1. **SonarQube**

Is an open-source platform developed by SonarSource for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities on 20+ programming languages. SonarQube offers reports on duplicated code, coding standards, unit tests, code coverage, code complexity, comments, bugs, and security vulnerabilities.

1. **SonarScanner for MSBuild**

The SonarScanner for MSBuild is the recommended way to launch an analysis for projects/solutions using MSBuild or dotnet command as a build tool. It is the result of a collaboration between SonarSource and Microsoft.

SonarScanner for MSBuild is distributed as a standalone command line executable, as a extension for Azure DevOps Server, and as a plugin for Jenkins.

It supports .Net Core multi-platform projects and it can be used on non-Windows platforms.

Installation guide: <https://docs.sonarqube.org/latest/analysis/scan/sonarscanner-for-msbuild/>

1. **SonarCube with Opencover .Net Framework – IFRS9 On-Perm**

Source**:** <https://maartenderaedemaeker.be/2017/11/05/nunit-opencover-results-in-sonarqube/>

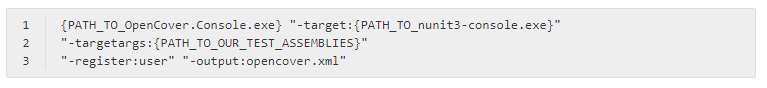
Install OpenCover and NUnit.ConsoleRunner nuget packages as given below.  


We are interested in the following two executables included in these packages:

* {OpenCoverFolder}\tools\OpenCover.Console.exe
* {NUnitConsoleFolder}\tools\nunit3-console.exe

Use OpenCover to run the NUnit console runner to measure our code coverage.

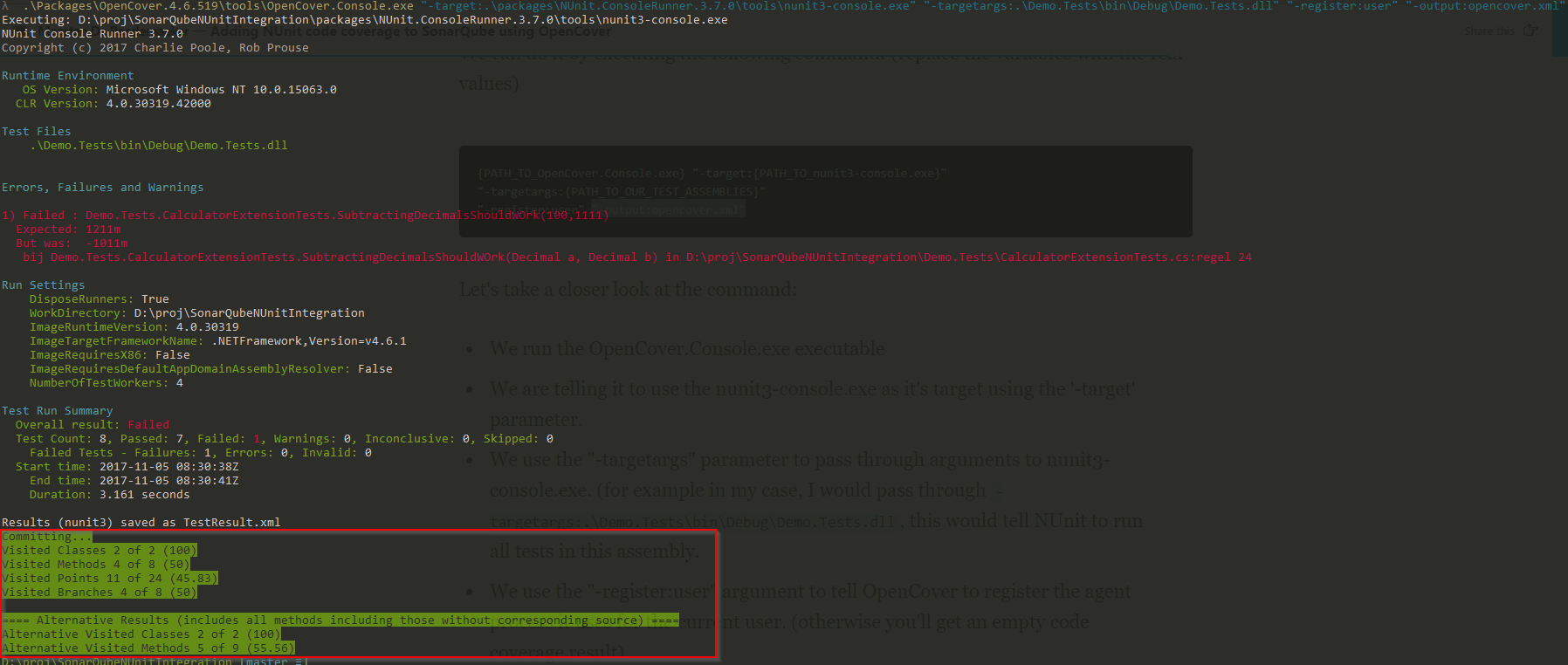
We can do it by executing the following command: (replace the variables with the real values)



Let’s take a closer look at the command:

* Run the OpenCover.Console.exe executable
* Use the nunit3-console.exe as it’s target using the -target parameter.
* Use the -targetargs parameter to pass through arguments to nunit3-console.exe. (for example in my case, I would pass through -targetargs:.\Demo.Tests\bin\Debug\Demo.Tests.dll, this would tell NUnit to run all tests in this assembly.
* Use the -register:user argument to tell OpenCover to register the agent process it uses for the current user. (otherwise you’ll get an empty code coverage result)
* Specify the -output parameter to choose a file where the OpenCover results get written to.

When running the command, NUnit running the test and the code coverage results being written.

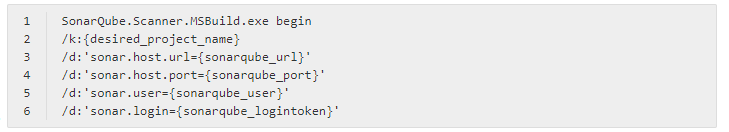


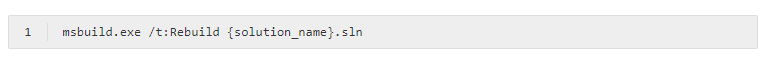
**Adding test coverage results to SonarQube**

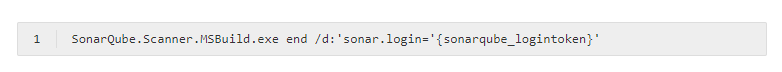
The general steps for running a SonarQube scan on a .NET project are the following:

1. Setup the SonarQube scanner for msbuild - using the begin argument (retrieve rules from SonarQube instance, inject analyzers in build pipeline)
2. Run msbuild
3. Use the SonarQube scanner for msbuild to generate and submit the report - using the end argument

This would give the following corresponding commands (replace the variables with your parameters): Now generate a code coverage result file, we can make SonarQube aware of where to pick up the results.

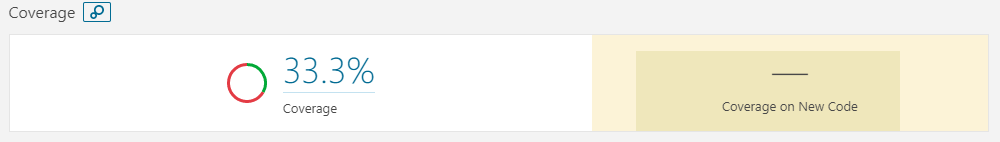






We can do this by modifying step 1 and specifying the sonar.cs.opencover.reportsPaths property in our SonarQube scanner command like this:

SonarQube will pick up our code coverage and show it in our projects dashboard:



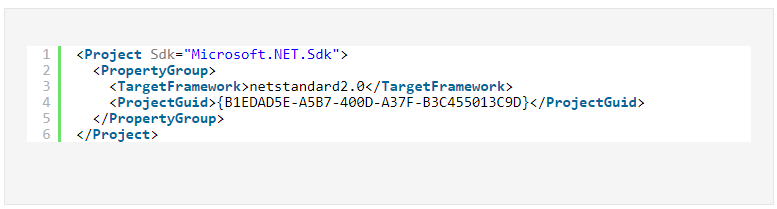
1. **Azure, .NET CORE, SONARQUBE and Opencover**

Source 1: <https://writeabout.net/2018/09/24/net-core-sonarqube-and-code-coverage/>

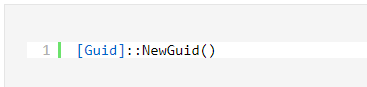
Source 2: <https://writeabout.net/2019/04/27/net-core-code-coverage-done-right/>

1. Modify .csproj file

First of all you need to modify the project files you want to analyse. You have to add a ‘ProjectGuid’ element. This is a new guid.

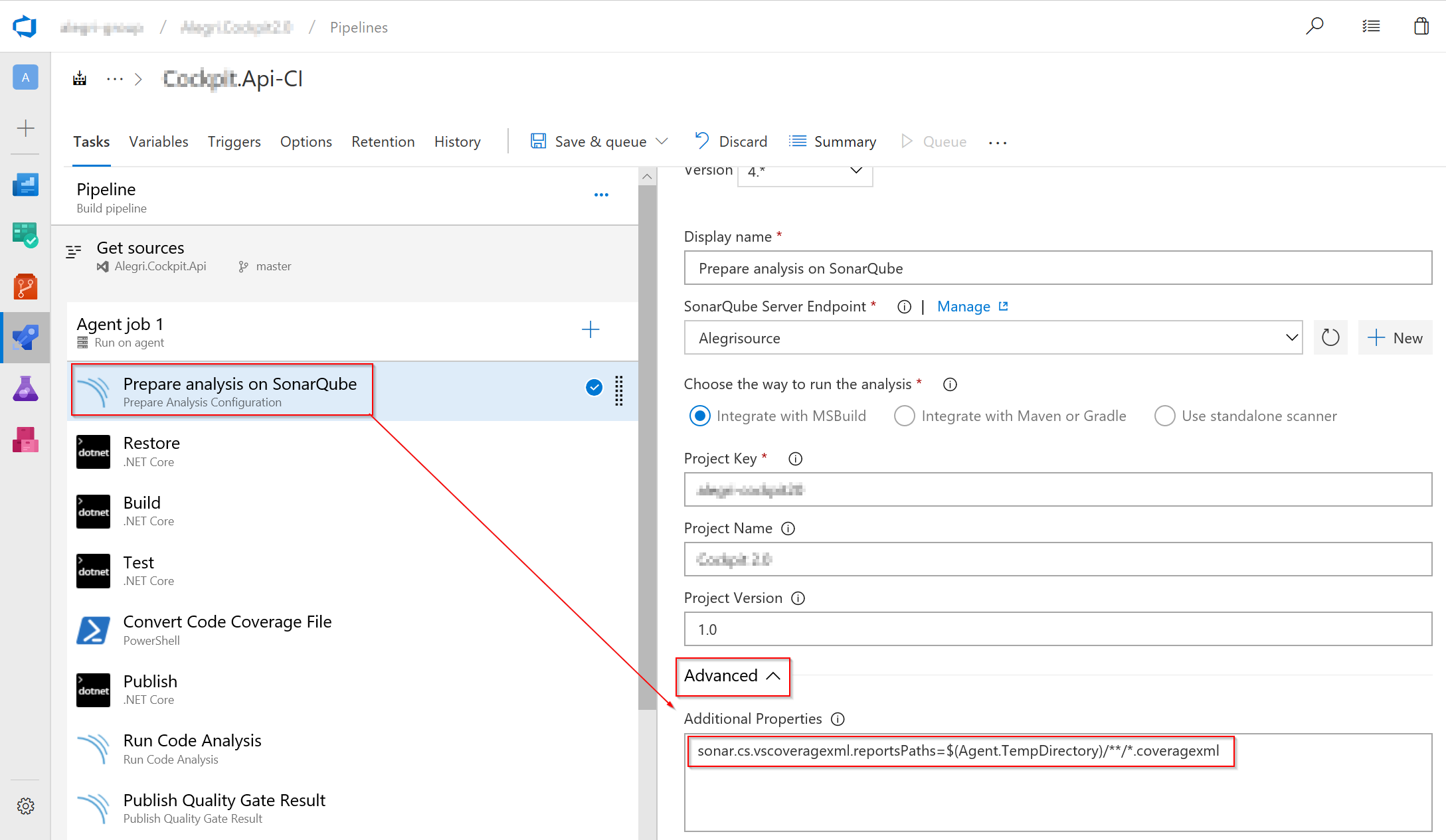


You can generate it with the following line:



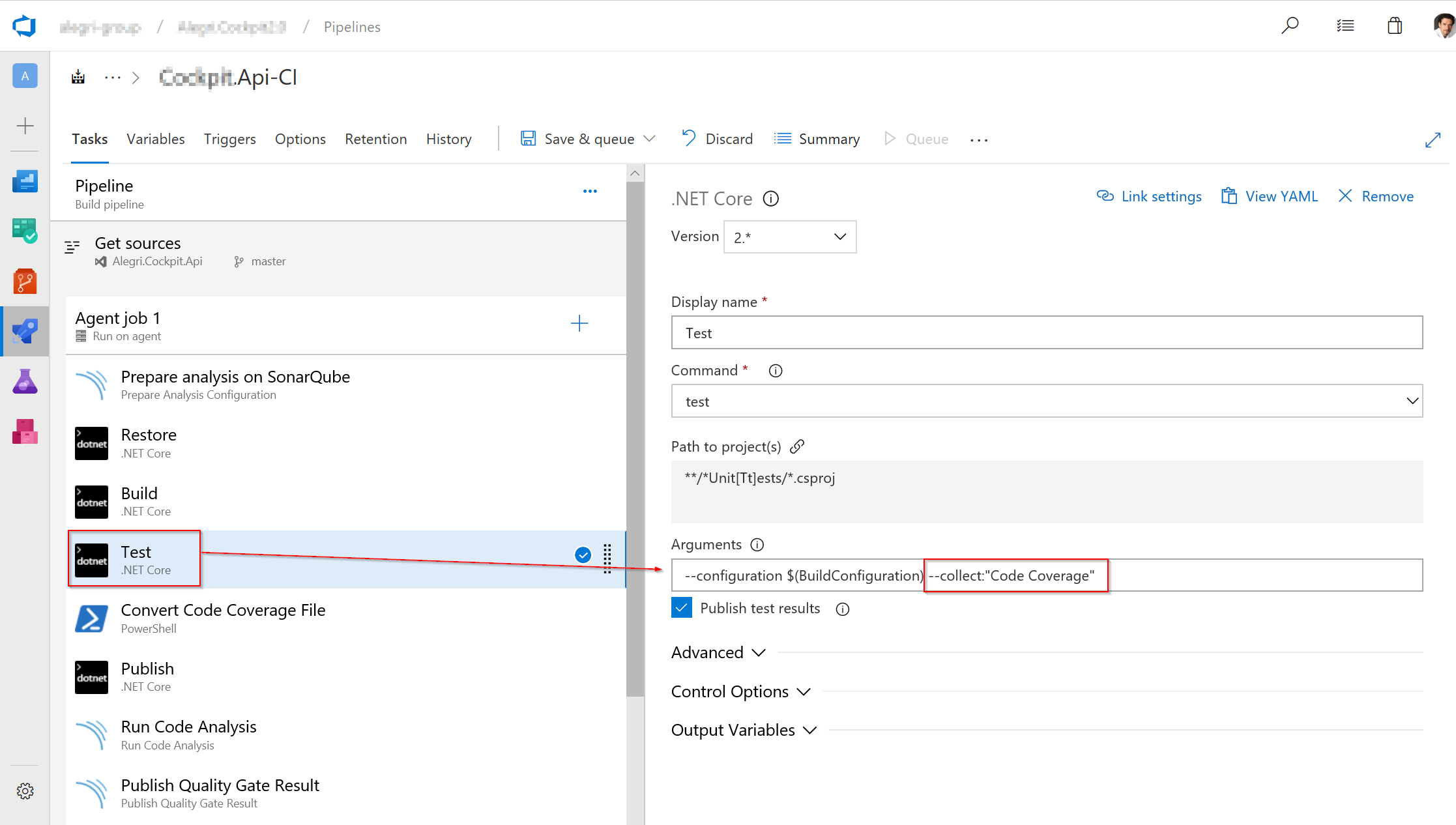
### Add “Prepare analysis on SonarQube” task to your pipeline

Add the task to pipeline and configure your endpoint. For the code coverage to work have to add the following attribute under Advanced:



### Configure test task

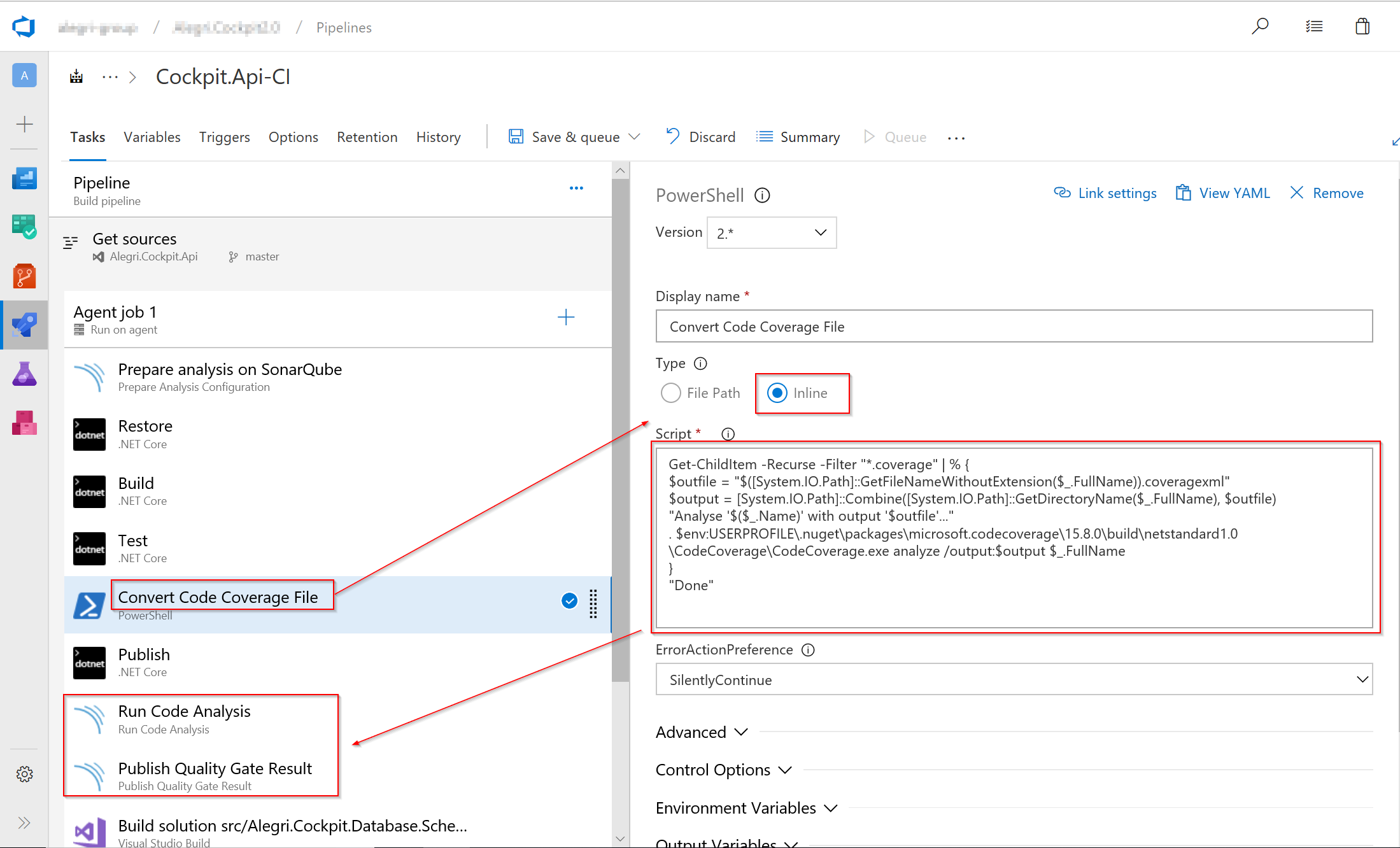
In the test task you have to add –collect:”Code Coverage” for the task to add a logger for code coverage.



### Convert Code Coverage Files

This is the tricky part. The test task only generates .coverage files for each test project. But SonarQube needs a .coveragexml and does not understand the .coverage file format. To convert the file you have to call CodeCoverage.exe with the (undocumented) parameter /analyse. I’ve created a PowerShell script for that.

You can add this script in a PowerShell task as a inline script. Make sure the WorkingDirectory is the $(Agent.TempDirectory).



.NET Core analysis should now show up in SonarQube including Code Coverage.

