



## **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**

<b>Programme</b>	: BACHELOR IN INFORMATION TECHNOLOGY (HONS) (INTERNET OF THINGS)
<b>Course Code &amp; Name</b>	: IIB43203 - CLOUD COMPUTING
<b>Course Lecturer(s)</b>	: Megat Norulazmi Megat Mohamed Noor
<b>Type of Submission</b>	: <b>GROUP PROJECT</b>
<b>Penalties</b>	: <ul style="list-style-type: none"><li>• 5% will be deducted per day to a maximum of four (4) working days, after which the submission will <b>not</b> be accepted.</li><li>• Plagiarised work is an Academic Offence in University Rules &amp; Regulations and will be penalised accordingly.</li></ul>

### **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)
JUNE 2025	SHAIFUL HAZIQ HIDZMI BIN SAIPOL AZMI	52224123153
	SHARIFAH ARLEENA BARAKBAH BINTI SYED ASWAD	52224123403
	NUR A'QILAH BINTI SUHAIMI	52224123216
	NIK NUR HUSNA BINTI NIK AHMED RAZANI	52224123376

-----

# CLOUD COMPUTING GROUP PROJECT

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

## Introduction

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

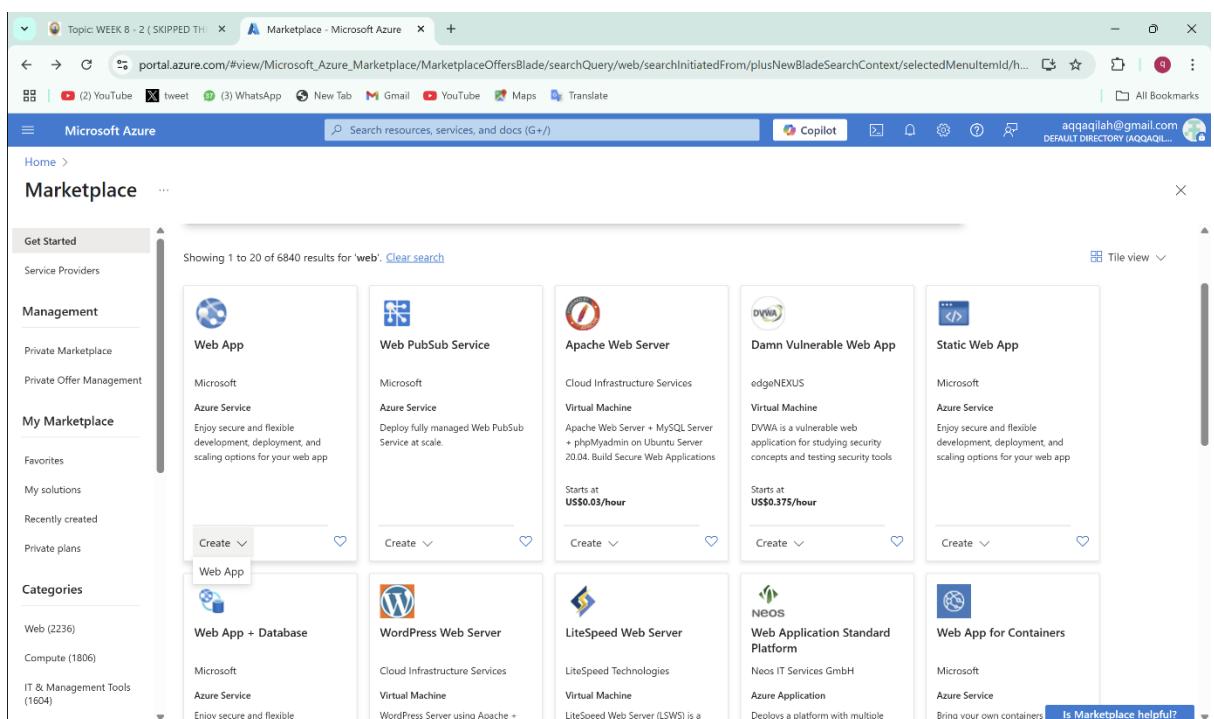
## 1. Create deployment slots and deploy a web app by using deployment slots

### Procedure

#### Step 1: Create a web app

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

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



#### 4. These are what on basic tab

The screenshot shows the 'Create Web App' wizard on the 'Basic' tab. The 'Region' dropdown is set to 'Malaysia West'. A note says 'Not finding your App Service Plan? Try a different region or select your App Service Environment.' The 'Pricing plans' section shows a 'Windows Plan (Malaysia West)' dropdown with '(New) ASP-mslearnslots-84c8' selected and a 'Create new' button. Below it is a 'Pricing plan' dropdown set to 'Free F1 (Shared infrastructure)'. The 'Zone redundancy' section has two options: 'Enabled' (Your App Service plan and the apps in it will be zone redundant. The minimum App Service plan instance count will be two.) and 'Disabled' (Your App Service plan and the apps in it will not be zone redundant. The minimum App Service plan instance count will be one.). The 'Disabled' option is selected. At the bottom are 'Review + create', '< Previous', and 'Next : Database >' buttons.

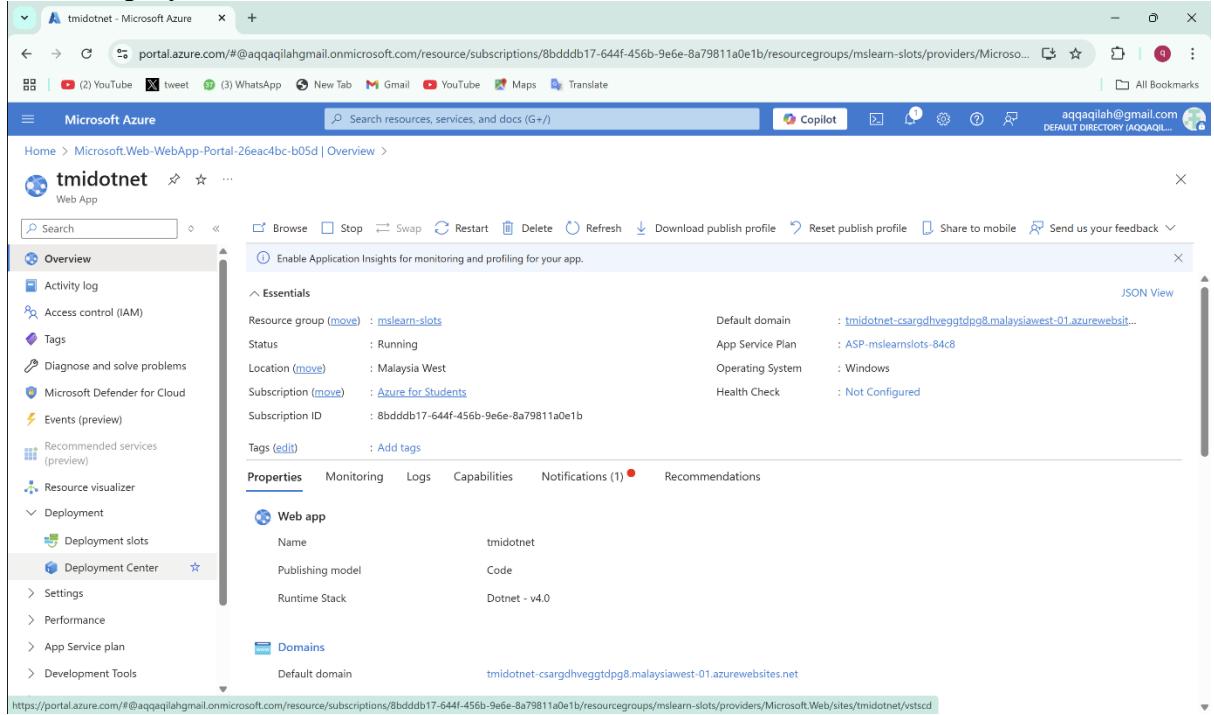
  

The screenshot shows the 'Create Web App' wizard on the 'Basic' tab. The 'Project Details' section includes a 'Subscription' dropdown for 'Azure for Students' and a 'Resource Group' dropdown for '(New) mslearn-slots'. The 'Instance Details' section includes a 'Name' input field with 'tmidotnet.azurewebsites.net', a 'Publish' section with 'Code' selected, and a 'Runtime stack' dropdown for 'ASP.NET V4.8'. A 'Secure unique default hostname' toggle is turned on. At the bottom are 'Review + create', '< Previous', and 'Next : Database >' buttons.

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

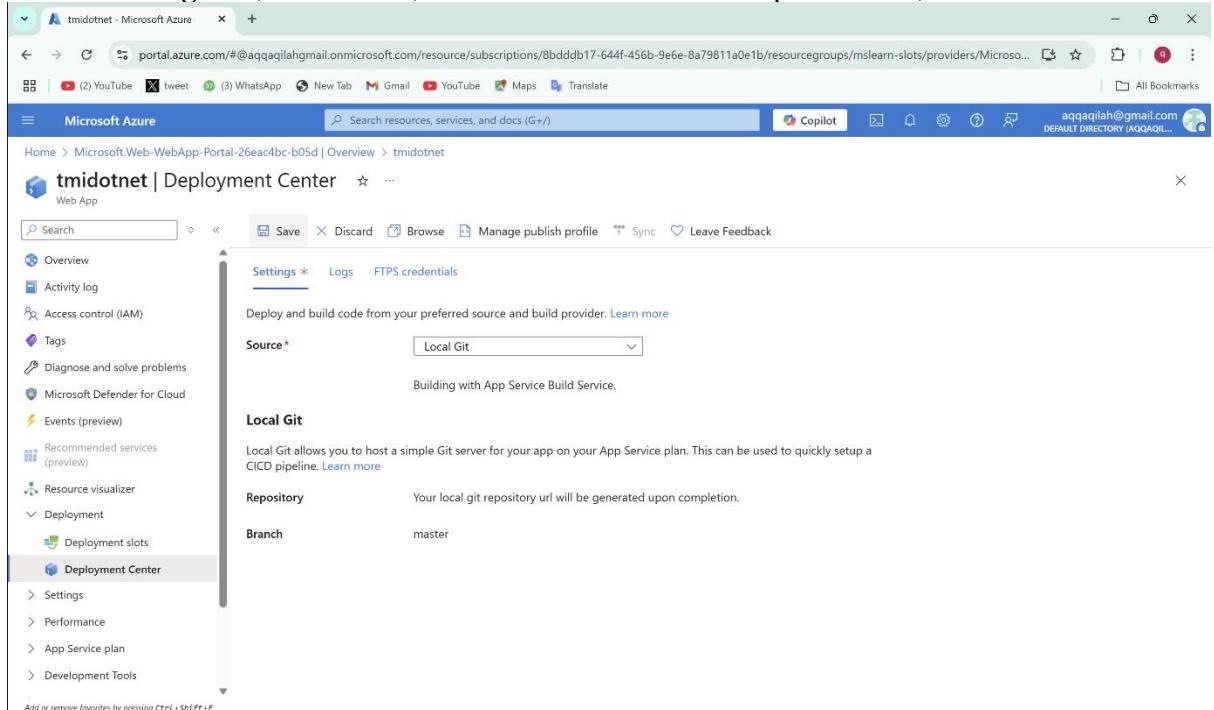
## Step 2: Configure git deployment

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



The screenshot shows the Azure portal interface for a web app named 'tmidotnet'. The left sidebar has 'Deployment' expanded, with 'Deployment Center' selected. The main content area displays the 'tmidotnet' web app's configuration, including its resource group (mslearn-slots), status (Running), location (Malaysia West), and service plan (Azure for Students). It also shows the default domain (tmidotnet-[csargdhveggtog8.malaysiawest-01.azurewebsites.net](#)). The 'Properties' tab is selected.

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



The screenshot shows the 'Deployment Center' page for the 'tmidotnet' web app. The left sidebar has 'Deployment' expanded, with 'Deployment Center' selected. The top navigation bar includes 'Save' and 'Discard' buttons. The 'Settings' tab is selected, showing the 'Source' dropdown set to 'Local Git'. Below this, there are sections for 'Local Git' (describing it as a simple Git server for App Service) and 'Repository' (specifying the local git repository URL and branch).

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

The screenshot shows the Microsoft Azure portal interface for a web application named 'tmidotnet'. The left sidebar has 'Deployment Center' selected under 'Web App'. The main content area is titled 'Deployment Center' and shows two tabs: 'Application scope' and 'User scope'. In the 'Application scope' section, it says 'Application scope credentials are auto-generated and provide access only to this specific app or deployment slot. These credentials can be used with FTPS, Local Git and WebDeploy. They cannot be configured manually, but can be reset anytime.' Below this, there are fields for 'FTPS Username' (set to 'tmidotnet\\${tmidotnet}'), 'Local Git Username' (set to 'Stmidotnet'), and 'Password' (a masked password). In the 'User scope' section, it says 'User scope credentials are defined by you, the user, and can be used with all the apps to which you have access. 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 following format: 'tmidotnet\<your username>'. Authenticating with Git requires only the username <your username> defined below.' Below this, there are fields for 'Username' (set to 'qilala'), 'Password' (set to 'Coffeenmatcha4life!!'), and 'Confirm Password' (set to 'Coffeenmatcha4life!!'). The top navigation bar includes 'Save', 'Discard', 'Browse', 'Manage publish profile', 'Sync', and 'Leave Feedback' buttons.

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

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

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

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

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

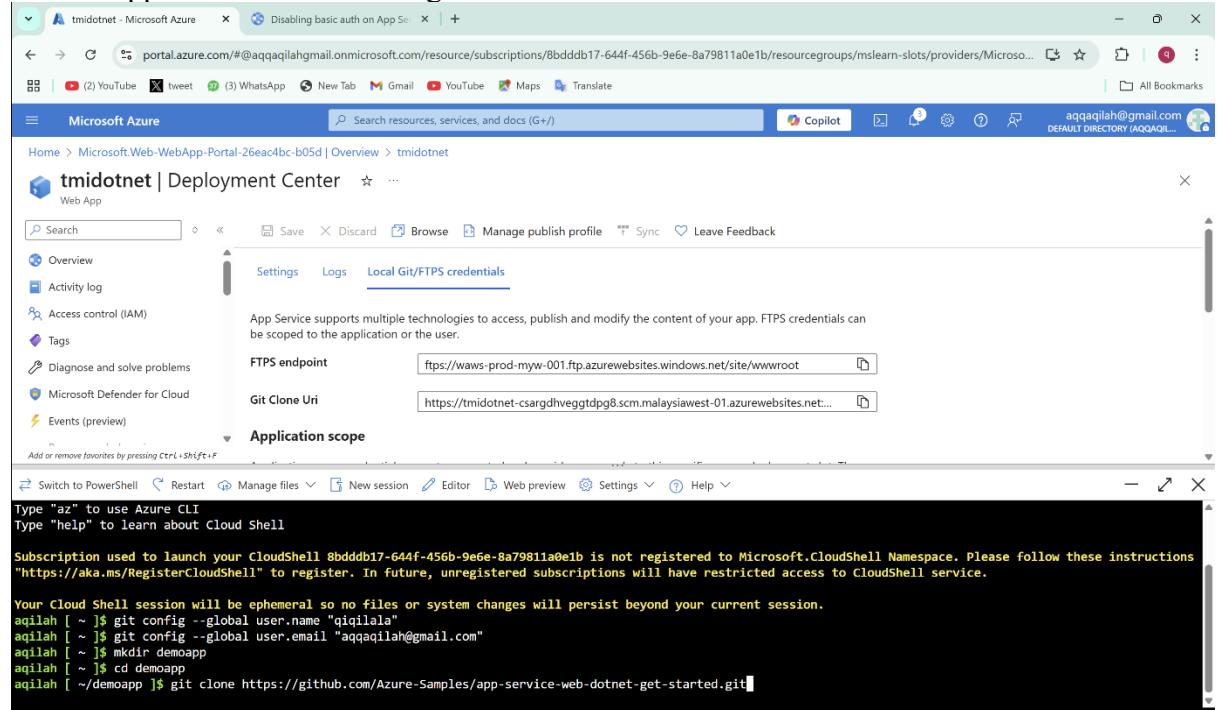
```
mkdir demoapp
```

```
cd demoapp
```

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

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

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



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

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

The screenshot shows the Azure portal interface for a Web App named 'tmidotnet'. The 'Overview' tab is selected. In the 'Essentials' section, you can see the following details:

Setting	Value
Resource group (move)	: mslearn-slots
Status	: Running
Location (move)	: Malaysia West
Subscription (move)	: Azure For Students
Subscription ID	: 8bdddb17-644f-456b-9e6e-8a79811a0e1b
Default domain	: tmidotnet-csargdhveggtdp8.malaysiawest-01.azurewebsites.net
App Service Plan	: ASP-mslearnslots-84c8
Operating System	: Windows
Health Check	: Not Configured
Git/Deployment user...	: qilala
Git clone url	: https://qilala@tmidotnet-csargdhveggtdp8.scm.malaysia...

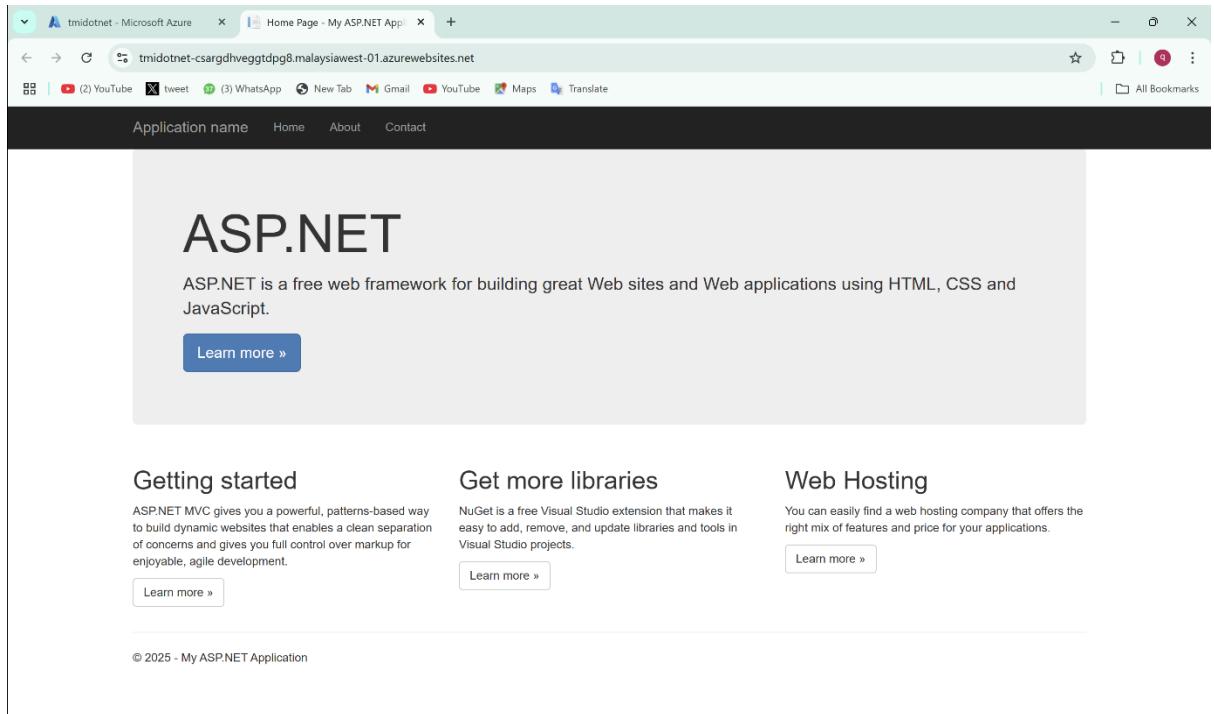
In the 'Deployment' section, under 'Web app', the configuration is as follows:

Setting	Value
Name	tmidotnet
Publishing model	Code
Runtime Stack	Dotnet - v4.0

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

```
aqilah [ ~/demoapp ]$ cd app-service-web-dotnet-get-started
aqilah [ ~/demoapp/app-service-web-dotnet-get-started ]$ git remote add production https://qilala@tmidotnet-csargdhveggtdp8.scm.malaysiawest-01.azurewebsites.net/tmidotnet.git
aqilah [ ~/demoapp/app-service-web-dotnet-get-started ]$ git push production
Password for 'https://qilala@tmidotnet-csargdhveggtdp8.scm.malaysiawest-01.azurewebsites.net':
fatal: Authentication failed for 'https://tmidotnet-csargdhveggtdp8.scm.malaysiawest-01.azurewebsites.net/tmidotnet.git/'
aqilah [ ~/demoapp/app-service-web-dotnet-get-started ]$ git push production
Password for 'https://qilala@tmidotnet-csargdhveggtdp8.scm.malaysiawest-01.azurewebsites.net':
Enumerating objects: 226, done.
Counting objects: 100% (226/226), done.
Delta compression using up to 3 threads
Compressing objects: 100% (122/122), done.
Writing objects: 100% (226/226), 963.27 KiB | 240.82 MiB/s, done.
```

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



## Step 5: Create a new staging slot

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

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

The screenshot shows two screenshots of the Microsoft Azure portal interface. The top screenshot is titled 'tmidotnet | Deployment slots' and shows the 'Add Slot' dialog box. The 'Name' field is set to 'staging'. The 'Clone settings from:' dropdown is set to 'Do not clone settings'. The bottom screenshot shows the same 'Deployment slots' page after the slot has been added. It lists two slots: 'tmidotnet (PRODUCTION)' and 'tmidotnet-staging'. Both slots are shown as 'Running' with an 'ASP-mslearnslots-84c8' app service plan. The traffic distribution is 100% for production and 0% for the staging slot.

Name	Status	App service plan	Traffic %
tmidotnet (PRODUCTION)	Running	ASP-mslearnslots-84c8	100
tmidotnet-staging	Running	ASP-mslearnslots-84c8	0

## Step 6: Set up git deployment for the staging slot

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

The screenshot shows the Azure portal interface with two windows open. Both windows are titled "staging (tmidotnet/staging) | Deployment Center".

**Top Window (Deployment Center - Application scope):**

- Application scope:** Describes auto-generated credentials for this specific app or deployment slot, used for FTPS, Local Git, and WebDeploy.
- FTPS Username:** tmidotnet\_staging\$tmidotnet\_staging
- Local Git Username:** \$tmidotnet\_staging
- Password:** (redacted)

**User scope:** Describes user-defined credentials for access to multiple apps.

- Username:** lala2.0
- Password:** Honeystar1scereal#
- Confirm Password:** Honeystar1scereal#

**Bottom Window (Deployment Center - Settings):**

- Source:** Local Git
- Local Git:** Describes Local Git as a simple Git server for hosting your app's code.
- Repository:** Your local git repository url will be generated upon completion.
- Branch:** master

In both windows, the top menu bar shows the URL `https://staging.tmidotnet.azurewebsites.net` and the status bar indicates the default directory is `aqqaqilah@gmail.com`.

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.

The screenshot shows the Azure Deployment Center interface for the 'staging (tmidotnet/staging)' app service slot. The left sidebar shows various tabs like Overview, Activity log, Access control (IAM), Tags, and Deployment slots. The 'Deployment Center' tab is selected. The main pane displays the 'Application scope' and 'User scope' sections. In the 'User scope' section, there are fields for 'Username' (set to 'lala2.0'), 'Password' (set to 'Honeystar1scereal#'), and 'Confirm Password' (also set to 'Honeystar1scereal#'). The URL in the browser bar is https://tmidotnet-staging.azurewebsites.net/. The top navigation bar includes 'Save', 'Discard', 'Browse', 'Manage publish profile', 'Sync', and 'Leave Feedback' buttons.

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

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

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

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

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

```
. code
```

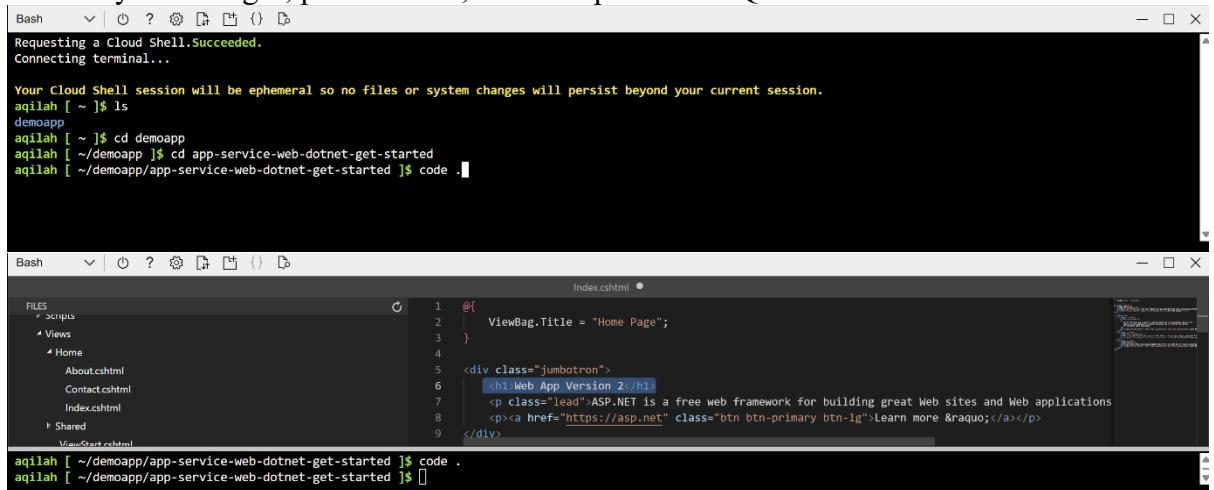
3. In the list of Files, expand demoapp > app-service-web-dotnet-get-started > aspnet-get-started > Views > Home.
4. Select Index.cshtml.
5. Locate the following code.

```
<h1>ASP.NET</h1>
```

6. Replace that code with this code.

```
<h1>Web App Version 2</h1>
```

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



The screenshot shows the Azure Cloud Shell interface. At the top, there is a terminal window titled 'Bash' showing the command history and session details. Below it is a code editor window titled 'Index.cshtml' containing the ASP.NET view code. The code editor has a sidebar showing the file structure of the 'Views/Home' folder. The code in the editor is as follows:

```
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 applications
8      <p><a href="https://asp.net" class="btn btn-primary btn-lg">Learn more &raquo;</a></p>
9  </div>
```

The terminal window shows the command history: 'cd demoapp', 'cd app-service-web-dotnet-get-started', and 'code .' followed by a prompt for saving changes.

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

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

```
Bash 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.
aqilah [ ~ ]$ ls
demoapp
aqilah [ ~ ]$ cd demoapp
aqilah [ ~/demoapp ]$ cd app-service-web-dotnet-get-started
aqilah [ ~/demoapp/app-service-web-dotnet-get-started ]$ code .
aqilah [ ~/demoapp/app-service-web-dotnet-get-started ]$ git add .
aqilah [ ~/demoapp/app-service-web-dotnet-get-started ]$ git commit -m "New version of web app."
```

```
Bash
aqilah [ ~/demoapp/app-service-web-dotnet-get-started ]$ git add .
aqilah [ ~/demoapp/app-service-web-dotnet-get-started ]$ git commit -m "New version of web app."
[main fdcc5a99] New version of web app.
 1 file changed, 2 insertions(+), 2 deletions(-)
aqilah [ ~/demoapp/app-service-web-dotnet-get-started ]$ git push staging
Password for 'https://lala2.0@midotnet-staging-b9eyfad7h6ewafby.scm.malaysiawest-01.azurewebsites.net':
Enumerating objects: 232, done.
Counting objects: 100% (232/232), done.
Delta compression using up to 3 threads
Compressing objects: 100% (128/128), done.
Writing objects: 100% (232/232), 963.66 KiB | 120.46 MiB/s, done.
Total 232 (delta 104), reused 221 (delta 99), pack-reused 0 (from 0)
```

## Step 9: Browse the staging slot

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

The screenshot shows a web browser window with the following details:

- Title Bar:** staging (midotnet/staging) - M | Home Page - My ASP.NET App | +
- Address Bar:** tmidotnet-staging-b9eyfad7h6ewafby.malaysiawest-01.azurewebsites.net
- Toolbar:** Back, Forward, Stop, Refresh, Home, All Bookmarks, etc.
- Page Content:**
  - Header:** Web App Version 2
  - Text:** ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.
  - Button:** Learn more »
  - Section:** Getting started
  - Text:** ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.
  - Link:** Learn more »
  - Section:** Get more libraries
  - Text:** NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.
  - Link:** Learn more »
  - Section:** Web Hosting
  - Text:** You can easily find a web hosting company that offers the right mix of features and price for your applications.
  - Link:** Learn more »

At the bottom of the page, there is a footer with the text: © 2025 - My ASP.NET Application.

## Step 10: Deploy a web app by using deployment slots

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

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

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

## Step 11: Configure a slot setting

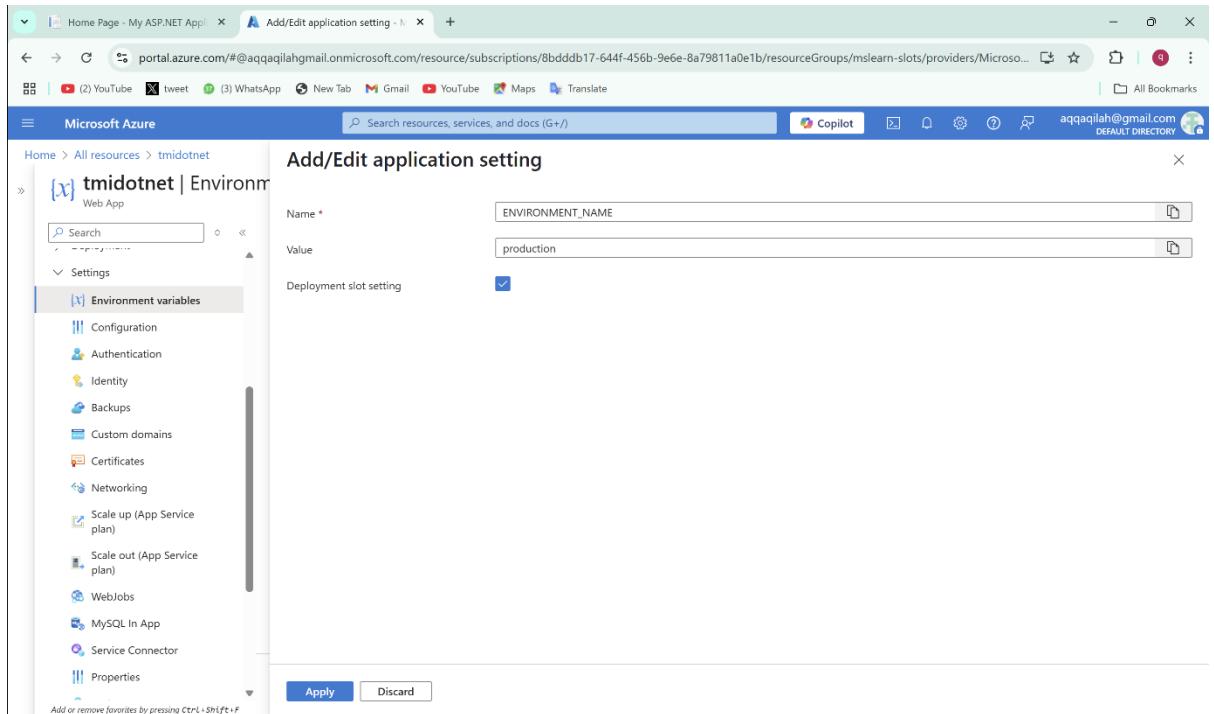
Before you deploy version 2 of the web app, configure a slot setting. The settings you'll configure here won't affect your demo app. The purpose of this exercise is just to see how the configurations work when you swap slots.

To configure slot settings:

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

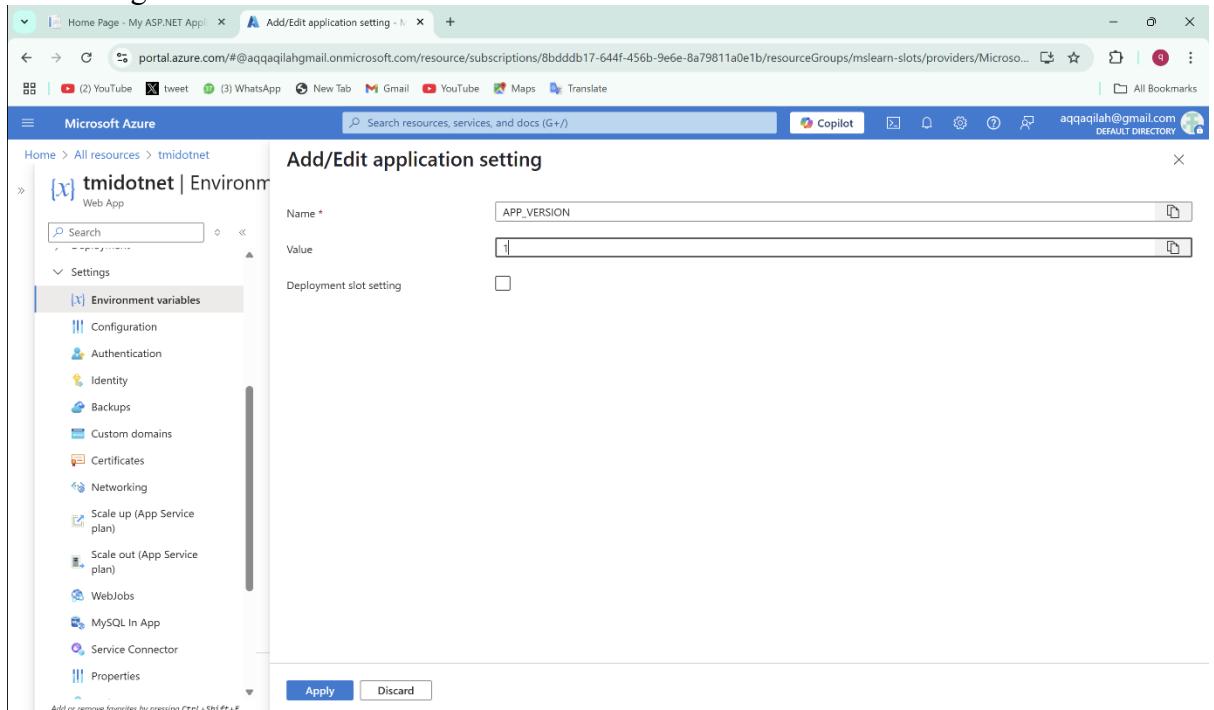
Name	Value	Deployment slot setting	Source
ASPNETCORE_ENVIRONMENT	Development	staging	Slot setting

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



The screenshot shows the Azure portal interface for managing an App Service. On the left, there's a sidebar with 'All resources' and 'tmidotnet' selected. Under 'Settings', 'Environment variables' is chosen. In the main area, a 'Add/Edit application setting' dialog is open. It has fields for 'Name' (set to 'ENVIRONMENT\_NAME'), 'Value' (set to 'production'), and a 'Deployment slot setting' checkbox which is checked. At the bottom of the dialog are 'Apply' and 'Discard' buttons.

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



This screenshot shows the same Azure portal interface and 'Add/Edit application setting' dialog as the previous one, but for a different setting. The 'Name' field is now 'APP\_VERSION', the 'Value' field is '1', and the 'Deployment slot setting' checkbox is unchecked. The rest of the interface is identical to the first screenshot.

## 5. select Apply.

The screenshot shows the Azure portal interface for managing environment variables. The left sidebar shows the navigation path: Home > All resources > tmidotnet. The main content area is titled 'tmidotnet | Environment variables'. Under 'App settings', there are two environment variables listed:

Name	Value	Deployment slot setting	Source	Delete
APP_VERSION	1		App Service	
ENVIRONMENT_NAME	production	✓	App Service	

At the bottom of the page are 'Apply' and 'Discard' buttons.

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

Name	Value	Deployment slot setting
ENVIRONMENT_NAME	staging	Yes
APP_VERSION	2	No

The screenshot shows the 'Add/Edit application setting' dialog for the 'staging' slot of the 'tmidotnet' app. The left sidebar shows the navigation path: Home > All resources > staging (tmidotnet/staging). The main content area is titled 'Add/Edit application setting' for 'ENVIRONMENT\_NAME'. The form fields are:

- Name: ENVIRONMENT\_NAME
- Value: staging
- Deployment slot setting:

At the bottom of the dialog are 'Apply' and 'Discard' buttons.

**Add/Edit application setting**

portal.azure.com/#@aqqaqilah@gmail.onmicrosoft.com/resource/subscriptions/8bdddb17-644f-456b-9e6e-8a79811a0e1b/resourceGroups/mslearn-slots/providers/Microsoft...

Microsoft Azure

Home > All resources > staging (tmidonet/staging)

**staging (tmidonet/staging)** App Service (Slot)

Search Overview Activity log Access control (IAM) Tags Diagnose and solve problems Microsoft Defender for Cloud Recommended services (preview) Resource visualizer Deployment Settings Environment variables Configuration Authentication Identity Backups Custom domains

Name \* APP\_VERSION  
Value 2  
Deployment slot setting

Apply Discard

Add or remove favorites by pressing Ctrl+Shift+F

**staging (tmidonet/staging) | Environment variables**

portal.azure.com/#@aqqaqilah@gmail.onmicrosoft.com/resource/subscriptions/8bdddb17-644f-456b-9e6e-8a79811a0e1b/resourceGroups/mslearn-slots/providers/Microsoft...

Microsoft Azure

Home > All resources > staging (tmidonet/staging)

**staging (tmidonet/staging)** App Service (Slot)

Search Overview Activity log Access control (IAM) Tags Diagnose and solve problems Microsoft Defender for Cloud Recommended services (preview) Resource visualizer Deployment Settings Environment variables Configuration Authentication Identity Backups Custom domains

**App settings** Connection strings

Name	Value	Deployment slot setting	Source	Delete
APP_VERSION	2		App Service	
ENVIRONMENT_NAME	staging	✓	App Service	
WEBSITE_NODE_DEFAULT_VERSION	Show value		App Service	

Apply Discard

Send us your feedback

## Step 12: Swap the slots

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

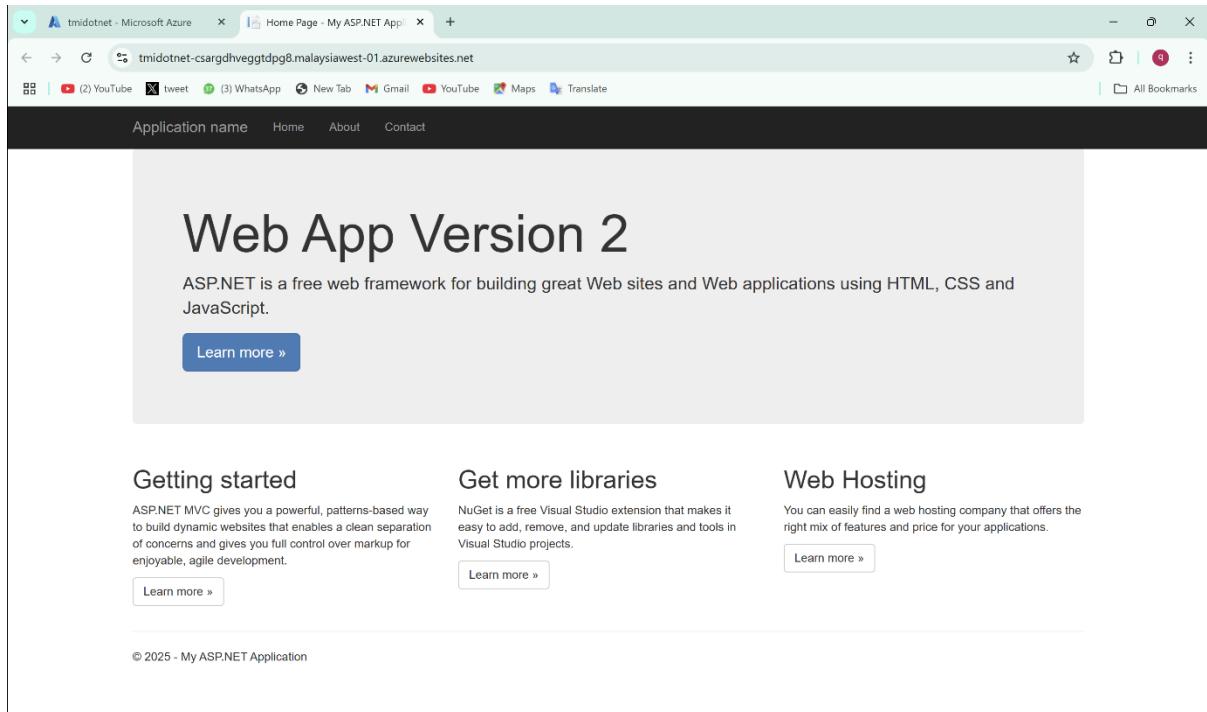
1. To make sure you're configuring the production slot, select **All resources**, and then select the production slot of the web app.
2. In the left menu pane, under **Deployment**, select **Deployment slots > Swap**.

Name	Status	App service plan	Traffic %
tmidotnet (PRODUCTION)	Running	ASP-mslearnslots-84c8	100
tmidotnet-staging	Running	ASP-mslearnslots-84c8	0

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

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

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

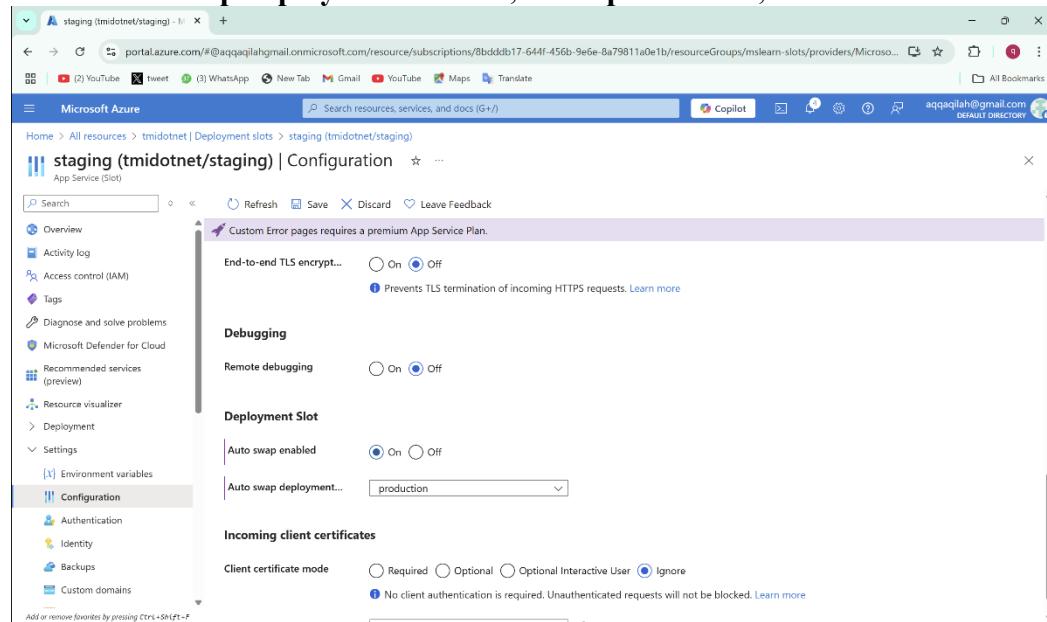


- Close the browser tab.

## Step 13: Configure auto swap for the staging slot

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

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



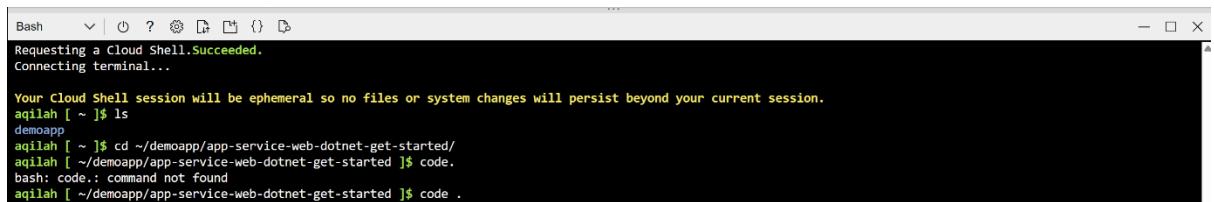
## Step 14: Deploy new code and auto swap it into production

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

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

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

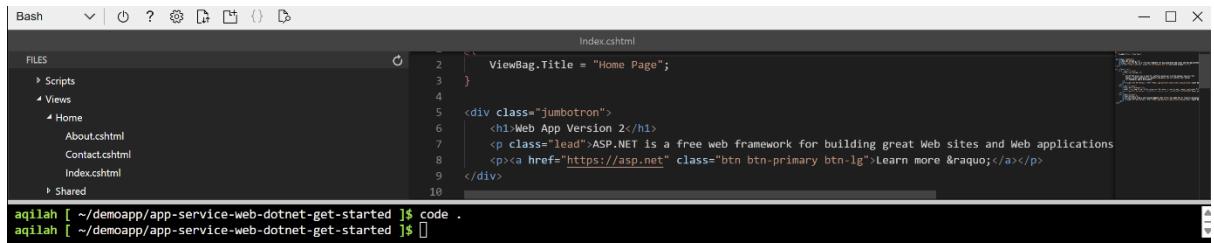
```
code .
```



The screenshot shows a terminal window titled 'Bash'. It displays the command 'code .' being run. The output shows a message about a ephemeral session and lists several files: 'about.cshtml', 'Contact.cshtml', 'Index.cshtml', and 'Shared'. The command 'code .' is run again, and the response 'bash: code.: command not found' is shown.

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.

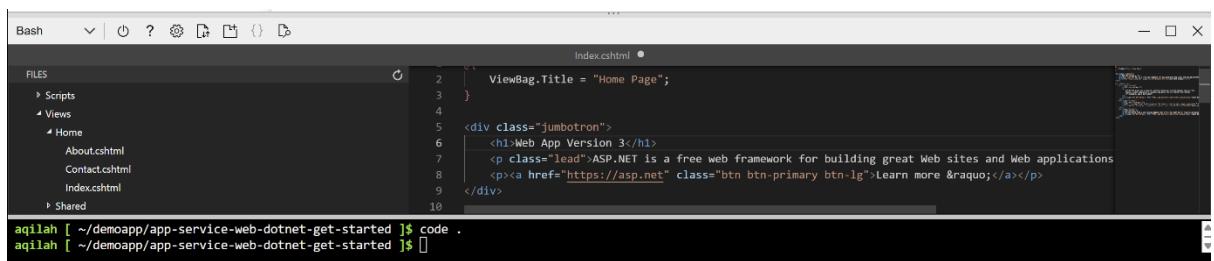
```
<h1>Web App Version 2</h1>
```



The screenshot shows the 'Index.cshtml' file in the code editor. The code includes a title assignment and a jumbotron section with an h1 header containing 'Web App Version 2'. Below the h1 is a paragraph with a link to 'asp.net'. The file path 'aspnet-get-started/Views/Home/Index.cshtml' is visible in the sidebar.

4. Replace that code with this code.

```
<h1>Web App Version 3</h1>
```



The screenshot shows the 'Index.cshtml' file in the code editor after modification. The h1 header now contains 'Web App Version 3'. The rest of the code remains the same, including the title assignment and the jumbotron section with its paragraph and link.

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

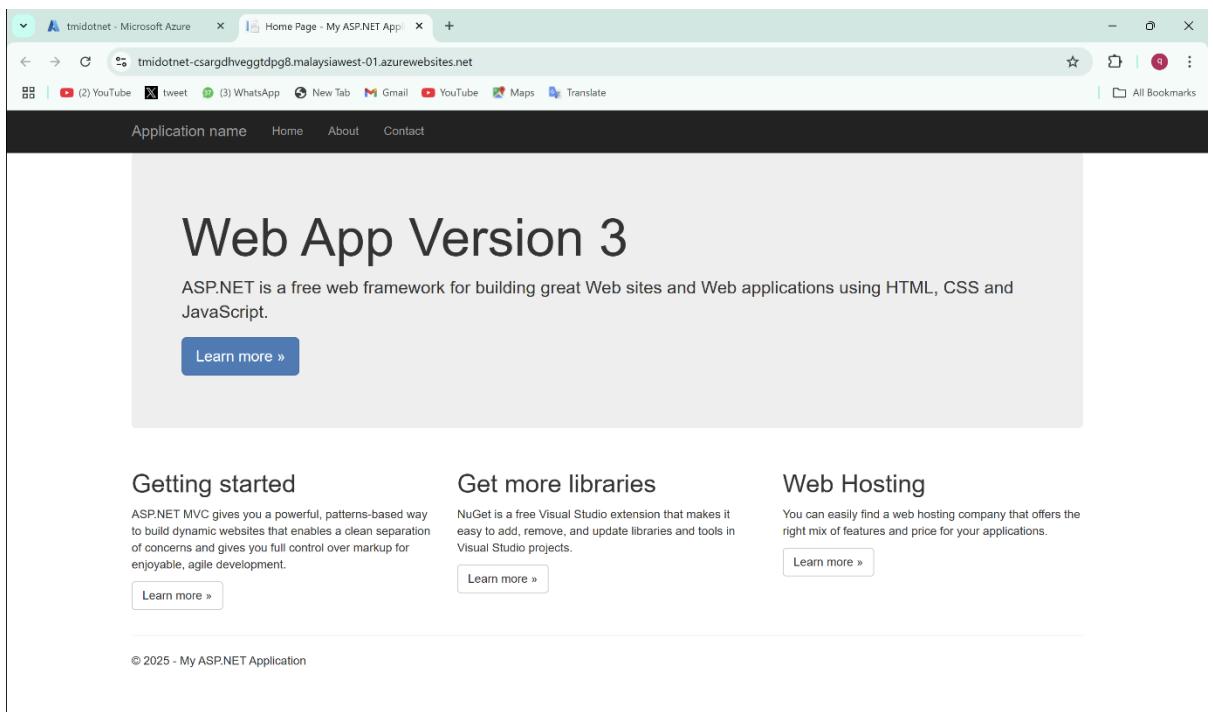
```
git add .
```

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

git push staging

```
[main 5504ab3] Third version of web app.
 1 file changed, 1 insertion(+), 1 deletion(-)
aqilah [ ~/demoapp/app-service-web-dotnet-get-started ]$ git push staging
Password for 'https://lala2.0@midonet-staging-b9eyfad7h6ewafby.scm.malaysiawest-01.azurewebsites.net':
fatal: Authentication failed for 'https://midonet-staging-b9eyfad7h6ewafby.scm.malaysiawest-01.azurewebsites.net/midotnet.git/'
aqilah [ ~/demoapp/app-service-web-dotnet-get-started ]$ git push staging
Password for 'https://lala2.0@midonet-staging-b9eyfad7h6ewafby.scm.malaysiawest-01.azurewebsites.net':
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 3 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (12/12), 975 bytes | 975.00 KiB/s, done.
Total 12 (delta 10), reused 0 (delta 0), pack-reused 0 (from 0)
```

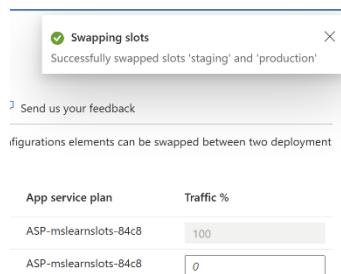
7. Wait for the deployment to finish. Near the end of the text output, you'll see a message that indicates that the deployment has requested an auto swap to the production slot.
  8. In the Azure portal, navigate to the Overview page for the production slot's web app, and select Browse. The third version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



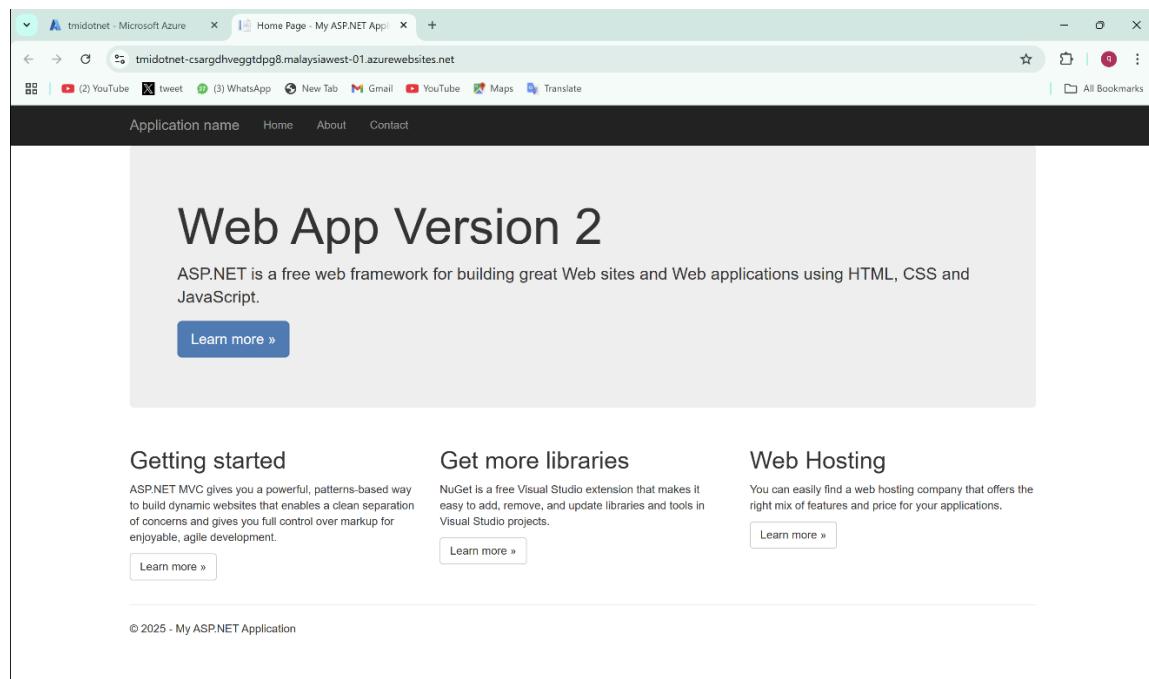
## Step 15: Roll back the new version

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

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



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



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

Procedure:

### Step 1: Sign in to Azure

### Step 2: Create a Web App

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

2.1 Click Create a resource

2.2 Search and select Web App

2.3 Fill out the Basics tab

**Project Details**

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ Azure subscription 1

Resource Group \* ⓘ (New) mslearn-scale

Create new

**Instance Details**

Name shhotelsystem.azurewebsites.net

Try a secure unique default hostname. [More about this update](#)

Publish \* Code Container

Runtime stack \* .NET 8 (LTS)

Operating System \* Linux Windows

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

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

## Step 3: Download the App Source Code

### 3.1 Open Cloud Shell

3.2 Run this command to download the hotel app code:

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

## Step 4: Build the Web App

4.1 Change to the folder with the source code:

4.2 Build everything:

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

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

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

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

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

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

Wait until it says Build succeeded.

```
Build succeeded.          37 Warning(s)
                           0 Error(s)
```

## Step 5: Prepare the App for Deployment

5.1 Go into the web app folder:

5.2 Publish it to a folder:

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

5.3 Zip the files:

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

## Step 6: Deploy to Azure

### 6.1 Deploy your code to the Azure Web App

```
shaiful [ ~/mslearn-hotel-reservation-system/src/HotelReservationSystem/website ]$ az webapp deployment source config-zip --src website.zip --name shyfulhotelsystem --resource-group mslearn
-scan
This command has been deprecated and will be removed in a future release. Use 'az webapp deploy' instead.
Getting scm site credentials for zip deployment
Starting zip deployment. This operation can take a while to complete ...
Deployment endpoint responded with status code 202
Polling the status of async deployment. Start Time: 2025-05-15 14:44:46.170560+00:00 UTC
{
  "active": true,
  "author": "N/A",
  "author_email": "N/A",
  "complete": true,
  "deployer": "zipDeploy",
  "end_time": "2025-05-15T14:44:49.1701258Z",
  "id": "f8d327ad9c9f44989db8e076b9c3b9d3",
  "is_readonly": true,
  "is_temp": false,
  "last_success_end_time": "2025-05-15T14:44:49.1701258Z",
  "log_url": "https://shyfulhotelsystem-fmdhhngje2h3dpdc.scm.canadacentral-01.azurewebsites.net/api/deployments/latest/log",
  "message": "Created via a push deployment",
  "progress": "",
  "provisioningState": "Succeeded",
  "received_time": "2025-05-15T14:44:46.6870957Z",
  "site_name": "shyfulhotelsystem",
  "start_time": "2025-05-15T14:44:46.8576512Z",
  "status": 4,
  "status_text": "",
  "url": "https://shyfulhotelsystem-fmdhhngje2h3dpdc.scm.canadacentral-01.azurewebsites.net/api/deployments/latest"
}
```

## Step 7: Test Your Web App

Open your browser and visit:

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

```
Pretty-print ▾
{"reservationID":1,"customerID":"662357894","hotelID":"Hotel_2056694718","checkin":"2025-05-26T00:13:05.7569334+00:00","checkout":"2025-07-18T04:13:50.7565924+00:00","numberOfGuests":1,"reservationComments":"u0g9D1CkOYwA+BUcfV1AbIyceXAjWvC14V/tQEtl6xxevKlw/0m6qr7J0+h7Jg/TgIDtQ+AX3G103/smA3QhfBo02drO1l7j4xqBdYKhnlkKPgiI44bWK18FoyQFYxASwC20rhqhV4CigJepeaux7Rc4/Syu0Mts467I/hAKyh07zid968fv119DpYvxEzzW0mIKKPH/6m/gIJF0LYF15v00A2z6yXT7VX0azw=="}
```

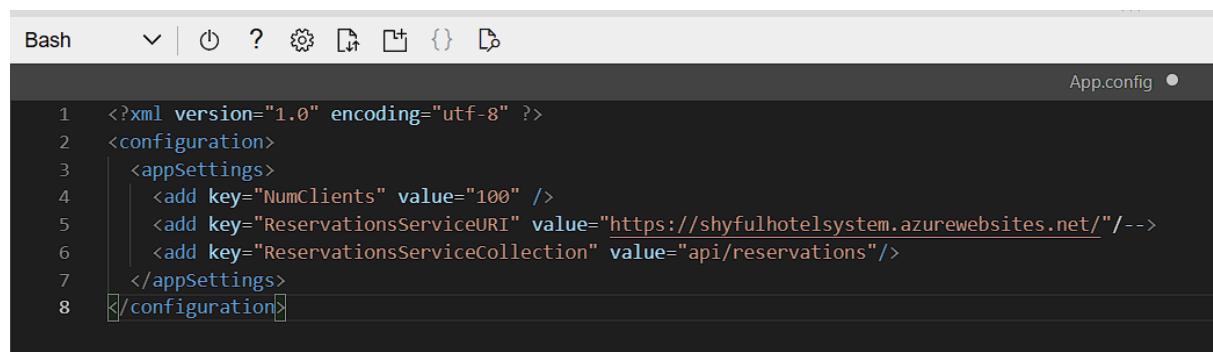
## Step 8: Prepare the Test Client (Simulates Many Users)

8.1 Go to the test client folder:

8.2 Open the config file:

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

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



```
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://shyfulhotelsystem.azurewebsites.net/" /-->
6  <add key="ReservationsServiceCollection" value="api/reservations"/>
7  </appSettings>
8  </configuration>
```

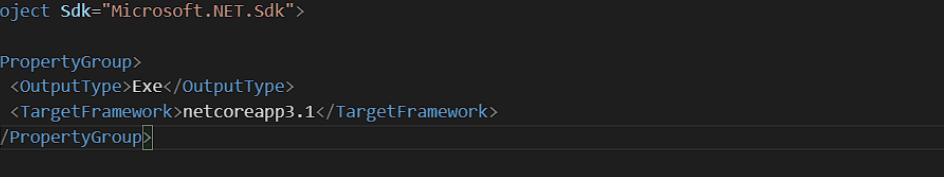
## **Step 9: Build and Fix the Framework**

### 9.1 Rebuild the test client:

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

## 9.2 If you get errors, edit this file:

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



```
Bash ▾ | ⌁ ? ⌂ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉
HotelReservationSystemTestClient.csproj ●

1 <Project Sdk="Microsoft.NET.Sdk">
2
3   <PropertyGroup>
4     <OutputType>Exe</OutputType>
5     <TargetFramework>netcoreapp3.1</TargetFramework>
6   </PropertyGroup>
7
8   <ItemGroup>
9     <PackageReference Include="Newtonsoft.Json" Version="12.0.1" />
10    <PackageReference Include="System.Configuration.ConfigurationManager" Version="4.5.0" />
11  </ItemGroup>
12
13  <ItemGroup>
14    <ProjectReference Include="..\HotelReservationSystemTypes\HotelReservationSystemTypes.csproj" />
15  </ItemGroup>
16
17 </Project>
18
```

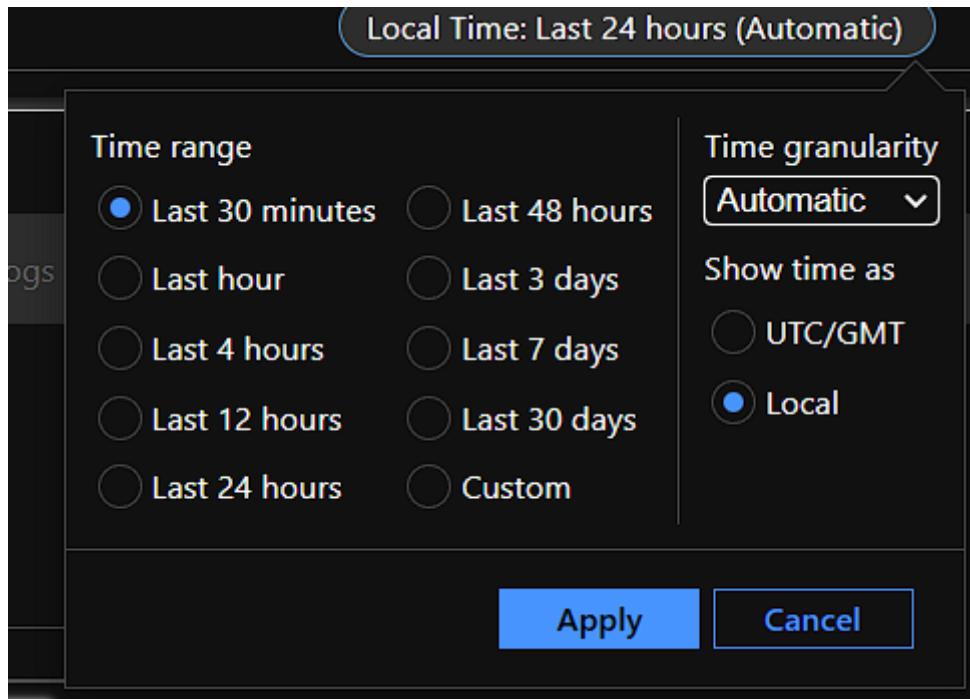
**Save and close.**

### **Step 10: Run the Client (Simulate 100 Users)**

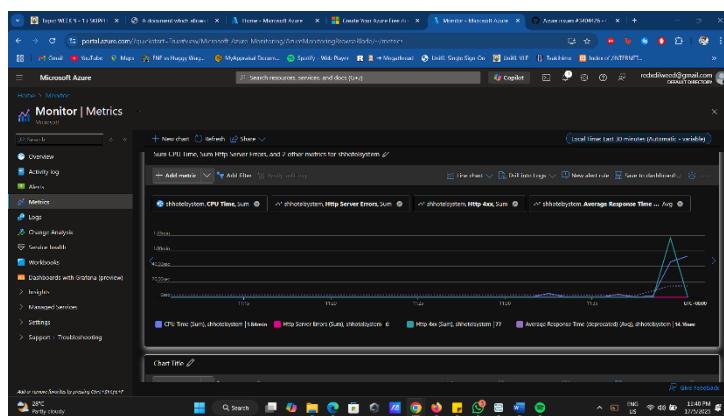
## Step 11: Monitor Web App Performance

### In Azure Portal:

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



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



5. Click **Pin to dashboard**

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

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

## Step 12: Scale Out the App

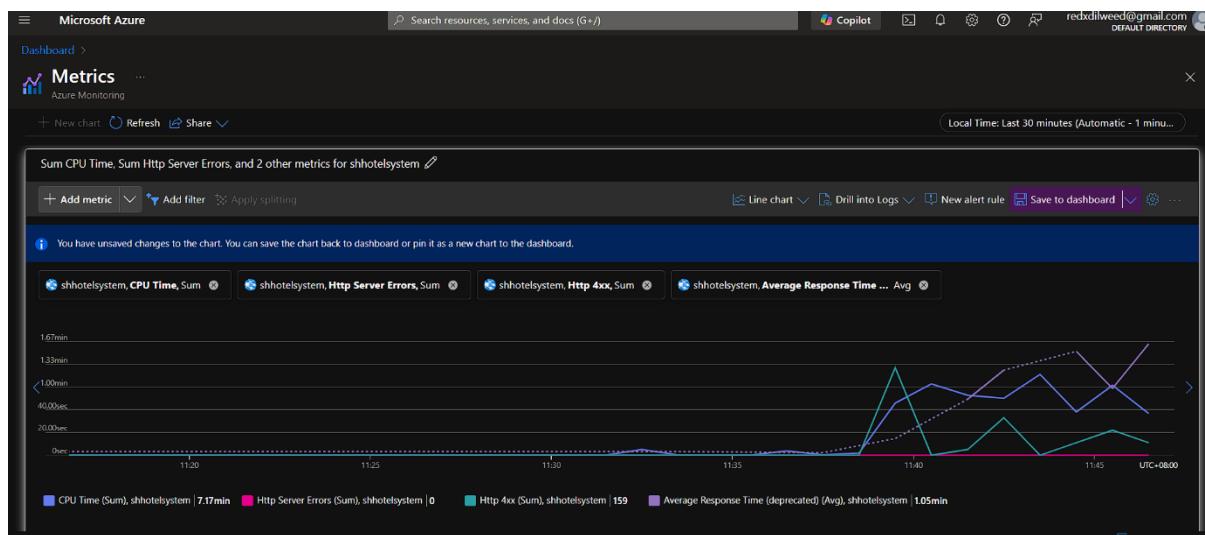
### In Azure Portal:

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

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

You'll start seeing:

- Fewer timeouts
- Lower response times
- Improved performance



## Step 14: Stop Everything

Once done:

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

## Step 15: Scale up

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

SKU	Cores	Memory (GB)	ACU	Request Rate (req/min)	Bandwidth (Mbps)	Price (USD)	Max Concurrency (req/sec)
Basic B2	100	2	3.5	10	3	0.165 USD	120.45 USD
Basic B3	100	4	7	10	3	0.33 USD	240.90 USD
Production (For most production workloads)							
Premium v3 P0V3	195*	1	4	250	30	0.164 USD	119.72 USD
Premium v3 P1V3	195	2	8	250	30	0.328 USD	239.44 USD
Premium v3 P1mv3	195*	2	16	250	30	0.362 USD	263.968 USD
Premium v3 P2V3	195	4	16	250	30	0.656 USD	478.88 USD
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							
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

## Step 16: Run the test client app

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

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

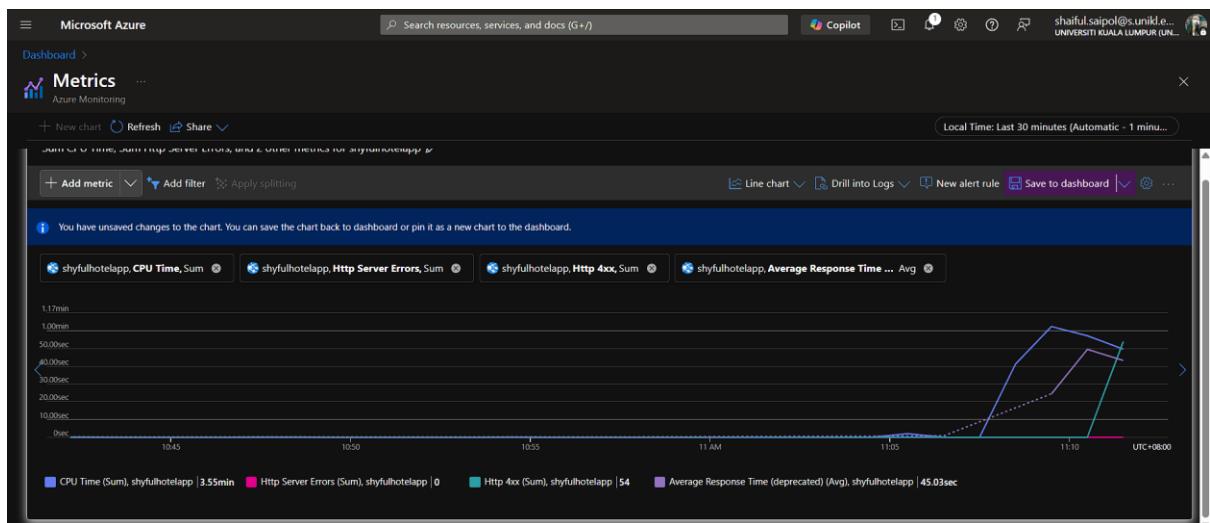
4. Let the app run for about five minutes. Monitor the performance metrics in the Azure portal (as described in the previous response). You'll likely see high CPU usage and response times.

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

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

SKU	Cores	Memory (GB)	ACU	Request Rate (req/min)	Bandwidth (Mbps)	Cost (USD)	Scalability (ACU)
Premium v3 P2Mv3	4	32	250	30	0.723 USD	527.936 USD	195*
Premium v3 P3Mv3	8	64	250	30	1.446 USD	1055.872 USD	195*
Premium v3 P4Mv3	16	128	250	30	2.893 USD	2111.744 USD	195*
Premium v3 P5Mv3	32	256	250	30	5.786 USD	4223.488 USD	195*
Legacy							
Standard S1	1	1.75	50	10	0.11 USD	80.30 USD	100
Standard S1	1	1.75	50	10	0.11 USD	80.30 USD	100
Standard S2	2	3.5	50	10	0.22 USD	160.60 USD	100
Standard S3	4	7	50	10	0.44 USD	321.20 USD	100
Premium P1	1	1.75	250	20	0.33 USD	240.90 USD	100
Premium P2	2	3.5	250	20	0.66 USD	481.80 USD	100
Premium P3	4	7	250	20	1.32 USD	963.60 USD	100
Premium v2 P1V2	1	3.5	250	30	0.22 USD	160.60 USD	210
Premium v2 P2V2	2	7	250	30	0.44 USD	321.20 USD	210
Premium v2 P3V2	4	14	250	30	0.88 USD	642.40 USD	210

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



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

## Conclusion

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