A black and white logo

Description automatically generated with low confidence

|  |  |
| --- | --- |
| Document name | Dot Net Core SCA and Code Coverage Using CLI and Azure Pipeline |
| Version no. | 1.1 |
| Release date | 31 October 2023 |
| Document ID | *1* |
| Classification | Internal |

|  |
| --- |
| **DOT NET CORE SCA AND CODE COVERAGE USING CLI AND AZURE PIPELINE** |

# Document History

| Ver. No | Release Date | Created By / Modified By and Date | Reviewed By and Date | Approved By and Date | Remarks and Changes Made |
| --- | --- | --- | --- | --- | --- |
| 1.0 | 8th February 2023 | Prasad Govindrao Gattewar/ 8th February 2023 |  |  |  |
| 1.1 | 31ST October 31, 2023 | Ramya Maddipati/31st October 2023 | Prasad Govindrao Gattewar |  |  |

# Table of Contents

[Document History 2](#_Toc149666153)

[1. sca and Code Coverage using cli 4](#_Toc149666155)

[1.1 Generate Token: - 4](#_Toc149666156)

[1.2 Prerequisites: - 5](#_Toc149666157)

[1.3 Performing sca using cli: 5](#_Toc149666158)

[1.4 Performing code coverage using cli: 6](#_Toc149666159)

[2. sca and Code Coverage using Azure Pipeline 7](#_Toc149666160)

[2.1 prerequisites: 7](#_Toc149666161)

[2.2 create a service connection: 7](#_Toc149666162)

[2.3 create a pipeline: 10](#_Toc149666163)

[3. References 18](#_Toc149666164)

# sca and Code Coverage using cli

## Generate Token: -

After setting up SonarQube need to generate a token to perform analysis, follow the steps listed below.

* Go to SonarQube dashboard

**Note:** If you are using local SonarQube (<http://localhost:9000/>)

If you are using Central SonarQube (<https://sonar.cybage.com/>)

* Login with SonarQube credential
* On top-right, click on your user icon
* Then click “**My Account**”
* Now click on “**Security**” tab
* Give a name to your token and click “**Generate**”
* You will get a randomly generated token
* Copy that token and use it in command

**Note: -** Once the page is refresh, you will not get back the token value, so make sure you are keeping that token safely somewhere. You can anytime revoke old token and regenerate new token as well.

## Prerequisites:

* Install SonarScanner
  + Run cmd as administration
  + Run below command

(dotnet tool install --global dotnet-sonarscanner)

**(Note:** Run above command to install SonarScanner for .net core globally)

## Performing sca using cli:

* Now prepare below commands as per your project

1. dotnet sonarscanner begin /k:" ProjectKey" /n:" ProjectName" /v:"1.0" /d:sonar.host.url=hostURL /d:sonar.login="paste your token here"
2. dotnet build example.sln
3. dotnet sonarscanner end /d:sonar.login="paste your token here"

**(Note:** Set the above highlighted things according to your requirement)

* + Now to perform analysis, open cmd from your project directory where the .csproj file is present or you can open cmd prompt and set working directory to project directory
  + Make sure sonar-project.properties file is not present in project directory
  + Now run all the above commands one by one to start analysis

## Performing code coverage using cli:

To perform code coverage, we need to install coverlet using below command. Open cmd paste the command.

**dotnet tool install --global coverlet.console**

Coverlet is a cross-platform that calculates code coverage for C#. Coverlet is part of the .NET Foundation. Coverlet collects Cobertura coverage test run data, which is used for report generation.

**Commands**:

1. Coverlet “D:\.NETCorecodes\ASP\_Core\New\_Eshop\_Project\tests\UnitTests\bin\Debug\net6.0\UnitTests.dll" --target:"C:\Program Files\Microsoft Visual Studio\2022\Professional\Common7\IDE\CommonExtensions\Microsoft\TestWindow\vstest.console.exe" --targetargs:"D:\.NETCorecodes\ASP\_Core\New\_Eshop\_Project\tests\UnitTests\bin\Debug\net6.0\UnitTests.dll" --format:"opencover" --output:"%CD%\opencover.xml"

(**Note**: Once it is completed successfully, go to your project directory and you can find opencover.xml and need to mention that path in 2 command.)

1. dotnet sonarscanner begin /k:" ProjectKey" /n:" ProjectName" /v:"1.0" /d:sonar.host.url=hostURL /d:sonar.login="paste your token here" /d:sonar.cs.opencover.reportsPaths="path of opencover.xml"
2. dotnet build example.sln
3. dotnet sonarscanner end /d:sonar.login="paste your token here"

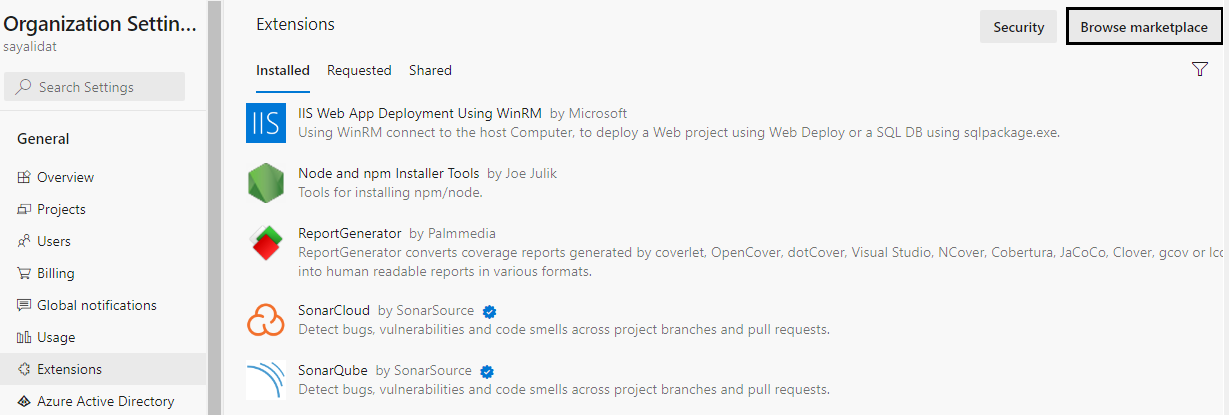
**(Note**: - Set the above highlighted things according to your requirement)

* + Now to perform analysis, open cmd from your project directory where the .csproj file is present or you can open cmd prompt and set working directory to project directory
  + Make sure sonar-project.properties file is not present in project directory
  + Now run all the above commands one by one to start analysis

# sca and Code Coverage using Azure Pipeline

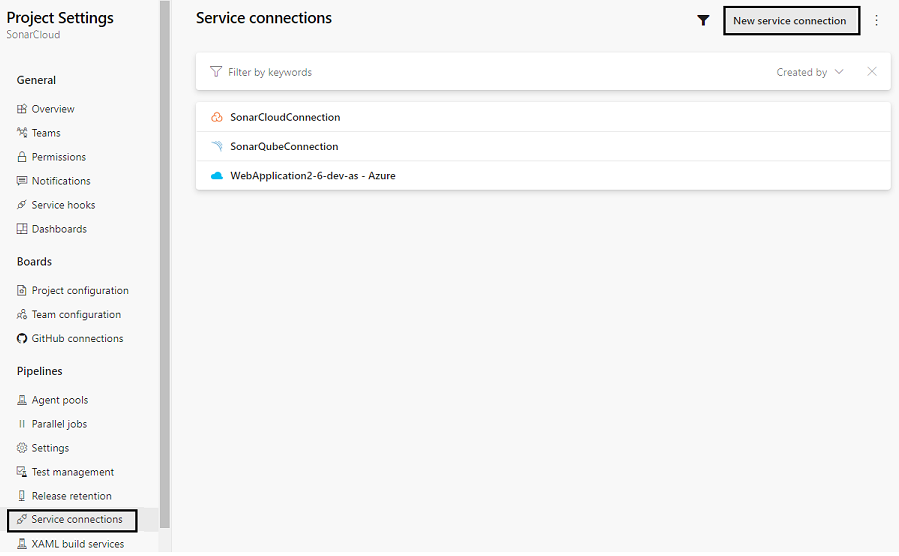
## prerequisites:

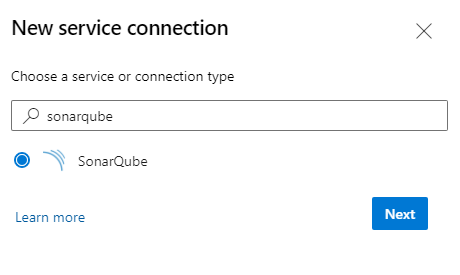
* **Download Extension** : Now navigate to your Azure DevOps Services 🡪 Organization settings 🡪 Extension 🡪 Browse marketplace 🡪 Install the SonarQube and Report Generator extension.

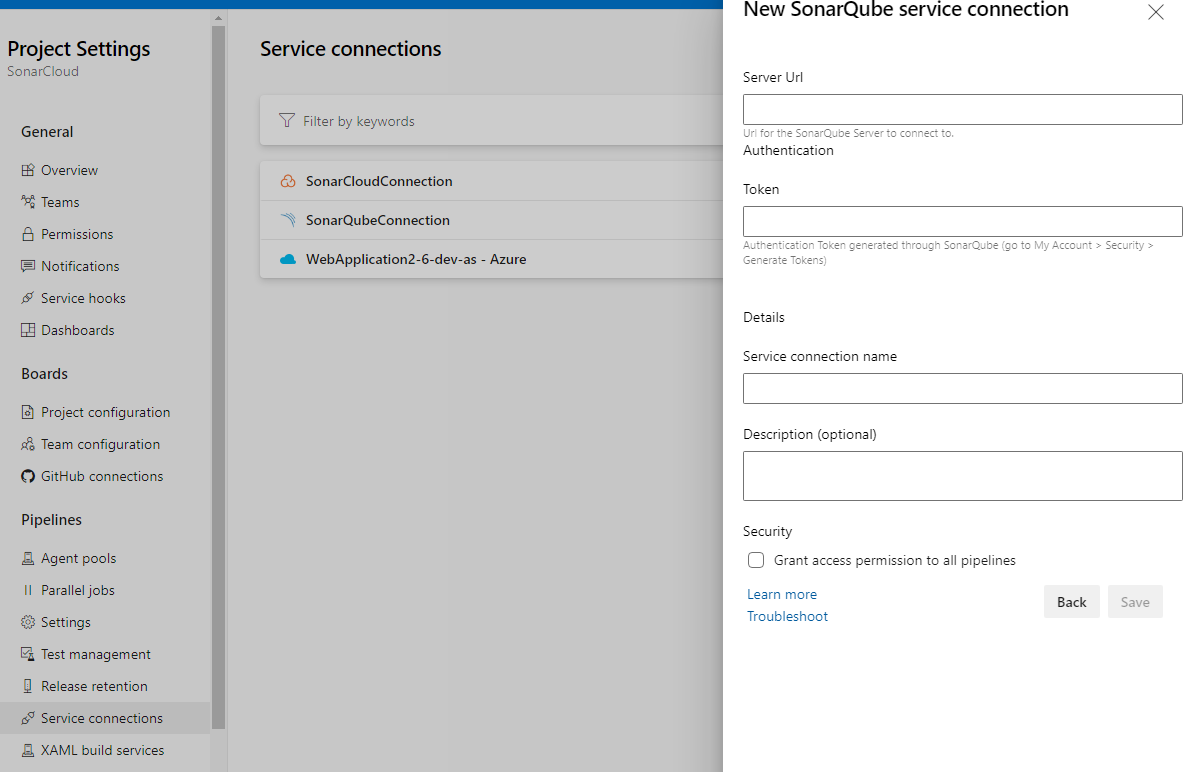


## create a service connection:

* **Create Service Connection** : Navigate to project settings of your project 🡪 Service connections 🡪 Click on New service connection 🡪 Select SonarQube from that list 🡪 Paste Url for the SonarQube Server to connect to 🡪 Create SonarToken on SonarQube and paste it 🡪 Give service connection name 🡪 Check mark Grant the access to all pipelines🡪 Click on Save.







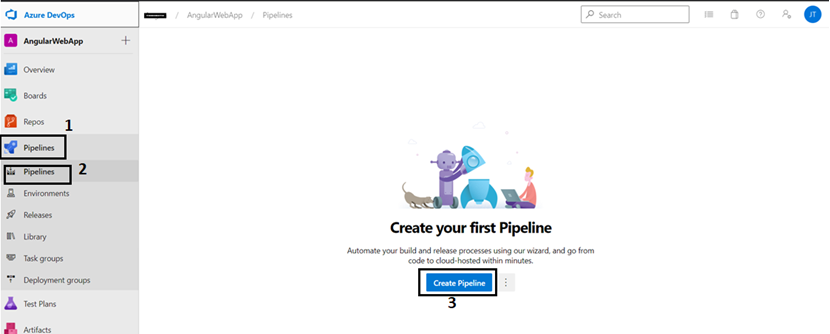
Server URL = Provide the SonarQube server URL

Token = Need to generate token on SonarQube server and paste it ([refer 1.1](#_Generate_Token:_-) for generating token).

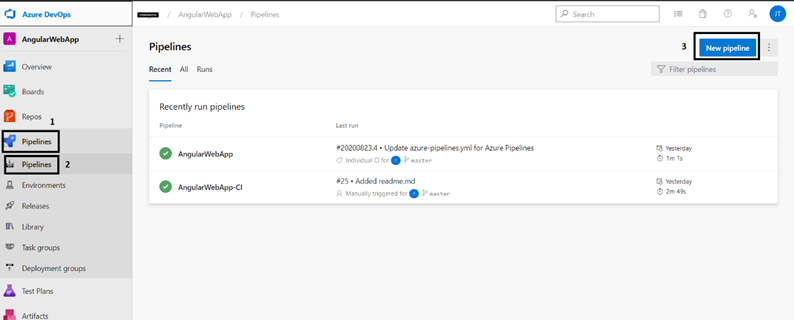
Service Connection Name = we can provide any random name it will be used in further steps.

## create a pipeline:

1. Login to Azure DevOps and select your project that you want to create build/release pipeline
2. To create a new build pipeline, click on Pipelines >> Pipelines >> Create Pipeline. You can see the following screen when you create a pipeline for any repo first time.

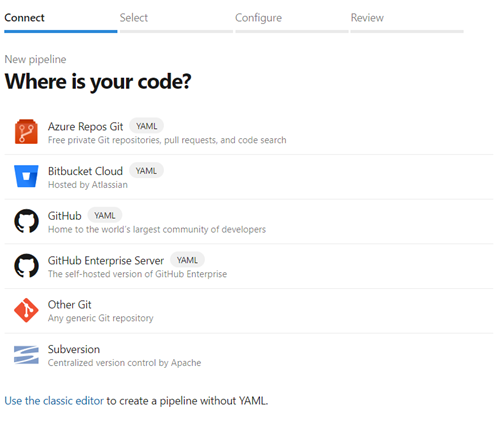


Subsequently, to create a new build pipeline, click on Pipelines >> Pipelines >> New Pipeline

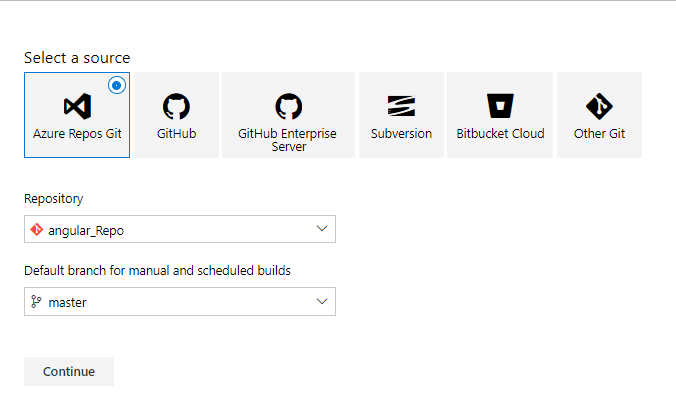


1. When you click on the “New Pipeline” or “Create pipeline” button, the following screen is appearing. Here, you have multiple choices to create DevOps build pipeline with.

The simplest way to create a pipeline is creating with “Classic editor”. In this article, I am going to explain with classic editor however it is very similar to “Azure Repos Git (YAML). You can also view an equivalent YAML script for all tasks defined in the editor.



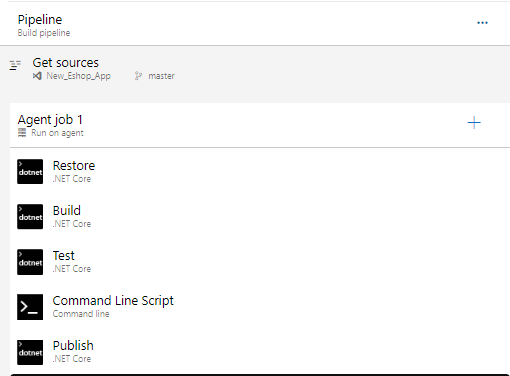
1. Click on Use the classic editor and then it will display below window for selecting the repository and branch.



1. For creating a dot net core pipeline, you need to select the different template for getting all the task which related with the dot net core as mentioned below.

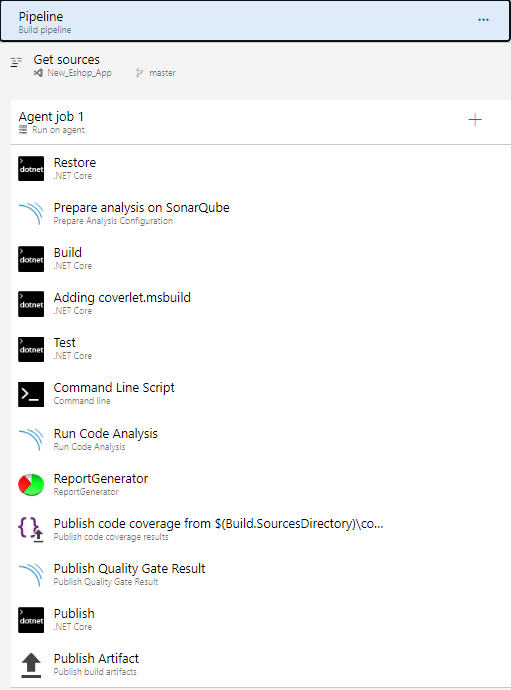


1. Once the template is added in the pipeline then it will add all the related task in the pipeline.



When you have added the task to the pipeline, configure it, click save, and then run the pipeline.

1. In order to obtain the analysis report and code coverage report, you need to include the SonarQube task and the appropriate position in the pipeline.



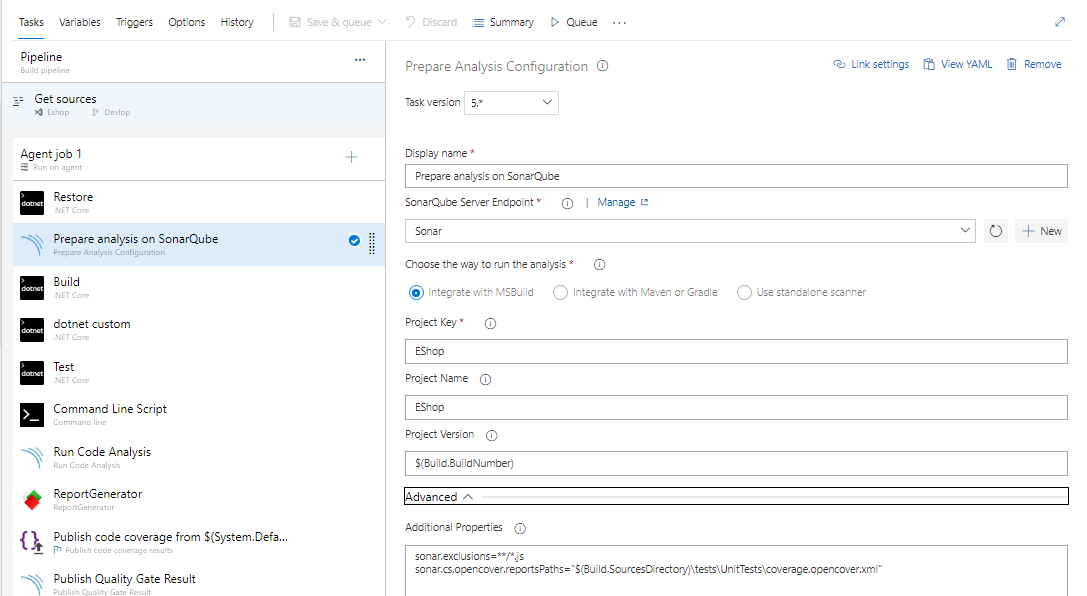
* AddSonarQube task **Prepare Analysis** and configure it
* Select the **SonarQube Server Endpoint** which we created from project settings.
* Provide **Project Key, Project Name** and **Project Version**.
* In **Additional Properties** add below two commands to generate opencover file

sonar.exclusions=\*\*/\*.js

**(Note:** To exclude any files or folder we have to use the above property)

* sonar.cs.opencover.reportsPaths="$(Build.SourcesDirectory)\opencover.xml"

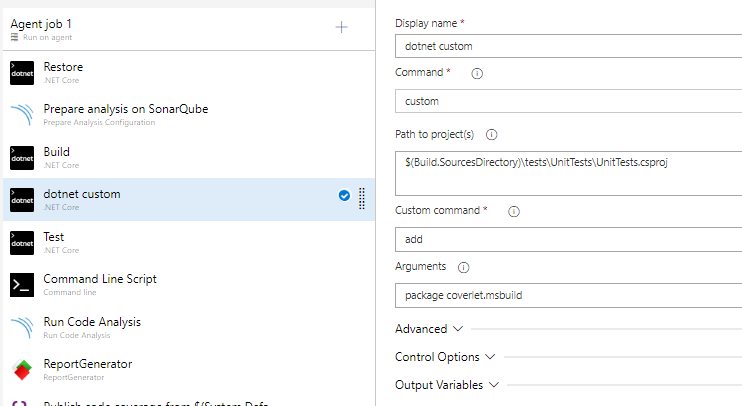
**(Note**: If you want to calculate code coverage need to mention above command)



* Custom task

This task is used to install coverlet which is used for calculating code coverage

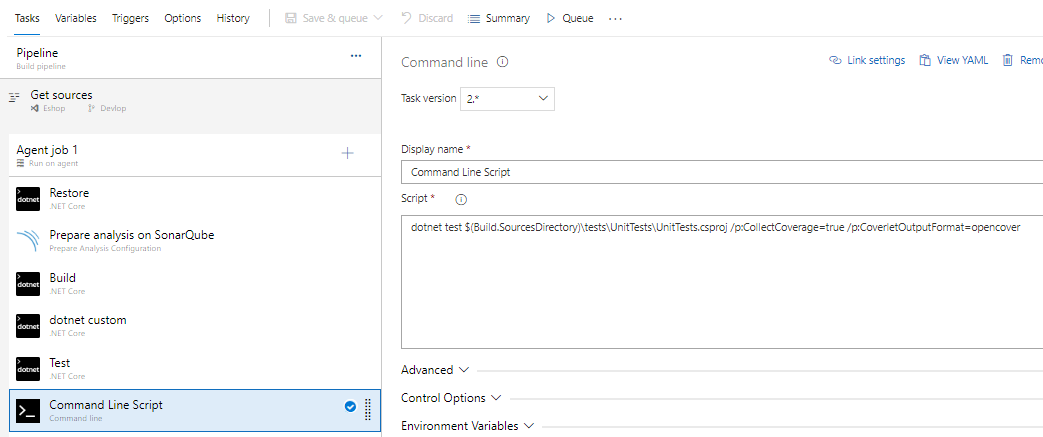
* Command: package coverlet.msbuild



* Add **Command Line Script** and configuration it.

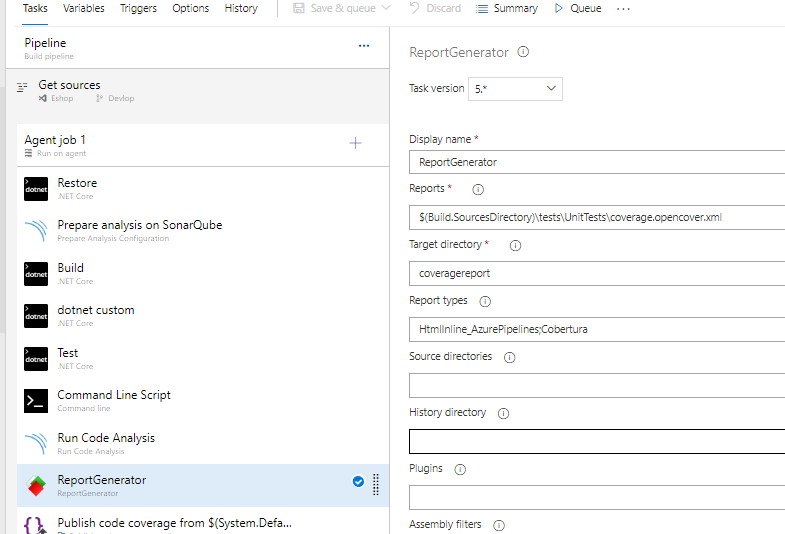
This task is used to calculate code coverage. For calculating code coverage unit test cases should be written in your project.

* Command : dotnet test $(Build.SourcesDirectory)\tests\UnitTests\UnitTests.csproj /p:CollectCoverage=true /p:CoverletOutputFormat=opencover

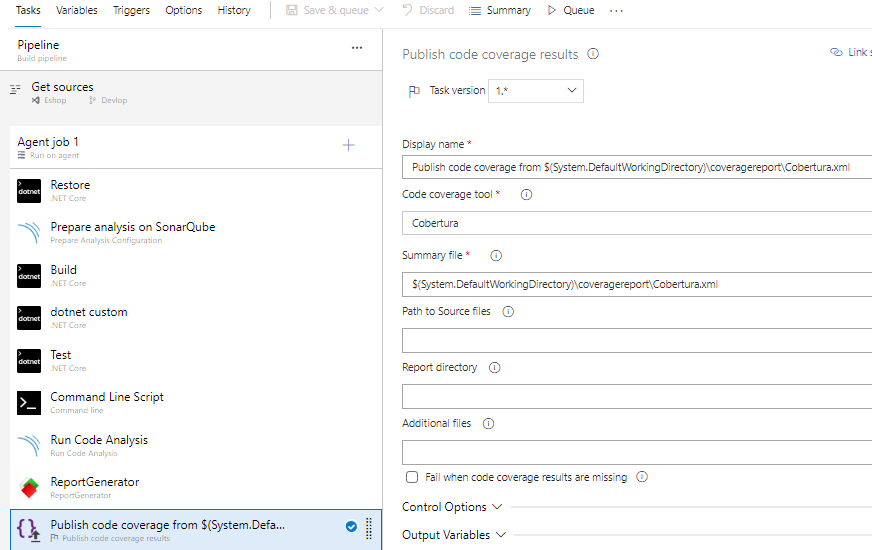


**(Note: -** Set the above highlighted things according to your requirement. Configure the above mentioned commands according to your system path)

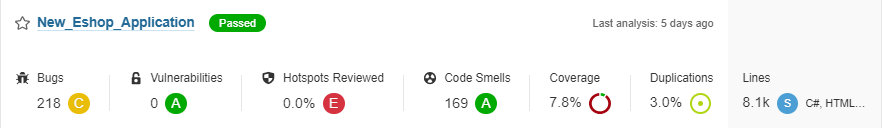
* Add **Run Code Analysis** task.
* Add **Publish Quality Gate Result** task.
* Add **Report Generator** task and configure the highlighted fields.



* Add **Publish code coverage results** task and configure the highlighted fields.



1. Click on **Save & queue**.
2. After successful CI run, report will generate with code coverage on SonarQube dashboard.



# References

<https://docs.sonarqube.org/9.6/requirements/prerequisites-and-overview/>

<https://docs.sonarqube.org/9.6/analyzing-source-code/scanners/sonarscanner-for-dotnet/>