



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 (HONOURS)
(INTERNET OF THINGS)
Course Code & Name : IIB43203 - CLOUD COMPUTING
Course Lecturer(s) : MEGAT NORULAZMI MEGAT MOHAMED NOOR
Type of Submission : **ASSIGNMENT**
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:

- | | |
|-------------------------------------|--|
| <input checked="" type="checkbox"/> | I/You have read and understood the UniKLs' policy on Plagiarism in University Rules & Regulations. |
| <input checked="" type="checkbox"/> | This assignment is own work, unless indicated with proper referencing. |
| <input checked="" type="checkbox"/> | This assignment not submitted and not published previously. |
| <input checked="" type="checkbox"/> | 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)
2 nd JUNE 2025	EVAN HAMO	52200225007
	MUHAMMAD UWAIS BIN MOHD	52224123579
	IMRAN	
	MUHAMMAD IZWAN BIN HUSSIN	52224123217
	MUHAMMAD HAZMAN BIN MOHD	52224123618
	SO FEE	

Student Receipt of Submission

This is your submission receipt, the only accepted evidence that you have submitted your work. Cut along the dotted lines above & retain this for **your record**.

Date & Time of Submission (Written)	Course Code	Submission Title	Student ID(s) & Signature(s)

Table of Contents

Introduction	4
Create deployment slots	5
Deploy a web app by using deployment slots	19
Scale a web app manually	25
Scale-Out 1 vs 5 Instance	28
Scale up the pricing plan	29
Conclusion	31

Introduction

Cloud computing is a modern technology that allows users to access computing resources such as servers, storage, databases, networking, software, and more over the internet. Instead of owning and maintaining physical hardware, users can rent these resources from cloud service providers based on their needs. This approach offers greater flexibility, scalability, and cost-efficiency, making it popular among businesses, developers, and students for hosting applications, storing data, and running services.

One of the most widely used cloud platforms is Microsoft Azure, which provides a wide range of cloud services to build, deploy, and manage applications through a global network of data centers. The Azure Portal is a web-based interface that allows users to manage their cloud resources easily. Through the portal, users can create virtual machines, databases, web applications, and more with just a few clicks. It also offers monitoring tools, security features, and pricing plan options, making it a powerful and user-friendly platform for cloud computing tasks.

Create deployment slots


Azure Lab #30A


Name: Izwan Hussin


➤ Create a web app


Getting started? [Try our Quickstart Center](#)


Popular Azure services [See more in All services](#)


Function App
[Create](#)


Web App
[Create](#)


Azure AI services
[Create](#)


Azure OpenAI
[Create](#)


Virtual network
[Create](#)

SQL Database
[Create](#)


Key Vault
[Create](#)


Virtual machine
[Create](#)


Storage account
[Create](#)


Data Factory
[Create](#)


Popular Marketplace products [See more in Marketplace](#)


Windows Server 2022 Datacenter: Azure Edition Hotpatch
[Create](#) | [Learn more](#)


Windows 10 Pro, version 22H2
[Create](#) | [Learn more](#)


Ubuntu Server 22.04 LTS
[Create](#) | [Learn more](#)


Free SQL Server License: SQL Server 2022 Developer on Windows Server 2022
[Create](#) | [Learn more](#)


Red Hat Enterprise Linux10 (latest minor version)
[Create](#) | [Learn more](#)

Debian 12 "Bookworm"
[Create](#) | [Learn more](#)

Visual Studio 2022 Pro on Windows 10 Enterprise (x64) + Microsoft 365 Apps (Microsoft Dev Box compatible)
[Create](#) | [Learn more](#)

Oracle Linux 9.4 (LVM)
[Create](#) | [Learn more](#)

CentOS-based 8.2
[Create](#) | [Learn more](#)

AlmaLinux OS 9
[Create](#) | [Learn more](#)

To begin, I signed in to the Azure Portal by visiting <https://portal.azure.com> and logging in with my Azure account. Once logged in, I started creating the web app by clicking on “Create a resource” from the Home or the left-hand menu. Then, I selected the “Web” category, searched for “Web App,” and clicked on it. After that, I proceeded by selecting the “Create” button to begin configuring the web app.

Microsoft Azure

Upgrade


Search resources, services, and docs (G+)


[Home](#) > [Create a resource](#) >

Create Web App ...

BasicsDatabaseDeploymentNetworkingMonitor + secureTagsReview + create

Summary

 **Web App**
by Microsoft

 Basic authentication for this app is currently disabled and may impact deployments. [Click to learn more.](#)

Details

Subscription	318605cf-a1d3-423e-be18-b3c30544fdf0
Resource Group	mslearn-slots2
Name	izwanwebapp
Secure unique default hostname	Enabled
Publish	Code
Runtime stack	ASP.NET V4.8

App Service Plan

Name	ASP-mslearnslots-8a6e
Operating System	Windows
Region	Southeast Asia
SKU	Standard
Size	Small
ACU	100 total ACU
Memory	1.75 GB memory

Monitor + secure (New)

Application Insights	Enabled
Name	izwanwebapp
Region	Southeast Asia

Create< PreviousNext >Download a template for automation

In the Basics tab of the Web App creation process, I began by selecting my Azure subscription and creating a new resource group named mslearn-slots. For the name of the web app, I entered a unique identifier, such as izwanwebapp, to ensure it wouldn't conflict with existing names. I chose "Code" for the publish method, selected "ASP.NET V4.8" as the runtime stack, and set the operating system to Windows. For the region, I picked the one closest to my location to reduce latency. I accepted the default App Service Plan (Windows Plan) and kept the default SKU and size settings.

After completing the Basics tab, I moved to the Deployment tab by clicking "Next: Deployment (Preview)," where I made no changes. I then advanced to the Monitoring tab and turned off Application Insights by toggling it to "No." Finally, I proceeded to the Review + Create tab, where I reviewed all the configurations. After validation was successful, I clicked "Create" to deploy the web app.

Delete Cancel Redeploy Download Refresh

Your deployment is complete

Deployment name: Microsoft.Web-WebApp-Portal-9a266cab... Start time: 6/2/2025, 9:27:12 PM
Subscription: [Azure subscription 1](#) Correlation ID: 932245e8-9dd7-4084-8390-288e3891cb7c
Resource group: [mslearn-slots2](#)

Deployment details

Next steps

[Manage deployments for your app.](#) Recommended

[Protect your app with authentication.](#) Recommended

[Go to resource](#)

Give feedback

[Tell us about your experience with deployment](#)

The deployment is complete

Home > [Microsoft.Web-WebApp-Portal-9a266cab-9cb3 | Overview](#) > [izwanwebapp](#)

izwanwebapp | Deployment Center ☆ ...
Web App

Save Discard Browse Manage publish profile Sync Leave Feedback

Overview
 Activity log
 Access control (IAM)
 Tags
 Diagnose and solve problems
 Microsoft Defender for Cloud
 Events (preview)
 Recommended services (preview)
 Log stream
 Resource visualizer
 Deployment
 Deployment slots
 Deployment Center
 Settings
 Environment variables
 Configuration
 Authentication
 Identity

Settings * [Logs](#) [FTPS credentials](#)

You're now in the production slot, which is not recommended for setting up CI/CD. [Learn more](#)

Deploy and build code from your preferred source and build provider. [Learn more](#)

Source *

Building with App Service Build Service.

Local Git

Local Git allows you to host a simple Git server for your app on your App Service plan. This can be used to quickly setup a CI/CD pipeline. [Learn more](#)

Repository Your local git repository url will be generated upon completion.

Branch master

To configure the deployment method for the web app, I opened the Deployment Center by selecting it from the left-hand menu under the Deployment section. Once inside the Deployment Center, I navigated to the Settings tab where I selected "Local Git" as the deployment source. After choosing Local Git, I clicked the Save button at the top of the page to apply the changes.

The screenshot shows the Azure Portal interface for the 'izwanwebapp' Deployment Center. The left-hand navigation menu is visible, with 'Deployment Center' selected under the 'Deployment' section. The main content area shows the 'Settings' tab, which is further divided into 'Settings', 'Logs', and 'FTPS credentials'. The 'FTPS credentials' tab is active, displaying the following configuration:

- FTPS endpoint:** `https://waws-prod-sg1-009.ftp.azurewebsites.windows.net/site/wwwroot`
- Application scope:** A section explaining that application scope credentials are auto-generated and provide access only to this specific app or deployment slot. It notes that these credentials can be used with FTPS, Local Git, and WebDeploy, but cannot be configured manually.
- FTPS Username:** `izwanwebapp$izwanwebapp`
- Password:** A masked password field with a 'Reset' button.
- User scope:** A section explaining that user scope credentials are defined by the user and can be used with all the apps to which you have access. It notes that these credentials can be used with FTPS, Local Git, and WebDeploy. Authenticating to an FTPS endpoint using user-level credentials requires a username in the format 'izwanwebapp\hazman'. Authenticating with Git requires only the username 'hazman'.
- Username:** `izwan`
- Password:** `Takingat123_!` with a 'Reset' button.
- Confirm Password:** A masked password field.

Next, I set up the Git/FTP credentials required for deploying code. I navigated to the "Local Git/FTPS credentials" tab, and under the "User scope" section, I created a deployment username and set a secure password. After filling in these credentials, I clicked the Save button to store the settings and enable authentication for future deployments.

➤ Configure git deployment

```
Switch to PowerShell Restart Manage files New session Editor Web preview Settings Help
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

Your Cloud Shell session will be ephemeral so no files or system changes will persist beyond your current session.
muhammad [ ~ ]$ git config --global user.name "izwan"
git config --global user.email "izwan.hussin@unikl.edu.my"
muhammad [ ~ ]$
```

Then , I opened the Azure Cloud Shell by clicking the terminal icon (>_) located in the top navigation bar of the Azure portal. When prompted to choose a shell, I selected Bash. I configured Git to use my personal information, I put this command ,

- `git config --global user.name "your-username"`
- `git config --global user.email "your-email@example.com"`

I then replaced "your-username" and "your-email@example.com" with my actual Git username and email address. After updating, I copied the commands into the Cloud Shell and ran them by pressing Enter. This ensured that my Git commits would be properly attributed.

```
Switch to PowerShell Restart Manage files New session Editor Web preview Settings Help
Your Cloud Shell session will be ephemeral so no files or system changes will persist beyond your current session.
muhammad [ ~ ]$ git config --global user.name "izwan"
git config --global user.email "izwan.hussin@unikl.edu.my"
muhammad [ ~ ]$ mkdir demoapp
cd demoapp
muhammad [ ~/demoapp ]$ git clone https://github.com/Azure-Samples/app-service-web-dotnet-get-started.git
Cloning into 'app-service-web-dotnet-get-started'...
remote: Enumerating objects: 226, done.
remote: Total 226 (delta 0), reused 0 (delta 0), pack-reused 226 (from 1)
Receiving objects: 100% (226/226), 963.26 KiB | 22.40 MiB/s, done.
Resolving deltas: 100% (99/99), done.
muhammad [ ~/demoapp/app-service-web-dotnet-get-started ]$
```

To continue setting up the environment, I first created a new project folder by running the following commands in Azure Cloud Shell:

- `mkdir demoapp`
- `cd demoapp`

This created a directory named demoapp and moved me into it, providing a dedicated workspace for the web application.

Next, I cloned a sample ASP.NET web application by executing:

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

These commands downloaded the sample project from GitHub into my demoapp directory and navigated into the cloned project folder. This gave me access to the source code needed for deployment.

➤ **Configure a git remote to deploy the app to production**

```
muhammad [ ~/demoapp/app-service-web-dotnet-get-started ]$ git push production main:master
Password for 'https://izwan@izwanwebapp-e9g8epgzd3c2cugv.scm.southeastasia-01.azurewebsites.net':
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Updating branch 'master'.
remote: Updating submodules.
remote: Preparing deployment for commit id '212a655211'.
remote: Generating deployment script.
remote: Project file path: .\aspnet-get-started\aspnet-get-started.csproj
```

To deploy the web app to the production slot, I first ensured I was in the correct project directory within Azure Cloud Shell by running:

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

Then, I added the Git remote for the production deployment by using the Git URL from the Azure Deployment Center:

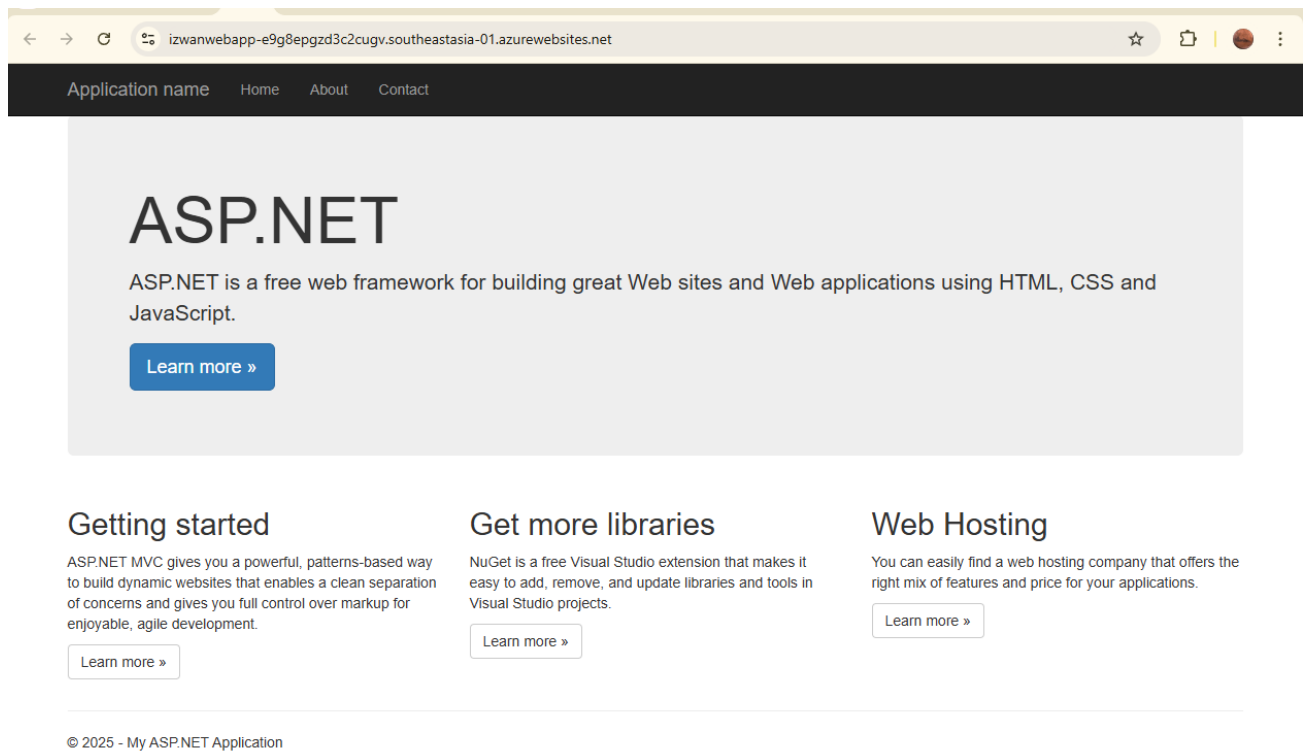
- `git remote add production https://<your-deployment-username>@<your-app-name>.scm.azurewebsites.net:443/<your-app-name>.git`

(Here, I replaced <your-deployment-username> and <your-app-name> with my actual deployment credentials and web app name.)

Once the remote was added, I pushed the code to the production slot using:

- `git push production main:master`

This command tells Git: "Push the local main branch to the remote master branch on the production remote." During the push, I was prompted to enter the deployment username and password I had set earlier in the Deployment Center. After the deployment completed successfully, the web application became live on the production slot.



In the Azure Portal, I go back to the Overview tab, click the URL, to open the deployed web app in my browser.

➤ Create a new staging slot

The screenshot shows the Azure Portal interface for a Web App named 'izwanwebapp'. The left-hand menu is expanded to 'Deployment slots'. The main content area displays a table of deployment slots. The table has four columns: Name, Status, App service plan, and Traffic %.

Name	Status	App service plan	Traffic %
izwanwebapp PRODUCTION	Running	ASP-mslearnslots-8a6e	100
izwanwebapp-staging	Running	ASP-mslearnslots-8a6e	0

Next i create staging slot ,To create a new staging slot, start by navigating to the Azure Portal. From the Home or menu panel, select "All resources" and filter the list by setting the Type to "App Service". Choose my existing Web App from the list. In the left-hand menu under the "Deployment" section, click on "Deployment slots". Once on the Deployment Slots page, click the "+ Add Slot" button. In the dialog that appears, enter "Staging" as the name of the slot. I Keep the default selection to clone settings from my production app to ensure configuration consistency. Finally, I click the "Add" button to create the staging slot.

➤ Set up git deployment for the staging slot

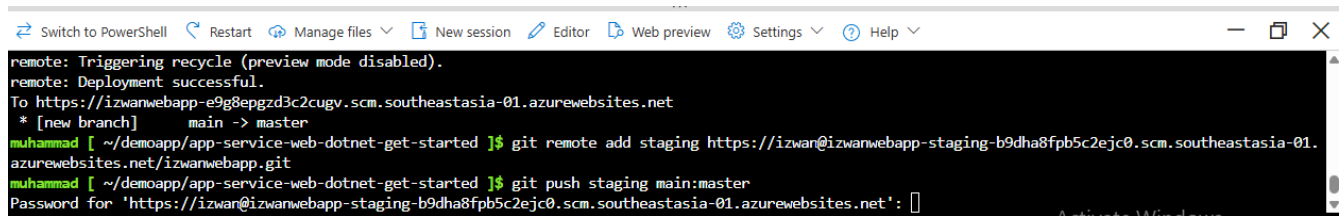
The screenshot shows the Azure Portal interface for a staging slot named 'staging (izwanwebapp/staging)'. The left-hand menu is expanded to 'Deployment Center'. The main content area displays the 'Settings' tab. The 'Source' dropdown is set to 'Local Git'. The 'Repository' section shows the local git repository url will be generated upon completion. The 'Branch' is set to 'master'.

To configure Git deployment for my staging slot, first I open the Azure Portal and navigate to your Staging Web App. In the left-hand menu, under the “Deployment” section, click on “Deployment Center.” In the Deployment Center interface, go to the Settings tab. Set the deployment Source to “Local Git” to allow pushing code directly from my local environment. Once selected, click the “Save” button in the top menu bar to apply the configuration.

The screenshot shows the Azure Portal interface for the 'staging (izwanwebapp/staging)' deployment slot. The left-hand navigation pane is open, showing the 'Deployment Center' tab selected under the 'Deployment' section. The main content area displays the 'Settings' tab for the deployment center. At the top, there is a breadcrumb trail: 'Home > Microsoft.Web-WebApp-Portal-9a266cab-9cb3 | Overview > izwanwebapp | Deployment slots > staging (izwanwebapp/staging)'. Below the breadcrumb, the title bar reads 'staging (izwanwebapp/staging) | Deployment Center' with a star icon and a close button. A search bar is located below the title bar. The main content area is divided into two sections: 'Application scope' and 'User scope'. The 'Application scope' section includes a text box for the deployment URL (ftp://waws-prod-sg1-009.ftp.azurewebsites.windows.net/site/wwwroot), a description of application scope credentials, and fields for 'FTP Username' (izwanwebapp__staging\$izwanwebapp__staging) and 'Password' (masked with dots). The 'User scope' section includes a description of user scope credentials, fields for 'Username' (izwan) and 'Password' (Takingat123_!), and a 'Confirm Password' field (masked with dots). A 'Reset' button is visible next to the password fields. At the bottom left, there is a small note: 'Add or remove features by pressing Ctrl+Shift+F'.

Next for the staging, I set up the Git/FTP credentials required for deploying code. I navigated to the "Local Git/FTP credentials" tab, and under the "User scope" section, I created a deployment username and set a secure password. After filling in these credentials, I clicked the Save button to store the settings and enable authentication for future deployments.

➤ Set up git to deploy the app to the staging slot



```
Switch to PowerShell Restart Manage files New session Editor Web preview Settings Help
remote: Triggering recycle (preview mode disabled).
remote: Deployment successful.
To https://izwanwebapp-e9g8epgzd3c2cugv.scm.southeastasia-01.azurewebsites.net
* [new branch]      main -> master
muhammad [ ~/demoapp/app-service-web-dotnet-get-started ]$ git remote add staging https://izwan@izwanwebapp-staging-b9dha8fb5c2ejc0.scm.southeastasia-01.azurewebsites.net/izwanwebapp.git
muhammad [ ~/demoapp/app-service-web-dotnet-get-started ]$ git push staging main:master
Password for 'https://izwan@izwanwebapp-staging-b9dha8fb5c2ejc0.scm.southeastasia-01.azurewebsites.net':
```

For the next step ,I added the Git remote for my staging slot. First, I went to the Azure Portal and opened the staging slot of my web app. From the Overview page, I copied the Git clone URL found under the Essentials section. After that, I returned to Azure Cloud Shell and made sure I was inside the correct directory by running:

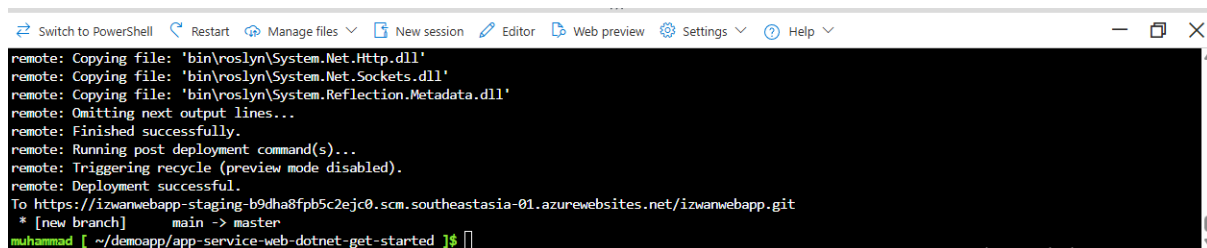
- `cd ~/demoapp/app-service-web-dotnet-get-started`

Once I was in the right place, I used this command:

- `git remote add staging <git-clone-url>`

replacing the placeholder with the actual URL I copied. This step allowed me to connect my local project to the staging, after that I push the staging with this command:

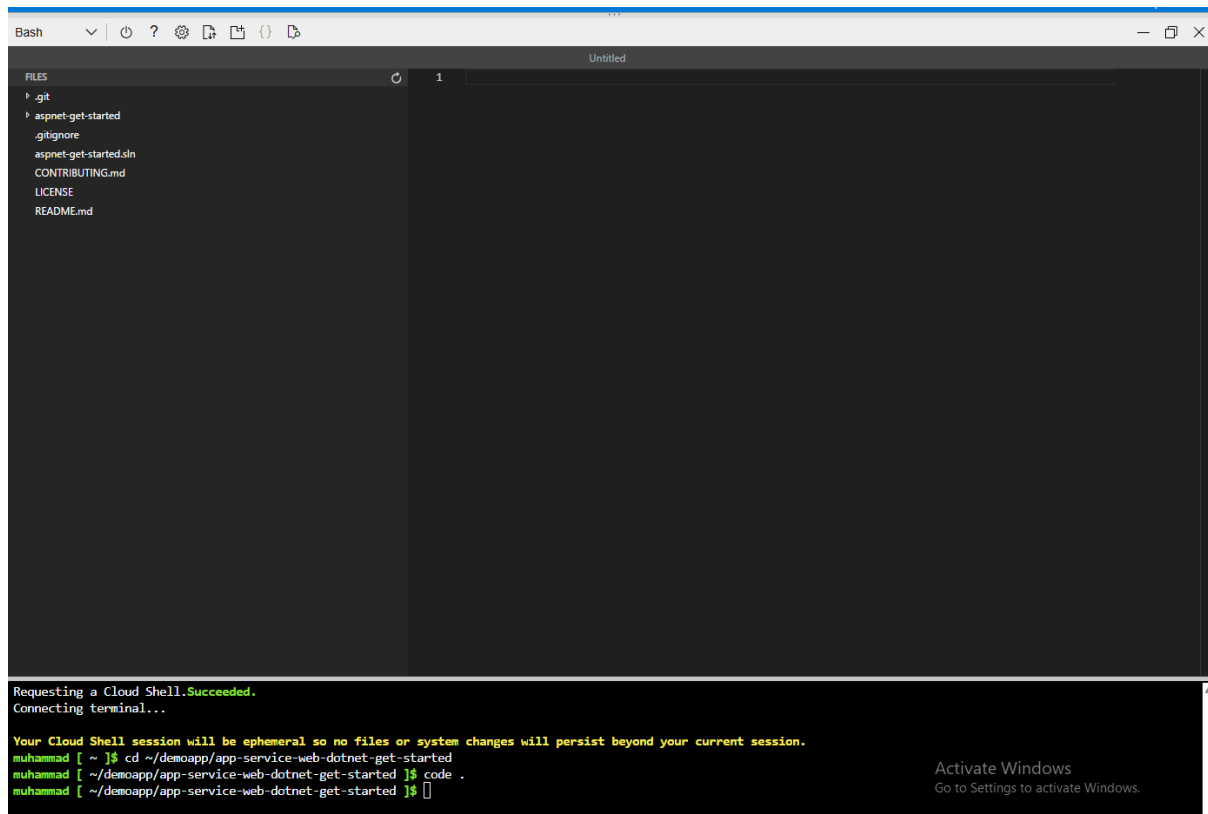
- `git push staging main:master`



```
Switch to PowerShell Restart Manage files New session Editor Web preview Settings Help
remote: Copying file: 'bin/roslyn\System.Net.Http.dll'
remote: Copying file: 'bin/roslyn\System.Net.Sockets.dll'
remote: Copying file: 'bin/roslyn\System.Reflection.Metadata.dll'
remote: Omitting next output lines...
remote: Finished successfully.
remote: Running post deployment command(s)...
remote: Triggering recycle (preview mode disabled).
remote: Deployment successful.
To https://izwanwebapp-staging-b9dha8fb5c2ejc0.scm.southeastasia-01.azurewebsites.net/izwanwebapp.git
* [new branch]      main -> master
muhammad [ ~/demoapp/app-service-web-dotnet-get-started ]$
```

Successfully deploy

➤ **Modify the app source code and deploy the app to the staging slot**



So the next process, I opened the Cloud Shell Editor by running `code .` in Azure Cloud Shell. I made sure I was using the Bash environment so the editor would open correctly. Once the editor loaded, I navigated through the file tree

The screenshot shows a Cloud Shell terminal window with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like `bootstrap.js`, `jquery-3.4.1.min.js`, and a `Views` folder containing `Home`, `About.cshtml`, `Contact.cshtml`, `Index.cshtml`, and `Shared`. The code editor shows the `Index.cshtml` file with the following content:

```
1 @{
2     ViewBag.Title = "Home Page";
3 }
4
5 <div class="jumbotron">
6     <h1>Web App Version 2</h1>
7     <p class="lead">ASP.NET is a free web framework for building great Web sites and Web applicat
8     <p><a href="https://asp.net" class="btn btn-primary btn-lg">Learn more &raquo;</a></p>
9 </div>
10
11 <div class="row">
12     <div class="col-md-4">
13         <h2>Getting started</h2>
14         <p>
15             ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that
16             enables a clean separation of concerns and gives you full control over markup
17             for enjoyable, agile development.
18         </p>
19         <p><a class="btn btn-default" href="https://go.microsoft.com/fwlink/?LinkId=301865">Learn
20         </div>
21     <div class="col-md-4">
22         <h2>Get more libraries</h2>
23         <p>NuGet is a free Visual Studio extension that makes it easy to add, remove, and update
24         <p><a class="btn btn-default" href="https://go.microsoft.com/fwlink/?LinkId=301866">Learn
25         </div>
26     <div class="col-md-4">
27         <h2>Web Hosting</h2>
28         <p>You can easily find a web hosting company that offers the right mix of features and pr
29         <p><a class="btn btn-default" href="https://go.microsoft.com/fwlink/?LinkId=301867">Learn
30         </div>
31 </div>
```

Below the code editor, a message states: "Requesting a Cloud Shell.Succeeded. Connecting terminal..." followed by a notice: "Your Cloud Shell session will be ephemeral so no files or system changes will persist beyond your current session." The terminal shows the following commands and output:

```
muhammad [ ~ ]$ cd ~/demoapp/app-service-web-dotnet-get-started
muhammad [ ~/demoapp/app-service-web-dotnet-get-started ]$ code .
muhammad [ ~/demoapp/app-service-web-dotnet-get-started ]$
```

An "Activate Windows" watermark is visible in the bottom right corner of the terminal window.

I navigated through the file tree by going into `demoapp > app-service-web-dotnet-get-started > aspNet-get-started > Views > Home`. There, I located and opened the `Index.cshtml` file. Inside the file, I looked for the line that had `<h1>ASP.NET</h1>` and I replaced it with `<h1>Web App Version 2</h1>`. After making that change, I saved the file using `Ctrl+S` and then exited the editor by pressing `Ctrl+Q`.


```

Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Your Cloud Shell session will be ephemeral so no files or system changes will persist beyond your current session.
muhammad [ ~ ]$ cd ~/demoapp/app-service-web-dotnet-get-started
muhammad [ ~/demoapp/app-service-web-dotnet-get-started ]$ code .
muhammad [ ~/demoapp/app-service-web-dotnet-get-started ]$ cd ~/demoapp/app-service-web-dotnet-get-started
git add .
git commit -m "New version of web app."
git push staging main:master
[main 1144ada] New version of web app.
1 file changed, 2 insertions(+), 2 deletions(-)
Password for 'https://izwanwebapp-staging-b9dha8f5c2e5c0.scm.southeastasia-01.azurewebsites.net':
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 3 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 508 bytes | 508.00 KiB/s, done.
Total 6 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Updating branch 'master'.
remote: Updating submodules.
remote: Preparing deployment for commit id '1144ada754'.
remote: Generating deployment script.
remote: Running deployment command...
remote: Handling .NET Web Application deployment.
remote: MSBuild auto-detection: using msbuild version '14.0.23107.0 built by: D14REL' from 'C:\Program Files (x86)\MSBuild\14.0\Bin'.
remote: All packages listed in packages.config are already installed.
remote: aspnet-get-started -> C:\home\site\repository\aspnet-get-started\bin\aspnet-get-started.dll
remote: Transformed Web.config using C:\home\site\repository\aspnet-get-started\Web.Release.config into obj\Release\TransformWebConfig\transformed\Web.config.
remote: Copying all files to temporary location below for package/publish:
remote: C:\local\Temp\8ddae810f88989.
remote: Creating app_offline.htm
remote: KuduSync.NET from: 'C:\local\Temp\8ddae810f88989' to: 'C:\home\site\wwwroot'
remote: Copying file: 'Views\Home\Index.cshtml'
remote: Deleting app_offline.htm
remote: Finished successfully.
remote: Running post deployment command(s)...
remote: Triggering recycle (preview mode disabled).
remote: Deployment successful.
To https://izwanwebapp-staging-b9dha8f5c2e5c0.scm.southeastasia-01.azurewebsites.net/izwanwebapp.git
212a655..1144ada main -> master
muhammad [ ~/demoapp/app-service-web-dotnet-get-started ]$

```

to commit the changes I made to the Index.cshtml file and push them to the staging slot. First, I made sure I was still inside the demoapp/app-service-web-dotnet-get-started directory in Cloud Shell. Then, I ran `git add .` to stage the changes, followed by :

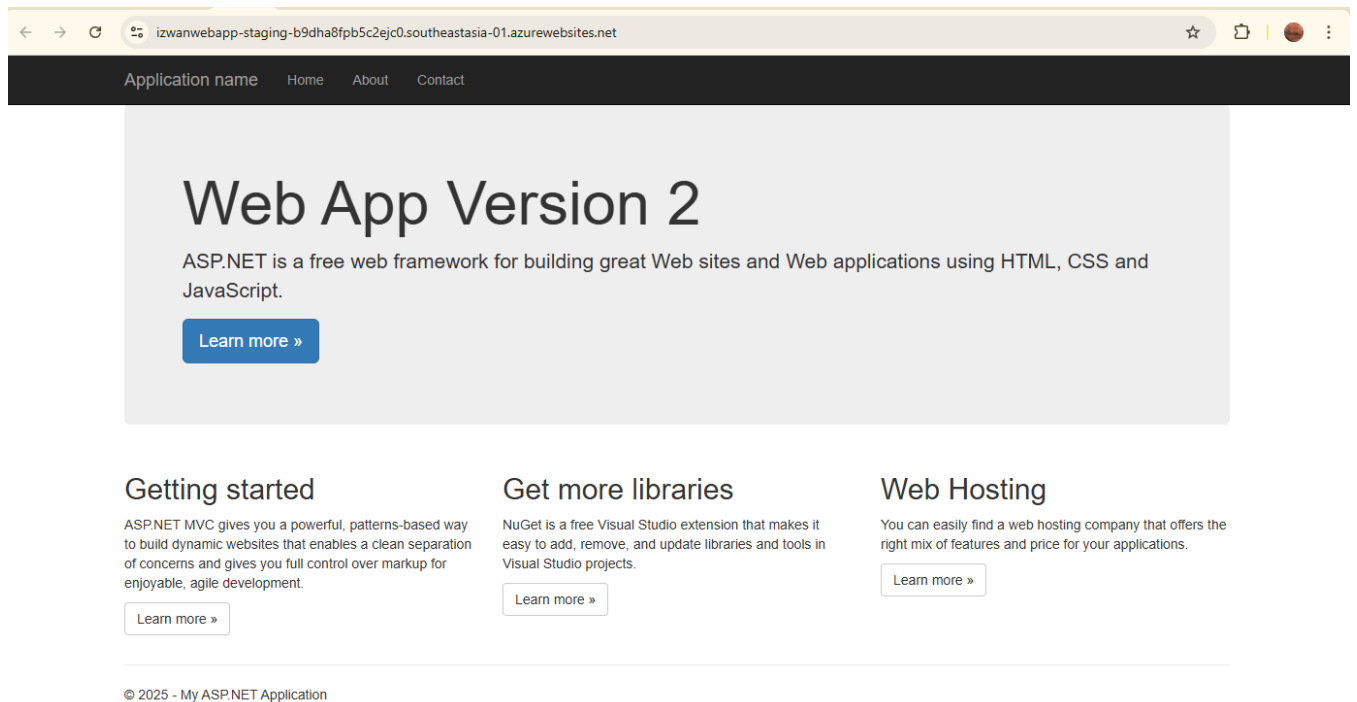
- `git commit -m "Updated to Web App Version 2"`

to create a commit with a message describing the update. After that, I deployed the updated version to the staging slot by running

- `git push staging main:master`

When prompted, I entered the deployment username and password that I had previously set in the Deployment Center. Once the push was complete, the updated web app showing “Web App Version 2” became visible on the staging URL.

➤ **Browse the staging slot**



Successfully create the web app version 2 , At this point, the staging slot has the new version of the code

Deploy a web app by using deployment slots

Azure Lab #30B

Name: Muhammad Hazman bin Mohd Sofee

To begin the process, I configured slot settings for both the production and staging slots of my web app. In the Azure portal, I navigated to the production slot's Configuration page and added a new application setting named `ENVIRONMENT_NAME` with the value `production`, marking it as a deployment slot setting. I also added another setting called `APP_VERSION` with the value `1`, but did not mark this one as a slot setting. After entering these, I saved the settings.

Add/Edit application setting

Name *	<input type="text" value="ENVIRONMENT_NAME"/>
Value	<input type="text" value="production"/>
Deployment slot setting	<input checked="" type="checkbox"/>

Add/Edit application setting

Name *	<input type="text" value="APP_VERSION"/>
Value	<input type="text" value="1"/>
Deployment slot setting	<input type="checkbox"/>

I repeated the configuration steps for the Staging slot. I added `ENVIRONMENT_NAME` with the value `staging`, marking it as a slot setting, and `APP_VERSION` with the value `2`, leaving the deployment slot setting option unchecked. This ensured that `ENVIRONMENT_NAME` would not be swapped between slots, while `APP_VERSION` would.

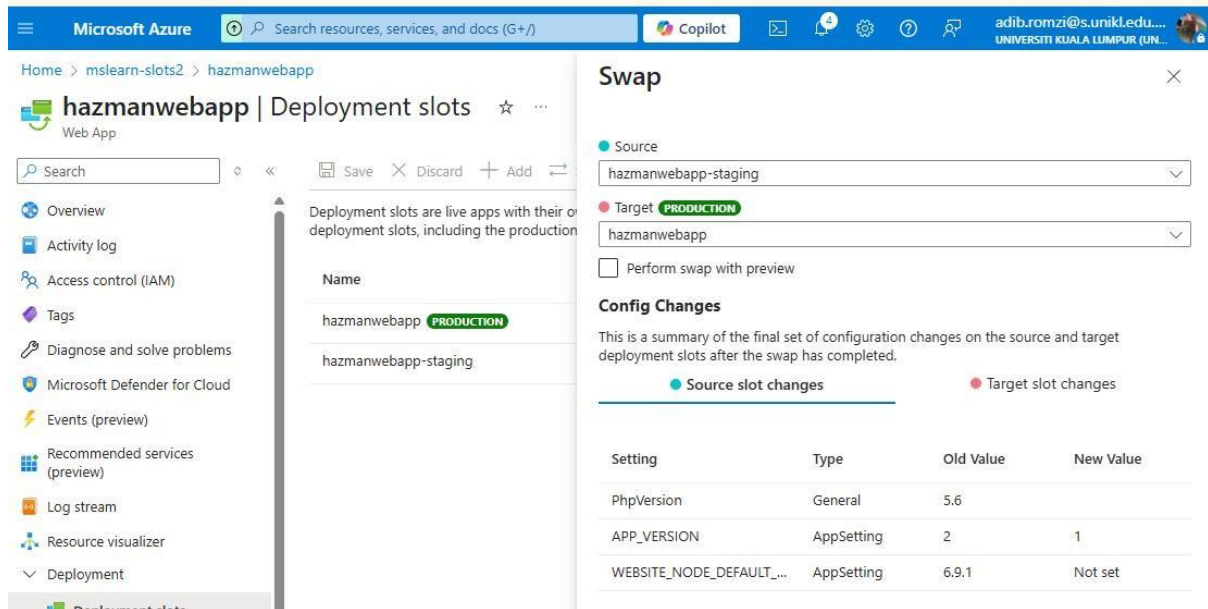
Add/Edit application setting

Name *	<input type="text" value="ENVIRONMENT_NAME"/>
Value	<input type="text" value="staging"/>
Deployment slot setting	<input checked="" type="checkbox"/>

Add/Edit application setting

Name *	<input type="text" value="APP_VERSION"/>
Value	<input type="text" value="2"/>
Deployment slot setting	<input type="checkbox"/>

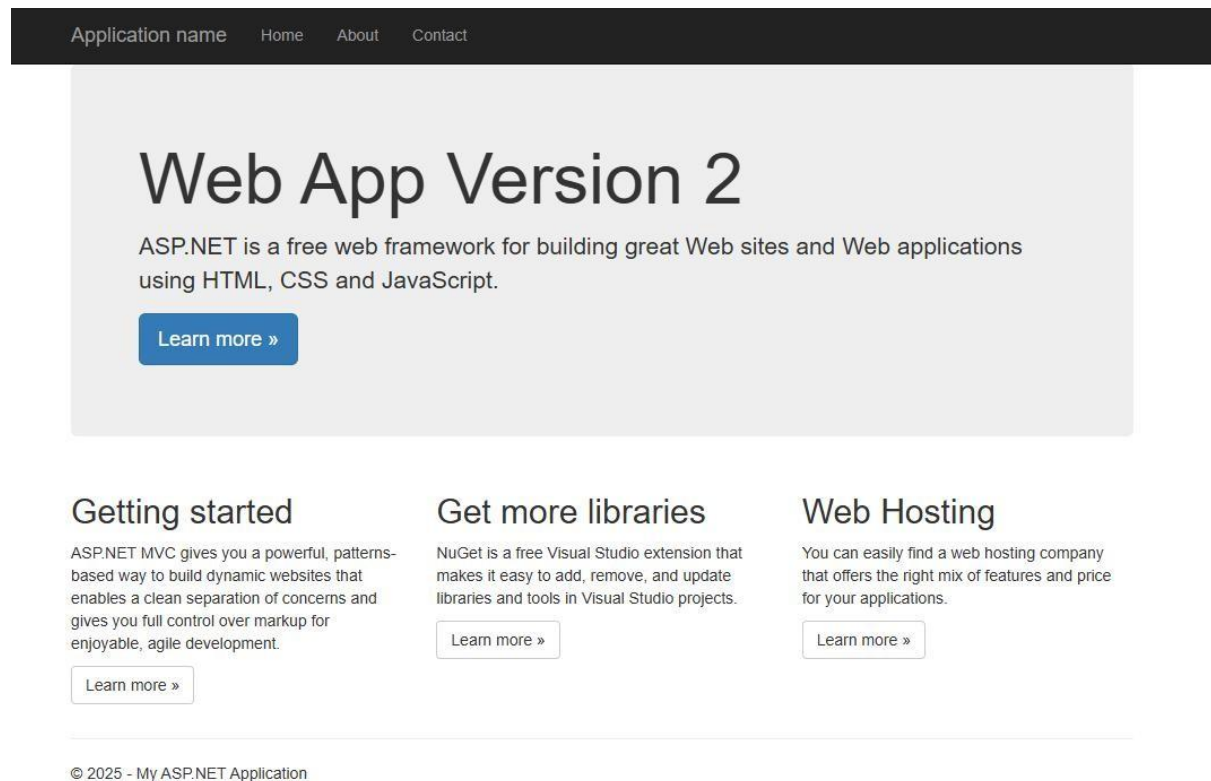
After confirming the settings, I proceeded to swap the Staging and Production slots. In the Azure portal, under the production web app's Deployment section, I chose Deployment slots Swap, verified the staging and production slots were selected correctly, and reviewed the setting behavior. I confirmed that APP_VERSION would be swapped, while ENVIRONMENT_NAME would stay in place, and then I selected Swap to execute it.



The screenshot displays the Azure portal interface for a web application named 'hazmanwebapp'. The left sidebar shows navigation options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Recommended services (preview), Log stream, Resource visualizer, and Deployment. The main area shows the 'Deployment slots' section with a list of slots: 'hazmanwebapp' (marked as PRODUCTION) and 'hazmanwebapp-staging'. A 'Swap' dialog box is open on the right, showing the configuration for swapping the source and target slots. The 'Source' slot is 'hazmanwebapp-staging' and the 'Target' slot is 'hazmanwebapp'. The 'Perform swap with preview' checkbox is unchecked. Below the slot selection, the 'Config Changes' section provides a summary of the final set of configuration changes on the source and target deployment slots after the swap has completed. The table below details these changes.

Setting	Type	Old Value	New Value
PhpVersion	General	5.6	
APP_VERSION	AppSetting	2	1
WEBSITE_NODE_DEFAULT_...	AppSetting	6.9.1	Not set

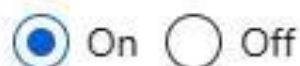
Once the swap completed, I navigated to the Overview page of the production slot and clicked Browse. The app opened in a new browser tab, now showing Version 2 of the web app as active in production. This confirmed a successful deployment.



Next, I enabled Auto Swap to streamline future deployments. I went to the Configuration page of the staging slot, selected the General settings tab, and turned on the Auto swap enabled setting. I chose production as the target slot and saved the configuration. This setup ensures any deployment to staging would automatically be swapped to production once warmed up.

Deployment Slot

Auto swap enabled



To test the auto swap feature, I updated the code to create Version 3 of the app. I navigated to the file `Index.cshtml` under Views Home, replaced the existing `<h1>Web App Version 2</h1>` line with `<h1>Web App Version 3</h1>`, and saved the changes. Then, using the Cloud Shell, I ran `git add .`, `git commit -m "Third version of web app."`, and `git push staging` to deploy the new version to the staging slot.

The screenshot shows the Microsoft Azure Cloud Shell environment. The top bar includes the Microsoft Azure logo, a search bar, and the user's email address: `adib.romzi@s.unikl.edu...`. The main area is split into a file explorer on the left and a code editor in the center. The file explorer shows a project structure with folders like `.git`, `App_Start`, `Content`, `Controllers`, `fonts`, `Properties`, `Scripts`, `Views`, and `Shared`. The `Views` folder is expanded, showing `Home` with files `About.cshtml`, `Contact.cshtml`, and `Index.cshtml`. The `Index.cshtml` file is open in the editor, showing the following code:

```

1 @{
2     ViewBag.Title = "Home Page";
3 }
4
5 <div class="jumbotron">
6     <h1>Web App Version 3</h1>
7     <p class="lead">ASP.NET is a free web framework for building great Web s
8     <p><a href="https://asp.net" class="btn btn-primary btn-lg">Learn more &
9 </div>
10
11 <div class="row">
12     <div class="col-md-4">
13         <h2>Getting started</h2>
14         <p>
15             ASP.NET MVC gives you a powerful, patterns-based way to build dy
16             enables a clean separation of concerns and gives you full contro
17             for enjoyable, agile development.
18         </p>
19         <p><a class="btn btn-default" href="https://go.microsoft.com/fwlink/
20     </div>
21     <div class="col-md-4">
22         <h2>Get more libraries</h2>
23         <p>NuGet is a free Visual Studio extension that makes it easy to add
24         <p><a class="btn btn-default" href="https://go.microsoft.com/fwlink/
25     </div>
26     <div class="col-md-4">
27         <h2>Web Hosting</h2>
28         <p>You can easily find a web hosting company that offers the right m
29         <p><a class="btn btn-default" href="https://go.microsoft.com/fwlink/
30     </div>
31 </div>

```

At the bottom, a terminal window shows the output of a `git push` command:

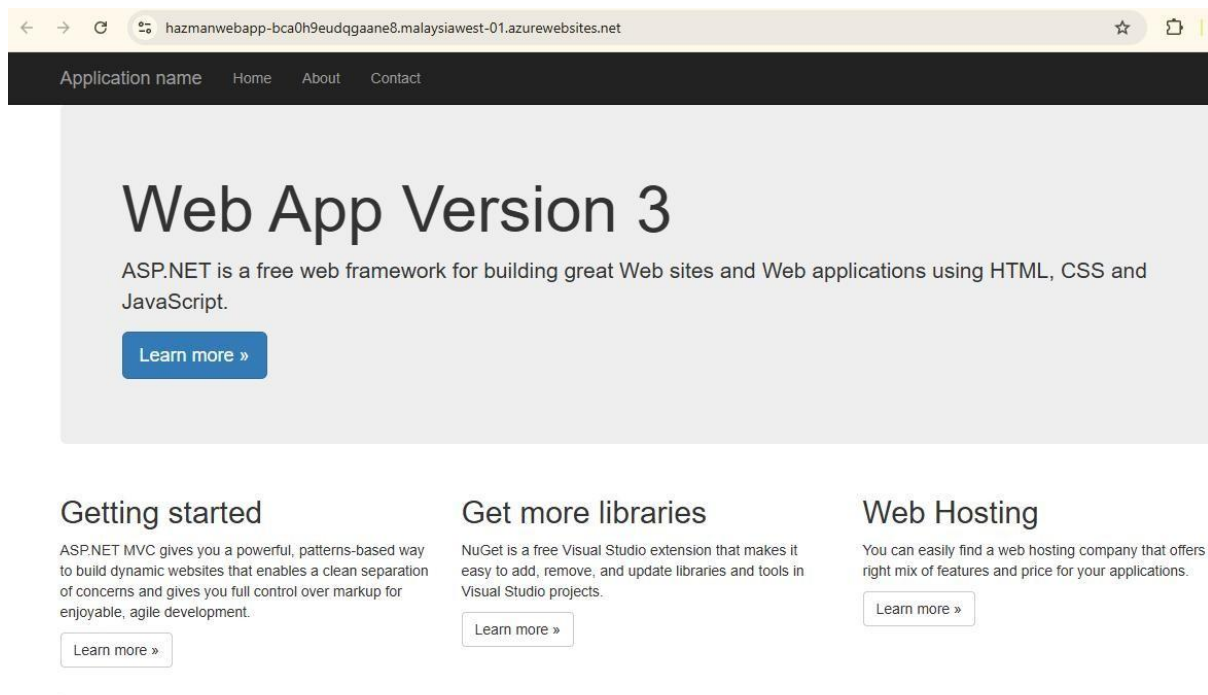
```

remote: Counting objects: 100% (233/233), done.
remote: Compressing objects: 100% (123/123), done.
remote: Total 233 (delta 104), reused 233 (delta 104), pack-reused 0 (from 0)
Receiving objects: 100% (233/233), 963.72 KiB | 3.90 MiB/s, done.
Resolving deltas: 100% (104/104), done.
muhammad [ ~ ]$
cd ~/demoapp/app-service-web-dotnet-get-started
muhammad [ ~/demoapp/app-service-web-dotnet-get-started ]$ code .
muhammad [ ~/demoapp/app-service-web-dotnet-get-started ]$

```

An "Activate Windows" watermark is visible in the bottom right corner of the terminal area.

After the push completed, Azure automatically swapped the staging slot into production. I went back to the Azure portal, accessed the production slot's Overview, and clicked Browse. The web app now displayed Version 3, confirming that auto swap worked as expected. If the old version appeared at first, refreshing the browser after a short wait resolved it.



Using Azure Deployment Slots made deploying the web app safe, efficient, and reliable. I was able to test updates fully in staging before pushing to production, automate deployments using Auto Swap, and quickly roll back to a previous version if necessary. This approach reduces downtime and minimizes risk during updates, improving the overall deployment process.

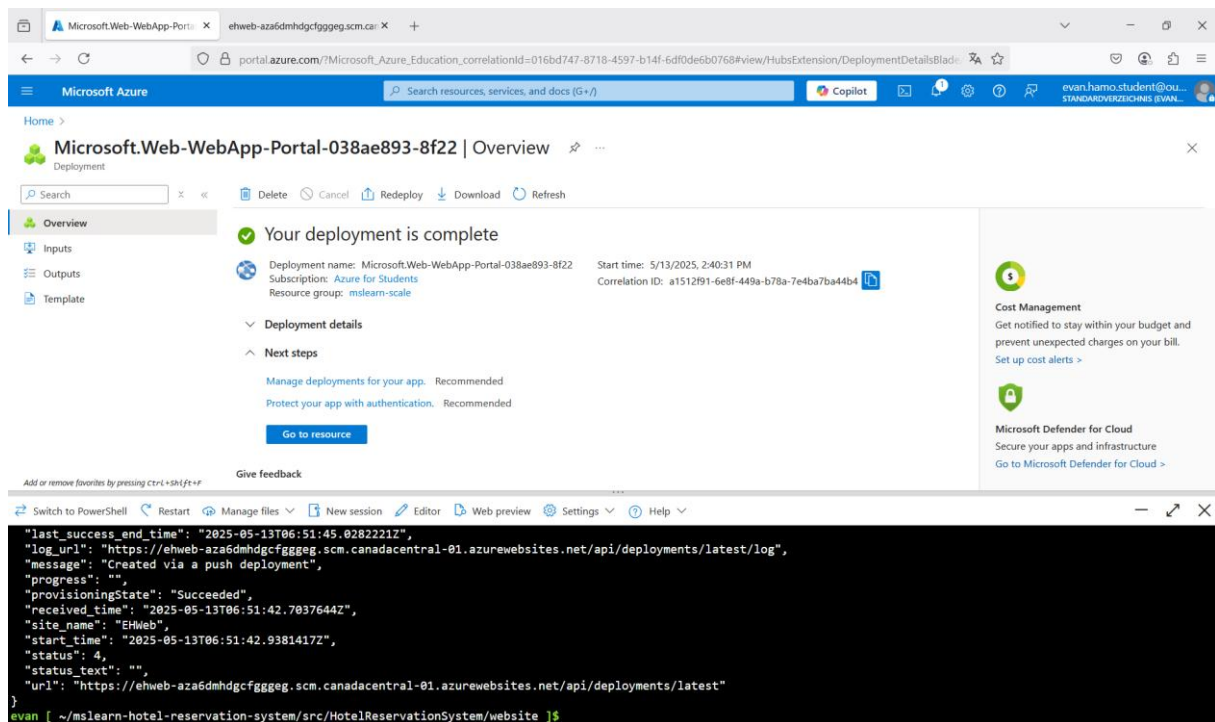
Scale a web app manually

Azure Lab #31

Name: Evan Hamo

Deployment Setup with Cloud Shell and Azure Portal

In this step, I used Azure Cloud Shell to set up the project. I cloned the repository using git clone, built the app, and then packaged it with zip website.zip *. After that, I deployed it to Azure using a ZIP deployment. The Azure Portal confirms that the deployment was completed successfully, and the app is now running in the cloud.



This screenshot shows that the deployment of the HotelReservationSystem app to Azure was successful. In the Azure portal, the message “Your deployment is complete” confirms that everything worked. In the terminal below, you can see the zip deployment command and the success message. I also tested the web app by opening the API link in the browser. It returned a JSON response with reservation data, which means the web app is running and correctly configured.

Microsoft Web-App-Portal x ehweb-aza6dmhdgcfgggeg.scm.ca: x

ehweb-aza6dmhdgcfgggeg.scm.canadacentral-01.azurewebsites.net/api/deployments/latest

JSON Rohdaten Kopfzeilen

Speichern Kopieren Alle einklappen Alle ausklappen JSON durchsuchen

```

{
  "id": "cf1e0f8d54ae4ac099dea365704f24f8",
  "status": 4,
  "status_text": "",
  "author_email": "W/A",
  "author": "W/A",
  "deployer": "ZipDeploy",
  "message": "Created via a push deployment",
  "progress": "",
  "received_time": "2025-05-13T06:51:42.7937644Z",
  "start_time": "2025-05-13T06:51:42.9381417Z",
  "end_time": "2025-05-13T06:51:45.8282221Z",
  "last_success_end_time": "2025-05-13T06:51:45.8282221Z",
  "complete": true,
  "active": true,
  "is_temp": false,
  "is_readonly": true,
  "url": "https://ehweb-aza6dmhdgcfgggeg.scm.canadacentral-01.azurewebsites.net/api/deployments/latest",
  "log_url": "https://ehweb-aza6dmhdgcfgggeg.scm.canadacentral-01.azurewebsites.net/api/deployments/latest/log",
  "site_name": "EWeb",
  "provisioningState": "Succeeded"
}

```

ehweb-aza6dmhdgcfgggeg.canadacentral-01.azurewebsites.net/api/reservations/1

JSON Rohdaten Kopfzeilen

Speichern Kopieren Alle einklappen Alle ausklappen JSON durchsuchen

```


{
  "reservationID": 1,
  "customerID": "2088876526",
  "hotelID": "Hotel_785982216",
  "checkin": "2025-06-05T12:44:12.578231+00:00",
  "checkout": "2025-06-12T13:07:11.4412964+00:00",
  "numberOfGuests": 1,
  "reservationComments": "W0P65Q04/sbRzVVBZ7pys2x1JMP/G3KprKwLpZE7cSWtMklyA0q4ZStyRqI/glbne8Plekj3z0Vkrmd30/d/tk8ZQ122jw30C6jTT+N15R4PZPHDkKicIE8Iq1+43h1do1ZcEHTQ2FPbW4JSZqbDv6PZ/zLSfVlpd3FFdtdkXMX4qmt1C98e0E260K2luntvc48Kj7GaIcuXp10VsKSE+or00ga5AMtAc/S86/n0rHrntk0Lry8Kze5uh+QyzAa9q+KDT1a35oAyZ2KA/FNAp0W6Z/z/do8Yj34/4mEau58zv1a8eK2H2HLRZc1qWZ456K+5899Hfah10Z/2A0H#MG/5vYD3R3QZ5m0FgI26d3Uv8n9wZd8KqHao5"
}

```

In this step, I edited the App.config file to connect the test client to my deployed Azure web app. I set the NumClients to 100 and inserted the correct URL for the ReservationsServiceURI. This setup makes sure that the client app will send requests to the right web service for the performance test.

```
Bash
App.config
1 <?xml version="1.0" encoding="utf-8" ?>
2 <configuration>
3   <appSettings>
4     <add key="NumClients" value="100" />
5     <add key="ReservationsServiceURI" value="https://ehweb-azafdmhdgcfgggqg.canadacentral-01.azurewebsites.net/" />
6     <add key="ReservationsServiceCollection" value="api/reservations" />
7   </appSettings>
8 </configuration>
9
evan [ ~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient ]$ code App.config
evan [ ~/mslearn-hotel-reservation-system/src/HotelReservationSystemTestClient ]$
```

Home > EHWeb

 **EHWeb | Scale out (App Service plan)** ☆ ...
Web App

◇ << Refresh Send us your feedback

Resource visualizer

> Deployment

▼ Settings

< Environment variables

< Configuration

< Authentication

< Identity

< Backups

< Custom domains

< Certificates

< Networking

< Scale up (App Service plan)

< Scale out (App Service plan)

< WebJobs

< MySQL In App

Maximum scale (instance) 10

Current instance 1

Scaling

App service provides multiple features that help applications perform their best when scaling demand changes. You can choose to scale your resource manually to a specific instance count, or via a custom Autoscale rule based policy that scales based on metric(s) thresholds, or schedule instance count which scales during designated time windows. You can also use Automatic Scaling features which enables platform managed scale in and scale out for your apps based on incoming HTTP traffic. [Learn more about Azure Autoscale, Automatic Scaling or view the how-to video.](#)

Scale out method

☒ Manual

Maintain a constant instance count for your application

☐ Automatic

Platform managed scale out and in based on traffic
Automatic scaling requires a Premium v2 or Premium v3 App Service Plan.
Upgrade your App Service Plan to enable this feature.
[See recommended pricing plan](#)

☐ Rules Based

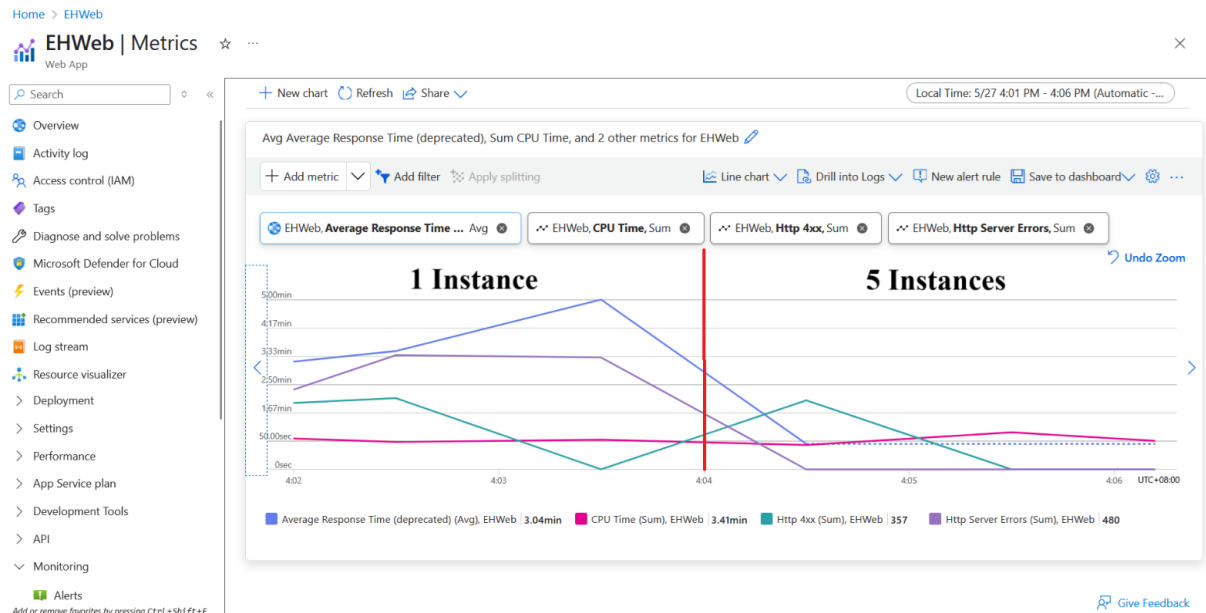
User defined rules to scale on a schedule or based on any app metric

Instance count

5

Scale-Out 1 vs 5 Instance

This chart shows the performance of the web app before and after scaling out. On the left side, with only 1 instance, the response time and error rate were higher. After increasing to 5 instances (right side of the red line), the CPU time increased slightly, but the average response time and HTTP errors decreased, showing that the app handled the load better.



Scale up the pricing plan

Azure lab #32

In Azure cloud computing, scaling up refers to increasing the capacity or performance of an existing resource by upgrading it to a higher-tier or more powerful configuration. This is also known as vertical scaling. For example, in the Azure Portal, scaling up can involve moving an App Service Plan to a higher pricing tier that offers more CPU power, memory, and additional features. Similarly, it could mean upgrading a virtual machine (VM) to a larger size that can handle heavier workloads. In the context of a hotel reservation system, scaling up is important when the application needs to support more advanced features or handle more data processing. It provides the necessary resources to maintain performance as the app becomes more complex. Scaling up can also support scaling out, which means adding more instances of a service to handle increased traffic. Therefore, scaling up is an essential step to ensure the system remains reliable and efficient as demand grows.

For this web application project, the default pricing tier is set to Standard S1, which offers limited memory and processing power. Specifically, the Standard S1 plan includes only 1 vCPU, 1.75 GB of RAM, and 50 GB of storage, which is not sufficient for handling larger volumes of client data or delivering faster response times. As a result, the performance of the web application is weak, especially when it comes to processing and managing data from registered clients efficiently.






To improve performance, the pricing plan needs to be upgraded to a more powerful tier. Initially, Premium v2 P2v2 was considered because it offers better performance, but due to quota limitations in the selected Azure region, it could not be used. Therefore, the plan was upgraded to Premium P2, which provides 2 vCPU, 3.5 GB of RAM, and 250 GB of storage. This Premium P2 tier offers better computing power and storage capacity, allowing the web application to perform faster and manage client data more efficiently, even though it comes at a higher cost.

To upgrade the pricing plan, users need to access the web application that has been created in the Azure Portal and navigate to Settings > App Service (Pricing Plan). In this section, Azure provides various pricing tiers that users can choose from, depending on their system requirements. Each tier offers different levels of performance in terms of CPU, memory, and storage capacity. However, it is important to note that the more powerful the plan, the higher the cost will be. Because of this, users must make careful and informed decisions when selecting a pricing plan. It is essential to balance performance and budget, and to choose a plan that can meet the system's current and future needs without overspending unnecessarily.

Microsoft Azure


Search resources, services, and docs (G+)

Copilot



uwaisimran83@gmail.com
DEFAULT DIRECTORY (UWAIMSML)

Home > muhammaduwaishotelsystem

 muhammaduwaishotelsystem | Scale up (App Service plan) ☆

Web App

Search

Environment variables

Backups

Custom domains

Certificates

Networking

Scale up (App Service plan) ☆

Scale out (App Service plan)

Webjobs

MySQL In App

Service Connector

Properties

Locks

Performance

App Service plan

Development Tools

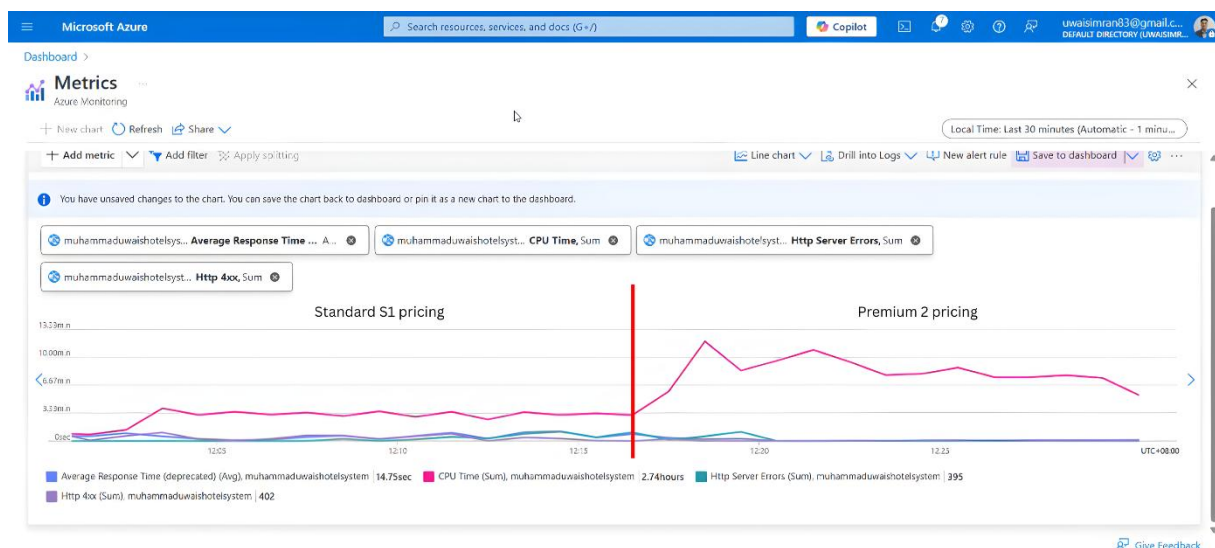
Premium v3 P3V3	195	8	32	250	30	1.312 USD	957.76 USD
Premium v3 P2mv3	195*	4	32	250	30	0.723 USD	527.936 USD
Premium v3 P3mv3	195*	8	64	250	30	1.446 USD	1055.872 USD
Premium v3 P4mv3	195*	16	128	250	30	2.893 USD	2111.744 USD
Premium v3 P5mv3	195*	32	256	250	30	5.786 USD	4223.488 USD
Legacy							
<input type="checkbox"/> Standard S1	100	1	1.75	50	10	0.11 USD	80.30 USD
Standard S2	100	2	3.5	50	10	0.22 USD	160.60 USD
Standard S3	100	4	7	50	10	0.44 USD	321.20 USD
Premium P1	100	1	1.75	250	20	0.33 USD	240.90 USD
<input checked="" type="checkbox"/> Premium P2	100	2	3.5	250	20	0.66 USD	481.80 USD
Premium P3	100	4	7	250	20	1.32 USD	963.60 USD
Premium v2 P1V2	210	1	3.5	250	30	0.22 USD	160.60 USD
Premium v2 P2V2	210	2	7	250	30	0.44 USD	321.20 USD
Premium v2 P3V2	210	4	14	250	30	0.88 USD	642.40 USD

Select

*ACU/vCPU is an approximation of the SKU's relative performance.

Learn more about App Service pricing

After upgrading the pricing plan, the graph statistics data displayed in the Azure Portal shows a clear increase in CPU performance and several other system processes. The diagram below highlights a significant difference in system performance before and after the upgrade. It is evident that the Standard S1 plan could only handle client data at a slower speed due to its limited CPU and memory capacity. In contrast, the Premium P2 pricing plan, with its higher CPU power and larger memory storage, is able to process data more efficiently and handle higher workloads. The average response time has also improved noticeably, resulting in faster interactions for users. Additionally, the number of server errors has decreased, indicating better overall system stability. This demonstrates that selecting a higher pricing plan, when necessary, can significantly improve the efficiency and reliability of a web application especially if the plan is chosen based on the actual needs of the system.



Conclusion

This project has provided valuable hands-on experience in managing web applications using Microsoft Azure, with a strong focus on deployment strategies, performance scaling, and configuration best practices. By exploring deployment slots, we were able to implement safe and structured app updates with minimal downtime, leveraging staging environments to test changes before pushing to production. The integration of Git and Azure's Deployment Center simplified version control and enabled efficient team collaboration.

Furthermore, scaling out and scaling up the web app allowed us to observe the impact of resource allocation on application performance, especially under load. Through detailed performance metrics, we saw firsthand how additional instances and higher-tier pricing plans can enhance responsiveness and reduce errors.

Overall, this assignment not only reinforced theoretical concepts from the Cloud Computing syllabus but also sharpened our practical skills in real-world cloud operations. It highlights the importance of planning, testing, and resource management in building robust, scalable, and efficient cloud-based systems. These are essential skills for any future IT professional working in cloud environments.

Link video of our project

IIB43203 - CLOUD COMPUTING (MARCH 2025) :

<https://youtu.be/ZhzSjRXQP4c>