

## Table of Contents

1. [Introduction](#)
2. [Prerequisites](#)
3. [Azure Setup](#)
4. [Infrastructure as Code \(Bicep\)](#)
5. [Minimal API Setup](#)
6. [CI/CD Pipeline with GitHub Actions](#)
7. [Running Unit Tests](#)
8. [Feature Flagging with LaunchDarkly](#)
9. [Automated Changelog](#)
10. [MS Teams Notifications](#)
11. [Trunk-Based Development & Squash Merging](#)
12. [Security Best Practices](#)
13. [Azure Service Principal Creation](#)
14. [Changelog Practices](#)

---

## 1 Introduction

This guide provides a **comprehensive step-by-step** approach to setting up a **CI/CD pipeline** for deploying a **.NET 8 Minimal API** to **Azure App Service** using **GitHub Actions** and **Infrastructure as Code (Bicep)**. It incorporates **feature flagging with LaunchDarkly**, **automated changelogs**, **Trunk-Based Development (TBD)**, **MS Teams notifications**, and **security best practices** to ensure a **scalable and secure workflow**.

---

## 2 Prerequisites

Ensure you have the following tools installed before proceeding:

### Required Tools

- [Azure CLI](#)

```
az --version
```

- [Bicep CLI](#)

```
az bicep install
```

- **GitHub CLI**

```
gh --version
```

- **.NET 8 SDK** ([Download](#))

```
dotnet --version
```

- **Docker** (Optional for containerization)

```
docker --version
```

## **Azure Access**

Login to Azure:

```
az login
```

Set your Azure Subscription:

```
az account set --subscription "YOUR_SUBSCRIPTION_ID"
```

---

## **3 Azure Setup**

### **Create a Resource Group**

```
az group create --name MyResourceGroup --location westeurope
```

### **Create Azure Key Vault**

```
az keyvault create --name my-keyvault --resource-group MyResourceGroup --  
location westeurope
```

### **Store Secrets in Key Vault**

```
az keyvault secret set set --vault-name my-keyvault --name sqlAdminUsername --value 'myAdminUser'
az keyvault secret set set --vault-name my-keyvault --name sqlAdminPassword --value 'SuperSecureP@ssw0rd!'
```

---

## 4 Infrastructure as Code (Bicep)

Define **Azure infrastructure** in `infra/main.bicep`:

```
param location string = 'westeurope'

resource keyVault 'Microsoft.KeyVault/vaults@2022-07-01' = {
  name: 'my-keyvault'
  location: location
}

resource sqlServer 'Microsoft.Sql/servers@2022-03-01' = {
  name: 'sqlserverdemo'
  location: location
  properties: {
    administratorLogin: secretUsername.properties.value
    administratorLoginPassword: secretPassword.properties.value
  }
}
```

### Deploy Infrastructure

```
az deployment group create --resource-group MyResourceGroup --template-file infra/main.bicep
```

---

## 5 Minimal API Setup

### Project Structure

Your project should have the following structure:

```
/MinimalApiProject
├─ Program.cs
├─ appsettings.json
├─ Dockerfile
├─ MinimalApiProject.csproj
└─ /Controllers
```

```
|— /Models  
|— /Services
```

## Program.cs (Minimal API)

```
using LaunchDarkly.Sdk;  
using LaunchDarkly.Sdk.Server;  
  
var builder = WebApplication.CreateBuilder(args);  
var sdkKey = builder.Configuration["LaunchDarkly:SdkKey"];  
var ldClient = new LdClient(sdkKey);  
  
var app = builder.Build();  
app.MapGet("/", async () => {  
    var user = User.WithKey("default-user");  
    bool featureEnabled = await ldClient.BoolVariationAsync("enableNewFeature",  
user, false);  
    return featureEnabled ? "🚀 New Feature is Enabled!" : "🔑 Old Version!";  
});  
app.Run();
```

## Dockerfile

```
FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base  
WORKDIR /app  
COPY . .  
ENTRYPOINT ["dotnet", "MinimalApiProject.dll"]
```

---

## 6 CI/CD Pipeline with GitHub Actions

### Optimized `.github/workflows/deploy.yml`

```
name: Deploy to Azure  
on:  
  push:  
    branches:  
      - main  
  pull_request:  
    branches:  
      - main  
  workflow_dispatch:  
    inputs:  
      environment:  
        description: 'Environment to deploy (staging/production)'
```

```

    required: true
    default: 'staging'

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v3

      - name: Setup .NET
        uses: actions/setup-dotnet@v3
        with:
          dotnet-version: '8.0.x'

      - name: Restore dependencies
        run: dotnet restore

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

      - name: Run Unit Tests
        run: dotnet test --no-build --configuration Release --verbosity normal

  deploy:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v3

      - name: Login to Azure
        uses: azure/login@v1
        with:
          creds: ${ secrets.AZURE_CREDENTIALS }

      - name: Deploy Bicep
        run: |
          az deployment group create \
            --resource-group MyResourceGroup \
            --template-file infra/main.bicep \
            --parameters environment=${ github.event.inputs.environment }

      - name: Post-Deployment Health Check
        run: |
          curl --fail https://minimal-api-demo.azurewebsites.net || exit 1

      - name: Send Deployment Notification
        if: success()
        uses: mspnp/teams-notify-action@v1
        with:
          webhook-url: ${ secrets.TEAMS_WEBHOOK_URL }

```

```
message: "🚀 Deployment to Azure (${github.event.inputs.environment }) successful! 🌿"
```

## 7 Running Unit Tests

### GitHub Actions: Run Tests on PRs & Deployment

```
name: Run Unit Tests
on:
  pull_request:
    branches:
      - main
  push:
    branches:
      - main

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v3

      - name: Setup .NET
        uses: actions/setup-dotnet@v3
        with:
          dotnet-version: '8.0.x'

      - name: Restore dependencies
        run: dotnet restore

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

      - name: Run Unit Tests
        run: dotnet test --no-build --configuration Release --verbosity normal
```

## 8 Feature Flagging with LaunchDarkly

1. **Sign up** at [LaunchDarkly](#).
2. **Get your SDK key** and store it in **Azure Key Vault**:

```
az keyvault secret set --vault-name my-keyvault --name LaunchDarklySdkKey --value 'your-sdk-key'
```

3. **Modify `Program.cs` to use feature flags** (see above).

---

## 9 Automated Changelog

Modify **GitHub Actions** to generate a changelog.

```
- name: Generate Changelog
  uses: TriPSs/conventional-changelog-action@v3
  with:
    github-token: ${{ secrets.GITHUB_TOKEN }}
    output-file: "CHANGELOG.md"
```

Add this step to your deployment workflow to automatically generate a changelog after each deployment.

---

## 10 MS Teams Notifications

1. **Create an Incoming Webhook in MS Teams** → Copy Webhook URL.
2. **Store in GitHub Secrets** → `TEAMS_WEBHOOK_URL`.
3. **Modify GitHub Actions:**

```
- name: Send MS Teams Notification
  uses: mspnp/teams-notify-action@v1
  with:
    webhook-url: ${{ secrets.TEAMS_WEBHOOK_URL }}
    message: "🚀 Deployment to Azure successful! 🎉"
```

Add this step to your deployment workflow to send notifications to MS Teams after successful deployments.

---

## 11 Trunk-Based Development & Squash Merging

Trunk-Based Development (TBD) follows these best practices:

- ☑ **Short-lived feature branches** – Merge daily or multiple times a day.
- ☑ **Squash merging** – Ensures a clean history by consolidating commits.
- ☑ **Require pull requests (PRs) and code reviews** before merging to `main`.
- ☑ **CI/CD tests must pass** before merging into `main`.

### Configuring GitHub Repository for Squash Merging

1. **Go to** your GitHub repo → **Settings** → **Merge Button**.
2. Enable:
  - ☑ **Allow squash merging**

- ✓ **Require pull requests before merging**
  - ✓ **Require passing CI checks before merging**
- 

## 1 2 **Security Best Practices**

- ✓ **Use Azure Key Vault** for storing secrets.
  - ✓ **Use GitHub Actions secrets** (`secrets.AZURE_CREDENTIALS`).
  - ✓ **Enable Role-Based Access Control (RBAC)** to restrict access.
  - ✓ **Run unit tests before merging and deploying** to prevent regressions.
  - ✓ **Implement Trunk-Based Development (TBD) with squash merging** to maintain code quality and prevent conflicts.
  - ✓ **Monitor deployments with MS Teams notifications** to ensure visibility into production changes.
  - ✓ **Use Azure Managed Identities** to avoid hardcoding credentials.
  - ✓ **Enable Multi-Factor Authentication (MFA)** for all Azure accounts.
  - ✓ **Regularly rotate secrets and credentials** to minimize risk.
- 

## 1 3 **Azure Service Principal Creation**

### Create a Service Principal

Create a service principal for GitHub Actions to authenticate with Azure:

```
az ad sp create-for-rbac --name "github-actions-sp" --role contributor --scopes /subscriptions/YOUR_SUBSCRIPTION_ID/resourceGroups/MyResourceGroup --sdk-auth
```

Copy the output JSON and add it to your GitHub repository secrets as `AZURE_CREDENTIALS`.

---

## 1 4 **Changelog Practices**

Maintaining a changelog is essential for tracking changes, improvements, and fixes in your project. Follow these practices to ensure a consistent and useful changelog:

### Commit Message Guidelines

- Use conventional commit messages to categorize changes:
  - `feat`: A new feature
  - `fix`: A bug fix
  - `docs`: Documentation changes
  - `style`: Code style changes (formatting, missing semi-colons, etc.)
  - `refactor`: Code refactoring without changing functionality
  - `test`: Adding or updating tests
  - `chore`: Maintenance tasks (updating dependencies, etc.)

### Generating Changelog



- Use the **conventional-changelog-action** in your GitHub Actions workflow to automatically generate a changelog.
- Ensure the changelog is updated with each deployment by including the changelog generation step in your CI/CD pipeline.

## Example Workflow

```
name: Deploy to Azure
on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main
  workflow_dispatch:
    inputs:
      environment:
        description: 'Environment to deploy (staging/production)'
        required: true
        default: 'staging'

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v3

      - name: Setup .NET
        uses: actions/setup-dotnet@v3
        with:
          dotnet-version: '8.0.x'

      - name: Restore dependencies
        run: dotnet restore

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

      - name: Run Unit Tests
        run: dotnet test --no-build --configuration Release --verbosity normal

  deploy:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v3

      - name: Login to Azure
```

```

    uses: azure/login@v1
    with:
      creds: ${ secrets.AZURE_CREDENTIALS }}

- name: Deploy Bicep
  run: |
    az deployment group create \
      --resource-group MyResourceGroup \
      --template-file infra/main.bicep \
      --parameters environment=${ github.event.inputs.environment }}

- name: Generate Changelog
  uses: TriPSs/conventional-changelog-action@v3
  with:
    github-token: ${ secrets.GITHUB_TOKEN }}
    output-file: "CHANGELOG.md"

- name: Post-Deployment Health Check
  run: |
    curl --fail https://minimal-api-demo.azurewebsites.net || exit 1

- name: Send Deployment Notification
  if: success()
  uses: mspnp/teams-notify-action@v1
  with:
    webhook-url: ${ secrets.TEAMS_WEBHOOK_URL }}
    message: "🚀 Deployment to Azure (${ github.event.inputs.environment }}) successful! 🎉"

```

By following these practices, you can maintain a clear and organized changelog that provides valuable insights into the evolution of your project.

## 🎯 Final Notes

☑️ **Fully automated CI/CD pipeline with Azure, GitHub Actions, Feature Flags, Changelogs, Trunk-Based Development, MS Teams Notifications, and Unit Testing!** 🚀