Roteiro de Deploy ASP.NET Web API no Azure

Guia Completo - Setembro 2025
Do projeto local ao Azure App Service

📋 Índice

---

1. Pré-requisitos

Requisitos Obrigatórios

- Projeto ASP.NET Core Web API funcionando localmente
- Conta Microsoft Azure (gratuita ou paga)
- Visual Studio Code com extensões .NET
- Git instalado e configurado

Ferramentas Necessárias

- **Azure CLI**: Para deploy via linha de comando
- **Extensão Azure Tools**: Para VS Code
- **Git**: Para controle de versão

🎯 Projeto Base

Este roteiro assume que você tem um projeto como criado no `install.md`:

```
MinhaAPITeste/
├── Controllers/WeatherForecastController.cs
├── Properties/launchSettings.json
├── appsettings.json
├── Program.cs
└── MinhaAPITeste.csproj
```

2. Preparação da Conta Azure

🆓 Criar Conta Azure Gratuita

1. **Acesse**: `https://azure.microsoft.com/free/`
2. **Clique em "Iniciar gratuitamente"**
3. **Faça login** com sua conta Microsoft
4. **Complete o cadastro** (pode solicitar cartão de crédito para verificação)

💰 Recursos Gratuitos Inclusos

- **App Service**: 750 horas/mês (suficiente para 1 app 24/7)
- **Bandwidth**: 15 GB/mês
- **Storage**: 5 GB
- **Duração**: 12 meses grátis

🔑 Verificar Acesso ao Portal

1. Acesse: `https://portal.azure.com`
2. Faça login com suas credenciais
3. Verifique se consegue acessar o dashboard

---

3. Instalação do Azure CLI

🪟 Windows

Opção 1: Installer MSI (Recomendado)

1. **Baixe**: https://aka.ms/installazurecliwindows
2. **Execute** o arquivo `.msi`
3. **Siga** o wizard de instalação

Opção 2: Via Chocolatey

```
choco install azure-cli -y
```

Opção 3: Via WinGet

```
winget install Microsoft.AzureCLI
```

☑️ Verificar Instalação

```
# Verificar versão
az --version

# Login no Azure
az login
```

---

4. Configuração do Projeto para Produção

📝 Configurar appsettings.Production.json

Crie o arquivo `appsettings.Production.json` no projeto:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

🛠️ Ajustar Program.cs para Produção

Verifique se o `Program.cs` está configurado corretamente:

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

// Importante: manter HTTPS redirect para produção
app.UseHttpsRedirection();

app.UseAuthorization();
app.MapControllers();
```

```
app.Run();
```

📦 Verificar .csproj

Certifique-se de que o arquivo `.csproj` está correto:

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="8.0.0" />
    <PackageReference Include="Swashbuckle.AspNetCore" Version="6.4.0" />
  </ItemGroup>

</Project>
```

🧪 Testar Build de Produção

```
# Testar build de produção
dotnet build --configuration Release

# Testar publicação
dotnet publish --configuration Release --output ./publish
```

Esse comando gera uma versão pronta para produção do seu projeto .NET e coloca todos os arquivos necessários na pasta publish, facilitando o deploy em servidores ou outros ambientes.

Passos:

- dotnet publish: Compila o projeto e prepara os arquivos para implantação (deploy), incluindo dependências e arquivos necessários para rodar a aplicação fora do ambiente de desenvolvimento.
- configuration Release: Usa a configuração de "Release", que geralmente ativa otimizações e desativa informações de depuração, tornando o aplicativo mais eficiente para produção.
- output ./publish: Especifica que os arquivos publicados serão colocados na pasta ./publish (relativa ao diretório atual). Resumo:

---

5. Deploy via Azure CLI

🔐 Login e Configuração Inicial

```
# Login no Azure
az login

# Listar assinaturas disponíveis
az account list --output table

# Selecionar assinatura (se tiver mais de uma)
az account set --subscription "<ID ou Nome exato da sua assinatura Azure>"

> Após executar este comando, o terminal permanece aberto e pronto para receber os próximos comandos. Você pode definir as variáveis de ambiente conforme mostrado abaixo, diretamente no mesmo terminal:

### 💡 **Como definir variáveis no terminal**

No terminal Bash (Linux, macOS, ou Windows via WSL/Git Bash), basta executar os comandos abaixo para definir variáveis de ambiente temporárias para a sessão atual:

```bash
export RESOURCE_GROUP="rg-minha-api"
export APP_NAME="minha-api-teste-2025"
export LOCATION="East US"
export APP_SERVICE_PLAN="asp-minha-api"
```

No PowerShell (Windows), use:

```
$env:RESOURCE_GROUP = "rg-minha-api"
$env:APP_NAME = "minha-api-teste-2025"
$env:LOCATION = "East US"
$env:APP_SERVICE_PLAN = "asp-minha-api"
```

Essas variáveis ficam disponíveis apenas enquanto o terminal estiver aberto. Execute-as antes dos comandos de criação de recursos para facilitar o uso dos parâmetros.

> Execute estes comandos diretamente no terminal antes de criar os recursos no Azure CLI. Recomenda-se rodar no mesmo terminal onde fará o deploy, logo após o login (`az login`) e antes dos comandos de criação do Resource Group, App Service Plan e Web App.

```


### **Criar Recursos no Azure**

```bash
# 1. Criar Resource Group
az group create --name $RESOURCE_GROUP --location "$LOCATION"

# 2. Criar App Service Plan (Free Tier)
az appservice plan create --name $APP_SERVICE_PLAN ---resource-group
```

```
$RESOURCE_GROUP --sku F1 --is-linux

# 3. Criar Web App
az webapp create \
  --name $APP_NAME \
  --resource-group $RESOURCE_GROUP \
  --plan $APP_SERVICE_PLAN \
  --runtime "DOTNETCORE:8.0"
```

🚀 Deploy da Aplicação

```
# Navegar para o diretório do projeto
cd MinhaAPITeste

# Publicar aplicação
dotnet publish --configuration Release --output ./publish

# Criar arquivo ZIP para deploy no Windows

No PowerShell, execute:

```powershell
Compress-Archive -Path .\publish* -DestinationPath .\app.zip
```

No CMD, use:

```
powershell Compress-Archive -Path .\publish* -DestinationPath .\app.zip
```

Deploy via Azure CLI

az webapp deployment source config-zip --resource-group $RESOURCE_GROUP --name $APP_NAME --src app.zip

Verificar status do deploy

az webapp show --resource-group $RESOURCE_GROUP --name $APP_NAME --query "state"

```
### 🌐 **Testar a Aplicação**

```bash
# Obter URL da aplicação
az webapp show --resource-group $RESOURCE_GROUP --name $APP_NAME --query
"defaultHostName" --output tsv
```

```
# Testar endpoint (substitua pela URL real)
curl https://minha-api-teste-2025.azurewebsites.net/weatherforecast
```

---

6. Deploy via VS Code

🔌 Instalar Extensão Azure Tools

1. **Abra VS Code**
2. **Acesse Extensions** (Ctrl+Shift+X)
3. **Procure por**: `Azure Tools`
4. **Instale**: `ms-vscode.vscode-azureextensionpack`

🔑 Login no Azure via VS Code

1. **Abra Command Palette** (Ctrl+Shift+P)
2. **Digite**: `Azure: Sign In`
3. **Siga** as instruções para fazer login

🚀 Deploy via Interface Visual

1. **Abra seu projeto** no VS Code
2. **Clique no ícone Azure** na barra lateral
3. **Expanda "App Service"`
4. **Clique com botão direito** na assinatura
5. **Selecione "Create New Web App..."**
6. **Configure**:
   - **Nome**: `minha-api-teste-2025` (deve ser único)
   - **Resource Group**: Criar novo ou usar existente
   - **Runtime**: `.NET 8 (STS)`
   - **Pricing Tier**: `Free (F1)`
7. **Deploy**:
   - **Botão direito** no Web App criado
   - **Selecione "Deploy to Web App..."**
   - **Escolha** a pasta do projeto
   - **Confirme** o deploy

---

7. Deploy via Portal Azure

🌐 Acessar Portal Azure

1. **Acesse**: `https://portal.azure.com`
2. **Faça login** com suas credenciais

🏗️ Criar App Service

1. **Clique em "Create a resource"**
2. **Procure por "Web App"**
3. **Clique em "Create"**
4. **Configure**:
    - **Subscription**: Sua assinatura
    - **Resource Group**: Criar novo `rg-minha-api`
    - **Name**: `minha-api-teste-2025`
    - **Publish**: `Code`
    - **Runtime stack**: `.NET 8 (LTS)`
    - **Operating System**: `Linux`
    - **Region**: `East US`
    - **Pricing plan**: `Free F1`
5. **Clique "Review + create"**
6. **Clique "Create"**

📤 Deploy via Deployment Center

1. **Acesse o Web App** criado
2. **No menu lateral**, clique em "Deployment Center"
3. **Selecione fonte**:
    - **Local Git**: Para deploy manual
    - **GitHub**: Para integração com repositório
    - **Azure DevOps**: Para CI/CD avançado

Opção A: Local Git

```
# Obter URL do Git remoto
# (disponível no Deployment Center)

# Adicionar remote do Azure
git remote add azure https://minha-api-teste-2025@minha-api-teste-
2025.scm.azurewebsites.net/minha-api-teste-2025.git

# Fazer push para deploy
git add .
git commit -m "Deploy inicial"
git push azure main
```

Opção B: GitHub

1. **Conecte** sua conta GitHub
2. **Selecione** o repositório
3. **Escolha** a branch (main/master)
4. **Configure** o workflow
5. **Save** - deploy automático será configurado

---

8. Configurações Pós-Deploy

⚙️ Configurar Application Settings

No Portal Azure ou via CLI:

```
# Via Azure CLI
az webapp config appsettings set \
  --resource-group $RESOURCE_GROUP \
  --name $APP_NAME \
  --settings ASPNETCORE_ENVIRONMENT=Production
```

Ou via Portal:

1. **App Service** → **Configuration**
2. **Application settings**
3. **+ New application setting**
4. **Name**: `ASPNETCORE_ENVIRONMENT`
5. **Value**: `Production`

🔒 Configurar Custom Domain (Opcional)

```
# Adicionar domínio customizado
az webapp config hostname add \
  --resource-group $RESOURCE_GROUP \
  --webapp-name $APP_NAME \
  --hostname "meudominio.com"

# Configurar SSL (requer domínio verificado)
az webapp config ssl bind \
  --resource-group $RESOURCE_GROUP \
  --name $APP_NAME \
  --certificate-thumbprint [THUMBPRINT] \
  --ssl-type SNI
```

📊 Configurar Application Insights (Recomendado)

```
# Criar Application Insights
az monitor app-insights component create \
  --app $APP_NAME \
  --location "$LOCATION" \
  --resource-group $RESOURCE_GROUP

# Obter connection string
az monitor app-insights component show \
  --app $APP_NAME \
  --resource-group $RESOURCE_GROUP \
  --query "connectionString"
```

---

9. Monitoramento e Logs

📋 Visualizar Logs em Tempo Real

```
# Via Azure CLI
az webapp log tail \
  --resource-group $RESOURCE_GROUP \
  --name $APP_NAME

# Baixar logs
az webapp log download \
  --resource-group $RESOURCE_GROUP \
  --name $APP_NAME
```

💻 Via Portal Azure

1. **App Service → Log stream**
2. **App Service → Logs**
3. **Application Insights → Live Metrics**

📊 Configurar Alertas

```
# Criar alerta para alta latência
az monitor metrics alert create \
  --name "API-High-Response-Time" \
  --resource-group $RESOURCE_GROUP \
  --scopes "/subscriptions/{subscription-
id}/resourceGroups/$RESOURCE_GROUP/providers/Microsoft.Web/sites/$APP_NAME" \
  --condition "avg http_response_time > 1000" \
  --description "Alerta quando tempo de resposta > 1s"
```

---

10. Solução de Problemas

❌ Erro: "Application startup failed"

Diagnóstico:

```
# Verificar logs
az webapp log tail --resource-group $RESOURCE_GROUP --name $APP_NAME
```

Soluções:

- Verificar se `ASPNETCORE_ENVIRONMENT` está correto
- Confirmar se todas as dependências estão no `.csproj`
- Verificar `appsettings.Production.json`

❌ Erro: "Site não carrega"

Verificações:

1. **Status do App Service**:

   ```
   az webapp show --resource-group $RESOURCE_GROUP --name $APP_NAME --query
   "state"
   ```

2. **Health check**:

   ```
   curl -I https://$APP_NAME.azurewebsites.net
   ```

❌ Erro: "502 Bad Gateway"

Soluções:

- Verificar se a aplicação está ouvindo na porta correta (variável `PORT`)
- Confirmar se `Program.cs` não tem configurações de porta específicas
- Verificar logs de startup

🔧 Comandos de Diagnóstico Úteis

```
# Status geral do App Service
az webapp show --resource-group $RESOURCE_GROUP --name $APP_NAME

# Informações de runtime
az webapp config show --resource-group $RESOURCE_GROUP --name $APP_NAME

# Reiniciar aplicação
az webapp restart --resource-group $RESOURCE_GROUP --name $APP_NAME
```

```
# Verificar métricas
az monitor metrics list \
  --resource
"/subscriptions/{id}/resourceGroups/$RESOURCE_GROUP/providers/Microsoft.Web/sites/
$APP_NAME" \
  --metric "Http2xx,Http4xx,Http5xx"
```

---

11. CI/CD com GitHub Actions

🗀 Estrutura do Workflow

Crie `.github/workflows/azure-deploy.yml`:

```
name: Deploy to Azure App Service

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - 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: Test
      run: dotnet test --no-build --verbosity normal --configuration Release

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

    - name: Deploy to Azure Web App
      uses: azure/webapps-deploy@v2
      with:
        app-name: 'minha-api-teste-2025'
        slot-name: 'production'
```

```
          publish-profile: ${{ secrets.AZURE_WEBAPP_PUBLISH_PROFILE }}
          package: ${{env.DOTNET_ROOT}}/myapp
```

🔑 Configurar Secrets

1. **Obter Publish Profile**:

```
az webapp deployment list-publishing-profiles \
  --resource-group $RESOURCE_GROUP \
  --name $APP_NAME \
  --xml
```

2. **No GitHub**:
   - **Settings → Secrets and variables → Actions**
   - **New repository secret**
   - **Name**: AZURE_WEBAPP_PUBLISH_PROFILE
   - **Value**: Conteúdo XML do publish profile

🚀 Configurar Deploy Automático

```
# Configurar GitHub Actions no Azure
az webapp deployment github-actions add \
  --resource-group $RESOURCE_GROUP \
  --name $APP_NAME \
  --repo "seuusuario/seu-repositorio" \
  --branch "main" \
  --login-with-github
```

---

12. Otimizações e Boas Práticas

⚡ Performance

Habilitar Compressão

```
// No Program.cs
builder.Services.AddResponseCompression(options =>
{
    options.EnableForHttps = true;
});

// Usar middleware
app.UseResponseCompression();
```

Configurar Cache

```
// Adicionar cache de resposta
builder.Services.AddResponseCaching();

// No controller
[ResponseCache(Duration = 300)]
[HttpGet]
public IEnumerable<WeatherForecast> Get()
{
    // ...
}
```

🔒 Segurança

Headers de Segurança

```
// No Program.cs
app.Use((context, next) =>
{
    context.Response.Headers.Add("X-Content-Type-Options", "nosniff");
    context.Response.Headers.Add("X-Frame-Options", "DENY");
    context.Response.Headers.Add("X-XSS-Protection", "1; mode=block");
    return next();
});
```

📊 Monitoramento

Health Checks

```
// Adicionar health checks
builder.Services.AddHealthChecks();

// Configurar endpoint
app.MapHealthChecks("/health");
```

💰 Custos

Monitorar Uso

```
# Verificar custos
az consumption usage list \
  --start-date 2025-09-01 \
  --end-date 2025-09-30
```

```
# Configurar budget alert
az consumption budget create \
  --budget-name "API-Budget" \
  --amount 10 \
  --time-grain Monthly \
  --time-period start-date=2025-09-01
```

🔄 Backup e Versionamento

```
# Configurar backup automático
az webapp config backup update \
  --resource-group $RESOURCE_GROUP \
  --webapp-name $APP_NAME \
  --container-url "https://mystorageaccount.blob.core.windows.net/backups" \
  --frequency 1 \
  --retention-period-in-days 30
```

☑️ Checklist de Deploy

Antes de colocar em produção:

- ☐ **Build local** funciona sem erros
- ☐ **Testes** passando (se existirem)
- ☐ **appsettings.Production.json** configurado
- ☐ **HTTPS** habilitado
- ☐ **Application Insights** configurado
- ☐ **Health checks** implementados
- ☐ **Logs** configurados adequadamente
- ☐ **Backup** configurado
- ☐ **CI/CD** funcionando
- ☐ **Alertas** de monitoramento ativos
- ☐ **Budget** configurado para controle de custos

🎉 Sucesso!

Sua ASP.NET Core Web API agora está rodando no Azure!

🧮 URLs Importantes

- **Aplicação**: `https://minha-api-teste-2025.azurewebsites.net`
- **API Endpoint**: `https://minha-api-teste-2025.azurewebsites.net/weatherforecast`
- **Portal Azure**: `https://portal.azure.com`
- **App Service**: Portal → Resource Groups → rg-minha-api → minha-api-teste-2025

🚀 Próximos Passos

1. **Configurar domínio customizado**
2. **Implementar autenticação JWT**
3. **Adicionar banco de dados**
4. **Configurar múltiplos ambientes** (dev, staging, prod)
5. **Implementar testes automatizados**
6. **Configurar Application Gateway** para alta disponibilidade

📚 Recursos Adicionais

- **Azure App Service Docs**: `https://docs.microsoft.com/azure/app-service/`
- **Azure CLI Reference**: `https://docs.microsoft.com/cli/azure/`
- **GitHub Actions for Azure**: `https://github.com/Azure/actions`

Parabéns pelo seu primeiro deploy no Azure! 🎉

Documento criado em setembro de 2025 - Versão 1.0
Para suporte, consulte a documentação oficial do Azure