Quick start

Developer docs.

# Source code structure changes

## .NET

1. Create QA / PROD env. inside .NET, update Properties/launchSettings.json, important:
   1. Environment name is CASE-SENSITIVE in angular
   2. And case-insensitive in .NET
2. Add Directory.Build.props to the top directory
3. Alter Startup.cs
   1. adjust how SPA run / building will be handled by .NET
   2. move some const to the top
4. Alter the \*.csproj so the SPA build / run process accepts and uses environment settings
   1. Alter the SPAROOT: <SpaRoot>..\SP.Test.Client\</SpaRoot>
   2. also we need to swap out: <Target Name="PublishRunWebpack" AfterTargets="ComputeFilesToPublish"> to:

<Target Name="PublishRunWebpack" AfterTargets="ComputeFilesToPublish">

<!-- As part of publishing, ensure the JS resources are freshly built in production mode -->

<Exec WorkingDirectory="$(SpaRoot)" Command="npm install" />

<Exec WorkingDirectory="$(SpaRoot)" Command="npm run build -- **--configuration $(EnvironmentName)**" />

<Exec WorkingDirectory="$(SpaRoot)" Command="npm run build:ssr -- **--configuration $(EnvironmentName)**" Condition=" '$(BuildServerSideRenderer)' == 'true' " />

<!-- Include the newly-built files in the publish output -->

<ItemGroup>

<DistFiles Include="$(SpaRoot)dist\\*\*; $(SpaRoot)dist-server\\*\*" />

<DistFiles Include="$(SpaRoot)node\_modules\\*\*" Condition="'$(BuildServerSideRenderer)' == 'true'" />

**<ResolvedFileToPublish Include="@(DistFiles->'%(FullPath)')" Exclude="@(ResolvedFileToPublish)">**

**<!--**

**The original line below works only if the front-end (SPA) is inside the main project folder (default .NET SPA scaffold).**

**To allow folders outside, we searched and found a solution here: https://www.codeofclimber.ru/2020/publishing-aspnetcore-app-web-application-with-frontend-monorepo-static-assets/**

**<RelativePath>%(DistFiles.Identity)</RelativePath>**

**-->**

**<RelativePath>spa-client/$([MSBuild]::MakeRelative($(MSBuildThisFileDirectory)$(SpaRoot)dist, %(DistFiles.FullPath)))</RelativePath>**

**<CopyToPublishDirectory>PreserveNewest</CopyToPublishDirectory>**

**<ExcludeFromSingleFile>true</ExcludeFromSingleFile>**

**</ResolvedFileToPublish>**

</ItemGroup>

</Target>

## Angular

1. Move the ClientApp to a separate folder (we will run Angular app separately from VSCode – faster dev. process)
2. Add environment settings
   1. Ensure RUN scripts for all environment
   2. Make sure enough memory is allocated for the app
3. Set up budgets
4. (optionally) Disable: AOT and OPTIMIZATIONS
5. Set up important TypeScript constraints (tsconfig.json):

  "compileOnSave": false,

  "compilerOptions": {

// …

  },

  "angularCompilerOptions": {

    "enableI18nLegacyMessageIdFormat": false,

    "strictInjectionParameters": true,

    "strictInputAccessModifiers": true,

    "strictTemplates": true,

    "enableIvy": true,

    "forceConsistentCasingInFileNames": true,

    "strict": true,

    "alwaysStrict": true,

    "noUnusedLocals": false,

    "noImplicitReturns": true,

    "noImplicitAny": true,

    "noImplicitThis": true,

    "strictNullChecks": true

  }

}

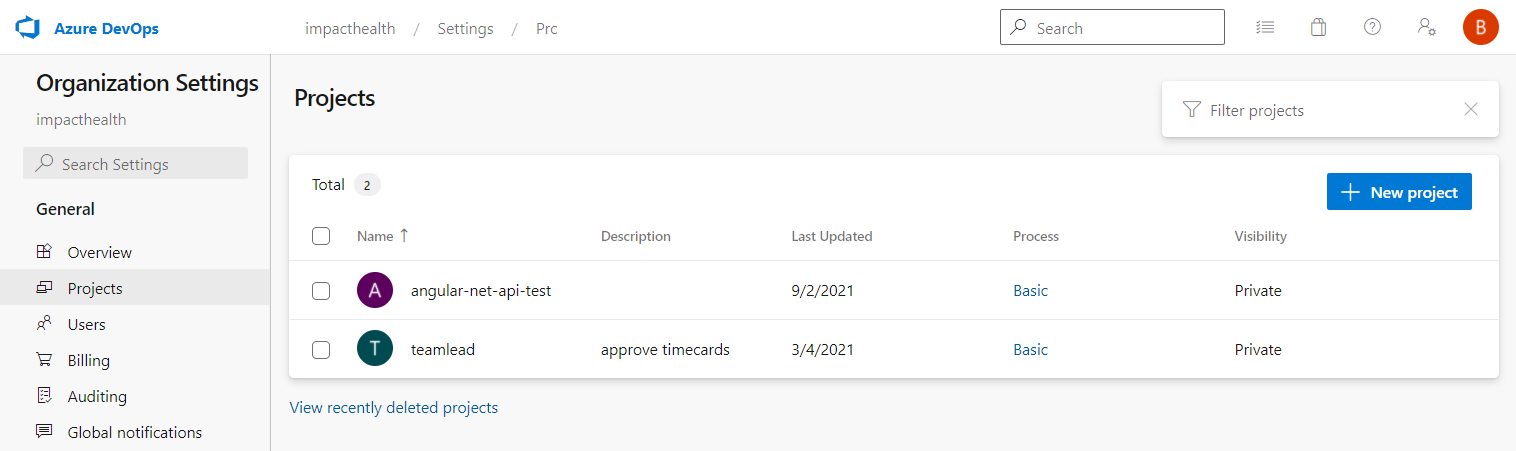
# Azure

## Portal

NOTE: we will use the TL subscription

1. Create new **resource group (e.g. Test-Dev, for the TL Subscription and for US East)**
2. Create at least a QA **app-service with production and deployment slots**

## DEVOPS



1. Create new **project**
2. Create service connections
   1. **Github** (name it github.com)
   2. **Azure deployment service connection** (name it: ‘Deployment Service Connection’) – the pipeline uses it (see: universalDeploymentServiceConnectionName)
      1. Select project settings
      2. Find service connections + press “New service connection”
      3. Select azure resource manager + Service principal (automatic)

Set Subscription and resource group, **give a meaningful name**

**Use that service connection name as a value for the AzureWebApp@1 tasks azureSubscription parameter!**

1. **Create new pipeline**
   1. Press CREATE
   2. Select: use the classic editor
   3. Select: github source
      1. Select the desired repo
      2. Select the feature branch where we got or 1st pipeline draft
   4. Select: configuration as code – YAML
      1. Browse YAML file inside the selected repo
   5. Press SAVE
      1. Select folder to save: /\_pipelines (in our case)
2. Navigate away, open pipeline, edit or RUN