Microsoft Azure

App Service

Deploy, Secure & Manage Web Applications — Hands-On Lab Guide

LAB 1	LAB 2	LAB 3	LAB 4
Create & Deploy Web App + Custom Domain + TLS	Basic Auth + Google OAuth (Easy Auth)	Advanced Tools — App Editor, FTP, Storage	Instances, Backups, Plan & Scaling

Ahmed Abdelwahed

Cloud & Systems Specialist | Technical Trainer

Email	Website	LinkedIn
ahmed_abdulwahed@outlook.com	www.abdelwahed.me	www.linkedin.com/in/ahmedmct

LAB 1 — Deploy Web App + Custom Domain + TLS Certificate

LAB 1: Create and Configure an Azure App Service Web App

⏱ Duration: 60–75 min	🎯 Level: Intermediate	💰 Cost: ~$0.05–0.10/hr — Basic B1 plan. Delete after lab.

Prerequisites

Active Azure subscription with Contributor or Owner role
A registered domain name (lab uses: linux-hub.net)
FileZilla FTP client installed: https://filezilla-project.org
Azure CLI installed or use Azure Cloud Shell

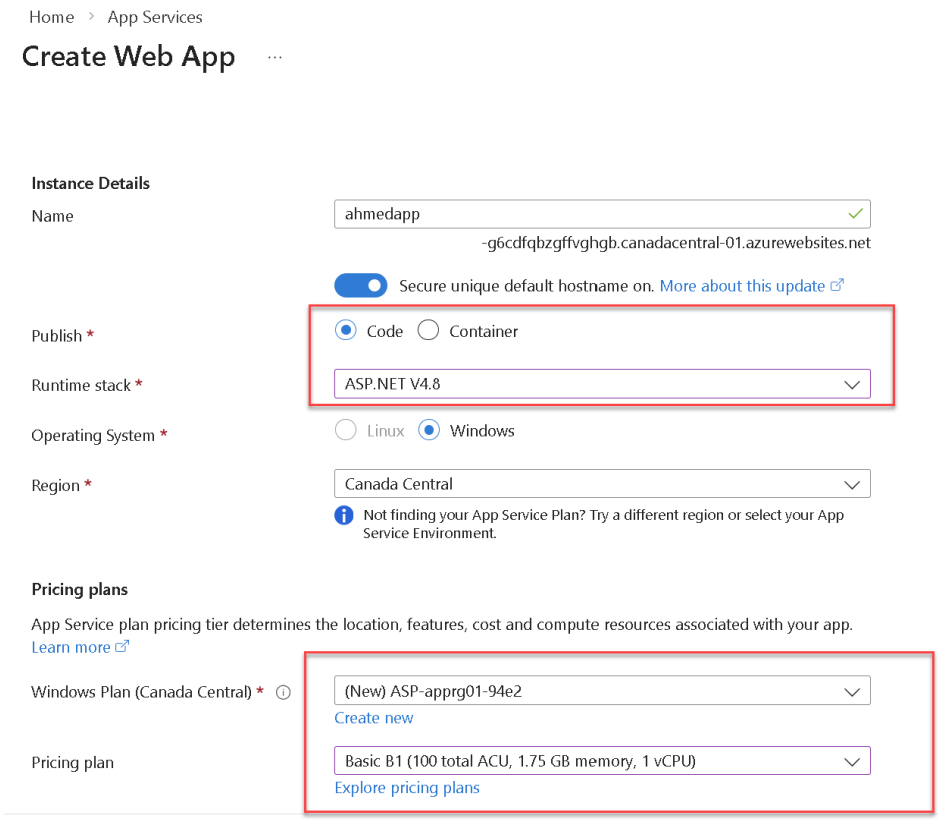
Lab 1.1 — Create the Web App

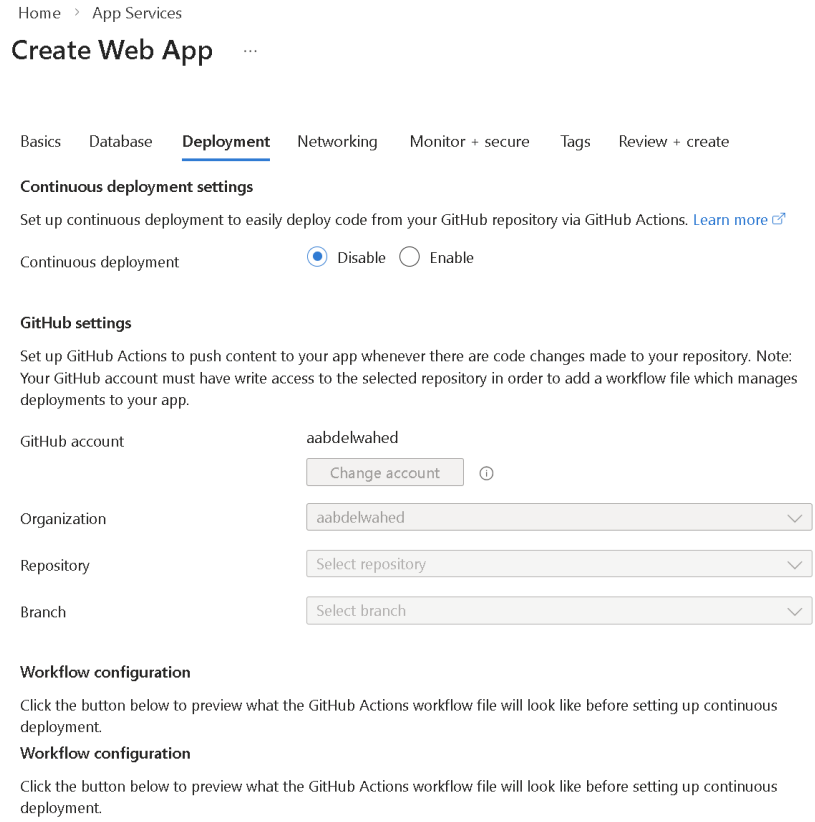
Create the App Service Plan and Web App from the Azure Portal. The App Service Plan defines the compute tier — use Basic B1 for this lab.

1 — Open Azure Portal → App Services
Navigate to portal.azure.com → search "App Services" → click Create.

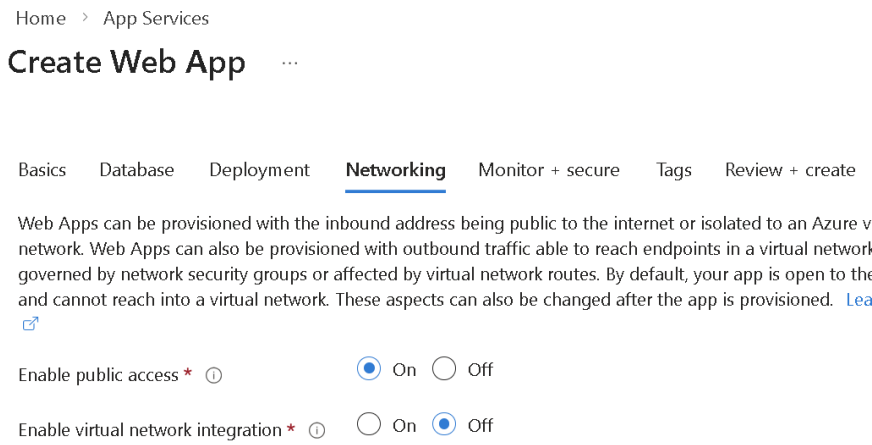
2 — Configure the basics
Subscription: your MCT subscription · Resource Group: app_rg01 (create new) · Name: ahmedapp · Runtime stack: your choice (PHP, Node, .NET) · Region: Canada Central · Plan: Basic B1

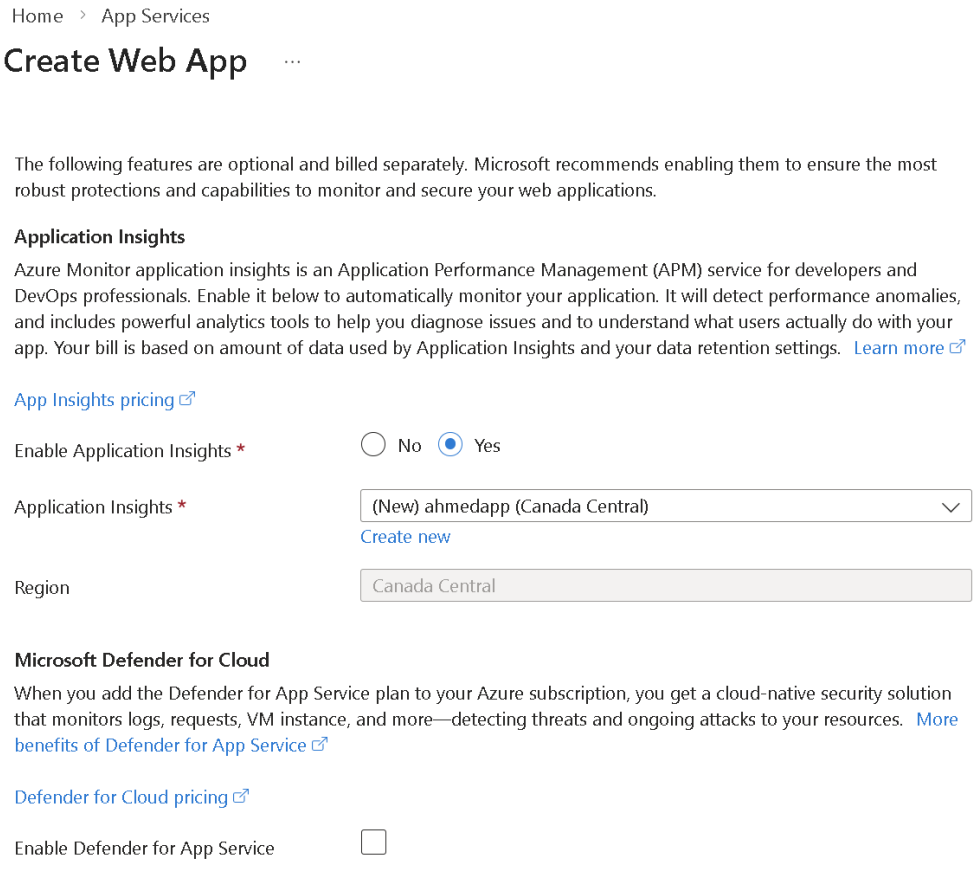
3 — Review + Create
Click Review + Create → verify settings → click Create. Wait ~2 minutes for deployment.





Workflow configuration
Click the button below to preview what the GitHub Actions workflow file will look like before setting up continuous deployment.


Home › App Services
Create Web App
Basics Database Deployment Networking Monitor + secure Tags Review + create
Web Apps can be provisioned with the inbound address being public to the internet or isolated to an Azure virtual network. Web Apps can also be provisioned with outbound traffic able to reach endpoints in a virtual network, be governed by network security groups or affected by virtual network routes. By default, your app is open to the internet and cannot reach into a virtual network. These aspects can also be changed after the app is provisioned. Learn more
Enable public access * On Off
Enable virtual network integration * On Off

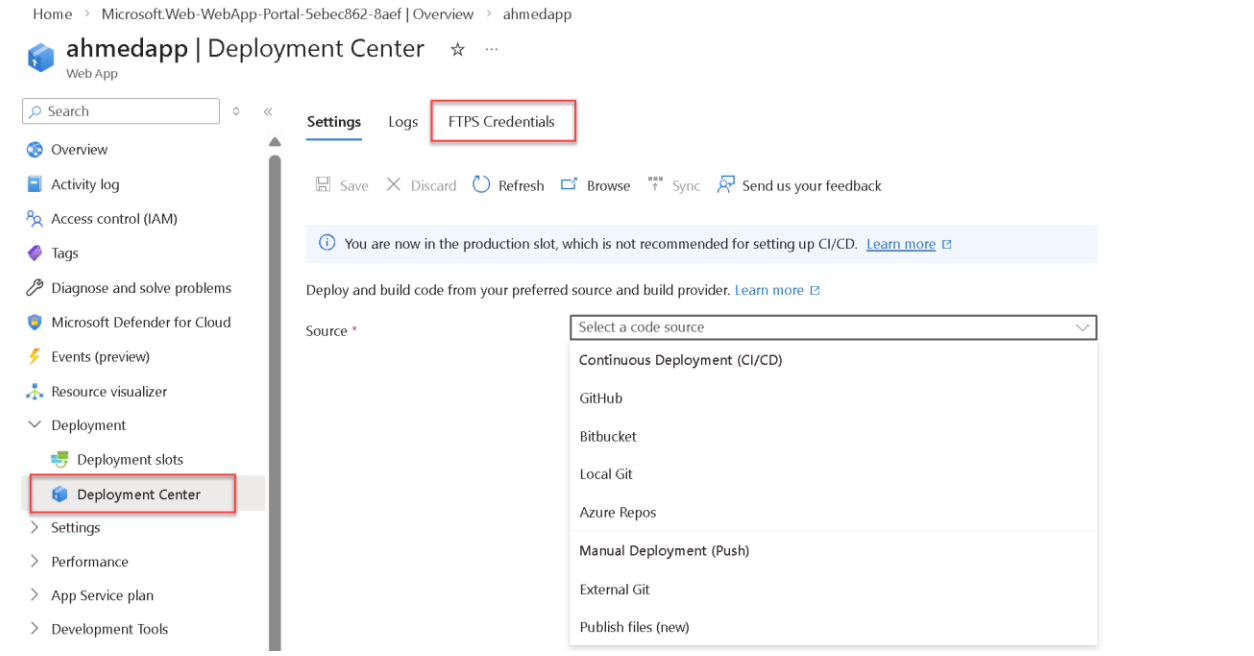

Home › App Services
Create Web App
The following features are optional and billed separately. Microsoft recommends enabling them to ensure the most robust protections and capabilities to monitor and secure your web applications.
Application Insights
Azure Monitor application insights is an Application Performance Management (APM) service for developers and DevOps professionals. Enable it below to automatically monitor your application. It will detect performance anomalies, and includes powerful analytics tools to help you diagnose issues and to understand what users actually do with your app. Your bill is based on amount of data used by Application Insights and your data retention settings. Learn more
App Insights pricing
Enable Application Insights * No Yes
Application Insights * (New) ahmedapp (Canada Central)
Create new
Region Canada Central
Microsoft Defender for Cloud
When you add the Defender for App Service plan to your Azure subscription, you get a cloud-native security solution that monitors logs, requests, VM instance, and more—detecting threats and ongoing attacks to your resources. More benefits of Defender for App Service
Defender for Cloud pricing
Enable Defender for App Service

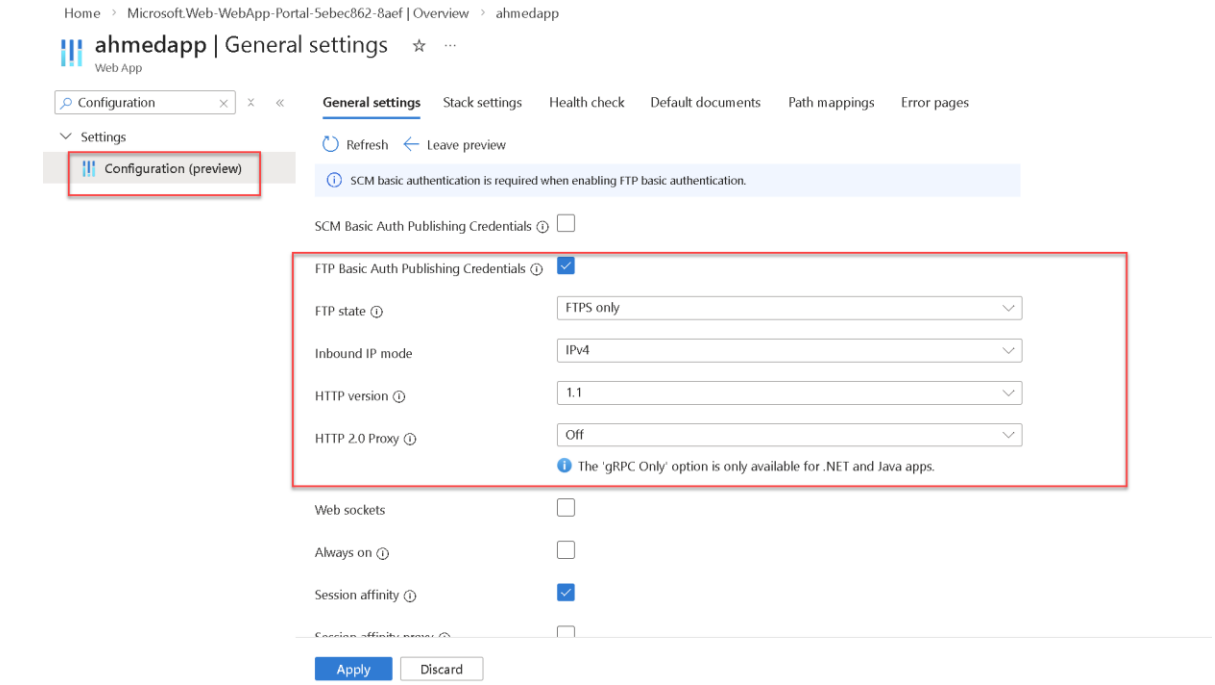
Lab 1.2 — Add a Website Data Source

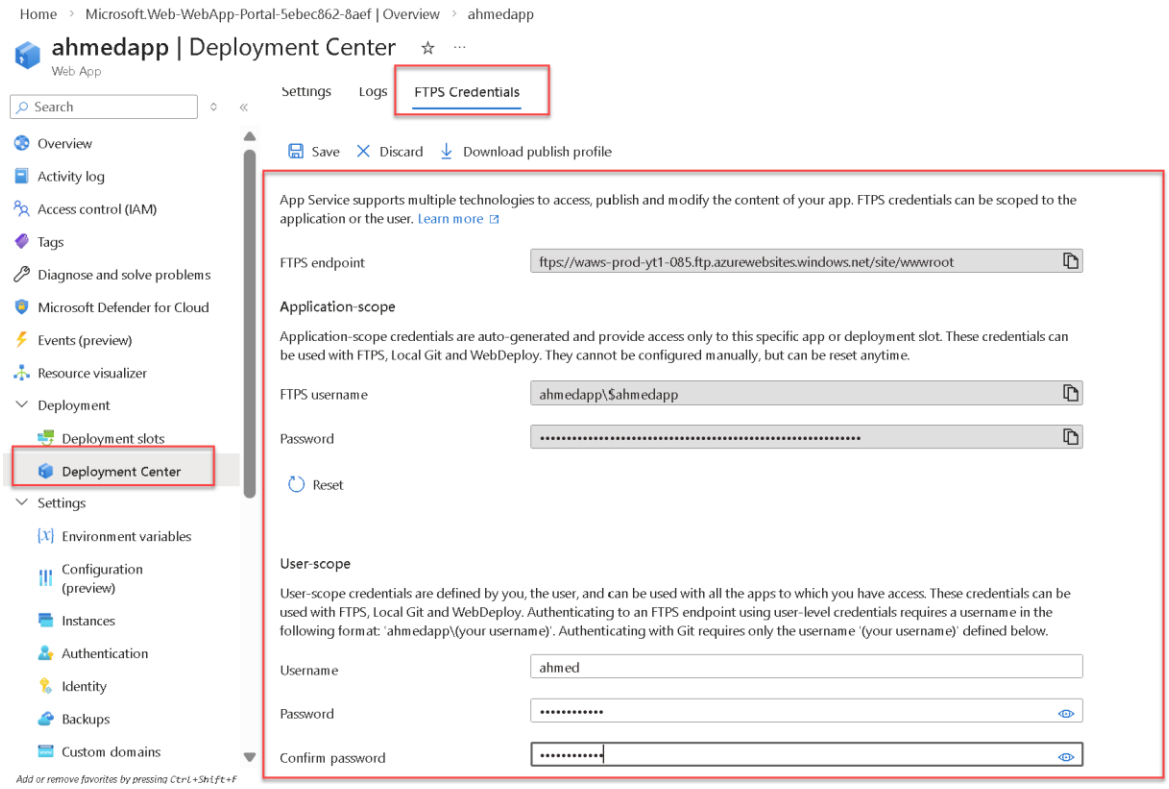
Upload your HTML website files to the web app using FTP, then verify they are served at the App Service URL before configuring the custom domain.

1 **Enable FTP access**

App Service → Deployment Center → FTPS credentials → set a username and password. Note the FTPS endpoint URL.







2

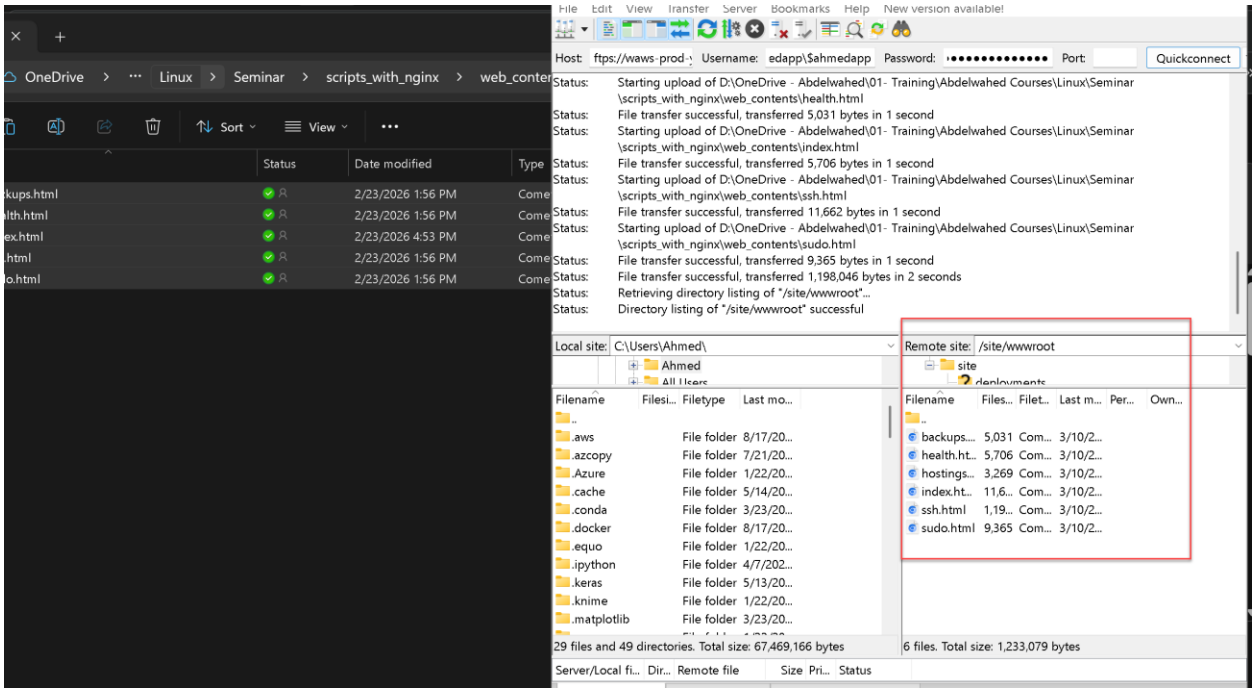
Connect with FileZilla

Open FileZilla → File → Site Manager → New Site. Protocol: FTP over TLS. Host: your FTPS endpoint. Username and password from Step 1.

3

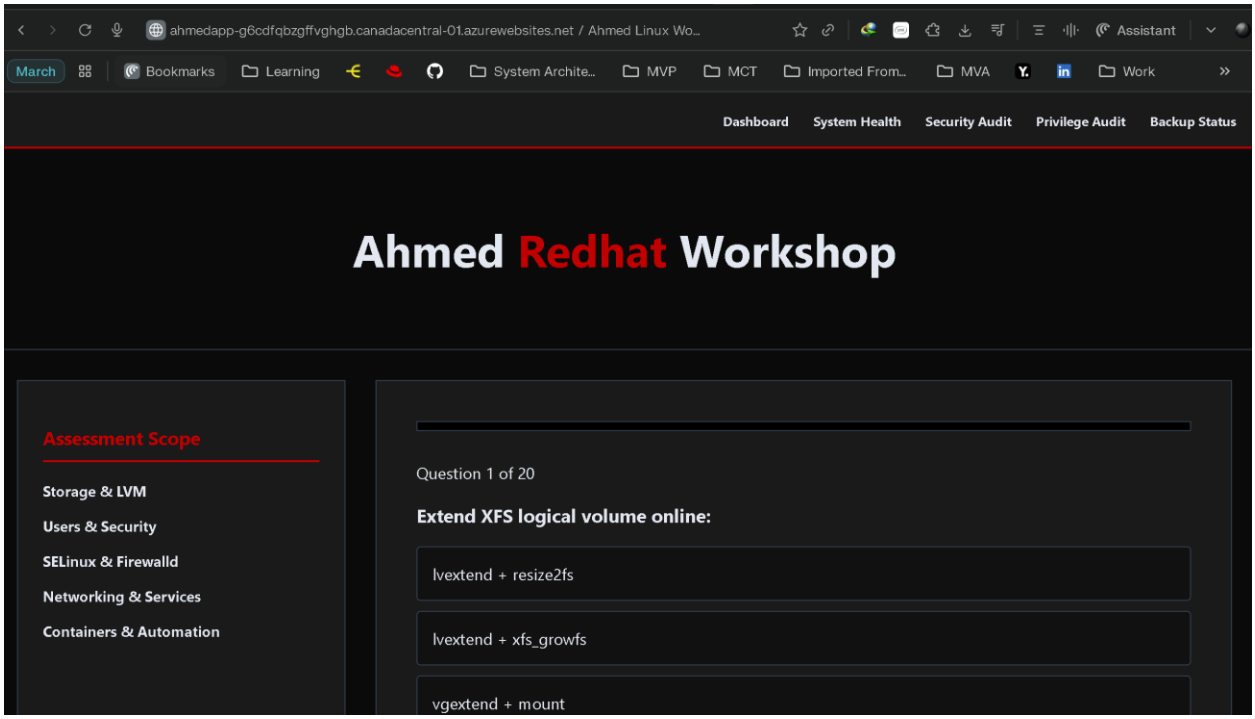
Upload site files

In FileZilla right panel, navigate to /site/wwwroot. Drag and drop your HTML files from your local machine into wwwroot.



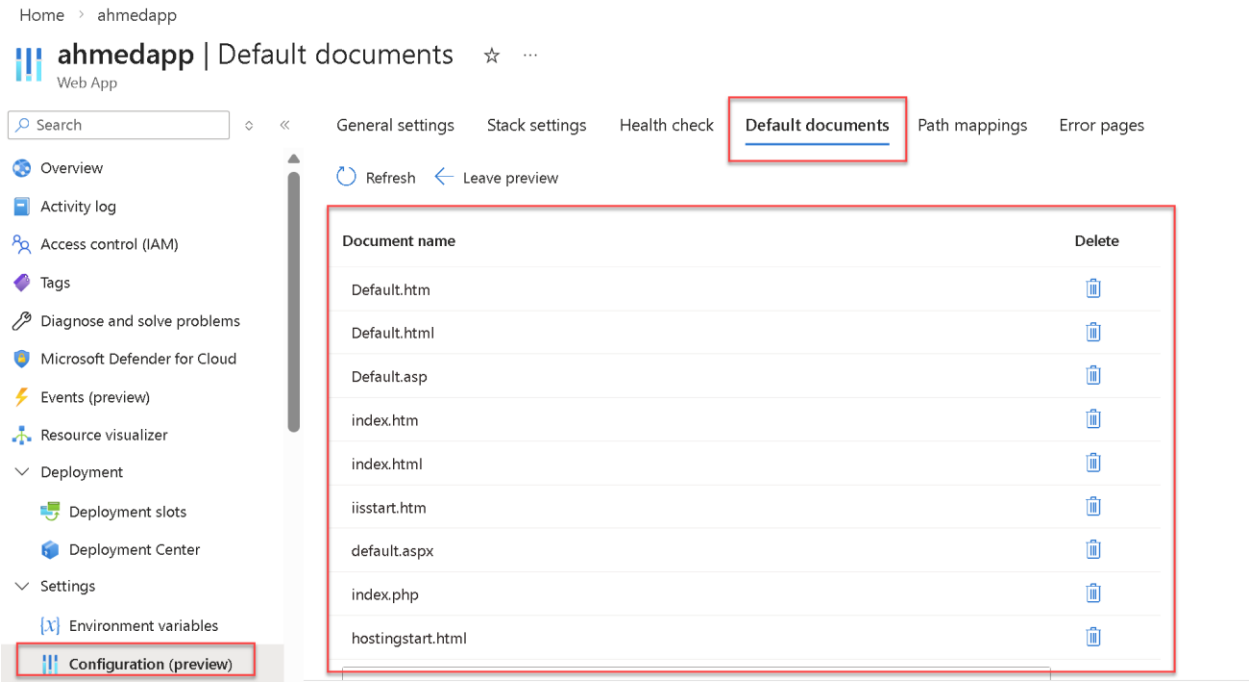
4 **Verify the upload**

Browse to https://ahmedapp.azurewebsites.net — your website should load with the uploaded content.



Note

App Service → Configuration → Application settings. You can reorder or manage included pages from the App Service editor. It is best to use the built-in App Editor for edits since it does not auto-save to production.



Lab 1.3 — Create an App Service Domain

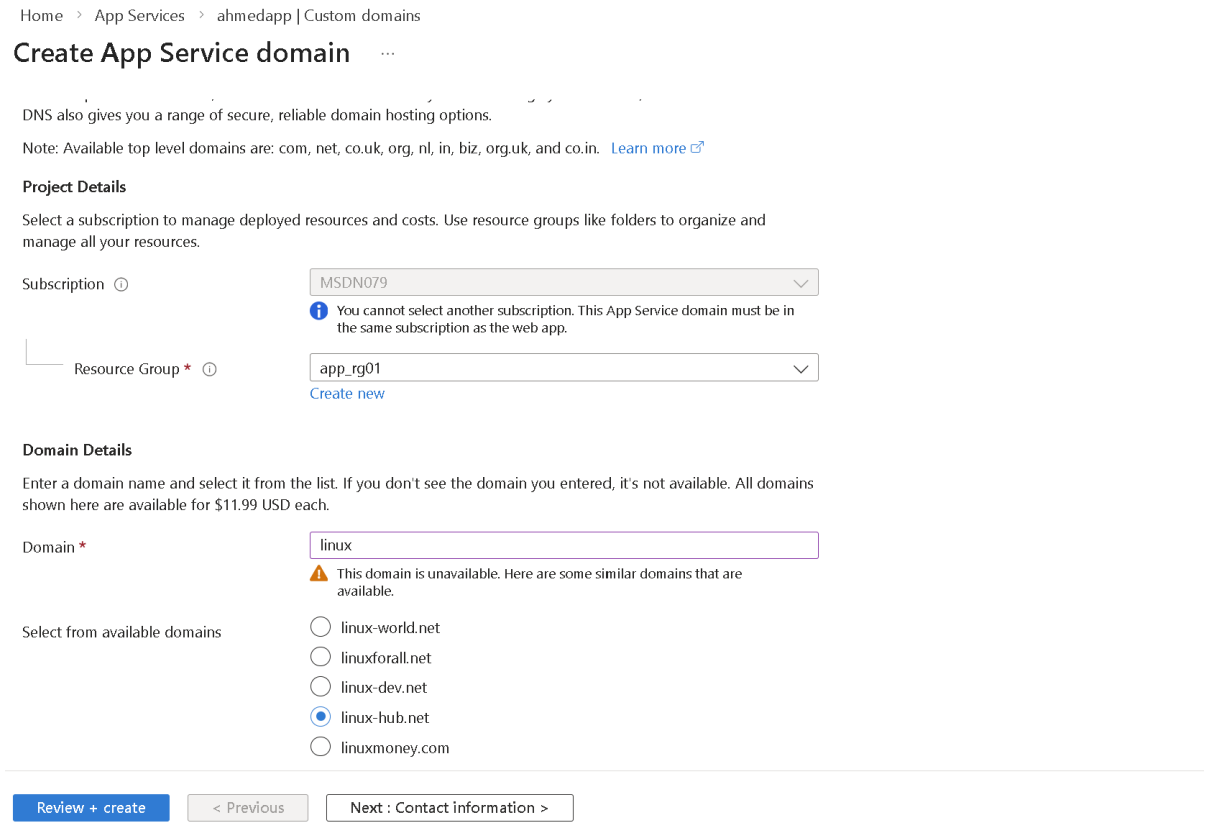
Purchase and configure a custom domain directly through Azure. This creates a DNS zone and links it to the App Service automatically.

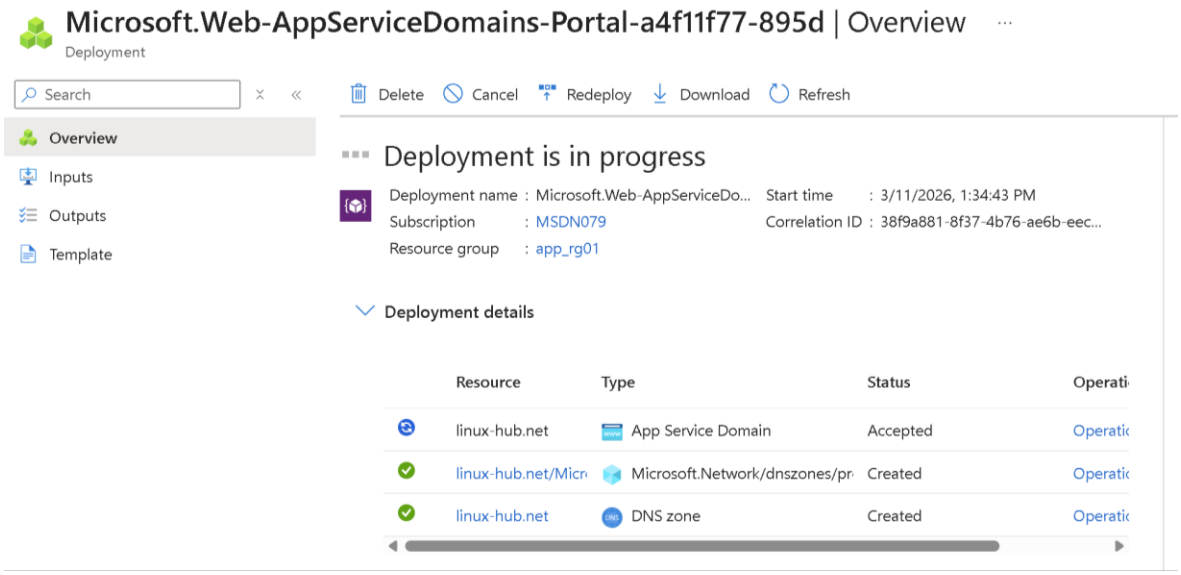
1 **Open App Service Domains**
Azure Portal → search "App Service Domains" → Create.

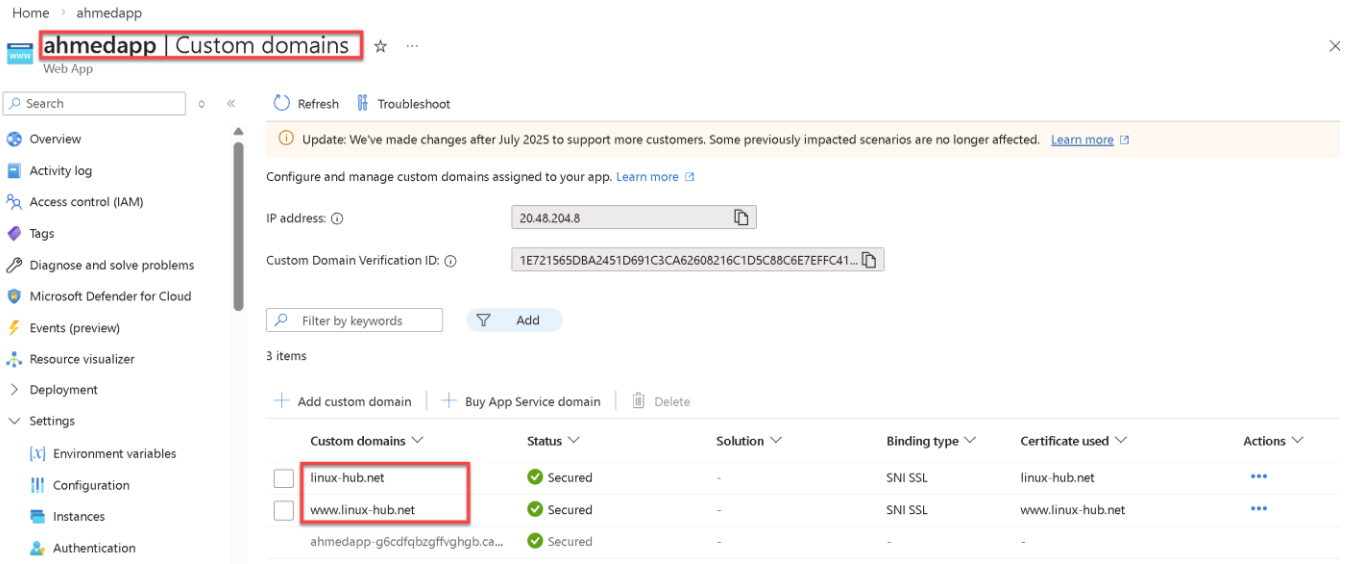
2 **Configure the domain**
Domain: linux-hub.net · Contact info: fill required fields · Auto-renew: Enabled (recommended) · Privacy protection: Enabled

3 **Add custom hostname to App Service**
App Service → Custom domains → Add custom domain → enter linux-hub.net → Validate → Add.

4 **Verify DNS propagation**
It may take up to 10 minutes. Test: nslookup linux-hub.net or ping linux-hub.net from terminal.




Microsoft.Web-AppServiceDomains-Portal-a4f11f77-895d | Overview
Deployment
Search
Overview
Inputs
Outputs
Template
Delete
Cancel
Redeploy
Download
Refresh
Deployment is in progress
Deployment name : Microsoft.Web-AppServiceDo...
Subscription : MSDN079
Resource group : app_rg01
Start time : 3/11/2026, 1:34:43 PM
Correlation ID : 38f9a881-8f37-4b76-ae6b-eec...
Deployment details
Resource
Type
Status
Operati
linux-hub.net
App Service Domain
Accepted
linux-hub.net/Micr
Microsoft.Network/dnszones/pr
Created
linux-hub.net
DNS zone
Created


Home
ahmedapp
ahmedapp | Custom domains
Web App
Search
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Microsoft Defender for Cloud
Events (preview)
Resource visualizer
Deployment
Settings
Environment variables
Configuration
Instances
Authentication
Refresh
Troubleshoot
Update: We've made changes after July 2025 to support more customers. Some previously impacted scenarios are no longer affected. Learn more
Configure and manage custom domains assigned to your app. Learn more
IP address: 20.48.204.8
Custom Domain Verification ID: 1E721565DBA2451D691C3CA626082 16C1D5C88C6E7EFFC41...
Filter by keywords
Add
3 items
Add custom domain
Buy App Service domain
Delete
Custom domains
Status
Solution
Binding type
Certificate used
Actions
linux-hub.net
Secured
SNI SSL
linux-hub.net
www.linux-hub.net
Secured
SNI SSL
www.linux-hub.net
ahmedapp-g6cdfqbzgffvghgb.ca...
Secured

Lab 1.4 — Configure TLS Certificate

Azure offers two options for TLS on App Service: Let's Encrypt (via extension) or Azure Managed Certificates (via CLI). The CLI method is more reliable and recommended for production.

> **Note**
> App Service → Extensions → Add → Let's Encrypt. Requires a storage account for certificate storage. Follow the wizard and provide the Service Principal credentials below.

Option A — Create Service Principal and Storage Account for Let's Encrypt

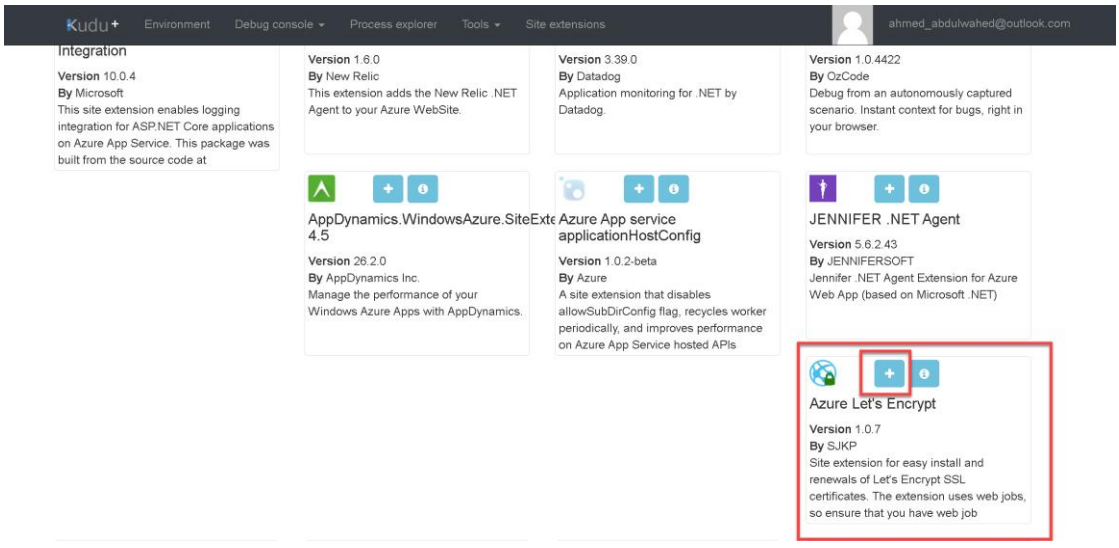
Azure CLI

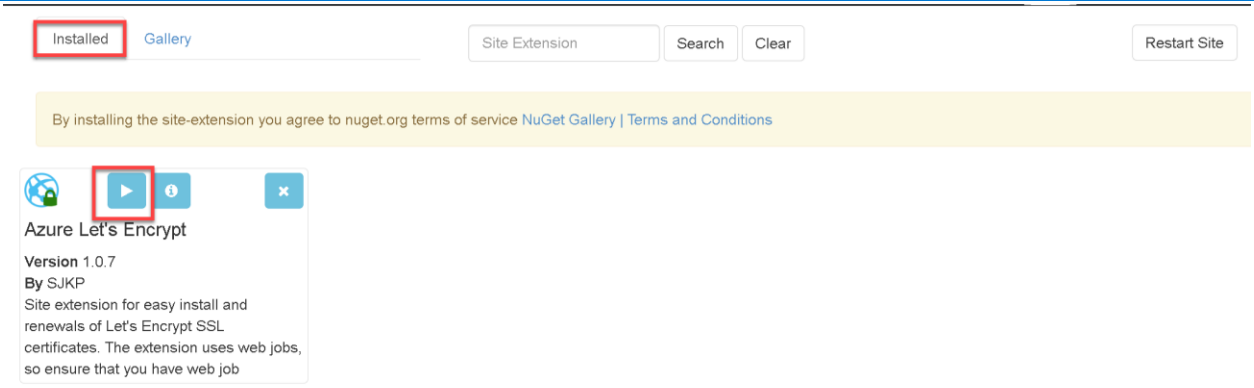
```
# Step 1: Create a Service Principal for Let's Encrypt extension
az ad sp create-for-rbac \
  --name "letsencrypt-sp" \
  --role Contributor \
  --scopes /subscriptions/b15d766xxf-80xxx21-486xxxx6-bb33xxx-5aad096ed0xxx

# Note the output: appId, password, tenant — needed in the extension wizard

# Step 2: Create storage account for certificate storage
az storage account create \
  --name "ahmedappstorage" \
  --resource-group app_rg01 \
  --location "Canada Central" \
  --sku Standard_LRS

# Step 3: Get the storage connection string
az storage account show-connection-string \
  --name "ahmedappstorage" \
  --resource-group "app_rg01" \
  -o tsv
# Copy this connection string — paste it into the Let's Encrypt extension setup
```





Option B — Azure Managed Certificate (Recommended — More Reliable)

If Let's Encrypt does not work correctly, remove the extension first, then use the Azure CLI to create a managed certificate directly. This is simpler and free for App Service.

Azure CLI

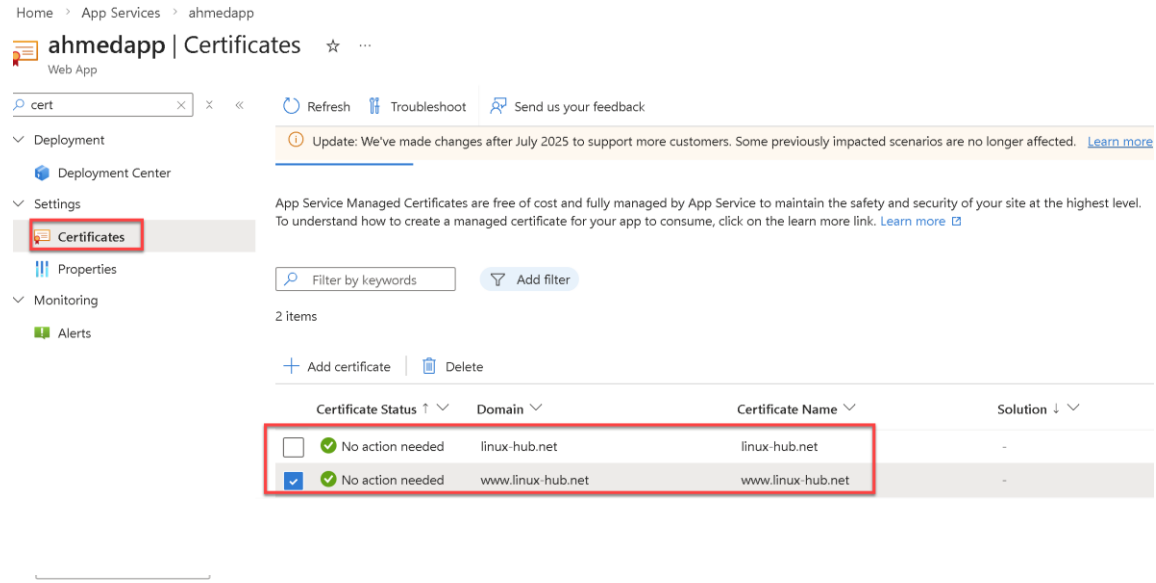
```
# Remove Let's Encrypt extension first (if installed)
# App Service → Extensions → Let's Encrypt → Delete

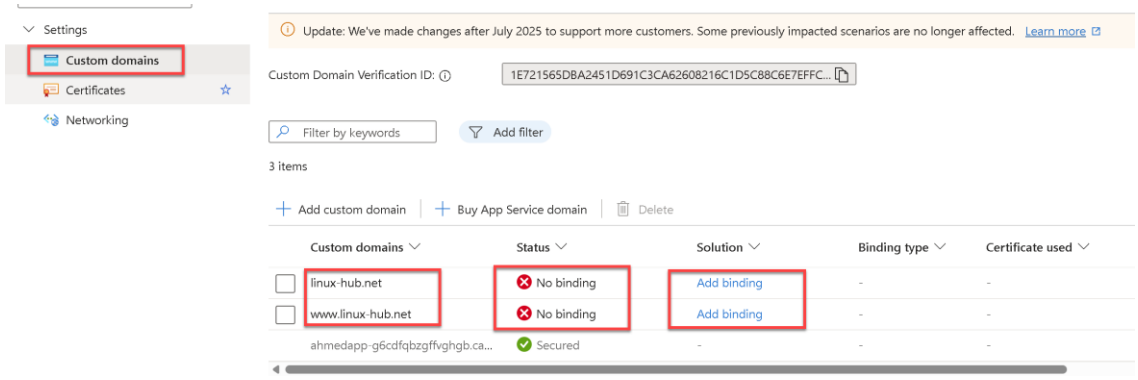
# Create Azure managed certificate for root domain
az webapp config ssl create \
  --name "ahmedapp" \
  --resource-group "app_rg01" \
  --hostname "linux-hub.net"

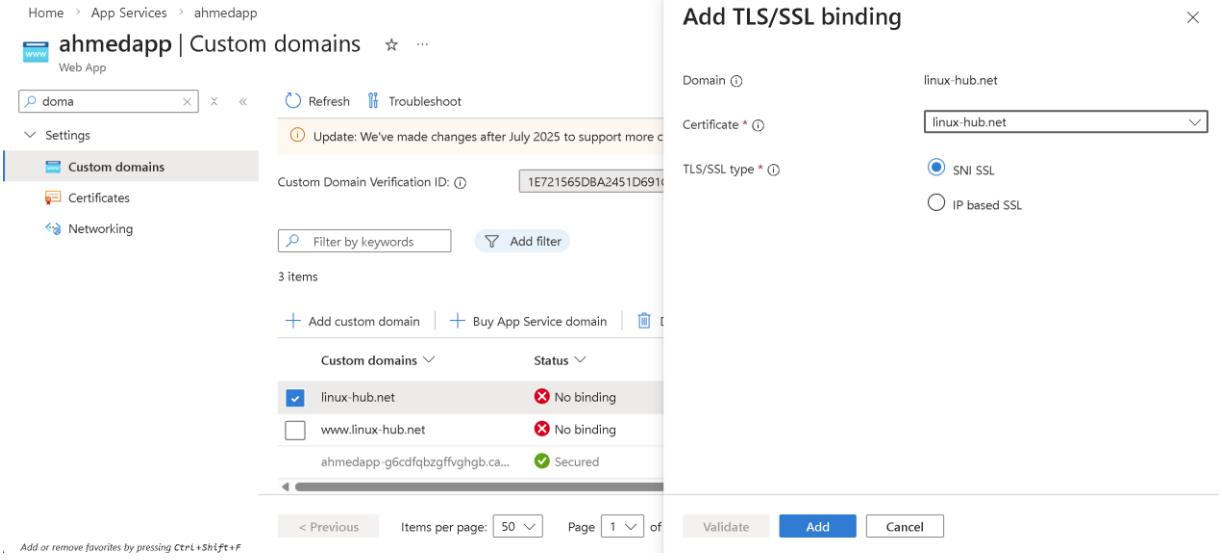
# Create managed certificate for www subdomain
az webapp config ssl create \
  --name "ahmedapp" \
  --resource-group "app_rg01" \
  --hostname "www.linux-hub.net"

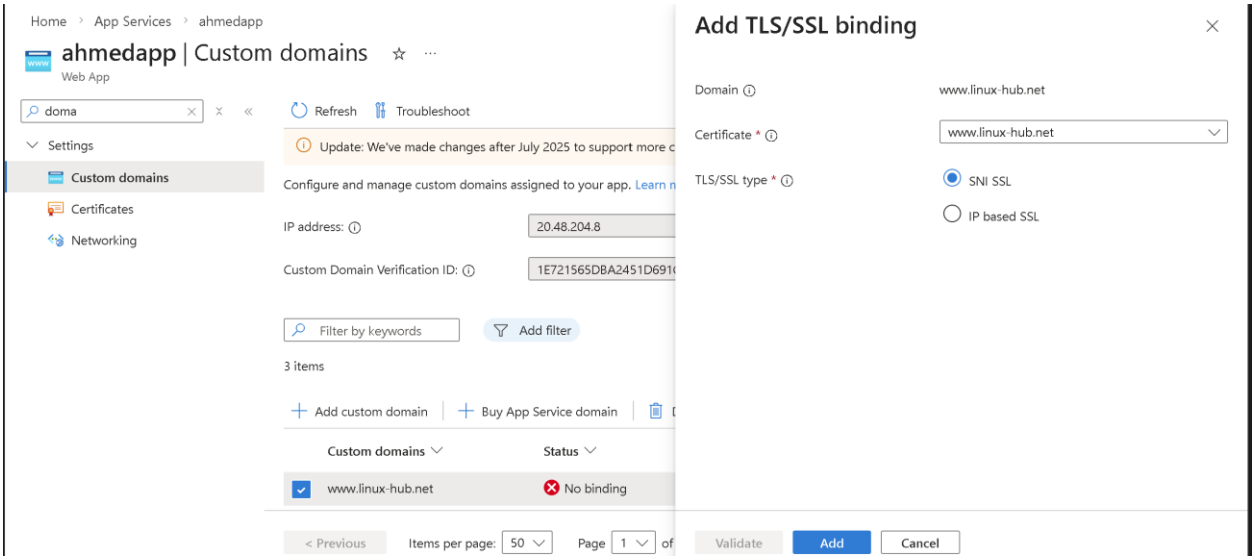
# Bind the certificate to the custom hostname
We will configure this part from the portal as shown down

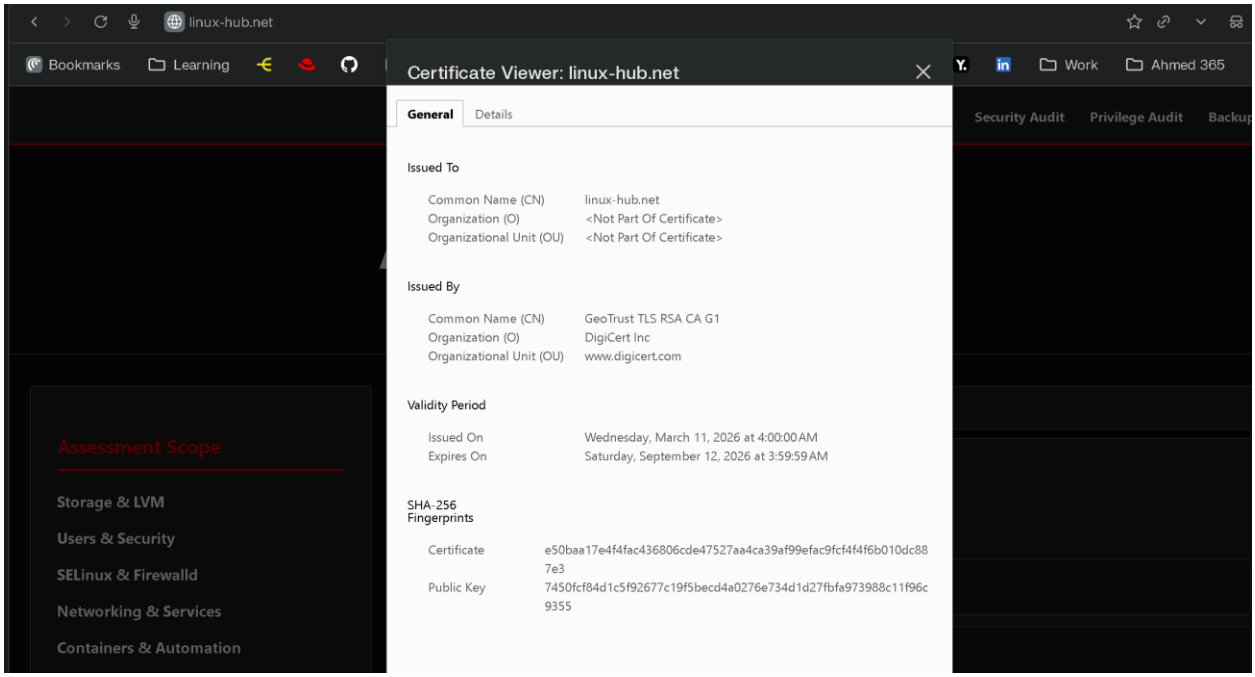
# Enforce HTTPS only
az webapp update \
  --name "ahmedapp" \
  --resource-group "app_rg01" \
  --https-only true
```


Home › App Services › ahmedapp
ahmedapp | Certificates
Web App
cert
Deployment
Deployment Center
Settings
Certificates
Properties
Monitoring
Alerts
Refresh
Troubleshoot
Send us your feedback
Update: We've made changes after July 2025 to support more customers. Some previously impacted scenarios are no longer affected. Learn more
App Service Managed Certificates are free of cost and fully managed by App Service to maintain the safety and security of your site at the highest level. To understand how to create a managed certificate for your app to consume, click on the learn more link. Learn more
Filter by keywords
Add filter
2 items
Add certificate
Delete
Certificate Status
Domain
Certificate Name
Solution
No action needed
linux-hub.net
linux-hub.net
No action needed
www.linux-hub.net
www.linux-hub.net


Settings
Custom domains
Certificates
Networking
Update: We've made changes after July 2025 to support more customers. Some previously impacted scenarios are no longer affected. Learn more
Custom Domain Verification ID:
1E721565DBA2451D691C3CA62608216C1D5C88C6E7EFFC...
Filter by keywords
Add filter
3 items
Add custom domain
Buy App Service domain
Delete
Custom domains
Status
Solution
Binding type
Certificate used
linux-hub.net
No binding
Add binding
www.linux-hub.net
No binding
Add binding
ahmedapp-g6cdfqbzgffvghgb.ca...
Secured


Home › App Services › ahmedapp
ahmedapp | Custom domains
Web App
doma
Settings
Custom domains
Certificates
Networking
Refresh
Troubleshoot
Custom Domain Verification ID:
Filter by keywords
Add filter
3 items
Add custom domain
Buy App Service domain
Custom domains
Status
linux-hub.net
No binding
www.linux-hub.net
No binding
ahmedapp-g6cdfqbzgffvghgb.ca...
Secured
< Previous
Items per page: 50
Page 1 of
Add or remove favorites by pressing Ctrl+Shift+F
Add TLS/SSL binding
Domain
linux-hub.net
Certificate *
linux-hub.net
TLS/SSL type *
SNI SSL
IP based SSL
Validate
Add
Cancel


Home › App Services › ahmedapp
ahmedapp | Custom domains
Web App
doma
Settings
Custom domains
Certificates
Networking
Refresh
Troubleshoot
Update: We've made changes after July 2025 to support more c
Configure and manage custom domains assigned to your app. Learn n
IP address:
20.48.204.8
Custom Domain Verification ID:
1E721565DBA2451D691
Filter by keywords
Add filter
3 items
Add custom domain
Buy App Service domain
Custom domains
Status
www.linux-hub.net
No binding
< Previous
Items per page: 50
Page 1 of
Add TLS/SSL binding
Domain
www.linux-hub.net
Certificate *
www.linux-hub.net
TLS/SSL type *
SNI SSL
IP based SSL
Validate
Add
Cancel


linux-hub.net
Bookmarks
Learning
Work
Ahmed 365
Certificate Viewer: linux-hub.net
General
Details
Issued To
Common Name (CN) linux-hub.net
Organization (O) <Not Part Of Certificate>
Organizational Unit (OU) <Not Part Of Certificate>
Issued By
Common Name (CN) GeoTrust TLS RSA CA G1
Organization (O) DigiCert Inc
Organizational Unit (OU) www.digicert.com
Validity Period
Issued On Wednesday, March 11, 2026 at 4:00:00 AM
Expires On Saturday, September 12, 2026 at 3:59:59 AM
SHA-256 Fingerprints
Certificate e50baa17e4f4fac436806cde47527aa4ca39af99efac9fcf4f4f6b010dc887e3
Public Key 7450fcf84d1c5f92677c19f5becd4a0276e734d1d27fbfa973988c11f96c9355
Security Audit
Privilege Audit
Assessment Scope
Storage & LVM
Users & Security
SELinux & Firewalld
Networking & Services
Containers & Automation

LAB 2 — Authentication: Basic Auth + Google OAuth

LAB 2: Protect Your Web App with Authentication

Duration: 45–60 min	Level: Intermediate	Cost: No additional cost — Easy Auth is free

Prerequisites

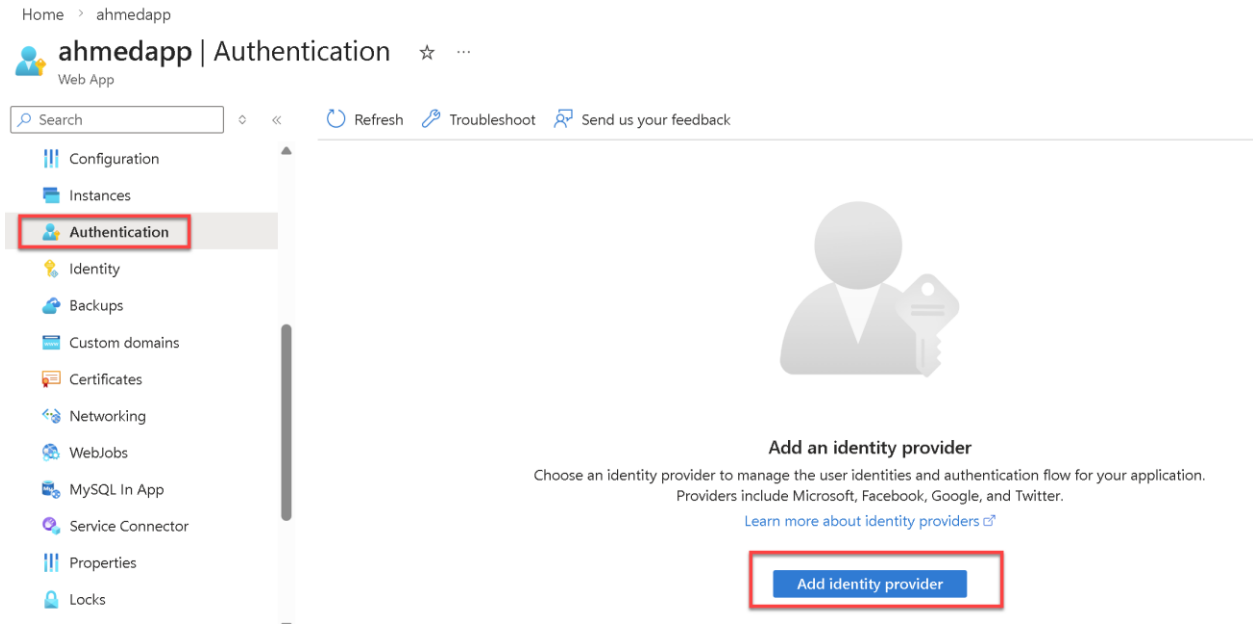
ahmedapp running at linux-hub.net with TLS enabled
Google account to access Google Cloud Console (console.cloud.google.com)
App Service Easy Auth (Azure App Service Authentication) is built-in — no extensions needed

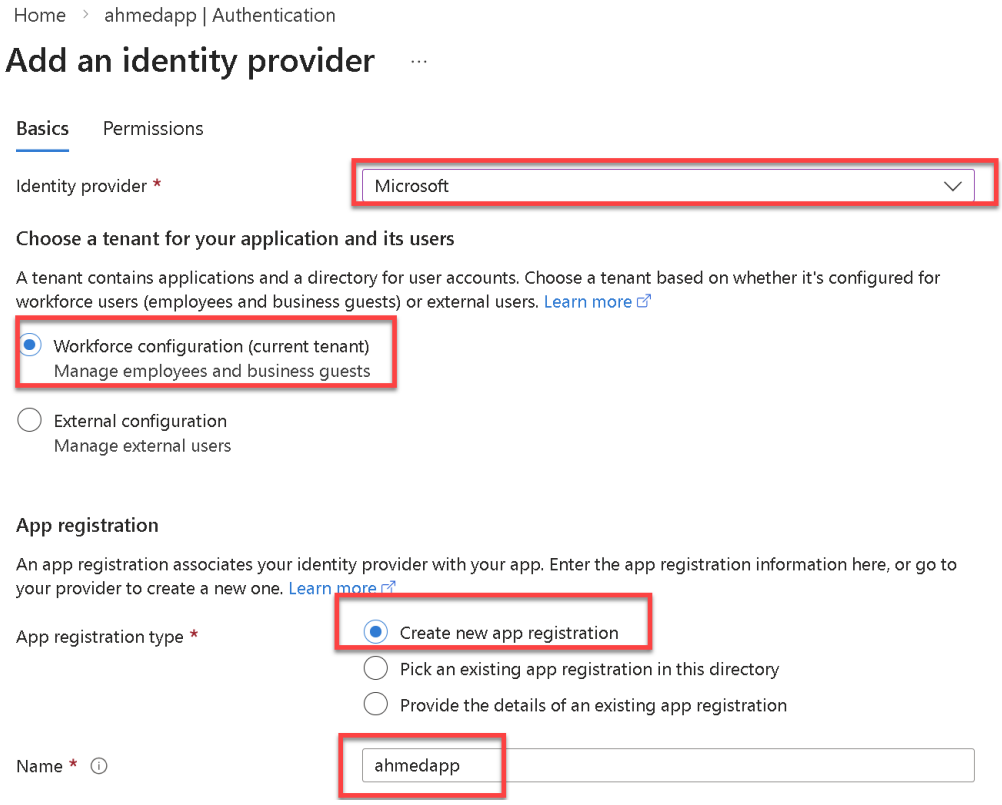
Lab 2.1 — Basic Authentication (Username & Password)

Azure App Service Easy Auth can enforce a login prompt on your web app without modifying any application code. This is useful for staging environments or internal tools. Users visiting linux-hub.net will see a login prompt.

1 Enable Authentication on the App Service

App Service → Authentication → Add identity provider → select "Microsoft" (for Entra ID / Microsoft accounts) or configure custom provider.





2 **Configure authentication settings**

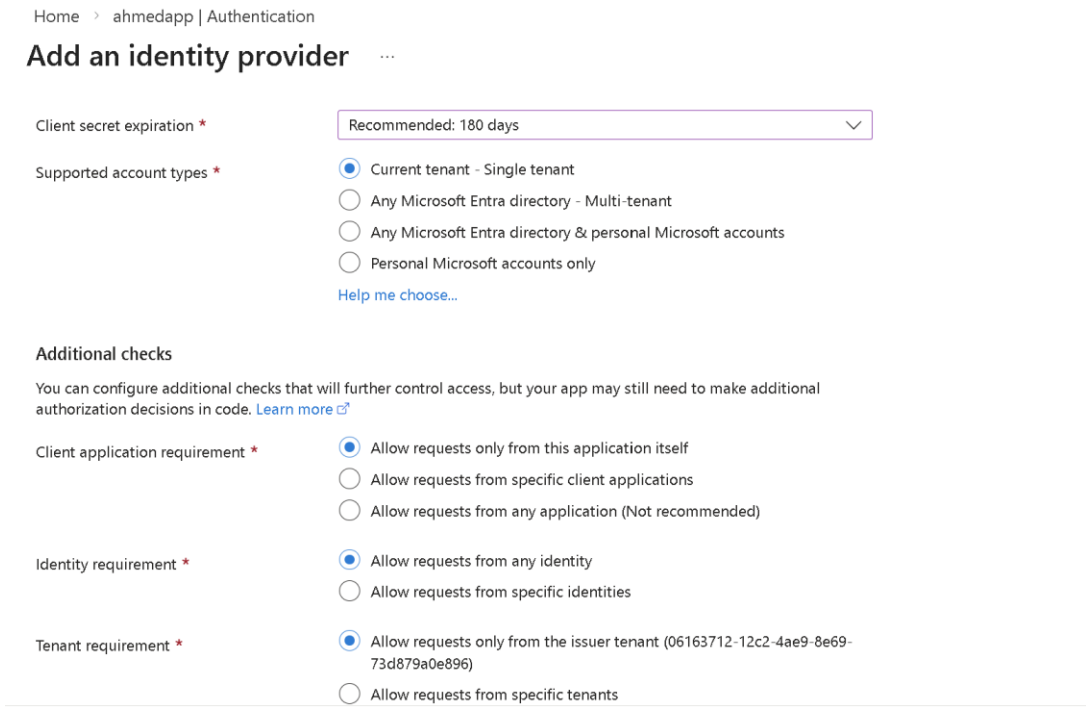
Action when request not authenticated: HTTP 401 Unauthorized (API) or HTTP 302 Redirect (web app — choose this) · Token store: Enabled · Allowed external redirect URLs: leave blank for now

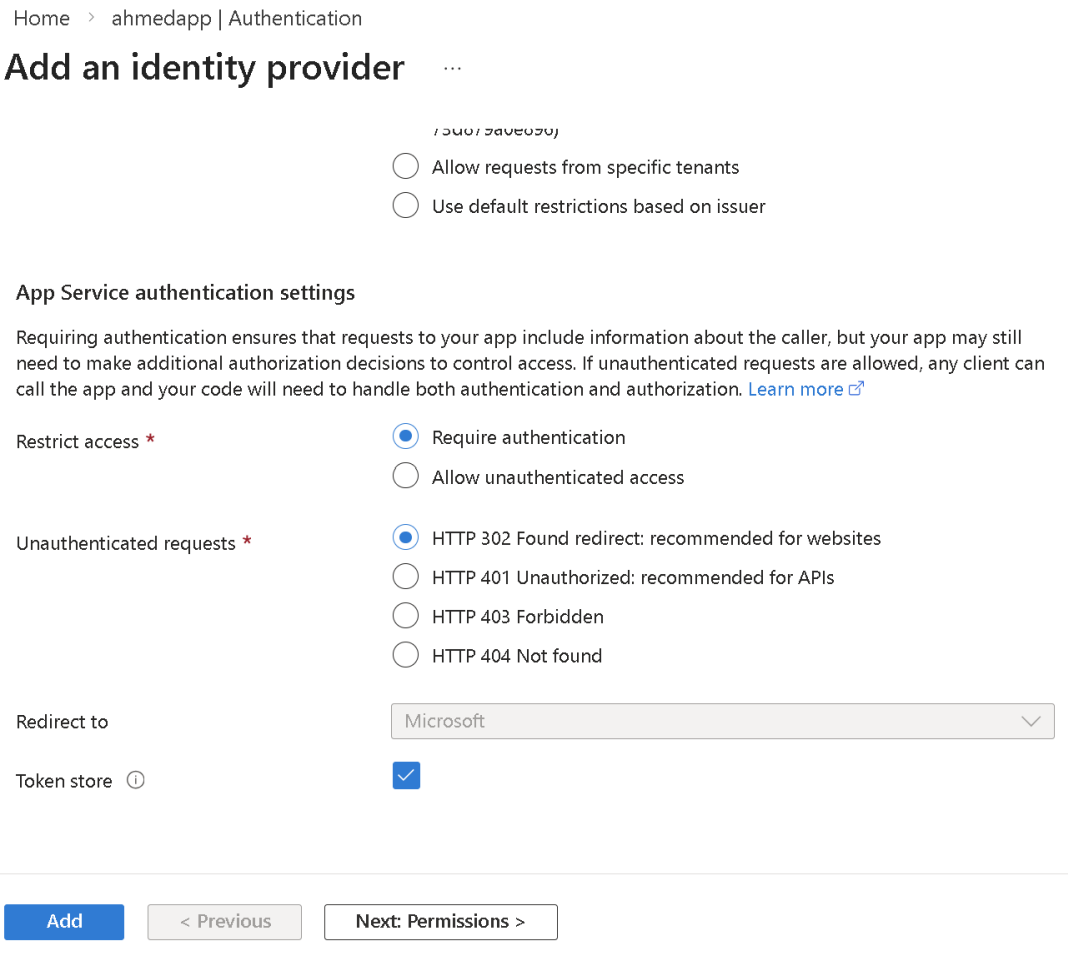
3 **Enable Basic Auth credentials**

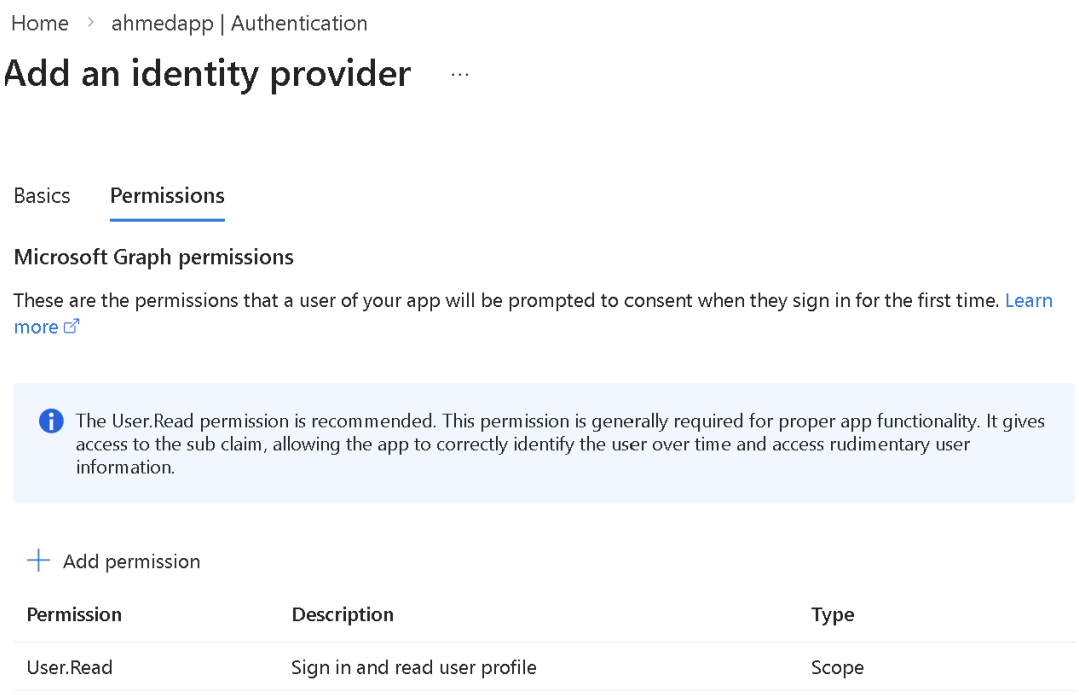
App Service → Configuration → General settings → Basic Auth Publishing Credentials → On. This enables username/password protection.

4 **Set platform authentication**

For a simple username/password gate without Entra ID: App Service → Authentication → Built-in authentication → Enable → Require authentication. Set Action to: Log in with Microsoft.


Home › ahmedapp | Authentication
Add an identity provider
Client secret expiration *
Recommended: 180 days
Supported account types *
Current tenant - Single tenant
Any Microsoft Entra directory - Multi-tenant
Any Microsoft Entra directory & personal Microsoft accounts
Personal Microsoft accounts only
Help me choose...
Additional checks
You can configure additional checks that will further control access, but your app may still need to make additional authorization decisions in code. Learn more
Client application requirement *
Allow requests only from this application itself
Allow requests from specific client applications
Allow requests from any application (Not recommended)
Identity requirement *
Allow requests from any identity
Allow requests from specific identities
Tenant requirement *
Allow requests only from the issuer tenant (06163712-12c2-4ae9-8e69-73d879a0e896)
Allow requests from specific tenants


Home › ahmedapp | Authentication
Add an identity provider
Allow requests from specific tenants
Use default restrictions based on issuer
App Service authentication settings
Requiring authentication ensures that requests to your app include information about the caller, but your app may still need to make additional authorization decisions to control access. If unauthenticated requests are allowed, any client can call the app and your code will need to handle both authentication and authorization. Learn more
Restrict access *
Require authentication
Allow unauthenticated access
Unauthenticated requests *
HTTP 302 Found redirect: recommended for websites
HTTP 401 Unauthorized: recommended for APIs
HTTP 403 Forbidden
HTTP 404 Not found
Redirect to
Microsoft
Token store
Add
< Previous
Next: Permissions >



5 **Test the login prompt**

Open an incognito/private browser window → navigate to https://linux-hub.net. You should be redirected to a login page. After logging in, the website loads normally.

Tip

If you want to allow some routes without auth (e.g. a public API endpoint), use App Service → Authentication → Edit provider → Excluded paths. Add /api/public/* to bypass the auth requirement for those paths only.

Lab 2.2 — Google OAuth (Allow Login with Gmail)

Configure Google as an identity provider so users can sign in with their Google/Gmail account. This uses the OAuth 2.0 authorization code flow through Azure Easy Auth — no code changes needed.

Step 1 — Create a Google OAuth Application

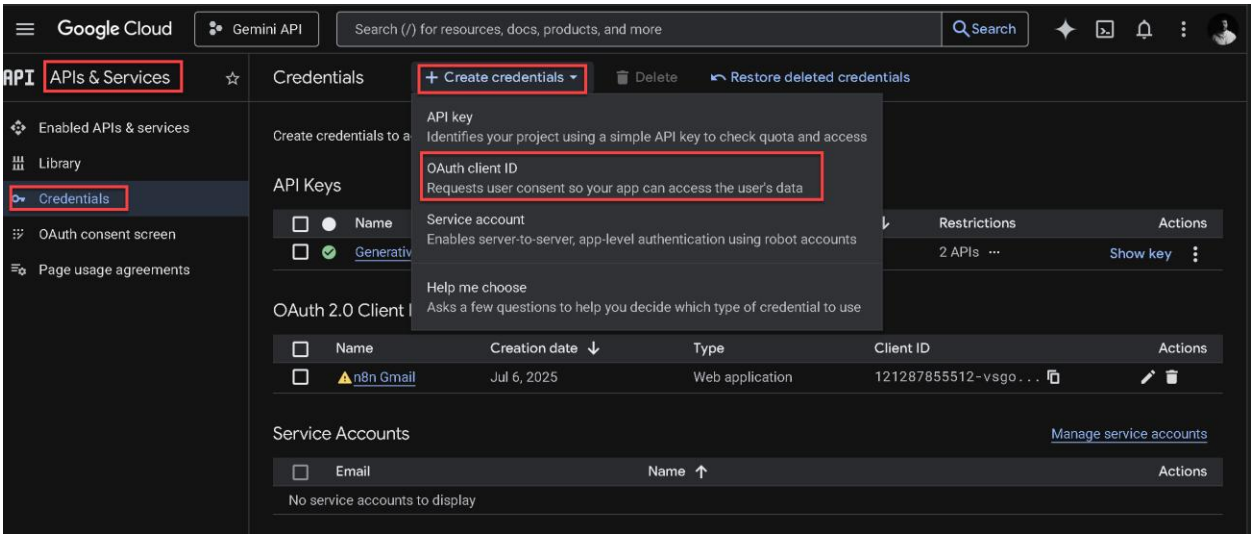
1 — Open Google Cloud Console
Navigate to https://console.cloud.google.com → select or create a project → APIs & Services → Credentials.

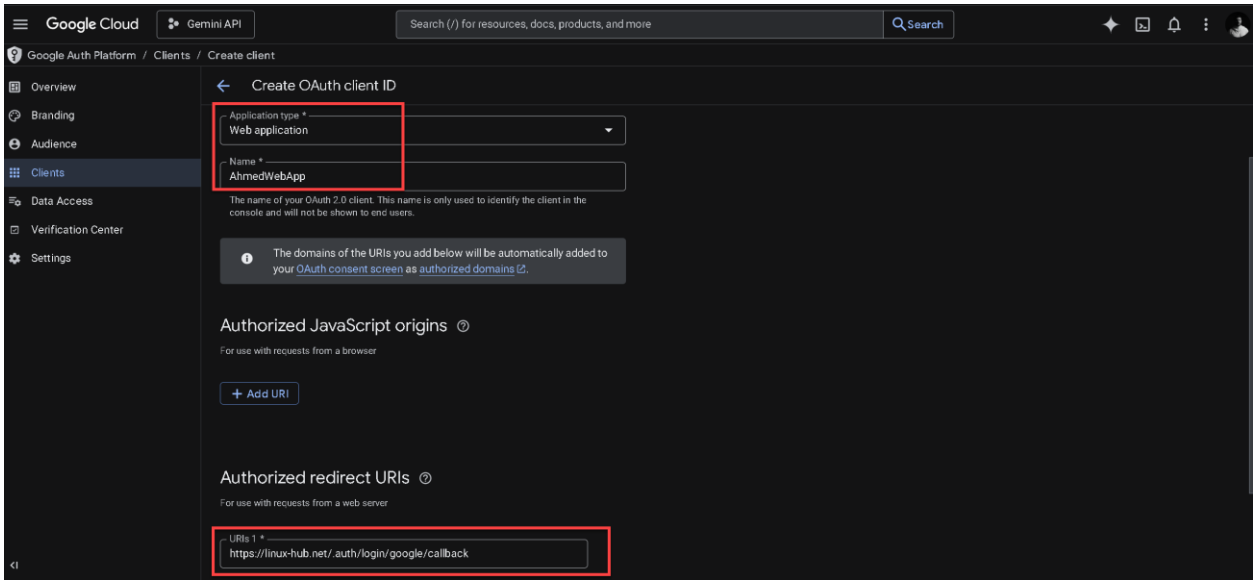
2 — Create OAuth 2.0 Client ID
Credentials → Create Credentials → OAuth client ID → Application type: Web application → Name: AhmedApp.

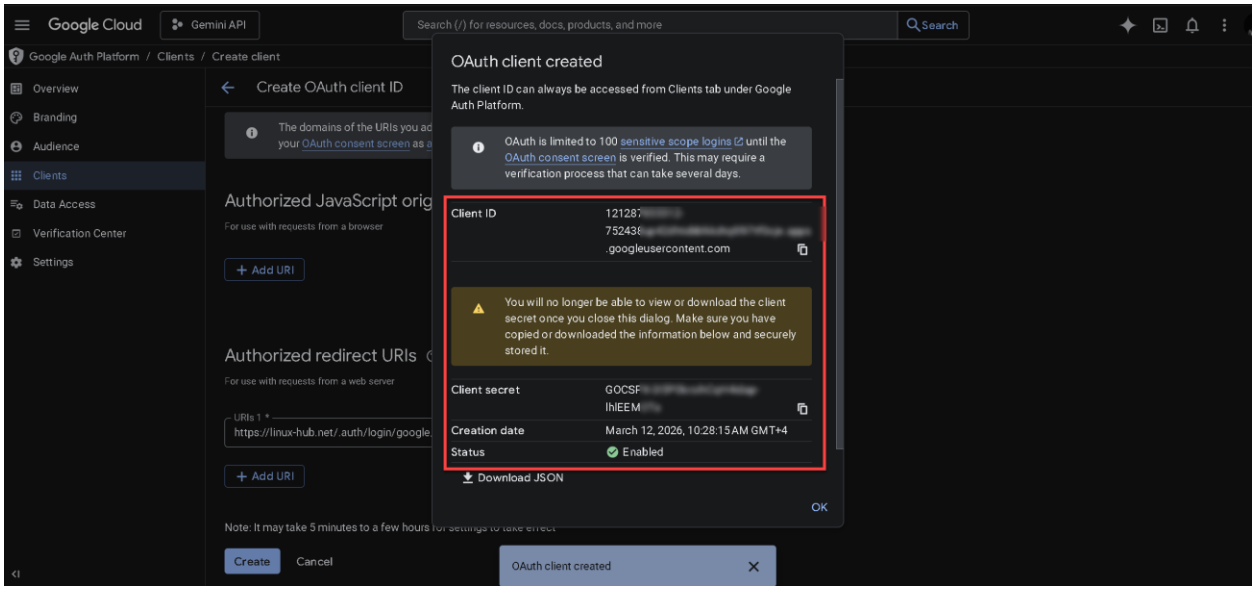
3 — Add authorised redirect URI
Authorised redirect URIs → Add URI → enter: https://ahmedapp.azurewebsites.net/.auth/login/google/callback. This is the Azure Easy Auth callback URL.

4 — Copy Client ID and Client Secret
After creating, copy the Client ID and Client Secret — you need these in Azure.

https://console.cloud.google.com

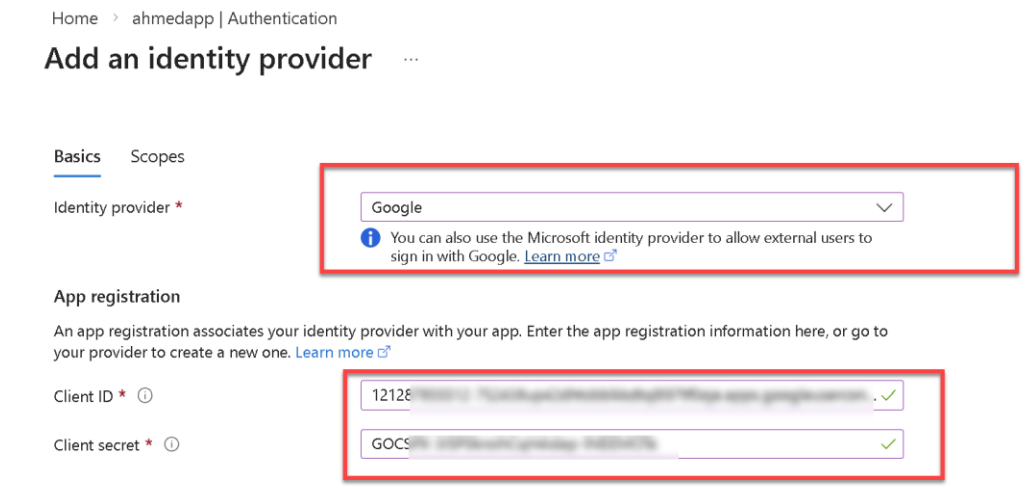


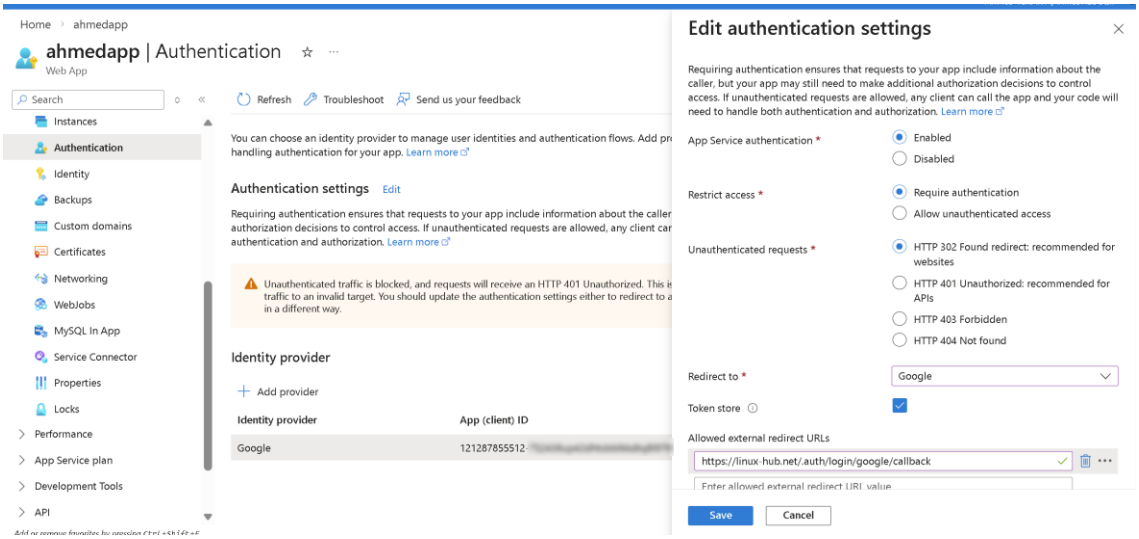




Step 2 — Configure Google Provider in Azure Easy Auth

1	**Open Easy Auth in Azure** App Service → Authentication → Add identity provider → Google.
2	**Enter Google credentials** Client ID: paste from Google Cloud Console · Client Secret: paste from Google Cloud Console · Allowed token audiences: leave default · Click Add
3	**Configure OAuth scopes** By default, Easy Auth requests the openid, profile, and email scopes. To control what data your app requests, edit the provider → Scopes field. Remove any scope your app does not need.

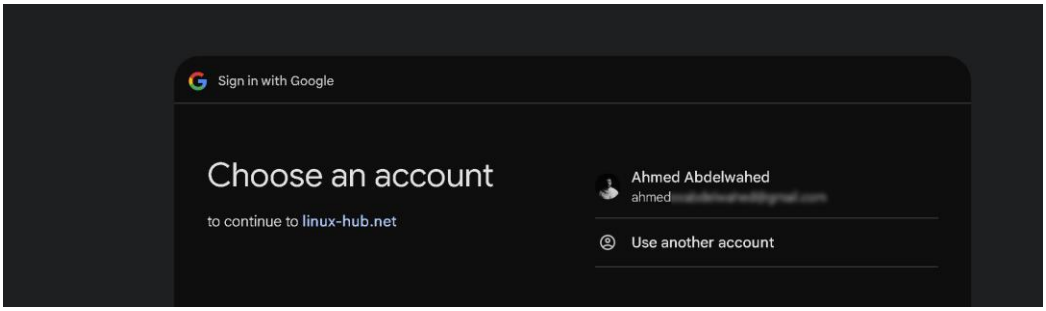


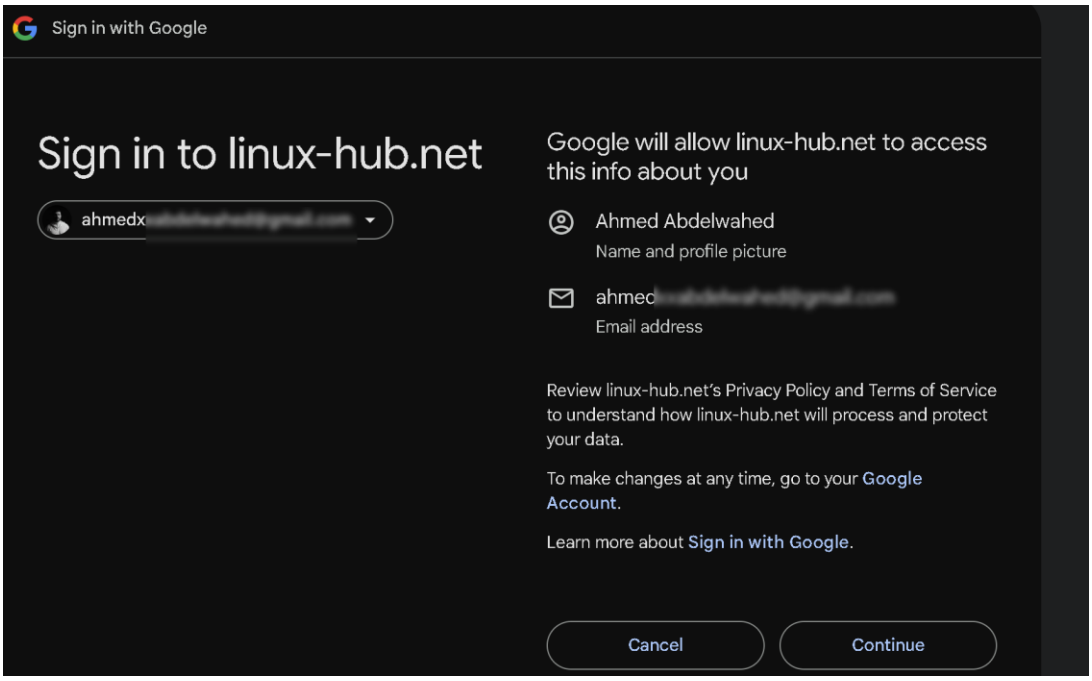


Step 3 — Test Google Login

1 **Open incognito browser**
Navigate to https://linux-hub.net in a private window.

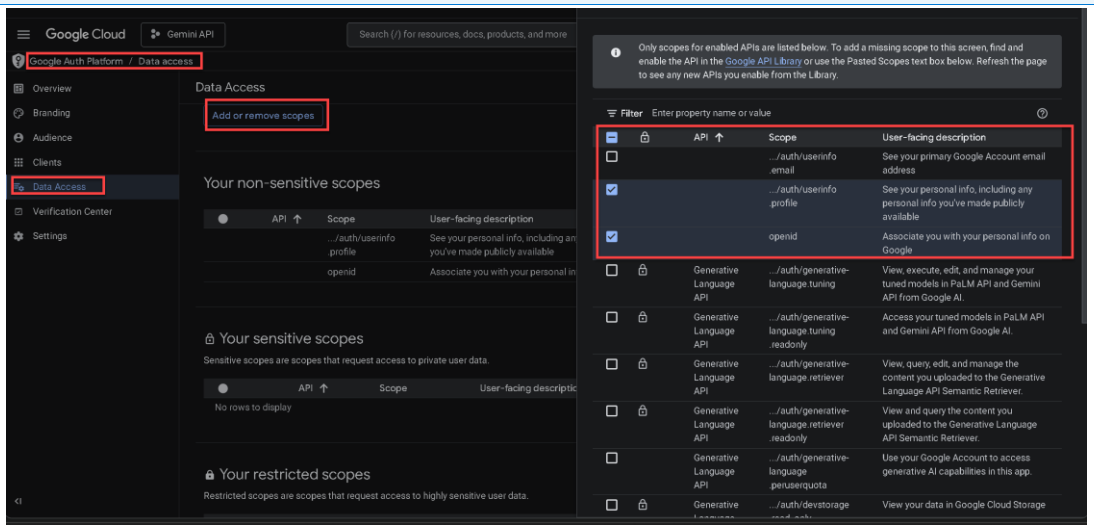
2 **Verify Google prompt**
You should see a "Sign in with Google" option on the login page. Click it.





OAuth Scope	Data Provided	Required?
openid	Authenticates the user (issues ID token)	Yes — always required
email	User's email address	Recommended
profile	Name, profile photo, locale	Optional
calendar	Access to Google Calendar events	Only if app needs it
drive	Access to Google Drive files	Only if app needs it

> **Note**
> Request only the scopes your application genuinely needs. Users see a consent screen listing requested permissions — fewer scopes increases user trust and Google approval rate for OAuth apps.



LAB 3 — Advanced Tools: Kudu, App Editor & FTP Management

LAB 3: App Service Advanced Tools and Content Management

Duration: 30–45 min	Level: Intermediate	Cost: No additional cost

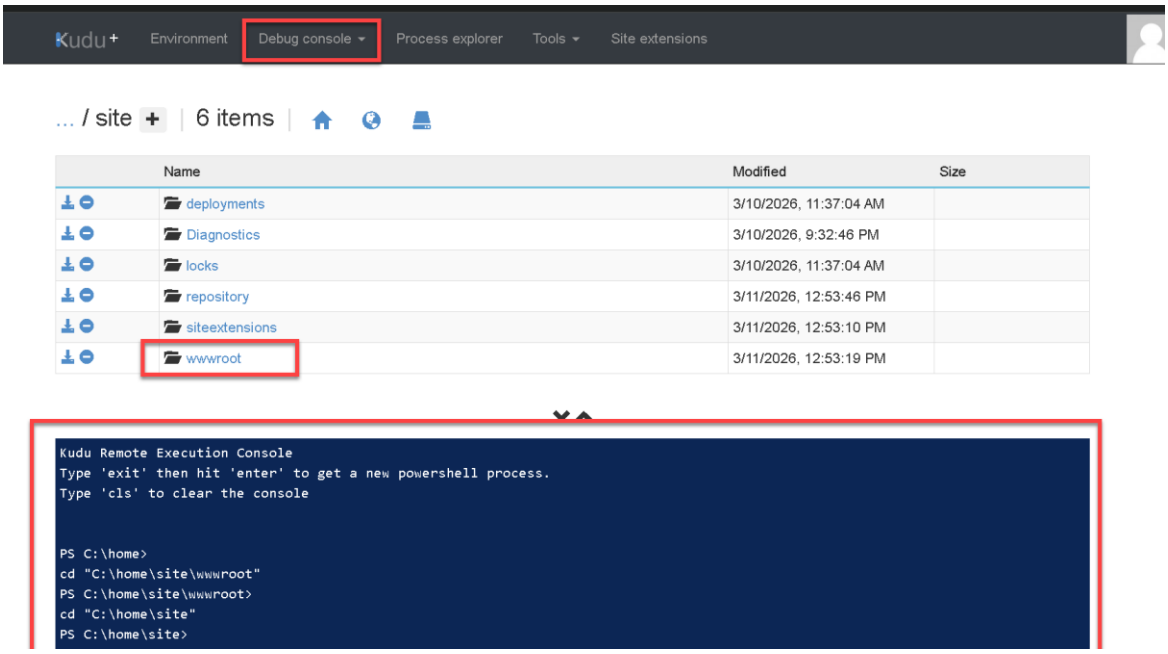
Prerequisites
ahmedapp running. FTP credentials configured.

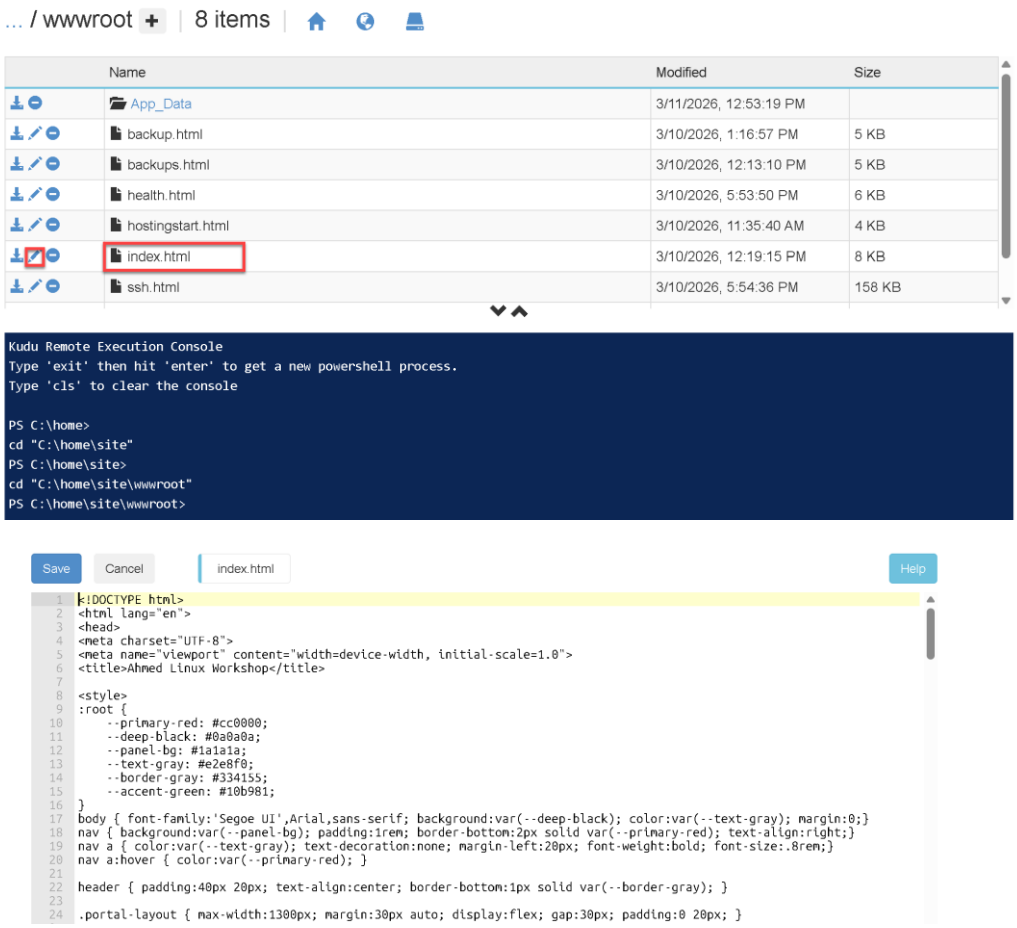
Lab 3.1 — Kudu Advanced Tools Console

Kudu is the deployment engine behind Azure App Service. It provides a web-based console, file explorer, process explorer, and REST API for managing the app without SSH or RDP.

1 Open Kudu
App Service → Advanced Tools → Go (opens in new tab). URL format: https://ahmedapp.scm.azurewebsites.net

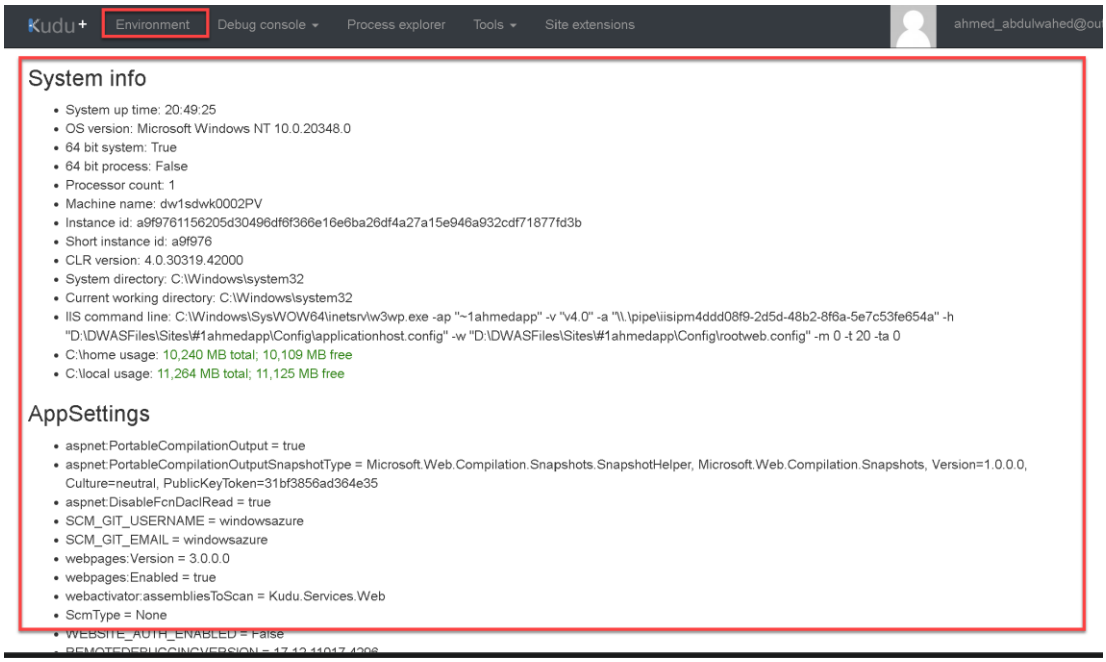
2 Explore the Kudu console
Kudu → Debug Console → CMD (Windows) or Bash (Linux). Navigate to D:\home\site\wwwroot or /home/site/wwwroot to see deployed files.



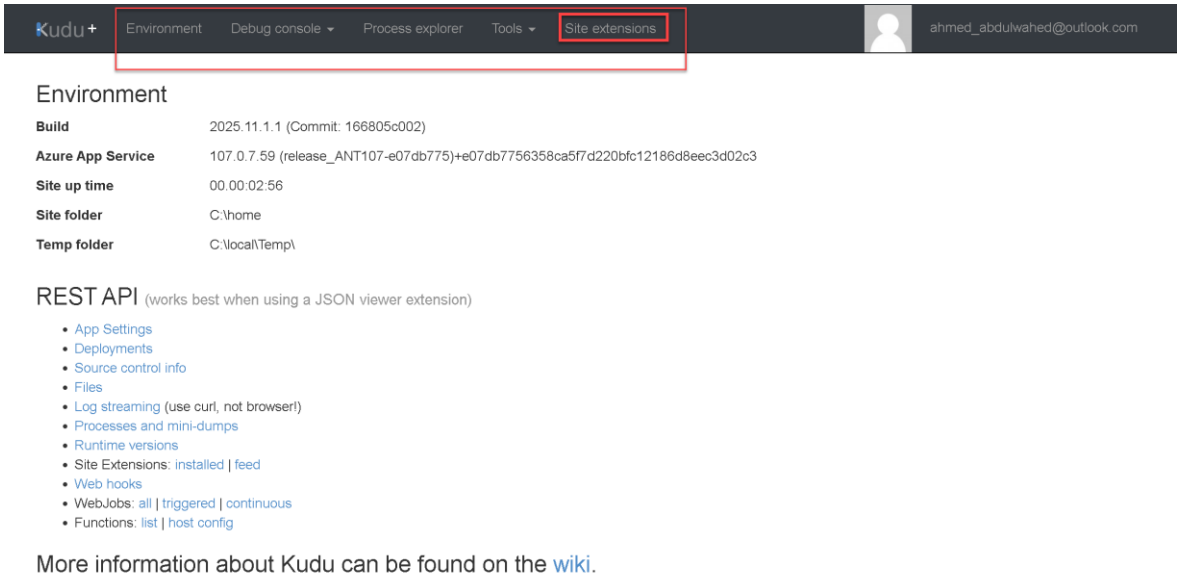


3 View running processes

Kudu → Process Explorer: see all running processes, memory usage, and CPU for your web app — useful for diagnosing high CPU or memory issues.



4 **Check environment variables**
Kudu → Environment: view all app settings, connection strings, and system environment variables as the app sees them.



Lab 3.2 — App Service Editor (Monaco)

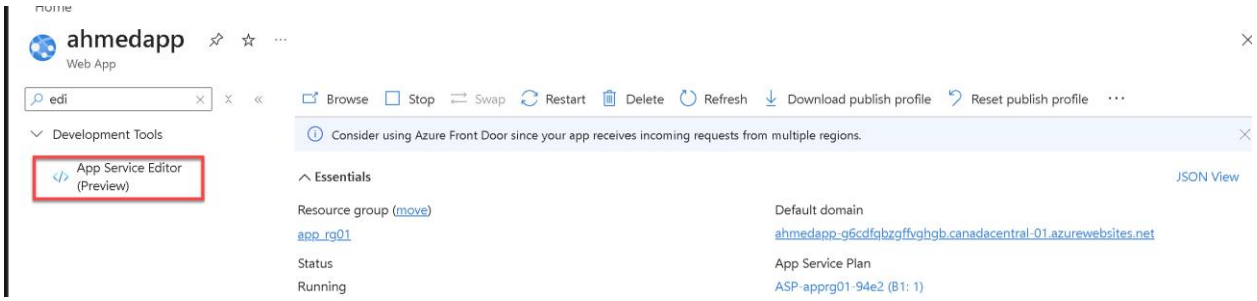
The App Service Editor is a web-based VS Code-like editor built into Kudu. It is best for quick edits to configuration files or templates without deploying a full release. It does NOT auto-save to production on every keystroke — you must explicitly save.

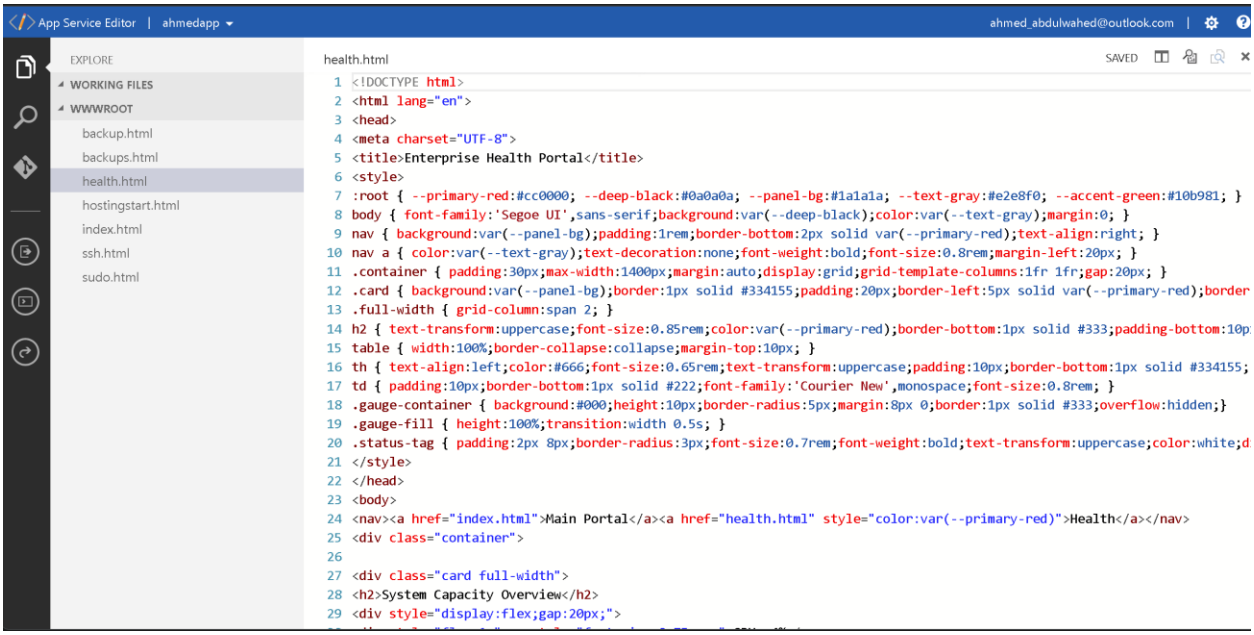
1 **Open App Service Editor**
App Service → App Service Editor (Preview) → Go. The Monaco editor opens with your wwwroot file tree on the left.

2 **Edit a file**
Click any file (e.g. index.html) in the left panel → edit in the editor → Ctrl+S to save.

3 **Verify the change**
Refresh your website in the browser — the change should be live immediately (no redeploy needed).

> ⚠ **Warning**
> Do not use the App Service Editor to manage production deployments. Use CI/CD (GitHub Actions, Azure DevOps) for production code changes. The editor is useful for config tweaks, environment-specific overrides, and diagnostic edits only.





Lab 3.3 — FTP Content Management Reference

Use FileZilla or any FTPS client to manage files in /site/wwwroot. This is useful when CI/CD is not configured or for bulk file uploads during initial deployment.

FTP Path	Contents	When to Edit
/site/wwwroot	Your web application files (HTML, JS, CSS, PHP, etc.)	Main content deployment
/site/deployments	Deployment history and logs	Debugging failed deployments
/LogFiles	Application and web server logs (stdout, stderr)	Diagnosing application errors
/LogFiles/http/RawLogs	IIS/HTTP access logs	Traffic analysis
/data/SitePackages	ZIP deployment packages	Run-from-package deployments

Azure CLI

```
# Get FTP credentials via CLI
az webapp deployment list-publishing-credentials \
  --name ahmedapp \
  --resource-group app_rg01 \
  --query "{FTPUrl:scmUri,User:publishingUserName}" -o json

# Reset FTP password
az webapp deployment user set \
  --user-name ahmedftpuser \
  --password "YourSecureP@ss1"

# Stream live logs from App Service
az webapp log tail \
  --name ahmedapp \
  --resource-group app_rg01

# Download logs for offline analysis
az webapp log download \
  --name ahmedapp \
  --resource-group app_rg01 \
  --log-file /tmp/ahmedapp-logs.zip
```

Quick Reference — App Service Key Settings

Feature	Portal Path	CLI Command	Notes
FTP credentials	Deployment Center → FTPS creds	az webapp deployment list-publishing-credentials	Reset if compromised
Custom domain	Custom domains → Add	az webapp config hostname add	DNS must point to app first
TLS certificate	TLS/SSL settings → Private Key Certs	az webapp config ssl create	Azure Managed = free
Authentication	Authentication → Add identity provider	az webapp auth update	Easy Auth — no code change
App Editor	App Service Editor → Go	N/A — portal only	Not for production deploys
Kudu console	Advanced Tools → Go	https://[app].scm.azurewebsites.net	Process, env, file explorer
Scale up/down	Scale up (App Service plan)	az appservice plan update --sku	Change tier without redeploy
Always On	Configuration → General → Always On	az webapp config set --always-on true	Prevents cold starts on Basic+
HTTPS only	TLS/SSL → HTTPS Only	az webapp update --https-only true	Redirects all HTTP to HTTPS
Log streaming	Log stream (sidebar)	az webapp log tail	Real-time stdout/stderr

LAB 4 — Instances, Backups, App Service Plan & Scaling

LAB 4: Monitor Instances, Configure Backups & Scale Your App

⏱ Duration: 45–60 min	Level: Intermediate → Advanced	💰 Cost: ~$0.18 SAR/hr Basic B1. Backup storage extra (~ahmedappstorage).

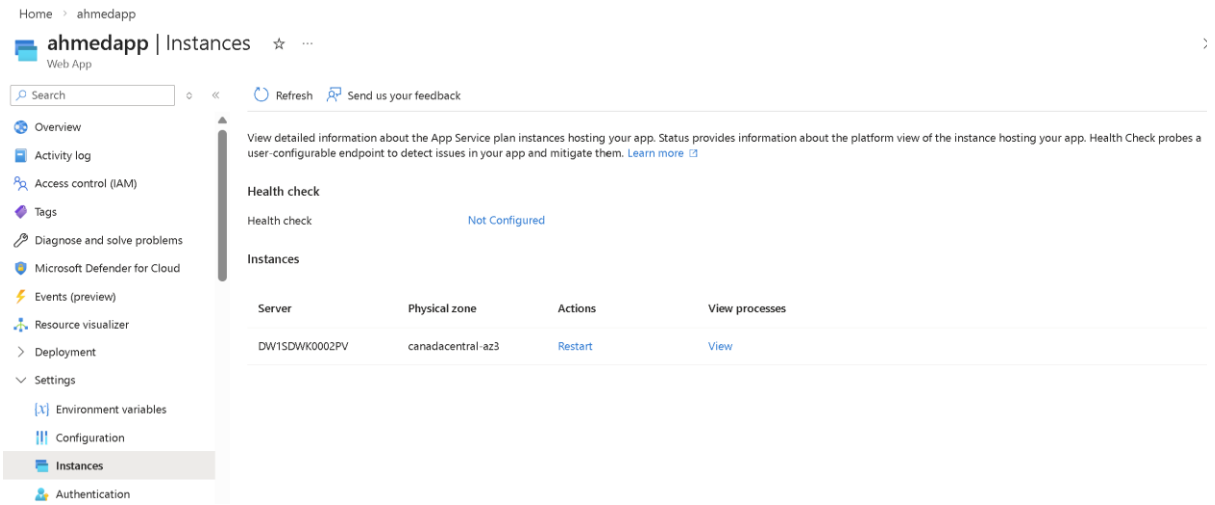
Lab 4.1 — Instances & Health Check

The Instances blade shows the physical compute instances hosting your web app. Each instance belongs to a physical availability zone in the region. Health Check actively probes a user-defined endpoint to detect unhealthy instances and automatically remove them from the load balancer.

1 **Open Instances blade**
App Service → Settings → Instances. You will see the server name, physical zone, and per-instance actions.

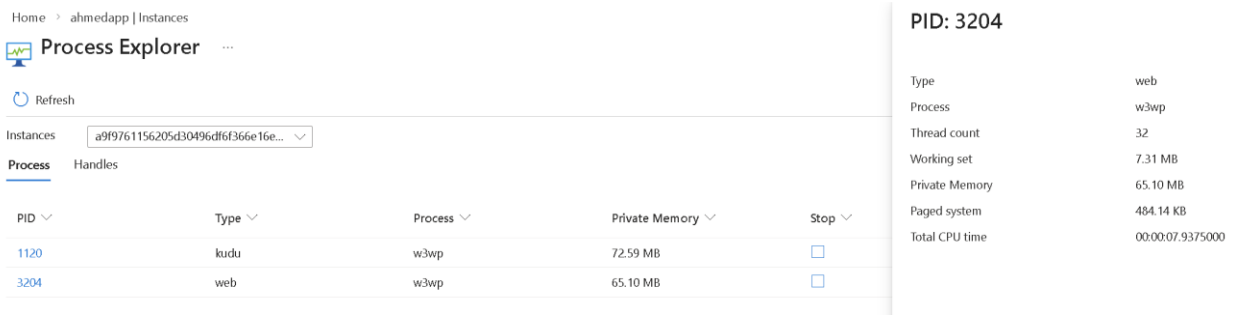
2 **Review your instance**
Your lab instance (e.g. DW1SDWK0002PV) is running in canadacentral-az3. Basic B1 supports up to 3 instances but runs 1 by default.

3 **Restart an instance**
Click Restart next to your instance to force a fresh process restart without redeploying the app. Useful for clearing memory leaks or hung worker processes.



4 **View running processes**

Click View under "View processes" — this opens the Kudu Process Explorer showing CPU and memory per worker process in real time.



Configure a Health Check Endpoint

Health Check requires your application to expose a lightweight HTTP endpoint (e.g. /health) that returns HTTP 200 when the app is healthy. Azure pings it every minute — if it fails consecutively, the instance is removed from load balancing.

Azure CLI

```
# Configure Health Check path on the App Service
az webapp config set \
  --name ahmedapp \
  --resource-group app_rg01 \
  --generic-configurations '{"healthCheckPath":"/health"}'

# Verify Health Check is configured
az webapp show \
  --name ahmedapp \
  --resource-group app_rg01 \
  --query "siteConfig.healthCheckPath" -o tsv
# Expected: /health

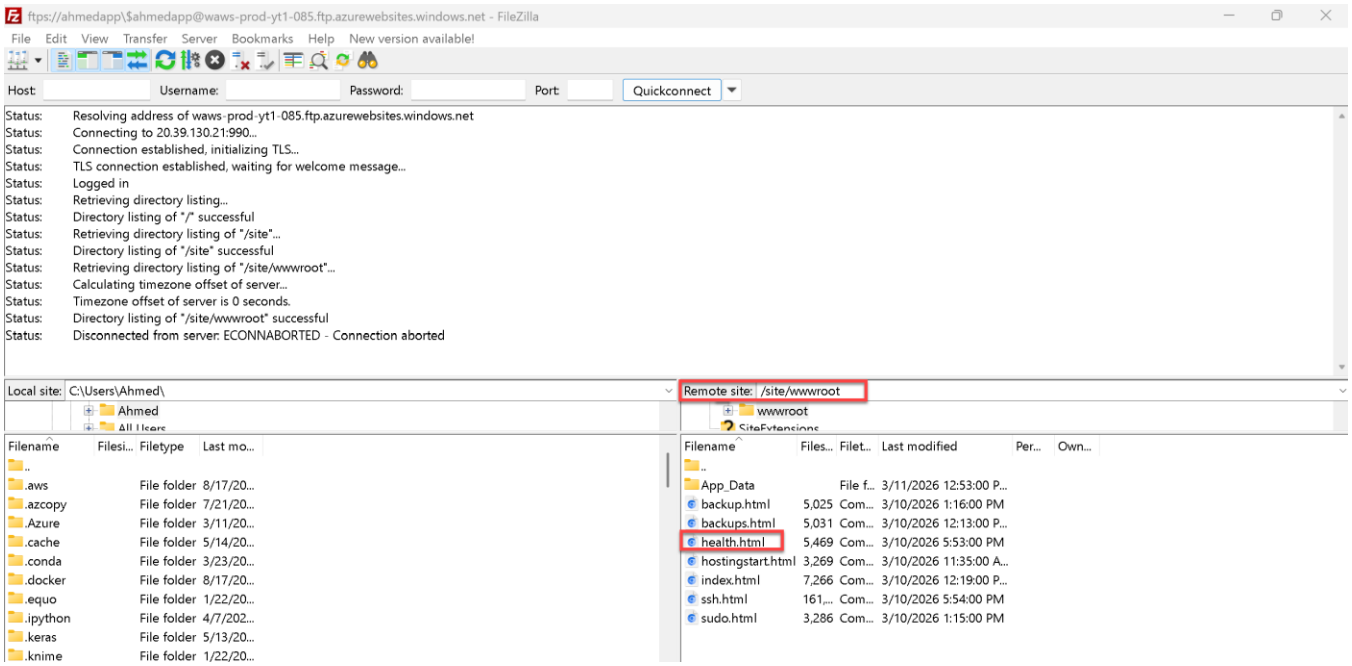
# Your app must return HTTP 200 at /health
# Example: add this route in your app code
# GET /health  →  200 OK  →  {"status":"healthy"}
```

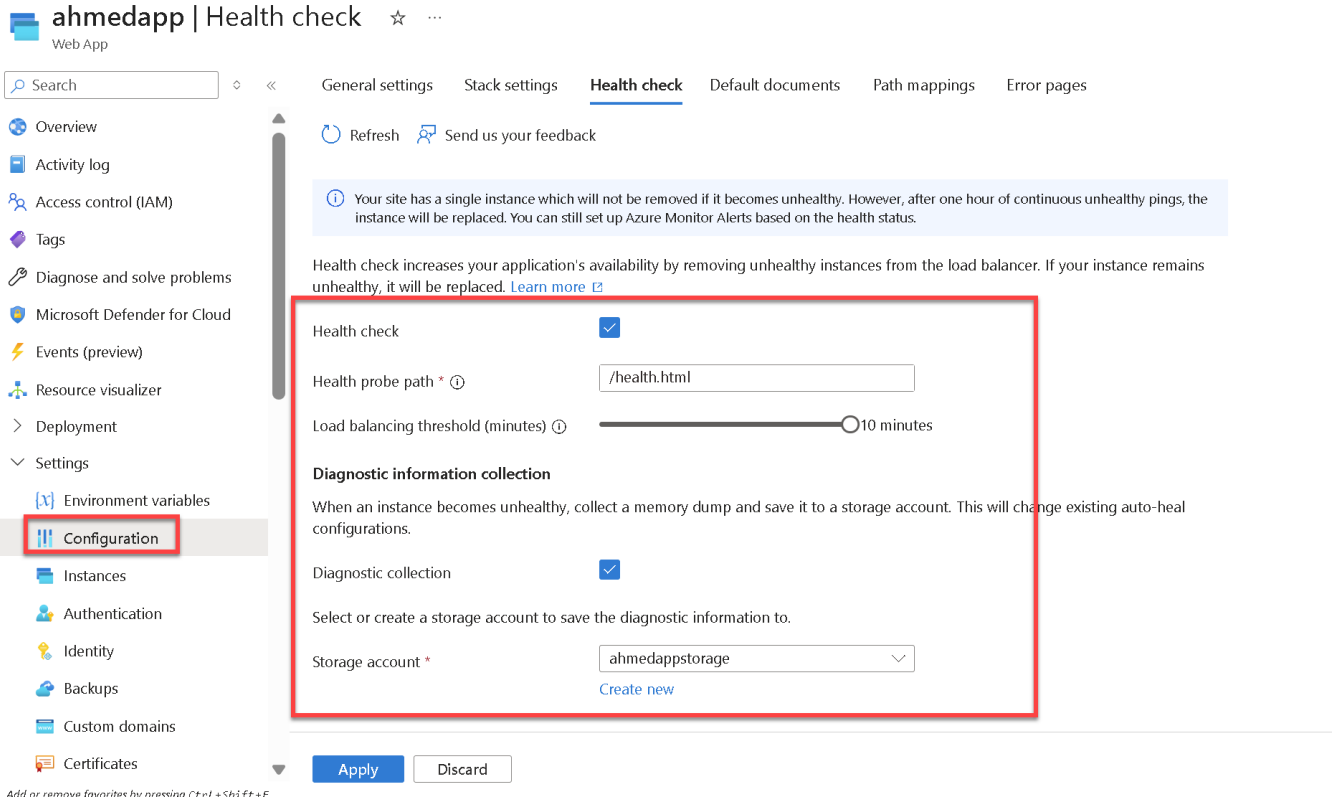
Note

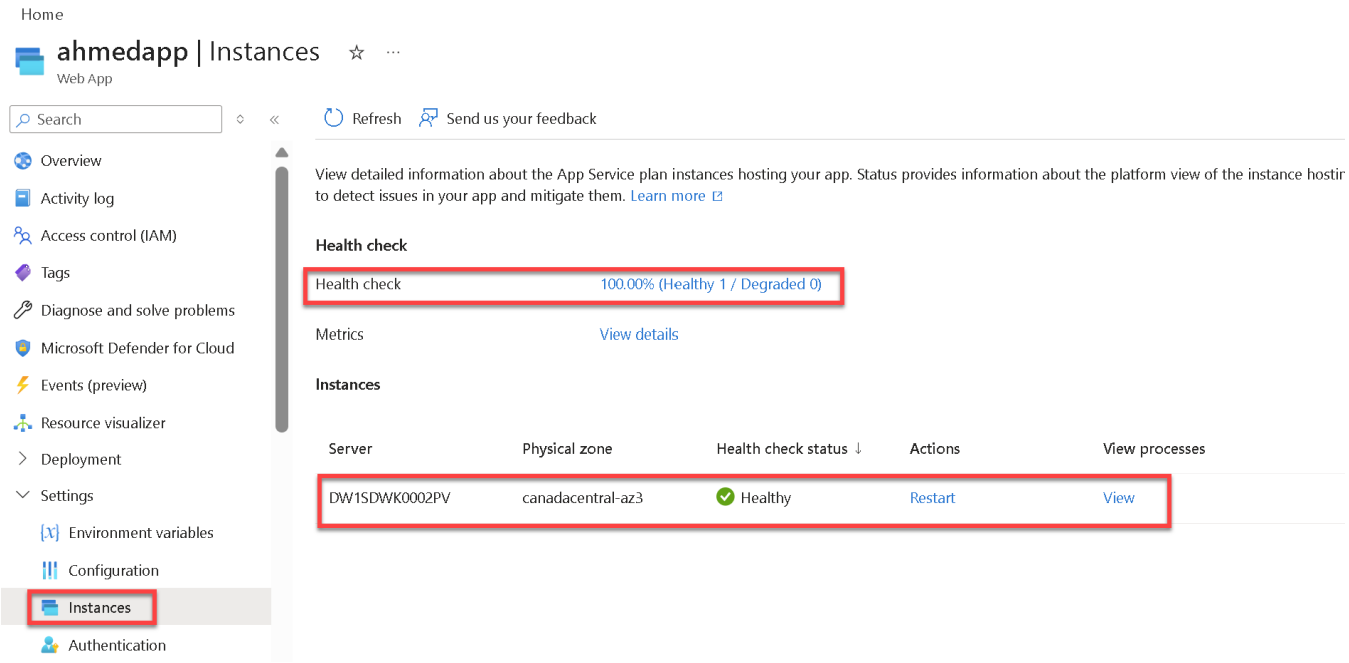
Health Check is available on Basic B1 and above. On Free and Shared tiers it is not available. For the Basic tier, it monitors but does not auto-replace unhealthy instances — that requires Standard tier or above with multiple instances.

Tier	Health Check?	Auto-replace unhealthy instance?	Max instances
Free / Shared	No	No	N/A
Basic B1–B3	Yes	No — alerts only	3
Standard S1+	Yes	Yes — auto-removes and replaces	10+
Premium v3	Yes	Yes + zone redundancy	30

Add empty health.html in /wwwroot or if it was there you can keep it, Azure will now ping https://linux-hub.net/health.html every minute







Lab 4.2 — Configure Automatic Backups

App Service Backups capture your web app files, configuration, and connected databases on a schedule. Backups are stored in an Azure Storage Account container. Your lab already shows 49 automatic backups running every hour since 3/10/2026.

1 **Open Backups blade**
App Service → Settings → Backups. You will see the backup history, oldest backup date, and schedule.

2 **Click Configure custom backups**
The Configure custom backups panel opens on the right. Select your storage account and container.

3 **Select storage**
Subscription: MSDN079 · Storage account: ahmedappstorage · Container: azure-jobs-host-output (or create a dedicated container named "appbackups").

4 **Set schedule**
Enable "Set schedule" → Repeats every: 1 Day (or 1 Hour for critical apps) → set start time → click Configure.

5 **Trigger a manual backup**
Click Backup Now to test immediately. Status should show Succeeded within 2–3 minutes.

Azure CLI

```
# Configure backup via CLI
STORAGE_KEY=$(az storage account keys list \
  --account-name ahmedappstorage \
  --resource-group app_rg01 \
  --query "[0].value" -o tsv)

# Create a dedicated container for backups
az storage container create \
  --name appbackups \
  --account-name ahmedappstorage \
  --account-key $STORAGE_KEY

# Configure backup schedule (every 24 hours, retain 10 backups)
az webapp config backup create \
  --resource-group app_rg01 \
  --webapp-name ahmedapp \
  --container-url "https://ahmedappstorage.blob.core.windows.net/appbackups?$(az storage
container generate-sas --name appbackups --account-name ahmedappstorage --account-key
$STORAGE_KEY --permissions rwd --expiry 2027-01-01 -o tsv)" \
  --frequency 24 \
  --retain-one true \
  --retention 10

# Trigger manual backup now
az webapp config backup create \
  --resource-group app_rg01 \
  --webapp-name ahmedapp \
  --backup-name "manual-backup-$(date +%Y%m%d)"

# List all backups
az webapp config backup list \
  --resource-group app_rg01 \
  --webapp-name ahmedapp \
  --query "[].{Name:name,Status:backupStatus,Created:created}" -o table
```

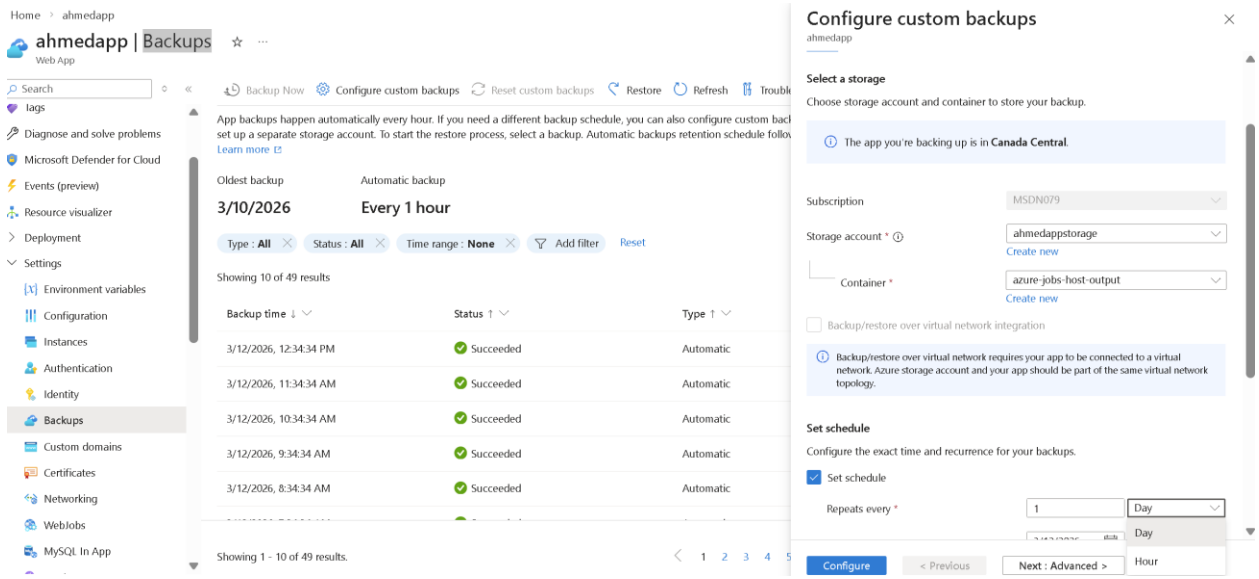
> ⚠ **Warning**
> Automatic backups are NOT available on Free or Shared plans. The storage account and the App Service must be in the same region (both Canada Central in this lab). Backup over VNet requires the app to be VNet-integrated.

Restore from Backup

Azure CLI

```
# List backups to find the backup ID to restore
az webapp config backup list \
  --resource-group app_rg01 \
  --webapp-name ahmedapp -o table

# Restore from a specific backup (overwrites current app)
az webapp config backup restore \
  --resource-group app_rg01 \
  --webapp-name ahmedapp \
  --backup-name "manual-backup-20260312" \
  --container-url "https://ahmedappstorage.blob.core.windows.net/appbackups?<SAS>" \
  --overwrite
```



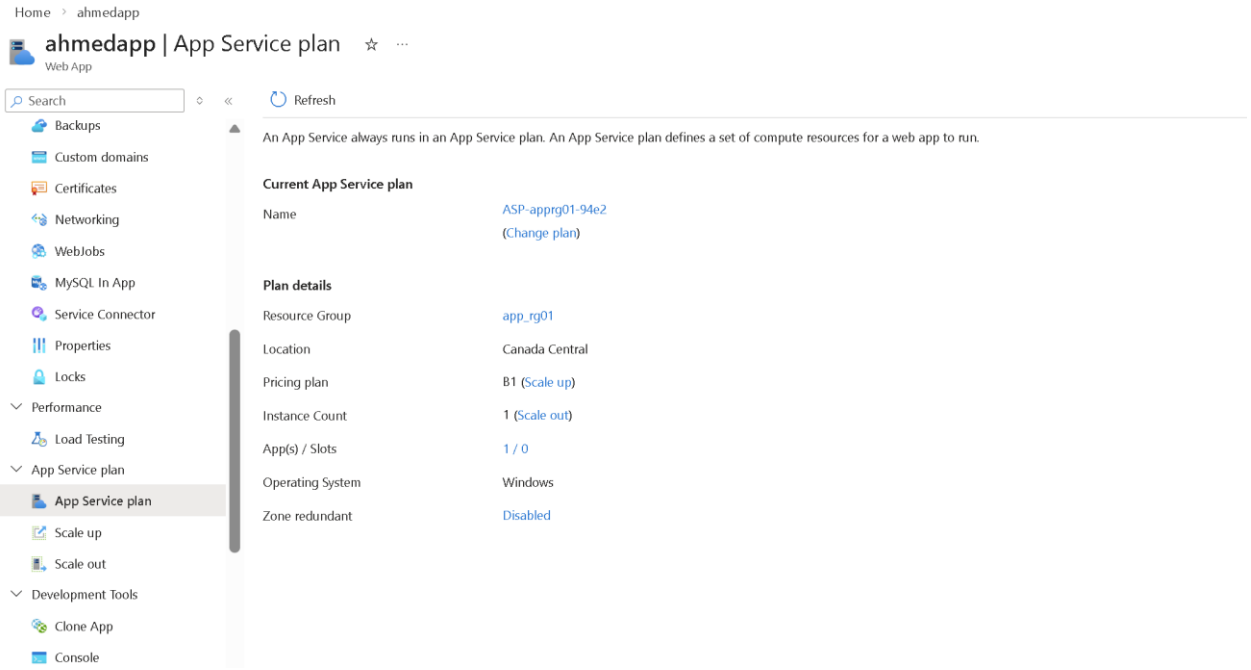
Lab 4.3 — App Service Plan Overview

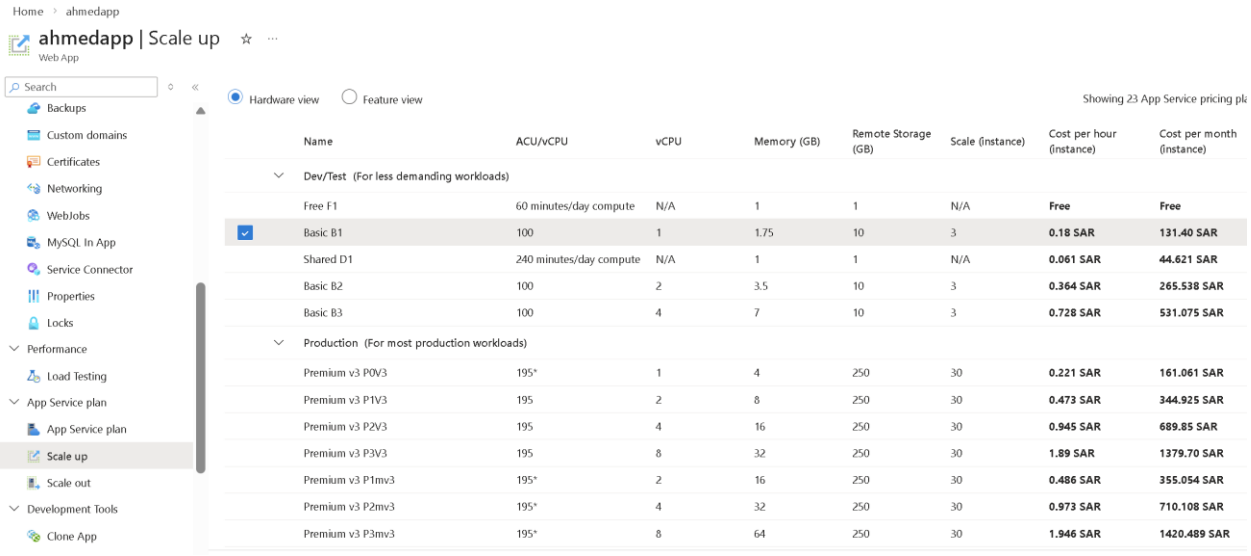
The App Service Plan is the compute foundation for your web app. It defines the region, OS, pricing tier, and number of instances. Multiple apps can share one plan — your lab shows 1 app / 0 deployment slots on ASP-apprg01 running Basic B1 in Canada Central.

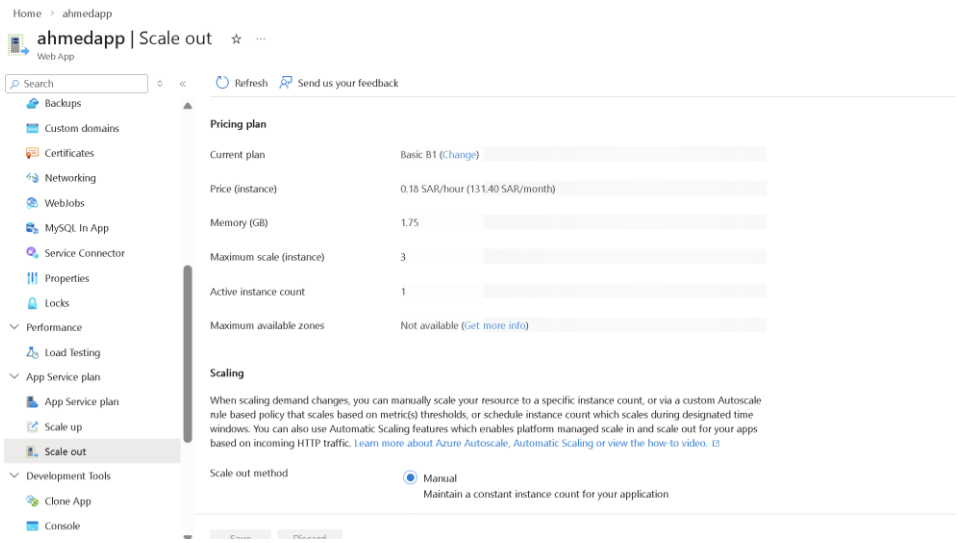
1 **Open App Service Plan**
App Service → App Service plan → App Service plan (in the left menu). This shows the current plan details.

2 **Review plan details**
Note: Resource Group = app_rg01, Location = Canada Central, Pricing plan = B1, Instance count = 1, OS = Windows, Zone redundant = Disabled.

Plan Property	Lab Value	What It Controls
Name	ASP-apprg01-94e2	Identifies the plan — multiple apps can share one plan
Pricing plan	Basic B1	CPU, RAM, storage, features — click Scale up to change
Instance Count	1	Number of parallel VM instances — click Scale out to change
OS	Windows	Set at plan creation — cannot change after. Linux plans are separate.
Zone redundant	Disabled	Requires Premium v2+ — spreads instances across AZs for HA
Apps / Slots	1 / 0	Apps sharing this plan — slots require Standard tier or above


Home › ahmedapp
ahmedapp | App Service plan
Web App
Search
Backups
Custom domains
Certificates
Networking
WebJobs
MySQL In App
Service Connector
Properties
Locks
Performance
Load Testing
App Service plan
App Service plan
Scale up
Scale out
Development Tools
Clone App
Console
Refresh
An App Service always runs in an App Service plan. An App Service plan defines a set of compute resources for a web app to run.
Current App Service plan
Name
ASP-apprg01-94e2
(Change plan)
Plan details
Resource Group
app_rg01
Location
Canada Central
Pricing plan
B1 (Scale up)
Instance Count
1 (Scale out)
App(s) / Slots
1 / 0
Operating System
Windows
Zone redundant
Disabled


Home › ahmedapp
ahmedapp | Scale up
Web App
Hardware view
Feature view
Showing 23 App Service pricing pl
Name
ACU/vCPU
vCPU
Memory (GB)
Remote Storage (GB)
Scale (instance)
Cost per hour (instance)
Cost per month (instance)
Dev/Test (For less demanding workloads)
Free F1 60 minutes/day compute N/A 1 1 N/A Free Free
Basic B1 100 1 1.75 10 3 0.18 SAR 131.40 SAR
Shared D1 240 minutes/day compute N/A 1 1 N/A 0.061 SAR 44.621 SAR
Basic B2 100 2 3.5 10 3 0.364 SAR 265.538 SAR
Basic B3 100 4 7 10 3 0.728 SAR 531.075 SAR
Production (For most production workloads)
Premium v3 P0V3 195* 1 4 250 30 0.221 SAR 161.061 SAR
Premium v3 P1V3 195 2 8 250 30 0.473 SAR 344.925 SAR
Premium v3 P2V3 195 4 16 250 30 0.945 SAR 689.85 SAR
Premium v3 P3V3 195 8 32 250 30 1.89 SAR 1379.70 SAR
Premium v3 P1mv3 195* 2 16 250 30 0.486 SAR 355.054 SAR
Premium v3 P2mv3 195* 4 32 250 30 0.973 SAR 710.108 SAR
Premium v3 P3mv3 195* 8 64 250 30 1.946 SAR 1420.489 SAR


Home › ahmedapp
ahmedapp | Scale out
Web App
Refresh
Send us your feedback
Pricing plan
Current plan
Basic B1 (Change)
Price (instance)
0.18 SAR/hour (131.40 SAR/month)
Memory (GB)
1.75
Maximum scale (instance)
3
Active instance count
1
Maximum available zones
Not available (Get more info)
Scaling
When scaling demand changes, you can manually scale your resource 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

Lab 4.4 — Scale Up (Change Pricing Tier)

Scale up changes the VM size (CPU, RAM, remote storage) and unlocks features. Your lab currently runs Basic B1 (1 vCPU, 1.75 GB RAM, 0.18 SAR/hr). The Scale up blade shows all available tiers side by side for comparison.

1 **Open Scale up**
App Service → App Service plan → Scale up. The hardware view shows all tiers with ACU/vCPU, vCPU count, memory, remote storage, cost per hour, and cost per month.

2 **Compare tiers**
Review the pricing table below — sourced directly from your lab screenshot. Basic B1 is currently selected (checkmark).

Tier	vCPU	RAM (GB)	Remote Storage (GB)	Max Instances	Cost/hr (SAR)	Cost/month (SAR)	Best For
Free F1	N/A	1	1	N/A	Free	Free	Dev/Test only — 60 min/day compute
Basic B1	1	1.75	10	3	0.18	131.40	Lab workloads (current)
Basic B2	2	3.5	10	3	0.364	265.54	Light production
Basic B3	4	7	10	3	0.728	531.08	Medium workloads
Premium v3 P0V3	1	4	250	30	0.221	161.06	Production — best value
Premium v3 P1V3	2	8	250	30	0.473	344.93	Production — standard
Premium v3 P2V3	4	16	250	30	0.945	689.85	High-traffic production
Premium v3 P3V3	8	32	250	30	1.89	1379.70	Large-scale production

Azure CLI

```
# Scale up to Standard S1 (enables slots, autoscale, custom domains with SSL)
az appservice plan update \
  --name ASP-apprg01-94e2 \
  --resource-group app_rg01 \
  --sku S1

# Scale up to Premium v3 P1V3 (best for production)
az appservice plan update \
  --name ASP-apprg01-94e2 \
```

```
  --resource-group app_rg01 \
  --sku P1V3

# Scale back down to Basic B1 (cost saving after lab)
az appservice plan update \
  --name ASP-apprg01-94e2 \
  --resource-group app_rg01 \
  --sku B1

# Check current plan SKU
az appservice plan show \
  --name ASP-apprg01-94e2 \
  --resource-group app_rg01 \
  --query "sku.{Tier:tier,Size:size,Capacity:capacity}" -o json
```

Note

Changing the pricing tier restarts your app briefly (~30 seconds). There is no data loss. Scale up before a traffic event, and scale back down after to save cost. The plan price applies to all apps sharing the plan.

Lab 4.5 — Scale Out (Add Instances)

Scale out adds parallel VM instances to distribute load. Your current Basic B1 plan supports up to 3 instances manually. The Scale out blade shows the current plan pricing, memory, and scaling method options.

1 — Open Scale out

App Service → App Service plan → Scale out. Review the current plan summary: Basic B1, 0.18 SAR/hr per instance (131.40 SAR/month), Memory = 1.75 GB, Max scale = 3 instances.

2 — Review scale out methods

Basic B1 supports Manual scaling only. Autoscale (metric-based rules) and Automatic Scaling (platform-managed) require Standard tier or above.

3 — Set manual instance count to 2

Under Scaling → Scale out method = Manual → Instance count = 2 → Save. Azure provisions a second instance within 2–3 minutes.

4 — Verify in Instances blade

Navigate to Settings → Instances — you should now see two server entries in the canadacentral zone.

Azure CLI

```
# Set manual instance count to 2
az appservice plan update \
  --name ASP-apprg01-94e2 \
  --resource-group app_rg01 \
  --number-of-workers 2

# Verify current instance count
az appservice plan show \
  --name ASP-apprg01-94e2 \
  --resource-group app_rg01 \
  --query "sku.capacity" -o tsv

# ── Autoscale (requires Standard tier or above) ──────────────
# First scale up: az appservice plan update --sku S1 ...

# Create autoscale setting
PLAN_ID=$(az appservice plan show \
  --name ASP-apprg01-94e2 \
  --resource-group app_rg01 --query id -o tsv)

az monitor autoscale create \
  --resource-group app_rg01 \
  --resource $PLAN_ID \
  --resource-type Microsoft.Web/serverFarms \
  --name autoscale-ahmedapp \
  --min-count 1 \
  --max-count 5 \
  --count 1

# Add scale-out rule: CPU > 70% for 5 min → add 1 instance
az monitor autoscale rule create \
  --resource-group app_rg01 \
  --autoscale-name autoscale-ahmedapp \
  --condition "CpuPercentage > 70 avg 5m" \
  --scale out 1

# Add scale-in rule: CPU < 30% for 10 min → remove 1 instance
az monitor autoscale rule create \
  --resource-group app_rg01 \
  --autoscale-name autoscale-ahmedapp \
  --condition "CpuPercentage < 30 avg 10m" \
  --scale in 1
```

Scale Method	Tier Required	Trigger	Max Instances	Best For
Manual	Basic B1+	You set instance count manually	3 (Basic)	Predictable load — set before peak
Autoscale (rules)	Standard S1+	CPU %, memory %, HTTP queue, custom metrics	10–30	Variable load with metric thresholds
Automatic Scaling	Premium v3+	Platform-managed — no rules needed	30	Unpredictable bursts — hands-off
Zone Redundant	Premium v2+	Spread instances across Availability Zones	30	High availability / disaster recovery