



Azure DevOps

Tooling and Code Analysis

.NET CORE

*DevOps technologies,
combined with people and
processes, enable teams
to implement CI/CD and
continually provide value
to customers.*

[HTTPS://AZURE.MICROSOFT.COM/EN-US/SOLUTIONS/DEVOPS/](https://azure.microsoft.com/en-us/solutions/devops/)

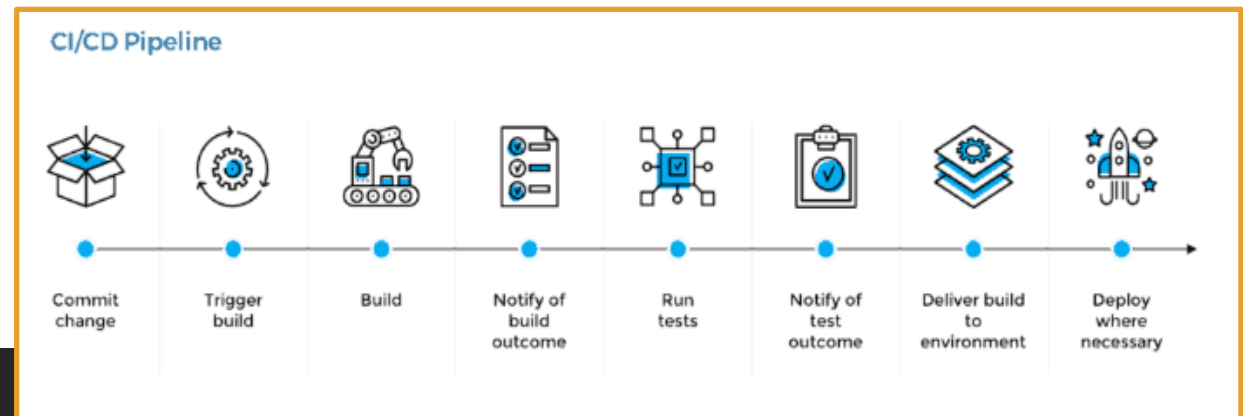
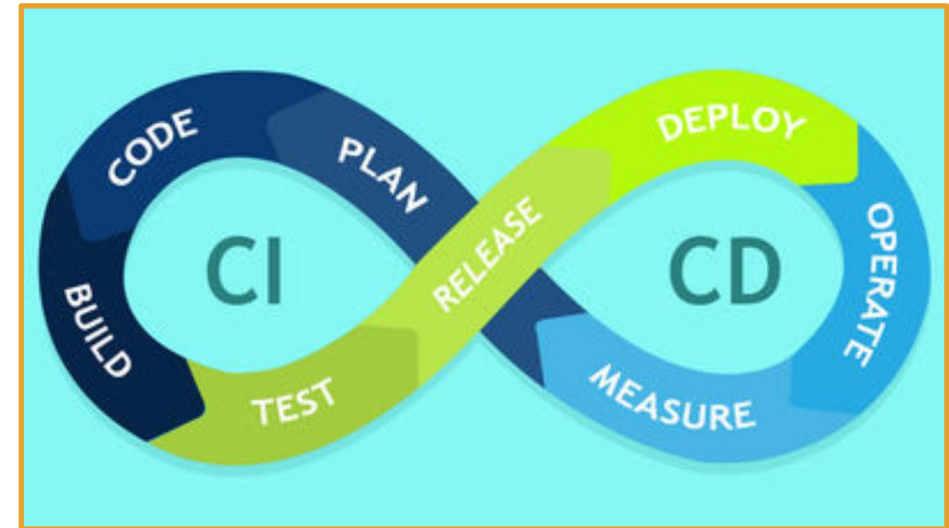
CI/CD and Continuous Testing (CT)

<https://docs.microsoft.com/en-us/azure/devops/pipelines/overview?view=azure-devops-2019>

Continuous Integration (CI) is the practice of automating the merging and testing of code. Implementing **CI** helps catch bugs early, which makes them less expensive to fix. Automated tests execute as part of the CI process.

Continuous Delivery (CD) is a process by which code is built, tested, and deployed to one or more test and production environments to help improve product quality.

Continuous Testing (CT) is the use of automated build-deploy-test workflows that test your changes continuously in a fast, scalable manner.



Azure DevOps - Introduction

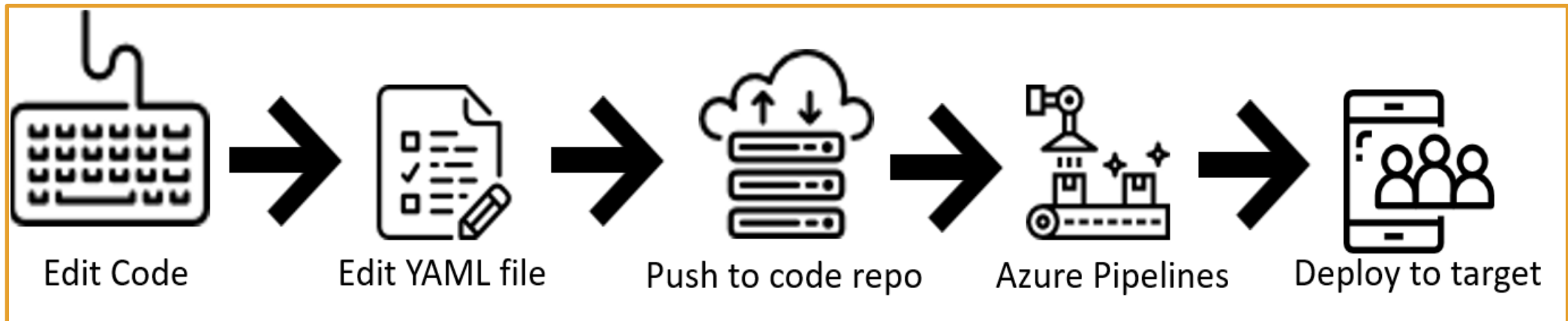
<https://docs.microsoft.com/en-us/azure/devops/pipelines/ecosystems/dotnet-core?view=azure-devops>

<https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/pipelines-get-started?view=azure-devops>

<https://docs.microsoft.com/en-us/azure/devops/pipelines/?view=azure-devops>

Azure Pipelines is a cloud service that you can use to automatically build and test your code and make it available to other users. **Azure Pipelines** works with many language or project types.

Azure Pipelines combines *Continuous Integration (CI)* and *Continuous Delivery (CD)* to constantly test and build your code to be shipped to any target.

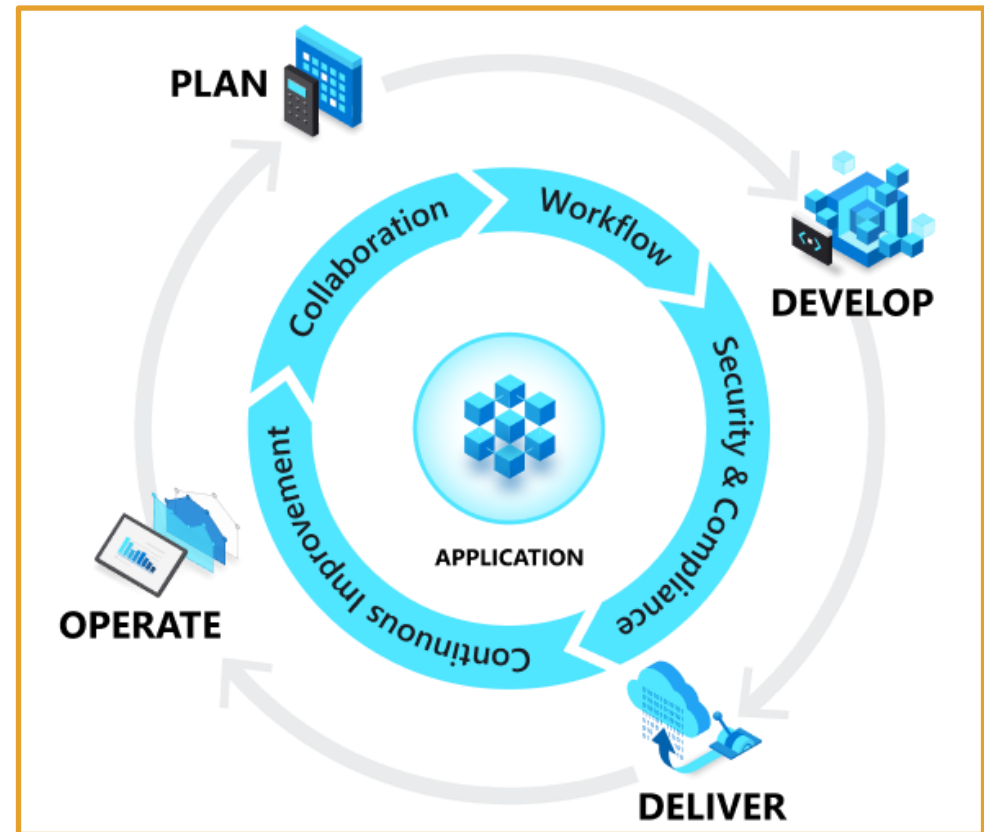


Build Definition

<https://docs.microsoft.com/en-us/aspnet/web-forms/overview/deployment/configuring-team-foundation-server-for-web-deployment/creating-a-build-definition-that-supports-deployment#task-overview>

A **build definition** is the mechanism that controls how and when builds occur. **Azure DevOps** uses a **.yaml** file to define a build. Each build definition specifies:

- The things you want to build.
- The criteria that determine when a build should take place
- The location to which the Build should send build outputs.
- The amount of time that each build should be retained.
- Various other parameters of the build process.

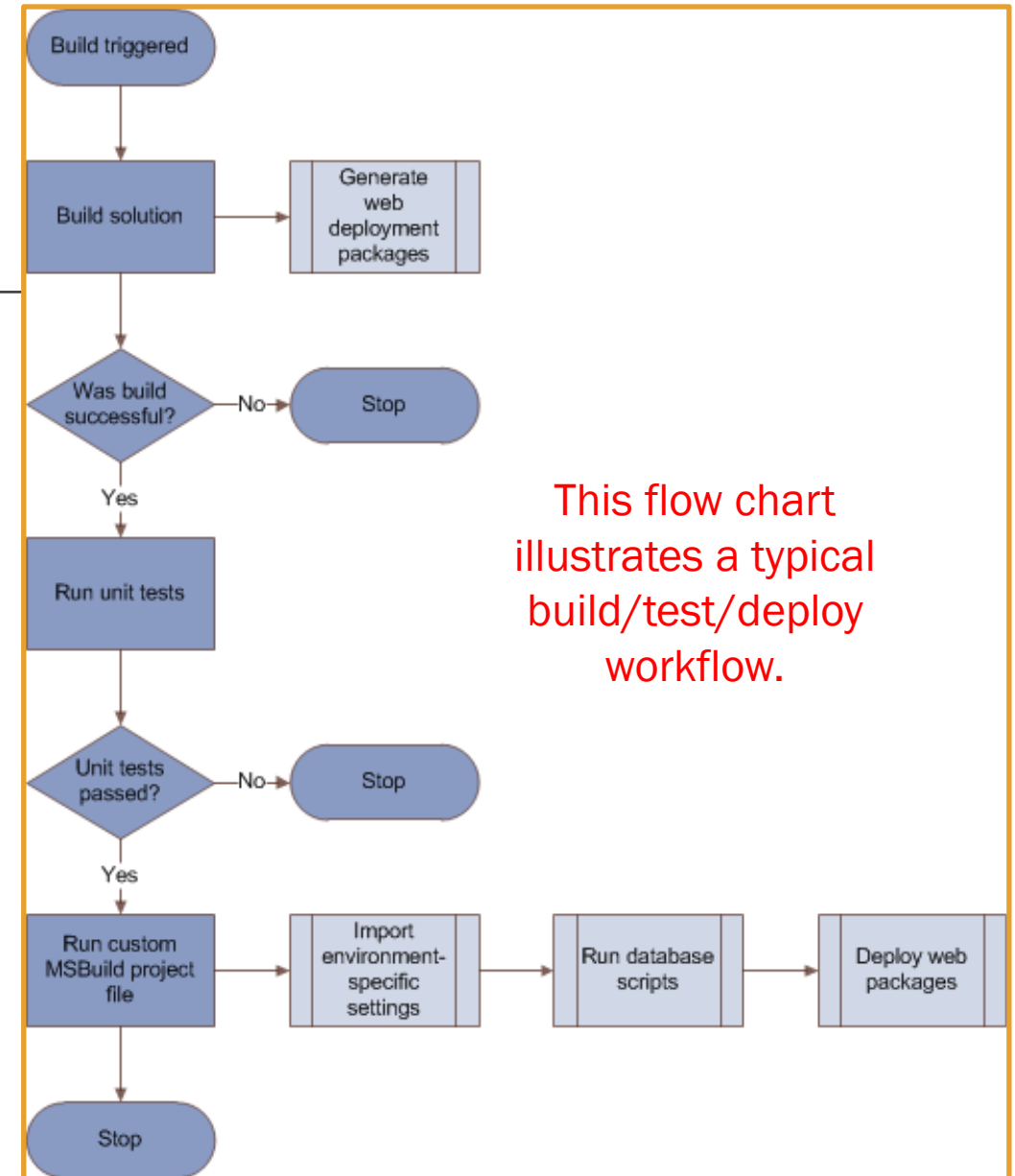
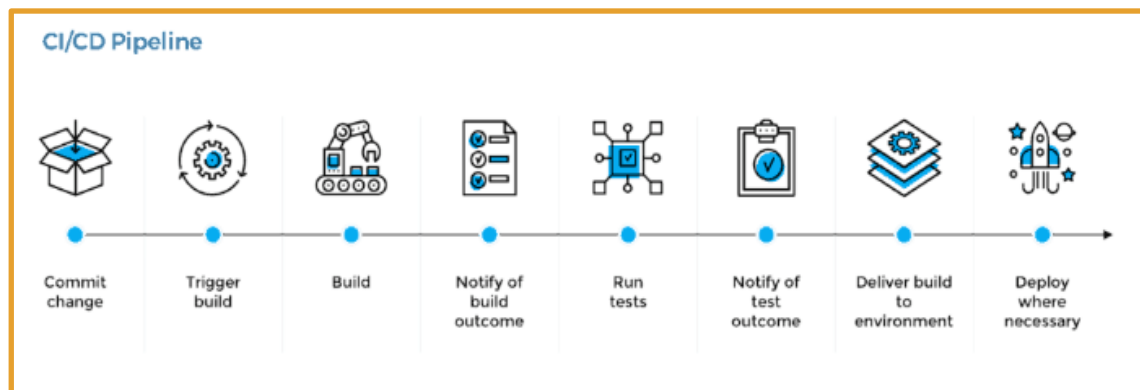


Release Pipeline

<https://docs.microsoft.com/en-us/azure/devops/pipelines/release/?view=azure-devops>

Release pipelines in **Azure Pipelines** help your team implement CI/CD and deliver software to your clients faster and with lower risk.

You can fully automate the testing, delivery, and analysis of your software all the way to production or set up semi-automated processes with required approvals and on-demand deployments.



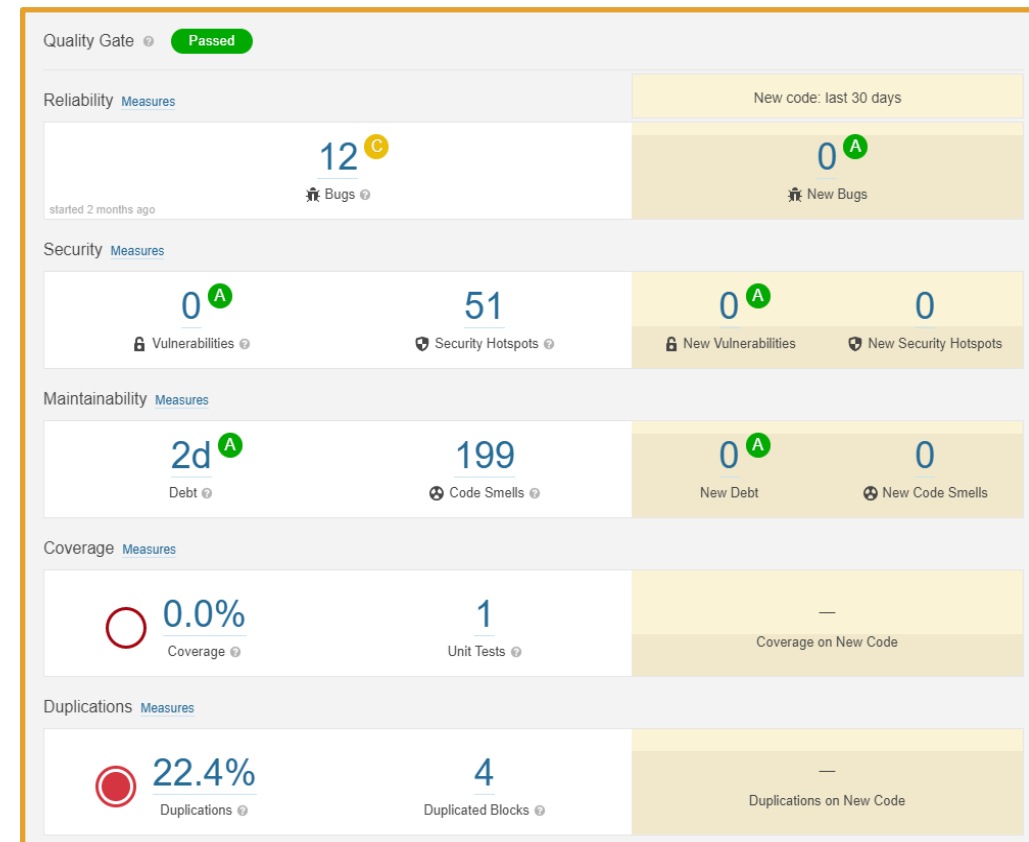
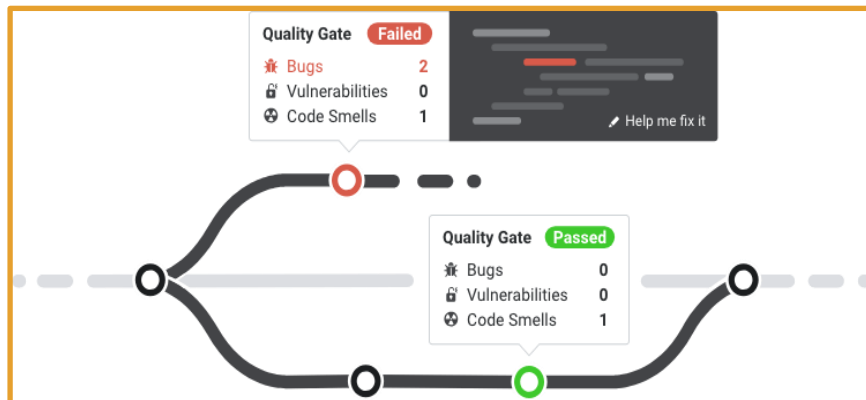
What is Static Code Analysis?

https://en.wikipedia.org/wiki/Static_program_analysis

Static code analysis is the analysis of computer software performed without executing the program. **Static code analysis** is usually performed on the source code.

The term is usually applied to the analysis performed by an automated tool. SonarCloud and SonarQube are popular **Static Code Analysis** tools.

Human analysis is called **Code Review**.



What is a Coverage Review?

<https://sonarcloud.io/documentation/user-guide/metric-definitions/>

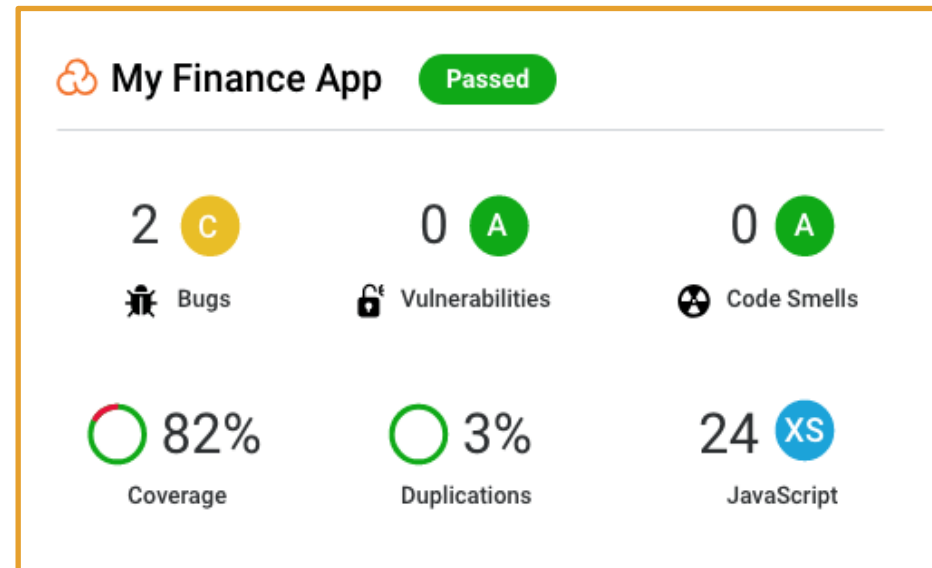
<https://sonarcloud.io/documentation/user-guide/concepts/>

How much of the source code has been covered by the unit tests?

Code Coverage is determined by evaluating what percentage of the total lines of code are covered by unit testing. It is a mix of Line coverage and Condition coverage.

$$\text{Coverage} = (CT + CF + LC) / (2*B + EL)$$

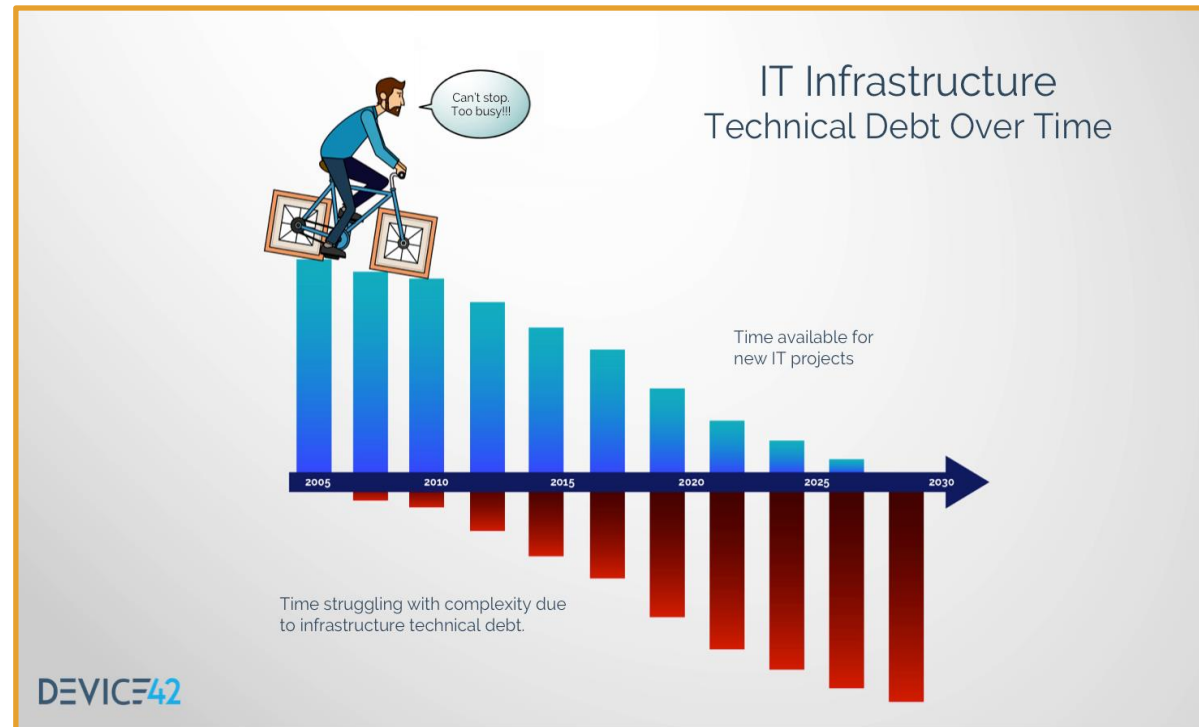
- CT = conditions that have been evaluated to 'true' at least once
- CF = conditions that have been evaluated to 'false' at least once
- LC = covered lines = lines_to_cover - uncovered_lines
- B = total number of conditions
- EL = total number of executable lines (lines_to_cover)



Technical Debt

<https://sonarcloud.io/documentation/user-guide/concepts/>

Technical Debt is the estimated time required to fix all Maintainability Issues/code smells.

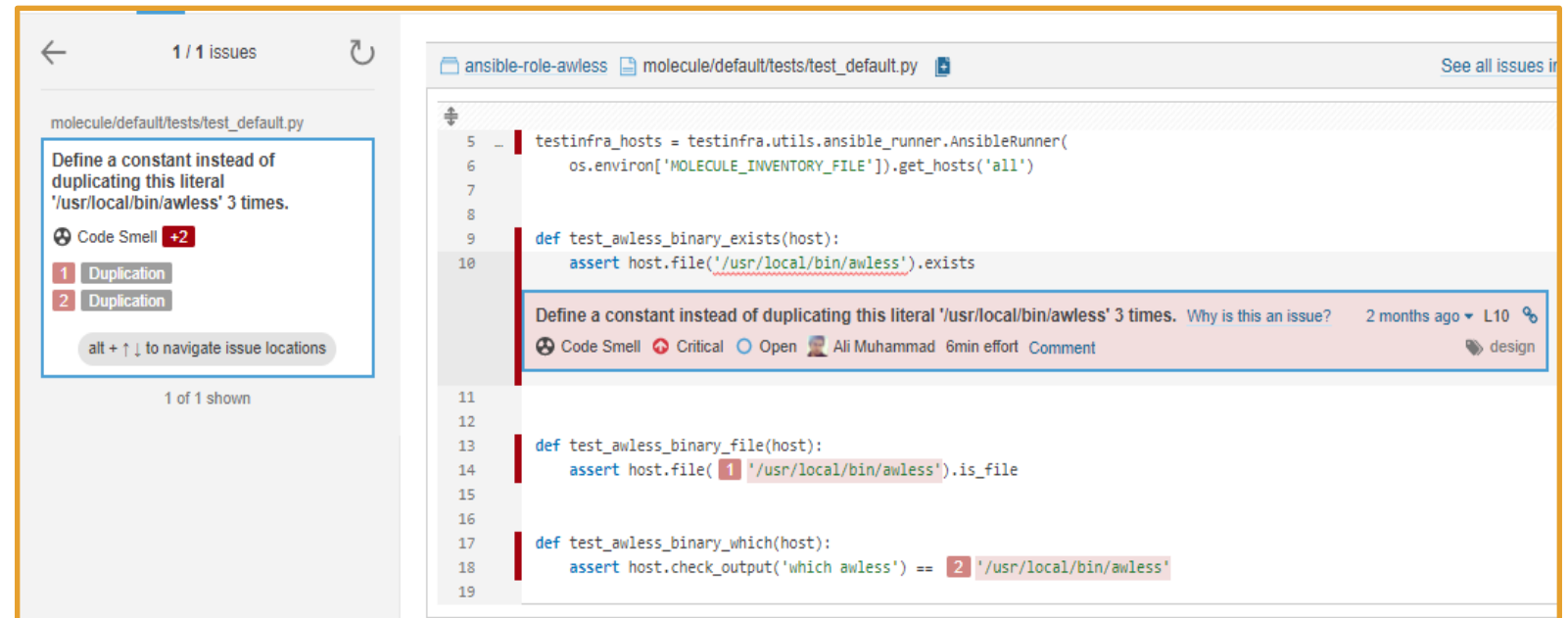


What is a Code Smell?

<https://sonarcloud.io/documentation/user-guide/concepts/>
https://sonarcloud.io/project/issues?id=ansible-role-awless&open=AXEseUF1IRsecPgXK050&resolved=false&types=CODE_SMELL

A **Code Smell** is any characteristic in the source code of a program that possibly indicates a deeper problem. Determining what is and is not a **Code Smell** is subjective, and varies by language, developer, and development methodology.

A **Code Smell** is an issue with long-term maintainability in the code. Leaving it as-is means that it will be more difficult for maintainers to make changes to the code. They'll risk introducing new errors as they make changes.



The screenshot displays the SonarCloud interface for a project named 'ansible-role-awless'. On the left, a sidebar shows '1 / 1 issues' and a list of two 'Duplication' issues. The main panel shows the source code of 'molecule/default/tests/test_default.py'. A red box highlights a 'Code Smell' issue with the message: 'Define a constant instead of duplicating this literal \'/usr/local/bin/awless\' 3 times.' The issue is marked as 'Critical' and 'Open'. The code snippet shows three functions: 'testinfra_hosts', 'test_awless_binary_exists', and 'test_awless_binary_file'. The 'test_awless_binary_file' function contains a line with a duplicated literal: 'assert host.file(1 \'/usr/local/bin/awless\').is_file'. The 'test_awless_binary_which' function contains a line with a duplicated literal: 'assert host.check_output('which awless') == 2 \'/usr/local/bin/awless\'

Duplication

https://sonarcloud.io/component_measures?id=microsoft_vscode-python&metric=Duplications

Duplication in code analysis indicates lines of code that are identical and could theoretically be separated into a method to be called or resolved using SOLID or DRY principles.

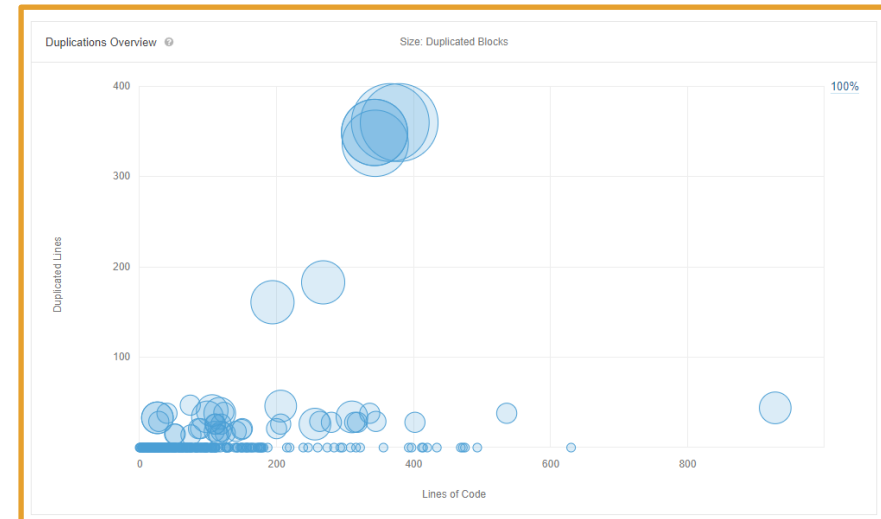
Duplications OVERVIEW	
Overview	
On new code	
Density	0.7%
Duplicated Lines	85
Duplicated Blocks	8
Overall	
Density	2.3%
Duplicated Lines	3,178
Duplicated Blocks	81
Duplicated Files	46

Inheritance Principle: Example

Duplicated Code

```
public class BusinessCustomer
{
    public int CustomerId { get; set; }
    public string CompanyName { get; set; }
    public string Address { get; set; }
    public string PostalCode { get; set; }
    public string Country { get; set; }
    public string Telephone { get; set; }
    public List<Order> Orders { get; set; }
    public DateTime Created { get; set; }
    public DateTime Modified { get; set; }
}

public class PrivateCustomer
{
    public int CustomerId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Title { get; set; }
    public string Address { get; set; }
    public string PostalCode { get; set; }
    public string Country { get; set; }
    public string Telephone { get; set; }
    public List<Order> Orders { get; set; }
    public DateTime Created { get; set; }
    public DateTime Modified { get; set; }
}
```



Quality Gate

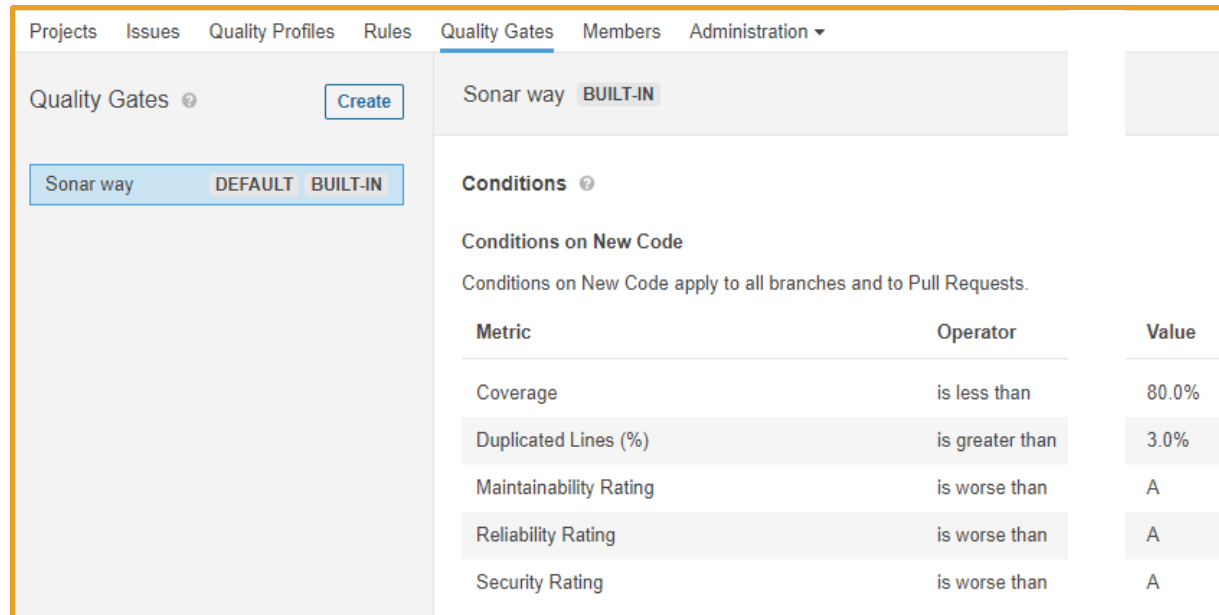
<https://sonarcloud.io/documentation/user-guide/quality-gates/>

A **Quality Gate** is the best way to [Fix the Water Leak](#) and enforce policies ensuring high quality code in your organization.

You can define as many quality gates as you wish. SonarCloud, by default, provides a built-in **Quality Gate** that is recommended for most projects. You can receive a notification when the **Quality Gate** fails.

To create a **Quality Gate**, define a set of Boolean conditions based on measure thresholds. Projects are then measured against them. For example:

- No new blocker issues
- Code coverage on new code greater than 80%



The screenshot displays the SonarCloud interface for configuring a Quality Gate. The top navigation bar includes links for Projects, Issues, Quality Profiles, Rules, Quality Gates (selected), Members, and Administration. The main content area is titled 'Quality Gates' and features a 'Create' button. Below this, there are tabs for 'Sonar way', 'DEFAULT', and 'BUILT-IN'. The 'Sonar way' tab is active, showing a table of conditions for new code.

Metric	Operator	Value
Coverage	is less than	80.0%
Duplicated Lines (%)	is greater than	3.0%
Maintainability Rating	is worse than	A
Reliability Rating	is worse than	A
Security Rating	is worse than	A

Monitoring Security and Vulnerability

<https://sonarcloud.io/documentation/user-guide/concepts/>

<https://sonarcloud.io/documentation/user-guide/metric-definitions/#security>

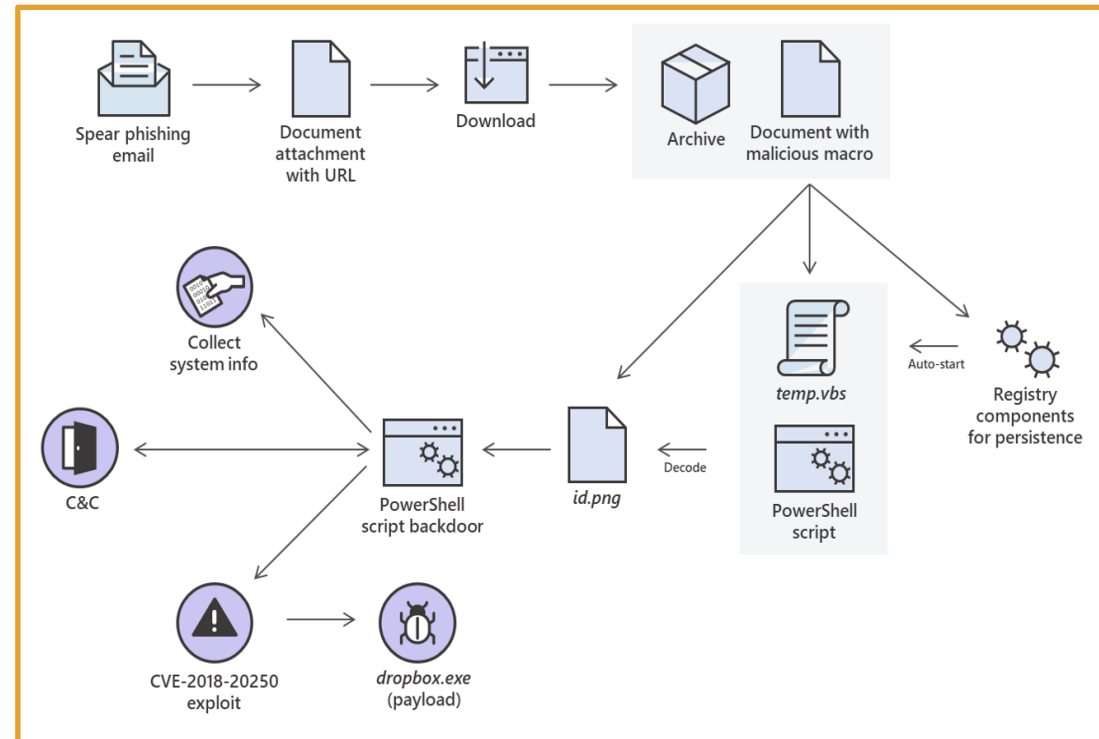
Security-related issues represent a place in your code that attackers could exploit.

Security hotspots are areas of the code that may cause security issues and therefore need to be reviewed.

The SonarCloud Quality Model has three different types of rules: Reliability (bug), Vulnerability (security), and Maintainability (code smell).

One of these rules will usually find and flag anything suspicious.

Then a human security auditor can manually review the report, delete the false positives, and send the appropriate issues for remediation.



End of Presentation

Going over Visual Studio stuff

DOES *dotnet build* in CLI - SHOWS RESTAURANT REVIEWS cli SCREEN TO SHOW HOW IT GETS COMPILED.

Code gets compiled in an order that makes sure each part has its' dependencies compiled before it gets compiled.

Running debug mode may take longer because there are things behind the scenes

Running in release mode

Looking at restaurantreviews.csproj file. Shows the ASP.NET uses the Microsoft.NET.SDK and what initial assemblies are included.

Framework – what is a framework and what does it mean to target a specific one. Shows how the dependencies are in the .csproj file. You could just type it in and that is the same as using NUGET package installing.

Uses ***dotnet clean*** to clean up the files in ***bin\debug\netcoreapp3.1***

Shows ***dotnet publish*** this will run a build first, compile the code and put the .dll in a default public folder inside the bin folder. The .dll contains everything to run the project.

(usually ***dotnet publish*** won't compile an entire solution. You have to specify the project in the solution to publish.

(user secrets are only for development so if your connections string is in there, it doesn't get sent with the publish .dll)

Talks about servers and Kestrel which fills the gap between our code the the internet.

There is IIS (Internet Information Services) that is much more elaborate that windows uses. IIS is deactivated by default in windows. You can set it up correctly, but Nick will show it.

Go to Windows Features and switch on IIS. (Look at [.NET Core app to IIS](#)) . GO through the steps on the Docs page.

Nicks publishing with IIS in VS. There is “publish” in VS but the meaning is extended.. More capable.

Ways to publish(after setting up IIS). Right click on startup project. .. Go under build-> publish.

There are a few options. Choose deploy to IIS. WebDeploy(means just copy the folder from A to B) He writes for server == localhost and gives a site name. He needed to launch VS in Admin mode.

SiteName Default WebSite

(in the publish → settings screen) Configuration → Release

Target Framework → netcoreapp3.1

Deployment Mode → Framework Dependent

Target Runtime → win-x86

File publish options → YES remove add files

..Some other things.....

Gets confirmation screen

Then goes to the default localhost port(localhost:80) to see his app. Then the regular localhost/restaurants to see his site.

The DB doesn't work BC the connection string wasn't sent to publish...

The error screen was very short because in startup.cs there is a different view for if you are in dev mode or production mode the error screen is different.

You can put the connection in the environment variables on the server site. And be able to get the DB working. Type it in the correct format.

This demo is not how the batch will do it. You CANNOT use IIS on Mac.

The Easy VS way to publish

1. Open your mvc app of choice.
2. R click on starter proj and choose publish
3. Azure App Service → create new → create profile
4. (my VS is freezing....)
5. In App Service Screen give a name of your choice.
6. Fill in the blanks and hit enter
7. Azure creates the app service plan and app service.
8. When it's finished the smaller window will disappear.
9. nick had a problem with publishing . You probably have to click publish to publish?

Add *PublishProfiles/* to the .gitignore

Put connection string on Azure

Go to portal.azure.com and log in.

Go to app services, or all resources to find your app service named after your site.

Click on Configuration button

Look at connection string section

Click New Connection String

Give it the same name “ConnectionString” from user secrets.

Value == the connection string w/o quotes

Type == SQLAzure.. You CAN use SQL Server but Nick doesn’t know the diff between them.

Enter and Save at top of Application Setting Panel.

NOW you still can’t access the DB bc the app service and SQL DB can’t talk.

Solve this problem with going to the SQLServer resource.

Go to DB server(nicksdb) and click on “set server firewall”. Click Firewall Settings. click YES to allow “Azure services and resources to access this server”. Click SAVE.. and SAVE again

ALSO.. In azure you can see the logs of your application.

Go to the named site on the left side panel. Go to app service logs. And switch on “application logging (filesystem). Also log Information to see it all. Click save

Click on log stream and wit.... It'll show that it is now connected..

Go back to the page and cause an error. ..

He's going to delete the connection string save. Let the app restart and go back to log stream .. Reconnect.. Refresh the same page(still not getting logs).. Goes to restaurants to see that there was an error(still nothing)

Now he's using the Azure CLI to deploy to Azure App service. <https://docs.microsoft.com/en-us/azure/app-service/app-service-web-tutorial-dotnetcore-sqldb>

<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest>

We install Azure CLI to our machines.(this is for the VC Code people)



App Service

Create new



revature.com

mark.moore@revature.com

Name

mark-mvc-test

Subscription

Azure subscription 1

Resource group

.NET_MarkMoore (East US)

[New...](#)

Hosting Plan

MvcProjectStarter20200401162531Plan (South Central L

[New...](#)

Application Insights 

None

Explore additional Azure services



[Create a storage account](#)



[Create a SQL Database](#)

Clicking the Create button will create the following Azure resources

App Service - mark-mvc-test

[Export...](#)

Create

Cancel

Mark and Angie step by step deployment

- 1- upgrade or create an Azure Subscription account by putting in your phone number in.
 - 2- Give the Subscription a name and choose the No Tech Support plan. Click upgrade. (wait a few minutes.)
 - 3- Go back home click more services.
 - 4- Click Resource Groups. Click Add
 - - -choose your Subscription, resource group name, appropriate Region
 - - - click Review + Create, Click Create (and wait)
 - 4- Then find App service plan under more services. Click Add
 - 5- Choose Subscription, Resource Group, Give it a name, select Windows. Select appropriate region
- Change size of pricing tier, Click Dev/Test. F1 level Click Apply
- Click review and create. Click create.

The above is a one-time thing,

Now go home and click App Service. Click Add.

Select correct Subscription, resource group

Give the website a name

Select Code.,

.NET Core 3.1(LTS)

Windows

Appropriate Region.

Verify the pricing tier.

Review and create. Create

M & A YAML

You may need to give the class privileges in SonarCloud.
