Agenda: Scan Code During CI

- Sources and Impacts of Technical Dept
- Managing Technical Dept with DevOps and Sonar Cloud
- Scan open-source components using WhiteSource Bolt

What is Ruged DevOps?

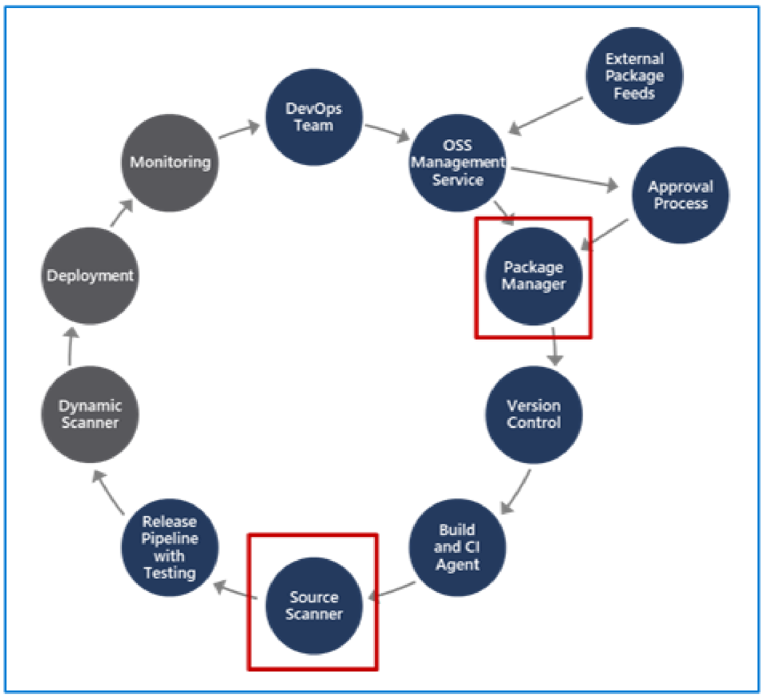
- Rugged DevOps is a set of practices designed integrate DevOps and security.
- The goal is to enable development teams to work fast without introducing unwanted vulnerabilities.
- Security strategy includes access control, environment hardening, perimeter protection, and more…

Rugged DevOps includes bigger questions such as:

- Is my pipeline consuming third-party components, and if so, are they secure?
- Are there known vulnerabilities within any of the third-party software we use?
- How quickly can I detect vulnerabilities (*time to detect*)?
- How quickly can I remediate identified vulnerabilities (*time to remediate*)?

Rugged DevOps Pipeline:

- **Source Scanner** is for performing a security scan to verify certain security vulnerabilities are not present in our application source code.
- **Package Management,** and the approval process associated with it, accounts for how software packages are added to the pipeline, and the approval process they need to go through.



Sources and Impacts of Technical Debt

Technical Debt is a term that describes the **future penalty** that you incur today by making easy or quick choices in software development practices, rather than taking harder or longer choices that are more appropriate longer-term. Over time, it accrues in much the same way that monetary debt does.

Common sources of technical debt are:

- Lack of coding style and standards.
- Insufficient comments and documentation.
- Not writing self-documenting code (including class, method and variable names that are descriptive or indicate intent).
- Lack of or poor design of unit test cases.
- Ignoring or not understanding object orient design principles.
- Monolithic classes and code libraries.
- Poorly envisioned use of technology, architecture and approach. (Forgetting that all attributes of the system, affecting maintenance, user experience, scalability, and others, need to be considered).
- Over-engineering code (adding or creating code that is not needed, adding custom code when existing libraries are sufficient, or creating layers or components that are not needed).
- Taking shortcuts to meet deadlines.
- Leaving dead code in place.

Over time, technical debt must be paid back. Otherwise, the team's ability to fix issues, and to implement new features and enhancements will take longer and longer, and eventually become cost-prohibitive.

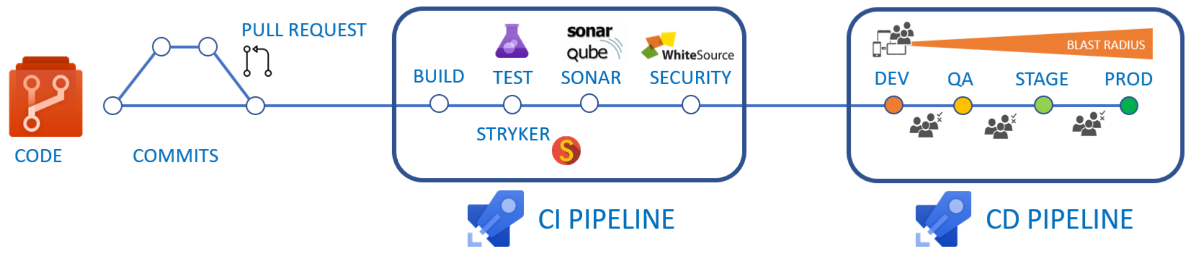
Using Automated Testing to Measure Technical Debt

- We have seen that technical debt adds a set of problems during development and makes it much more difficult to add additional customer value.
- Having technical debt in a project saps productivity, frustrates development teams, makes code both hard to understand and fragile, increases the time to make changes, and to validate those changes. Unplanned work frequently gets in the way of planned work.
- Longer term, it also saps the organization's strength. Technical debt tends to creep up on an organization. It starts small, and grows over time. Every time a quick hack is made, or testing is circumvented because changes needed to be rushed through, the problem grows worse and worse. Support costs get higher and higher, and invariably, a serious issue arises.
- Eventually, the organization cannot respond to its customers' needs in a timely and cost efficient way.

Automated Measurement for Monitoring

One key way to minimize the constant acquisition of technical debt, is to use automated testing and assessment.

SonarCloud is a cloud service offered by SonarSource and based on SonarQube. SonarQube is a widely adopted open source platform to inspect continuously the quality of source code and detect bugs, vulnerabilities and code smells in more than 20 different languages.



Managing Technical Debt with Azure DevOps and SonarCloud

Step1: Sonar Project and Security Token

1. Go to https://sonarcloud.io and Login with Azure DevOps account
2. MyProjects → Analyze new project → Import a new organization
    a. Azure DevOps organization name = <Name of your DevOps Organization>
    b. Personal Access Token: <Generated PAT Token in Azure DevOps Portal>
    c. Continue
    d. Select Free plan → Create Organization
3. Select Your Azure DevOps Project in which we want to use Sonar Cloud. → Setup
4. Select With Azure DevOps Pipelines (Recommended)

Step 2: Add a new SonarCloud Service Endpoint in DevOps Project

1. Azure DevOps → Project → **Marketplace** → **Browse Marketplace**.
2. Search for **"SonarCloud"** → Select the **SonarCloud** → . . . → Proceed to Organization
3. Go to **Project settings > Service connections**
4. Add a new service connection of the type SonarCloud
5. Use this token: ce3983468d6d765fc74dd2923bd86221b5bf6f3d
6. Click on **Verify** to check that everything is linked correctly.

Step3: Integrating a build with SonarCloud

1. Create a **New Pipeline (Classic)** and use the template **.NET Core with SonarCloud. This will add 3 steps related to Sonar Cloud**
    a. Prepare analysis on Sonar Cloud
        i. Select the SonarCloud endpoint.
        ii. Select the SonarCloud organization DevOpsDemoOrg

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

iv. In the Project Key field, enter DevOpsDemoOrg_HelloWorldApp

v. In the Project Name field, enter HelloWorldApp

b. Run Code Analysis: This task needs to run after your build step.

c. Publish Quality Gate Result*

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

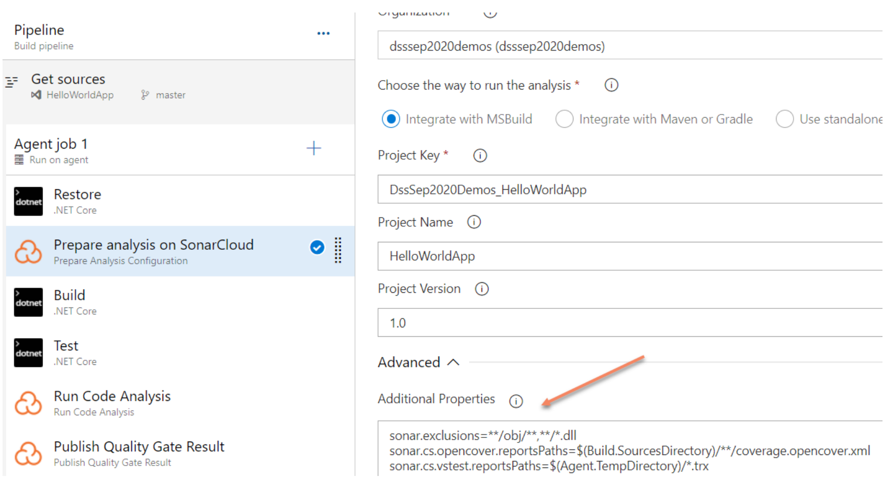
ii. If you plan not to use such a feature, you can omit it. Be aware that this task may increase your build time.

2. Use **Windows Agent for the Job**
3. **Add the following to Additional Section of Prepare analysis on SonarCloud**

```
sonar.exclusions=**/obj/**,**/*.dll
sonar.cs.opencover.reportsPaths=$(Build.SourcesDirectory)/**/coverage.opencover.xml
sonar.cs.vstest.reportsPaths=$(Agent.TempDirectory)/*.trx
```

4. Add the following arguments to the Test task

```
--configuration $(BuildConfiguration) /p:CollectCoverage=true /p:CoverletOutputFormat=opencover --logger trx
```



Organization: dsssep2020demos (dsssep2020demos)

Choose the the way to run the analysis *: (● Integrate with MSBuild) ○ Integrate with Maven or Gradle ○ Use standalone

Project Key *: DssSep2020Demos_HelloWorldApp

Project Name: HelloWorldApp

Project Version: 1.0

Advanced

Additional Properties:

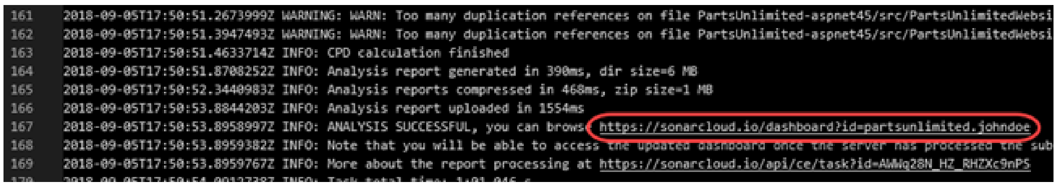
```
sonar.exclusions=**/obj/**,**/*.dll
sonar.cs.opencover.reportsPaths=$(Build.SourcesDirectory)/**/coverage.opencover.xml
sonar.cs.vstest.reportsPaths=$(Agent.TempDirectory)/*.trx
```

Step 4: Configuring Sonar Qube

Goto Sonar Cloud Website → Select Project → Administration → New Code → Select Previous Version

Step 5: Run the Pipeline and review the report in the Sonar Cloud Website

1. From the left panel, select the **Run Code Analysis** task. This contains the processes where SonarCloud analyzes the code.
2. Near the end of the log, locate the URL to the results viewer and open it.

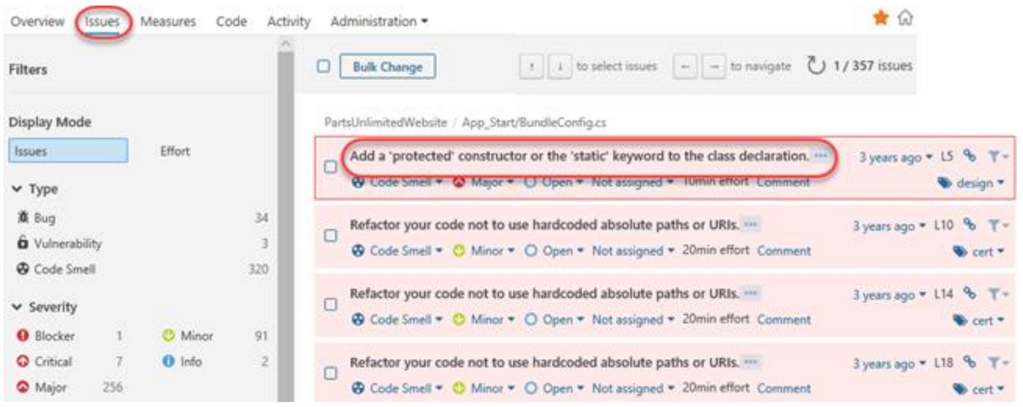


```
2018-09-05T17:50:51.2673999Z WARNING: WARN: Too many duplication references on file PartsUnlimited-aspnet45/src/PartsUnlimitedWebsi
2018-09-05T17:50:51.3947493Z WARNING: WARN: Too many duplication references on file PartsUnlimited-aspnet45/src/PartsUnlimitedWebsi
2018-09-05T17:50:51.4633714Z INFO: CPD calculation finished
2018-09-05T17:50:51.8708252Z INFO: Analysis report generated in 390ms, dir size=6 MB
2018-09-05T17:50:52.3440983Z INFO: Analysis reports compressed in 468ms, zip size=1 MB
2018-09-05T17:50:53.8844203Z INFO: Analysis report uploaded in 1554ms
2018-09-05T17:50:53.8958997Z INFO: ANALYSIS SUCCESSFUL, you can browse https://sonarcloud.io/dashboard?id=partsunlimited.johndoe
2018-09-05T17:50:53.8959382Z INFO: Note that you will be able to access the updated dashboard once the server has processed the sub
2018-09-05T17:50:53.8959767Z INFO: More about the report processing at https://sonarcloud.io/api/ce/task?id=AWWq28N_HZ_RHZXc9nPS
```

3. Open the URL in new browser tab.

Terms	Description
Bugs	An issue that represents something wrong in the code. If this has not broken yet, it will, and probably at the worst possible moment. This needs to be fixed
Vulnerabilities	A security-related issue which represents a potential backdoor for attackers
Code Smells	A maintainability-related issue in the code. Leaving it as-is means that at best maintainers will have a harder time than they should make changes to the code. At worst, they'll be so confused by the state of the code that they'll introduce additional errors as they make changes
Coverage	To determine what proportion of your project's code is actually being tested by tests such as unit tests, code coverage is used. To guard effectively against bugs, these tests should exercise or 'cover' a large proportion of your code
Duplications	The duplications decoration shows which parts of the source code are duplicated
Size	Provides the count of lines of code within the project including the number of statements, Functions, Classes, Files and Directories

4. Select the **Issues** tab. This provides a convenient way to filter and sort the results so that you can attack the section you feel needs immediate attention.



PartsUnlimitedWebsite / App_Start/BundleConfig.cs

Add a 'protected' constructor or the 'static' keyword to the class declaration. 3 years ago L5
Code Smell Major Open Not assigned 10min effort Comment design

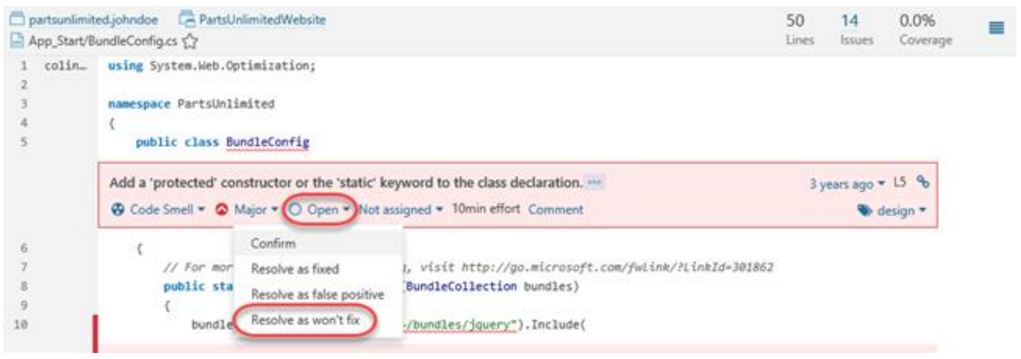
Refactor your code not to use hardcoded absolute paths or URIs. 3 years ago L10
Code Smell Minor Open Not assigned 20min effort Comment cert

Refactor your code not to use hardcoded absolute paths or URIs. 3 years ago L14
Code Smell Minor Open Not assigned 20min effort Comment cert

Refactor your code not to use hardcoded absolute paths or URIs. 3 years ago L18
Code Smell Minor Open Not assigned 20min effort Comment cert

5. The code view provides an in-depth review of each issue, along with suggestions and configuration options.

   For this issue, select **Open | Resolve as won't fix**.


partsunlimited.johndoe PartsUnlimitedWebsite
App_Start/BundleConfig.cs
50 Lines 14 Issues 0.0% Coverage

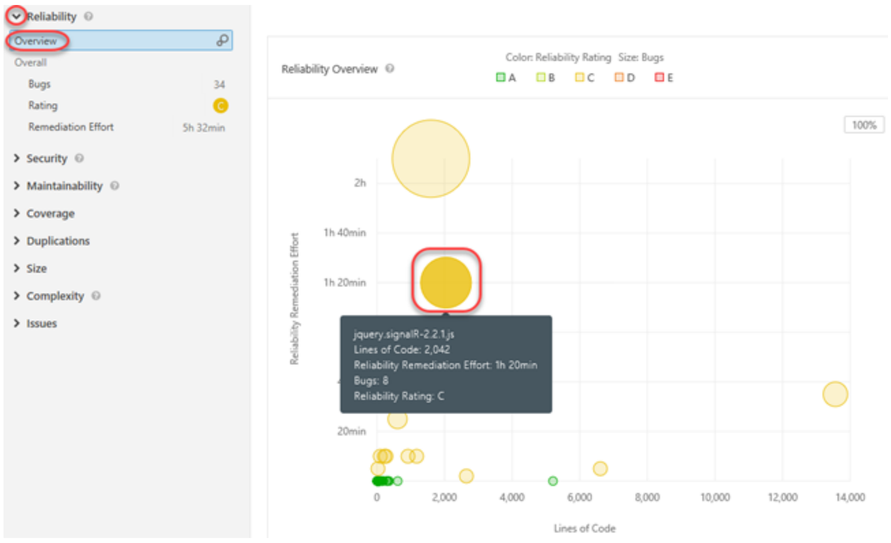
```
colin…  using System.Web.Optimization;

        namespace PartsUnlimited
        {
            public class BundleConfig
```

Add a 'protected' constructor or the 'static' keyword to the class declaration. 3 years ago L5
Code Smell Major Open Not assigned 10min effort Comment design

```
            {
                // For mor…, visit http://go.microsoft.com/fwlink/?LinkId=301862
                public sta…BundleCollection bundles)
                {
                    bundle…/bundles/jquery").Include(
```

6. Select the **Measures** tab. This provides a visualization of issues as selected by the available filters.
7. Filter down to see the **Reliability | Overview**. This enables you to hover over the various assets to see the amount of effort required to fix and/or maintain various components for reliability.



8. Select the **Code** tab and drill into the project. This provides a way to review project issues at a file level.
9. Open the file with Bugs and see the issues.
10. Expand the **Administration** option. Note that there is an incredible amount of flexibility available here for customizing your SonarCloud analysis.

Set up Pull request Validation Integration

- A SonarCloud project needs to be provided with an access token so it can add PR comments to Azure DevOps, and
- Branch Policy needs to be configured in Azure DevOps to trigger the PR build

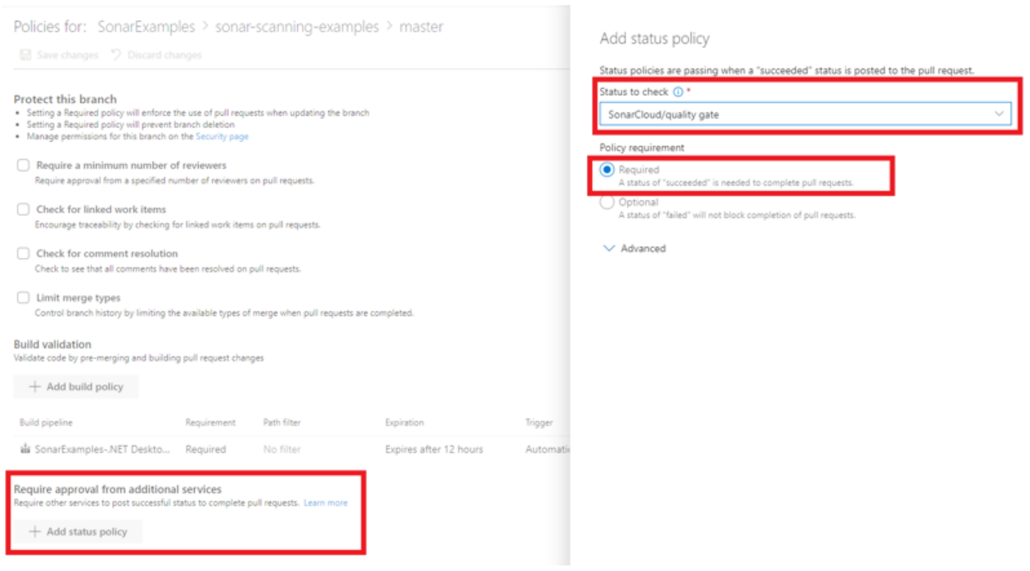
1. Configure Sonar Cloud to be able to post comments in Pull Request.
    a. **Azure DevOps Site**: Create a **Personal Access Token** in Azure DevOps with **Code (read and write)** scope.
    b. Sonar Cloud Site → Select the Project → **Administration** → General Settings → Pull Requests → Provider = **Azure DevOps Services**, Personal Access Token = <Paste from previous Step>.

Note: This is required for posting comments in Pull Request.

2. Configure the branch policy for the project in Azure DevOps
    a. Repos → Branches → . . . → Branch policies
    b. Click **Add Build Policy . . . → Display name = Sonar Cloud analysis → Save**
3. Create a Pull Request
    a. Create a New Branch
    b. Edit some code in New Branch
    c. Create a Pull Request from New Branch to Master
4. If the pull request integration is correctly configured the UI will show that.
5. Complete the Pull Request and Note the pipeline has Run and you will be able to complete the pull request only if Pipeline is completed successfully.

Part2:

6. Block pull requests if the Code Quality check failed
    a. Return to the **Master Branch Policy** page → Click **Add status policy**
    b. Select **SonarCloud/quality gate** from the **Status to check** drop-down
    c. Set the **Policy requirement** to **Required** → Click **Save**


Policies for: SonarExamples > sonar-scanning-examples > master

Save changes Discard changes

Protect this branch
- Setting a Required policy will enforce the use of pull requests when updating the branch
- Setting a Required policy will prevent branch deletion
- Manage permissions for this branch on the Security page

☐ Require a minimum number of reviewers
Require approval from a specified number of reviewers on pull requests.

☐ Check for linked work items
Encourage traceability by checking for linked work items on pull requests.

☐ Check for comment resolution
Check to see that all comments have been resolved on pull requests.

☐ Limit merge types
Control branch history by limiting the available types of merge when pull requests are completed.

Build validation
Validate code by pre-merging and building pull request changes

+ Add build policy

Build pipeline	Requirement	Path filter	Expiration	Trigger
SonarExamples-.NET Deskto...	Required	No filter	Expires after 12 hours	Automati

Require approval from additional services
Require other services to post successful status to complete pull requests. Learn more

+ Add status policy

Add status policy

Status policies are passing when a "succeeded" status is posted to the pull request.

Status to check *: SonarCloud/quality gate

Policy requirement

[✓ Required]
A status of "succeeded" is needed to complete pull requests.

Optional
A status of "failed" will not block completion of pull requests.

Advanced

7. Create a New Branch
8. Edit Program.cs and add the following inside the class.

```
public void Unused()
{
}
```

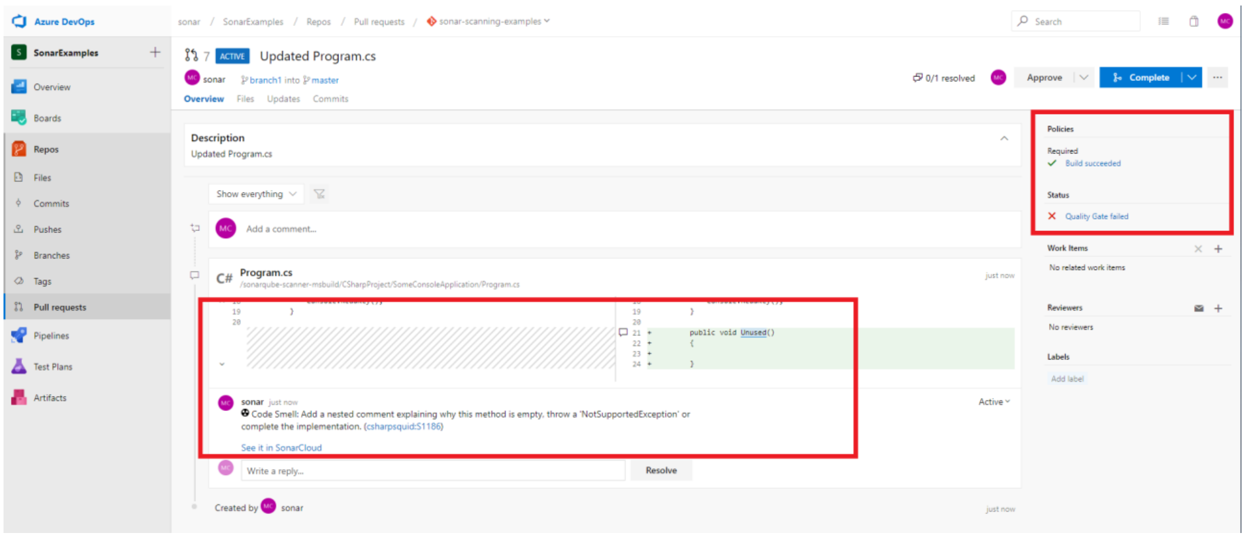
9. Save and Commit
10. Create a Pull Request and note that this time the Build Succeeds but **Gate is failed**.

Note that the only issues in code that was changed or added in the pull request are reported - pre-existing issues in **Program.cs** and other files are ignored.

11. Users will now be unable to merge the pull request until the Code Quality check is successful, either because all of the issues have been fixed or the issues have been marked as **confirmed** or **resolved** in SonarCloud.

Check the SonarCloud Quality Gate status in a Continuous Deployment scenario

12. Review the results of the Pull Request analysis



7 ACTIVE Updated Program.cs

sonar branch1 into master

0/1 resolved

Description
Updated Program.cs

Program.cs
/sonarqube-scanner-msbuild/CSharpProject/SomeConsoleApplication/Program.cs
just now

```
19          }                  19          }
20                             20
                               21 +        public void Unused()
                               22 +        {
                               23 +
                               24 +        }
```

sonar just now
Code Smell: Add a nested comment explaining why this method is empty, throw a 'NotSupportedException' or complete the implementation. (csharpsquid:S1186)
See it in SonarCloud

Created by sonar just now

Policies
Required
✓ Build succeeded
Status
✗ Quality Gate failed

Work Items
No related work items

Reviewers
No reviewers

Labels

Sonar Cloud Tasks in YAML.

```
- task: SonarCloudPrepare@1
  displayName: 'Prepare SonarCloud analysis'
  inputs:
    SonarCloud: 'SonarCloud connection 1'
    organization: '$(SonarOrganization)'
    scannerMode: 'MSBuild'
    projectKey: '$(SonarProjectKey)'
    projectName: '$(SonarProjectName)'
    projectVersion: '$(Build.BuildNumber)'
    extraProperties: |
            sonar.exclusions=**/obj/**,**/*.dll
            sonar.cs.opencover.reportsPaths=$(Build.SourcesDirectory)/**/coverage.opencover.xml
            sonar.cs.vstest.reportsPaths=$(Agent.TempDirectory)/*.trx
```

```
//.....Task for Build and Test
- script: dotnet build --configuration $(buildConfiguration)
  displayName: 'dotnet build $(buildConfiguration)'

- script: dotnet test --configuration $(BuildConfiguration) /p:CollectCoverage=true
/p:CoverletOutputFormat=opencover --logger trx'
  displayName: 'dotnet test'

- task: SonarCloudAnalyze@1
  displayName: 'Run SonarCloud code analysis'

- task: SonarCloudPublish@1
  displayName: 'Publish SonarCloud quality gate results'
```

This condition limits scans to only when the build is for a pull request to the master branch. condition: |

```
condition: |
and
(
  succeeded(),
  eq(variables['Build.Reason'], 'PullRequest'),
  eq(variables['System.PullRequest.TargetBranch'], 'master')
)
```

Note: SonarCloud extension is not available for Azure DevOps Server. You can use SonarQube which is an onpremise version for code analysis which can be integrated with both Azure DevOps services and Server.

Managing technical debt with SonarQube and Azure DevOps

https://azuredevopslabs.com/labs/vstsextend/sonarqube/

Scan open-source components using WhiteSource Bolt

Open-source software (OSS) is a type of computer software in which source code is released under a license in which the copyright holder grants users the rights to study, change, and distribute the software to anyone and for any purpose

Corporate Concerns with Open-Source Software Components:

- **Are of low quality**: This would impact maintainability, reliability, and performance of the overall solution

- **Have no active maintenance**: The code would not evolve over time or be alterable without making a copy of the source code, effectively forking away from the origin
- **Contain malicious code**: The entire system that includes and uses the code will be compromised. Potentially the entire company's IT and infrastructure is affected
- **Have security vulnerabilities**: The security of a software system is as good as its weakest part. Using source code with vulnerabilities makes the entire system susceptible to attack by hackers and misuse
- **Have** unfavorable **licensing restrictions:** The effect of a license can affect the entire solution that uses the open-source software.

Important Notes about OSSC:

- It's important to have an **inventory of the open-source components** that your project uses.
- It's important to understand which **libraries** have **vulnerability** issues, when those vulnerabilities were addressed, and what versions you can use. Based on this information, you might even choose to use a different library or write your own.
- You also need to understand what **licenses** these libraries use. Some licenses require you to make public any code that uses that library if you've made changes to the library's code. This requirement is problematic when your source code is not open source as well. You'll need to check with your own legal team to determine which licenses you can use.

WhiteSource is the leader in continuous open source **software security and compliance management**. WhiteSource integrates into your build process, irrespective of your programming languages, build tools, or development environments. It works automatically, continuously, and silently in the background, **checking the security, licensing, and quality** of your open source components against WhiteSource constantly-updated definitive database of open source repositories.

Azure DevOps integration with WhiteSource Bolt will enable you to:

1. Detect and remedy vulnerable open source components.
2. Generate comprehensive open source inventory reports per project or build.
3. Enforce open source license compliance, including dependencies' licenses.
4. Identify **outdated** open source libraries with recommendations to update.

Walkthrough

1. From Gallery install WhiteSource Bolt
2. Pipelines → **WhiteSource Bolt** tab → You see a form that asks for your email address and company name.
3. Enter Work Email = <Your Email Id>, Company Name and Country → Get Started

You will get a message: ✓ **You are using a FREE version of WhiteSource Bolt**

4. Add the following step to the end of the YAML file

```
- task: WhiteSource Bolt@20
  displayName: 'Run WhiteSource Bolt'
```

5. Save and Run the Build Pipeline

The following comprehensive reports and dashboards will be generated automatically:

- Security vulnerabilities dashboard
- Security vulnerabilities report
- Outdated libraries report
- License risks and compliance dashboard
- Inventory report

6. When the build completes, navigate to the **WhiteSource Bolt Build Report** tab.

Other Vulnerability Tools for OSS Code Scanning

Micro Focus Fortify:

- Can add build tasks to CI/CD pipeline to help identify vulnerabilities in your source code
- Provides a comprehensive set of software security analyzers
- Fortify Static Code Analyzer: Identifies root causes of software security vulnerabilities
- Fortify on Demand: Automatically submits static and dynamic scan requests.

Checkmarx:

Identifies, tracks, and fixes technical and logical security flaws:

- Best fix location
- Quick and accurate scanning
- Incremental scanning
- Seamless integration
- Code portfolio protection
- Easy to initiate Open-Source Analysis with *Checkmarx Open-Source Analysis (CxOSA)* and *Checkmarx Static Application Security Testing (CxSAST)*

Static versus Dynamic Application Security Testing.

- Static application security testing takes place while the application is not running and identifies flaws, vulnerabilities, back doors in the code while in that state. Static analysis can take place early in the dev cycle and help identify potential issues early on.

- Dynamic application security testing takes place while the code is running, it will monitor memory usage, functional behaviours, performance, responses times and other items.

Veracode:

- Integrate application security into your development tools
- Don’t stop for false alarms
- Align your application security practices with your development practices
- Don’t just find vulnerabilities, fix them
- Onboarding process allows for scanning on day one

Secure DevOps Kit for Azure (AzSK)

https://github.com/azsk/DevOpsKit-docs

AzSK is a collection of scripts, tools, extensions, automations

- Helps secure the subscription
- Enables secure development
- Integrates security into CI/CD
- Provides continuous assurance
- Assists with alerts & monitoring
- Governing cloud risks

Azure Governance

Azure Policy

- Is a service in Azure that you use to create, assign, and manage policies
- Provides enforcement by using policies and initiatives
- Runs evaluations on your resources and scans for those not compliant
- Comes with several built-in policy and initiative definitions

An example of an Azure policy that you can integrate with your DevOps pipeline is the Check Gate task

Azure Security Center

- Provide security recommendations
- Monitor security settings
- Monitor all your services
- Use Azure Machine Learning
- Analyze and identify potential attacks
- Supports Windows and Linux operating systems

Scenario 1:

Many organizations learn how to respond to security incidents only after suffering an attack. To reduce costs and damage, it's important to have an incident response plan in place before an attack occurs. You can use Azure Security Center in different stages of an incident response.

You can use Security Center during the detect, assess, and diagnose stages.

- **Detect:** Review the first indication of an event investigation. Example: Use the Security Center dashboard to review the initial verification that a high-priority security alert was raised.
- **Assess:** Perform the initial assessment to obtain more information about the suspicious activity. Example: Obtain more information about the security alert.
- **Diagnose:** Conduct a technical investigation and identify containment, mitigation, and workaround strategies. Example: Follow the remediation steps described by Security Center in that particular security alert.

Scenario 2:

You can reduce the chances of a significant security event by configuring a security policy, and then implementing the recommendations provided by Azure Security Center.

- A *security policy* defines the set of controls that are recommended for resources within that specified subscription or resource group. In Security Center, you define policies according to your company's security requirements.
- Security Center analyzes the security state of your Azure resources. When Security Center identifies potential security vulnerabilities, it creates recommendations based on the controls set in the security policy. The recommendations guide you through the process of configuring the needed security controls

Resource Locks

Prevent accidental deletion or modification of your Azure resources.

Locks help you prevent accidental deletion or modification of your Azure resources. You can manage these locks from within the Azure portal. To view, add, or delete locks, go to the SETTINGS section of any resource's settings blade. You may need to lock a subscription, resource group, or resource to prevent other users in your organization from accidentally deleting or modifying critical resources. You can set the lock level to CanNotDelete or ReadOnly.

Azure Blueprints:

Allows for the definition of a repeatable set of Azure resources that implement and adhere to an organization's standards, patterns, and requirements

Azure Advanced Thread Protection (ATP)

- **Azure ATP portal.** Azure ATP has its own portal, through which you can monitor and respond to suspicious activity. The portal allows you to create your Azure ATP instance, and view the data received from Azure ATP sensors.

You can also use the portal to monitor, manage, and investigate threats in your network environment. You sign in to the Azure ATP portal at https://portal.atp.azure.com. However, note that you must sign in with a user account that is assigned to an Azure AD security group that has access to the Azure ATP portal.

- **Azure ATP sensor**. Azure ATP sensors are installed directly on your domain controllers. The sensor monitors domain controller traffic without requiring a dedicated server or configured port mirroring.
- **Azure ATP cloud service.** Azure ATP cloud service runs on the Azure infrastructure, and is currently deployed in the United States, Europe, and Asia. The cloud service is connected to Microsoft Intelligent Security Graph