IIB43203 ASSESSMENT COVERSHEET

Attach this coversheet as the cover of your submission. All sections must be completed.

Section A: Submission Details

Programme	:	BACHELOR IN INFORMATION TECHNOLOGY (HONS) (INTERNET OF THINGS)
Course Code & Name	:	IIB43203 – CLOUD COMPUTING
Assessment	:	**GROUP PROJECT ASSIGNMENT**
Dateline	:	**3/6/2025**
Penalties	:	• **10%** will be deducted for late submission. • Plagiarised work is an Academic Offence in University Rules & Regulations and will be penalised accordingly. • **AI content** will affect your mark

Section B: Academic Integrity

Tick (√) each box below if you agree:

√	I have read and understood the UniKL's policy on Plagiarism in University Rules & Regulations.
√	This submission is my own, unless indicated with proper referencing.
√	This submission has not been previously submitted or published.
√	This submission follows the requirements stated in the course.

Section C: Submission Detail

Office Receipt of Submission

No	Student Name(s)	Student ID(s)
1.	NUR AISHAH KAMALIAH BINTI MEZLAN	52224123034
2.	SHARIFAH NUR QISTINA NABILA BINTI SYED ABD RANI	52224123537
3.	SOLIHAH NAFISATUL 'ILMI BINTI MUHAMMAD NAZRI	52224123588
4.	NUR HUDA BATRISYIA BINTI HARMIZI	52224123502

Student Receipt of Submission

The submission receipt of the student task will be based on the **VLE submission date and time**.

Jobscope

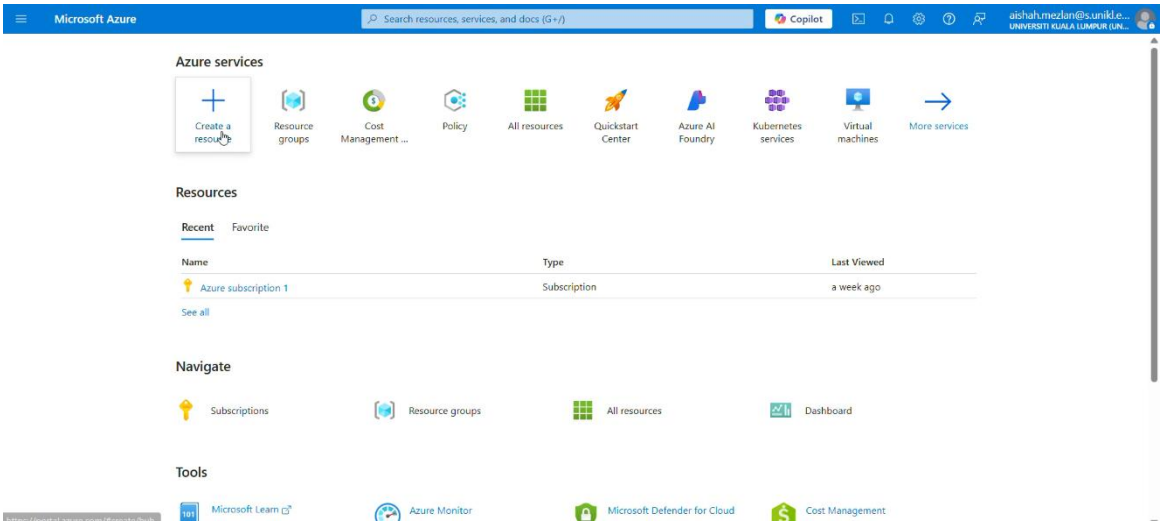
Name	Lab 30A	Lab 30B	Lab 31	Lab 32	Report	Video
NUR AISHAH KAMALIAH BINTI MEZLAN	/	/			/	
SHARIFAH NUR QISTINA NABILA BINTI SYED ABD RANI			/	/	/	
SOLIHAH NAFISATUL ‘ILMI BINTI MUHAMMAD NAZRI	/				/	/
NUR HUDA BATRISYIA BINTI HARMIZI			/		/	/

Lab 30A: Create Deployment Slots

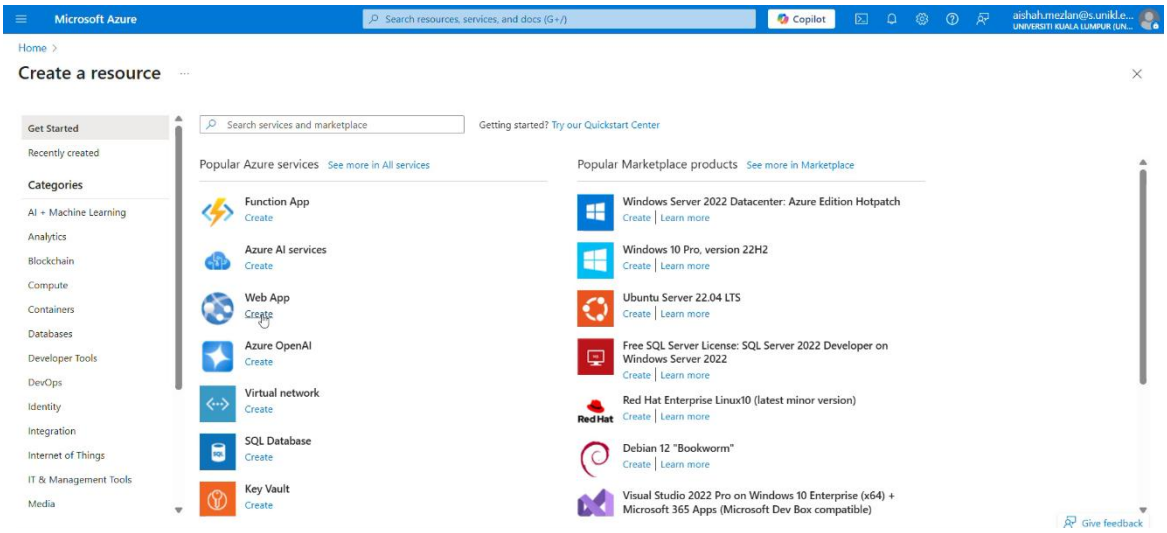
A deployment slot serves as a separate environment for a web application, allowing new versions to be tested before being released to the live site. By swapping slots, updates can be deployed seamlessly without causing downtime. For instance, if an organization chooses to use Azure App Service deployment slots to streamline the update process for its social media web application, the next step would be to configure the application in Azure and set up the necessary deployment slots.

Create a web app

1. On the Azure portal menu or from the Home page, select Create a resource.

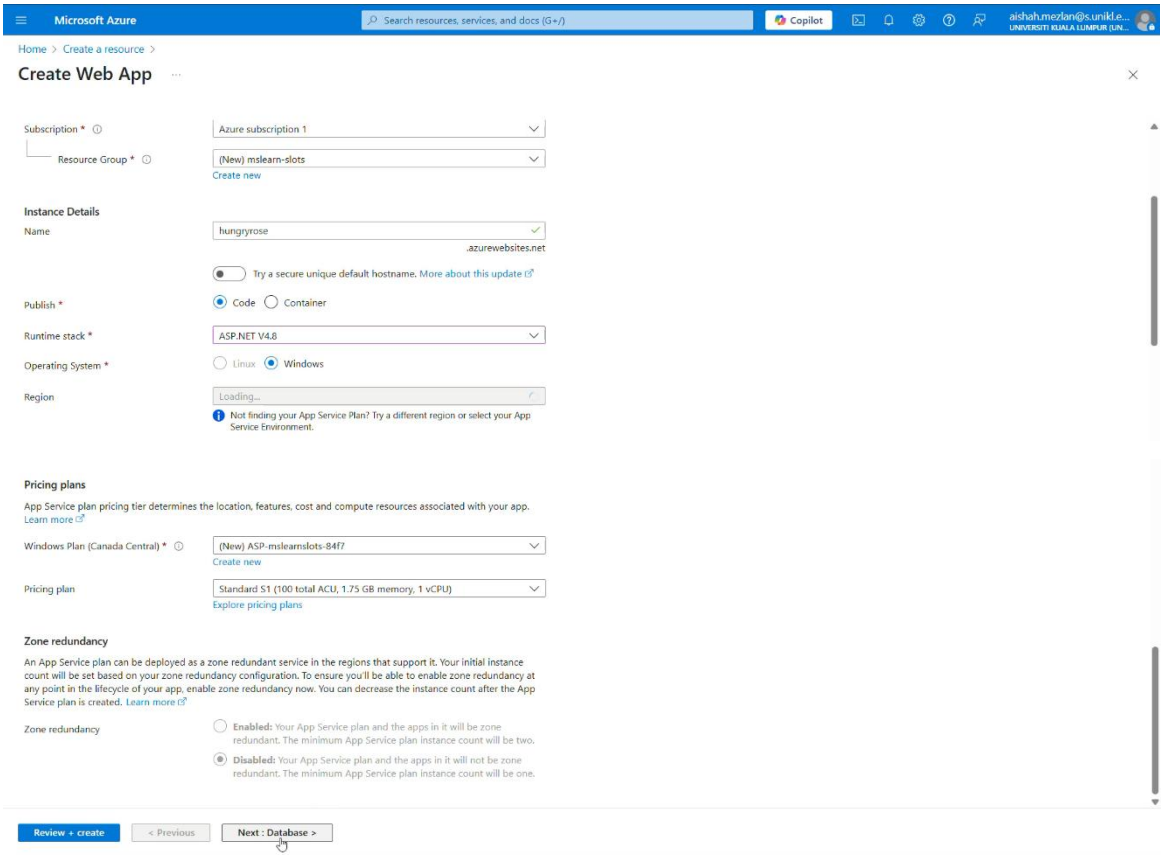


2. From the left-hand menu, click on Web, then search for and choose Web App.



3. On the Basics tab, enter the following values for each setting.

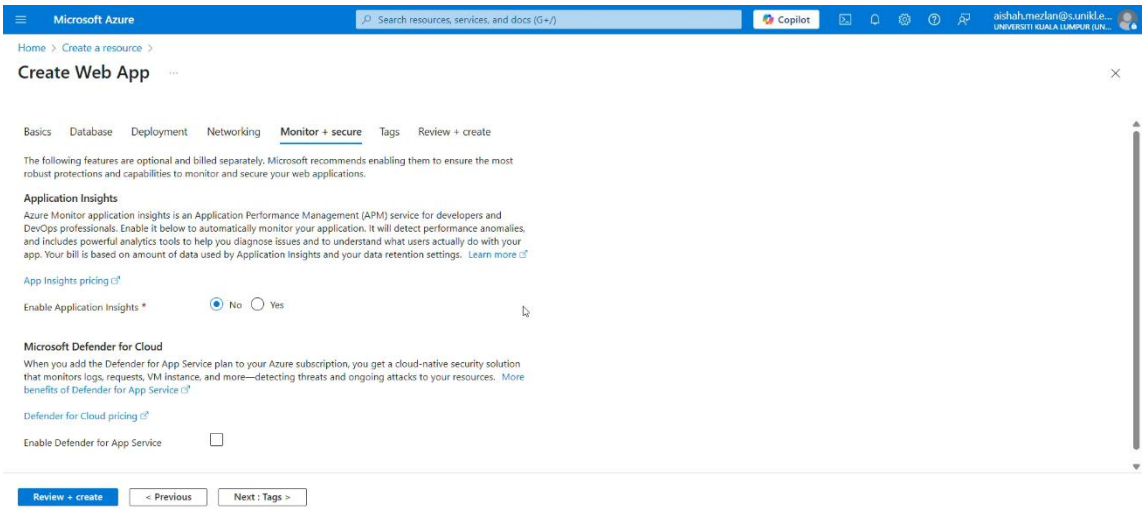
	Setting	Value
Project Details	Subscription	Select the subscription you'd like to use to complete the exercise
	Resource Group	Create a new resource group named mslearn-slots
Instance Details	Name	Enter a unique name
	Publish	Code
	Runtime stack	ASP.NET V4.8
Operating System Windows	Region	Select a region near you
App Service Plan	Windows Plan	Accept default
	Pricing Plan	Accept default



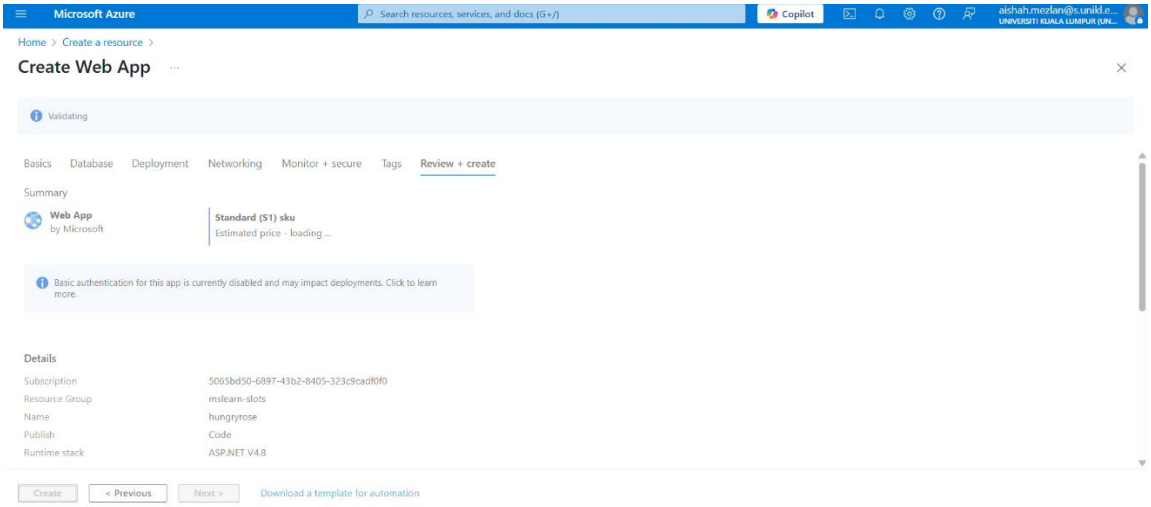
4. Select Next: Deployment and select Next: Networking. Do not change anything.

5. Select Next: Monitoring, and enter the following value for the setting.

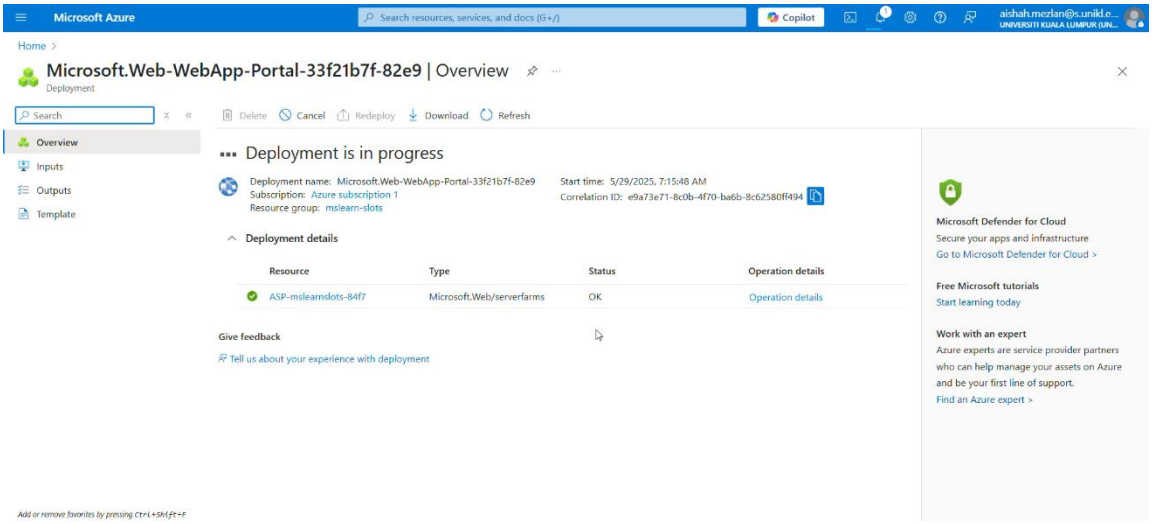
	Setting	Value
Application Insights	Enable Application Insights	No



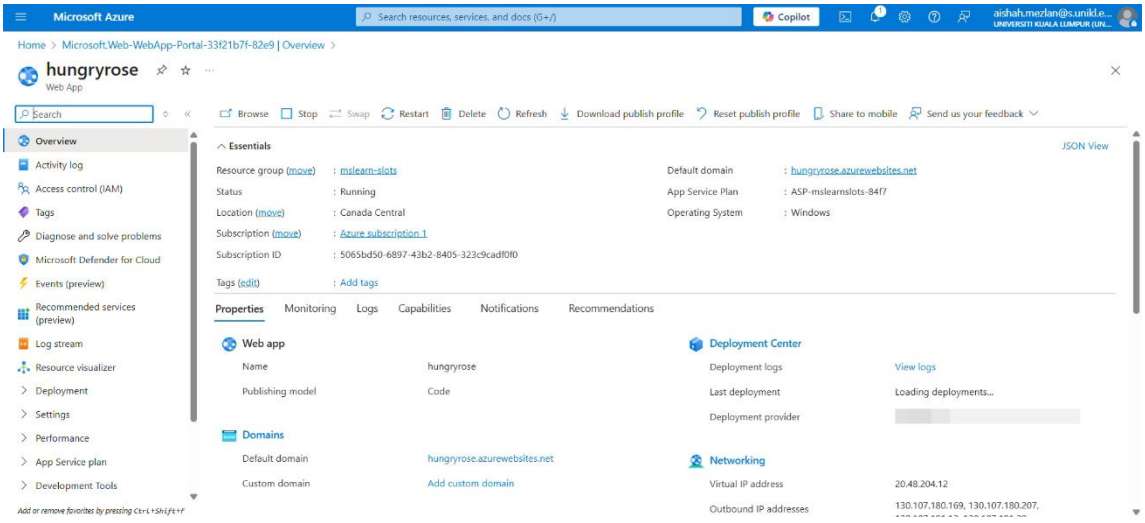
6. Select Review + Create and wait for it to be validated.



7. Once the information is validated, click **Create** and wait for Azure to finish setting up the web app.



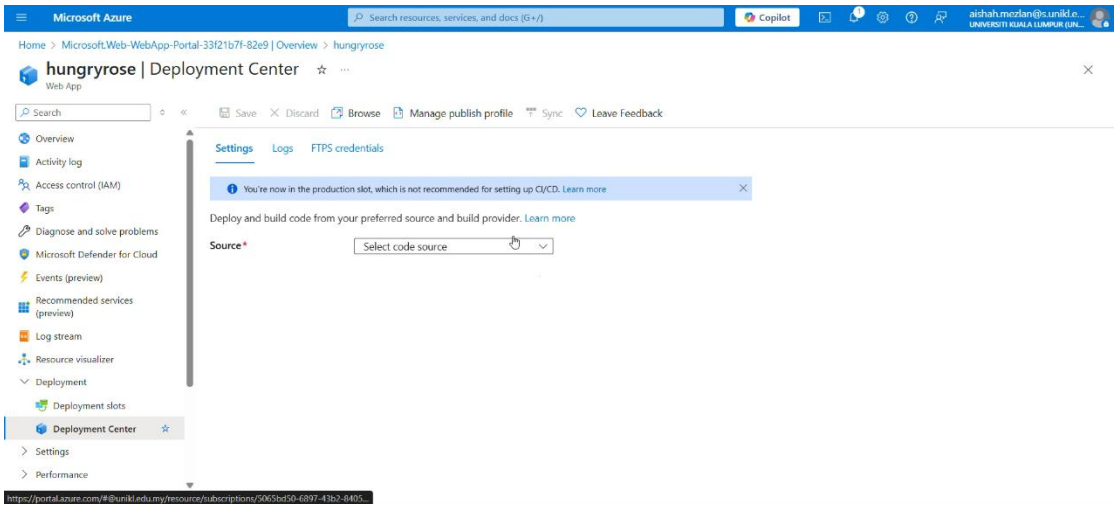
8. When deployment successfully completes, select Go to resource. The App Service pane appears.



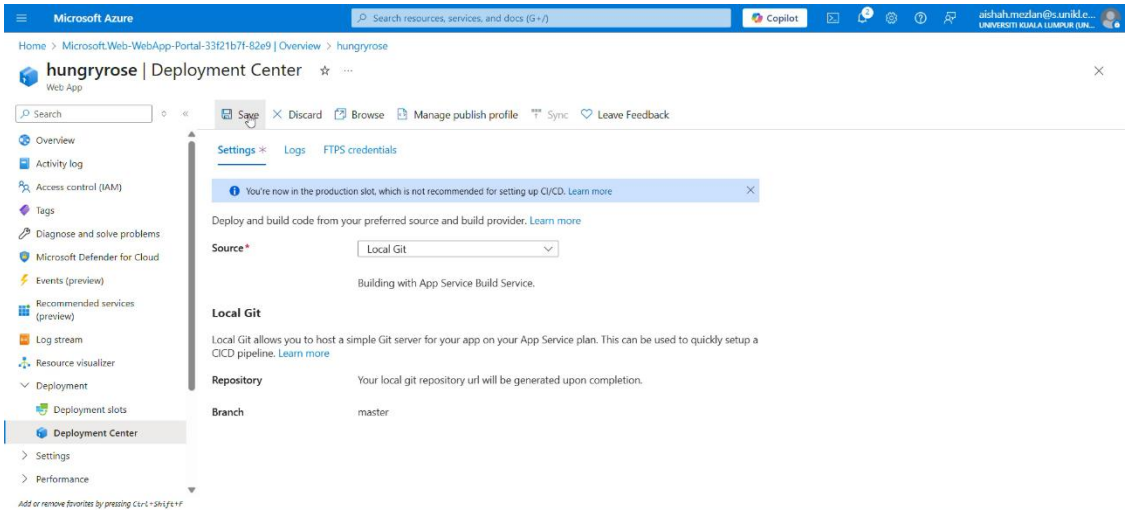
Configure git deployment

Any standard deployment tools can be used to deploy a web application and manage its deployment slots. In this section, the web application will be configured to utilize a local Git repository for deployment purposes. The following steps outline the setup process.

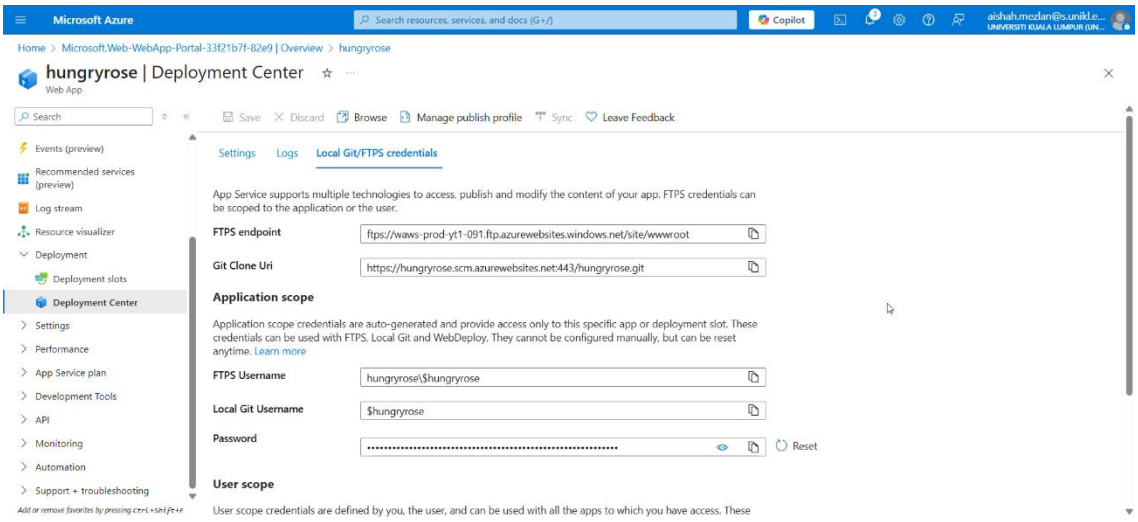
1. On your web app App Service page, in the left menu, under Deployment, select Deployment Center.



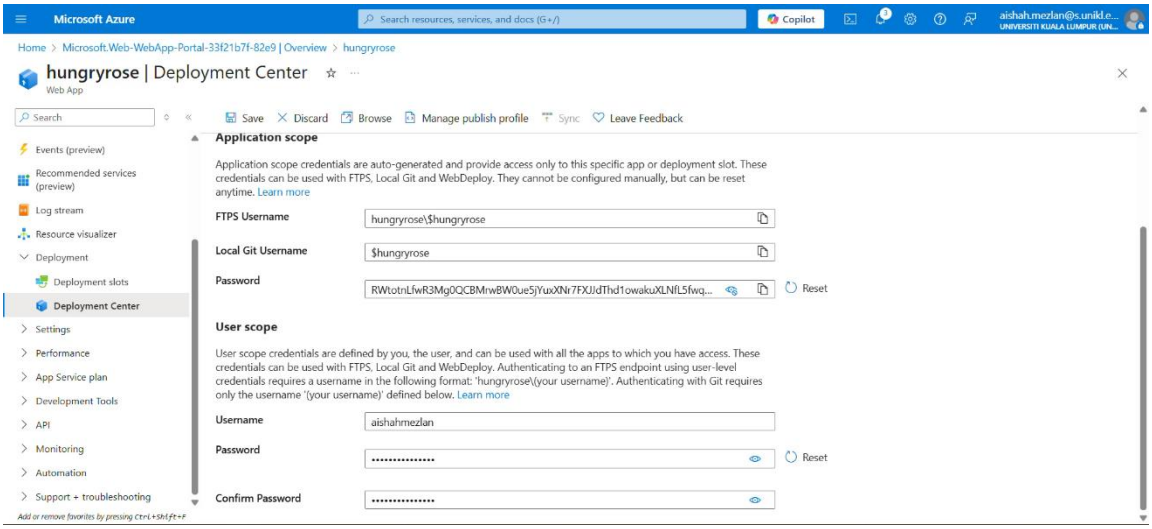
2. On the Settings tab, for Source, select Local Git. Make sure to enable your SCM Basic Authentication. On the top menu bar, select Save.



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



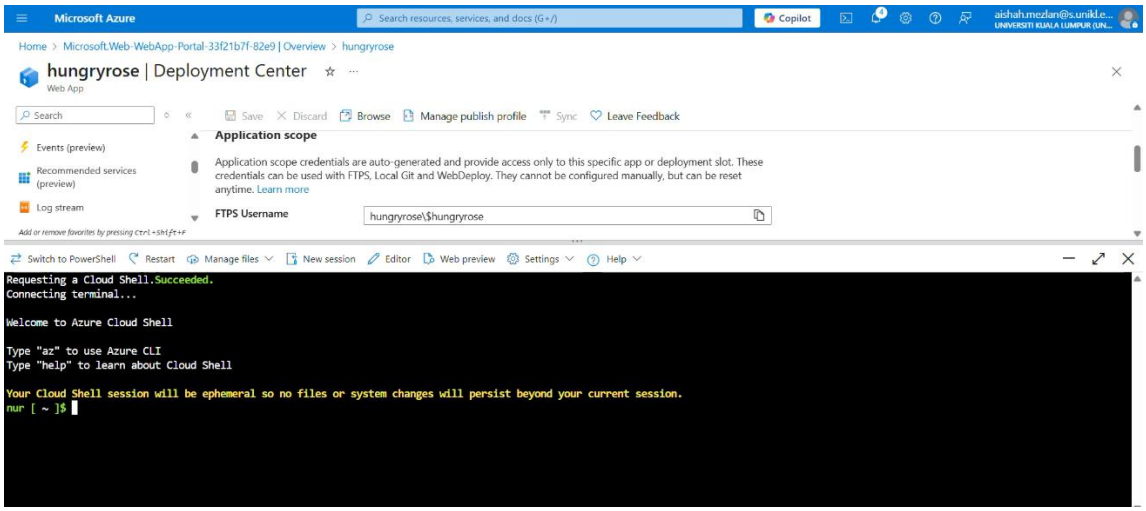
4. In the User Scope section, enter a username and password of your choice. Then, click Save in the top menu bar. Be sure to note down the username and password for future use.



Configure the git client and clone the web app source code

Next, the Git client will be configured within the Cloud Shell environment and used to clone a sample web application. The steps outlined below will guide this process.

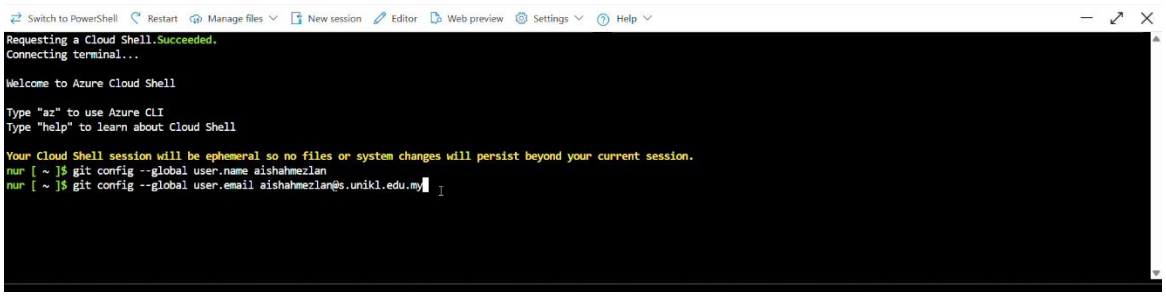
1. In the Azure portal, open Azure Cloud Shell by selecting its icon on the top toolbar.



2. In the Cloud Shell toolbar, ensure that Bash is selected. Then, copy the code provided below into a Notepad file and replace the quoted values with a preferred username and email address. These configuration details are not associated with any specific Azure account or subscription, so any chosen values may be used.

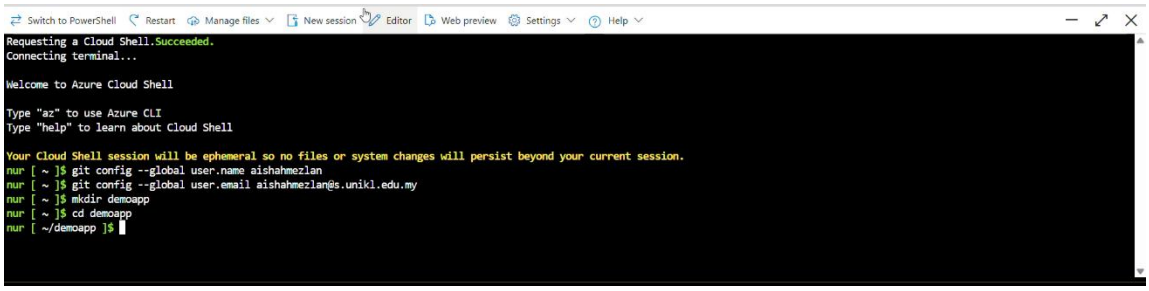
```
git config --global user.name "your-username"
git config --global user.email "your-email-address"
```

3. Copy and paste your edited code into the Cloud Shell and run it.



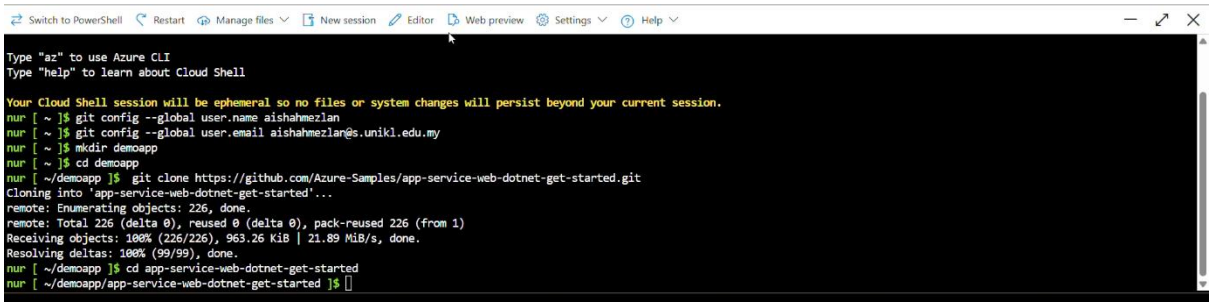
4. Create a folder for the source code. Run the following commands.

   mkdir demoapp
   cd demoapp



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

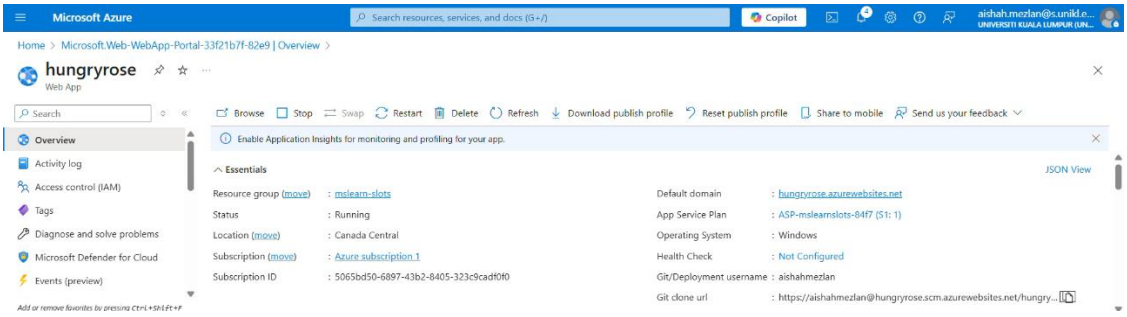
   git clone https://github.com/Azure-Samples/app-service-web-dotnet-get-started.git
   cd app-service-web-dotnet-get-started



Configure a git remote to deploy the app to production.

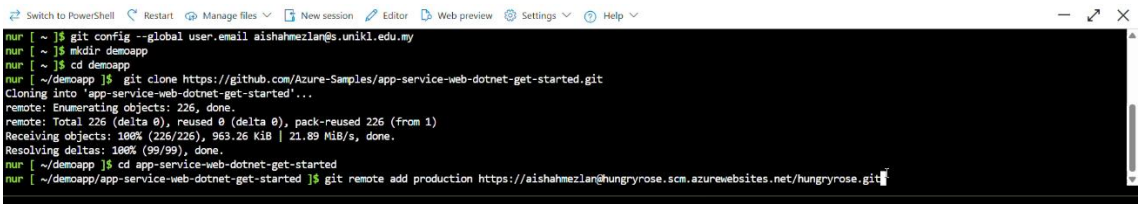
To deploy the source code to the production slot of the web app using Git, it is necessary to configure the app's Git URL as a remote repository. The steps below outline how to complete this setup.

1. In the Azure portal, make sure your web app is running. From the left-hand menu, click on Overview.
2. In the Overview pane, locate the Essentials section. Here, you'll find the Default domain and the Git clone URL. Hover over the Default domain and click the Copy to clipboard icon. This domain includes the deployment name of your web app.



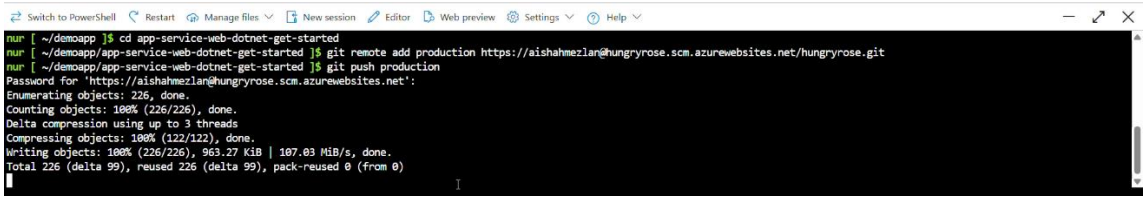
3. Hover over the Git clone URL and click the Copy to clipboard icon. Keep in mind that this URL also includes your deployment username. (Refresh if it does not appear).
4. In Cloud Shell, run the following command to add the Git remote named production. Replace git-clone-url with the URL you copied in the previous step.

```
cd app-service-web-dotnet-get-started
git remote add production <git-clone-url>
```

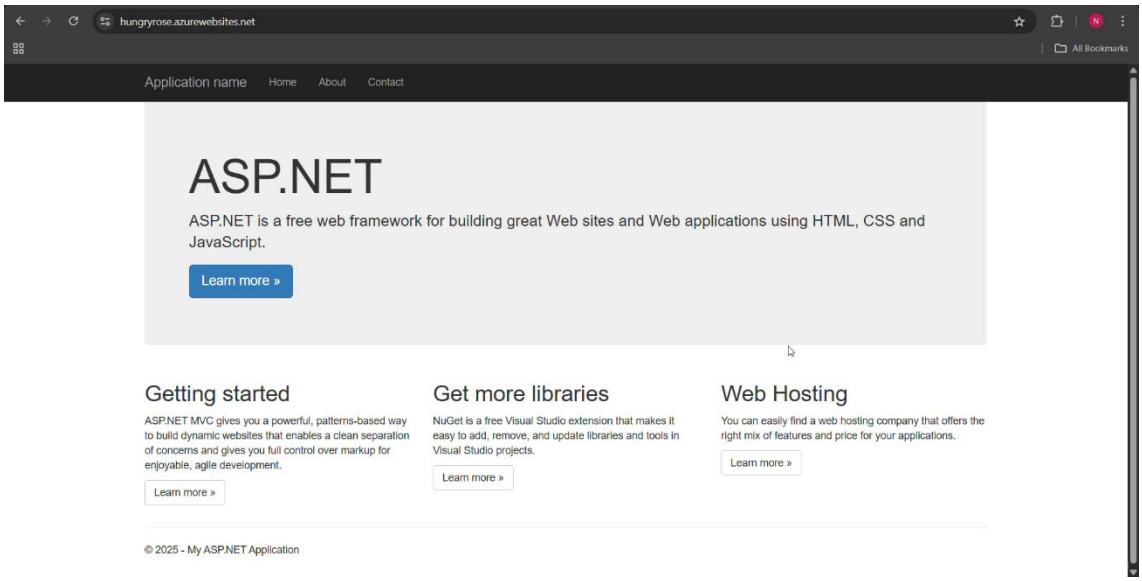


5. To deploy the web app to the production slot, run the following command. When you're prompted for the password, enter your deployment password you created previously.

   git push production



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

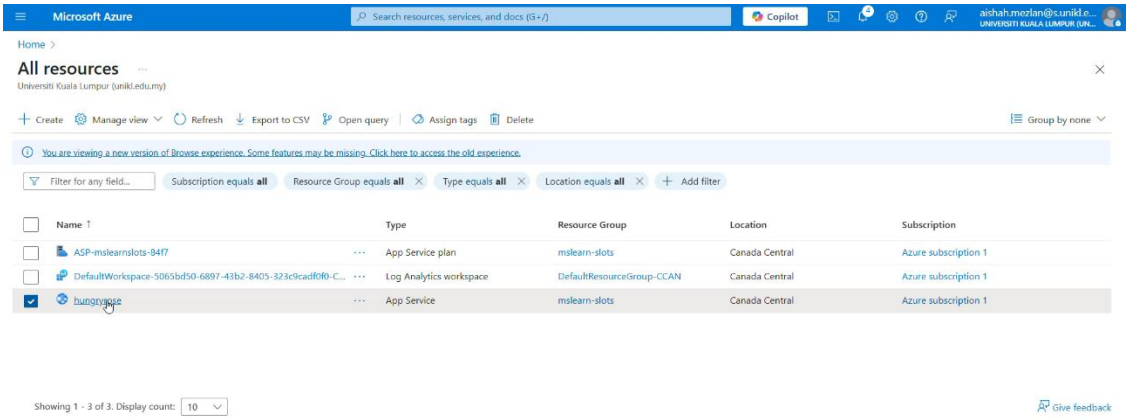


7. Close the browser tab that displays the web app.

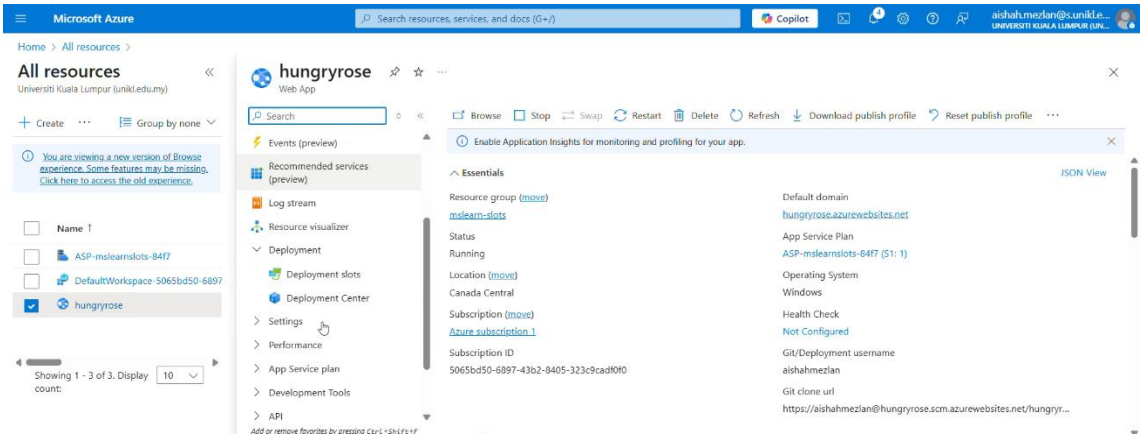
Create a new staging slot

The newly created web app currently includes only a single slot, which is the production slot where the initial source code has been deployed. The next step involves creating an additional deployment slot to stage and test new versions of the web app before pushing them to production.

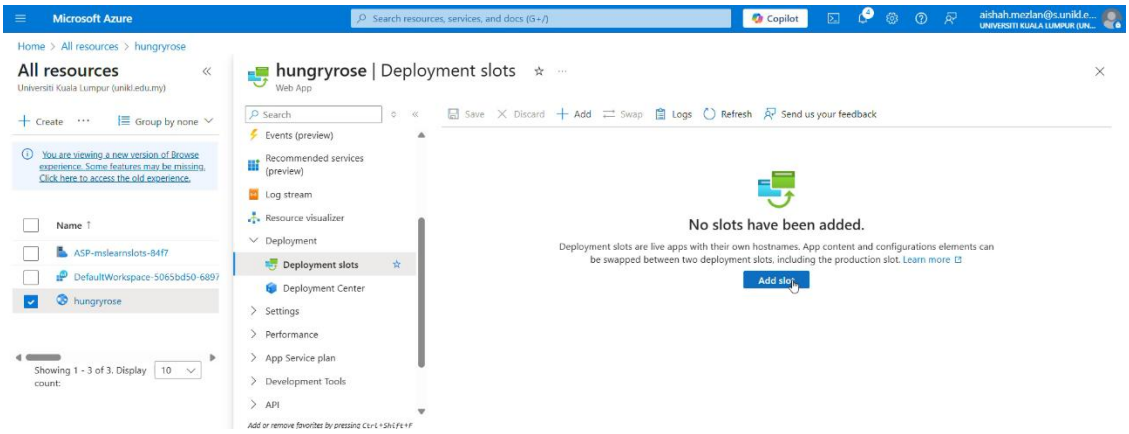
1. On the Azure portal menu or from the Home page, select All resources.



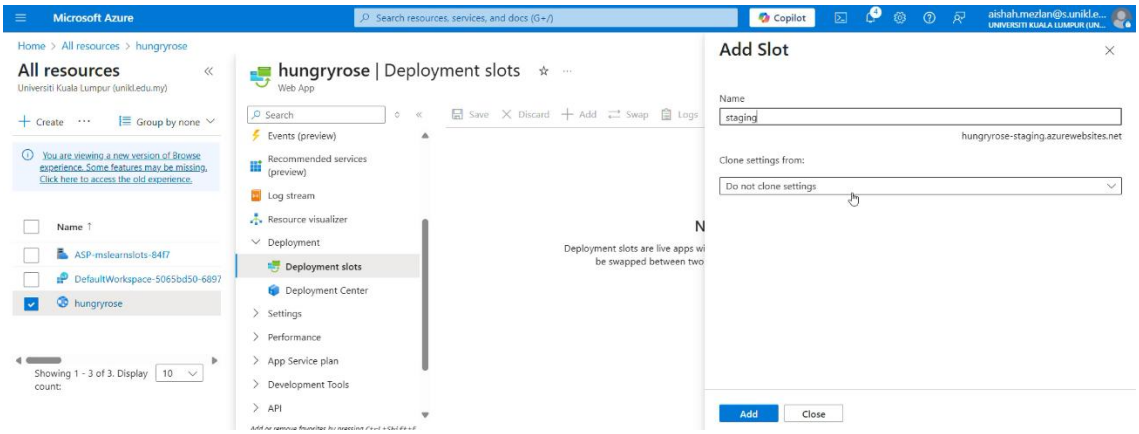
2. Select your web app. The web app App Service pane appears.



3. In the menu pane, under Deployment, select Deployment slots. The Deployment slots pane appears.



4. 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.

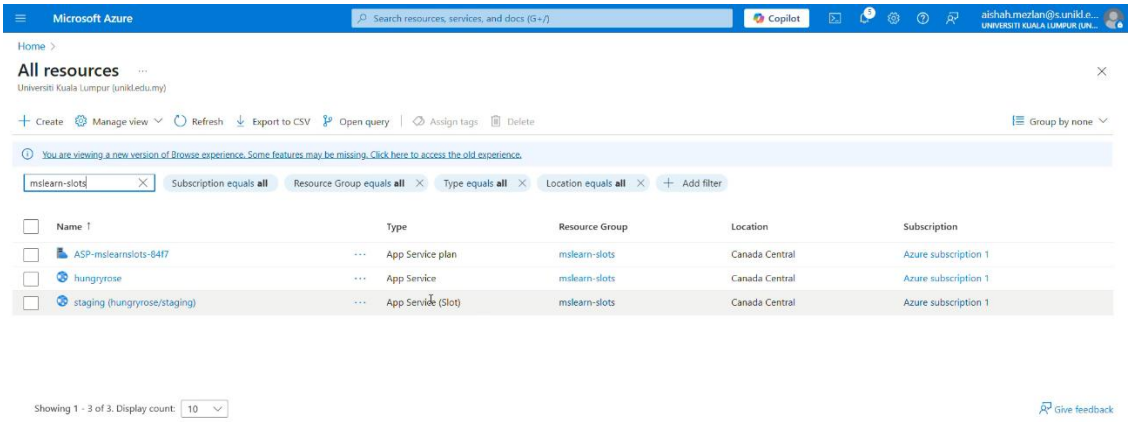


6. After the deployment slot is successfully created, select Close.

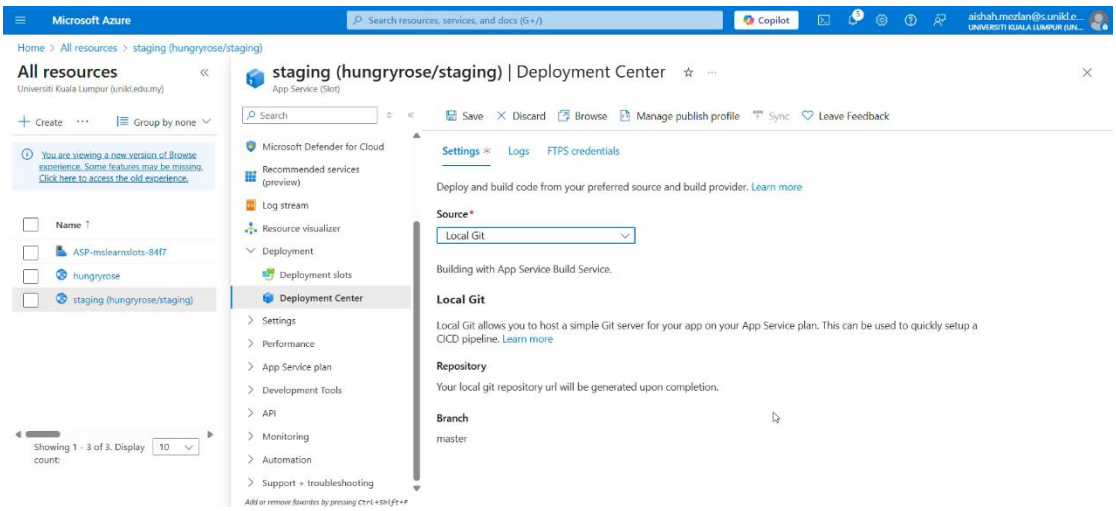
Set up git deployment for the staging slot

Configure the newly created deployment slot to use Git for deploying source code, similar to how the production slot was set up. Follow the steps below to complete the Git deployment configuration for this slot.

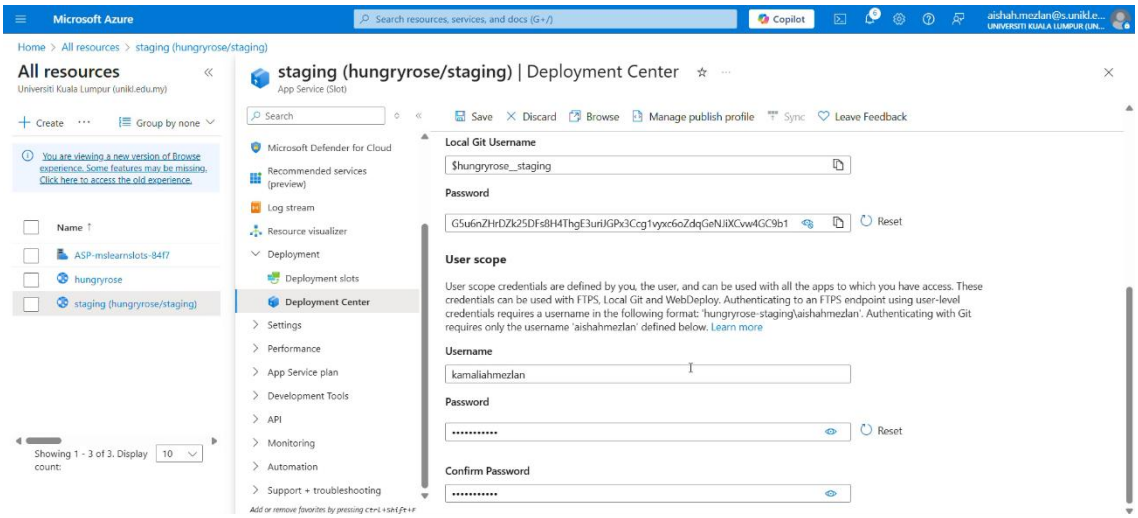
1. In the Azure portal menu or on the Home page, select All resources. In the list of resources, filter by Resource group set to mslearn-slots. You will see two App Service entries—deployment slots appear as separate apps in the portal. Click on the entry for the staging slot to open its Overview pane.



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.



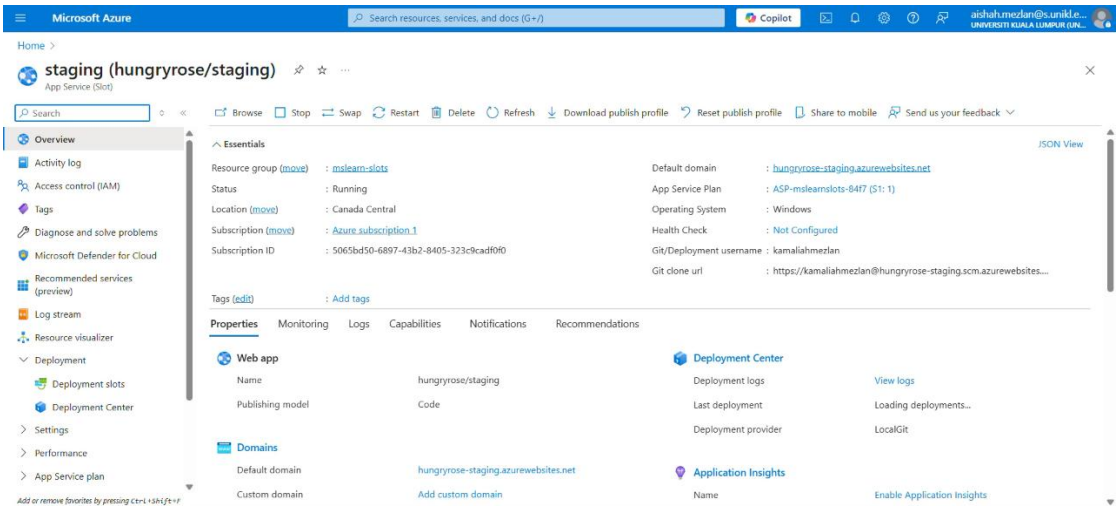
4. On the resulting Deployment Center pane, select the Local Git/FTPS credentials tab.
5. Under User scope, enter a new username and password of your choice, and from the top menu bar, select Save. Make a note of the username and password for later.



Set up git to deploy the app to the staging slot

To deploy the source code to the new deployment slot using the Git client, an extra remote must be added to the existing Git configuration. Follow the instructions below to complete this setup.

1. In the Azure portal, on the staging web app's Overview page, in the Essentials section, copy the Git clone url. Note that the URL contains your deployment username. (Refresh if it does not appear)

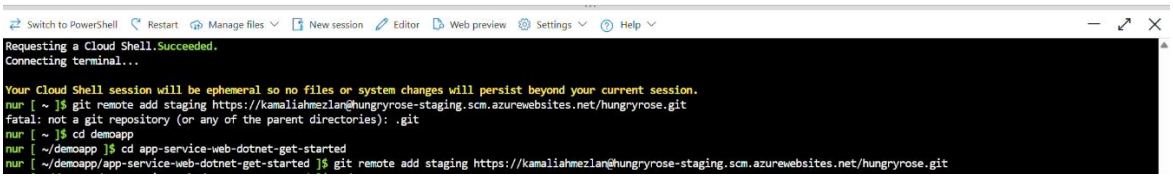


2. To add the remote for the staging slot, run the following commands in Cloud Shell. Replace git-clone-uri with the URI from the previous step.

   cd demoapp

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

   git remote add staging git-clone-uri

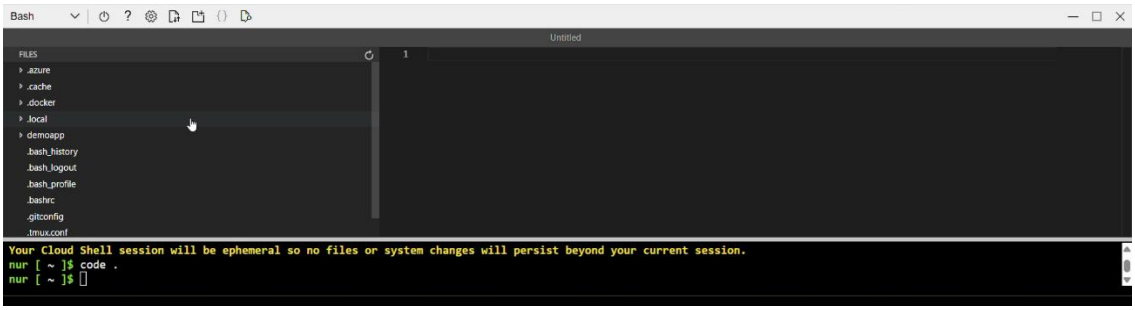


Modify the app source and deploy the app to the staging slot

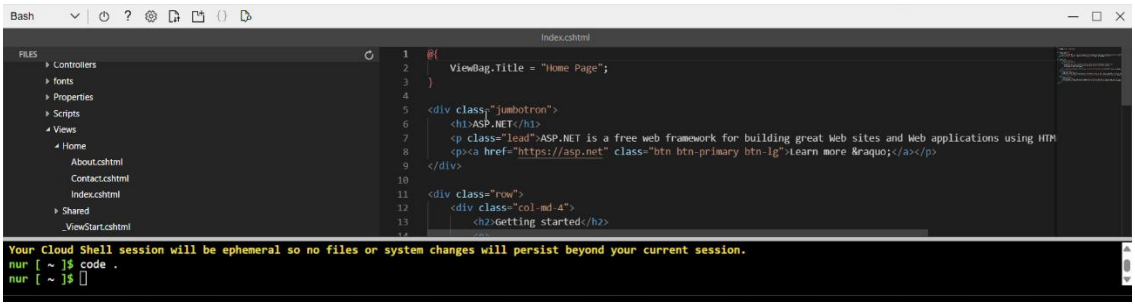
Next, make a small change to the web app, and then use git to deploy the new version to the staging slot:

1. In Cloud Shell, run the following command.

   code .



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

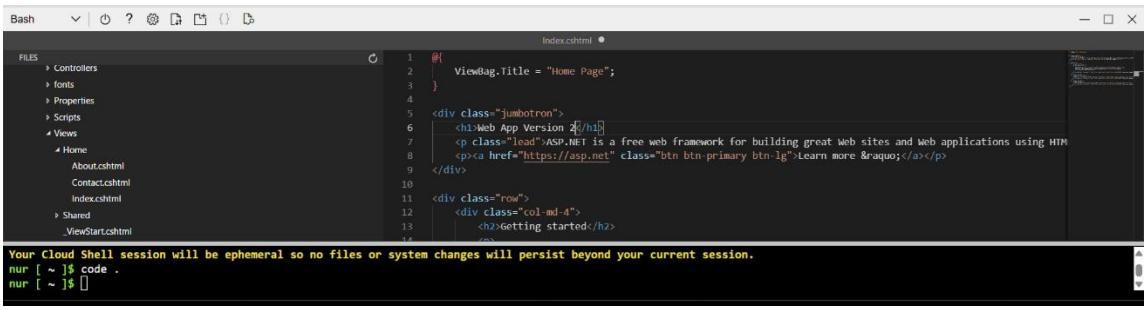


4. Locate the following code.

   <h1>ASP.NET</h1>

5. Replace that code with this code.

   <h1>Web App Version 2</h1>



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

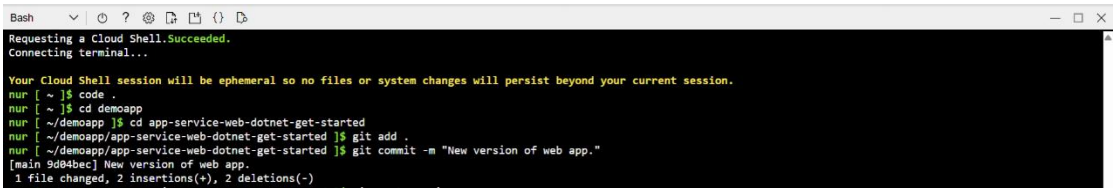
   In Cloud Shell, execute the following commands to commit the updated version of your app to Git and deploy it to the staging slot. When asked, enter your deployment password.

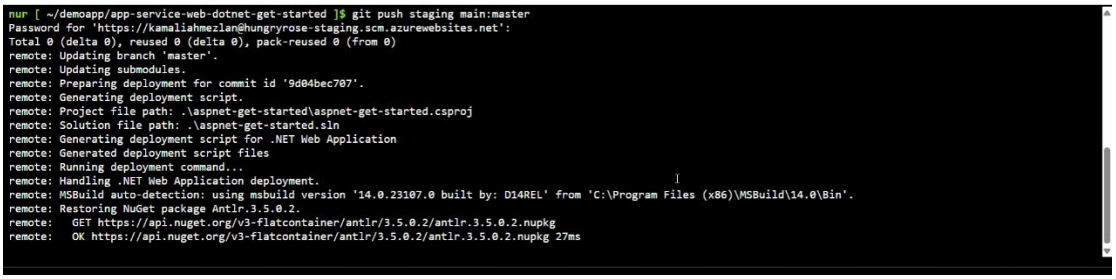
   cd /demoapp/app-service-web-dotnet-get-started

   git add .

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

   git push staging main:master

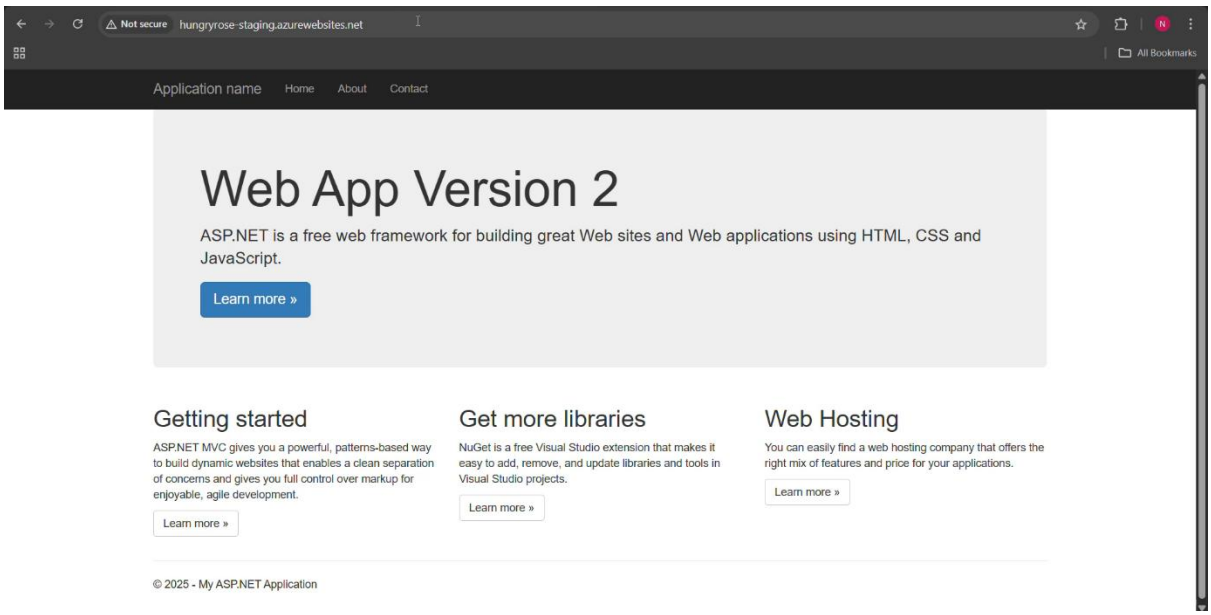




Browse the staging slot

Now you can see the new version of the web app by visiting the URL of the staging deployment slot.

In the Azure portal, navigate to the Overview page for the staging slot, then click Browse in the top menu bar. The updated web app will open in a new browser tab.



At this stage, the staging slot contains the new version of your code, allowing you to test it. Keep in mind that the production slot still runs the previous version, so users have not seen your new updates yet.

Lab 30B: Deploy a web app by using deployment slots

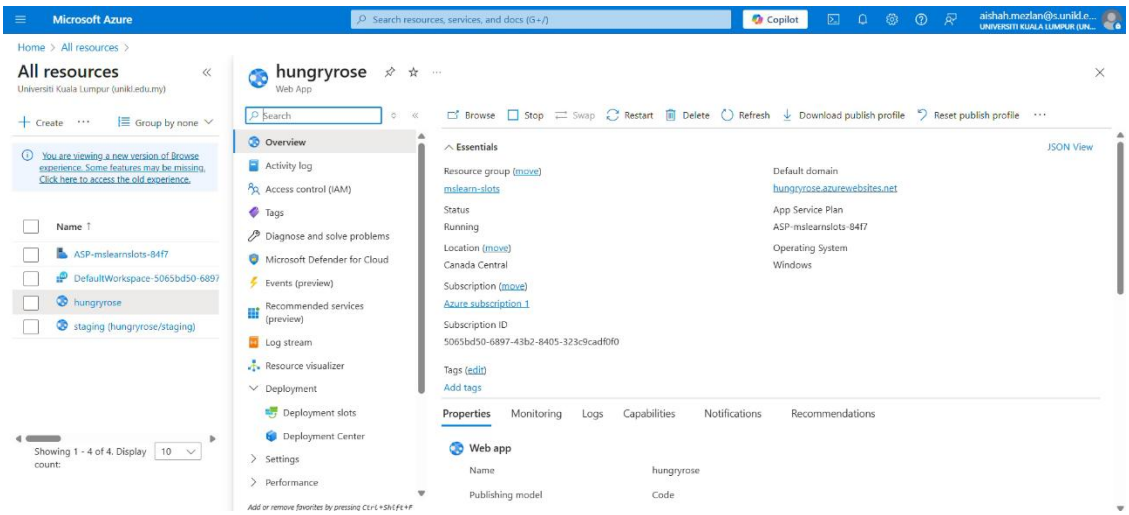
When it's time to swap two deployment slots, it's important to confirm that each slot has the appropriate configuration after the swap. For instance, after completing tests on version 2 of a social media web application, the next step would be to deploy it to the production environment. To streamline future deployments, automatic slot swapping can also be configured. This section explains how to carry out both manual and automatic slot swaps.

Configure a slot setting

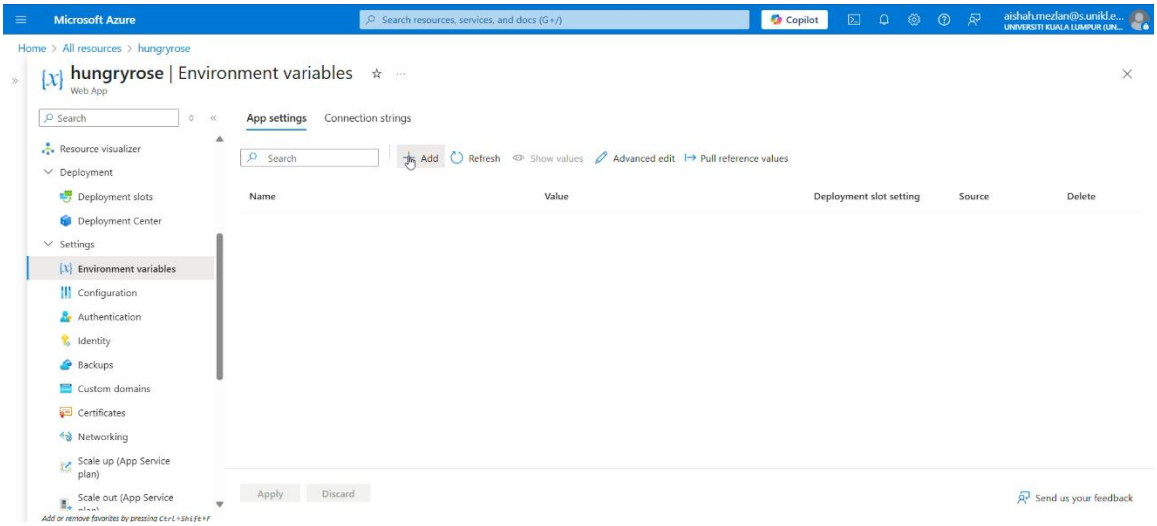
Before deploying version 2 of the web app, a slot setting should be configured. While these settings will not affect the demo app, this exercise is designed to demonstrate how configurations behave during slot swaps.

To configure slot settings:

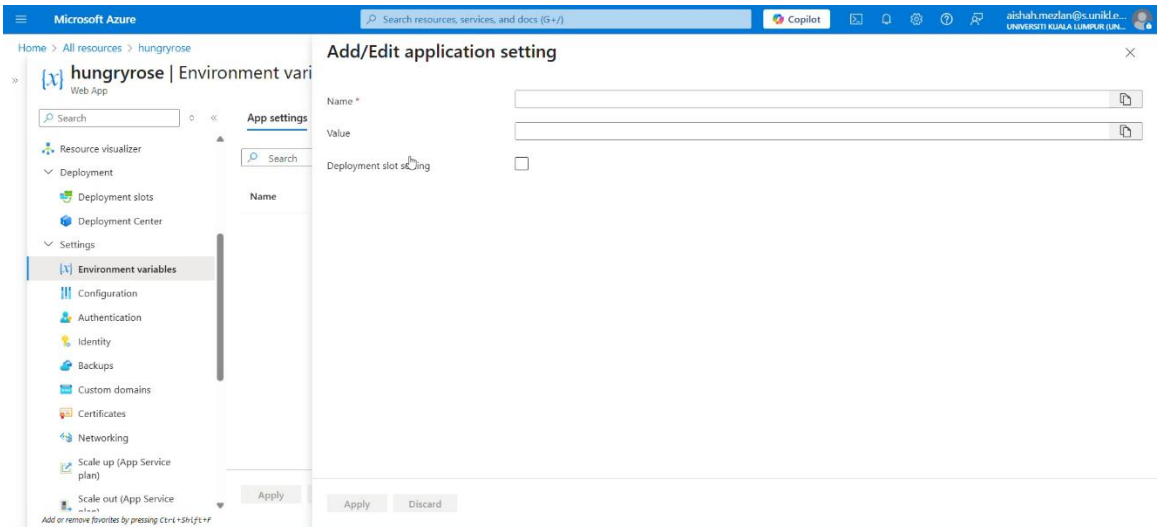
1. Select All resources in the Azure portal, and navigate the Overview page of the production slot of the web app (the main web app).



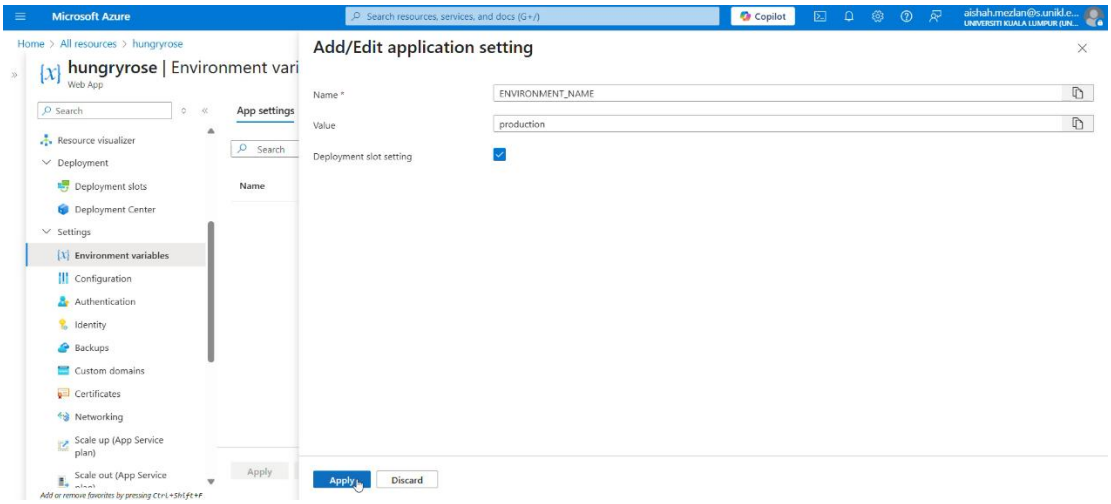
2. Under Settings, select Environment Variables



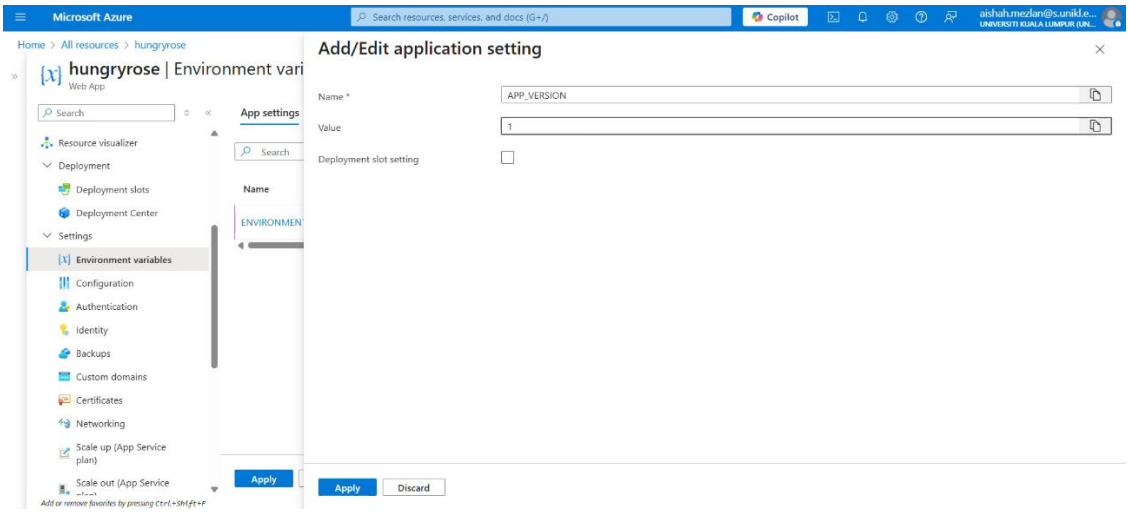
3. In the App settings section, click Add. Add/Edit application setting pane appears.



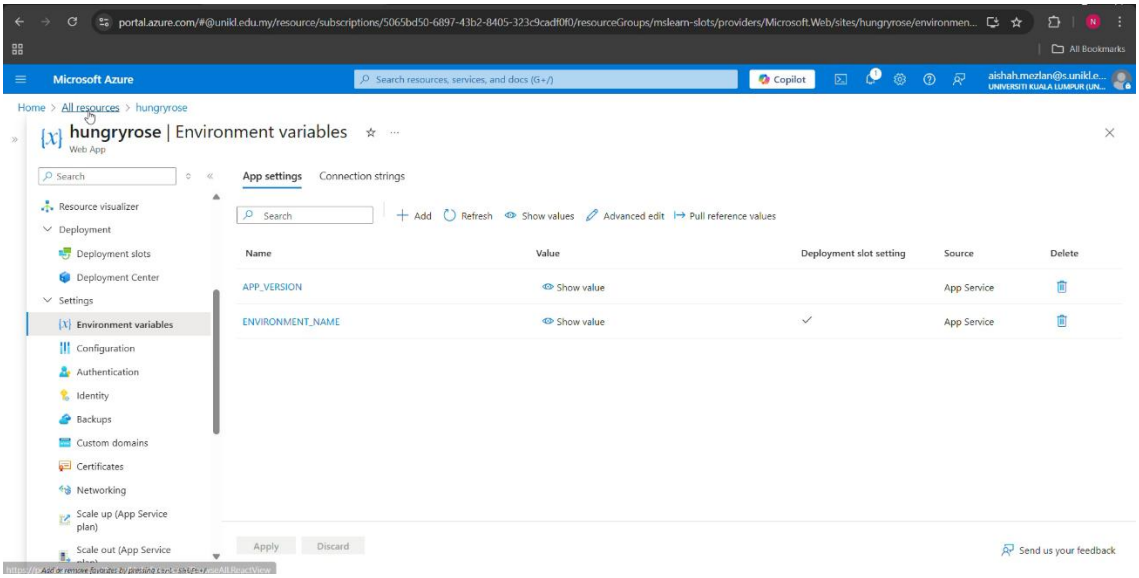
4. Create a new setting named ENVIRONMENT_NAME with the value production**.** Make sure to check the deployment slot setting box to designate it as a slot-specific setting.



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

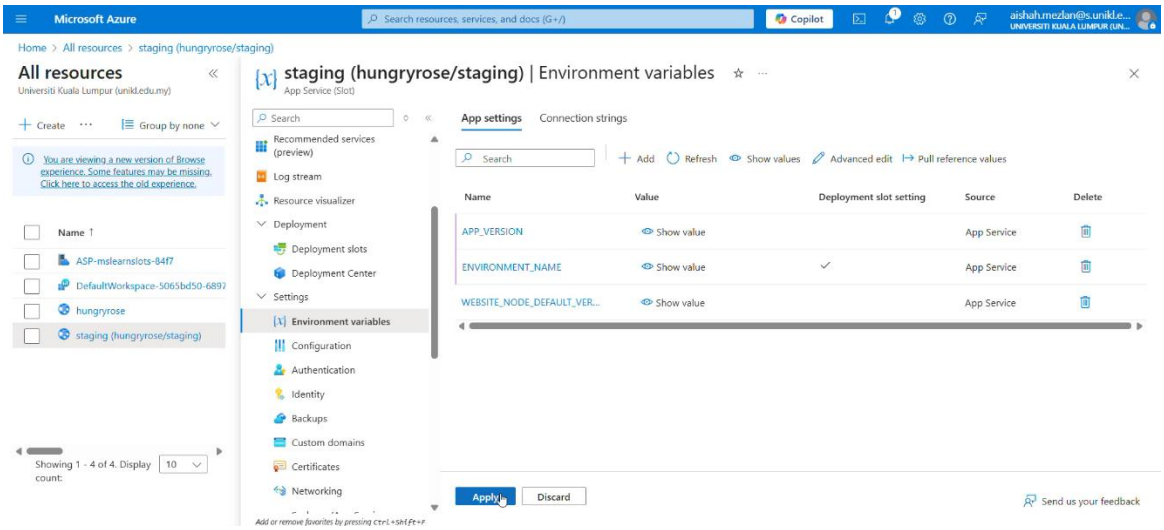


6. After both are done, click Apply to save changes.



7. Repeat the steps on the Staging slot, but use the following values

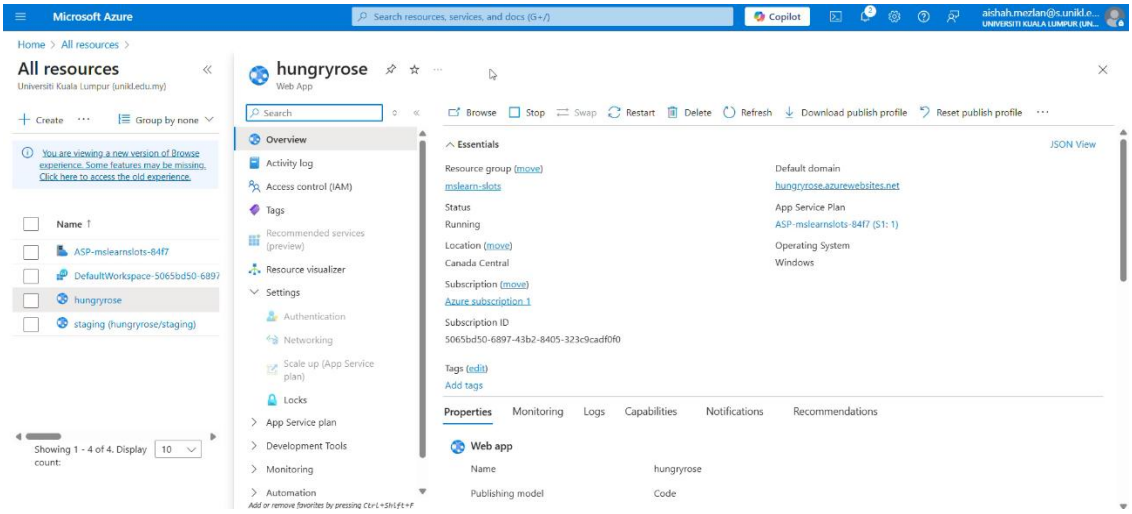
Name	Value	Deployment slot setting
ENVIRONMENT_NAME	staging	Yes
APP_VERSION	2	No



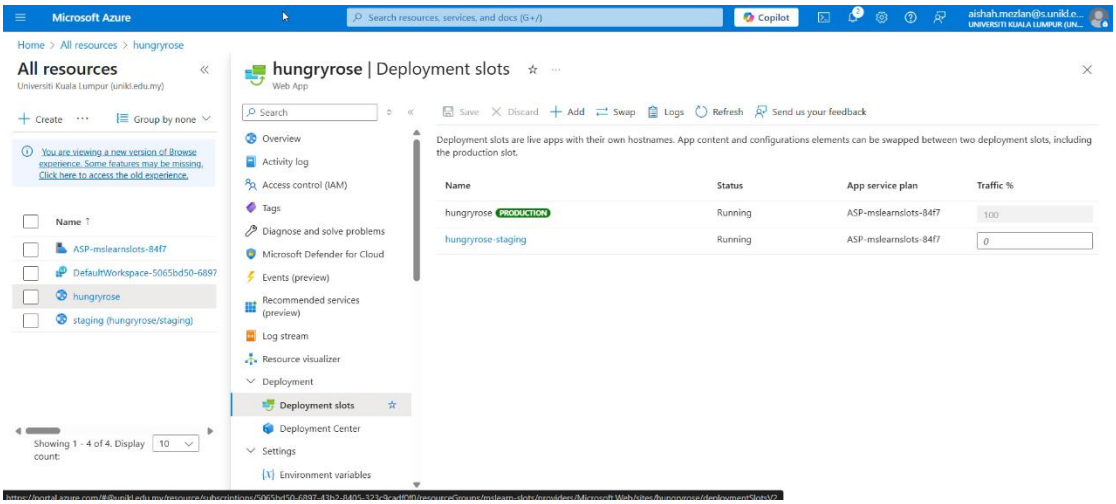
Swap the slots

After testing version 2 of the web app in the staging slot, it can be deployed by swapping the slots. Follow the steps below to proceed:

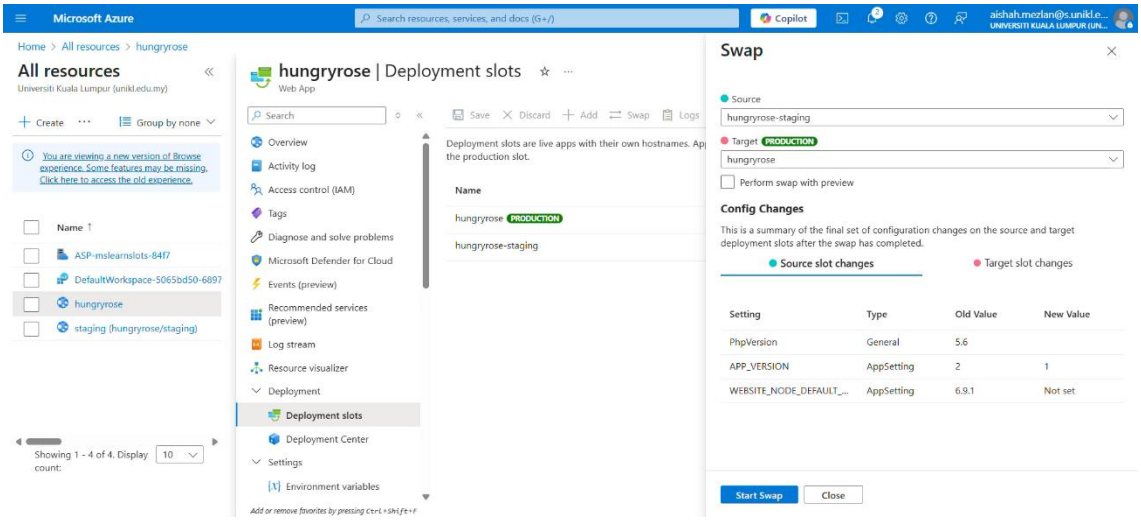
1. To ensure you are configuring the production slot, go to All resources and then select the production slot of your web app.



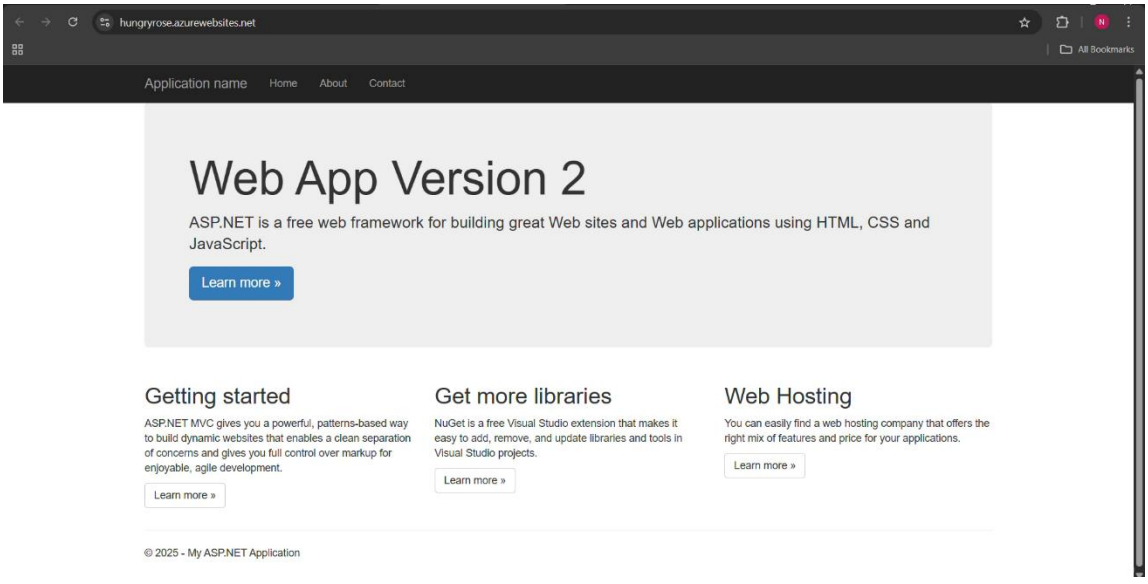
2. In the left menu pane, under Deployment, select Deployment slots. On the top, click Swap



3. Confirm that you are swapping the staging and production slots. Notice how the swap will impact settings: the APP_VERSION setting value will be exchanged between the slots, but the ENVIRONMENT_NAME slot setting will remain unchanged. Then, click Start Swap.



4. Once the swap is finished, navigate to the Overview page of the production slot's web app and click **Browse**. The web app will open in a new browser tab, showing that version 2 is now live in production.

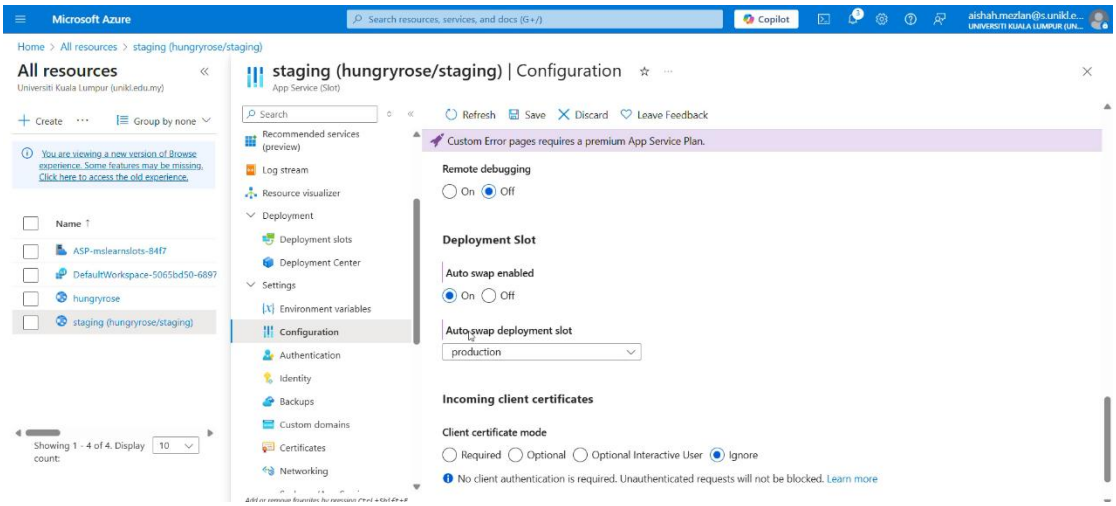


5. Close the browser tab

Configure auto swap for the staging slot

With deployment slots in use, enabling continuous deployment can be achieved by turning on the auto swap feature for the web app. When auto swap is activated, deploying new code to the staging slot prompts Azure to automatically warm it up and then swap the staging and production slots, deploying the update to production. 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. Then, select production in the Auto swap deployment list, and then select Save.



Deploy new code and auto swap it into production

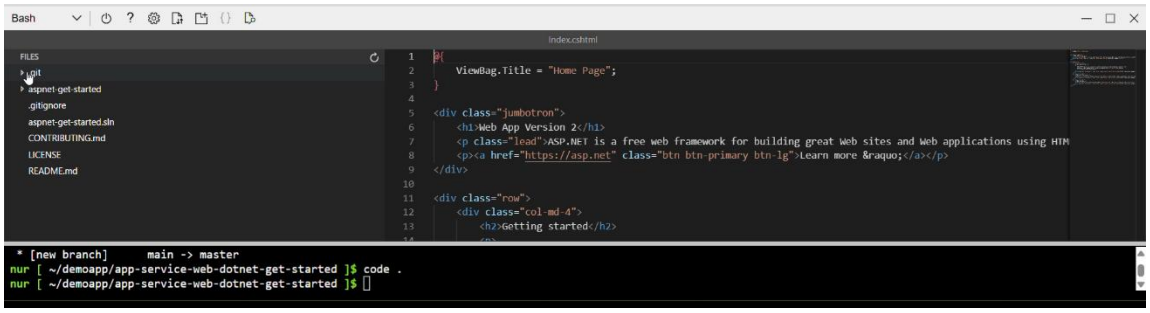
Next, the code will be updated to create version 3 of the web app. Deploying this version to the staging slot will allow observation of the auto swap feature in action. Follow these steps:

1. Open Cloud Shell and execute the following commands.

   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.

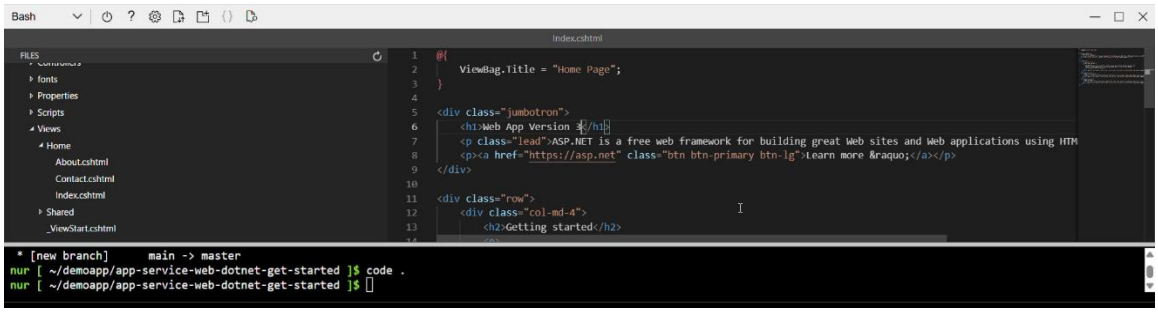


3. Locate the following code.

   <h1>Web App Version 2</h1>

4. Replace that code with this code.

   <h1>Web App Version 3</h1>

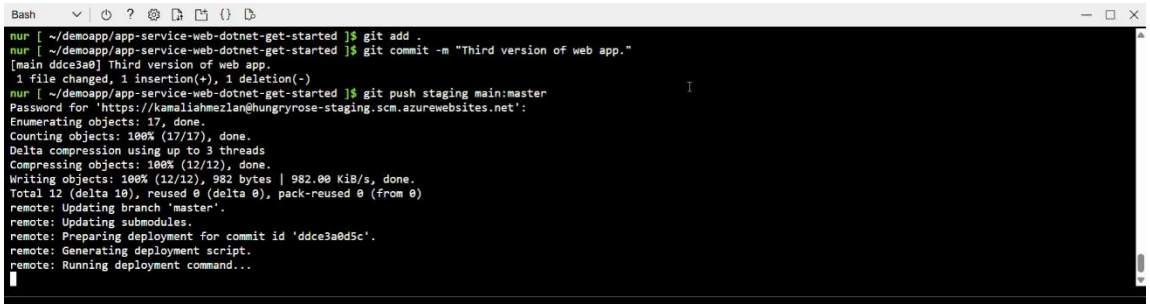


5. To save your changes, press Ctrl+S
6. In Cloud Shell, enter the following commands. Enter your deployment password when you're prompted.

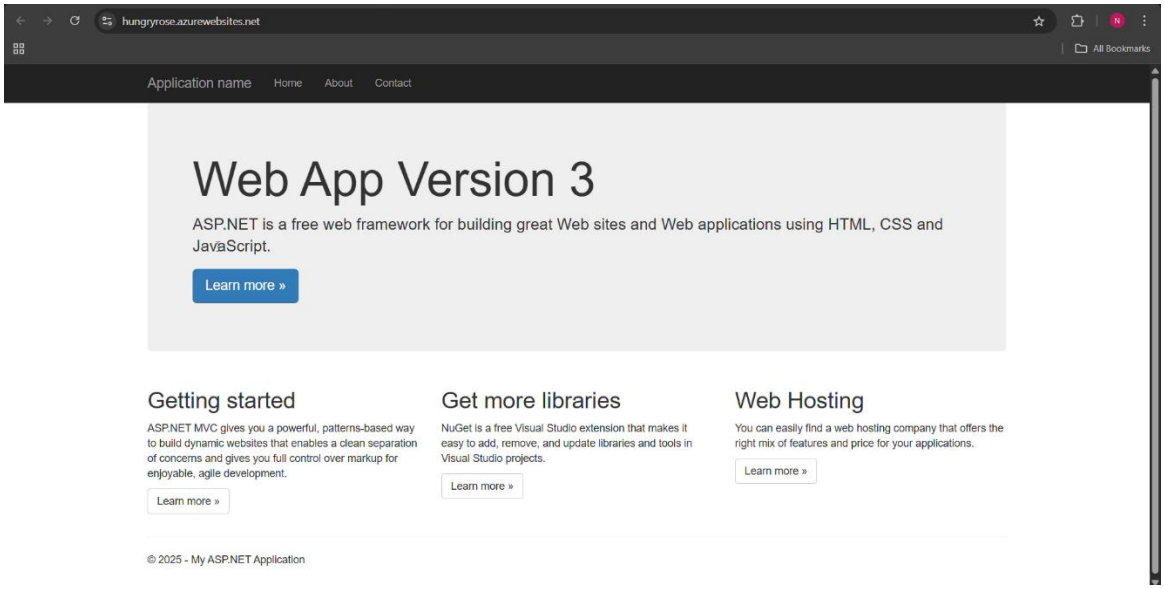
   git add .

   git commit -m “Third version of tweb app.”

   git push staging main:master



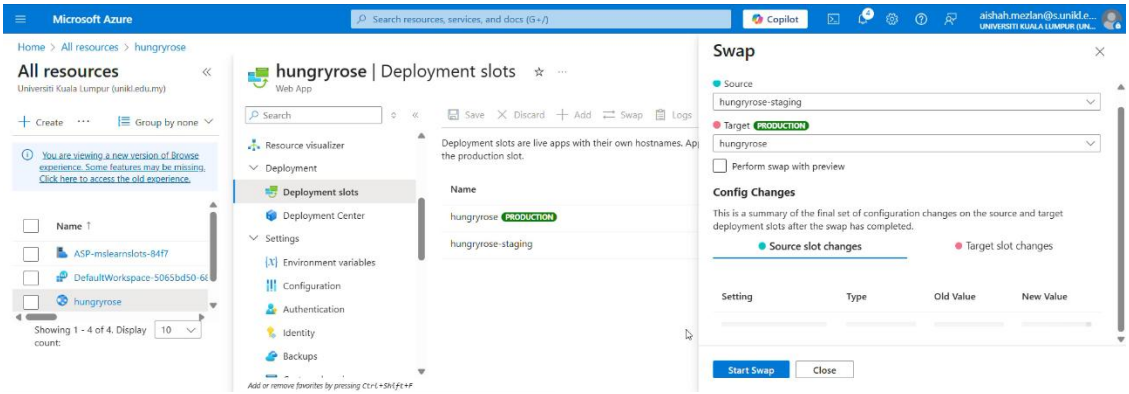
7. Wait for the deployment to complete. Near the end of the output, you’ll see a message indicating that an auto swap to the production slot has been requested.
8. In the Azure portal, go to the Overview page of the production slot’s web app and click Browse. Version 3 of the web app should open in a new browser tab. If you still see the previous version, wait a moment and refresh the page. Although the swap is atomic and happens instantly, App Service may take a few moments to prepare and execute the operation.



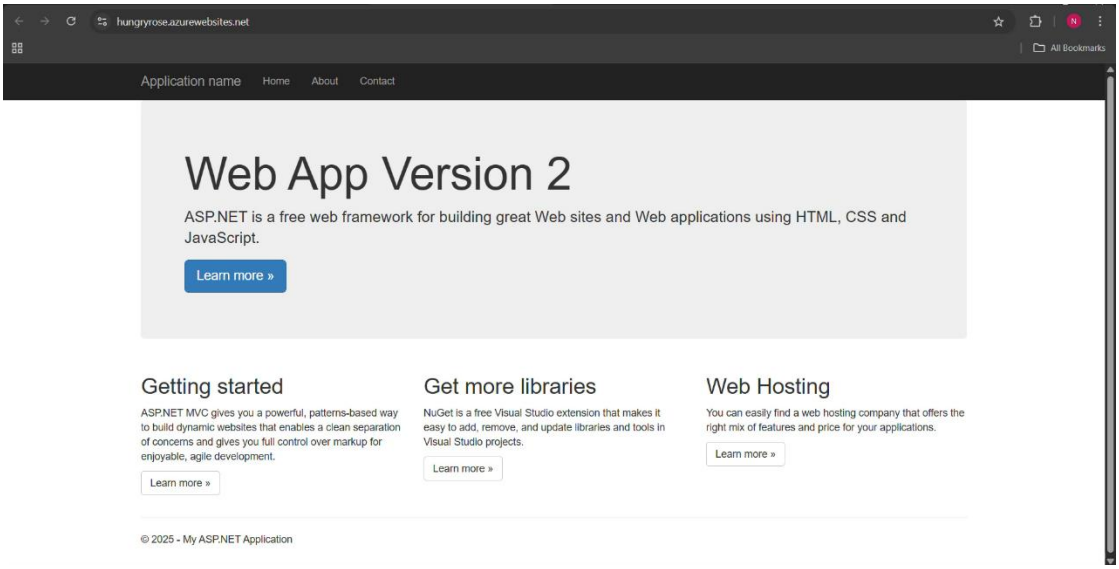
Roll back the new version

If deploying version 3 of the app to production causes an unexpected issue, it can be quickly resolved by rolling back to the previous version through swapping the deployment slots again.

1. In the Azure portal, go to the Deployment slots page of the production slot's web app.
2. Swap the staging and production slots.



3. Once the swap is complete, go to the Overview page and select Browse to view the app. You'll see that version 2 has been successfully redeployed to production.



Lab 31: Scale a web app manually

A system should be scaled out when an increase in traffic is expected or when there is a decline in performance.

In the case of a hotel reservation system, the number of web app instances may be increased to accommodate extra traffic caused by special events, promotional offers, or seasonal fluctuations. Once demand decreases, the system can be scaled back accordingly.

In this section, an App Service plan will be created, and a web app will be deployed using that plan. The performance of the web app will be monitored under load. Afterward, the app will be scaled out, and the improved performance will be verified.

This section uses a sample web app that implements a web API, which includes HTTP POST and GET operations to create and retrieve customer bookings for a hotel reservation website. These bookings are not saved—GET operations return dummy data.

A client app will also be run to simulate multiple users sending POST and GET requests simultaneously. This provides the workload needed to test the web app's performance before and after scaling.

Create an App Service plan and web app

An individual Azure subscription is required to complete this section, and there may be associated charges. If an Azure subscription is not already available, a free account should be created before beginning.

1. In the Azure portal, from the menu or the Home page, select Create a resource. The Create a resource pane will appear.
2. In the menu under Categories, select Web, then search for and select Web App, and click Create. The Create Web App pane will open.
3. On the Basics tab, provide the following values for each setting as required.

	Setting	Value
Project Details	Subscription	Select the Azure subscription you'd like to use for this exercise.
	Resource Group	Create a new resource group called mslearn-scale.
Instance Details	Name	Enter a unique name that you can remember for later in this exercise.
	Publish	Code
	Runtime Stack	.NET Core 3.1 (LTS)
	Operating System	Windows
	Region	Leave default
App Service Plan	App Service plan	Leave default
	Location	Leave default

4. Select Review + Create, and when validation completes, select Create. Wait for the web app to be created and deployed.

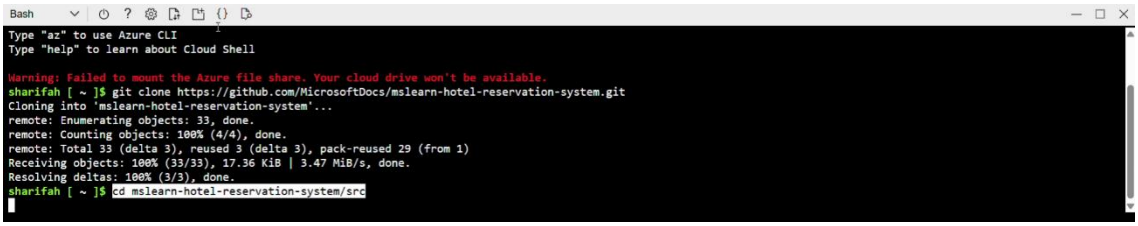
Build and deploy the web app

1. Open Cloud Shell and run the following command to download the source code for the hotel reservation system.

   git clone https://github.com/MicrosoftDocs/mslearn-hotel-reservation-system.git

2. Go to mslearn-hotel-reservation-system/src folder.

   cd mslearn-hotel-reservation-system/src



3. Build the applications for the hotel reservation system. There are two components:

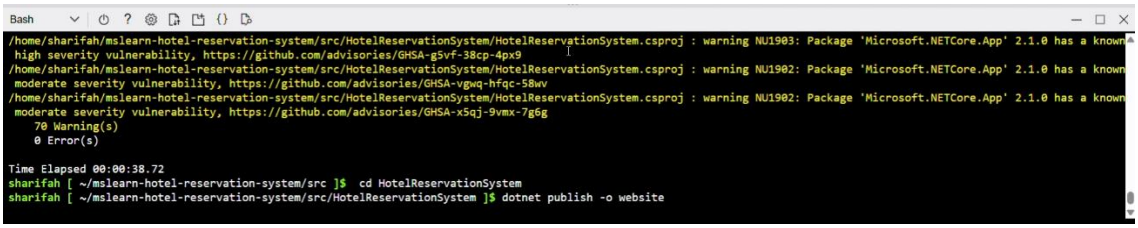
   - A **web app** that implements the web API for the system.
   - A **client app** used to perform load testing on the web app.

   dotnet build

4. Prepare the HotelReservationSystem web app for publishing.

   cd HotelReservationSystem

   dotnet publish -o website

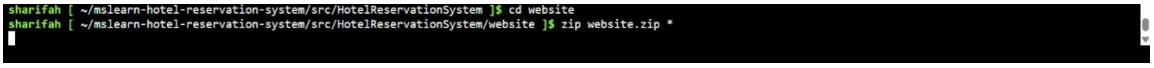


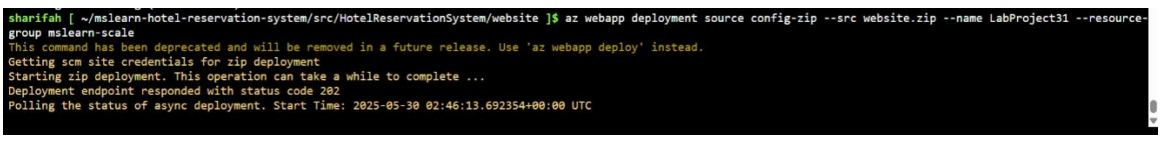
5. Go to the folder that holds the published website files. Compress these files into a ZIP archive. Then, upload the ZIP file to the web app you set up earlier. Copy the code below into Notepad, replace <your-webapp-name> with your actual web app name, and then paste the updated code into Cloud Shell to run it.

   cd website

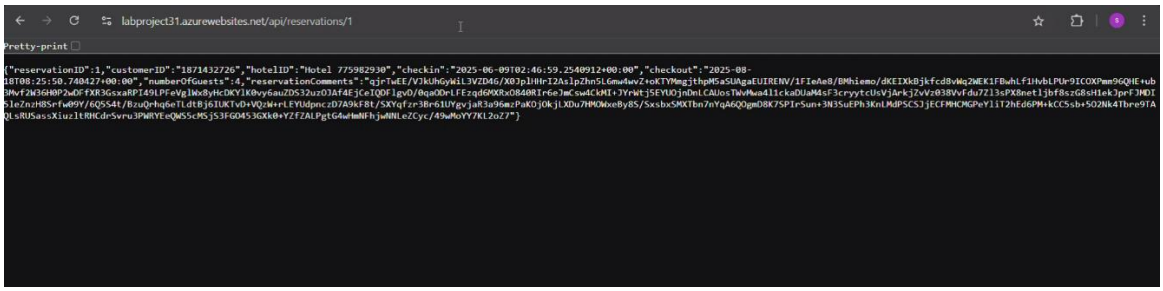
   zip website.zip *

   az webapp deployment source config-zip --src website.zip --name YOUR-WEBAPP-NAME --resource-group mslearn-scale





6. Open your web browser and navigate to http://<your-webapp-name>.azurewebsites.net/api/reservations/1. You should see a JSON document displaying the information for reservation number 1.

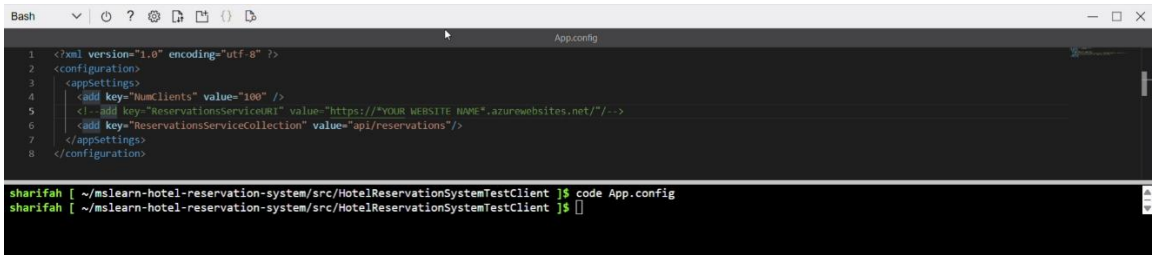


Monitor the performance of the web app before scaling out

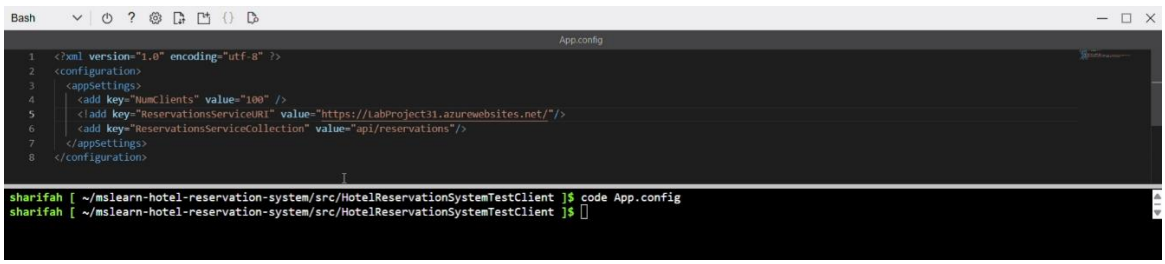
1. Return to the Cloud Shell and go to the *~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient* folder.

   cd ~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient
2. Edit the App.config file in this folder by using the code editor.

   code App.config



3. Uncomment the line that specifies the ReservationsServiceURI and replace the value *YOUR WEBSITE NAME* with the name of your web app.

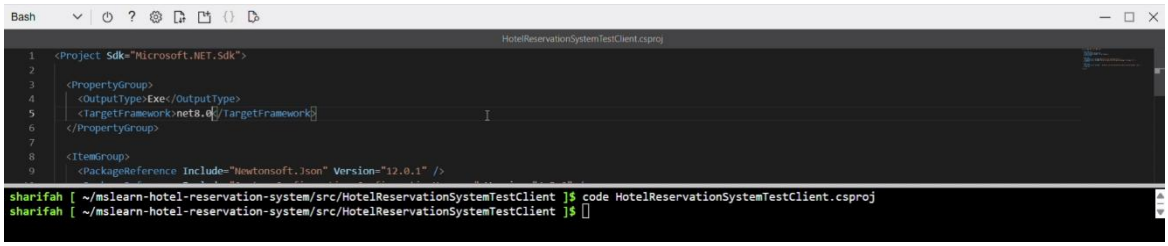


4. Save the file (Ctrl+S) and close the code editor (Ctrl+Q)
5. Rebuild the test client app with the new configuration

   dotnet build

6. Edit the HotelReservationSystemTestClient.csproj file to use the correct framework.

   code HotelReservationSystemTestClient.csproj

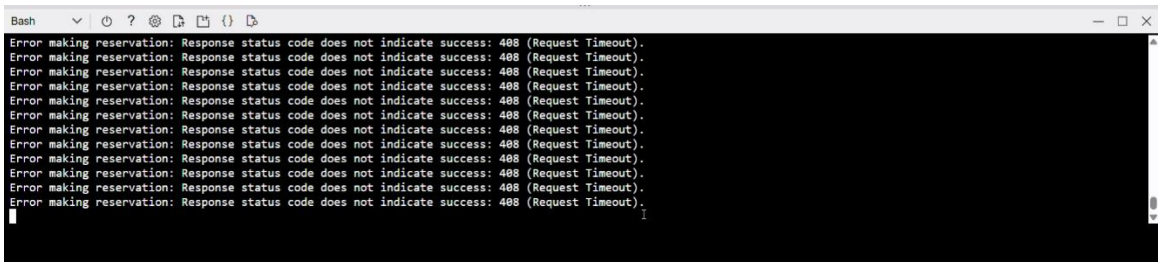


   In this case, net8.0 is being used instead of netcore2.1.

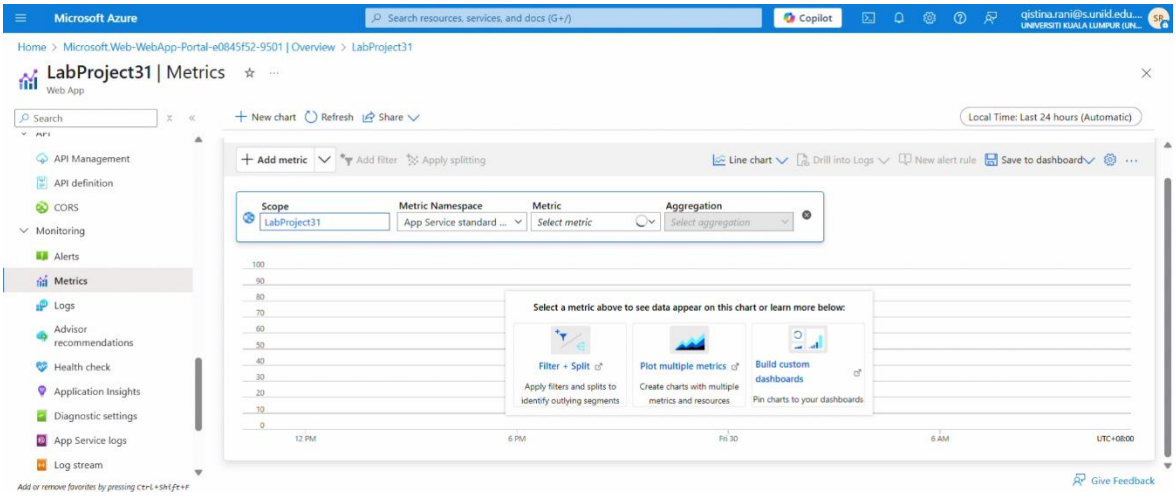
7. Save the file (Ctrl+S) and close the code editor (Ctrl+Q)
8. Run the client app

   dotnet run

   Several messages will appear as the clients begin running, making reservations, and sending queries. Let the system run for a few minutes. You'll notice that the responses become slow, and the client requests will eventually start failing with HTTP 408 (Timeout) errors.

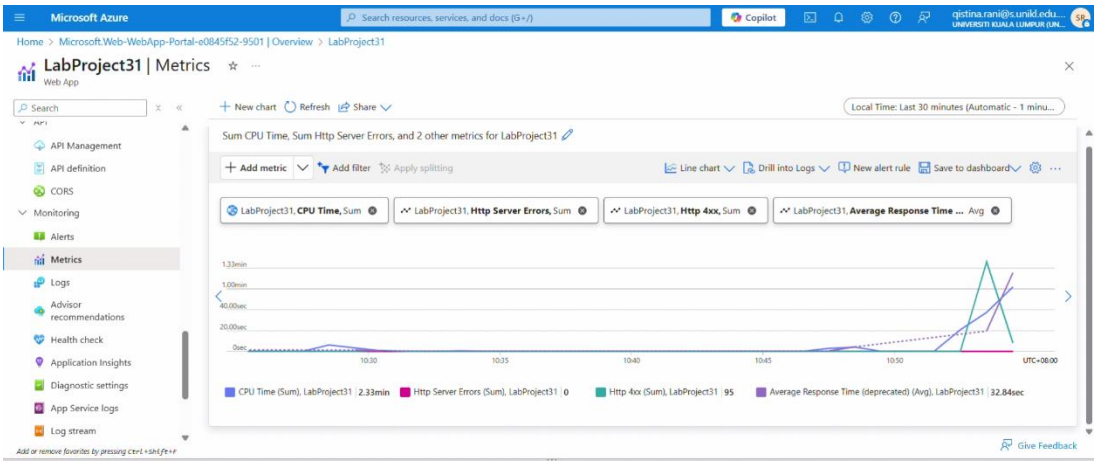


9. In the Azure portal, click Go to resource to open your web app.
10. In the left-hand menu, scroll down to the Monitoring section and select Metrics.
11. In the top menu bar of the Metrics page, click the time range option that says Local Time: Last 24 hours (Automatic). Change it to Last 30 minutes, then click Apply.

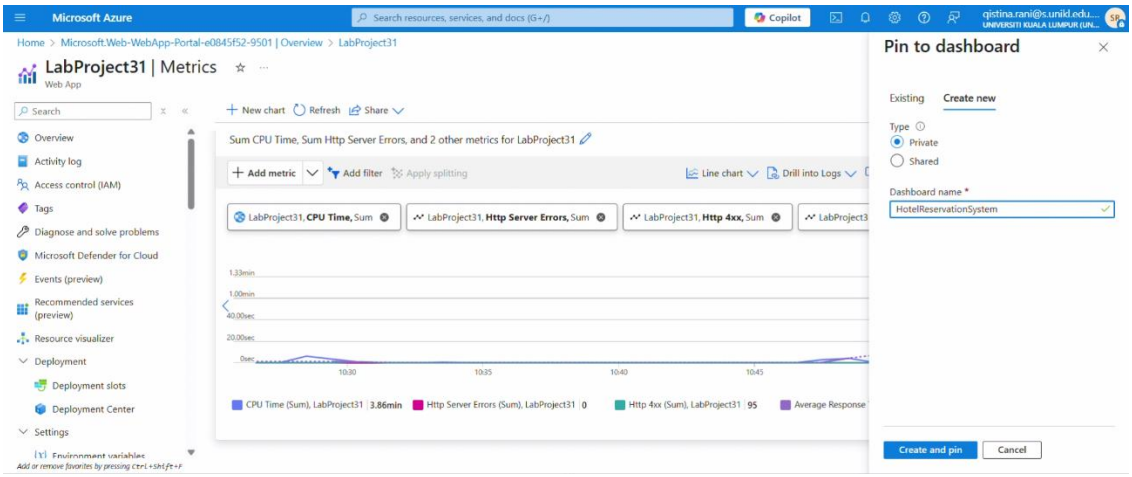


12. In the menu under Chart Title, add the following metrics to the chart.

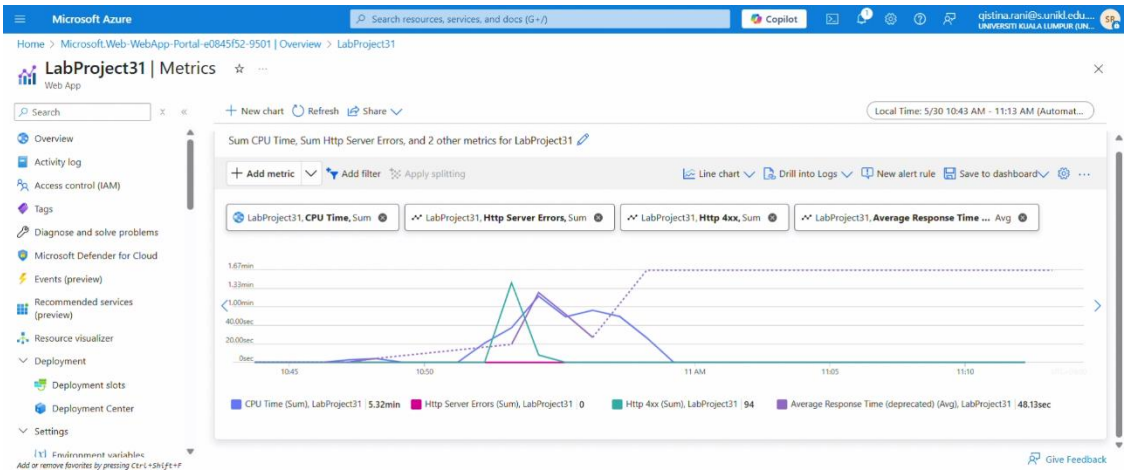
- Select Add metric CPU Time. Select the Sum aggregation.
- Select Add metric Http Server Errors. Select the Sum aggregation.
- Select Add metric Http 4xx. Select the Sum aggregation.
- Select Add metric Average Response Time. Select the Sum aggregation.



13. In the menu under Chart Title, select Pin to dashboard, then click Pin. Create dashboard first if you don't have any.



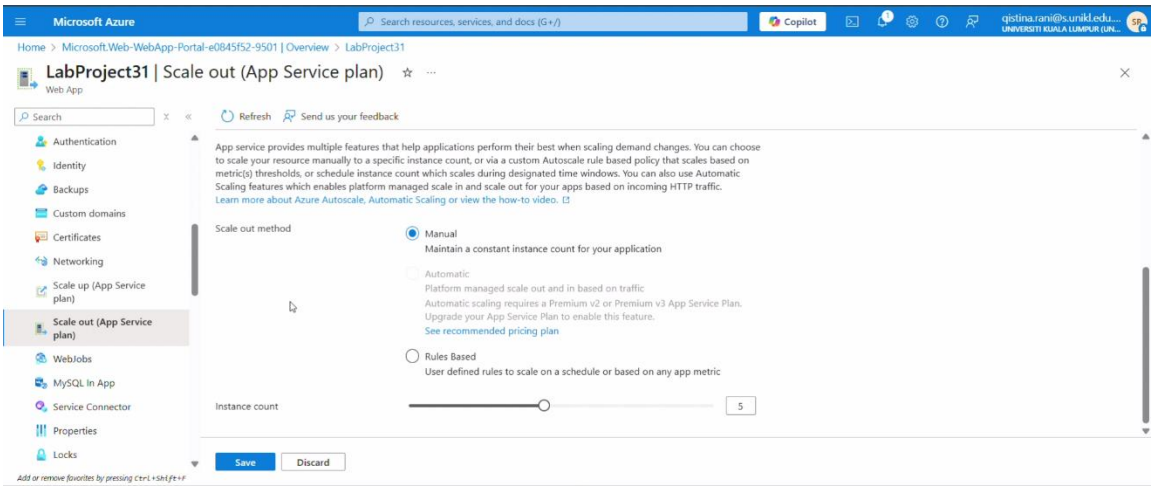
14. Let the system run for about five minutes to stabilize. After that, take note of the CPU Time, the number of HTTP 4xx errors, and the average response time. You should observe a high number of HTTP 4xx errors—specifically, HTTP 408 Timeout errors—and an average response time of several seconds. Depending on how well the web server is handling the load, you might also see occasional HTTP server errors.



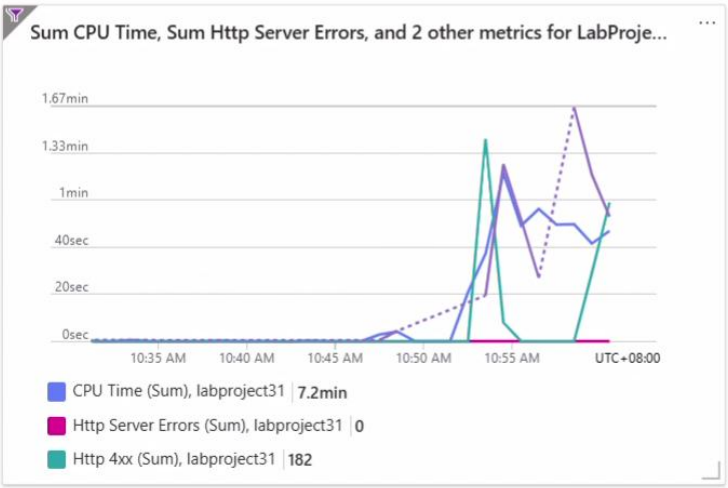
15. Leave the client app running while you perform the next task.

Scale out the web app and verify the performance improvement

1. In the Azure portal, click on your web app name under App Services. In the menu, under Settings, select Scale-out (App Service Plan).
2. Change the Instance count to 5, then click Save.

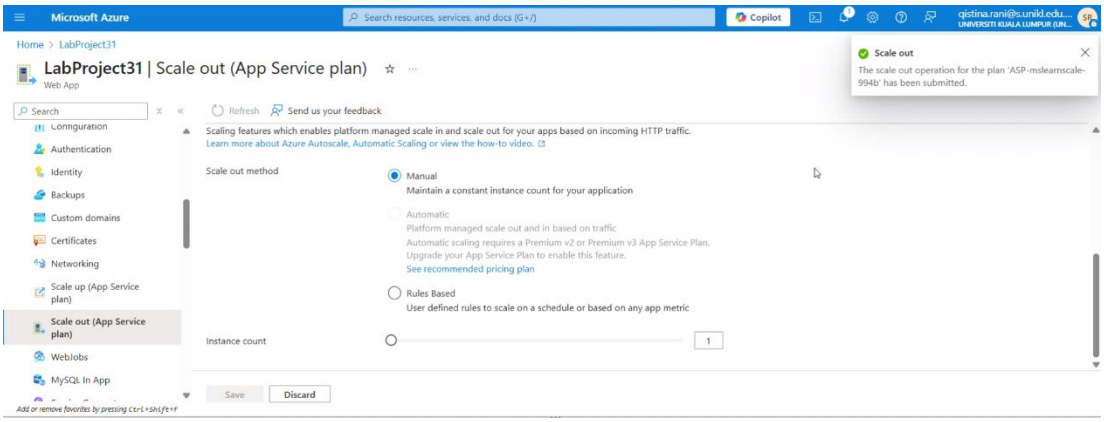


3. Open Cloud Shell. You should notice a decrease in the number of requests failing with errors, although some requests may still time out.
4. Let the app run for another five minutes. Then, go to the dashboard in the Azure portal and view the chart displaying the app's metrics. You should observe a significant increase in CPU time, reflecting the added processing power from scaling out to five instances. The average response time should have decreased, and the number of HTTP 4xx errors should have gone down as well. The chart will typically indicate the point at which the scale-out took place.



5. If you'd like to explore further, try increasing the instance count of your App Service plan to 10, which is the maximum supported by the S1 pricing tier. You should see an additional rise in CPU time, along with a reduction in response time and fewer HTTP 4xx errors.
6. When you're done, return to the Cloud Shell running the client app and press Enter to stop it.
7. Then, in the Azure portal, reset the instance count to 1:

    - Select your web app.
    - In the menu, choose Scale out (App Service plan).
    - On the Configure tab, set the instance count to 1.
    - Click Save in the menu bar at the top of the App Service pane.



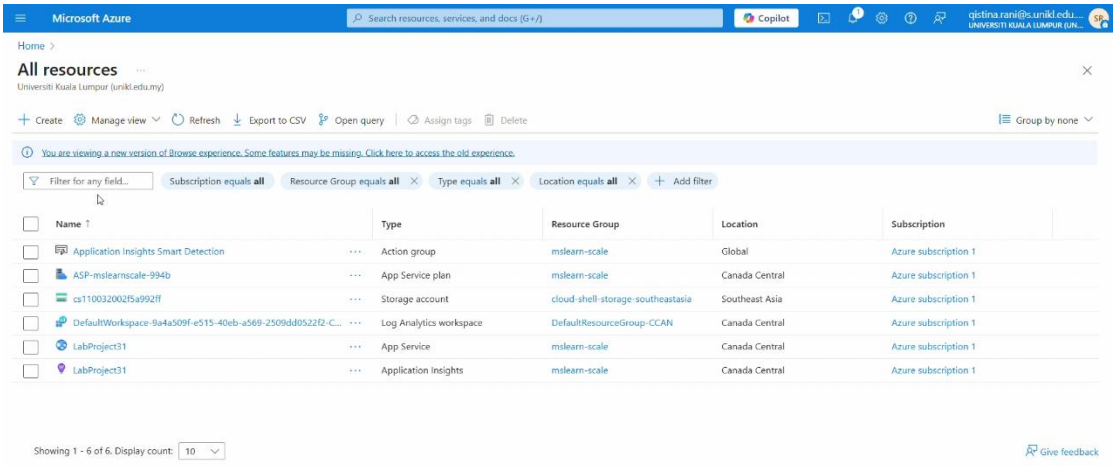
Lab 32: Scale up a web app

Scaling up provides more powerful resources for running a web app and also increases the number of instances available for scaling out. In the hotel reservation system, scaling out is necessary to accommodate a growing number of visitors. Scaling up enables further scaling out and is likely required to support any new functionality added to the web app. In this section, the hotel reservation system web app that was previously deployed will be scaled up. The same test client application used earlier will be run again to simulate user activity, and the web app's performance will be monitored.

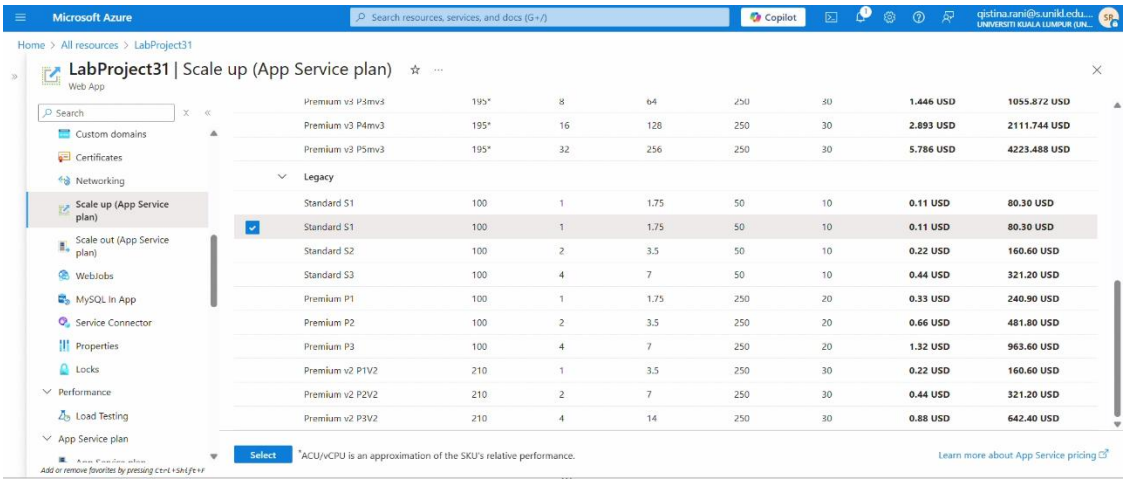
Examine the current pricing tier for the web app

An individual Azure subscription is required to complete this section, and there may be associated charges. If an Azure subscription is not already available, a free account should be created before beginning.

1. In the Azure portal, from the menu or the Home page, select All resources, then navigate to your App Service plan.



2. Under Settings, select Scale up (App Service plan). This will display the details of the current pricing tier. The plan is set to S1, which offers 100 Azure Compute Units and 1.75 GB of memory, running on an A-Series virtual machine.



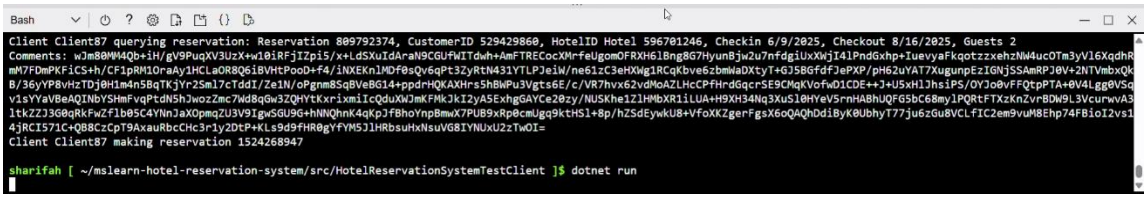
Run the test client app

1. In the Cloud Shell, go to the ~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient folder:

   cd ~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient

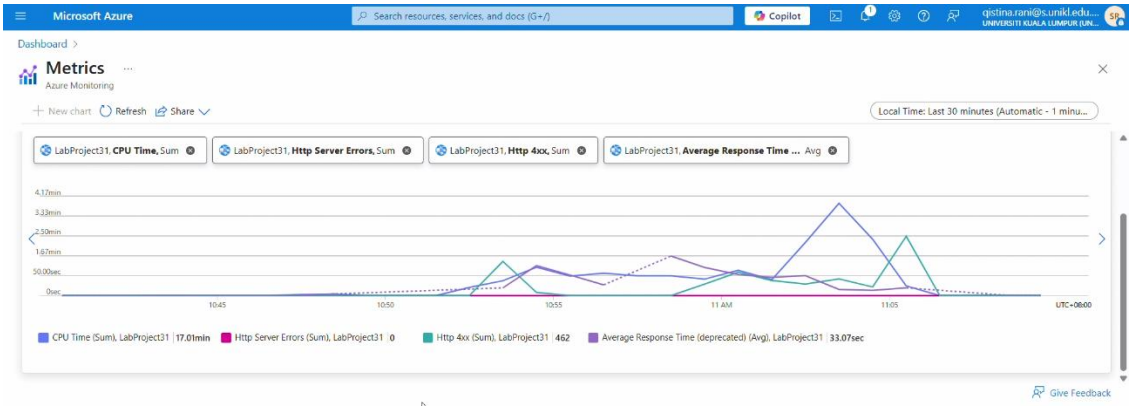
2. Run the client app:

   dotnet run



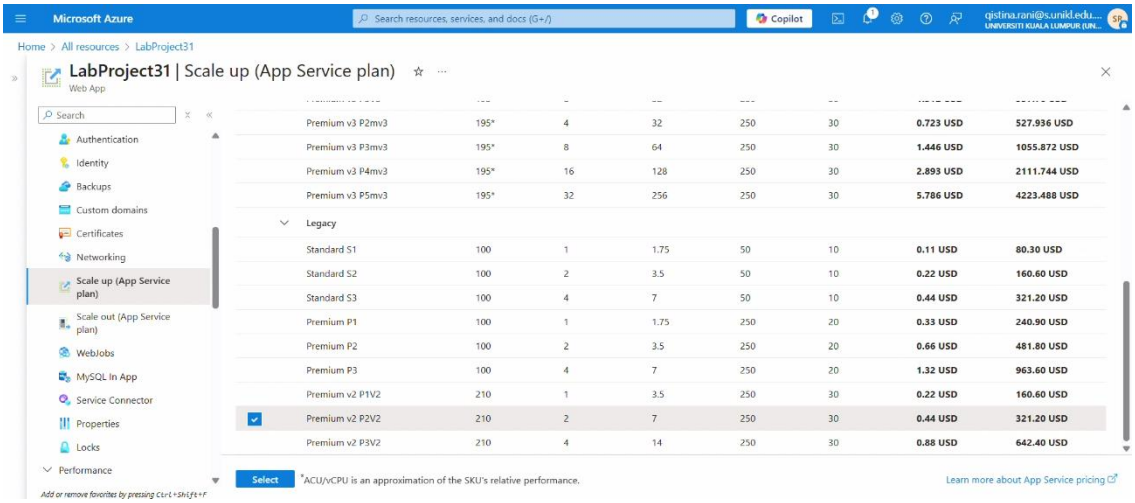
Let the system run for a few minutes. As in the beginning of the previous exercise, the responses will be slow, and the client requests will soon begin to fail with HTTP 408 (Timeout) errors.

3. Continue running the app and wait for another five minutes. Then, navigate to the chart displaying the web app metrics on the Azure portal dashboard. As observed in the previous section, the statistics should show relatively slow response times and a high number of HTTP 4xx errors.

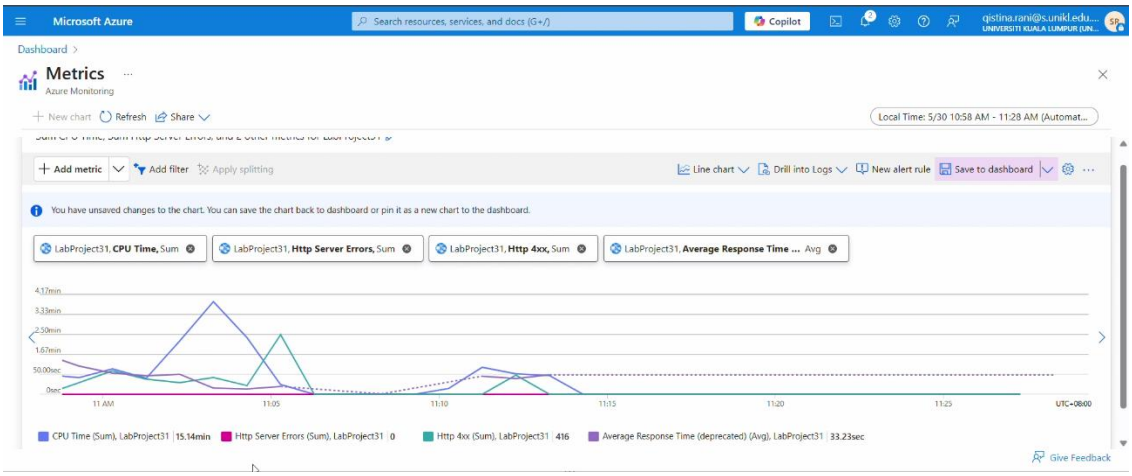


Scale up the web app and monitor the results

1. In the Azure portal, go back to the page for the App Service plan.
2. Under Settings, select Scale up (App Service plan)**.**
3. Choose the P2V2 pricing tier and click Apply**.** This tier provides 420 Azure Compute Units (ACU)—more than four times the power of the S1 tier—and 7 GB of memory, running on a Dv2-Series virtual machine. However, it costs approximately four times more than the S1 pricing tier.



4. After scaling up, wait for another five minutes, then check the performance chart on the dashboard in the Azure portal.



5. Return to the Cloud Shell that's running the client app. Select Enter to stop the app.

Link to our YouTube video: https://youtu.be/Fxcf24ehqeA?si=ZW5AbQzCrx6L3v7w