Lab 2: Azure Web app

Objective

1. Provision an Azure Web App
2. Create a sample local Web project
3. Create a Repo on GitHub and push the local code

Note:

1. Steps to log into VM (Each participant will have a separate user/pwd)
    a. Open in a private window https://training.datacouch.io/pluralsight
    b. Enter the provided username and password.

    All the following steps are to be done within the Virtual Machine.

    c. Steps to Log into Azure Portal (4-5 participants will be in each group)
        i. Go to https://portal.azure.com
        ii. Login with the supplied credentials (username and password).
            1. Each group has a unique integer for their login [1-4] eg. **usergroup[1-4]** that will remain same for the duration of the course.
            2. Complete username and Password will be provided in the class.
            3. Each usergroup has an associated resource group which is **rg-usergroup[1-4]**

Section 1: Provision an Azure Web app

Steps

1. Login into the Virtual Machine
2. Login into Azure Portal
3. Type **"App Services"** on the search bar and select "App Services" from dropdown.
4. Click on **"+Create"** button
5. **Basics Tab**
    a. Select the resource group from the dropdown.
    b. Give unique name to Web App name as **"runwebapp"+"group number"+"participant id"** add your group number and instance as suffix e.g. if your group number is 4 and participant id is 22, name the machine as **"runwebapp422"** .
    c. Publish: Choose **"Code"**
    d. Runtime stack: Select **".NET 6"**
    e. Operating System: Choose **"Windows"**

f. Region: Choose **East US**
g. Give unique name to App Service plan name as **"appserviceplan"+"group number"+"participant id"** add your group number and participant id as suffix e.g. if your group number is 4 and participant id is 22, name the app service plan as **"appservice422"** .
h. SKU and size: Select **"Free F1" or "Standard S1" . If Free is not available, then choose standard**
i. Don't change the other defaults and click on **"Next: Deployment"**

6. **Deployment Tab**
   a. Continuous Deployment: Keep the default as **"Disable"** and click on **"Next: Networking"**
7. **Networking Tab**
   a. Don't change the defaults and click on **"Next: Monitoring"**
8. **Monitoring Tab**
   a. Don't change the defaults and click on **"Review + Create"**
9. **Review+Create Tab**
   a. Let the validation run and pass.
   b. Click on **"Create"** and wait for the deployment to complete
   c. Click on **"Go to resource"**. This will take you to the overview page of the newly created Web App
10. **Download Publish Profile**
    a. On the overview tab of the Azure Web App, click on **"Download publish profile"** to download the publish settings file. We will use the contents of this file when creating the secret in our Github repo.

Observations

1. In the overview section for the Web App, Click on the "Browse" button or on the URL link.
   a. It will open a new browser window and you will see the default sample web app.
2. In the overview section->Monitoring for the Web App, notice the different metrics related to Web traffic.

Section 2: Create a local webapp

Steps

1. Login into the Virtual Machine
2. Open VS Code Terminal
3. Type **"mkdir azuregithubwebapp"**
4. Type **"cd azuregithubwebapp"**
5. Check if dotnet 6 is installed.
   a. Type **"dotnet –version"** and you should see output starting with "6.x"
   b. If dotnet is not installed, you can install the same by running the following command

i. **sudo snap install dotnet-sdk --classic --channel=6.0.**
   Please connect for the sudo password if asked.

6. Type **"dotnet new mvc"** to create a sample Asp.net core Web Application
7. Type **"code . -r "** to reload VS Code with the newly created project folder or you can manually open the project folder.
8. Type **"dotnet new gitignore"** to add the file to the project
9. Type **"git init"** to initialize git
10. Type **"git add ."** to add all the files
11. Type **"git commit -m "First commit"** to commit locally

Section 3: Create a Repo on GitHub and push the local code

Steps

1. Login into the Virtual Machine
2. Login to your GitHub account and create a new repo with the defaults. You can keep the same name as the dotnet project **"azuregithubwebapp"**
3. Copy the **github repo url** and go back to VS Code terminal
4. Type the following commands
   a. **git branch -M main**
   b. **git remote add origin *<repo>***
   c. **git push -u origin main**
5. Refresh your GitHub repo and you should see all the files uploaded.
6. Go to settings and click on **"secrets and variables".** Click on **Actions**.
   a. On the right side, click on **"New repository secret"** .
      i. Name: **AZURE_WEBAPP_PUBLISH_PROFILE**
      ii. Secret: Paste the entire contents of the publish settings file that you downloaded from section 1 and click on **"Add Secret"**
7. Go to Actions and find the **"Deploy a .NET Core app to an Azure Web App"** workflow and click on **Configure**
8. In the default **azure-webapps-dotnet-core.yml** file, remove all the default code and copy and paste the below contents and commit and save the file.
   Please change the name of the webapp marked in red to your Azure web application name.

```
name: AzureWebApp

env:
  AZURE_WEBAPP_NAME: <webapp name>    # set this to the name of your Azure Web App
  AZURE_WEBAPP_PACKAGE_PATH: '.'      # set this to the path to your web app project, defaults to
the repository root
  DOTNET_VERSION: '6'                 # set this to the .NET Core version to use

on:
#  push:
 #   branches: [ "main" ]
  workflow_dispatch:
```

```yaml
permissions:
  contents: read

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3

      - name: Set up .NET Core
        uses: actions/setup-dotnet@v3
        with:
          dotnet-version: ${{ env.DOTNET_VERSION }}

      - name: Set up dependency caching for faster builds
        uses: actions/cache@v3
        with:
          path: ~/.nuget/packages
          key: ${{ runner.os }}-nuget-${{ hashFiles('**/packages.lock.json') }}
          restore-keys: |
            ${{ runner.os }}-nuget-
      - name: Build with dotnet
        run: dotnet build --configuration Release

      - name: dotnet publish
        run: dotnet publish -c Release -o ${{env.DOTNET_ROOT}}/myapp

      - name: Upload artifact for deployment job
        uses: actions/upload-artifact@v3
        with:
          name: .net-app
          path: ${{env.DOTNET_ROOT}}/myapp

  deploy:
    permissions:
      contents: none
    runs-on: ubuntu-latest
    needs: build
    environment:
      name: 'Development'
      url: ${{ steps.deploy-to-webapp.outputs.webapp-url }}

    steps:
      - name: Download artifact from build job
        uses: actions/download-artifact@v3
        with:
          name: .net-app

      - name: Deploy to Azure Web App
        id: deploy-to-webapp
        uses: azure/webapps-deploy@v2
        with:
          app-name: ${{ env.AZURE_WEBAPP_NAME }}
          publish-profile: ${{ secrets.AZURE_WEBAPP_PUBLISH_PROFILE }}
          package: ${{ env.AZURE_WEBAPP_PACKAGE_PATH }}
```

9. We have configured this workflow to run manually. Click on Actions again. Then click on the workflow name "**AzureWebApp**" and click on **"Run workflow**".

10. Wait for the workflow to start the jobs and review both the build and deploy jobs and see their progress. Once the entire workflow is complete, go back to the Web Application url (Azure web app) and you will see the changed Asp.net web application.
11. You can edit the YAML file and uncomment the push trigger and every time you have a commit and push to main, the workflow will automatically run.

End of Lab.