

# This is John Kear's ppt. Connecting Azure DevOps to Sonar Cloud with Code Coverage using Coverlet

Pre-requisite:

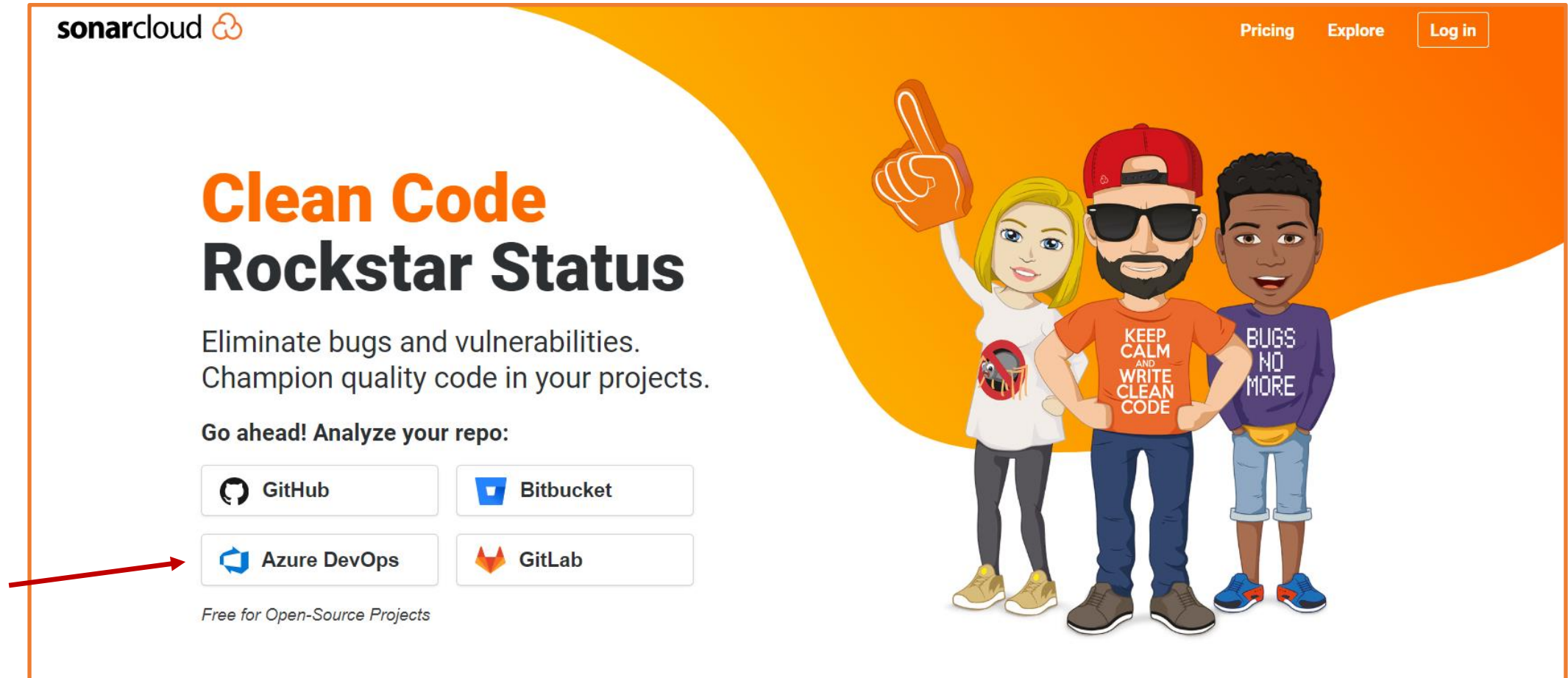
- 1) An Azure DevOps account
- 2) A GitHub account with repository

# Darius Vallejo example


- Darius Vallejo based his integration on the below tutorial.  
<https://azuredevopslabs.com/labs/vstsextend/sonarcloud//>

# Sonarcloud.io

## Get started using Azure DevOps



The image is a screenshot of the Sonarcloud.io website. The header features the Sonarcloud logo on the left and navigation links for 'Pricing', 'Explore', and 'Log in' on the right. The main content area has a large orange wave graphic on the right side. On the left, the text 'Clean Code Rockstar Status' is prominently displayed, followed by the tagline 'Eliminate bugs and vulnerabilities. Champion quality code in your projects.' and the instruction 'Go ahead! Analyze your repo:'. Below this, there are four buttons for different code hosting services: GitHub, Bitbucket, Azure DevOps, and GitLab. A red arrow points to the 'Azure DevOps' button. At the bottom left, it says 'Free for Open-Source Projects'. On the right side of the main content area, there is a cartoon illustration of three people: a woman with blonde hair pointing upwards, a man with a beard and sunglasses wearing a red cap, and a man with dark skin wearing a purple shirt that says 'BUGS NO MORE'. The man in the middle is wearing an orange shirt that says 'KEEP CALM AND WRITE CLEAN CODE'.



sonarcloud 



Pricing Explore Log in

## Clean Code Rockstar Status

Eliminate bugs and vulnerabilities.  
Champion quality code in your projects.

Go ahead! Analyze your repo:

 GitHub  Bitbucket

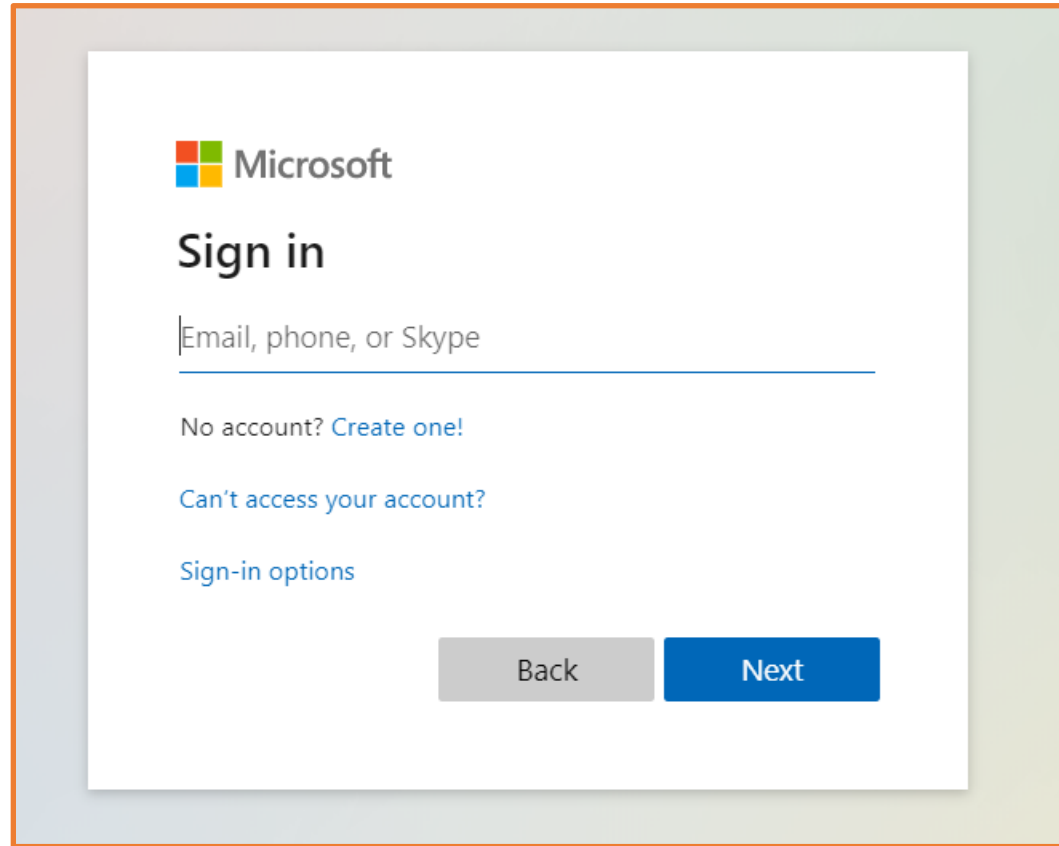
 Azure DevOps  GitLab


Free for Open-Source Projects

KEEP CALM AND WRITE CLEAN CODE

BUGS NO MORE

# Sign in with your Microsoft account

A screenshot of the Microsoft sign-in page. The page is white with a light gray border. At the top left is the Microsoft logo. Below it is the text "Sign in". There is a text input field with the placeholder text "Email, phone, or Skype". Below the input field are three links: "No account? Create one!", "Can't access your account?", and "Sign-in options". At the bottom right are two buttons: a gray "Back" button and a blue "Next" button.

 Microsoft

## Sign in

Email, phone, or Skype

No account? [Create one!](#)

[Can't access your account?](#)

[Sign-in options](#)

[Back](#) [Next](#)

## Create an organization

An organization is a space where a team or a whole company can collaborate accross many projects.

### 1 Enter your organization details

Key\*



Up to 255 characters. All chars must be lower-case letters (a to z), digits or dash (but dash can neither be trailing nor heading).

[Hide additional info](#) ▲

Display Name



Up to 255 characters

Avatar

Url of a small image that represents the organization (preferably 30px height).

Description

URL

[Continue](#)

### 2 Choose a plan

☒ Free plan

€0

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

☐ Paid plan

from €10

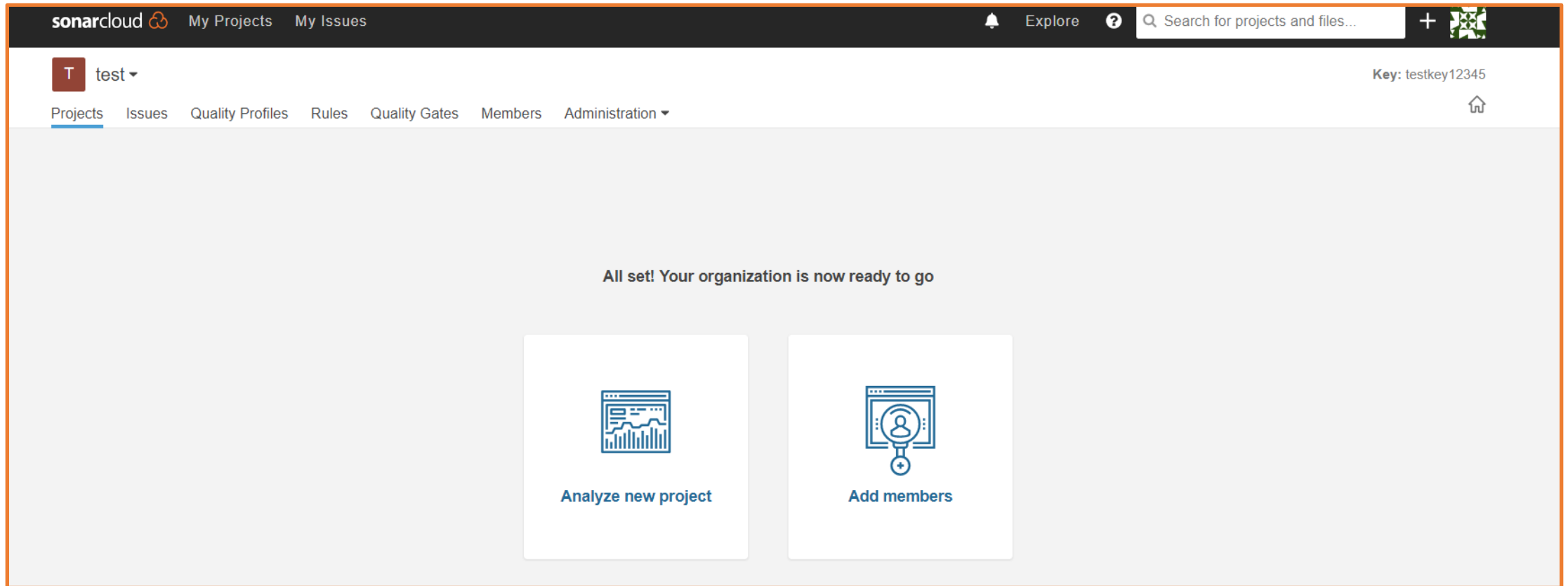
- ✓ Unlimited private projects
- ✓ Strict control over who can view your private data
- ✓ No commitments, cancel anytime
- ✓ 14 days free trial.

[Learn more](#)

[Create Organization](#)

- Create an organization
- Enter a key for your organization
- Click “Continue”
- Select “Free plan”
- Click “Create Organization”
- Click on “Create manually” when you see it after making the organization.
- DO NOT fill in the next part they give you.
- Go straight to generating the token in the next slide.

# Analyze a new project



# Analyze project setup

## Analyze projects - Set up manually

Organization\*

test testkey12345

[Create another organization](#)

Project key\* ?

thisIsMyProjKey123



Up to 400 characters. All letters, digits, dash, underscore, period or colon.

Display name\* ?

thisIsMyProjKey123



Up to 255 characters

☒ Public

Anyone will be able to browse your source code and see the result of your analysis.

☐ Private

Only members of the organization will be able to browse your source code and see the result of your analysis.

[Set Up](#)

Analyze private projects with our Paid Plan

from €10

- ✓ Unlimited private projects
- ✓ Strict control over who can view your private data
- ✓ No commitments, cancel anytime
- ✓ **14 days free trial.**

[Upgrade](#)

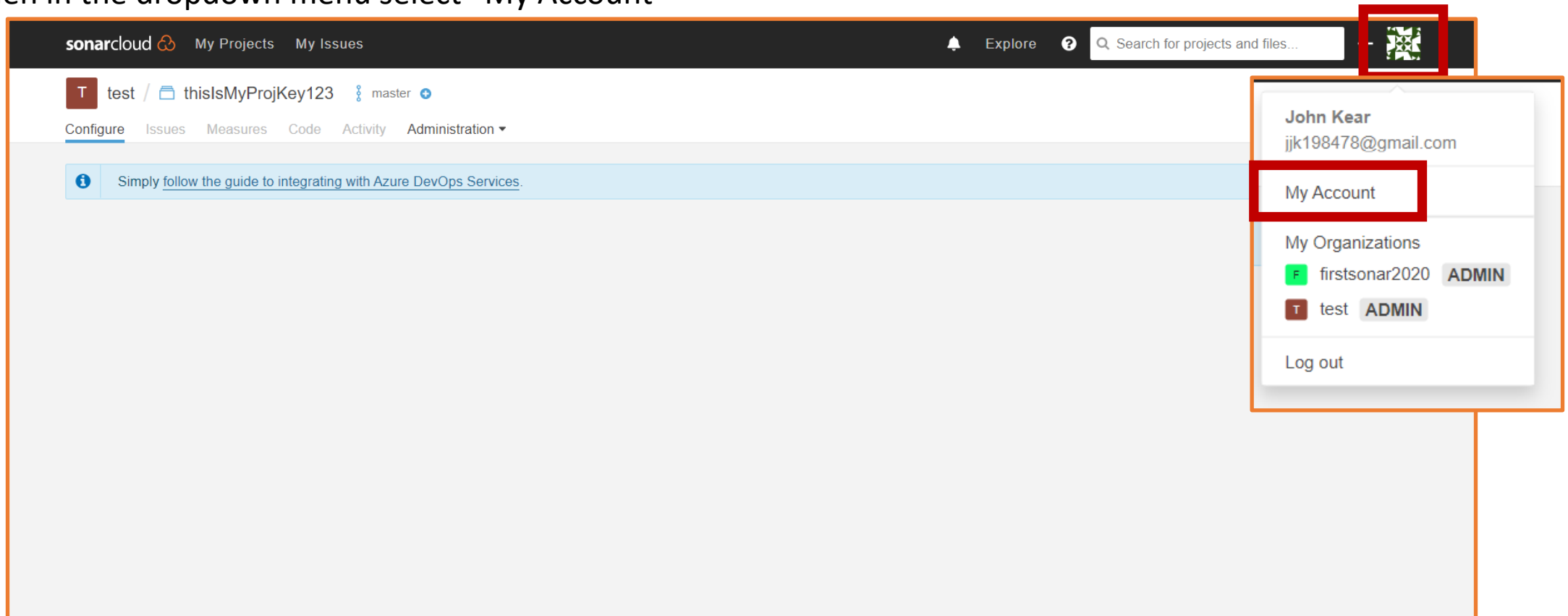
[Learn more](#)

# Generate Token

When configuring Sonar Cloud in Azure DevOps, a token for your Sonar Cloud project will be needed.

-Click on your account image in the top right of the screen

-Then in the dropdown menu select “My Account”



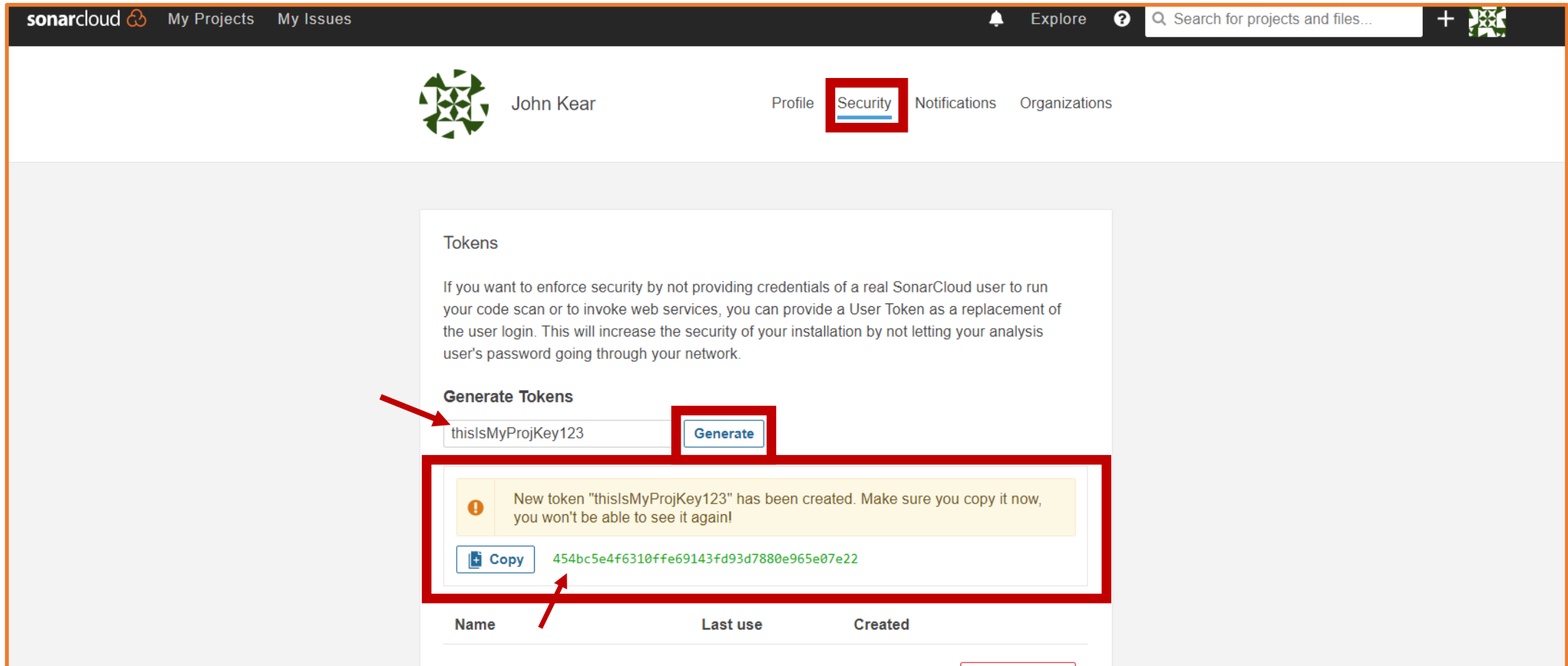


-Select “Security” in the header menu

-type the name of the project key we just created in the “Generate Tokens” input box

-click “Generate” this will generate a token for the specified project key

-be sure to copy and save this key somewhere, it will be used in future steps



The screenshot shows the SonarCloud interface. At the top, the header includes the SonarCloud logo, navigation links for 'My Projects' and 'My Issues', a search bar, and a user profile icon. Below the header, the user's name 'John Kear' is displayed next to a profile icon. A navigation menu contains 'Profile', 'Security' (highlighted with a red box), 'Notifications', and 'Organizations'. The main content area is titled 'Tokens' and contains explanatory text about user tokens. Below this, the 'Generate Tokens' section features an input field containing 'thisIsMyProjKey123' and a 'Generate' button (both highlighted with a red box). A red arrow points from the input field to the 'Generate' button. Below the button, a yellow notification box states: 'New token "thisIsMyProjKey123" has been created. Make sure you copy it now, you won't be able to see it again!'. This box contains a 'Copy' button and the token value '454bc5e4f6310ffe69143fd93d7880e965e07e22'. A red box highlights the entire notification area, and a red arrow points from the 'Copy' button to the token value. At the bottom, a table with columns 'Name', 'Last use', and 'Created' is partially visible.

**Tokens**

If you want to enforce security by not providing credentials of a real SonarCloud user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

**Generate Tokens**

thisIsMyProjKey123 **Generate**

**Copy** 454bc5e4f6310ffe69143fd93d7880e965e07e22

Name	Last use	Created
------	----------	---------

- Now that we have a new token, we
- From your SonarCloud home page, select the project

The screenshot shows the SonarCloud interface with the 'My Projects' tab selected. The left sidebar contains filters for Quality Gate, Reliability, Security, Maintainability, and Coverage. The main area displays a list of projects. The project 'test / thisIsMyProjKey123' is highlighted with a red box.

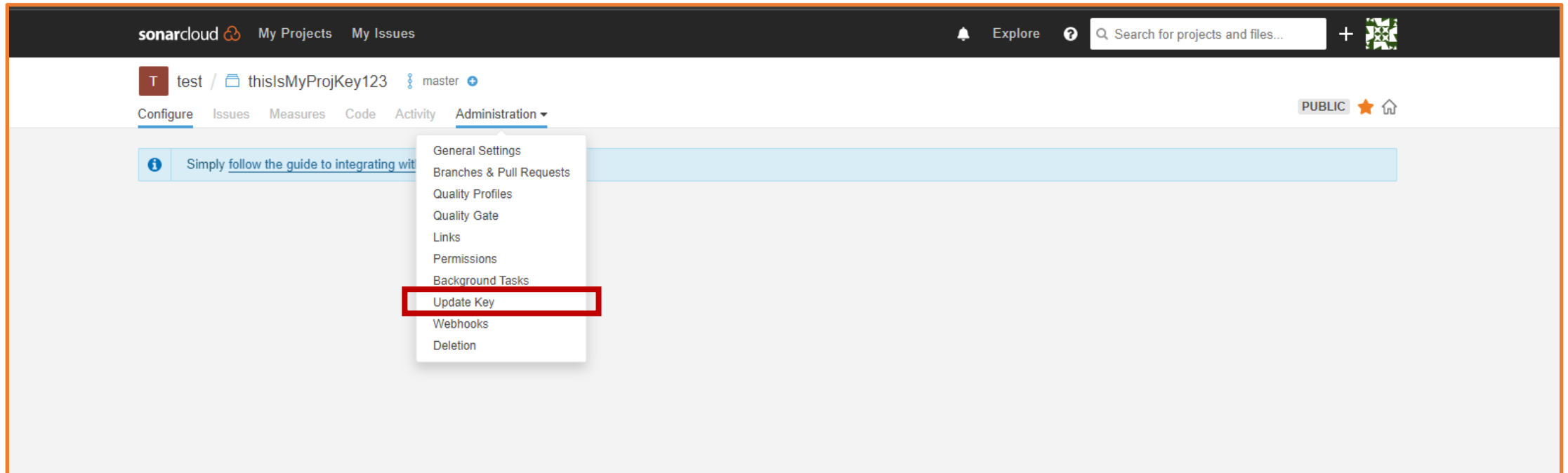
**Filters:**

- Quality Gate:** Passed (2), Warning (0), Failed (0)
- Reliability (Bugs):** A (2), B (0), C (0), D (0), E (0)
- Security (Vulnerabilities):** A (2), B (0), C (0), D (0), E (0)
- Maintainability (Code Smells):** A (2), B (0), C (0), D (0), E (0)
- Coverage:** ≥ 80% (0)

**Projects List:**

- firstsonar2020 / proj0** (NEW, Passed, PUBLIC)  
Last analysis: May 17, 2020, 6:52 PM  
0 Bugs, 0 Vulnerabilities, 97 Code Smells, 3.6% Coverage, 0.0% Duplications, 1.1k C#, PL/SQL
- firstsonar2020 / proj0** (NEW, PUBLIC)  
Project is not analyzed yet. [Configure analysis](#)
- test / project0** (NEW, Passed, PUBLIC)  
Last analysis: May 17, 2020, 7:14 PM  
0 Bugs, 0 Vulnerabilities, 97 Code Smells, 3.6% Coverage, 0.0% Duplications, 1.1k C#, PL/SQL
- test / thisIsMyProjKey123** (NEW, PUBLIC)  
Project is not analyzed yet. [Configure analysis](#)

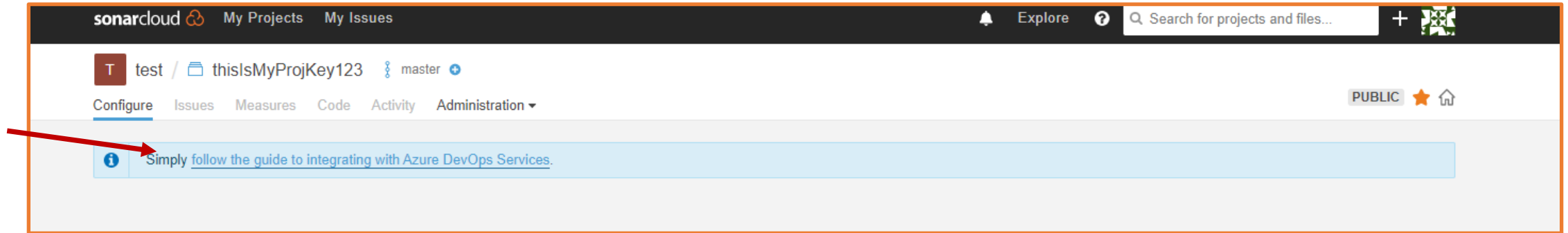
- Select the “Administration” option from the header menu
- In the drop down menu select “Update Key”



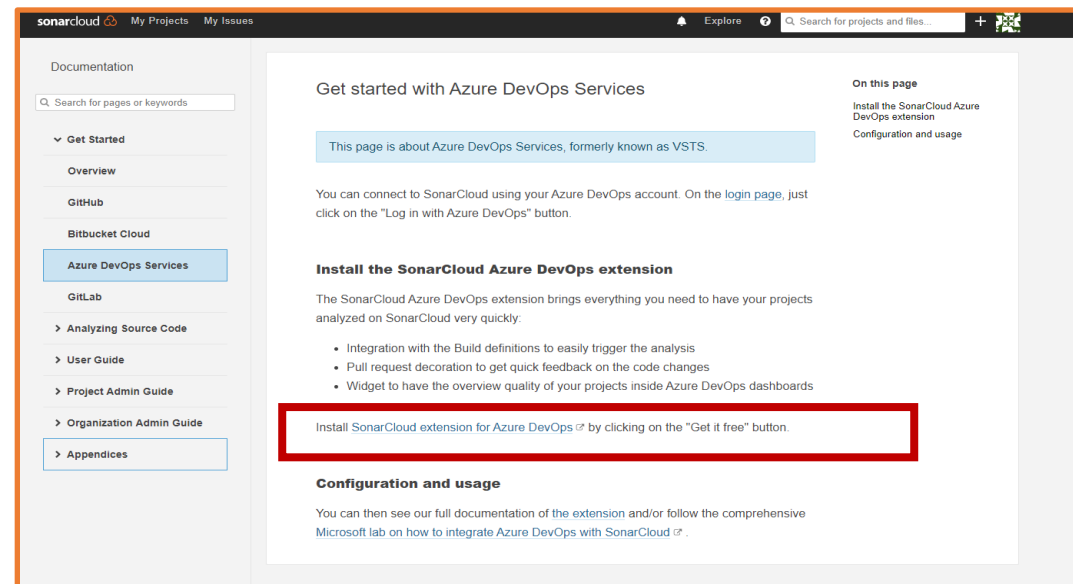
- On the “Update Key” page, paste the token that you generated into the text input box and select update.
- This step is necessary for the SonarCloudAnalyze step in Azure DevOps to properly execute

The screenshot shows the SonarCloud interface. At the top, there's a navigation bar with 'sonarcloud', 'My Projects', and 'My Issues'. Below that, a breadcrumb trail shows 'test / thisIsMyProjKey123 / master'. The 'Administration' tab is selected. The main content area is titled 'Update Key' and contains the instruction: 'Edit the key of a project. Key changes must be made here BEFORE analyzing the project with the new keys, otherwise the analysis will simply create another project with the new key, rather than updating the existing project.' Below this instruction is a text input field containing 'thisIsMyProjKey123'. To the right of the input field is a red arrow pointing left with the text 'Paste the token here'. Below the input field are two buttons: 'Update' and 'Reset'. A red arrow points up to the 'Update' button.

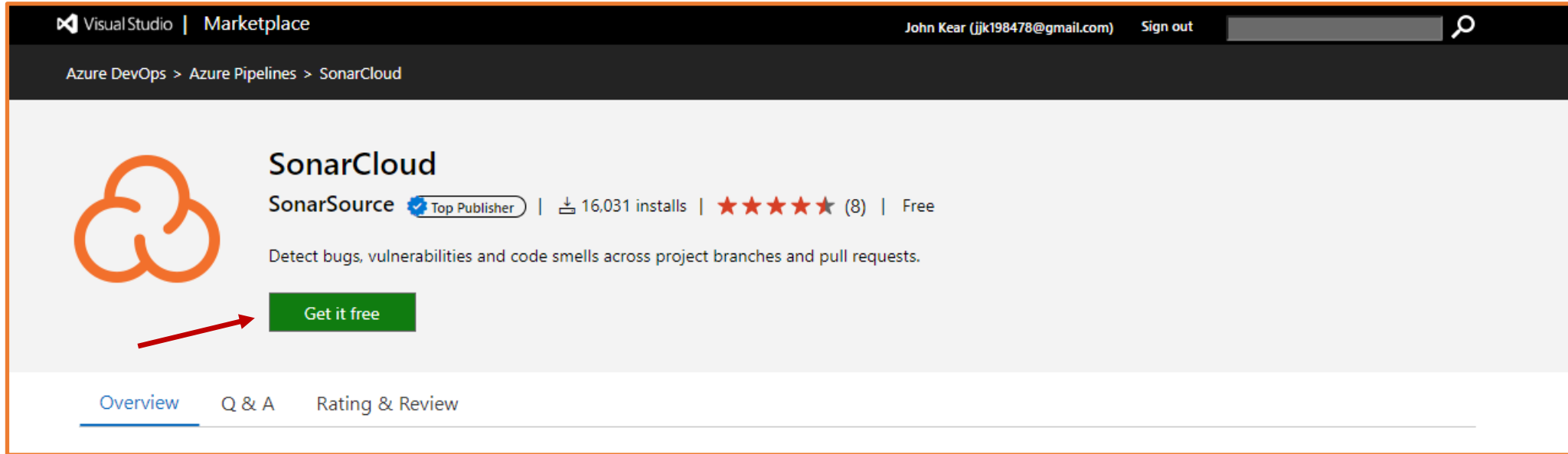
- Now we will need to install the SonarCloud extension to Azure DevOps
- From the Project page in SonarCloud click the “follow the guide to integrating with Azure DevOps Services” link (<https://sonarcloud.io/documentation/integrations/vsts/>)



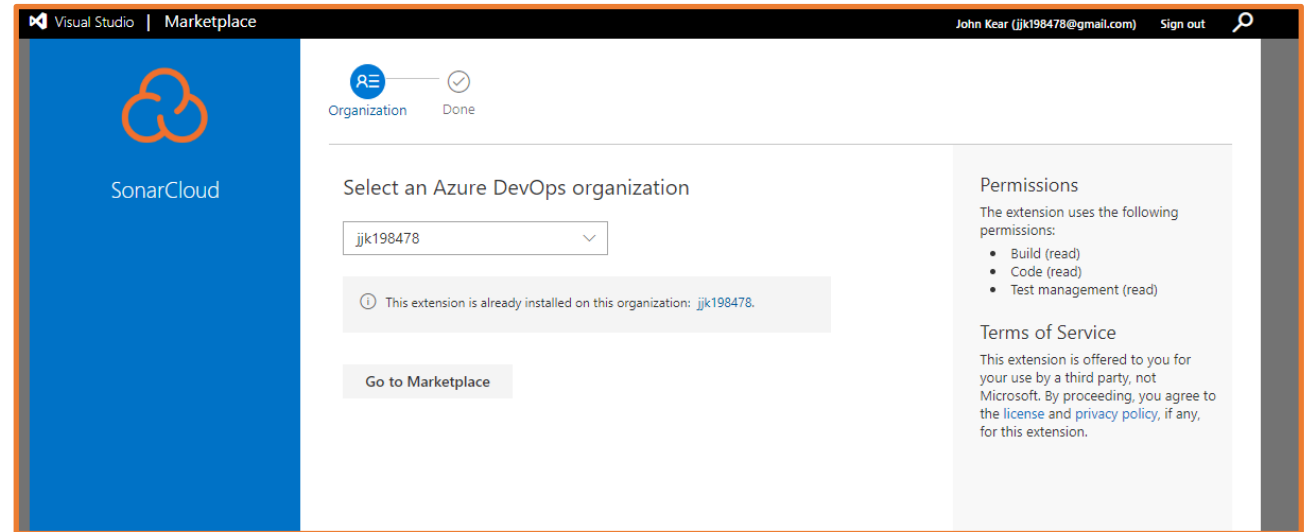
- Click the “Install SonarCloud extension for Azure DevOps” link (<https://marketplace.visualstudio.com/items?itemName=SonarSource.sonarcloud>)



- Click “Get it free”

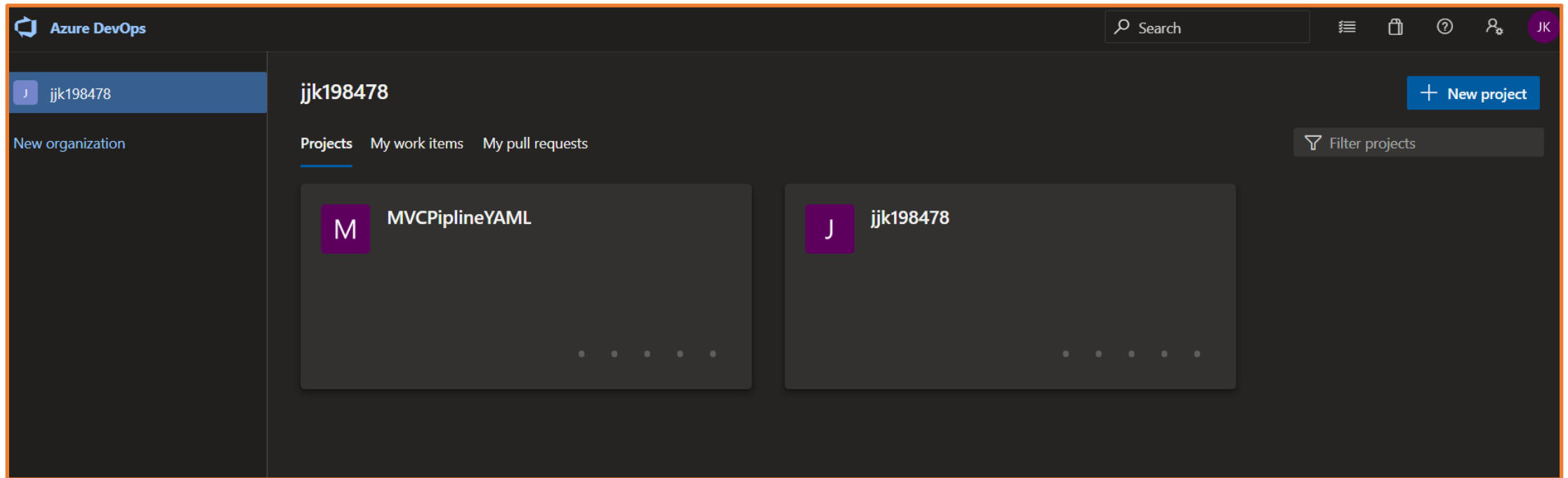


- Select your Azure DevOps organization from the drop down menu, then follow the prompts for installation.
- I already have the extension installed so I am unable to show a walkthrough of this process.

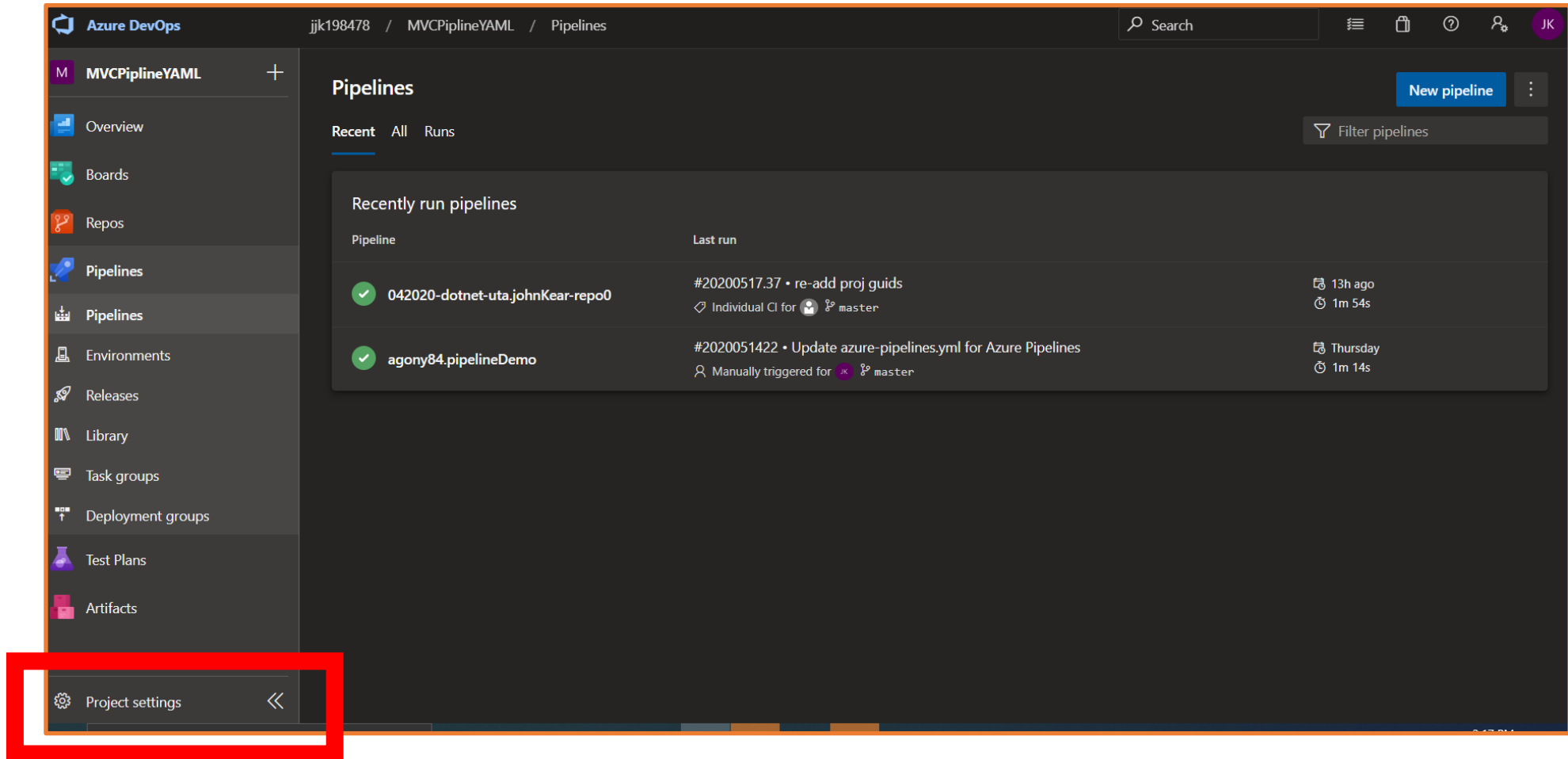


# Configuring Sonar Cloud in Azure DevOps

- With the SonarCloud extension installed to Azure DevOps we can now add the service to our Azure Project
- In Azure DevOps, select the project you wish to use with sonar cloud

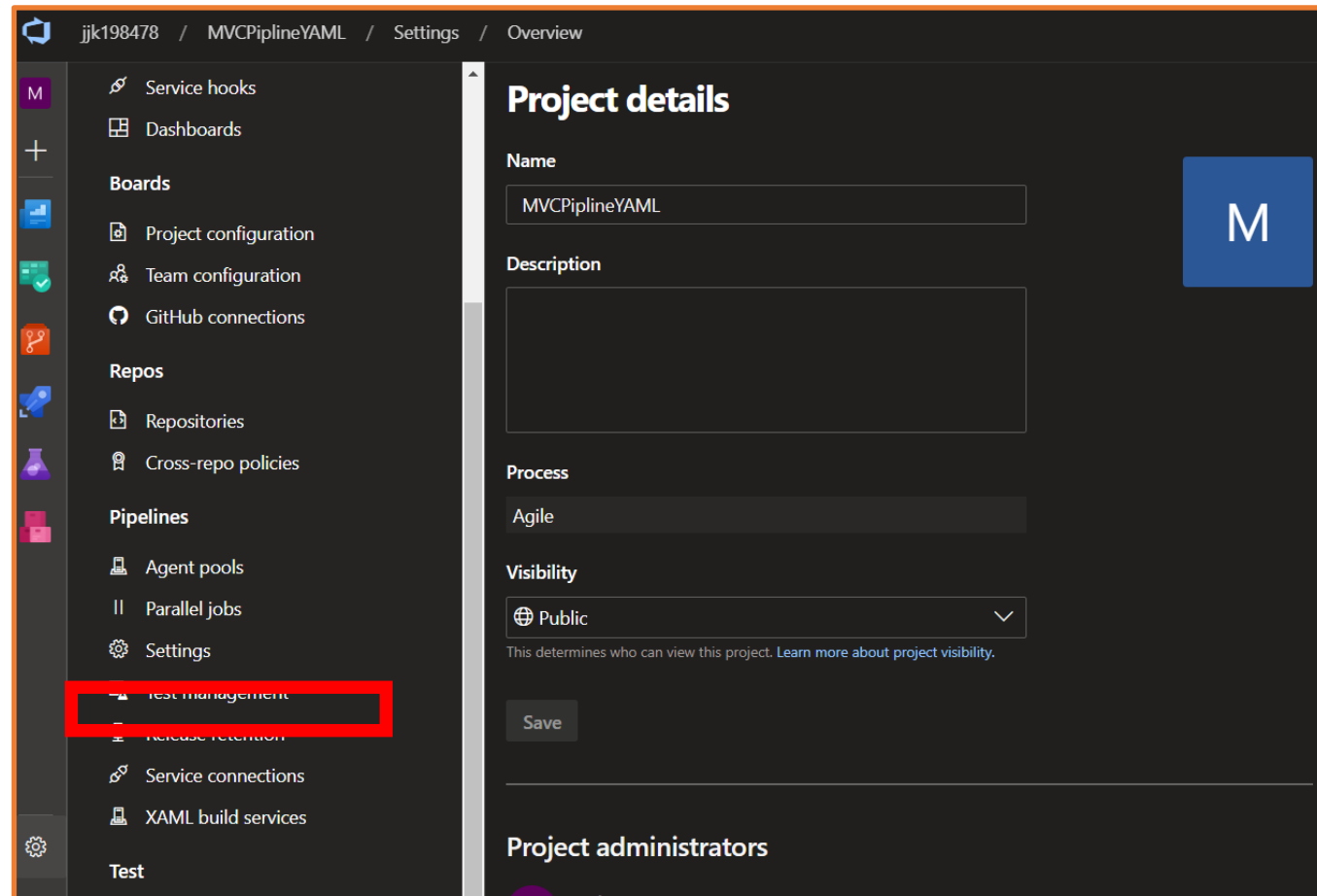


# Select Project settings in the bottom left

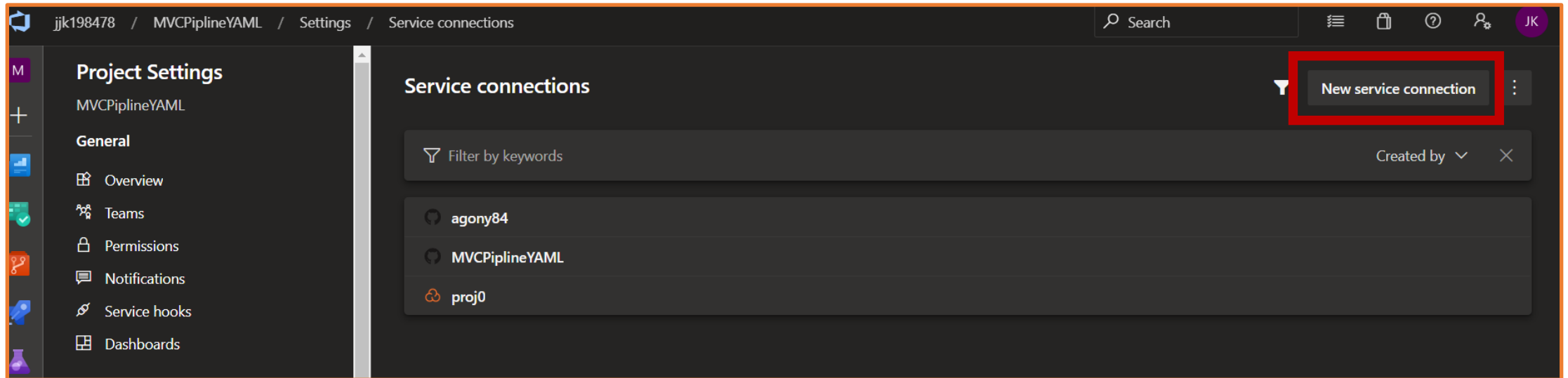




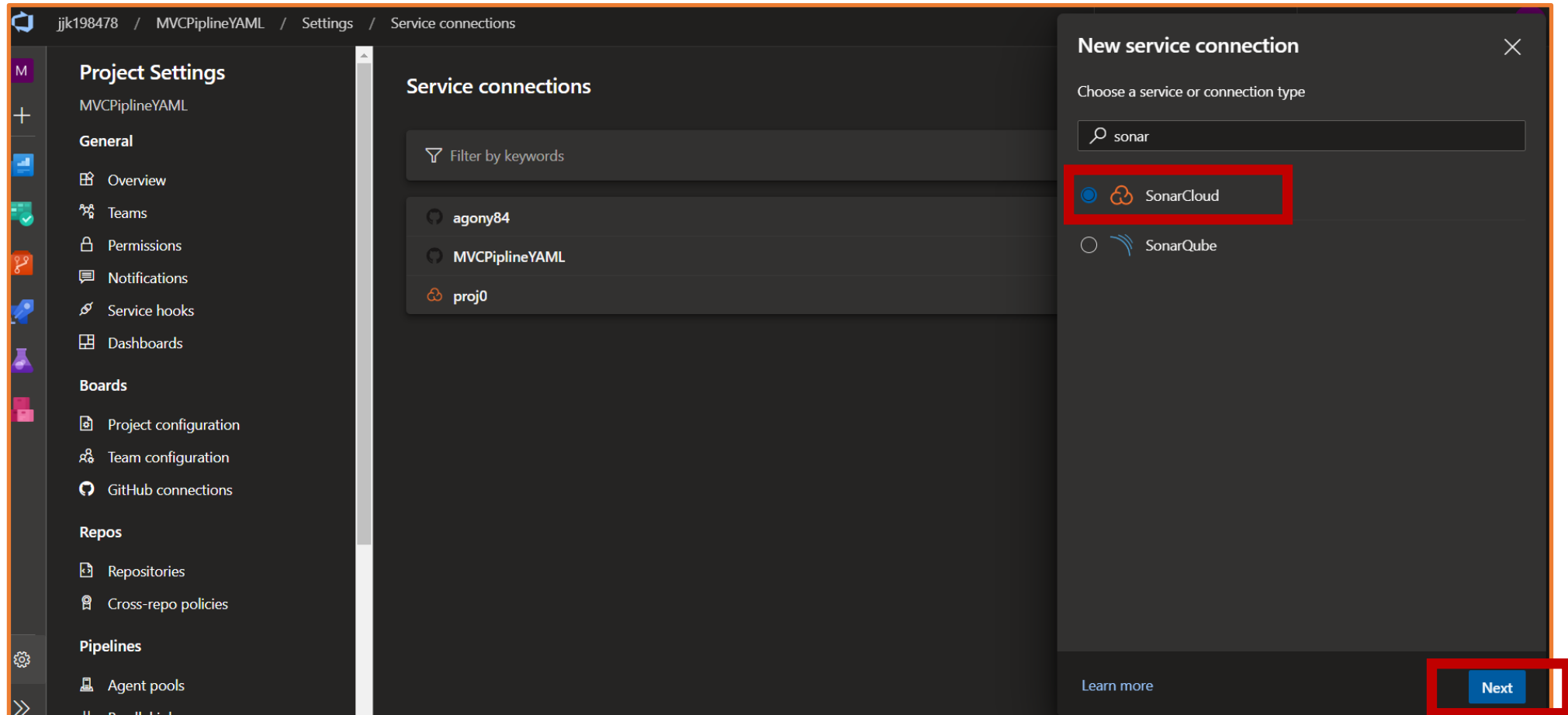
In project settings scroll down on the left and select “Service Connections” under the Pipelines heading



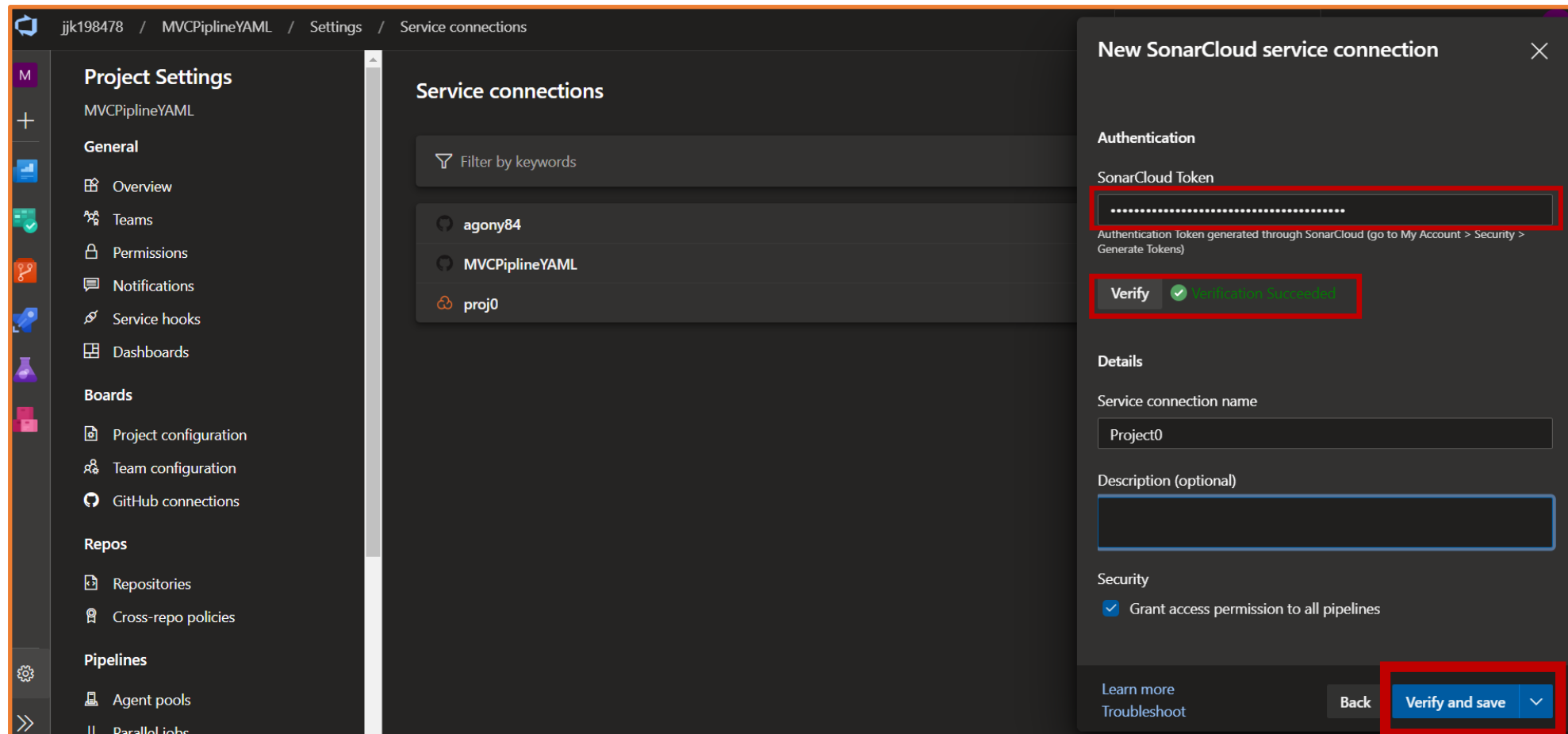
In service connections select “New service connection” at the top right



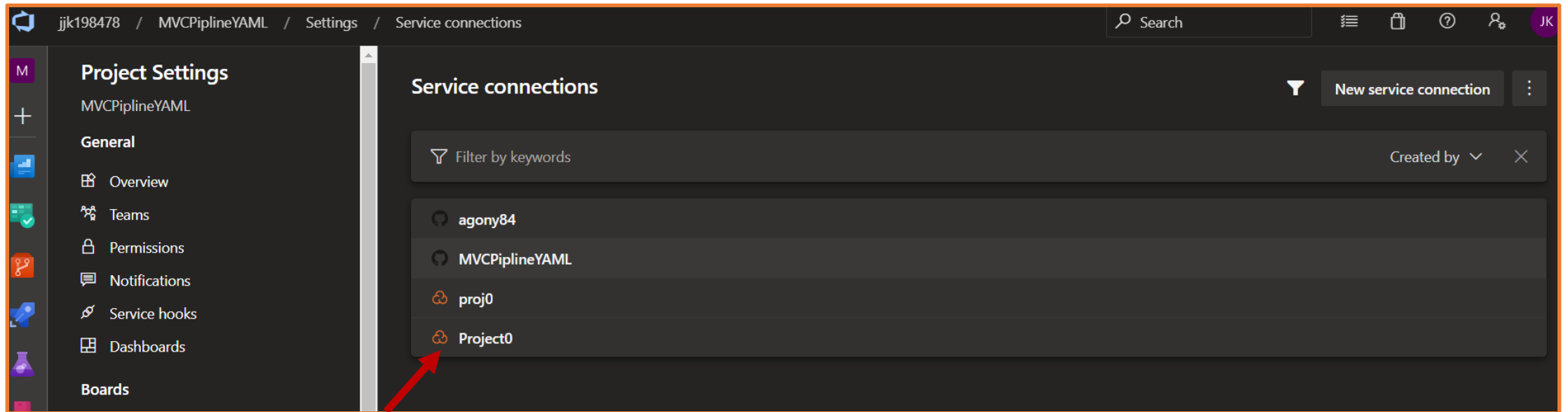
# Search for “SonarCloud”. Select it and click “Next”



- Paste the token from sonar cloud into the “SonarCloud Token” input box
- Click “Verify” to make sure that it actually works. If not, you will need to go SonarCloud and generate a new token
- Enter a name for the “Service connection name”
- Enter a description (optional)
- Leave the option “Grant access permission to all pipelines” selected
- Click “Verify and save”



- Once verified you should see the new sonar cloud connection in your “Service connections”
- Now any pipeline in the project will have access to this sonar cloud service connection



- Go back to your Azure DevOps project page and select “Pipelines” in the left menu
- Select the pipeline you wish to add the sonar cloud service to

The screenshot shows the Azure DevOps interface for a project named 'MVCPIPELINEYAML'. The left sidebar contains a menu with items: Overview, Boards, Repos, Pipelines (highlighted with a red box), Environments, Releases, Library, Task groups, Deployment groups, and Test Plans. The main content area is titled 'Pipelines' and has tabs for 'Recent', 'All', and 'Runs'. A 'New pipeline' button is in the top right. Below the tabs is a 'Filter pipelines' search bar. The 'Recently run pipelines' section contains a table with the following data:

Pipeline	Last run
✓ 042020-dotnet-uta.johnKear-repo0	#20200517.37 • re-add proj guides Individual CI for [user] master
✓ agony84.pipelineDemo	#20200514.22 • Update azure-pipelines.yml for Azure Pipelines Manually triggered for [user] master

-Select “Edit” at the top right to edit the pipeline .yaml file

The screenshot displays the Azure DevOps web interface. The top navigation bar includes the 'Azure DevOps' logo, a breadcrumb trail 'jjk198478 / MVCPipelineYAML / Pipelines / 042020-dotnet-uta.johnKear...', a search bar, and user profile icons. The left sidebar contains a list of project items: 'MVCPipelineYAML' (selected), 'Overview', 'Boards', 'Repos', 'Pipelines', and 'Environments'. The main content area is titled '042020-dotnet-uta.johnKear-repo0' and features tabs for 'Runs', 'Branches', and 'Analytics'. In the top right corner of this area, the 'Edit' button is highlighted with a red rectangular box, next to a 'Run pipeline' button and a menu icon. Below the tabs, a table lists pipeline runs with columns for 'Description', 'Stages', and timing. The first two runs are successful, indicated by green checkmarks in the 'Stages' column.

Description	Stages	Timing
#20200517.37 re-add proj guides Individual CI for 042020-dotnet-uta/johnKear-repo0 master 110dbe2	✓	13h ago 1m 54s
#20200517.36 test no guid Individual CI for 042020-dotnet-uta/johnKear-repo0 master a635946	✓	13h ago 1m 51s

## ← 042020-dotnet-uta.johnKear-repo0

master

042020-dotnet-uta/johnKear-repo0 / azure-pipelines.yml \*

```
1  # Starter pipeline
2  # Start with a minimal pipeline that you can customize to build and deploy your code.
3  # Add steps that build, run tests, deploy, and more:
4  # https://aka.ms/yaml
5  # Whenever a change is made to the master branch
6  trigger:
7    - master
8
9  # using ubuntu-latest vmImage
10 pool:
11   - vmImage: 'ubuntu-latest'
12
13 pr: - 'none'
14
15 stages:
16   - stage:
17     jobs:
18       - job: 'build'
19         steps:
20           - script: dotnet build 'Proj0/Proj0/Proj0.sln'
21           - script: echo 'Project built'
22       - job: 'test'
23         dependsOn: 'build'
24         steps:
25           - script: dotnet test 'Proj0/XUnitTest_proj0/XUnitTest_proj0.csproj'
26           - script: echo 'tests run'
27       - job: 'sonarcloud'
28         dependsOn: 'test'
29       - job: 'publish'
30         dependsOn: 'test'
31       - steps:
```

If you are using the pipeline created for your proj0 your .yml will look similar to this.



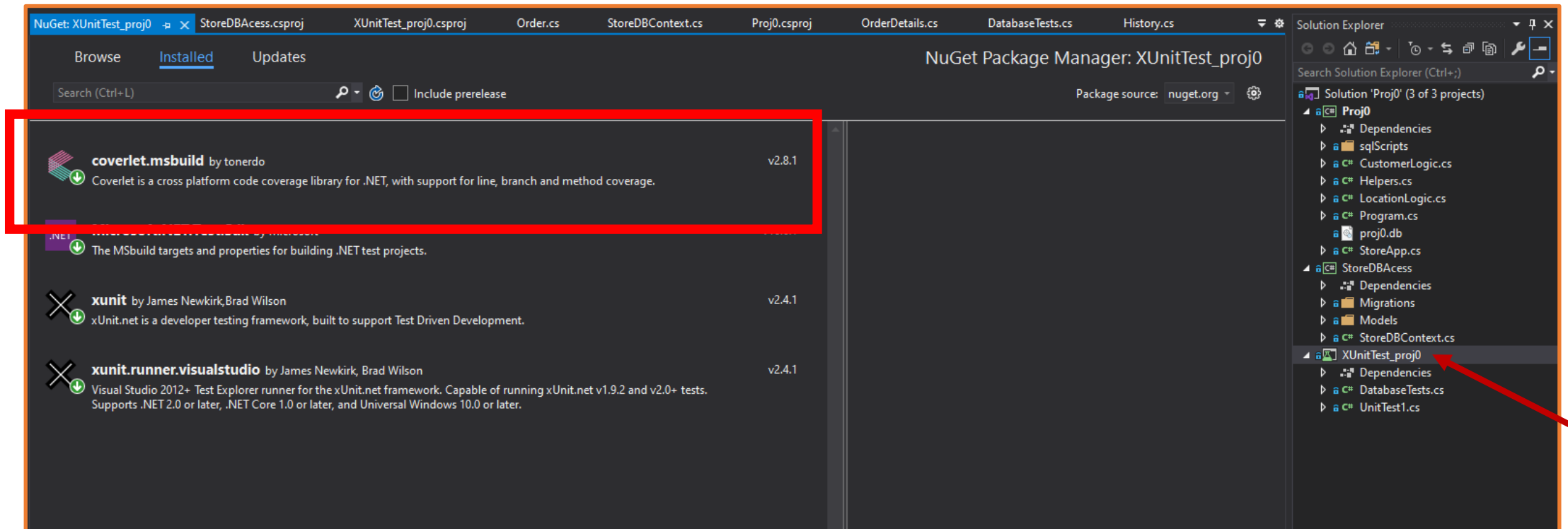
-At this point there are some things that need to be done to the .NET Core project we will be working with. Otherwise Sonar Cloud won't be happy.

-On your computer, open the project you will be analyzing. For me it will be project0

-For the code coverage analysis and report to work, your test project needs to use the "coverlet.msbuild" package. (it may be possible to use something else, but this is what I used)

NOTE: Be sure to ONLY install this package in your test project.

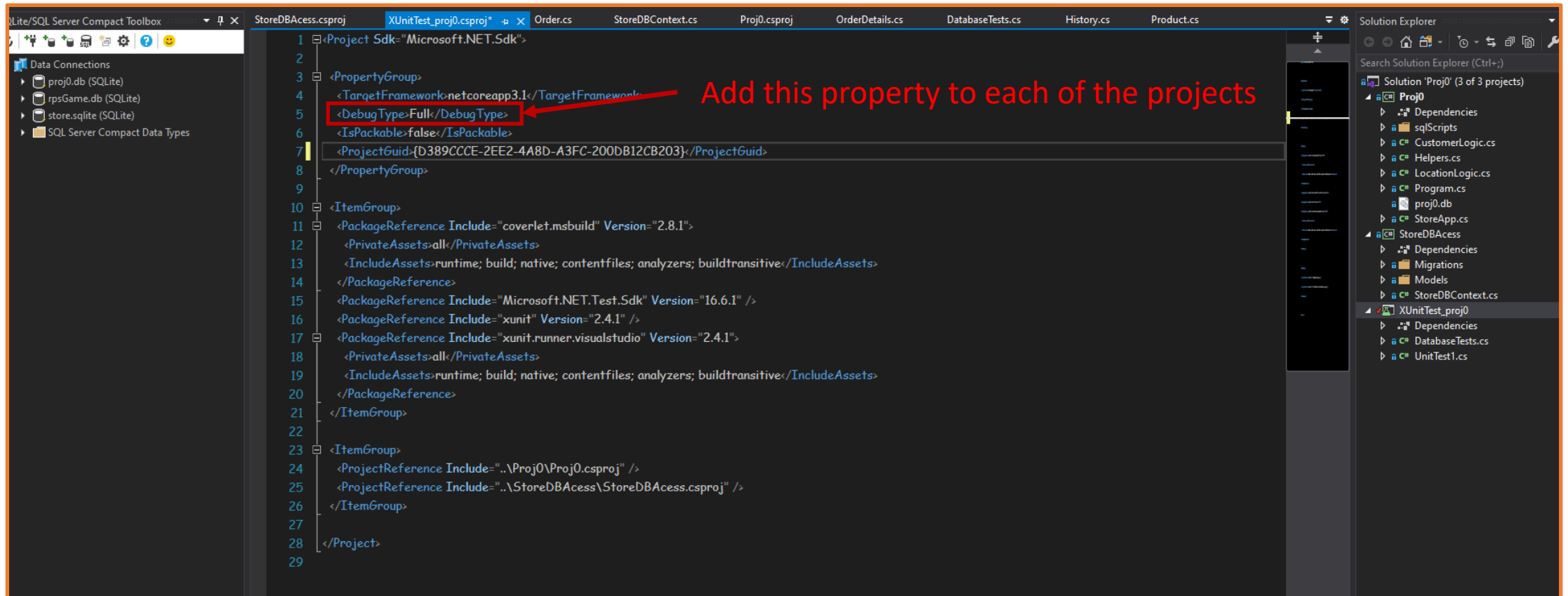
NOTE: The resource I found said to install through the PMC otherwise it would cause issues, however I installed through NuGet Package Manager and it works fine



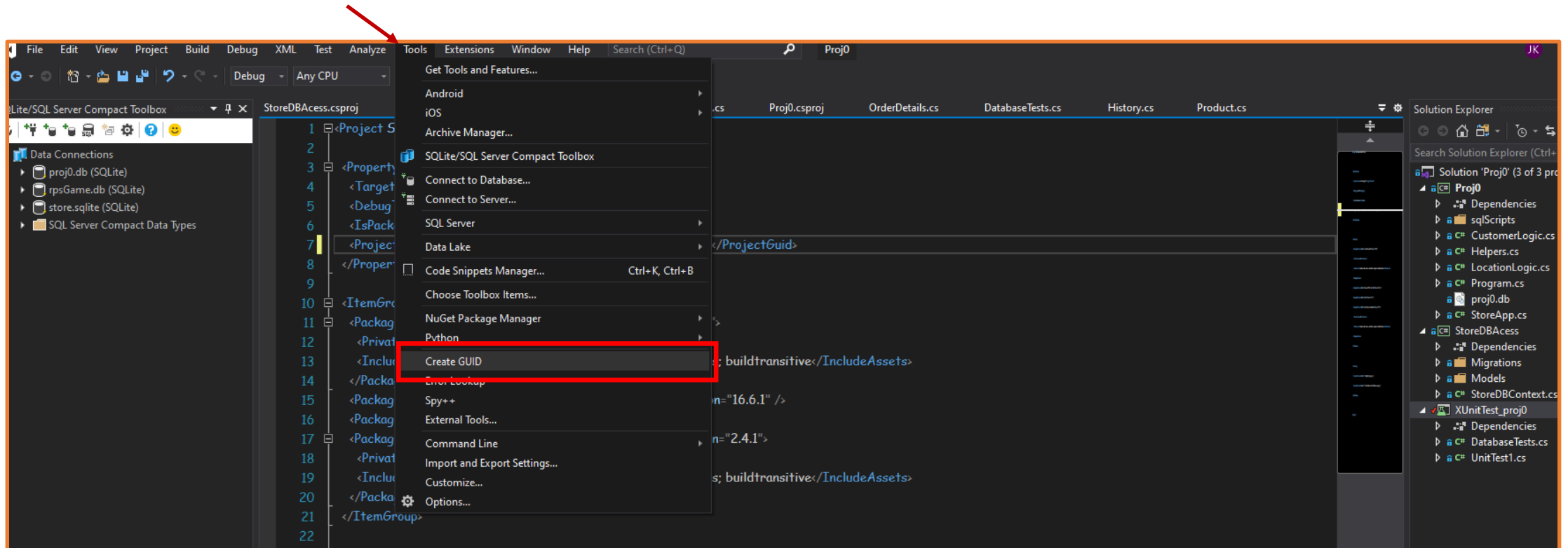
-Now that coverlet.msbuild is installed we will need to add some properties to each of the projects in the application solution

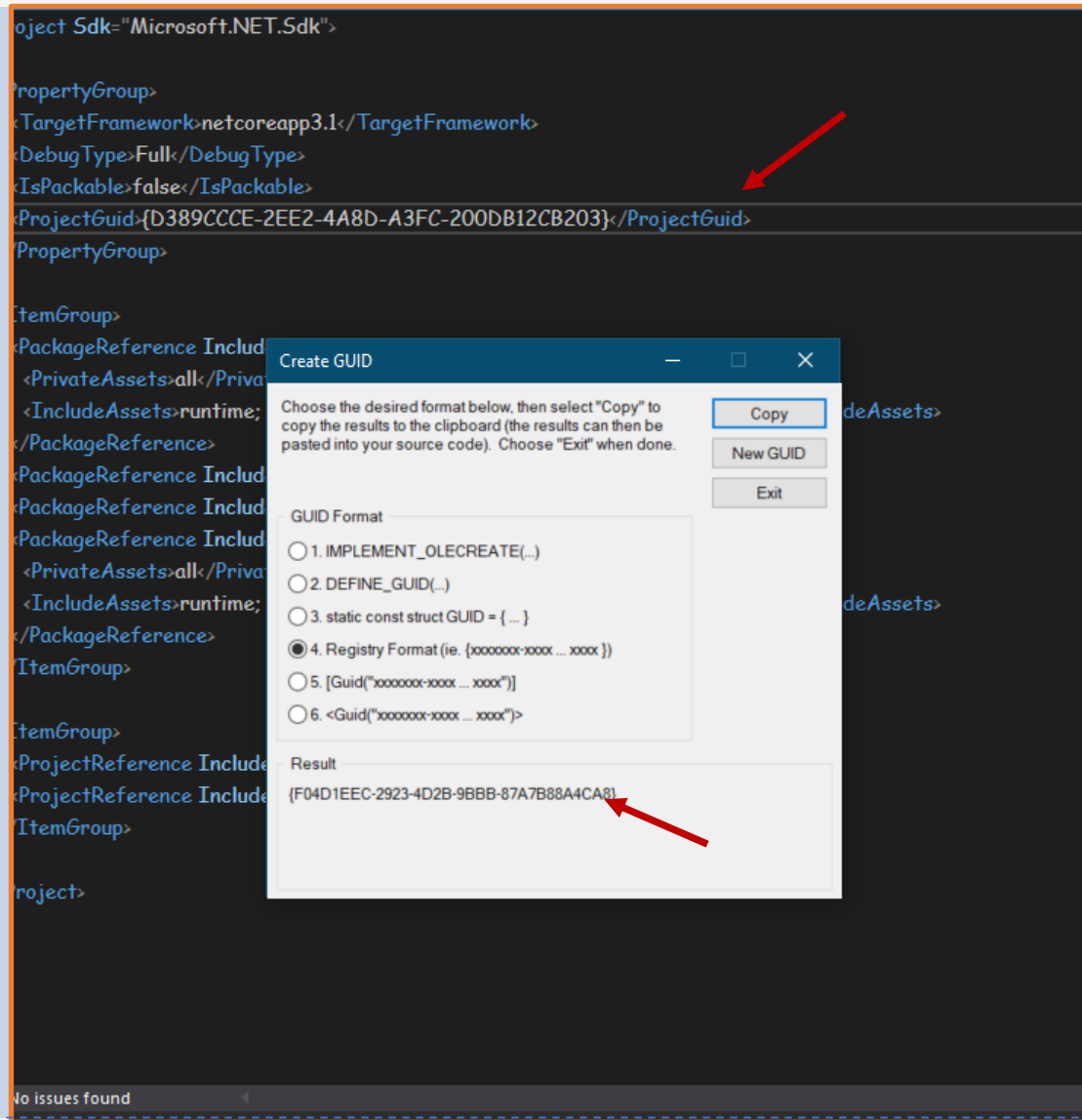
1)Each project will need a <Debug Type>Full</Debug Type> property otherwise the coverlet.msbuild tool may not function properly

2)Each Project will need a <ProjectGuid>###</ProjectGuid> property so that sonar cloud is able to differentiate between them and analyze them properly. I will show how to generate guides in the following slides



- Each project will need its own unique Guid
- To generate a guid in Visual Studio 2019 Community edition click Tools->Create Guid





- A “Create GUID” window will pop up
- Select option 4 “Registry Format”
- The GUID will be shown in the “Results” section at the bottom of the window.
- Copy this GUID somewhere for safe keeping and select “New GUID” to generate a new GUID for each of your projects
- Close the “Create GUID” window and add the property  
<ProjectGuid>projectuniqueguid</ProjectGuid> to each of your projects under  
<PropertyGroup>
- Be sure to save your changes and push the project to your Github repo
- Now that these changes have been made, you are ready to edit your .yaml in Azure DevOps

Although we were taught to setup our .yml file in this manner, sonar cloud uses three tools (Analysis Configuration, Analysis, and Publish) that must be run in the same step or they won't work.

```
1 # Starter pipeline
2 # Start with a minimal pipeline that you can customize to build and deploy your code.
3 # Add steps that build, run tests, deploy, and more:
4 # https://aka.ms/yaml
5 # Whenever a change is made to the master branch
6 trigger:
7   - master
8
9 # using ubuntu latest vmImage
10 pool:
11   - vmImage: 'ubuntu-latest'
12
13 pr: - 'none'
14
15
16
17 stages:
18   - stage:
19     jobs:
20       - job: 'build'
21         steps:
22           - script: dotnet build 'Proj0/Proj0/Proj0.sln'
23           - script: echo 'Project built'
24       - job: 'test'
25         dependsOn: 'build'
26         steps:
27           - script: dotnet test 'Proj0/XUnitTest_proj0/XUnitTest_proj0.csproj'
28           - script: echo 'tests run'
29       - job: 'sonarcloud'
30         dependsOn: 'test'
31       - job: 'publish'
32         dependsOn: 'test'
33         steps:
34           - script: dotnet publish 'Proj0/Proj0/Proj0.sln'
```

-The first thing to do is to reorganize our file and put the build, test and publish all in the same job and step

```
1  # Starter pipeline
2  # Start with a minimal pipeline that you can customize to build and deploy your code.
3  # Add steps that build, run tests, deploy, and more:
4  # https://aka.ms/yaml
5  # Whenever a change is made to the master branch
6  trigger:
7    - master
8
9  # using ubuntu-latest vmImage
10 pool:
11   - vmImage: 'ubuntu-latest'
12
13 pr: - 'none'
14
15
16
17 stages:
18   - stage:
19     jobs:
20       - job: 'build'
21         steps:
22           - script: dotnet build 'Proj0/Proj0/Proj0.sln'
23           - script: echo 'Project built'
24           - script: dotnet test 'Proj0/XUnitTest_proj0/XUnitTest_proj0.csproj'
25           - script: echo 'tests run'
26           - script: dotnet publish 'Proj0/Proj0/Proj0.sln'
```

- In the “Tasks” search type “sonar”
- Select “Prepare Analysis Configuration”

The screenshot displays the Azure DevOps web interface. On the left, a sidebar contains navigation links: Overview, Boards, Repos, Pipelines (selected), Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area shows the '042020-dotnet-uta.johnKear-repo0' pipeline configuration. The 'trigger' section is set to 'master'. The 'pool' is 'ubuntu-latest'. The 'stages' section contains a single stage with a job named 'build' and several steps: 'dotnet build', 'echo', 'dotnet test', 'echo', and 'dotnet publish'. On the right, the 'Tasks' panel is open, showing a search bar with 'sonar' entered. Below the search bar, a list of tasks is displayed. The first task, 'Prepare Analysis Configuration' (Prepare SonarCloud analysis configuration), is highlighted with a red box. A red arrow points from the search bar to this task. Other tasks in the list include 'Prepare Analysis Configuration' (Prepare SonarQube analysis configuration), 'Publish Quality Gate Result' (Publish SonarQube's Quality Gate result on the A...), 'Publish Quality Gate Result' (Publish SonarCloud's Quality Gate result on the A...), 'Run Code Analysis' (Run scanner and upload the results to the SonarQ...), and 'Run Code Analysis' (Run scanner and upload the results to the SonarC...).

```
1 # Starter pipeline
2 # Start with a minimal pipeline that you can customize to build and deploy your code.
3 # Add steps that build, run tests, deploy, and more:
4 # https://aka.ms/yaml
5 # Whenever a change is made to the master branch
6 trigger:
7   - master
8
9 # using ubuntu-latest vmImage
10 pool:
11   vmImage: 'ubuntu-latest'
12
13 pr: 'none'
14
15
16
17 stages:
18   - stage:
19     jobs:
20       - job: 'build'
21         steps:
22           - script: dotnet build 'Proj0/Proj0/Proj0.sln'
23             - script: echo 'Project built'
24             - script: dotnet test 'Proj0/UnitTest_proj0/UnitTest_proj0.csproj'
25             - script: echo 'tests run'
26             - script: dotnet publish 'Proj0/Proj0/Proj0.sln'
```

- From the dropdown select the SonarCloud Service Endpoint we created for the project
- From the dropdown, select the SonarCloud organization
- We are using the coverlet.msbuild tool for reports so select the “Integrate with MSBuild” option for analysis selected
- For the “Project Key” paste the generated token from sonar cloud
- Enter a project name
- Expand the “Advanced” section
- In the additional properties section add the following:  
`sonar.exclusions=**/obj/**,**/*.dll`  
`sonar.branch.name=$(Build.SourceBranchName)`  
`sonar.cs.vstest.reportsPaths=$(Agent.TempDirectory)/*.trx`  
`sonar.cs.opencover.reportsPaths=$(Build.SourcesDirectory)/**/*.xml`
- Click “Add”

← Prepare Analysis Configuration

SonarCloud Service Endpoint \* ⓘ  
Project0

Organization \* ⓘ  
test (testkey12345)

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

Project Key \* ⓘ  
722ecd6f75aee346b10372b711721ca33042b4

Project Name ⓘ  
Project0

Project Version ⓘ  
1.0

Advanced ^

Additional Properties ⓘ  
 sonar.exclusions=\*\*/obj/\*\*,\*\*/\*.dll  
 sonar.branch.name=\$(Build.SourceBranchName)

About this task Add



- A resulting task should have been added and look similar to this
  - This task needs to be placed before the “Build” command as shown in the screenshot on the right.
- Pay attention to indentation!

```
steps:
- script: dotnet build 'Proj0/Proj0/Proj0.sln'
- task: SonarCloudPrepare@1
  inputs:
    SonarCloud: 'Project0'
    organization: 'testkey12345'
    scannerMode: 'MSBuild'
    projectKey: 'thisIsMyProjectKey'
    projectName: 'project0'
    extraProperties: |
      sonar.exclusions=**/obj/**, **/*.dll
      sonar.branch.name=$(Build.SourceBranchName)
      sonar.cs.vstest.reportsPaths=$(Agent.TempDirectory)/*.trx
      sonar.cs.opencover.reportsPaths=$(Build.SourcesDirectory)/**/*.xml
    - script: echo 'Project built'
    - script: dotnet test 'Proj0/XUnitTest_proj0/XUnitTest_proj0.csproj'
    - script: echo 'tests run'
    - script: dotnet publish 'Proj0/Proj0/Proj0.sln'
```

```
stages:
- stage:
  jobs:
  - job: 'build'
    steps:
      Settings
      - task: SonarCloudPrepare@1
        inputs:
          SonarCloud: 'Project0'
          organization: 'testkey12345'
          scannerMode: 'MSBuild'
          projectKey: 'thisIsMyProjectKey'
          projectName: 'project0'
          extraProperties: |
            sonar.exclusions=**/obj/**, **/*.dll
            sonar.branch.name=$(Build.SourceBranchName)
            sonar.cs.vstest.reportsPaths=$(Agent.TempDirectory)/*.trx
            sonar.cs.opencover.reportsPaths=$(Build.SourcesDirectory)/**/*.xml
          - script: dotnet build 'Proj0/Proj0/Proj0.sln'
          - script: echo 'Project built'
          - script: dotnet test 'Proj0/XUnitTest_proj0/XUnitTest_proj0.csproj'
          - script: echo 'tests run'
          - script: dotnet publish 'Proj0/Proj0/Proj0.sln'
```

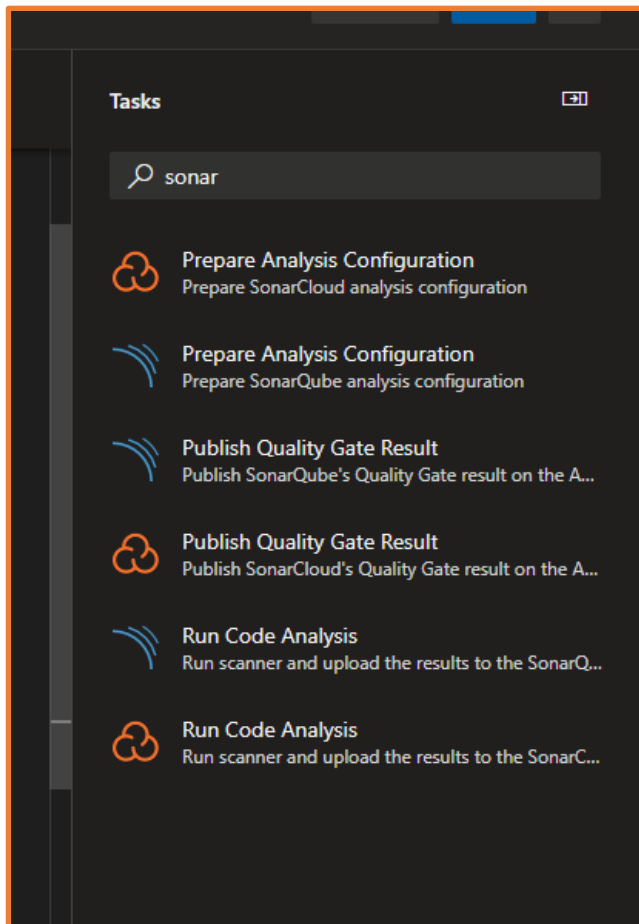
- In the “Tasks” search enter sonar again
- Select “Run Code Analysis” make sure to select the option with the sonar cloud logo
- A tasks will be auto generated as shown
- Move the task to AFTER the test script

The image displays the Azure DevOps interface for adding a task to a build pipeline. On the left, the 'Tasks' search results are shown, with the 'Run Code Analysis' task (featuring the SonarCloud logo) highlighted by a red box. A red arrow points from this task to the right, where the generated YAML code is displayed. The code is divided into two sections: the first section shows the 'stages' and 'jobs' configuration, and the second section shows the 'steps' configuration. The 'steps' configuration includes a task named 'SonarCloudPrepare@1' and a task named 'SonarCloudAnalyze@1'. The 'SonarCloudAnalyze@1' task is highlighted by a red box, and a red arrow points to its 'script' field, which contains the command 'dotnet publish'. The 'SonarCloudPrepare@1' task is also highlighted by a red box, and a red arrow points to its 'script' field, which contains the command 'dotnet build'.

```
stages:
- stage:
  jobs:
    job: 'build'
    steps:
      Settings
      - task: SonarCloudPrepare@1
        inputs:
          SonarCloud: 'Project0'
          organization: 'testkey12345'
          scannerMode: 'MSBuild'
          projectKey: 'thisIsMyProjectKey'
          projectName: 'project0'
          extraProperties: |
            sonar.exclusions=**/obj/**,*/*.dll
            sonar.branch.name=$(Build.SourceBranchName)
            sonar.cs.vstest.reportsPaths=$(Agent.TempDirectory)/**/*.trx
            sonar.cs.opencover.reportsPaths=$(Build.SourcesDirectory)/**/*.xml
      - task: SonarCloudAnalyze@1
        script: dotnet build 'Proj0/Proj0/Proj0.sln'
        script: echo 'Project built'
        script: dotnet test 'Proj0/XUnitTest_proj0/XUnitTest_proj0.csproj'
        script: echo 'tests run'
        script: dotnet publish 'Proj0/Proj0/Proj0.sln'
```

```
stages:
- stage:
  jobs:
    job: 'build'
    steps:
      Settings
      - task: SonarCloudPrepare@1
        inputs:
          SonarCloud: 'Project0'
          organization: 'testkey12345'
          scannerMode: 'MSBuild'
          projectKey: 'thisIsMyProjectKey'
          projectName: 'project0'
          extraProperties: |
            sonar.exclusions=**/obj/**,*/*.dll
            sonar.branch.name=$(Build.SourceBranchName)
            sonar.cs.vstest.reportsPaths=$(Agent.TempDirectory)/**/*.trx
            sonar.cs.opencover.reportsPaths=$(Build.SourcesDirectory)/**/*.xml
      - script: An inline script Proj0/Proj0/Proj0.sln
      - script: echo 'Project built'
      - script: dotnet test 'Proj0/XUnitTest_proj0/XUnitTest_proj0.csproj'
      - script: echo 'tests run'
      Settings
      - task: SonarCloudAnalyze@1
        script: dotnet publish 'Proj0/Proj0/Proj0.sln'
```

- In the “Tasks” search enter sonar again
- Select “Publish Quality Gate Result” and click “Add”. make sure to select the option with the sonar cloud logo
- A tasks will be auto generated as shown
- Move the task somewhere after the SonarCloudAnalyze task



```
stages:
- stage:
- jobs:
- job: 'build'
- steps:
  Settings
  - task: SonarCloudPrepare@1
    inputs:
      SonarCloud: 'Project0'
      organization: 'testkey12345'
      scannerMode: 'MSBuild'
      projectKey: 'thisIsMyProjectKey'
      projectName: 'project0'
      extraProperties: |
        sonar.exclusions=**/obj/**/*.dll
        sonar.branch.name=$(Build.SourceBranchName)
        sonar.cs.vstest.reportsPaths=$(Agent.TempDirectory)/*.trx
        sonar.cs.opencover.reportsPaths=$(Build.SourcesDirectory)/**/*.xml
      script: dotnet build 'Proj0/Proj0/Proj0.sln'
      script: echo 'Project built'
      script: dotnet test 'Proj0/XUnitTest_proj0/XUnitTest_proj0.csproj'
      script: echo 'tests run'
  Settings
  - task: SonarCloudAnalyze@1
    Settings
    - task: SonarCloudPublish@1
      inputs:
        pollingTimeoutSec: '300'
      script: dotnet publish 'Proj0/Proj0/Proj0.sln'
```

- In order for SonarCloudAnalyze and SonarCloudPublish to analyze and publish the Code Coverage, it must first be generated. Generating the Code Coverage analysis file is done during testing.
- To generate the Code Coverage analysis file, we must alter the test process.
- Instead of using the script dotnet test, we will replace it with a task as shown
- The task is a DotNetCoreCLI@2
- The display name is merely for Azure DevOps when running the task. We will see this later.
- The command is 'test'
- In arguments we set the configuration parameters. These include setting Collect Coverage to true, setting the CoverletOutputFormat to opencover (very important!!!), the Coverletoutput path, and turn on the logger for test results.
- In "projects" we specify the test proj
- "nobuild" can be set to true

```
master 042020-dotnet-uta/johnKear-repo0 / azure-pipelines.yml *
20      -- job: 'build'
21      -- steps:
22          Settings
23          -- task: SonarCloudPrepare@1
24          -- inputs:
25              SonarCloud: 'Project0'
26              organization: 'testkey12345'
27              Project Key: 'MSBuild'
28              projectKey: 'thisIsMyProjectKey'
29              projectName: 'project0'
30              extraProperties: |
31                  sonar.exclusions=**/obj/**, **/*.dll
32                  sonar.branch.name=$(Build.SourceBranchName)
33                  sonar.cs.vstest.reportsPaths=$(Agent.TempDirectory)/*.trx
34                  sonar.cs.opencover.reportsPaths=$(Build.SourcesDirectory)/**/*.xml
35          -- script: dotnet build 'Proj0/Proj0/Proj0.sln'
36          -- script: echo 'Project built'
37
38      -- Settings
39      -- task: DotNetCoreCLI@2
40      -- displayName: dotnet test
41      -- inputs:
42          command: 'test'
43          arguments: '--configuration $(BuildConfiguration)
44              /p:CollectCoverage=true
45              /p:CoverletOutputFormat=opencover
46              /p:CoverletOutput=$(Build.SourcesDirectory)/TestResults/Coverage
47              --logger:trx'
48          projects: '**/*XUnitTest_proj0.csproj'
49          nobuild: true
50
51      -- Settings
52      -- task: SonarCloudAnalyze@1
53
54      -- Settings
55      -- task: SonarCloudPublish@1
56      -- inputs:
57          pollingTimeoutSec: '300'
58      -- script: dotnet publish 'Proj0/Proj0/Proj0.sln'
```

- Now that the test has been configured and code coverage configured we can save our .yaml file and run it.
- The run should complete as shown
- By clicking the back arrow, we can go back to see the information generated

The screenshot displays the Azure DevOps web interface. The top navigation bar shows the user 'jjk198478' and the current path: 'MVCPIplineYAML / Pipelines / 042020-dotnet-uta.johnKear... / 20200518.11'. A search bar is located in the top right corner.

The left sidebar contains a list of navigation items: 'MVCPIplineYAML' (selected), 'Overview', 'Boards', 'Repos', 'Pipelines', 'Environments', 'Releases', 'Library', 'Task groups', 'Deployment groups', 'Test Plans', and 'Artifacts'. The 'Pipelines' item is highlighted.

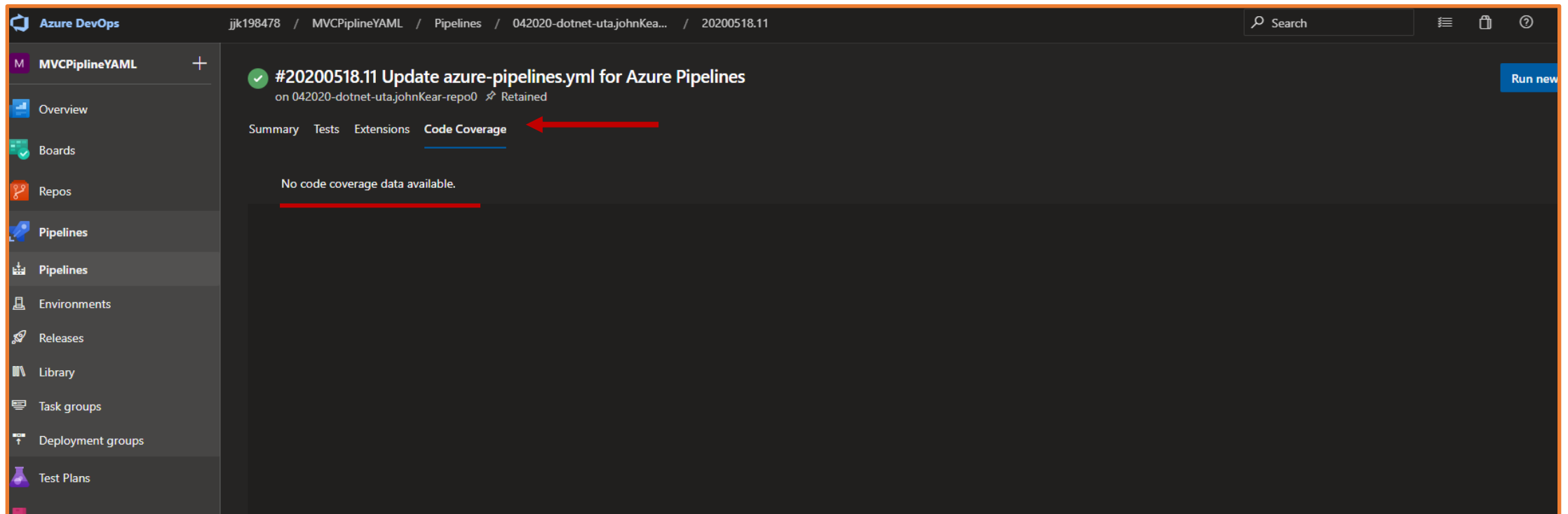
The main content area is divided into two panels. The left panel, titled 'Jobs in run #20200518.11', shows a list of jobs for the pipeline '042020-dotnet-uta.johnKear-repo0'. A red box highlights a back arrow icon next to the title. The jobs listed are:

Job Name	Status	Duration
build	Completed	2m 22s
Initialize job	Completed	4s
Checkout 042020-dotnet-uta/joh...	Completed	1s
SonarCloudPrepare	Completed	17s
CmdLine	Completed	43s
dotnet test	Completed	21s
SonarCloudAnalyze	Completed	45s
SonarCloudPublish	Completed	3s
CmdLine	Completed	4s
Post-job: Checkout 042020-dotn...	Completed	<1s
Finalize Job	Completed	<1s

The right panel shows the details for the 'build' job, which is marked as 'Completed'. It lists the following information:

- 1 Pool: [Azure Pipelines](#)
- 2 Image: ubuntu-latest
- 3 Agent: Hosted Agent
- 4 Started: Today at 7:39 PM
- 5 Duration: 2m 22s
- 6
- 7 ▶ Job preparation parameters
- 8 [100% tests passed](#)

- If you click on the “Code Coverage” option you will notice there is no Code Coverage published to Azure DevOps
- This is because we did not add a task to publish code coverage to Azure. We will do this in the next slide
- Even though we have no code coverage report in Azure, we should still be able to see them in SonarCloud



```

34 | | | | | Settings
35 | | | | | --task: DotNetCoreCLI@2
36 | | | | | displayName: dotnet test
37 | | | | | inputs:
38 | | | | |   command: 'test'
39 | | | | |   arguments: '--configuration $(BuildConfiguration)
40 | | | | |   /p:CollectCoverage=true /p:CoverletOutputFormat=opencover
41 | | | | |   /p:CoverletOutput=$(Build.SourcesDirectory)/TestResults/Coverage
42 | | | | |   --logger:trx'
43 | | | | | projects: '**/*XUnitTest_proj0.csproj'
44 | | | | | nobuild: true
45 | | | | | Settings
46 | | | | | --task: PublishCodeCoverageResults@1
47 | | | | | displayName: 'Publish Code Coverage'
48 | | | | | inputs:
49 | | | | |   codeCoverageTool: Cobertura
50 | | | | |   summaryFileLocation: '$(Build.SourcesDirectory)**/*.xml'
51 | | | | |   reportDirectory: '$(Build.SourcesDirectory)/CodeCoverage'
52 | | | | | Settings
53 | | | | | --task: SonarCloudAnalyze@1
54 | | | | | Settings
55 | | | | | --task: SonarCloudPublish@1
56 | | | | | inputs:
57 | | | | |   pollingTimeoutSec: '300'
58 | | | | | script: dotnet publish 'Proj0/Proj0/Proj0.sln'

```

- Optionally, we can add a CodeCoveragePublish task so that we will also have the code coverage report in our Azure DevOps.
- This is done with a PublishCodeCoverageResults@1 task
- The task must come somewhere AFTER the test task (the code coverage report is generate in the test task)
- The input codeCoverageTool will be Cobertura
- The summaryFileLocation we can set to \$(Build.SourcesDirectory)\*\*/\*.xml
- the report Directory we can set to \$(Build.SourcesDirectory)/CodeCoverage

- After you save the changes and the pipeline runs, you will see that we now have results in the “Code Coverage” section.
- These results should match what is shown in sonar cloud.
- If you have followed this tutorial, you will also notice two warnings generated by the PublishCodeCoverageResults task. I have yet to investigate these and don’t know what is causing them.

#20200518.12 Update azure-pipelines.yml for Azure Pipelines

on 04/20/20-dotnet-utaJohnKear-repo0 Retained

Summary

Tests

Extensions

Code Coverage

Summary

Generated on:	5/18/20 - 1:54:23 AM
Parser:	OpenCoverParser
Assemblies:	2
Classes:	14
Files:	15
Covered lines:	32
Uncovered lines:	904
Coverable lines:	936
Total lines:	1702
Line coverage:	3.4% (32 of 936)
Covered branches:	1
Total branches:	204
Branch coverage:	0.4% (1 of 204)

Risk Hotspots

Assembly	Class	Method	Cyclomatic complexity	NPath complexity
Proj0	Proj0.CustomerLogic	CustomerOptions(...)	76	0
Proj0	Proj0.CustomerLogic	CreateNewCustomer()	45	0
Proj0	Proj0.StoreApp	Run()	35	0

Warnings 2

!

Ignoring coverage report directory with Html content as we are auto-generating Html content  
Publish Code Coverage

!

No coverage data found. Check the build errors/warnings for more details.  
Publish Code Coverage



- Back in sonar cloud you should see your project with the result including Coverage

The screenshot displays the SonarCloud web interface. At the top, a green notification bar states: "Project 'thisIsMyProjKey123' has been successfully deleted." The navigation bar includes "sonarcloud", "My Projects", and "My Issues". A search bar is present with the text "Search for projects and files...".

On the left, the "Filters" section is visible, containing:

- Quality Gate:** Passed (2), Warning (0), Failed (0).
- Reliability (Bugs):** A (2), B (0), C (0), D (0), E (0).
- Security (Vulnerabilities):** A (2), B (0), C (0), D (0), E (0).
- Maintainability (Code Smells):** A (2), B (0), C (0).

The main content area shows a list of projects. The first project, "firstsonar2020 / proj0", is in a "NEW" state and "Passed". It has 0 Bugs, 0 Vulnerabilities, 97 Code Smells, 3.6% Coverage, and 0.0% Duplications. The last analysis was on May 17, 2020, at 6:52 PM. The second project, "test / project0", is also in a "NEW" state and "Passed", with identical metrics and analysis time (May 17, 2020, at 7:55 PM). This second project card is highlighted with a red border. Below the project cards, it indicates "3 of 3 shown".