Github Actions Lab Practice

By the end of this lab, you will: - Build and test a .NET **Web API** with GitHub Actions. - Deploy the Web API directly (no Docker) to **Azure App Service** using GitHub Actions.

Prerequisites

- GitHub account & repository.
- **.NET 8 SDK** installed locally (`dotnet --list-sdks`).
- **Azure subscription** + **Azure CLI** (`az version`).
- A created **Azure App Service** (Windows/Linux runtime, .NET 8).

---

0) Create the solution locally

```
# Create solution & API
mkdir simple-api && cd simple-api
 dotnet new webapi -n SimpleApi --use-controllers
 dotnet new sln -n SimpleApi
 dotnet sln add SimpleApi/SimpleApi.csproj

# Add test project
 dotnet new xunit -n SimpleApi.Tests
 dotnet add SimpleApi.Tests/SimpleApi.Tests.csproj package coverlet.collector
--version 6.*
 dotnet sln add SimpleApi.Tests/SimpleApi.Tests.csproj
 dotnet add SimpleApi.Tests reference SimpleApi/SimpleApi.csproj
```

Run locally:

```
 dotnet restore
 dotnet build -c Release
 dotnet test -c Release
```

Initialize git & push:

```
git init
git add .
git commit -m "chore: simple .NET 8 Web API"
git branch -M main
git remote add origin https://github.com/<YOUR_ORG_OR_USER>/<YOUR_REPO>.git
git push -u origin main
```

---

1) Setup Azure App Service (one-time)

Login & set subscription:

```
az login
az account set --subscription <SUBSCRIPTION_ID>
```

Create a resource group + plan + app:

```
RG=rg-simple-api
LOC=eastus
PLAN=plan-simple-linux
APP=simple-api-<unique>

az group create -n $RG -l $LOC
az appservice plan create -g $RG -n $PLAN --sku B1 --is-linux
az webapp create -g $RG -p $PLAN -n $APP --runtime "DOTNET|8.0"
```

---

2) Configure GitHub → Azure OIDC

1. In **Microsoft Entra ID** → App registrations → Register new app `github-oidc-simple`.
2. Note **Application (client) ID** and **Tenant ID**.
3. Add a federated credential → Template: GitHub Actions → Repo: `OWNER/REPO`, Environment: `production`.
4. In Azure, give the app **Contributor** access on the resource group.
5. In GitHub repo → Settings → Secrets and variables → Actions:
   - `AZURE_CLIENT_ID` = app client ID
   - `AZURE_TENANT_ID` = tenant ID
   - `AZURE_SUBSCRIPTION_ID` = subscription ID

---

3) Create GitHub Actions workflow

Create **.github/workflows/deploy.yml**

```yaml
name: build-and-deploy

on:
  push:
    branches: [ main ]

permissions:
  id-token: write   # OIDC
  contents: read

env:
```

```yaml
  AZURE_WEBAPP_NAME: simple-api-<unique> # your app name

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v5

      - name: Setup .NET 8
        uses: actions/setup-dotnet@v5
        with:
          dotnet-version: '8.x'

      - name: Restore
        run: dotnet restore

      - name: Build
        run: dotnet build --no-restore -c Release

      - name: Test
        run: dotnet test --no-build -c Release

      - name: Publish
        run: |
          dotnet publish SimpleApi/SimpleApi.csproj -c Release -o publish
/p:UseAppHost=false
          cd publish && zip -r ../app.zip .

      - name: Azure login
        uses: azure/login@v2
        with:
          client-id: ${{ secrets.AZURE_CLIENT_ID }}
          tenant-id: ${{ secrets.AZURE_TENANT_ID }}
          subscription-id: ${{ secrets.AZURE_SUBSCRIPTION_ID }}

      - name: Deploy to Azure Web App
        uses: azure/webapps-deploy@v3
        with:
          app-name: ${{ env.AZURE_WEBAPP_NAME }}
          package: app.zip
```

---

4) Test deployment

1. Push code to `main` branch.
2. GitHub Actions workflow runs.
3. After success, visit:

```
https://<AZURE_WEBAPP_NAME>.azurewebsites.net/swagger
```

Checkpoint

- CI runs (restore, build, test).
- Deployment succeeds without manual publish profile.
- API responds on Azure.

Clean-up

```
az group delete -n rg-simple-api --yes --no-wait
```