Azure pipelines

Where Azure Pipelines?

- Azure Pipelines are a service within Azure Devops
- Azure Devops is a SaaS platform from Microsoft that allows you to develop and deploy the software with all its various services.

What is Azure Pipelines?

- It is an Azure service within Azure Devops that automatically builds, tests, and deploys code.
- It supports a ton of major languages, project types, and platforms.
- These pipelines can be done with templates or written with YAML.
- Its whats in charge of pushing your build artifact to wherever you need it to.

What does CI/CD stand for?

- CI/CD stands for Continuous Integration and Continuous Deployment!
- Continuous Integration means that developers in the team push their code to a repo in our case Azure git repo and it continuously merges the code they push, which then triggers a build which then tests the code.
- Continuous Deployment because as you push code, it will trigger a build and then push

that build artifact to one your environment that you have set up

Yaml file

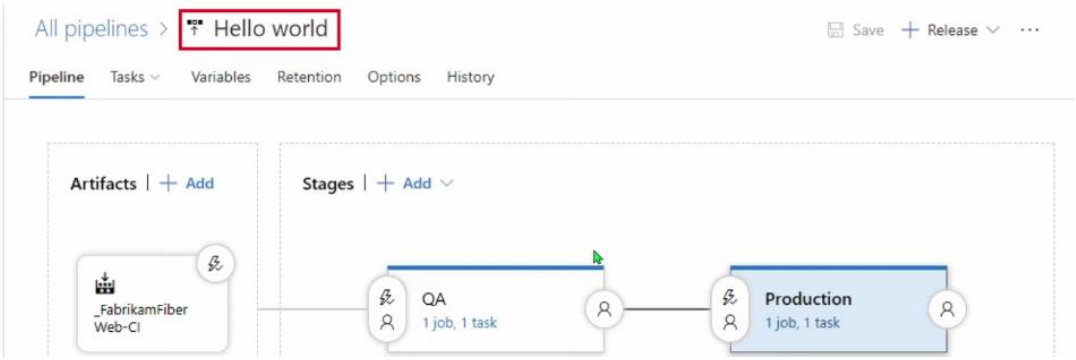
- So there are two ways of building your pipeline, you can use the visual designer Devops gives you or you can use a YAML file which then can be managed just like any other file in your project.

- I chose to use YAML files for this video since it allows me to have full control as well as use it multiple times if i need to for some reason in the future, it's a big more scalable.

- A Step is the smallest building block of a pipeline and it executes a specific command.

- Writing the YAML isn't that bad since Devops gives you many Tasks which are basically templates you just have to fill in. (we will see this when we create them)

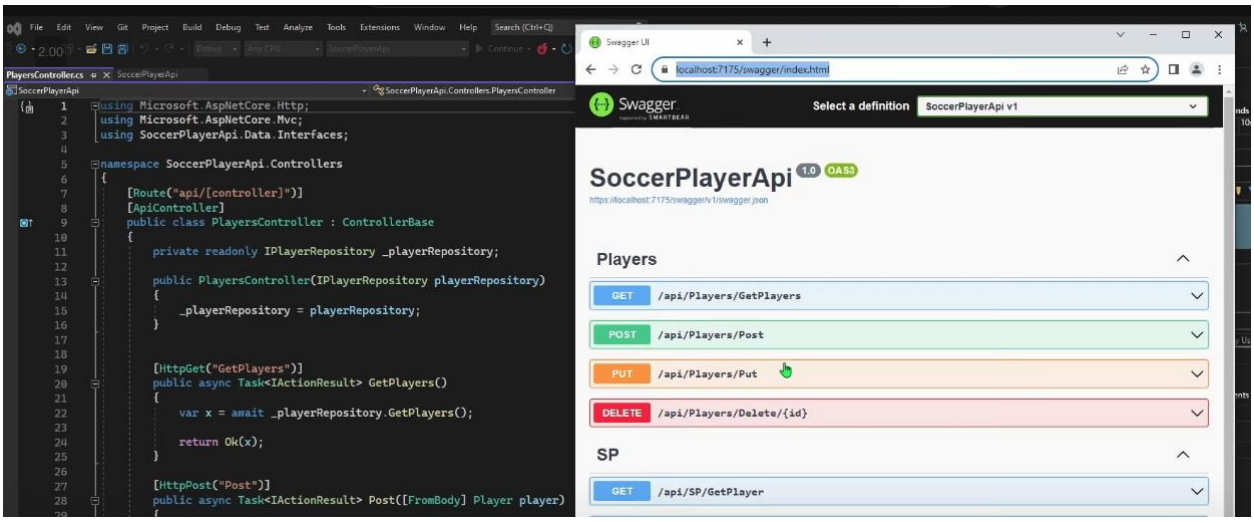
Visual Designer For Pipelines



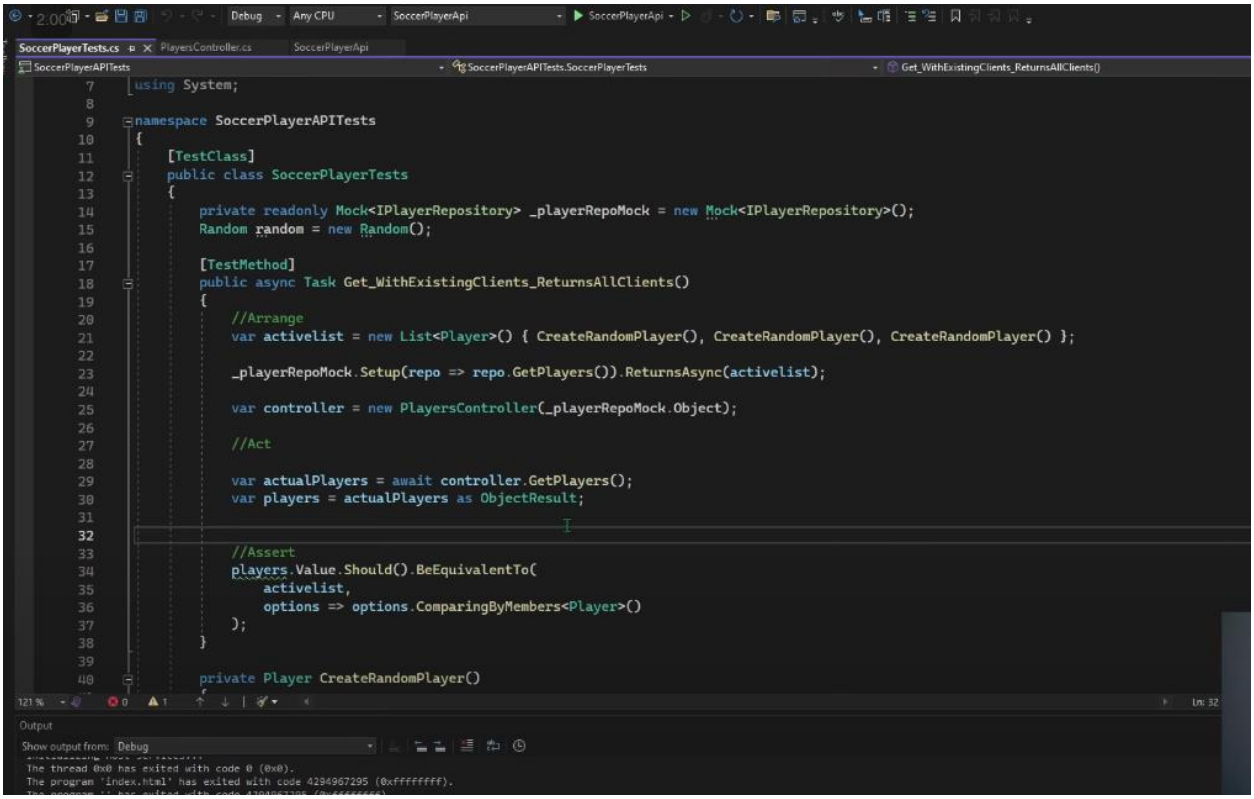
azure-pipelines.yml | azure-pipelines-build-template.yml | azure-pipelines-deployment-template.yml

```yaml
parameters: # defaults for any parameters that aren't specified
  buildConfiguration: 'Release'
  buildPlatform: 'Any CPU'
  vmImage: 'vs2017-win2016'

jobs:
  - job: Build
    displayName: 'Build job'
    pool:
      vmImage: ${{parameters.vmImage}}
    steps:
    # DotNet Framework build items
    - task: NuGetToolInstaller@0
      displayName: 'Use NuGet 4.4.1'
      inputs:
        versionSpec: 4.4.1

    - task: NuGetCommand@2
      displayName: 'NuGet restore'
      inputs:
        projects: |
          SamLearnsAzure/SamLearnsAzure.Database/SamLearnsAzure.Database.sqlproj
        vstsFeed: '030a758a-428f-4445-bce8-2c19ad9a56b3'

    - task: VSBuild@1
      displayName: 'Build database project'
      inputs:
        platform: '${{parameters.buildPlatform}}'
        configuration: '${{parameters.buildConfiguration}}'
        solution: SamLearnsAzure/SamLearnsAzure.Database/SamLearnsAzure.Database.sqlpr
```

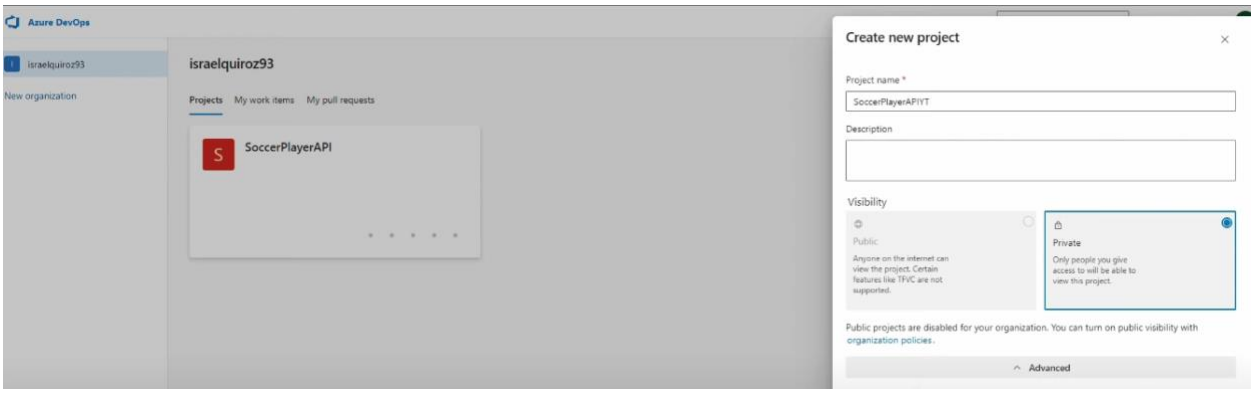
We want to push this website to production environment:

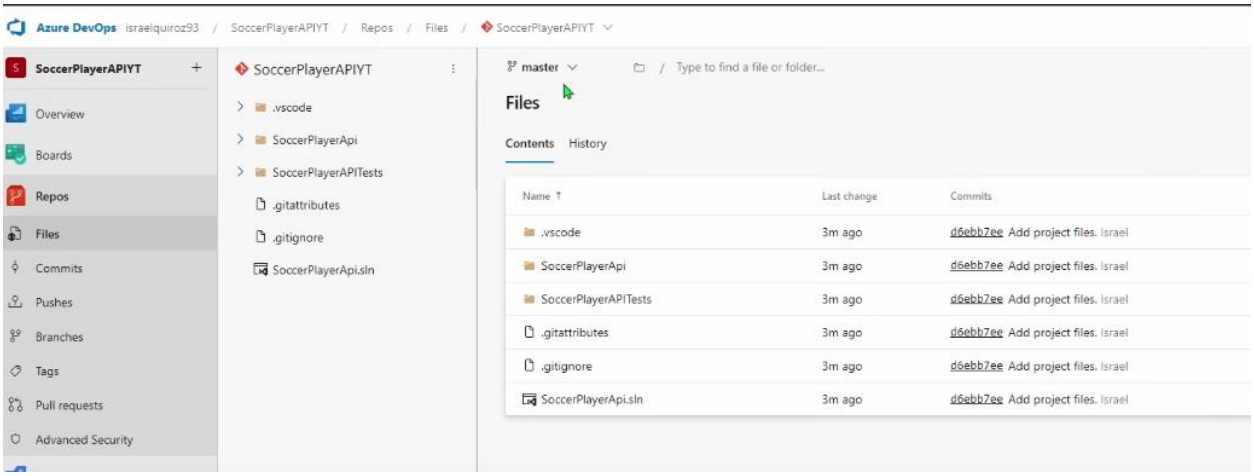


Unit test



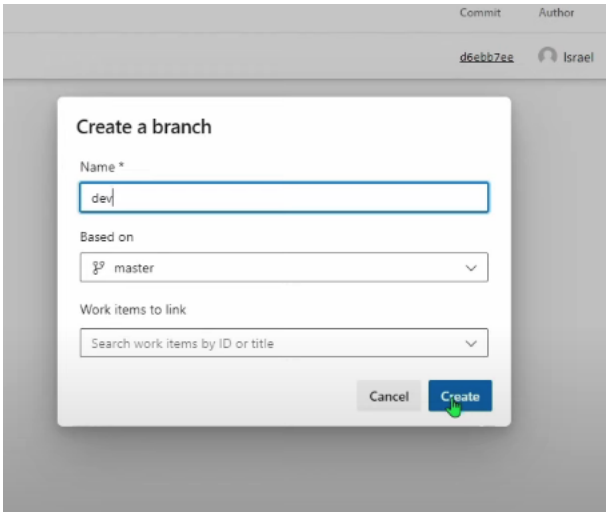
```
using System;

namespace SoccerPlayerAPITests
{
    [TestClass]
    public class SoccerPlayerTests
    {
        private readonly Mock<IPlayerRepository> _playerRepoMock = new Mock<IPlayerRepository>();
        Random random = new Random();

        [TestMethod]
        public async Task Get_WithExistingClients_ReturnsAllClients()
        {
            //Arrange
            var activelist = new List<Player>() { CreateRandomPlayer(), CreateRandomPlayer(), CreateRandomPlayer() };

            _playerRepoMock.Setup(repo => repo.GetPlayers()).ReturnsAsync(activelist);

            var controller = new PlayersController(_playerRepoMock.Object);

            //Act

            var actualPlayers = await controller.GetPlayers();
            var players = actualPlayers as ObjectResult;


            //Assert
            players.Value.Should().BeEquivalentTo(
                activelist,
                options => options.ComparingByMembers<Player>()
            );
        }

        private Player CreateRandomPlayer()
```



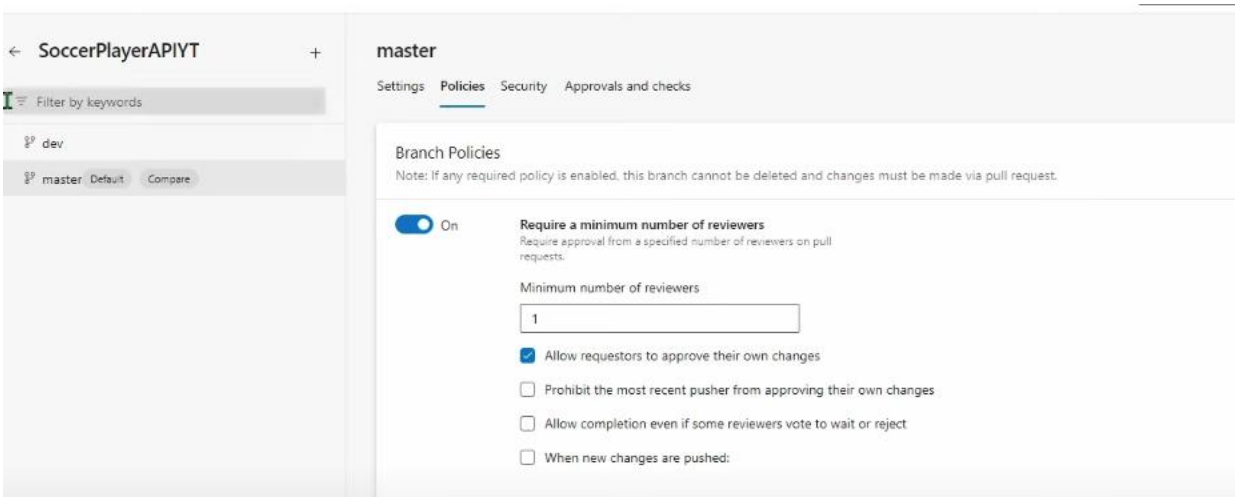


All code pushed

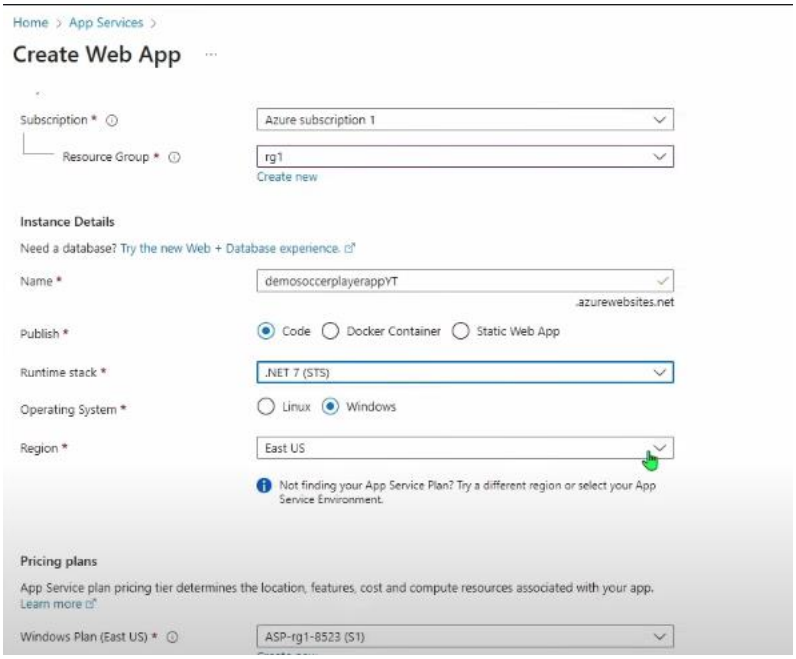
Create a dev branch :



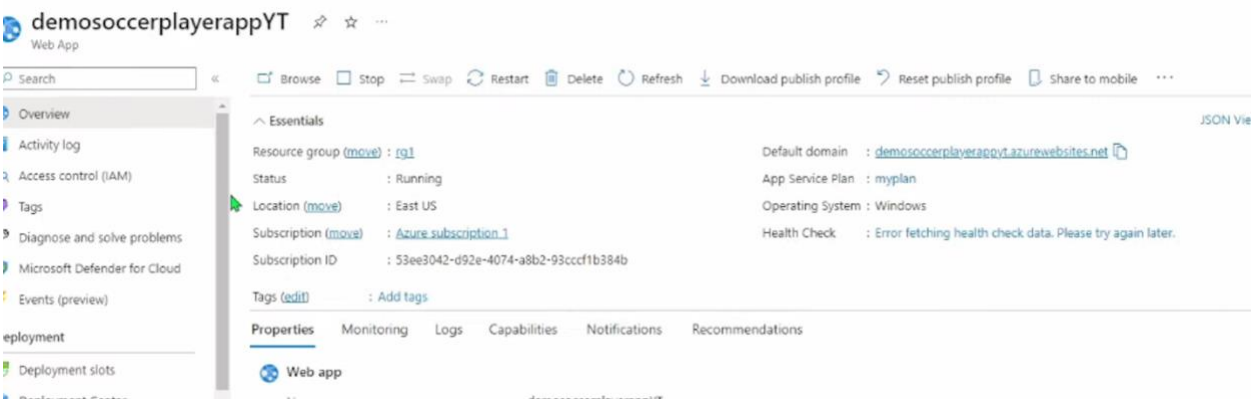
Edit branch policies



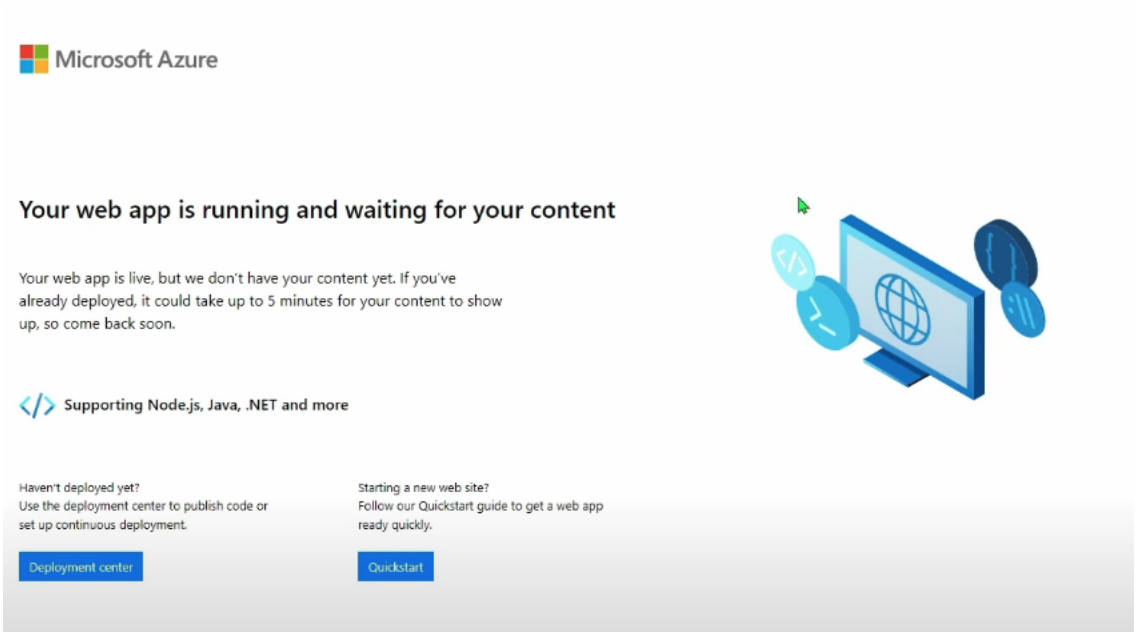
Go to App services and create web app



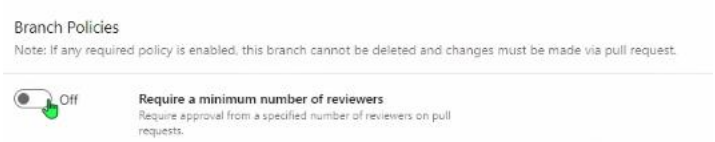
Click in default domain



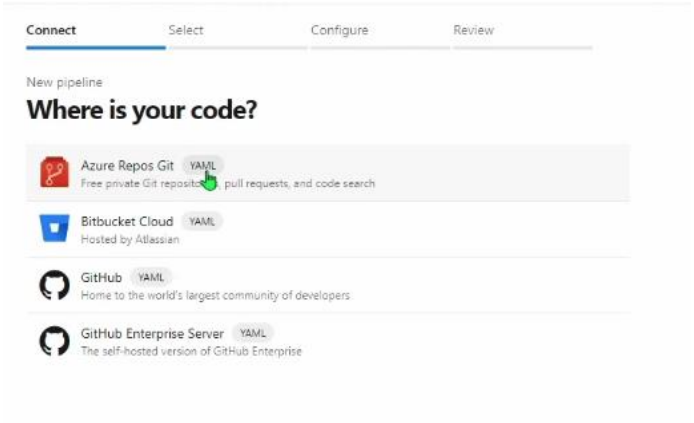
Webapp waiting for content:

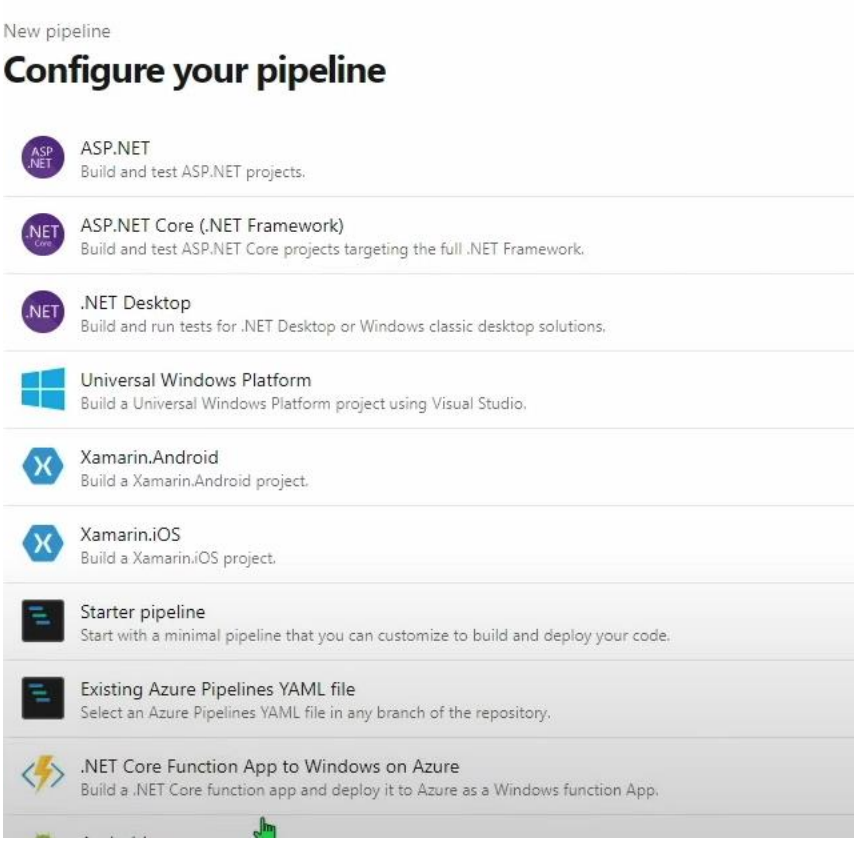


Put off

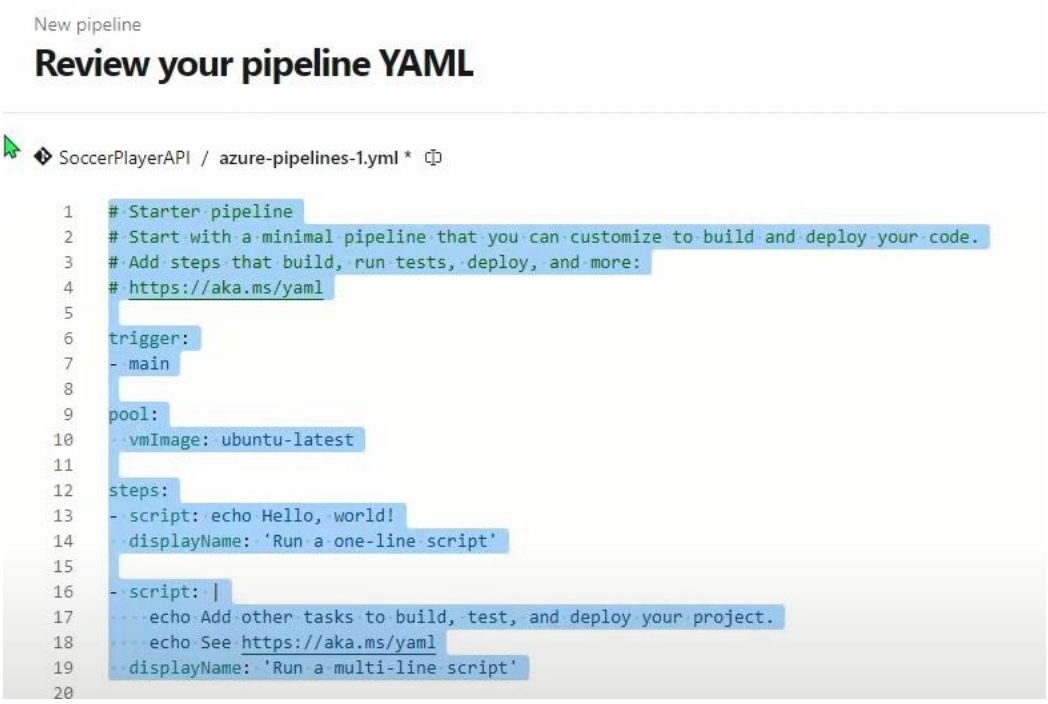



New pipeline
Filter pipelines


Connect
Select
Configure
Review
New pipeline
Where is your code?
Azure Repos Git YAML
Free private Git repositories, pull requests, and code search
Bitbucket Cloud YAML
Hosted by Atlassian
GitHub YAML
Home to the world's largest community of developers
GitHub Enterprise Server YAML
The self-hosted version of GitHub Enterprise


New pipeline
Configure your pipeline
ASP.NET
Build and test ASP.NET projects.
ASP.NET Core (.NET Framework)
Build and test ASP.NET Core projects targeting the full .NET Framework.
.NET Desktop
Build and run tests for .NET Desktop or Windows classic desktop solutions.
Universal Windows Platform
Build a Universal Windows Platform project using Visual Studio.
Xamarin.Android
Build a Xamarin.Android project.
Xamarin.iOS
Build a Xamarin.iOS project.
Starter pipeline
Start with a minimal pipeline that you can customize to build and deploy your code.
Existing Azure Pipelines YAML file
Select an Azure Pipelines YAML file in any branch of the repository.
.NET Core Function App to Windows on Azure
Build a .NET Core function app and deploy it to Azure as a Windows function App.

Starter pipeline:

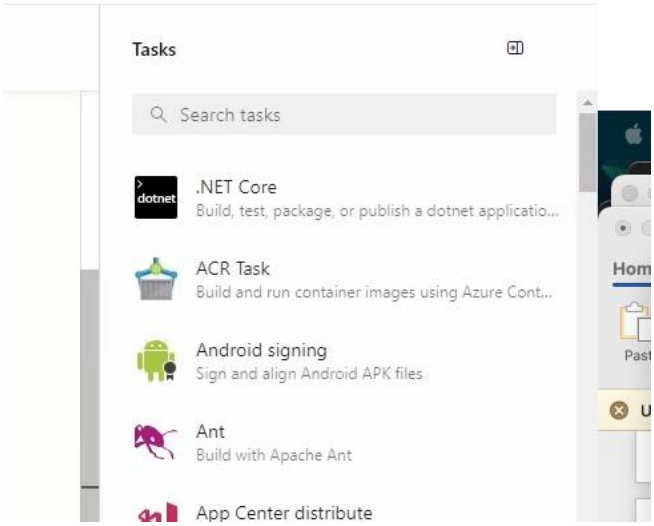


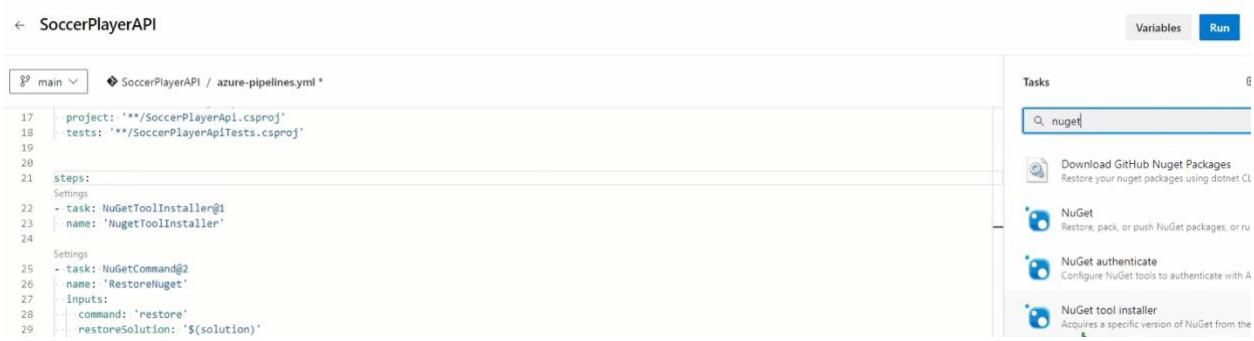
SoccerPlayerAPI

main / SoccerPlayerAPI / azure-pipelines.yml

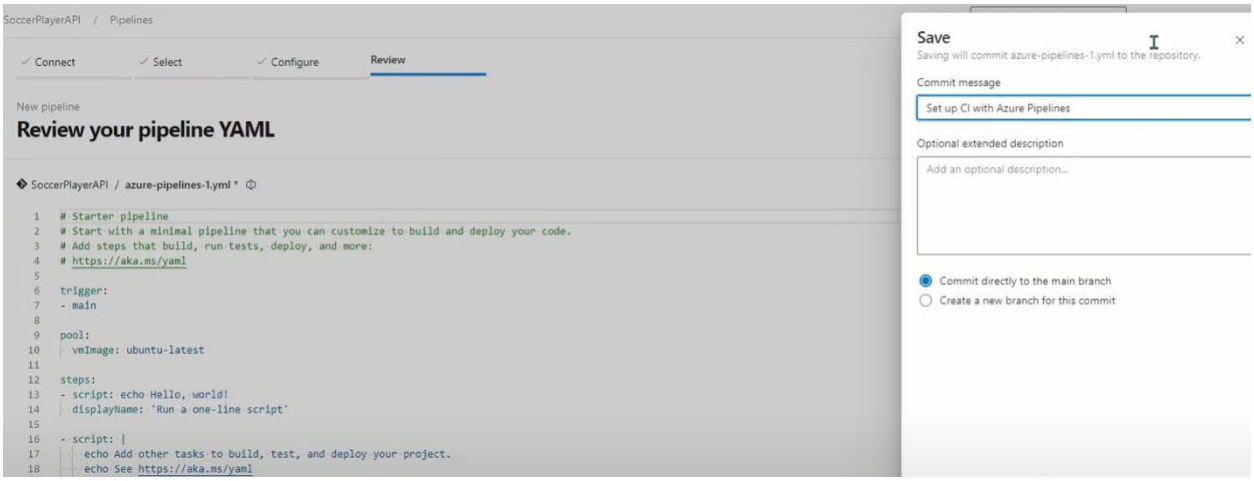
```yaml
# ASP.NET Core
# Build and test ASP.NET Core projects targeting .NET Core.
# Add steps that run tests, create a NuGet package, deploy, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/dotnet-core

trigger:
  branches:
    include:
      - dev

pool:
  vmImage: 'windows-latest'

variables:
  buildConfiguration: 'Release'
  solution: '**/SoccerPlayerApi.sln'
  project: '**/SoccerPlayerApi.csproj'
  tests: '**/SoccerPlayerApiTests.csproj'


steps:
Settings
- task: NuGetToolInstaller@1
  name: 'NugetToolInstaller'

Settings
```

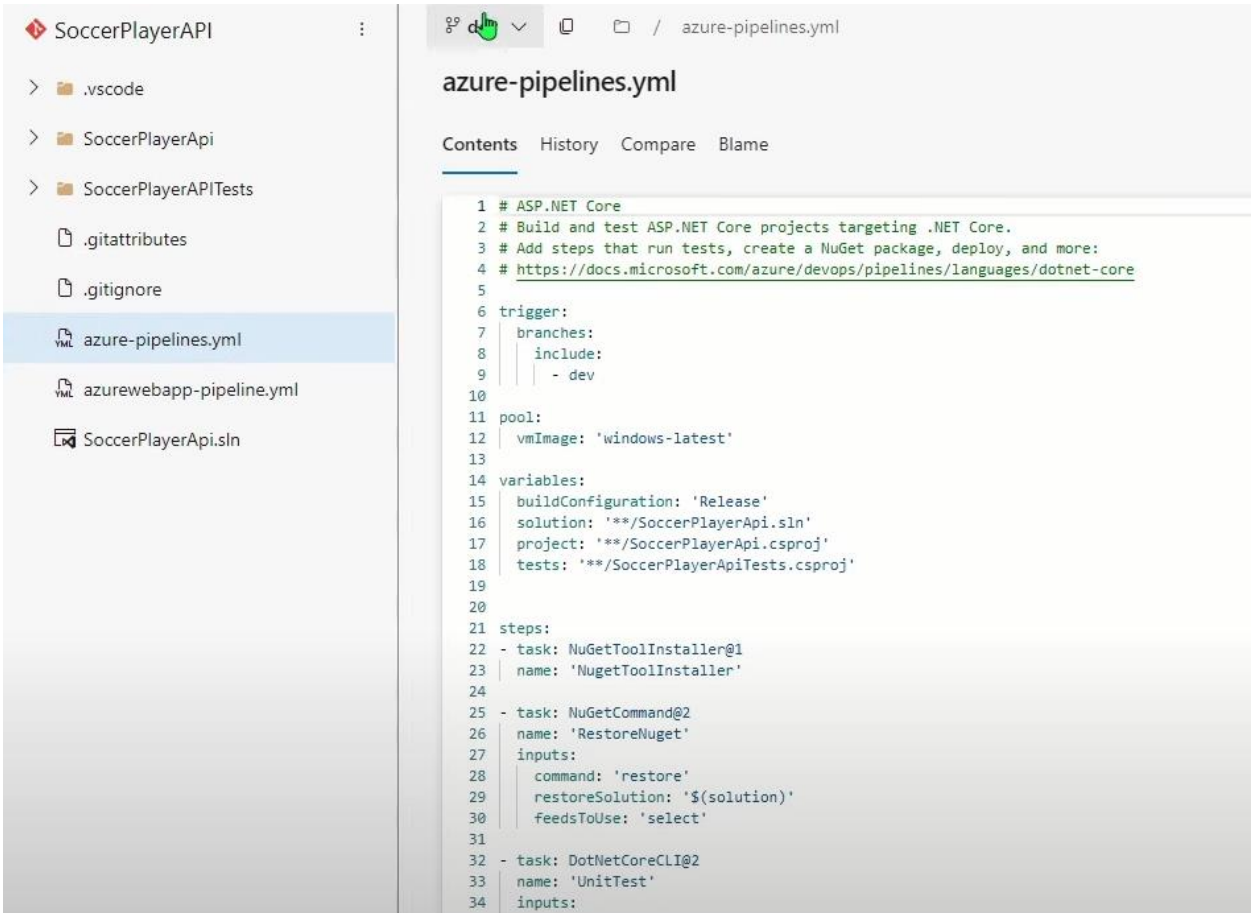
You can use templates:




SoccerPlayerAPI
Variables
Run
main
SoccerPlayerAPI / azure-pipelines.yml *
project: '**/SoccerPlayerApi.csproj'
tests: '**/SoccerPlayerApiTests.csproj'
steps:
Settings
- task: NuGetToolInstaller@1
name: 'NugetToolInstaller'
Settings
- task: NuGetCommand@2
name: 'RestoreNuget'
inputs:
command: 'restore'
restoreSolution: '$(solution)'
Tasks
nuget
Download GitHub Nuget Packages
NuGet
NuGet authenticate
NuGet tool installer

Save and run

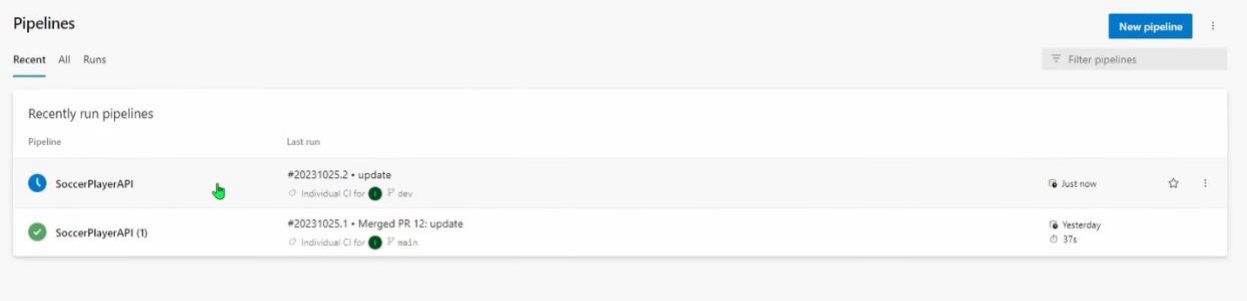

SoccerPlayerAPI / Pipelines
Connect
Select
Configure
Review
New pipeline
Review your pipeline YAML
SoccerPlayerAPI / azure-pipelines-1.yml *
Starter pipeline
Start with a minimal pipeline that you can customize to build and deploy your code.
Add steps that build, run tests, deploy, and more:
https://aka.ms/yaml
trigger:
- main
pool:
vmImage: ubuntu-latest
steps:
- script: echo Hello, world!
displayName: 'Run a one-line script'
- script: |
echo Add other tasks to build, test, and deploy your project.
echo See https://aka.ms/yaml
Save
Saving will commit azure-pipelines-1.yml to the repository.
Commit message
Set up CI with Azure Pipelines
Optional extended description
Add an optional description...
Commit directly to the main branch
Create a new branch for this commit

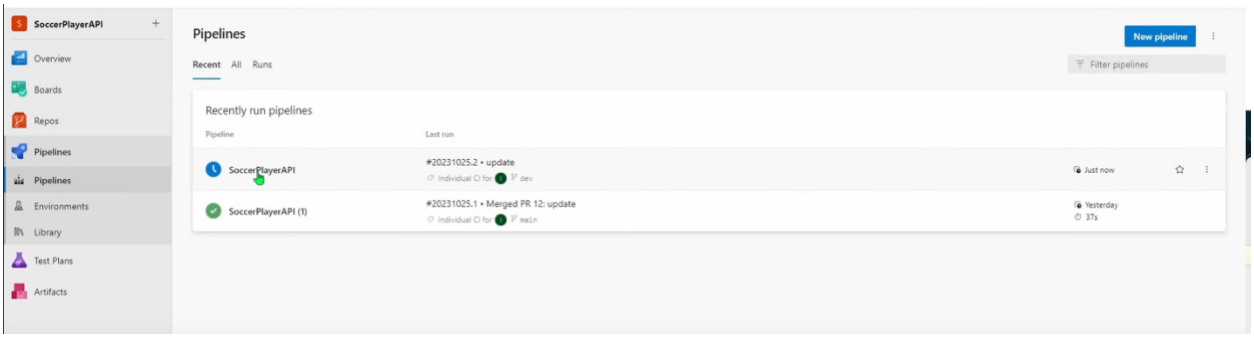


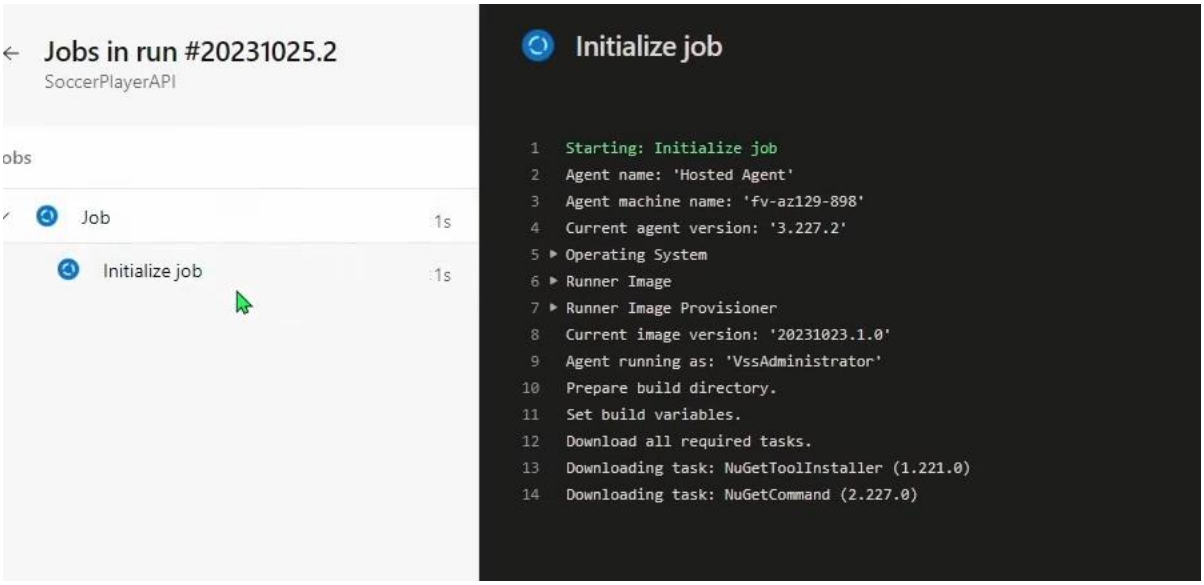
Do a change, from GetAllPlayers to GetPlayers:

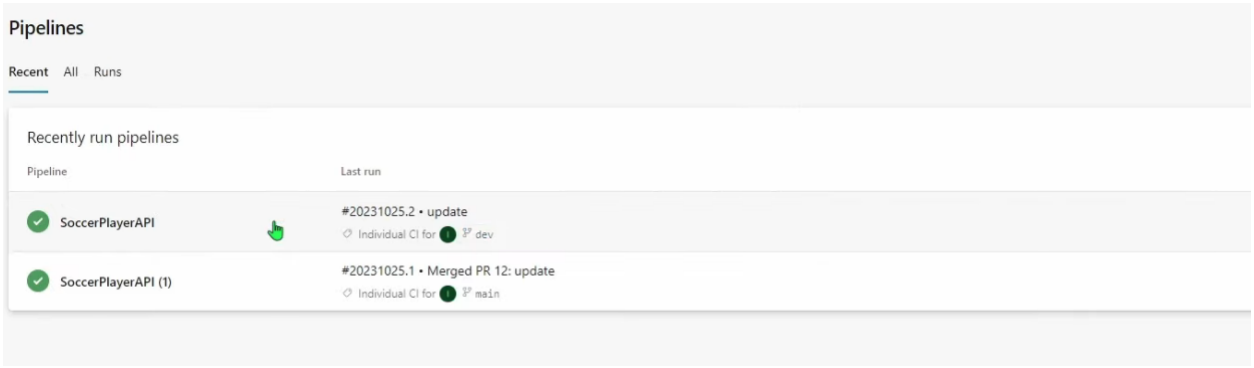
```
erApi                                                    SoccerPlayerApi.Controllers.PlayersController
10      {
11          private readonly IPlayerRepository _playerRepository;
12
13          public PlayersController(IPlayerRepository playerRepository)
14          {
15              _playerRepository = playerRepository;
16          }
17
18
19          [HttpGet("GetPlayers")]
20          public async Task<IActionResult> GetPlayers()
21          {
22              var x = await _playerRepository.GetPlayers();
23
24              return Ok(x);
25          }
26
27          [HttpPost("Post")]
28          public async Task<IActionResult> Post([FromBody] Player player)
29          {
30              var players = await _playerRepository.CreatePlayer(player);
31              return Ok(players);
32          }
33
34
35          [HttpPut("Put")]
36          public async Task<IActionResult> Put([FromBody] Player player)
37          {
38              var players = await _playerRepository.EditPlayer(player);
39              return Ok(players);
40          }
41
42
            [HttpDelete("Delete/{id}")]
```

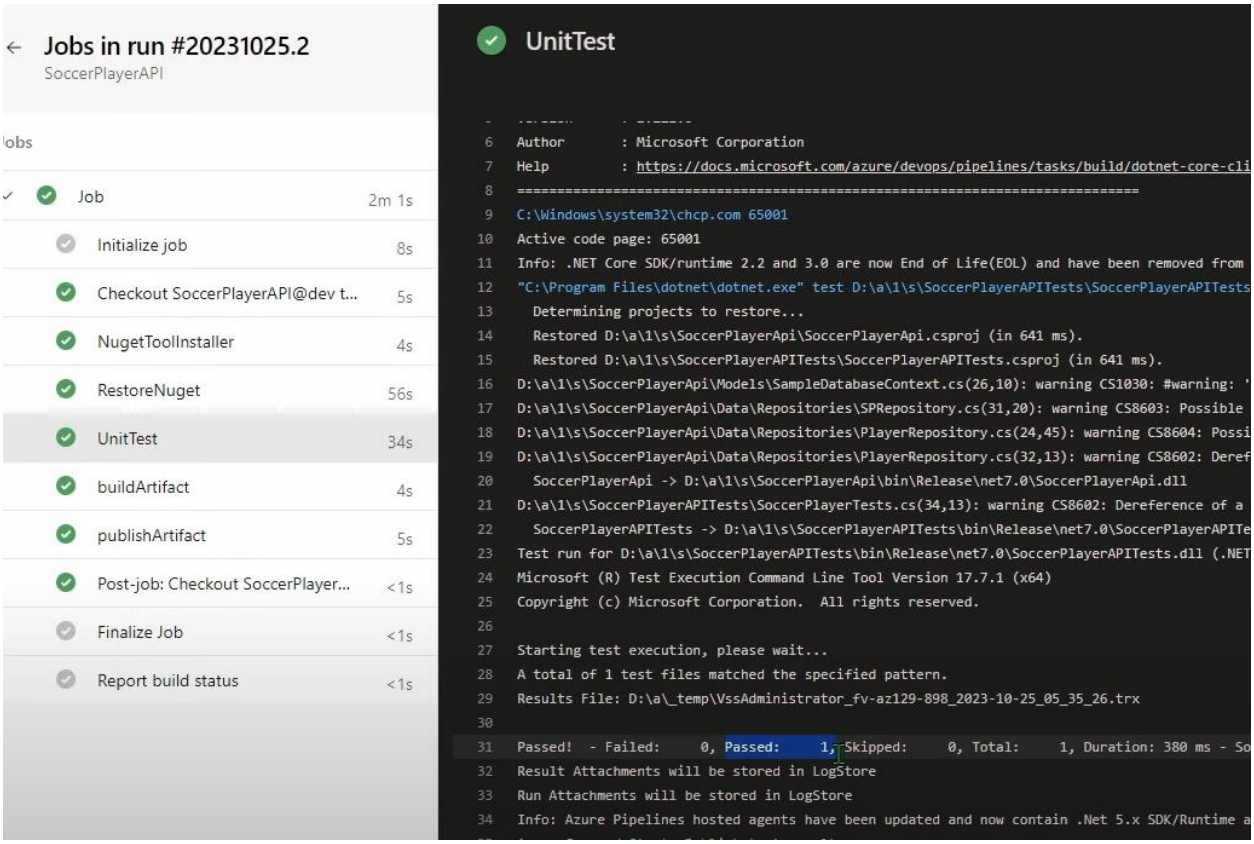
Push it



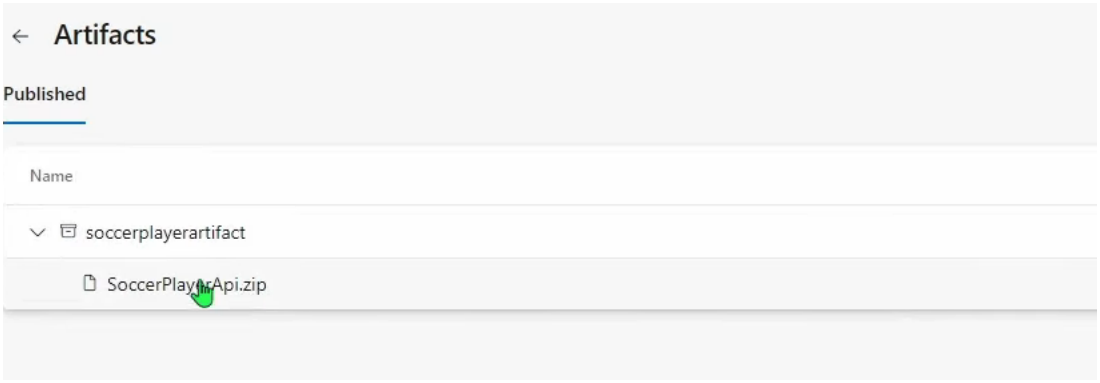

SoccerPlayerAPI
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Pipelines
New pipeline
Recent All Runs
Filter pipelines
Recently run pipelines
Pipeline
Last run
SoccerPlayerAPI
#20231025.2 • update
Individual CI for dev
Just now
SoccerPlayerAPI (1)
#20231025.1 • Merged PR 12: update
Individual CI for main
Yesterday
37s


Jobs in run #20231025.2
SoccerPlayerAPI
obs
Job
1s
Initialize job
1s
Initialize job
1 Starting: Initialize job
2 Agent name: 'Hosted Agent'
3 Agent machine name: 'fv-az129-898'
4 Current agent version: '3.227.2'
5 Operating System
6 Runner Image
7 Runner Image Provisioner
8 Current image version: '20231023.1.0'
9 Agent running as: 'VssAdministrator'
10 Prepare build directory.
11 Set build variables.
12 Download all required tasks.
13 Downloading task: NuGetToolInstaller (1.221.0)
14 Downloading task: NuGetCommand (2.227.0)

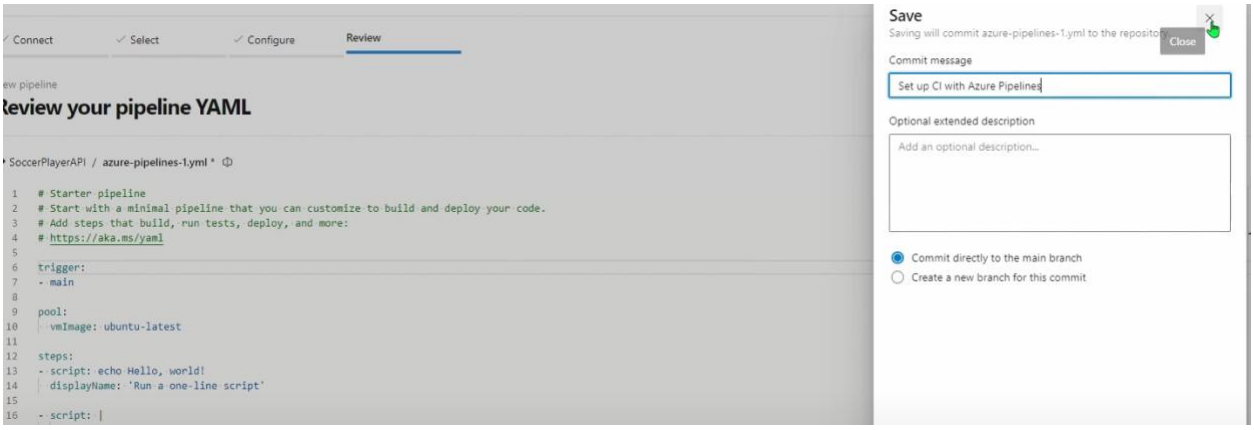

Pipelines
Recent All Runs
Recently run pipelines
Pipeline
Last run
SoccerPlayerAPI
#20231025.2 • update
Individual CI for dev
SoccerPlayerAPI (1)
#20231025.1 • Merged PR 12: update
Individual CI for main

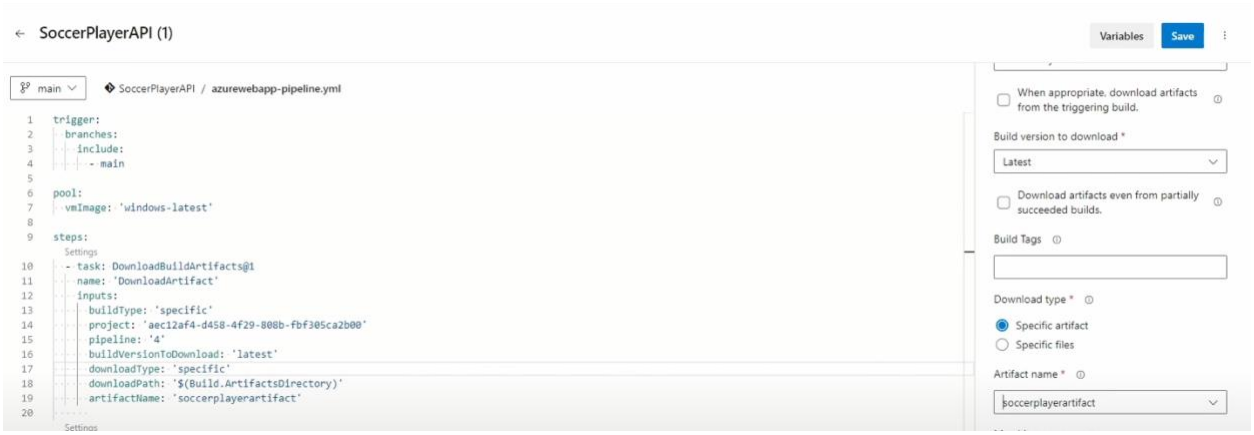


1 artifact created:

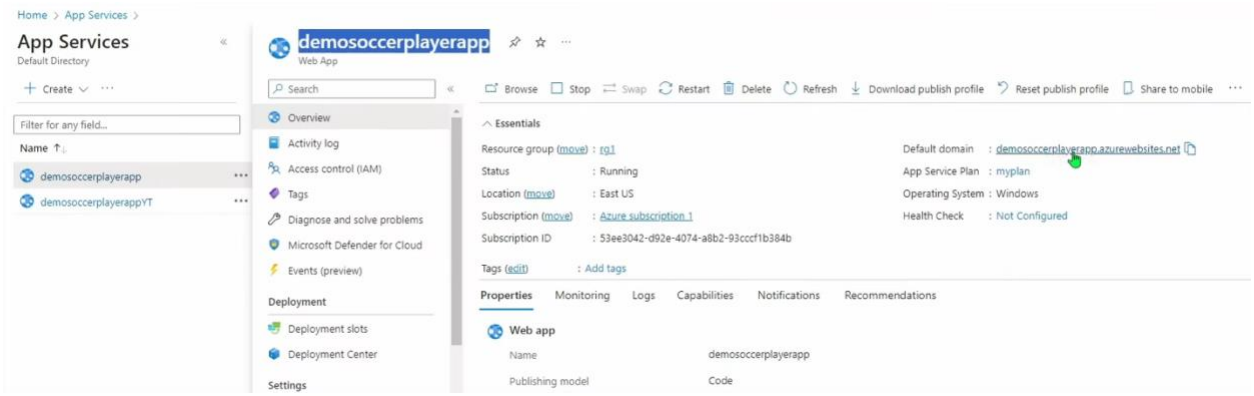


Create a new pipeline:





Click in default domain and we see the app:




Swagger
Select a definition
SoccerPlayerApi v1
SoccerPlayerApi
1.0
OAS3
https://demosoccerplayerapp.azurewebsites.net/swagger/v1/swagger.json
Players
GET
/api/Players/GetAllPlayers
POST
/api/Players/Post
PUT
/api/Players/Put
DELETE
/api/Players/Delete/{id}