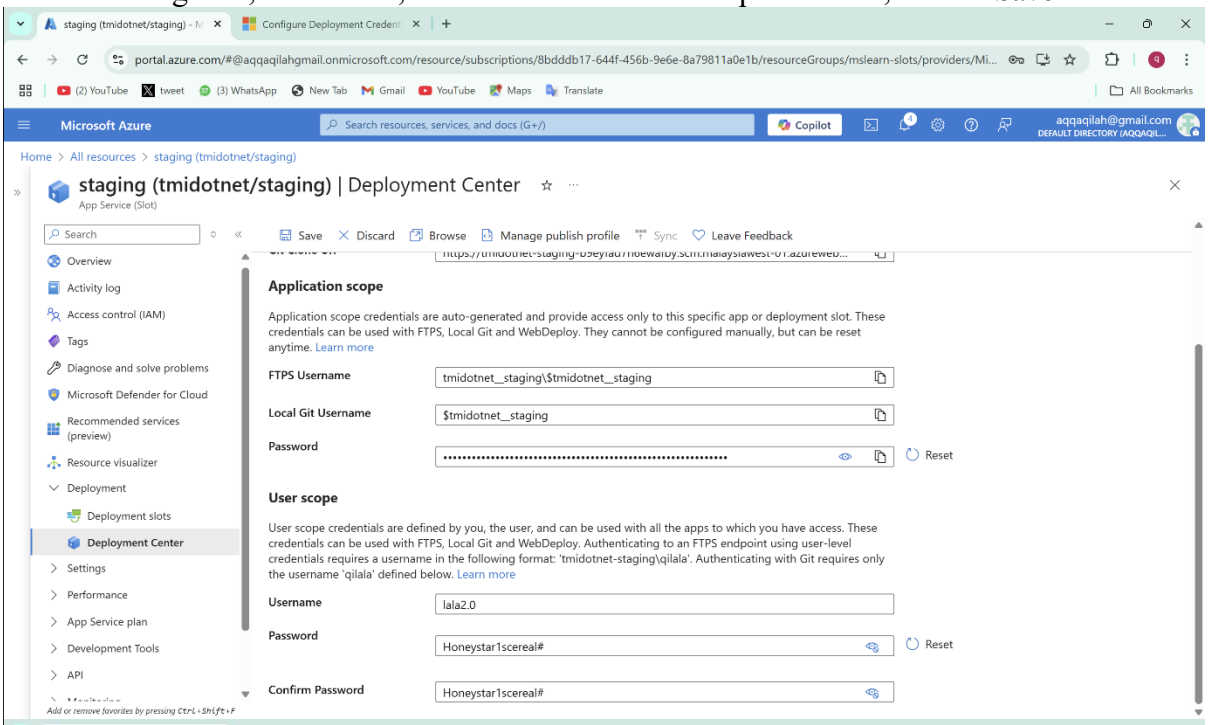
Next, you'll create a deployment slot where you can stage new versions of the web app.

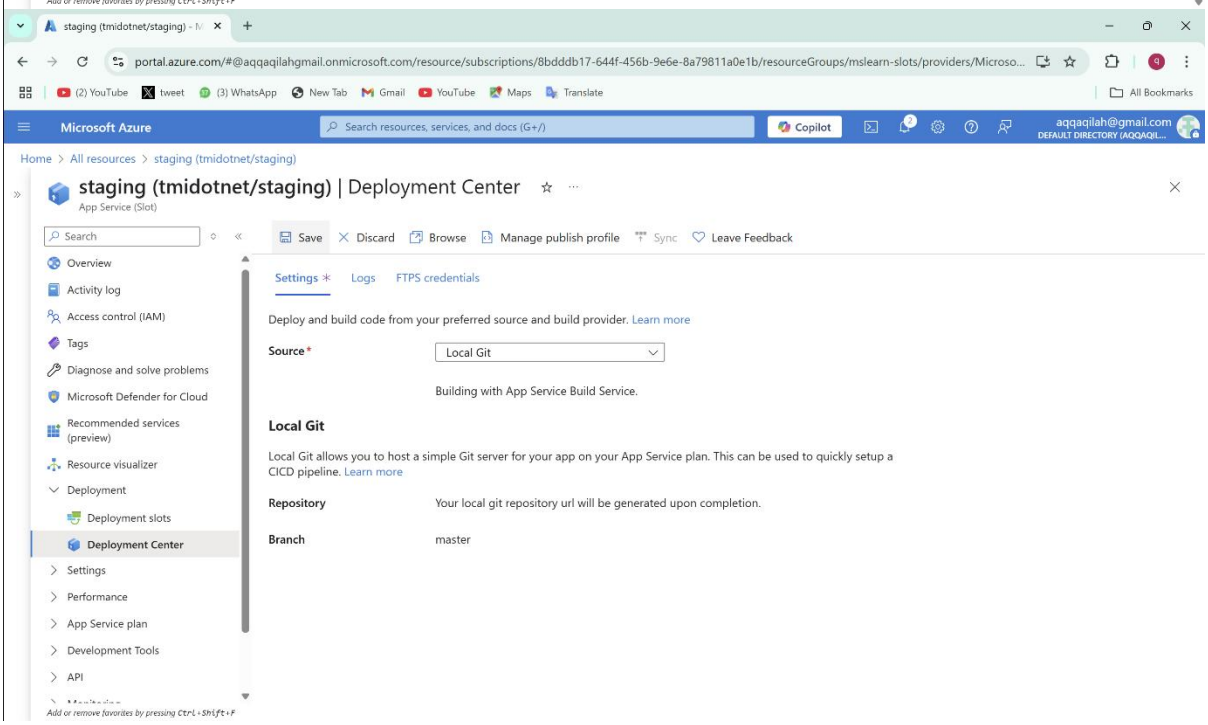
1. On the Azure portal menu or from the Home page, select All resources, and then filter by Type == App Service.
2. Select your web app. The web app App Service pane appears.
3. In the menu pane, under Deployment, select Deployment slots. The Deployment slots pane appears.
4. From the top menu bar, select Add Slot. The Add a slot pane appears.
5. In the Name field, enter Staging, accept the default for Clone settings from, and then select Add.
6. After the deployment slot is successfully created, select Close



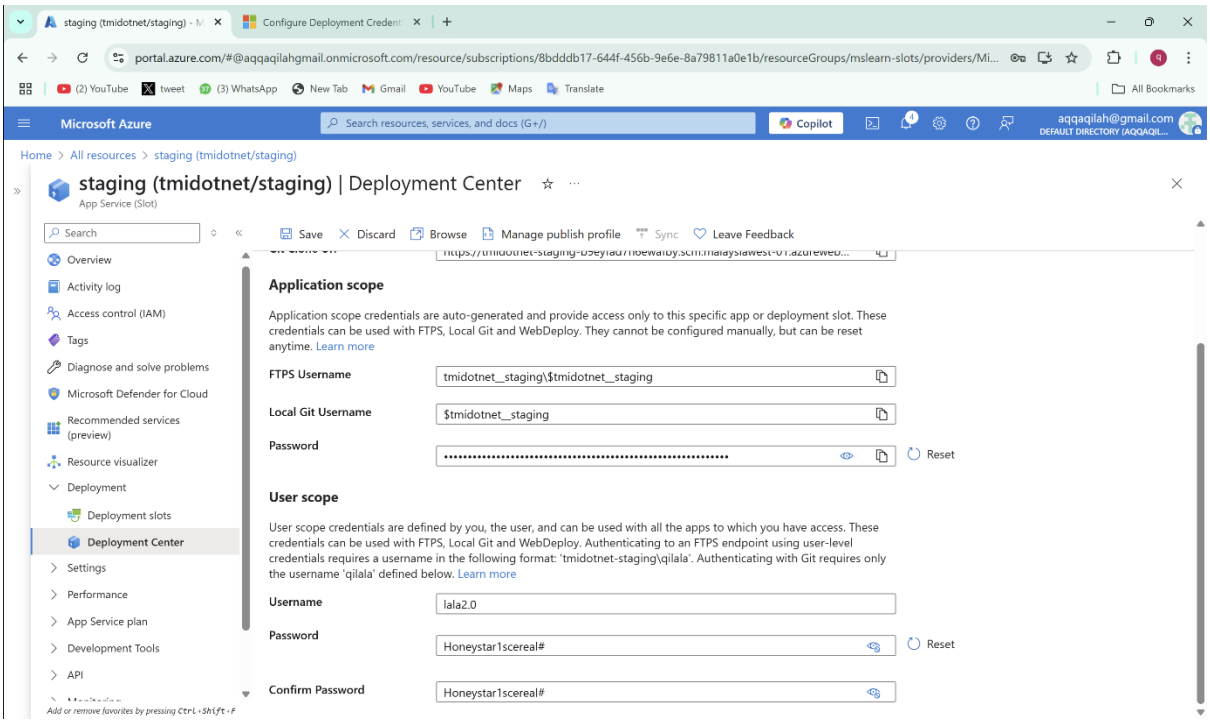
Step 6: Set up git deployment for the staging slot

1. On the Azure portal menu or from the Home page, select All resources. In the list of all resources, you can filter on *Resource group == mslearn-slots*. You will see two App Service entries. Deployment slots are represented as separate apps in the portal. Select the entry representing the staging slot to go to its Overview pane.
2. On the Overview pane, in the left menu, under Deployment, select Deployment Center.
3. On the Settings tab, for Source, select Local Git. In the top menu bar, select **Save*.





4. On the resulting Deployment Center pane, select the Local Git/FTPS credentials tab.
5. Under User scope, enter a new username and password of your choice, and from the top menu bar, select Save. Make a note of the username and password for later.



Step 7: Set up git to deploy the app to the staging slot

1. In the Azure portal, on the staging web app's **Overview** page, in the **Essentials** section, select the *Copy to clipboard* icon for the **Git clone url**. Note that the URL contains your deployment username.
2. To add the remote for the staging slot, run the following command in Cloud Shell. Replace *git-clone-uri* with the URI from the previous step.

```
aqilah [ ~/demoapp/app-service-web-dotnet-get-started ]$ git remote add staging https://lala2.0@tmidotnet-staging-b9eyfad7h6ewafby.scm.malaysiawest-01.azurewebsites.net/tmidotnet.git
```

Step 8: Modify the app source code and deploy the app to the staging slot

1. Next, make a small change to the web app, and then use git to deploy the new version to the staging slot:
2. In Cloud Shell, run the following command.

   . code

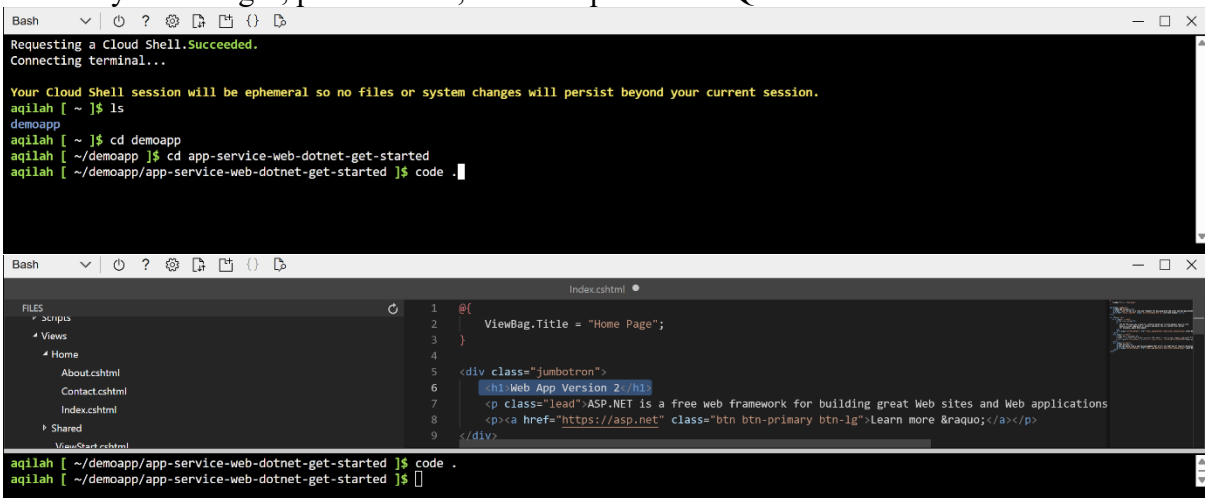
3. In the list of Files, expand demoapp > app-service-web-dotnet-get-started > aspnet-get-started > Views > Home.
4. Select Index.cshtml.
5. Locate the following code.

   <h1>ASP.NET</h1>

6. Replace that code with this code.

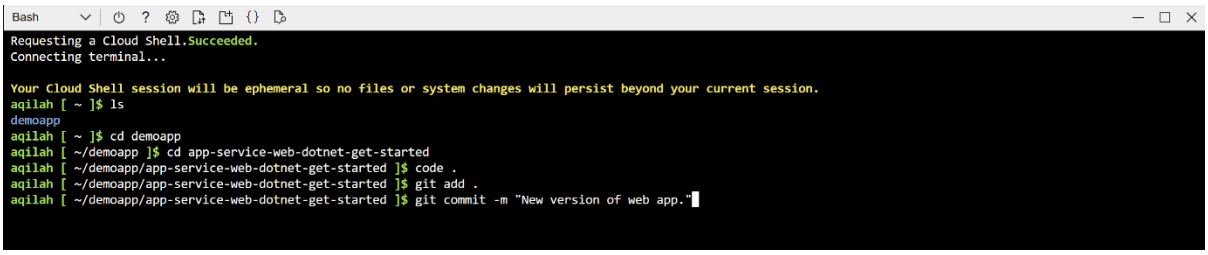
   <h1>Web App Version 2</h1>

7. To save your changes, press Ctrl+S, and then press Ctrl+Q to close the editor.



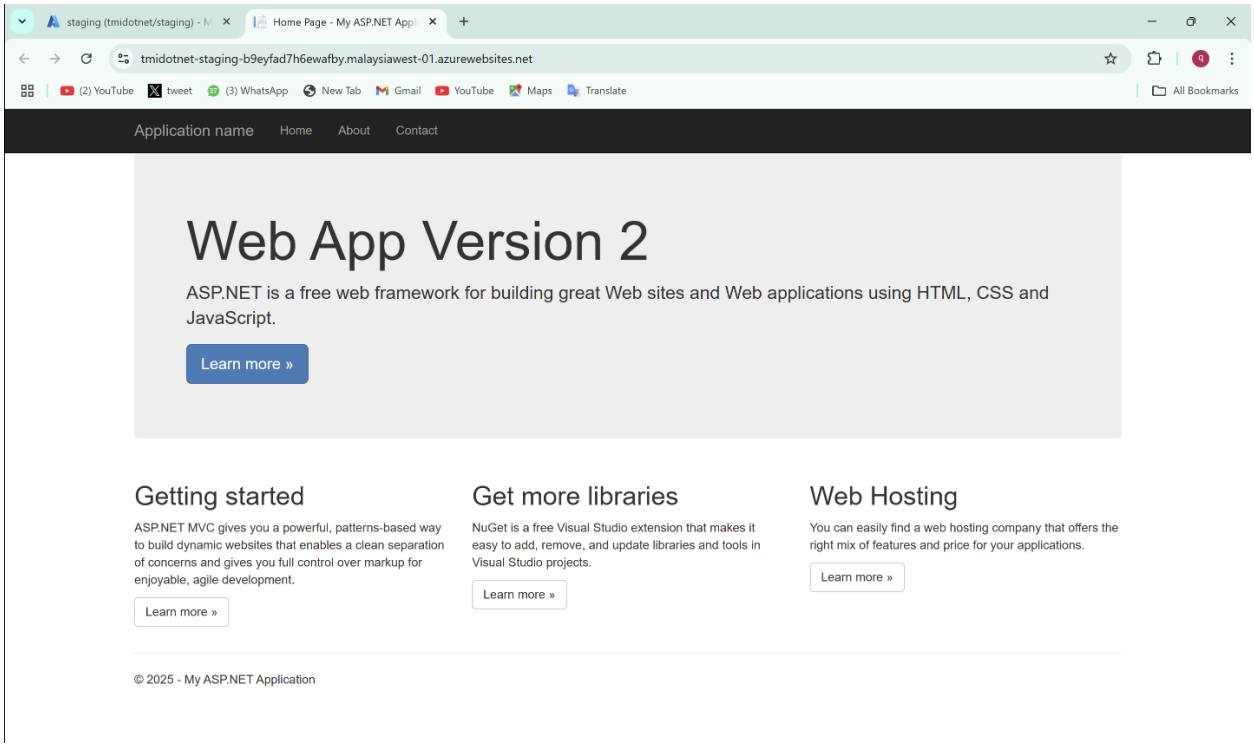
8. In Cloud Shell, run the following commands to commit the new version of the app to git, and deploy it to the staging slot. When prompted, enter your deployment password.

    cd /demoapp/app-service-web-dotnet-get-started
    git add .
    git commit -m "New version of web app."
    git push staging



Step 9: Browse the staging slot

Now you can view the new version of the web app by browsing to the staging deployment slot's URL. In the Azure portal, go to the Overview page for the staging slot. In the top menu bar, select Browse. The new version of the web app appears in a browser tab.



Step 10: Deploy a web app by using deployment slots

When you're ready to swap two slots, make sure you've applied the correct configuration to the swapped slots.

Suppose you've finished testing version 2 of your social media web app. Now, you want to deploy that version to production. You want to further streamline deployment by automatically swapping future versions of the app.

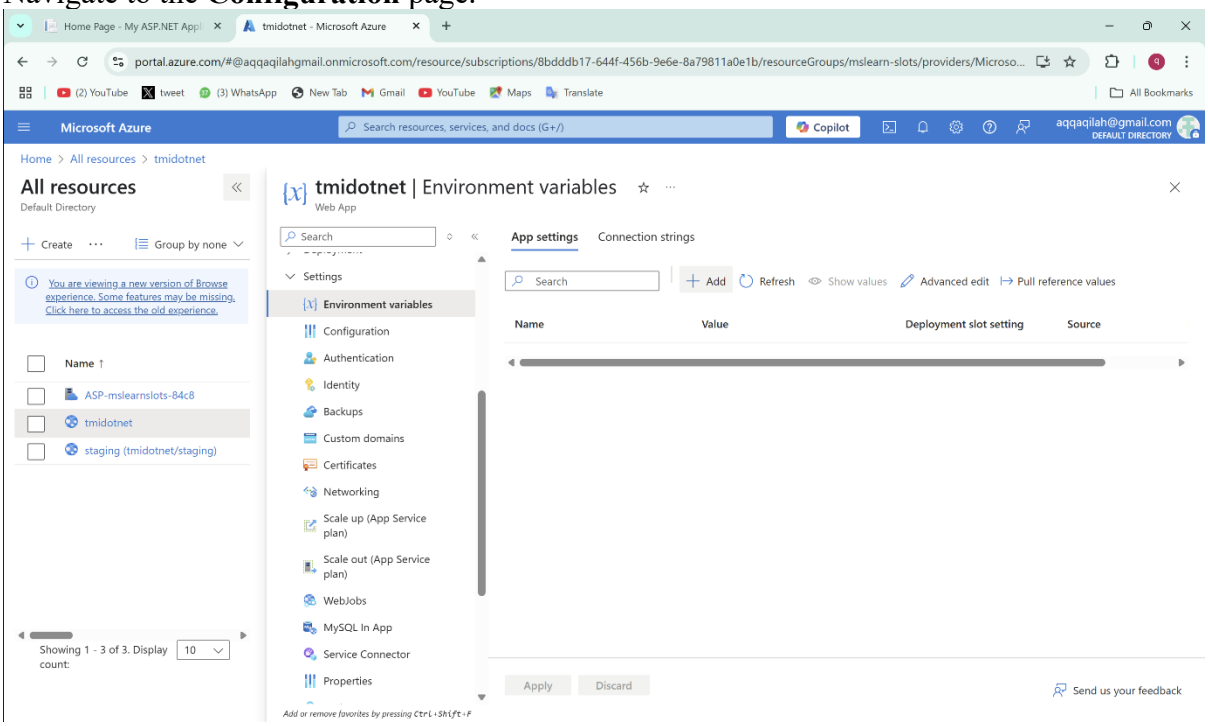
Here, you'll learn how to swap manually and automatically.

Step 11: Configure a slot setting

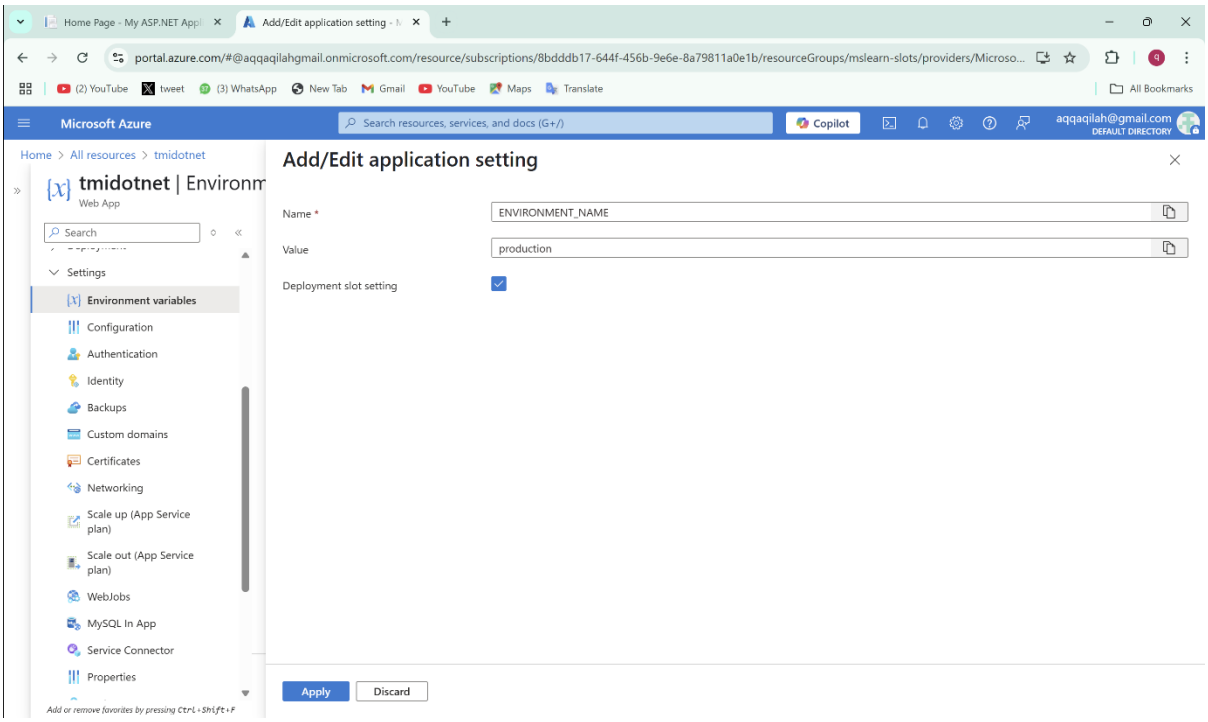
Before you deploy version 2 of the web app, configure a slot setting. The settings you'll configure here won't affect your demo app. The purpose of this exercise is just to see how the configurations work when you swap slots.

To configure slot settings:

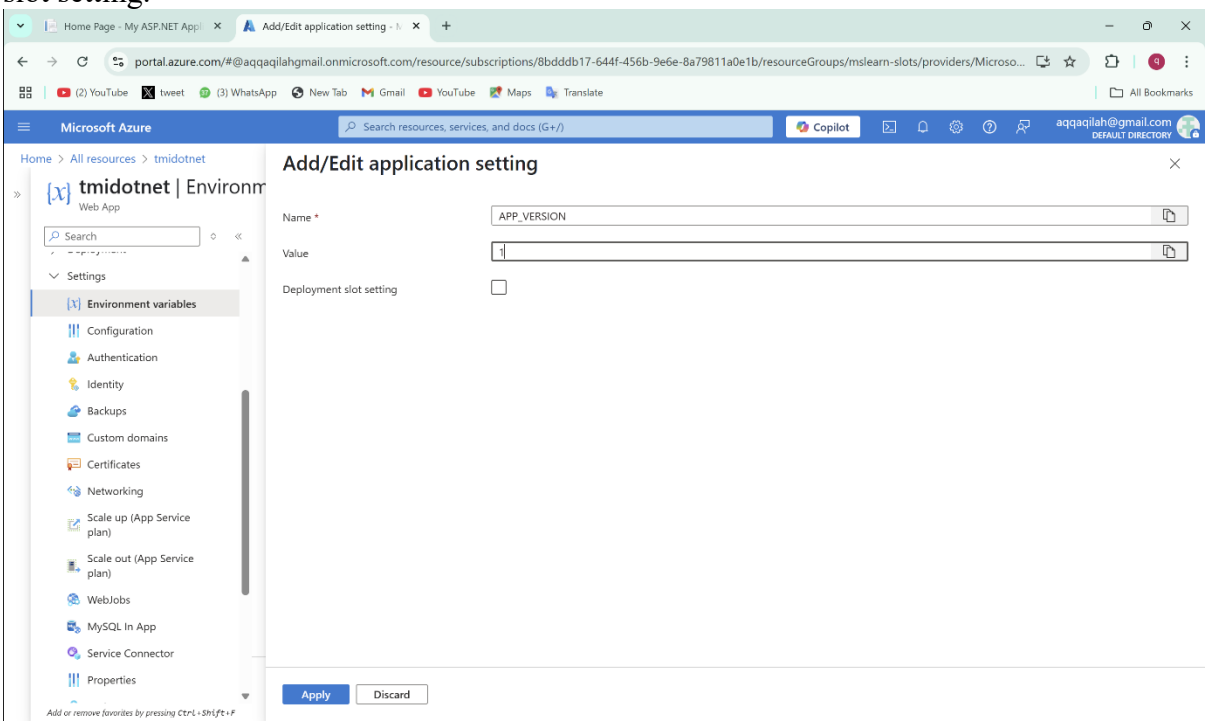
1. From the **All resources** view in the Azure portal, navigate to the **Overview** page of the production slot of the web app (this is the main webapp).
2. Navigate to the **Configuration** page.



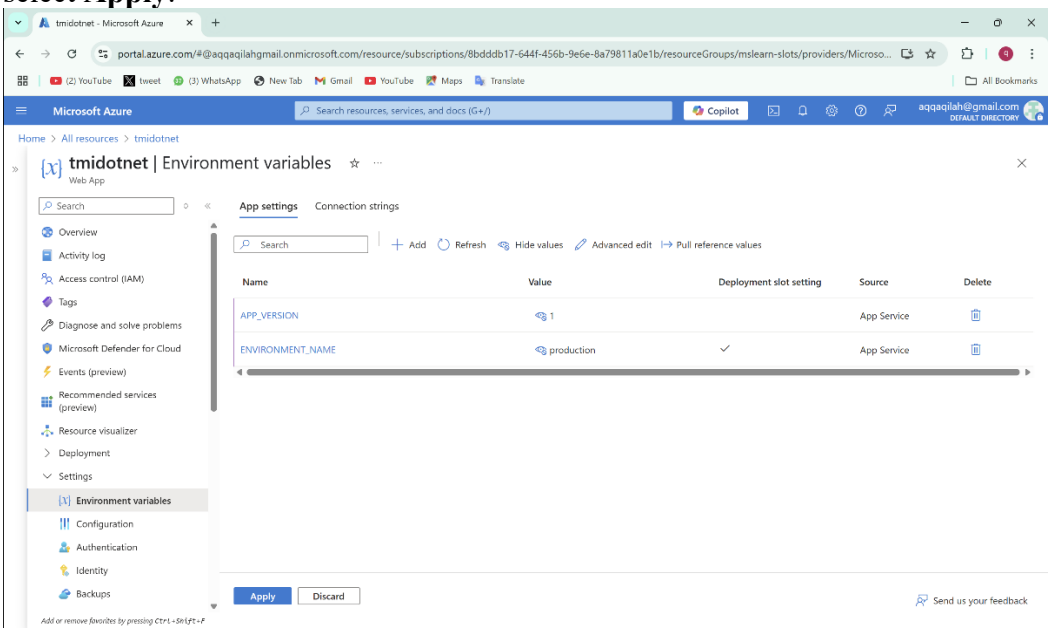
3. Select **New application setting**. Add a new setting with the name **ENVIRONMENT_NAME** and a value of **production**. Check the **deployment slot setting** box to make this a slot setting.



4. Add another setting called **APP_VERSION**, and enter the value **1**. *Don't* make this a slot setting.

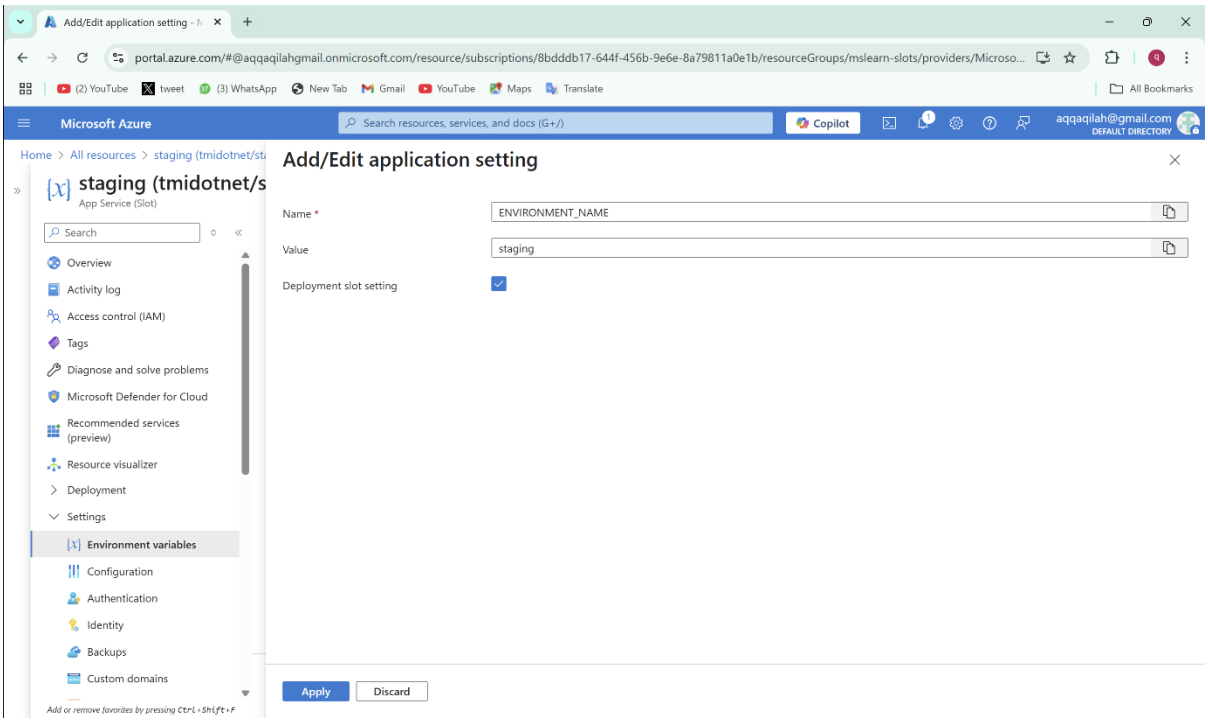


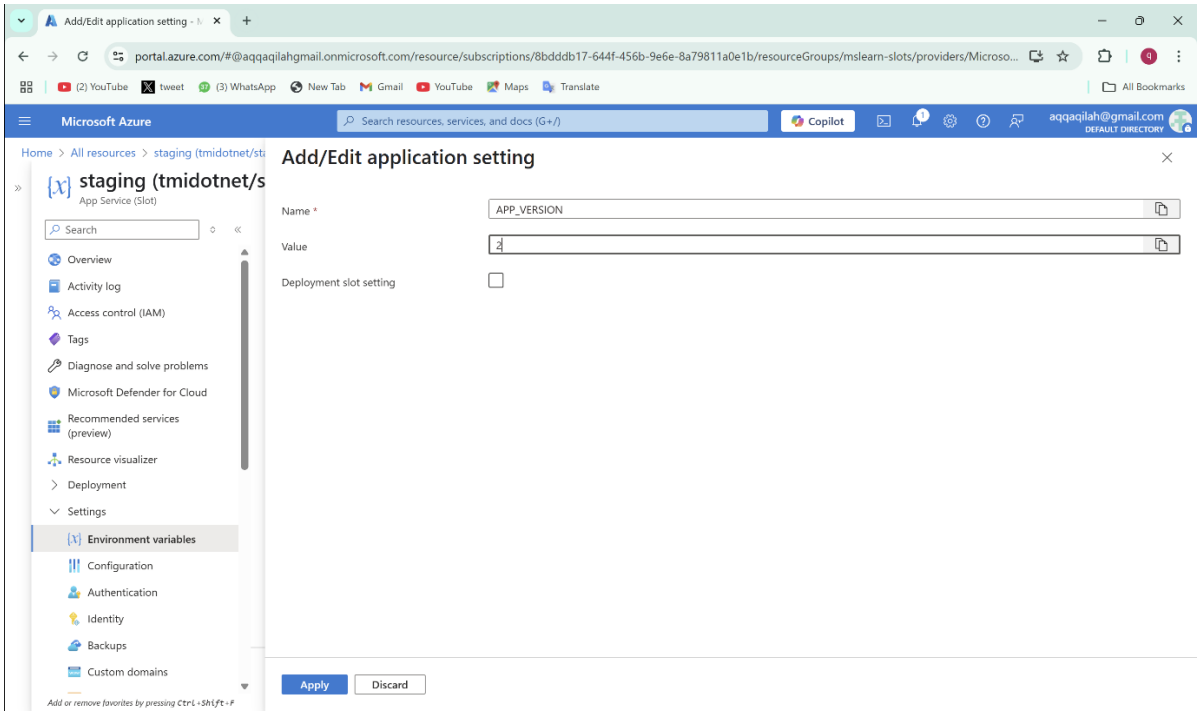
5. select **Apply**.

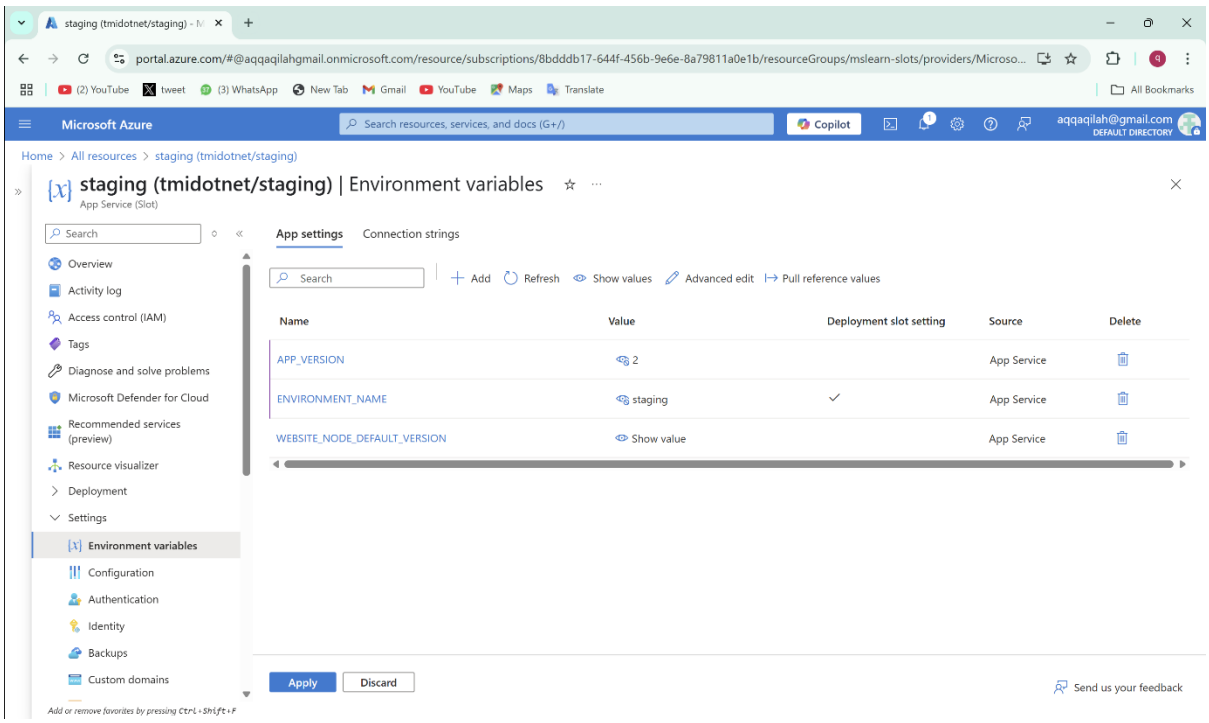


6. Repeat the preceding steps on the **Staging** slot, but use the following values:

Name	Value	Deployment slot setting
ENVIRONMENT_NAME	staging	Yes
APP_VERSION	2	No



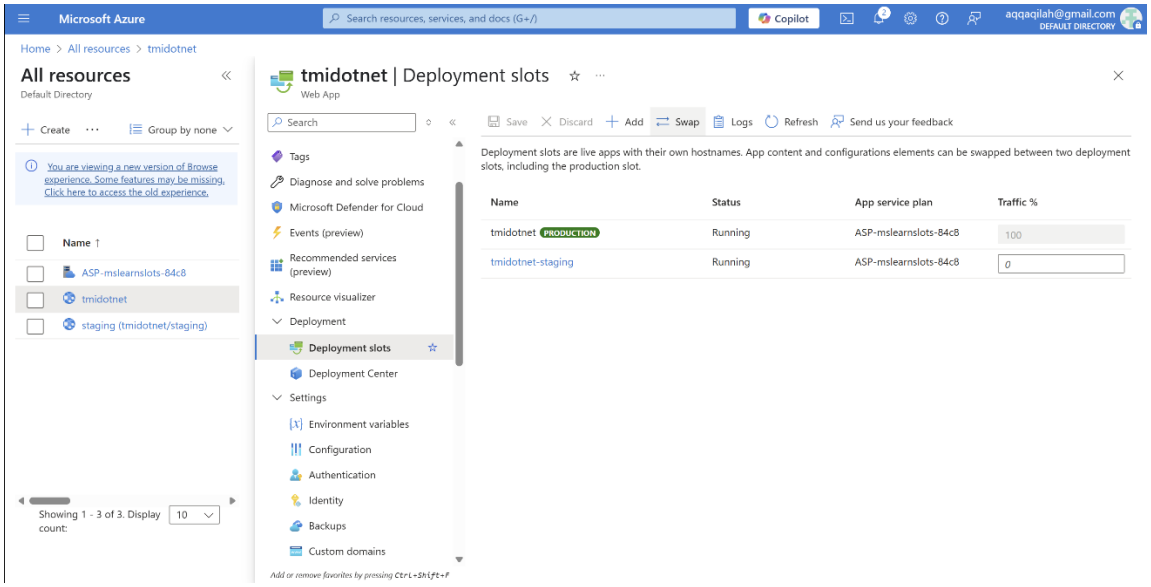

Add/Edit application setting
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
Home > All resources > staging (tmidotnet/sta
staging (tmidotnet/s
App Service (Slot)
Search
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Microsoft Defender for Cloud
Recommended services (preview)
Resource visualizer
Deployment
Settings
Environment variables
Configuration
Authentication
Identity
Backups
Custom domains
Add/Edit application setting
Name *
APP_VERSION
Value
2
Deployment slot setting
Apply
Discard


staging (tmidotnet/staging)
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
Home > All resources > staging (tmidotnet/staging)
staging (tmidotnet/staging) | Environment variables
App Service (Slot)
App settings
Connection strings
Add
Refresh
Show values
Advanced edit
Pull reference values
Name
Value
Deployment slot setting
Source
Delete
APP_VERSION
2
App Service
ENVIRONMENT_NAME
staging
App Service
WEBSITE_NODE_DEFAULT_VERSION
Show value
App Service
Apply
Discard
Send us your feedback

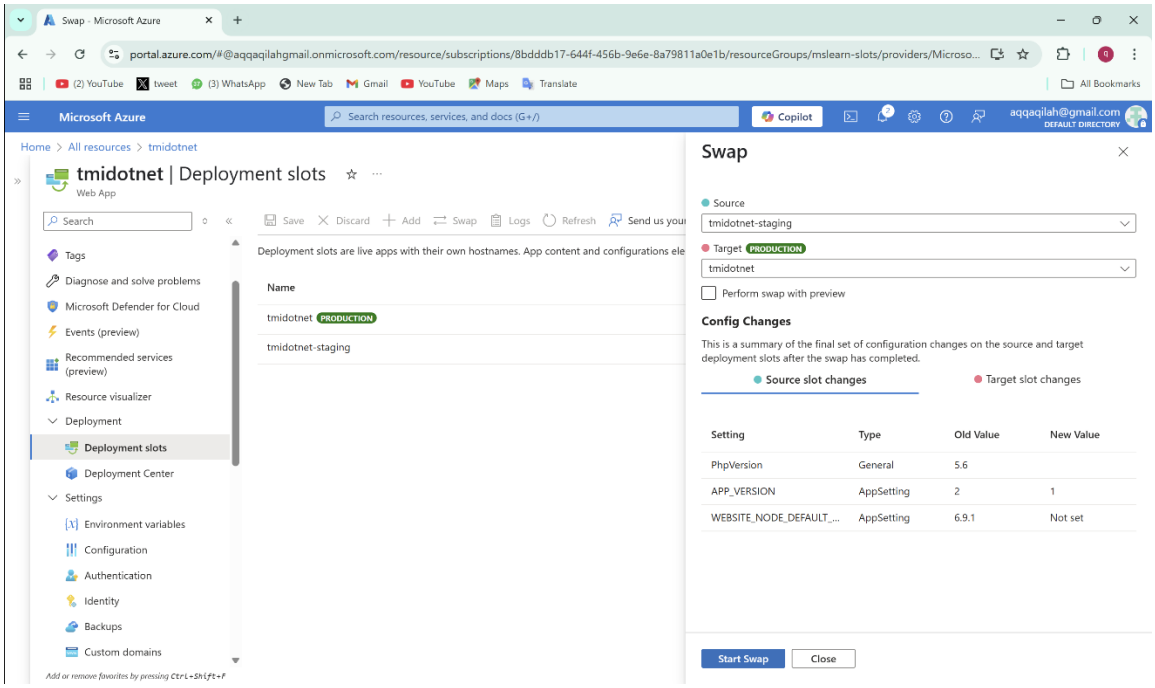
Step 12: Swap the slots

Now that you've tested version 2 of the web app in the staging slot, you can deploy it by swapping the slots. Follow these steps:

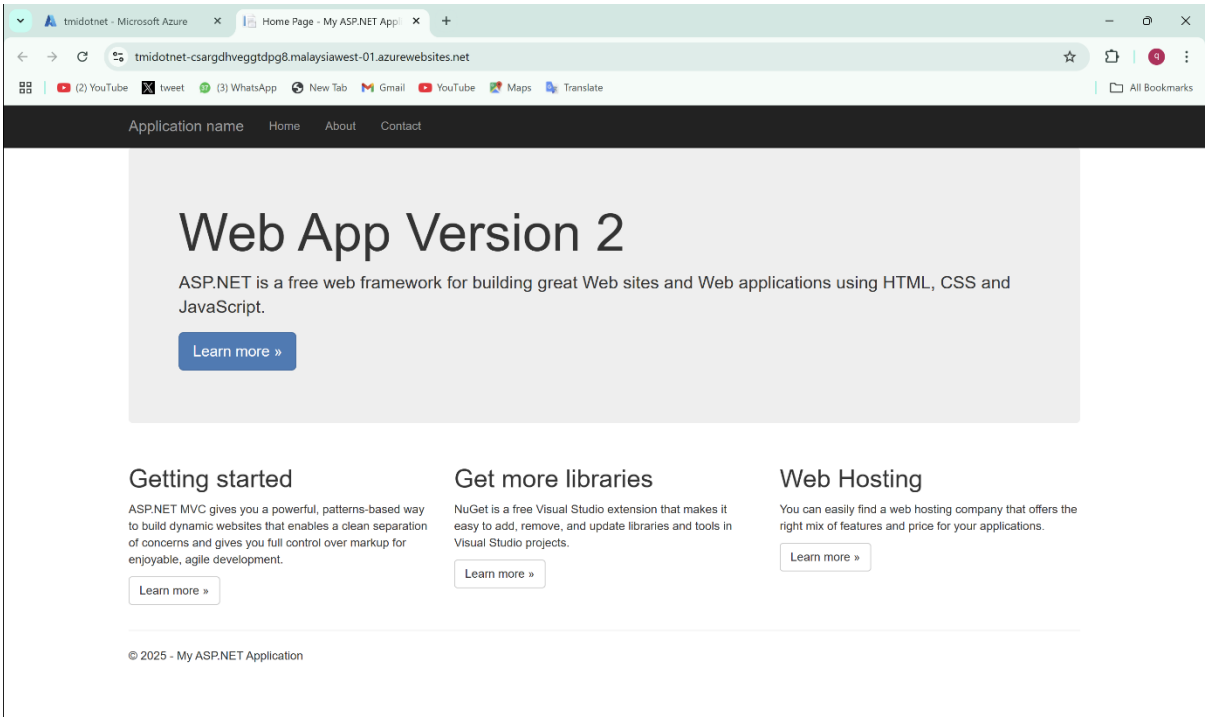
1. To make sure you're configuring the production slot, select **All resources**, and then select the production slot of the web app.
2. In the left menu pane, under **Deployment**, select **Deployment slots** > **Swap**.



3. Make sure you're about to swap the staging and production slots. Notice how the swap will affect settings. The value of the APP_VERSION setting will be exchanged between the slots, but the value of the ENVIRONMENT slot setting won't be swapped. Select Swap



4. When the swap is complete, go to the Overview page of the production slot's web app, and select Browse. The web app appears on a new browser tab. Notice that version 2 of the web app is now in production.

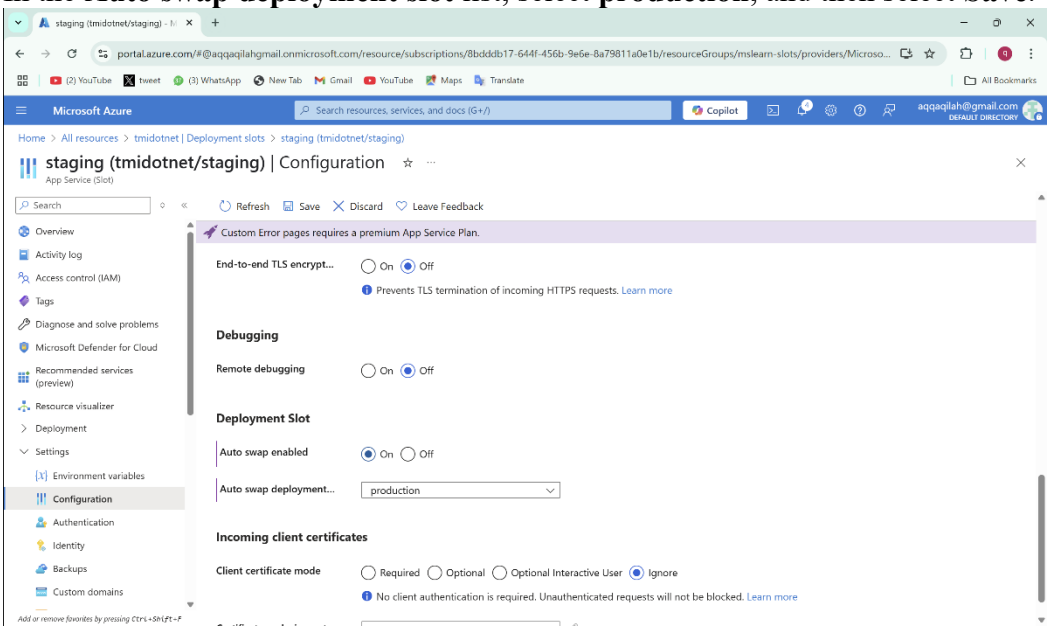


5. Close the browser tab.

Step 13: Configure auto swap for the staging slot

Suppose that now that you're using deployment slots, you want to enable continuous deployment. You'll do this by using the auto swap feature for your web app. In a system that uses auto swap, when you deploy new code to the staging slot, Azure automatically warms it up and deploys it to production by swapping the staging and production slots. To configure auto swap, follow these steps:

1. Go to the **Configuration** page of the staging slot's web app, and navigate to the **General settings** tab.
2. Set **Auto swap enabled** to **On**.
3. In the **Auto swap deployment slot** list, select **production**, and then select **Save**.



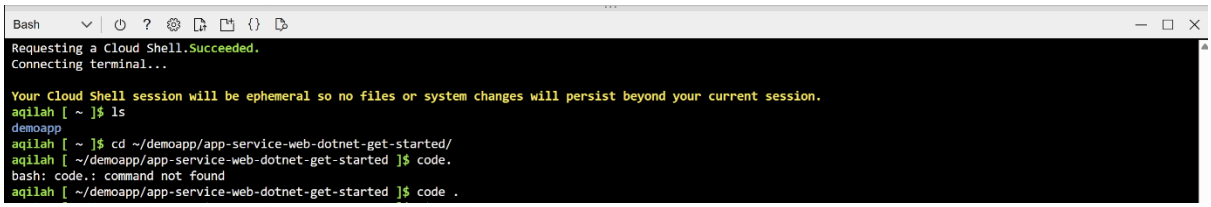
Step 14: Deploy new code and auto swap it into production

Now, you'll modify the code to create version 3 of the web app. When you deploy it to the staging slot, you'll see an auto swap in action. Follow these steps:

1. On the right side of the Cloud Shell window, restart the editor if it's not already running.

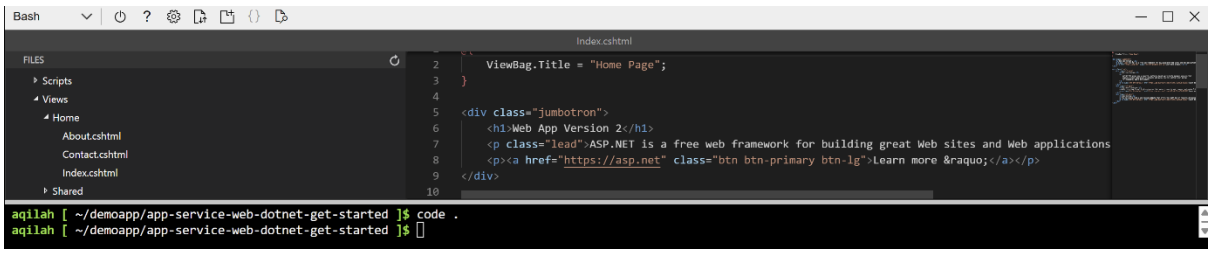
   cd ~/demoapp/app-service-web-dotnet-get-started/

   code .



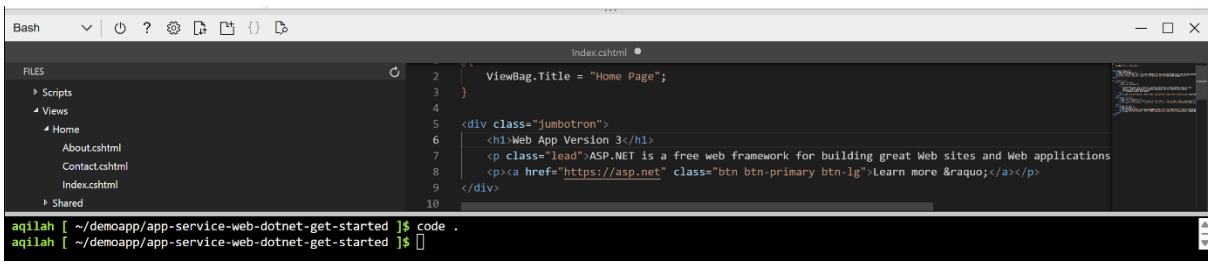
2. In the code editor, in the File list on the left, expand aspnet-get-started > Views > Home, and then select Index.cshtml.
3. Locate the following code.

   <h1>Web App Version 2</h1>



4. Replace that code with this code.

   <h1>Web App Version 3</h1>

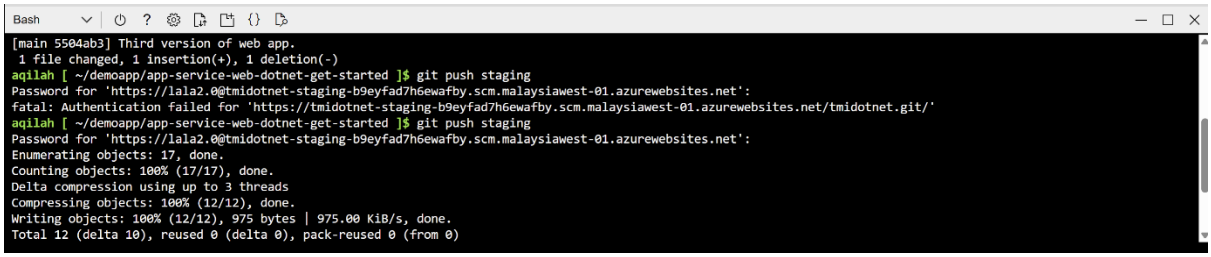


5. To save your changes, press Ctrl+S.
6. In Cloud Shell, enter the following commands. Enter your deployment password when you're prompted.

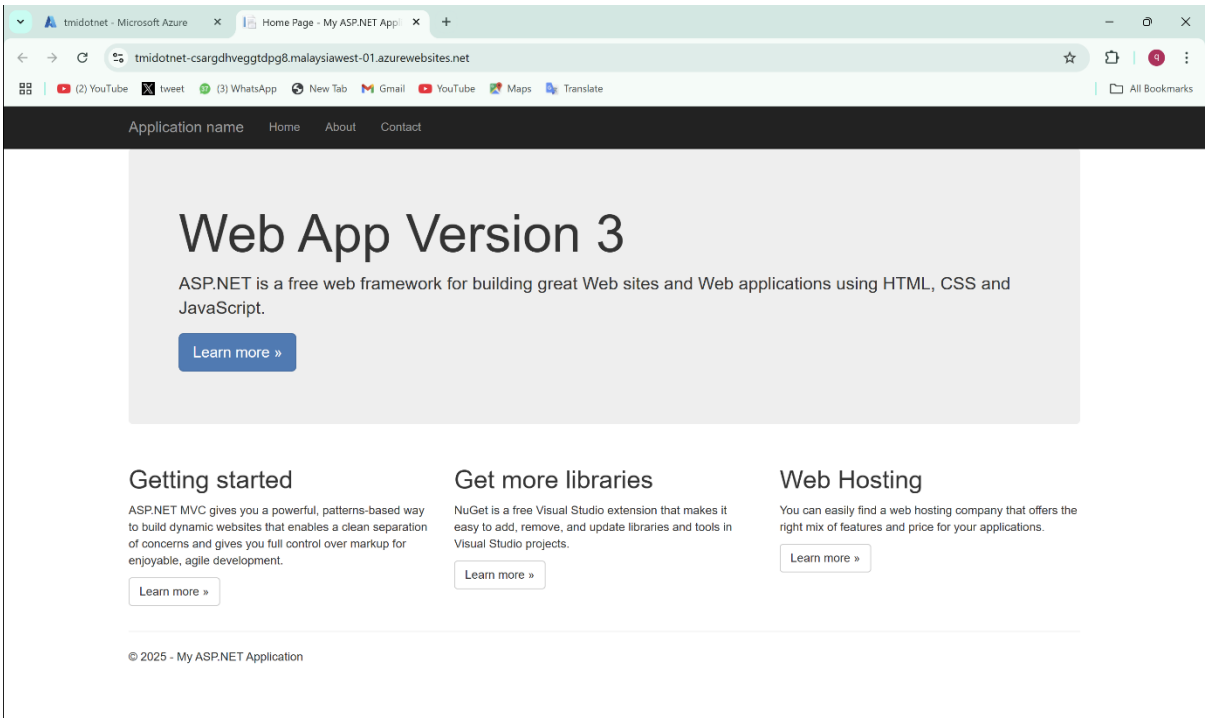
   git add .

   git commit -m "Third version of web app."

   git push staging



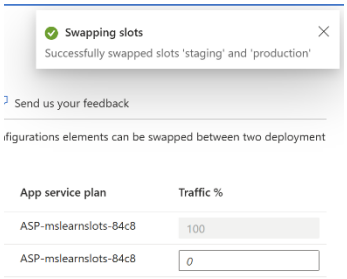
7. Wait for the deployment to finish. Near the end of the text output, you'll see a message that indicates that the deployment has requested an auto swap to the production slot.
8. In the Azure portal, navigate to the Overview page for the production slot's web app, and select Browse. The third version of the web app appears on a new browser tab. If the old version is shown, you may need to wait briefly and then refresh the page - the swap operation is atomic and occurs instantly, but it takes App Service a few moments to prepare the swap operation before it's execute



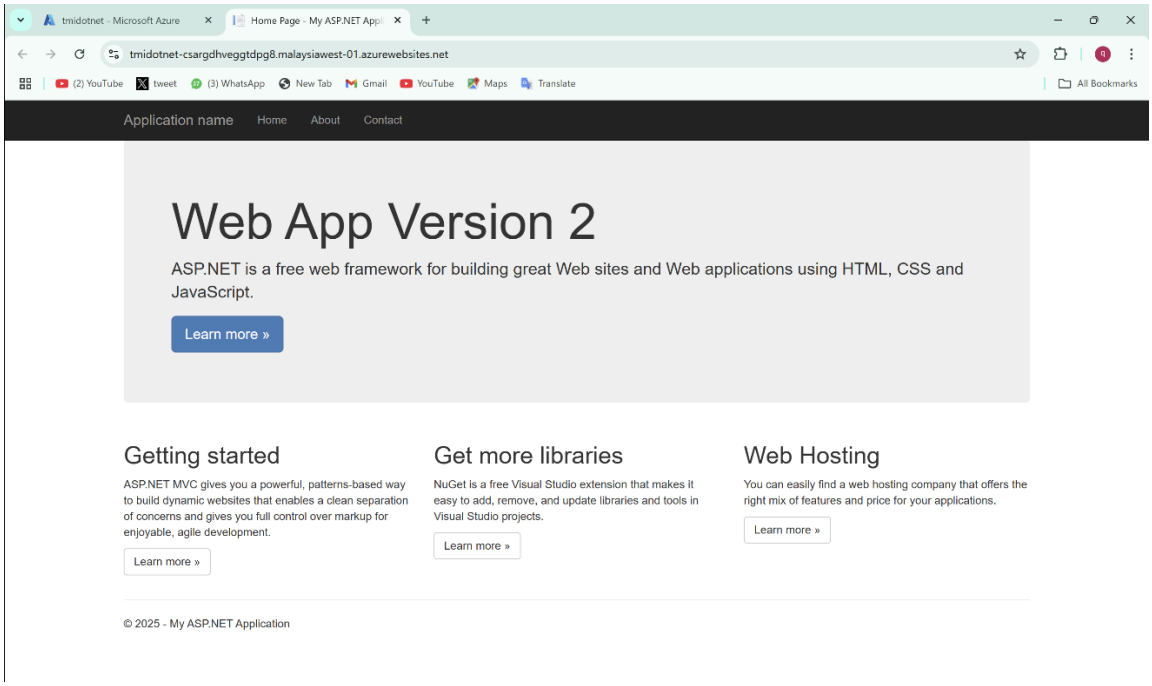
Step 15: Roll back the new version

Suppose that deploying version 3 of your app to production revealed an unexpected problem. To quickly resolve it, you can roll back to the previous version of the site by swapping the slots again.

1. Go to the **Deployment slots** page of the production slot's web app.
2. Swap the staging and production slots



3. When the swap finishes, on the **Overview** page, select **Browse** to view the app one last time. You'll see that version 2 has been redeployed to production.



2. Scale a web app manually and scale up a web app

Procedure:

Step 1: Sign in to Azure

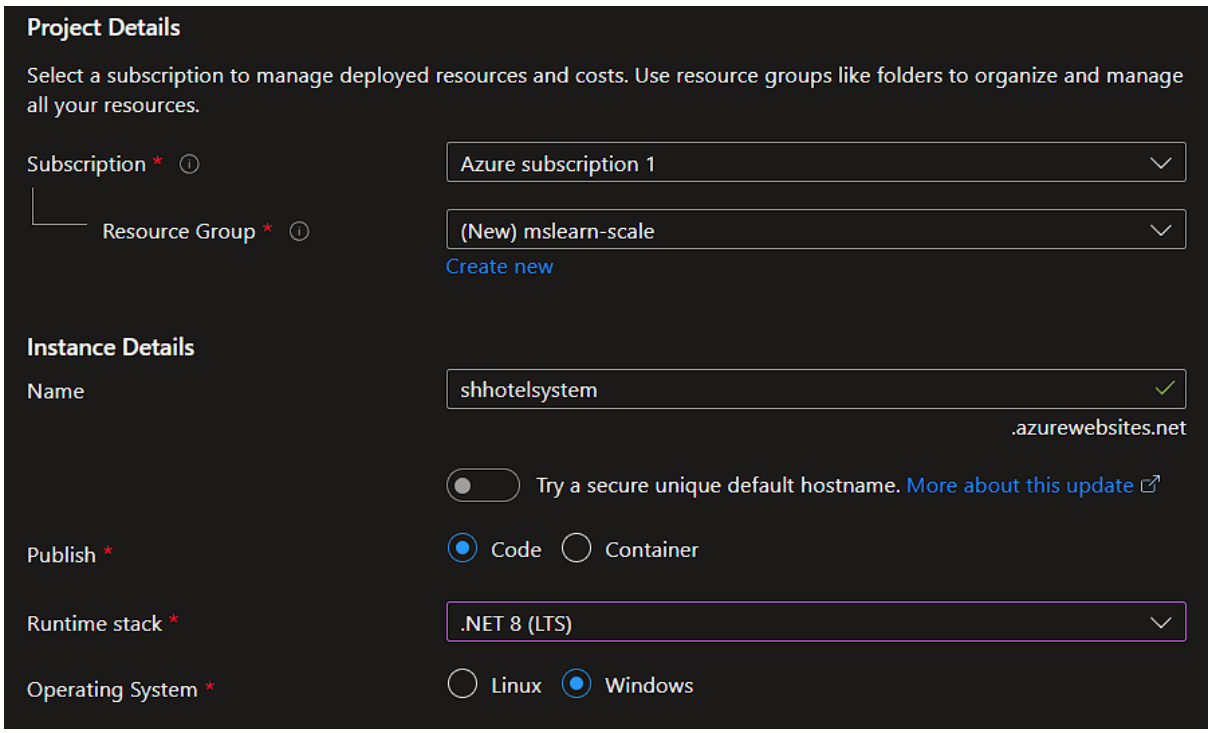
Step 2: Create a Web App

This is the website that we'll simulate traffic to.

2.1 Click Create a resource

2.2 Search and select Web App

2.3 Fill out the Basics tab



Then click Review + Create, wait for validation, then click Create.

Wait for Azure to finish creating your app (it takes a minute).

Step 3: Download the App Source Code

3.1 Open Cloud Shell

3.2 Run this command to download the hotel app code:

```
Your Cloud Shell session will be ephemeral so no files or system changes will persist beyond you
shaiful [ ~ ]$ git clone https://github.com/MicrosoftDocs/mslearn-hotel-reservation-system.git
Cloning into 'mslearn-hotel-reservation-system'...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (4/4), done.
remote: Total 33 (delta 3), reused 3 (delta 3), pack-reused 29 (from 1)
Receiving objects: 100% (33/33), 17.36 KiB | 8.68 MiB/s, done.
Resolving deltas: 100% (3/3), done.
```

Step 4: Build the Web App

4.1 Change to the folder with the source code:

4.2 Build everything:

```
shaiful [ ~ ]$ cd mslearn-hotel-reservation-system/src
shaiful [ ~/mslearn-hotel-reservation-system/src ]$ dotnet build

Welcome to .NET 8.0!
---------------------
SDK Version: 8.0.408

Telemetry
---------
The .NET tools collect usage data in order to help us improve your experience. It is collected by Microsoft and shared with the community. You can opt-out of telemetry by setting the DOTNET
_CLI_TELEMETRY_OPTOUT environment variable to '1' or 'true' using your favorite shell.

Read more about .NET CLI Tools telemetry: https://aka.ms/dotnet-cli-telemetry

----------------
Installed an ASP.NET Core HTTPS development certificate.
To trust the certificate, view the instructions: https://aka.ms/dotnet-https-linux

----------------
Write your first app: https://aka.ms/dotnet-hello-world
Find out what's new: https://aka.ms/dotnet-whats-new
Explore documentation: https://aka.ms/dotnet-docs
Report issues and find source on GitHub: https://github.com/dotnet/core
Use 'dotnet --help' to see available commands or visit: https://aka.ms/dotnet-cli
--------------------------------------------------------------------------------------
```

Wait until it says `Build succeeded`.

Step 5: Prepare the App for Deployment

5.1 Go into the web app folder:

5.2 Publish it to a folder:

```
shaiful [ ~/mslearn-hotel-reservation-system/src ]$ cd HotelReservationSystem
shaiful [ ~/mslearn-hotel-reservation-system/src/HotelReservationSystem ]$ dotnet publish -o website
  HotelReservationSystemTypes -> /home/shaiful/mslearn-hotel-reservation-system/src/HotelReservationSystemTypes/bin/Debug/netcoreapp2.1/HotelReservationSystemTypes.dll
  HotelReservationSystem -> /home/shaiful/mslearn-hotel-reservation-system/src/HotelReservationSystem/bin/Debug/netcoreapp2.1/HotelReservationSystem.dll
  HotelReservationSystem -> /home/shaiful/mslearn-hotel-reservation-system/src/HotelReservationSystem/website/
```

5.3 Zip the files:

```
shaiful [ ~/mslearn-hotel-reservation-system/src/HotelReservationSystem ]$ cd website
zip website.zip *
  adding: appsettings.Development.json (deflated 36%)
  adding: appsettings.json (deflated 24%)
  adding: dotnet-aspnet-codegenerator-design.dll (deflated 50%)
  adding: HotelReservationSystem.deps.json (deflated 85%)
  adding: HotelReservationSystem.dll (deflated 59%)
  adding: HotelReservationSystem.pdb (deflated 49%)
  adding: HotelReservationSystem.runtimeconfig.json (deflated 37%)
  adding: HotelReservationSystemTypes.dll (deflated 63%)
  adding: HotelReservationSystemTypes.pdb (deflated 44%)
  adding: Microsoft.CodeAnalysis.CSharp.Workspaces.dll (deflated 60%)
  adding: Microsoft.CodeAnalysis.Workspaces.dll (deflated 60%)
  adding: Microsoft.VisualStudio.Web.CodeGeneration.Contracts.dll (deflated 50%)
  adding: Microsoft.VisualStudio.Web.CodeGeneration.Core.dll (deflated 53%)
  adding: Microsoft.VisualStudio.Web.CodeGeneration.dll (deflated 48%)
  adding: Microsoft.VisualStudio.Web.CodeGeneration.EntityFrameworkCore.dll (deflated 55%)
  adding: Microsoft.VisualStudio.Web.CodeGeneration.Templating.dll (deflated 47%)
  adding: Microsoft.VisualStudio.Web.CodeGeneration.Utils.dll (deflated 49%)
  adding: Microsoft.VisualStudio.Web.CodeGenerators.Mvc.dll (deflated 67%)
  adding: NuGet.Frameworks.dll (deflated 55%)
  adding: System.Composition.AttributedModel.dll (deflated 45%)
  adding: System.Composition.Convention.dll (deflated 53%)
  adding: System.Composition.Hosting.dll (deflated 52%)
  adding: System.Composition.Runtime.dll (deflated 44%)
  adding: System.Composition.TypedParts.dll (deflated 52%)
  adding: web.config (deflated 40%)
```

Step 6: Deploy to Azure

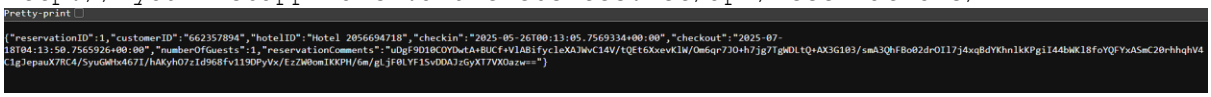
6.1 Deploy your code to the Azure Web App

```
shaiful [ ~/mslearn-hotel-reservation-system/src/HotelReservationSystem/website ]$ az webapp deployment source config-zip --src website.zip --name shyfulhotelsystem --resource-group mslearn
-scale
This command has been deprecated and will be removed in a future release. Use 'az webapp deploy' instead.
Getting scm site credentials for zip deployment
Starting zip deployment. This operation can take a while to complete ...
Deployment endpoint responded with status code 202
Polling the status of async deployment. Start Time: 2025-05-15 14:44:46.170560+00:00 UTC
{
  "active": true,
  "author": "N/A",
  "author_email": "N/A",
  "complete": true,
  "deployer": "ZipDeploy",
  "end_time": "2025-05-15T14:44:49.1701258Z",
  "id": "f8d327ad9c9f44989db8e076b9c3b9d3",
  "is_readonly": true,
  "is_temp": false,
  "last_success_end_time": "2025-05-15T14:44:49.1701258Z",
  "log_url": "https://shyfulhotelsystem-fmdhhngje2h3dpdc.scm.canadacentral-01.azurewebsites.net/api/deployments/latest/log",
  "message": "Created via a push deployment",
  "progress": "",
  "provisioningState": "Succeeded",
  "received_time": "2025-05-15T14:44:46.6070957Z",
  "site_name": "shyfulhotelsystem",
  "start_time": "2025-05-15T14:44:46.8576512Z",
  "status": 4,
  "status_text": "",
  "url": "https://shyfulhotelsystem-fmdhhngje2h3dpdc.scm.canadacentral-01.azurewebsites.net/api/deployments/latest"
}
```

Step 7: Test Your Web App

Open your browser and visit:

`http://<your-webapp-name>.azurewebsites.net/api/reservations/1`



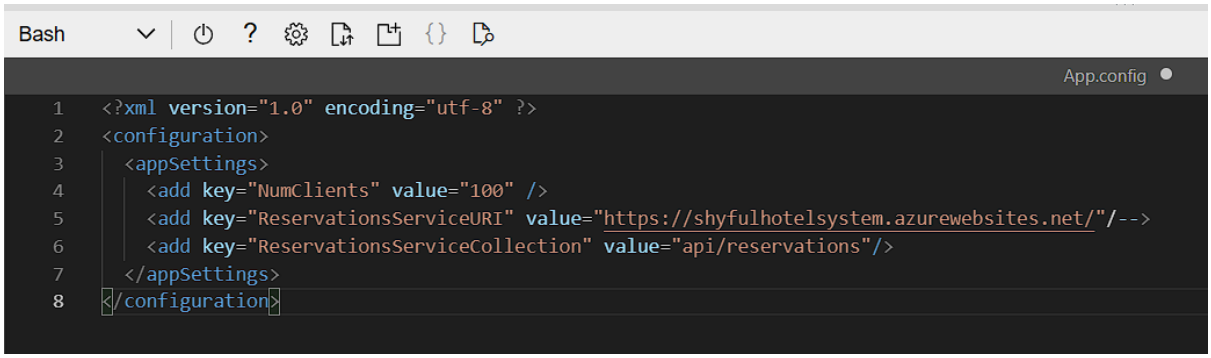
Step 8: Prepare the Test Client (Simulates Many Users)

8.1 Go to the test client folder:

8.2 Open the config file:

```
shaiful [ ~/mslearn-hotel-reservation-system/src/HotelReservationSystem/website ]$ cd ~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient
shaiful [ ~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient ]$ cd ~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient
shaiful [ ~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient ]$ code App.config
```

8.3 Change this line (replace with your real app name):



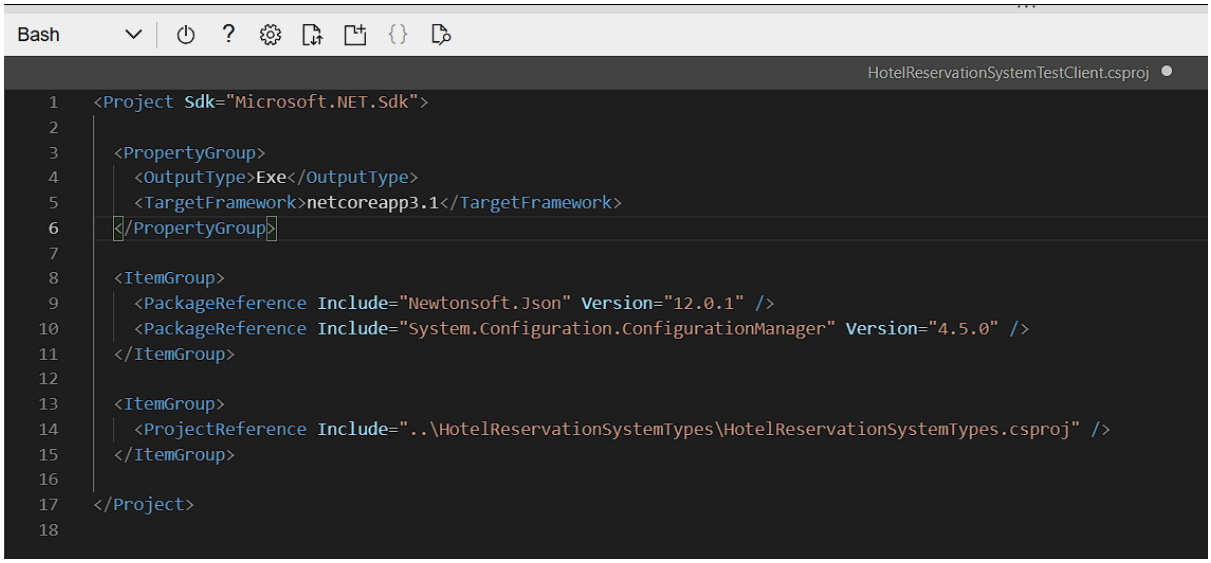
Step 9: Build and Fix the Framework

9.1 Rebuild the test client:

```
shaiful [ ~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient ]$ dotnet build
  Determining projects to restore...
```

9.2 If you get errors, edit this file:

```
Time Elapsed 00:00:01.22
shaiful [ ~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient ]$ code HotelReservationSystemTestClient.csproj
shaiful [ ~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient ]$
```



Save and close.

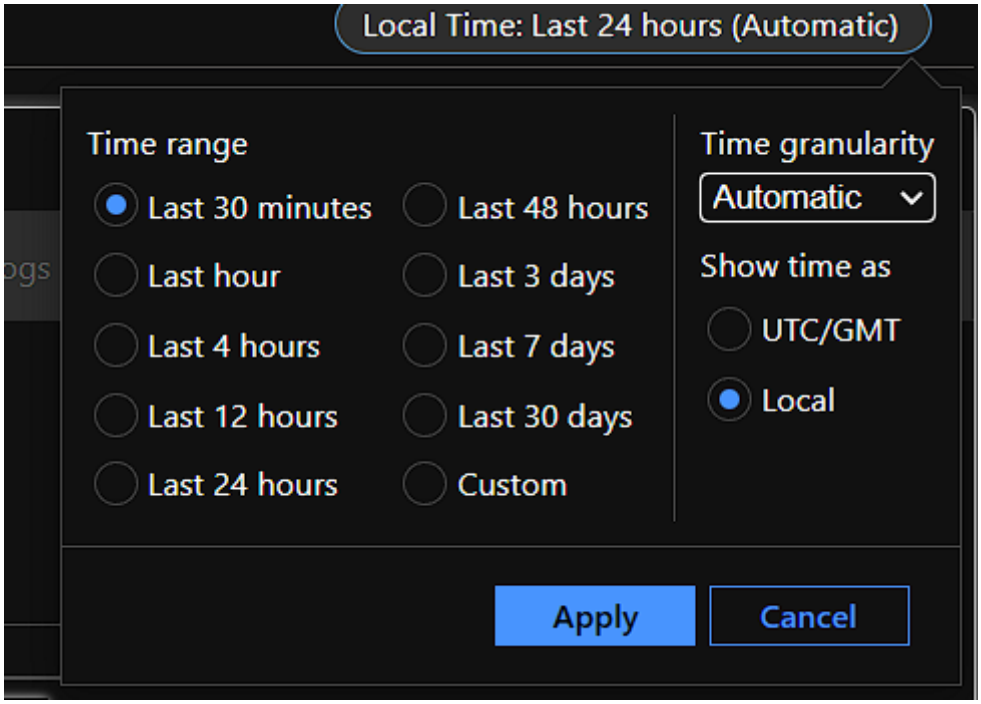
Step 10: Run the Client (Simulate 100 Users)

```
shaiful [ ~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient ]$ dotnet run
/home/shaiful/mslearn-hotel-reservation-system/src/HotelReservationSystemTypes/HotelReservationSystemTypes.csproj : warning NU1903: Package 'Microsoft.NETCore.App' 2.1.0 ha
s a known high severity vulnerability, https://github.com/advisories/GHSA-2xjx-v99w-gqf3 [/home/shaiful/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClien
t/HotelReservationSystemTestClient.csproj]
/home/shaiful/mslearn-hotel-reservation-system/src/HotelReservationSystemTypes/HotelReservationSystemTypes.csproj : warning NU1902: Package 'Microsoft.NETCore.App' 2.1.0 ha
s a known moderate severity vulnerability, https://github.com/advisories/GHSA-3gp9-h8hw-pxpw [/home/shaiful/mslearn-hotel-reservation-system/src/HotelReservationSystemTestC
lient/HotelReservationSystemTestClient.csproj]
/home/shaiful/mslearn-hotel-reservation-system/src/HotelReservationSystemTypes/HotelReservationSystemTypes.csproj : warning NU1903: Package 'Microsoft.NETCore.App' 2.1.0 ha
s a known high severity vulnerability, https://github.com/advisories/GHSA-3w5p-jhp5-c29q [/home/shaiful/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClien
t/HotelReservationSystemTestClient.csproj]
/home/shaiful/mslearn-hotel-reservation-system/src/HotelReservationSystemTypes/HotelReservationSystemTypes.csproj : warning NU1902: Package 'Microsoft.NETCore.App' 2.1.0 ha
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
 Error making reservation: Response status code does not indicate success: 408 (Request Timeout).
```

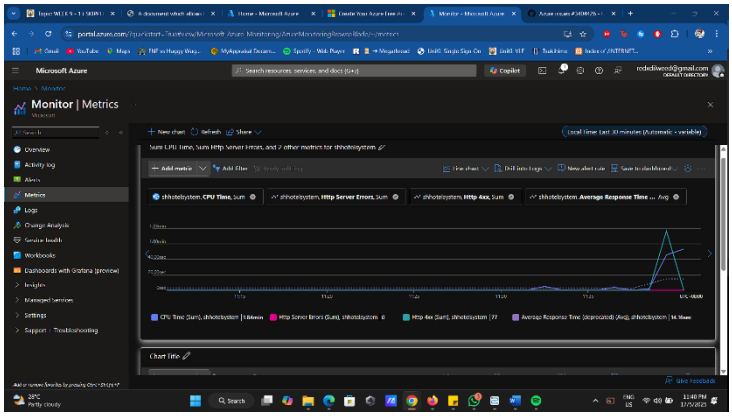
Step 11: Monitor Web App Performance

In Azure Portal:

1. Go to your Web App.
2. Scroll down and select **Monitoring > Metrics**.
3. Set **Time range** to "Last 30 minutes".



4. Click **Add metric** multiple times and select:



5. Click **Pin to dashboard**

Let the test client run for **about 5 minutes**. You'll likely see:

- High CPU usage
- Many 408 errors (Http 4xx)
- Long response times

Step 12: Scale Out the App

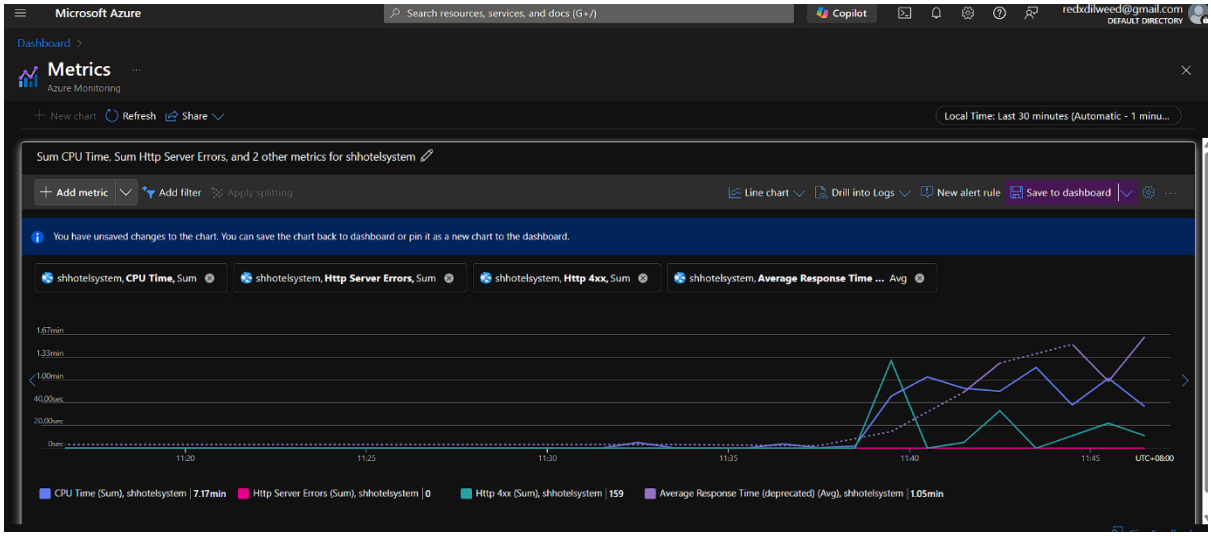
In Azure Portal:

1. Go to your Web App.
2. Under **Settings**, choose **Scale-out (App Service plan)**.
3. Set **Instance count** to **5**
4. Click **Save**

Now the app runs on 5 servers instead of 1. Wait and watch the test client again.

You'll start seeing:

- Fewer timeouts
- Lower response times
- Improved performance



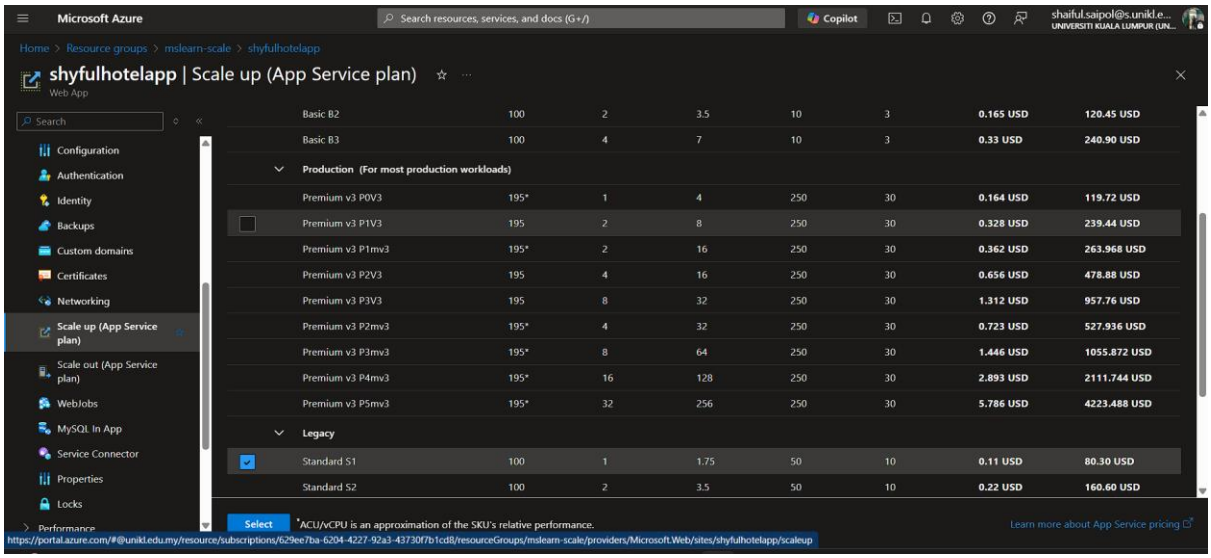
Step 14: Stop Everything

Once done:

1. In the **Cloud Shell**, press `Enter` to stop the client.
2. Go back to **Scale-out**, and set instance count back to **1** to avoid extra charges.

Step 15: Scale up

1. Navigate to your **App Service plan**. You can typically find this by searching for "App Service plans" in the portal's search bar.
2. In the left-hand menu, look for a section like **Settings** and then **Scale up (App Service plan)**.
3. You'll see details about your current pricing tier. Note the ACU and memory. The original exercise uses S1.



Step 16: Run the test client app

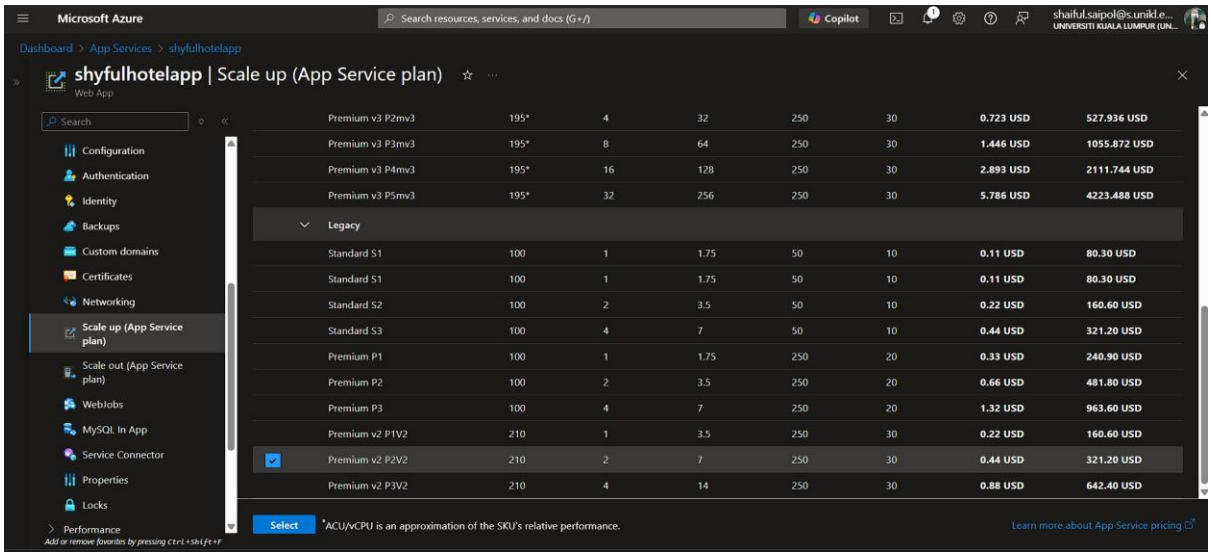
1. Open **Cloud Shell**.
2. Navigate to the test client folder:
3. Run the client app:

As before, observe the slow responses and potential HTTP 408 (Timeout) errors.

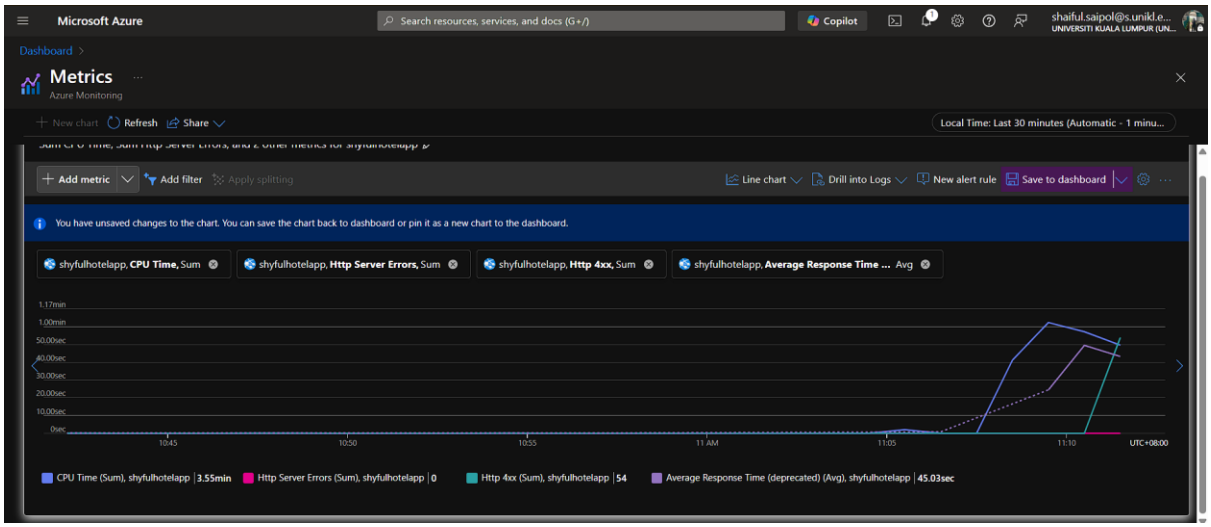
4. Let the app run for about five minutes. Monitor the performance metrics in the Azure portal (as described in the previous response). You'll likely see high CPU usage and response times.

Step 17: Scale up the web app and monitor the results

1. In the **Azure portal**, return to your **App Service plan**.
2. Go to **Settings** and then **Scale up (App Service plan)**.
3. Select a higher pricing tier. The exercise mentions P2V2. *Instead*, look for a tier that offers significantly more ACU and memory than your current one. Be mindful of the cost implications, as higher tiers are more expensive.
4. Select **Apply**.



5. Wait about five minutes, then check the performance metrics in the Azure portal. You should see a noticeable improvement: lower response times and fewer errors.



6. Return to **Cloud Shell** and press **Enter** to stop the client app.

Conclusion

This project demonstrated a full lifecycle deployment of a web application in Azure using Git and deployment slots. It covered crucial cloud management tasks such as version control, autoswap configuration, performance monitoring, and both horizontal and vertical scaling. These skills are critical for managing production-grade cloud applications.