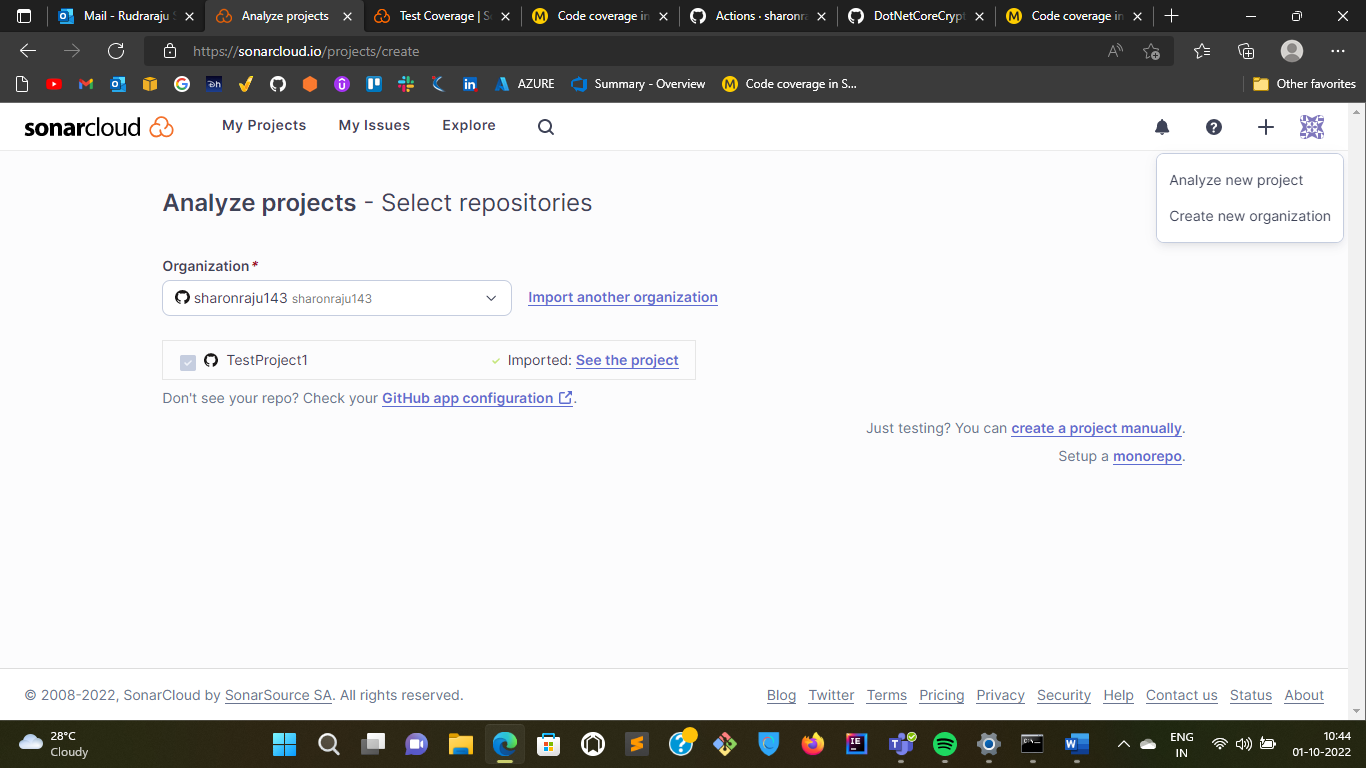
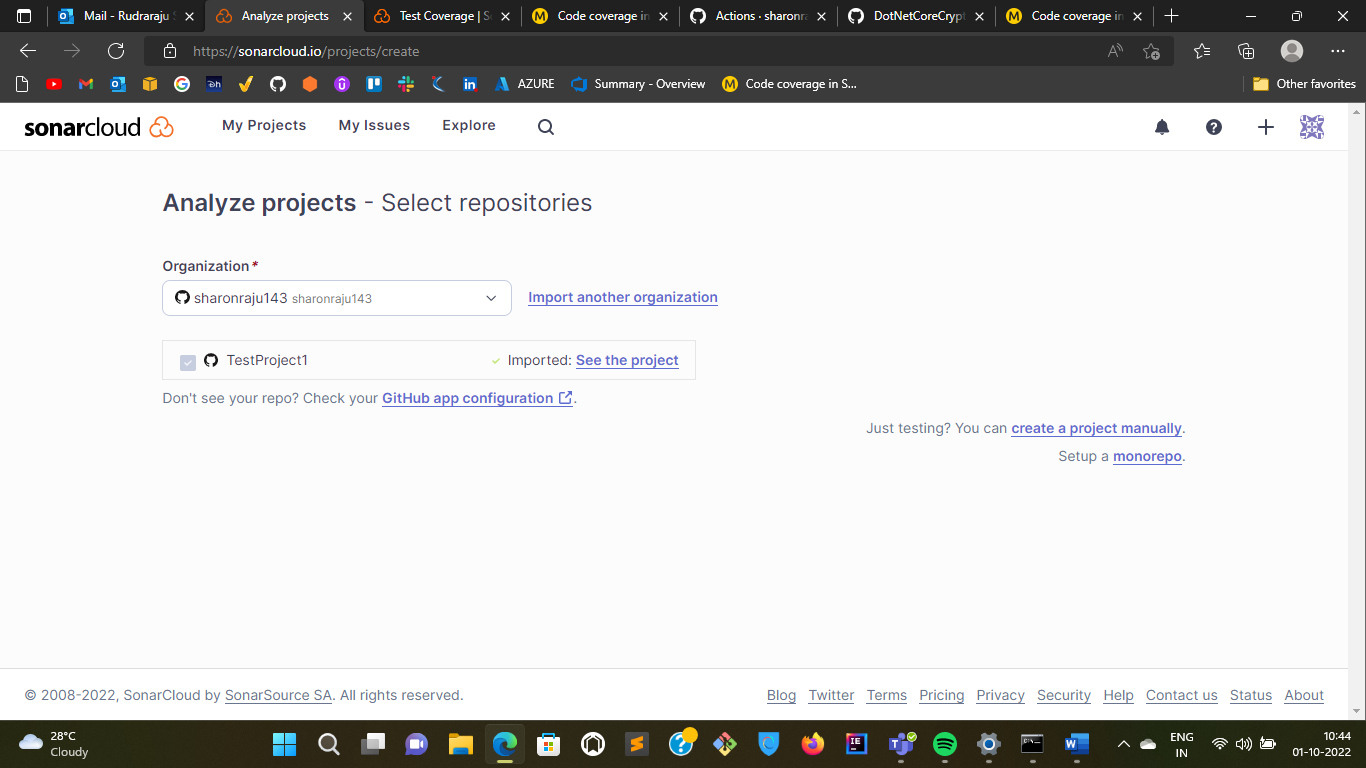
Steps to analyse .net code using Sonarcloud:

1. First get the code to be analysed into GitHub as we using GitHub to analyse the code in sonarcloud.
2. Login to Sonarcloud and click on Analyze new project



1. Select the repository that you want to analyse from Github



1. Once you select the repository follow the instructions that appear on the sonarcloud account.

* **Create a new file in the Github with the following code “.github/workflows/build.yml”**

name: Build

on:

push:

branches:

- main

pull\_request:

types: [opened, synchronize, reopened]

jobs:

build:

name: Build

runs-on: windows-latest

steps:

- name: Set up JDK 11

uses: actions/setup-java@v1

with:

java-version: 1.11

- uses: actions/checkout@v2

with:

fetch-depth: 0 # Shallow clones should be disabled for a better relevancy of analysis

- name: Cache SonarCloud packages

uses: actions/cache@v1

with:

path: ~\sonar\cache

key: ${{ runner.os }}-sonar

restore-keys: ${{ runner.os }}-sonar

- name: Cache SonarCloud scanner

id: cache-sonar-scanner

uses: actions/cache@v1

with:

path: .\.sonar\scanner

key: ${{ runner.os }}-sonar-scanner

restore-keys: ${{ runner.os }}-sonar-scanner

- name: Install SonarCloud scanner

if: steps.cache-sonar-scanner.outputs.cache-hit != 'true'

shell: powershell

run: |

New-Item -Path .\.sonar\scanner -ItemType Directory

dotnet tool update dotnet-sonarscanner --tool-path .\.sonar\scanner

- name: Build and analyze

env:

GITHUB\_TOKEN: ${{ secrets.GITHUB\_TOKEN }} # Needed to get PR information, if any

SONAR\_TOKEN: ${{ secrets.SONAR\_TOKEN }}

shell: powershell

run: |

.\.sonar\scanner\dotnet-sonarscanner begin /k:"sharonraju143\_TestProject1" /o:"sharonraju143" /d:sonar.login="${{ secrets.SONAR\_TOKEN }}" /d:sonar.host.url="https://sonarcloud.io"

dotnet build

.\.sonar\scanner\dotnet-sonarscanner end /d:sonar.login="${{ secrets.SONAR\_TOKEN }}"

**Code coverage in SonarCloud and GitHub Actions:**

1. As first step I’ve included a .NET core tool configuration file in .config/dotnet-tool.json directory, with the following content:

{

"version": 1,

"isRoot": true,

"tools": {

"gitversion.tool": {

"version": "5.2.4",

"commands": [

"dotnet-gitversion"

]

},

"dotnet-sonarscanner": {

"version": "5",

"commands": [

"dotnet-sonarscanner"

]

}

}

}

This file allows me to fix the exact version of all .NET core related tool that I need to use in this repository. As you can see I’ve included both GitVersion and dotnet-sonarscanner.

1. This is the complete script that analyze my project. It has sonarSecret as the only parameter, it does not need anything else. You can use this PowerShell file to launch an analysis in your Machine without committing anything. It is not a good practice to analyze uncommitted code, but it is really good for troubleshooting.

Here is the complete content of the file:

param(

[string] $sonarSecret

)

Install-package BuildUtils -Confirm:$false -Scope CurrentUser -Force

Import-Module BuildUtils

$runningDirectory = Split-Path -Parent -Path $MyInvocation.MyCommand.Definition

$testOutputDir = "$runningDirectory/TestResults"

if (Test-Path $testOutputDir)

{

Write-host "Cleaning temporary Test Output path $testOutputDir"

Remove-Item $testOutputDir -Recurse -Force

}

$version = Invoke-Gitversion

$assemblyVer = $version.assemblyVersion

$branch = git branch --show-current

Write-Host "branch is $branch"

dotnet tool restore

dotnet tool run dotnet-sonarscanner begin `

/k:"alkampfergit\_DotNetCoreCryptography" ` # Key of the project

/v:"$assemblyVer" ` # Version of the assemly as calculated by gitversion

/o:"alkampfergit-github" ` # account

/d:sonar.login="$sonarSecret" ` # Secret

/d:sonar.host.url="https://sonarcloud.io" `

/d:sonar.cs.vstest.reportsPaths=TestResults/\*.trx ` # Path where I'm expecting to find test result in trx format

/d:sonar.cs.opencover.reportsPaths=TestResults/\*/coverage.opencover.xml ` # Name of the code coverage file

/d:sonar.coverage.exclusions="\*\*Test\*.cs" ` # asembly names to be excluded from code coverage

/d:sonar.branch.name="$branch" # Actual branch I'm analyzing.

dotnet restore src

dotnet build src --configuration release

# Now execute tests with special attention to produce output

# that can be easily read by SonarCloud analyzer

dotnet test "./src/DotNetCoreCryptography.Tests/DotNetCoreCryptography.Tests.csproj" `

--collect:"XPlat Code Coverage" ` # cross platform code coverage

--results-directory TestResults/ ` # Test Result directory

--logger "trx;LogFileName=unittests.trx" ` # Use standard trx format for logger output

--no-build `

--no-restore `

--configuration release `

-- DataCollectionRunSettings.DataCollectors.DataCollector.Configuration.Format=opencover # Special open cover data collector

dotnet tool run dotnet-sonarscanner end /d:sonar.login="$sonarSecret"

1. The following code enables the code coverage of your code:

param(

[string] $sonarSecret

)

Install-package BuildUtils -Confirm:$false -Scope CurrentUser -Force

Import-Module BuildUtils

$runningDirectory = Split-Path -Parent -Path $MyInvocation.MyCommand.Definition

$testOutputDir = "$runningDirectory/TestResults"

if (Test-Path $testOutputDir)

{

Write-host "Cleaning temporary Test Output path $testOutputDir"

Remove-Item $testOutputDir -Recurse -Force

}

$version = Invoke-Gitversion

$assemblyVer = $version.assemblyVersion

$branch = git branch --show-current

Write-Host "branch is $branch"

dotnet tool restore

dotnet tool run dotnet-sonarscanner begin /k:"alkampfergit\_DotNetCoreCryptography" /v:"$assemblyVer" /o:"alkampfergit-github" /d:sonar.login="$sonarSecret" /d:sonar.host.url="https://sonarcloud.io" /d:sonar.cs.vstest.reportsPaths=TestResults/\*.trx /d:sonar.cs.opencover.reportsPaths=TestResults/\*/coverage.opencover.xml /d:sonar.coverage.exclusions="\*\*Test\*.cs" /d:sonar.branch.name="$branch"

dotnet restore src

dotnet build src --configuration release

dotnet test "./src/DotNetCoreCryptography.Tests/DotNetCoreCryptography.Tests.csproj" --collect:"XPlat Code Coverage" --results-directory TestResults/ --logger "trx;LogFileName=unittests.trx" --no-build --no-restore --configuration release -- DataCollectionRunSettings.DataCollectors.DataCollector.Configuration.Format=opencover

dotnet tool run dotnet-sonarscanner end /d:sonar.login="$sonarSecret"

1. Now see the results in Sonarcloud

