



Publish Code Coverage to SonarCloud

.NET

SonarCloud is a cloud-based code analysis service designed to detect code quality issues in 25 different programming languages, continuously ensuring the maintainability, reliability and security of your code.

[HTTPS://SONARCLOUD.IO/DOCUMENTATION](https://sonarcloud.io/documentation)

First Steps

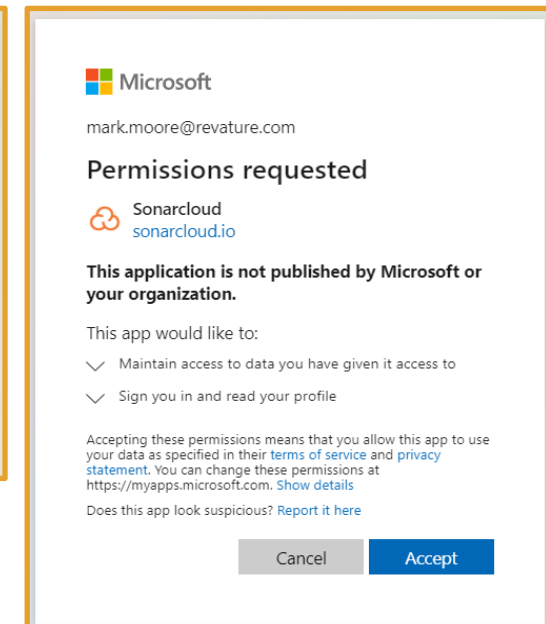
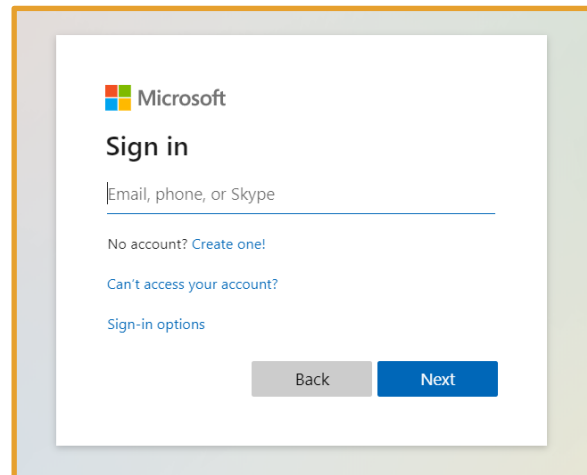
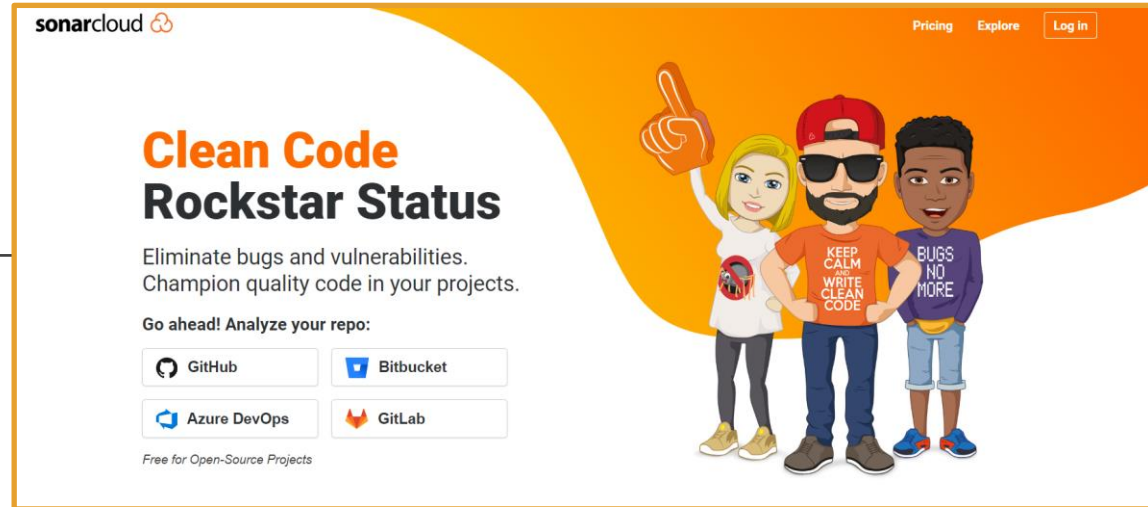
<https://sonarcloud.io/>

First, create a pipeline that successfully builds and deploys to your Website.

Next, log in to www.sonarcloud.io.

(Important) Use your Azure DevOps credentials.

If asked, accept any request for permission or access by Sonarcloud.

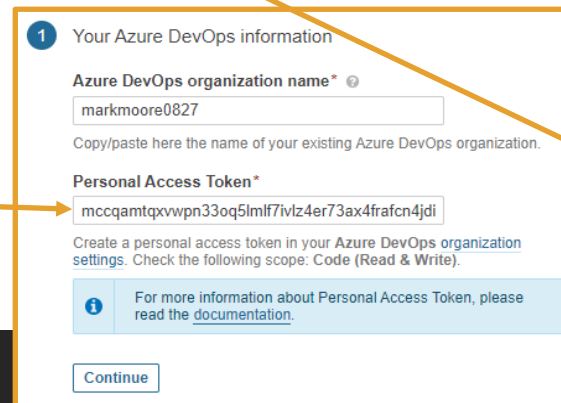
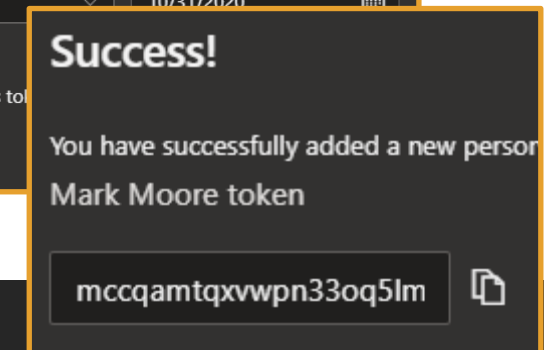
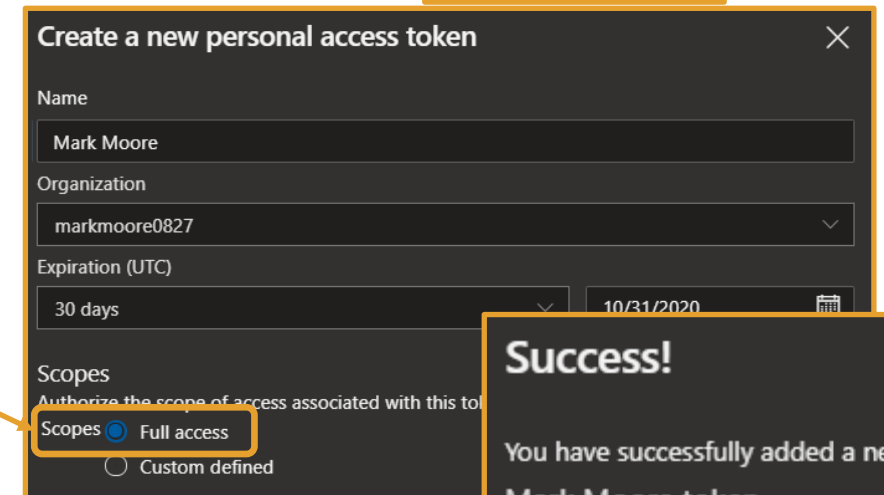
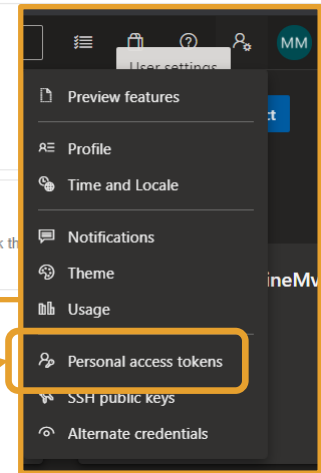
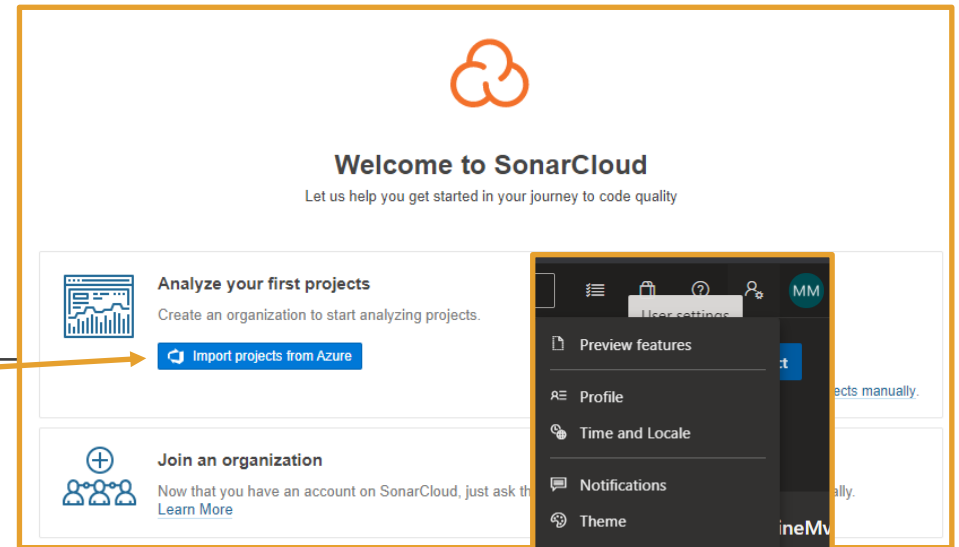


Create SonarCloud Organization

1. Create an organization
2. Enter the correct name of your Azure DevOps Organization and a Personal Access Token (P.A.T.).

1. Log into your Azure Account.
2. In the upper right, click on **User Settings > Personal Access tokens**.
3. Click **'New Token'**.
4. Enter your name and select **'Full Access'**.
5. Select **'Create'**.

6. Copy your P.A.T.
3. Enter the P.A.T.
4. Click **'Continue'**




Create SonarCloud Organization

1. Click 'Continue' to
Import your
organization details

2. Select 'Free plan'

3. Click "Create
Organization"

2 Import organization details

Import  markmoore0827 into a SonarCloud organization

Key*

Up to 255 characters. Only lower-case letters (a to z), digits or dashes are permitted. A dash cannot occur as a leading or trailing character.

[Add additional info](#) ▼

[Continue](#)

3 Choose a plan

☒ Free plan €0

All projects you analyze will be public.
Anyone can browse source code.

☐ Paid plan

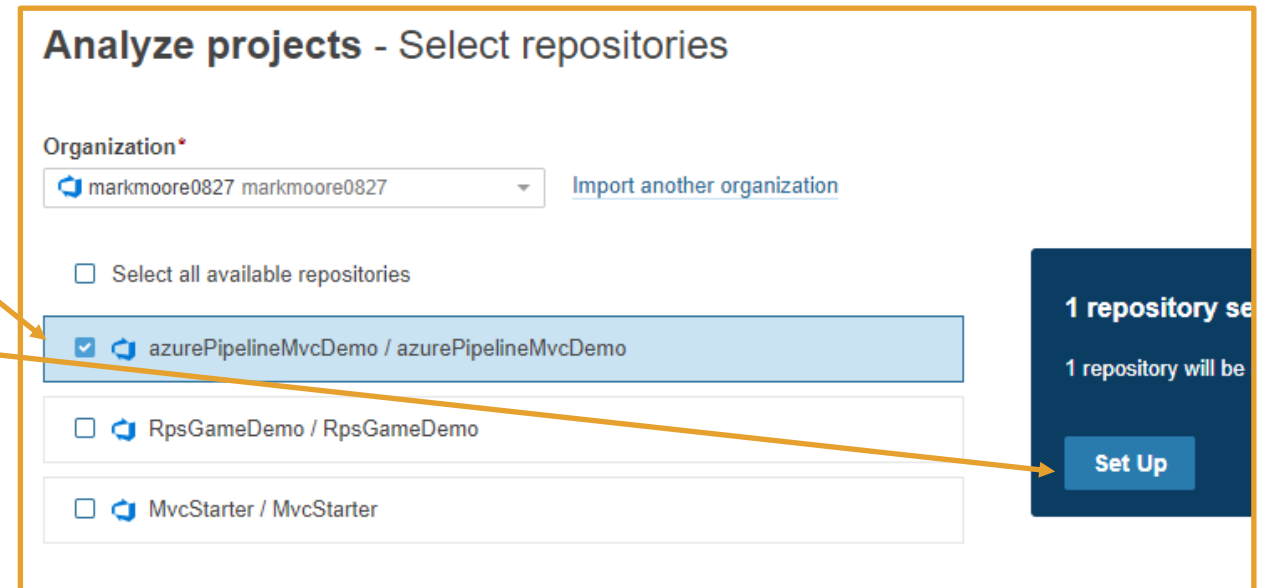
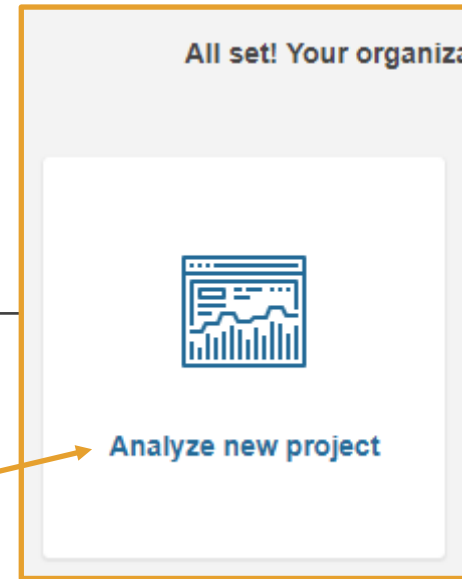
- ✓ Unlimited private projects
- ✓ Strict control over visibility
- ✓ No commitments, cancel anytime
- ✓ 14 days free trial.

[Learn more](#)

[Create Organization](#)

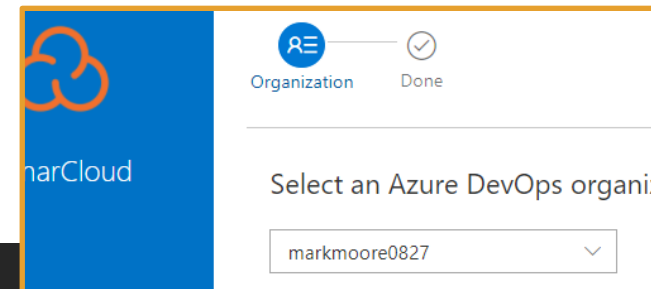
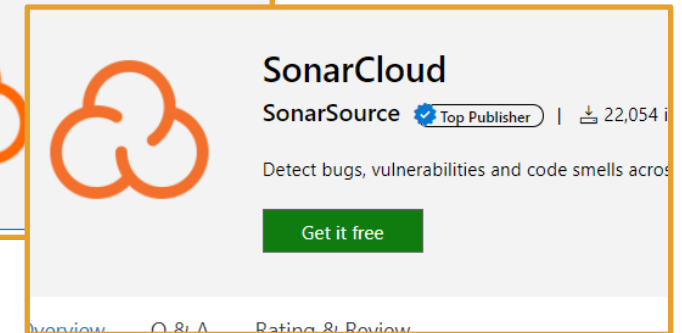
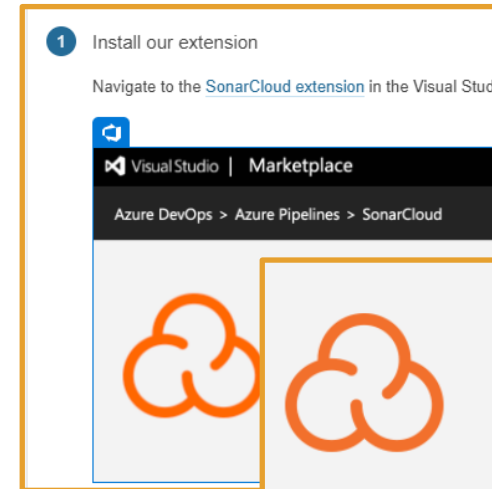
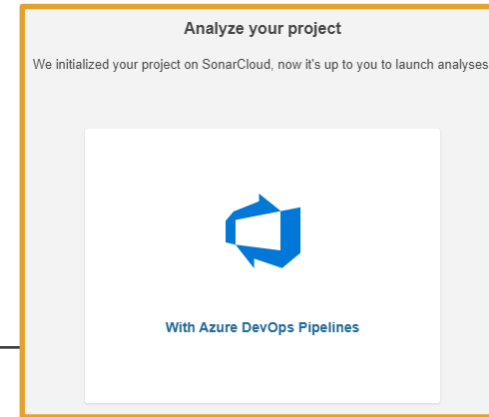
Analyse a New Project

4. Click 'analyse new project'.
5. Choose the pipeline you want to analyse.
6. Click 'Set Up'.



Configure Code Analysis

7. Click on 'With Azure DevOps Pipelines'
8. Click to download the SonarCloud Extension from the Visual Studio Marketplace.
9. Make sure to choose the correct organization when installing in the Marketplace.



Configure Azure Pipeline

2 Configure Azure Pipeline

What option best describes your build?

Maven

Gradle

.NET

Other (for JS, TS, Go, Python, PHP, ...)



Create a new pipeline

Follow the steps on Azure to initialize your pipeline and link it to your repository.

Add a new SonarCloud Service Endpoint

1 Go [to Project settings > Service connections](#)

2 Add a new service connection of the type SonarCloud

3 Use this token: `f3cf46527df3aec0546d71961a7358d9ecfcfb29`  

4 Click on **Verify** to check that everything is linked correctly.

Add the following three tasks to your pipeline


1. Prepare Analysis Configuration

1 Select the SonarCloud endpoint.

2 Select the SonarCloud organization `markmoore0827` 

3 In Choose the way to run the analysis, select **Integrate with MSBuild**.

4 In the Project Key field, enter `markmoore0827_azurePipelineMvcDemo` 

5 In the Project Name field, enter `azurePipelineMvcDemo` 



Please ensure this task runs before your build step.

2. Run Code Analysis



This task needs to run after your build step.

The steps on the following pages will guide you through setup on the *Azure DevOps* side.

See the Step-By-Step on the following slides.

2. Run Code Analysis



This task needs to run after your build step.

3. Publish Quality Gate Result

This task is not mandatory but will allow you to decorate your Pull Request.

If you plan not to use such a feature, you can omit it. Be aware that this task may increase your build time.

Enable the pipeline on push

Enable [Continuous Integration for your pipeline](#) to make sure your code will be analysed after every update.

Enable the PR Decoration

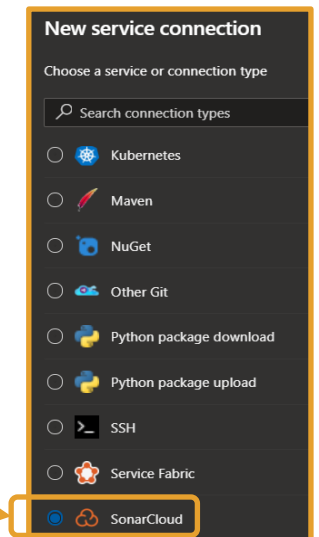
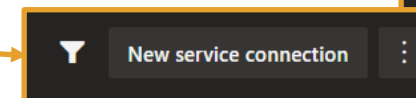
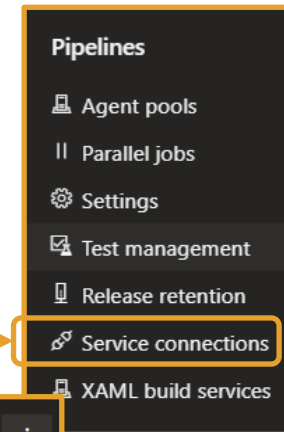
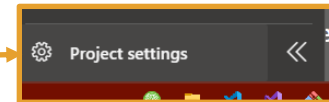
Enable [PR trigger](#) to make sure your code will be analysed after every update.

Add a new SonarCloud Service Endpoint

1. In your Azure DevOps account, click on the project you will be adding the Code Analysis to.
2. Click Project settings at the bottom left of your project home page.
3. Click the 'back' arrow next to Project details.
4. Click Service Connections under Pipelines.
5. Click 'New service connection' in the upper right.
6. Select 'SonarCloud' in the 'New Service Connection' List
7. Click Next.

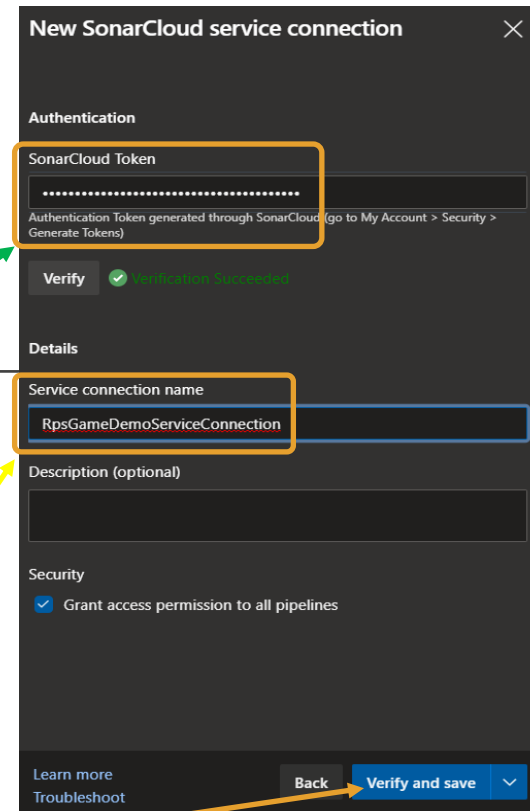
Add a new SonarCloud Service Endpoint

- 1 Go to **Project settings > Service connections**
- 2 Add a new service connection of the type **SonarCloud**
- 3 Use this token: `f3cf46527df3aec0546d71961a7358d9ecfcfb29`
- 4 Click on **Verify** to check that everything is linked correctly.



New SonarCloud Connection

- Paste in the Personal Access Token under 'SonarCloud Token'.
- Create a name for your connection under 'Service connection name'.
- Click 'Verify and Save'.
- Verify the connection was successful.



The screenshot shows the 'New SonarCloud service connection' form. A green arrow points from the first bullet point to the 'SonarCloud Token' field. A yellow arrow points from the second bullet point to the 'Service connection name' field. An orange arrow points from the third bullet point to the 'Verify and save' button. The form includes sections for Authentication, Details, and Security.

New SonarCloud service connection

Authentication

SonarCloud Token
Authentication Token generated through SonarCloud (go to My Account > Security > Generate Tokens)

Verify ✓ Verification Successful

Details

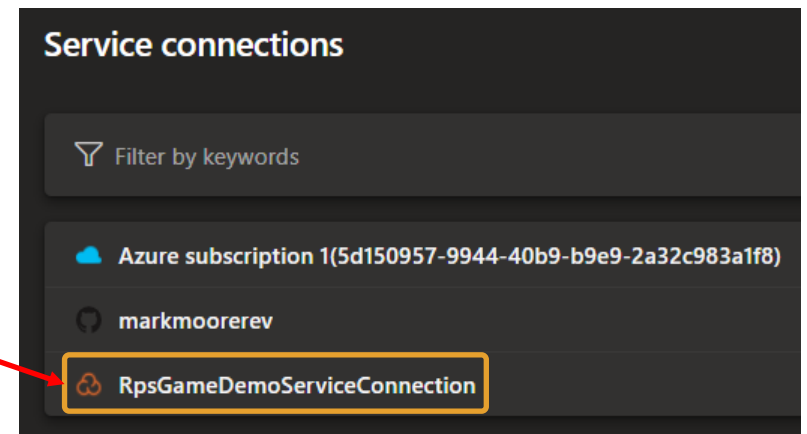
Service connection name
RpsGameDemoServiceConnection

Description (optional)

Security

☒ Grant access permission to all pipelines

Learn more Troubleshoot Back Verify and save



Add publish task to your pipeline YAML

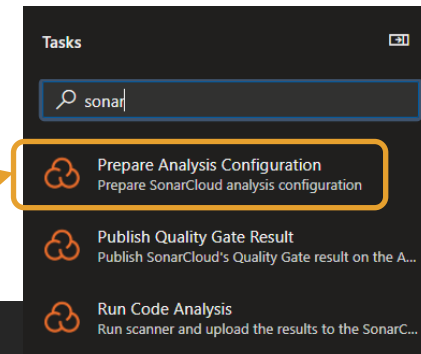
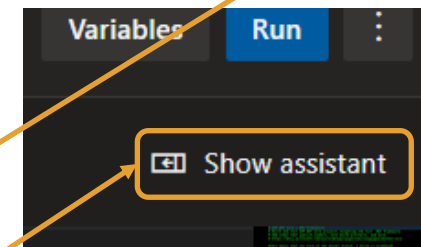
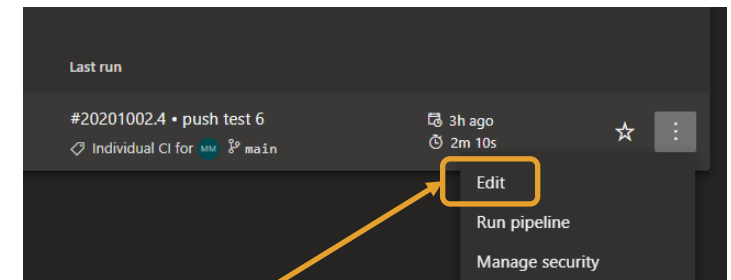
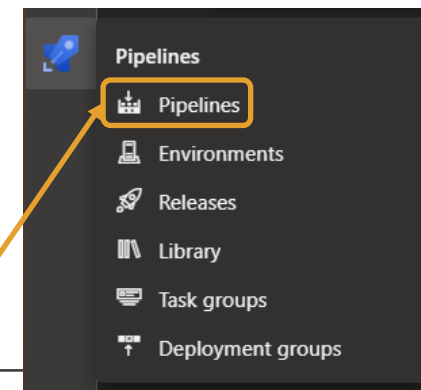
Add the following three tasks to your pipeline.

1. Prepare Analysis Configuration

- 1 Select the SonarCloud endpoint.
- 2 Select the SonarCloud organization `markmoore0827`
- 3 In Choose the way to run the analysis, select **Integrate with MSBuild**.
- 4 In the Project Key field, enter `markmoore0827_azurePipelineMvcDemo`
- 5 In the Project Name field, enter `azurePipelineMvcDemo`

Please ensure this task runs before your build step.

1. Click Pipelines under the Rocket (Pipelines) avatar in the left column.
2. Click Edit under the pipeline you want to add code analysis to.
3. Click 'Show Assistant' in the upper left.
4. Select 'Prepare Analysis Configuration'.



Add Code Analysis configuration task to your pipeline YAML

1. Prepare Analysis Configuration

- 1 Select the SonarCloud endpoint.
- 2 Select the SonarCloud organization `markmoore0827`
- 3 In Choose the way to run the analysis, select **Integrate with MSBuild**.
- 4 In the Project Key field, enter `markmoore0827_azurePipelineMvcDemo`
- 5 In the Project Name field, enter `azurePipelineMvcDemo`



Please ensure this task runs before your build step.

You don't have to change anything under 'Advanced'.
Place the cursor on the line above your 'build' Task.
Click 'Add'

Prepare Analysis Configuration

SonarCloud Service Endpoint * ⓘ
RpsGameDemoServiceConnection

Organization * ⓘ
markmoore0827 (markmoore0827)

Choose the way to run the analysis * ⓘ
☒ Integrate with MSBuild
☐ Integrate with Maven or Gradle
☐ Use standalone scanner

Project Key * ⓘ
markmoore0827_azurePipelineMvcDemo

Project Name ⓘ
azurePipelineMvcDemo

Project Version ⓘ
1.0

```
Settings
- task: SonarCloudPrepare@1
  inputs:
    SonarCloud: 'RpsGameDemoServiceConnection'
    organization: 'markmoore0827'
    scannerMode: 'MSBuild'
    projectKey: 'markmoore0827_azurePipelineMvcDemo'
    projectName: 'azurePipelineMvcDemo'

Settings
- task: DotNetCoreCLI@2
  displayName: Build
  inputs:
```

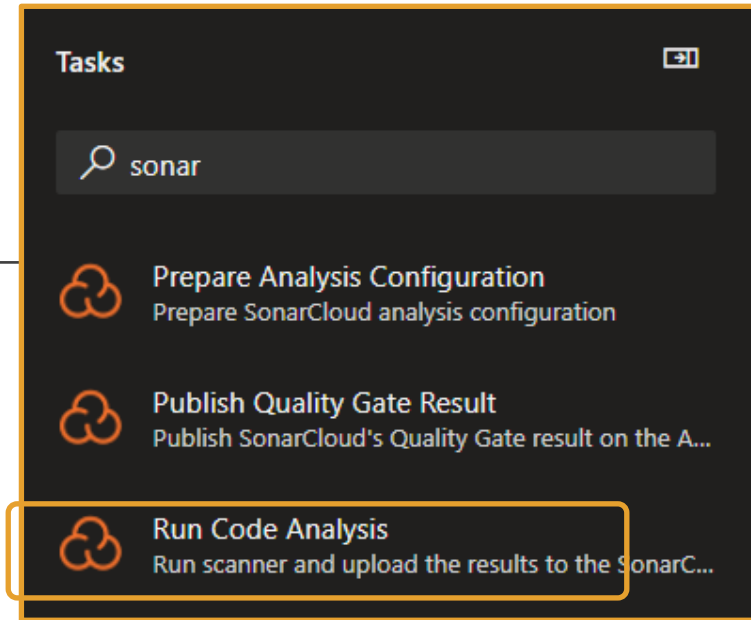
Add Code Analysis to your YAML

2. Run Code Analysis



This task needs to run after your build step.

1. Click 'Show Assistant' to return to the Assistant.
2. Make sure the cursor is immediately below the 'build' task.
3. Click 'Run Code Analysis'.



```
- task: DotNetCoreCLI@2
  inputs:
    command: 'test'
    projects: 'Demos/JavaScriptWithDotnetAPI/memesaver/memesaver.sln'
    #tell the compiler to collect code coverage. This artifact will be saved in a default location
    arguments: '--configuration $(buildConfiguration) --collect "Code Coverage"'
    testRunTitle: 'Dotnet tests running'
    workingDirectory: 'Demos/JavaScriptWithDotnetAPI/memesaver'

Settings
- task: PublishCodeCoverageResults@1
  inputs:
    #tell the compiler what code coverage tool you used.
    codeCoverageTool: 'Cobertura'
    #this is the path to the default location of the coverage artifact file.
    summaryFileLocation: '**/cobertura/coverage.xml'
```

Complete Pipeline YAML file (1 / 2)

```
6 trigger:
7   - branches:
8       - include:
9           - main
10  #the below is required only if you have multiple .sln files in your repo.
11  - paths:
12      - include:
13          - Demos/JavaScriptWithDotnetAPI
14
15
16 pool:
17   - vmImage: 'windows-latest'
18
19 variables:
20   - solution: 'Demos/JavaScriptWithDotnetAPI/memesaver/memesaver.sln'
21   - buildPlatform: 'Any CPU'
22   - buildConfiguration: 'Release'
23
24
25 #this installs the newest SDK for this build
26 steps:
27   Settings
28   - task: UseDotNet@2
29     displayName: 'Install .NET Core SDK'
30     inputs:
31       - packageType: 'sdk'
32       - version: '5.0.x'
33       - #performMultiLevelLookup: true
34       - #includePreviewVersions: false #Required 'true' to use preview versions
```

```
35   Settings
36   - task: UseDotNet@2
37     displayName: 'Install .NET Core Runtime'
38     inputs:
39       - packageType: 'runtime'
40       - version: '2.x'
41       - #performMultiLevelLookup: true
42
43   Settings
44   - task: NuGetToolInstaller@1
45
46   Settings
47   - task: NuGetCommand@2
48     inputs:
49       - restoreSolution: '$(solution)'
50
51   #this must go before the 'build' task.
52   Settings
53   - task: SonarCloudPrepare@1
54     inputs:
55       - SonarCloud: '03012021BatchServiceToken1'
56       - organization: '03012021batch'
57       - scannerMode: 'MSBuild'
58       - projectKey: '03012021Batch_03012021BatchPipelineDemo'
59       - projectName: '03012021BatchPipelineDemo'
60
61   #this will restore and build the directory.
62   Settings
63   - task: DotNetCoreCLI@2
64     displayName: 'building'
65     inputs:
66       - command: 'build'
67       - #path to project .sln
68       - projects: 'Demos/JavaScriptWithDotnetAPI/memesaver/*.sln'
69       - #path to root of project
70       - workingDirectory: 'Demos/JavaScriptWithDotnetAPI/memesaver'
71       - arguments: '--configuration $(buildConfiguration)'
```


Complete Pipeline YAML file (2/2)

```
68
69 #
70 Settings
71 - task: DotNetCoreCLI@2
72   inputs:
73     command: 'test'
74     projects: 'Demos/JavaScriptWithDotnetAPI/memesaver/memesaver.sln'
75     #tell the compiler to collect code coverage. This artifact will be saved in a default location
76     arguments: '--configuration $(buildConfiguration) --collect "Code Coverage"'
77     testRunTitle: 'Dotnet tests running'
78     workingDirectory: 'Demos/JavaScriptWithDotnetAPI/memesaver'
79
80 Settings
81 - task: PublishCodeCoverageResults@1
82   inputs:
83     #tell the compiler what code coverage tool you used.
84     codeCoverageTool: 'Cobertura'
85     #this is the path to the default location of the coverage artifact file.
86     summaryFileLocation: '**/cobertura/coverage.xml'
87
88 Settings
89 - task: SonarCloudAnalyze@1
90
91 Settings
92 - task: SonarCloudPublish@1
93   inputs:
94     pollingTimeoutSec: '300'
95
96 Settings
97 - task: DotNetCoreCLI@2
98   displayName: 'Publishing'
99   inputs:
100     command: 'publish'
101     publishWebProjects: false
102     #modifyOutputPath: false
103     #workingDirectory: 'Demos/JavaScriptWithDotnetAPI/memesaver'
104     projects: '$(solution)'
105     zipAfterPublish: true
```

```
101
102 #publish to Azure AppService
103 Settings
104 - task: AzureRmWebAppDeployment@4
105   inputs:
106     ConnectionType: 'AzureRM'
107     azureSubscription: 'Azure subscription 1(5d150957-9944-40b9-b9e9-2a32c983a1f8)'
108     appType: 'webApp'
109     WebAppName: 'memesaver'
110     packageForLinux: '$(System.DefaultWorkingDirectory)/**/*.zip'
```

A Microsoft Docs Lab Tutorial

<https://azuredevopslabs.com/labs/vstsextend/sonarcloud//>