Part 12: Deployment

Azure Deployment

1. Prerequisites

* Azure Account: Ensure you have an Azure subscription (free or paid).
* Azure CLI: Install the Azure CLI for managing Azure resources from the command line: [Installation Guide](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli).
* SQL Server: If using Azure SQL Database, set up your database.

2. Prepare the Application

* Set ASPNETCORE\_ENVIRONMENT to Production in your environment variables or appsettings.Production.json.
* Update connection strings in appsettings.Production.json for production:

{

"ConnectionStrings": {

"DefaultConnection": "Server=<azure-sql-server>.database.windows.net;Database=<your-database>;User Id=<username>;Password=<password>;"

}

}

3. Publish the Application

Run in your project directory:

dotnet publish -c Release -o ./publish

4. Create an Azure App Service

Log in to Azure:

az login

Create Resource Group:

az group create --name ECommerceResourceGroup --location "East US"

Create App Service Plan:

az appservice plan create --name ECommercePlan --resource-group ECommerceResourceGroup --sku B1 --is-linux

Create Web App:

az webapp create --resource-group ECommerceResourceGroup --plan ECommercePlan --name ECommerceAPI --runtime "DOTNET|8.0"

5. Deploy the Application

Deploy Files to Azure:

az webapp deploy --resource-group ECommerceResourceGroup --name ECommerceAPI --src-path ./publish

Set Environment Variables:

az webapp config appsettings set --resource-group ECommerceResourceGroup --name ECommerceAPI --settings ASPNETCORE\_ENVIRONMENT=Production ConnectionStrings\_\_DefaultConnection="..."

* Verify Deployment: Navigate to https://ECommerceAPI.azurewebsites.net.

6. Set Up Azure SQL Database

Create SQL Server & Database:

az sql server create --name ECommerceSQLServer --resource-group ECommerceResourceGroup --location "East US" --admin-user <username> --admin-password <password>

az sql db create --resource-group ECommerceResourceGroup --server ECommerceSQLServer --name ECommerceDB --service-objective S0

Run migrations:

dotnet ef database update

7. Configure Logging

Enable Logging:

az webapp log config --resource-group ECommerceResourceGroup --name ECommerceAPI --application-logging true --level information

View Logs:

az webapp log tail --resource-group ECommerceResourceGroup --name ECommerceAPI

8. Test the Deployed Application

* Test endpoints using the live URL, e.g., GET /api/v1/Products.
* Check Swagger documentation at https://ECommerceAPI.azurewebsites.net/swagger.

9. Future Enhancements

* Custom Domain, SSL Certificates, Scaling options can be configured through Azure portal or CLI.

SmarterASP.NET Deployment

1. Prepare the Application

* Similar to Azure, ensure your appsettings.Production.json is configured for production.

2. Publish the Application

Publish your application:

dotnet publish -c Release -o ./publish

3. Set Up SmarterASP.NET

* Account Setup: Ensure you have an account with SmarterASP.NET; sign up if needed.
* Create Site:
  + Log into SmarterASP.NET control panel.
  + Go to Create Site or Add New Site.
  + Choose your domain or subdomain, select .NET Core hosting, and create the site.

4. Deploy the Application

* FTP/SFTP Upload:
  + Use the FTP/SFTP credentials provided by SmarterASP.NET.
  + Upload the contents of the publish folder to the root of your site directory on the server.
* Configure Web.config:
  + If needed, create or modify web.config in the root to configure ASP.NET Core module:

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<location path="." inheritInChildApplications="false">

<system.webServer>

<handlers>

<add name="aspNetCore" path="\*" verb="\*" modules="AspNetCoreModuleV2" resourceType="Unspecified" />

</handlers>

<aspNetCore processPath="dotnet" arguments=".\YourApplication.dll" stdoutLogEnabled="false" stdoutLogFile=".\logs\stdout" hostingModel="inprocess" />

</system.webServer>

</location>

</configuration>

* Set Environment Variables:
  + SmarterASP.NET provides an interface for setting environment variables for your site. Use this to set ASPNETCORE\_ENVIRONMENT=Production and add your connection strings.

5. Test the Deployed Application

* Visit your site URL (e.g., http://yourdomain.smarterasp.net) to ensure it's running.
* Test endpoints and check Swagger if deployed.

6. Future Enhancements

* SSL: Enable SSL through SmarterASP.NET's control panel.
* Database: Configure external or SmarterASP.NET's database services as needed.