1. Variables in Build Pipeline



The most common use of variables is to define a value that can be used into various stages of pipeline. All variables are stored as string and are immutable. In this YAML pipeline, we can set the variables in different scope as defined above in RED

* **Root level variables ->** This is global level variables that can be used by all stages and can be overwrite by subsequent stages variables.
* **Stage level variables ->** Stage level variables can overwrite the variables at root level.
* **Job level variables ->** Variables at job level can overwrite variables at stage and root level.

1. Production Deployment Strategy

* **Blue-Green Deployment ->** In blue-green deployments, there are two identical production environments: blue and green.
  + 1. Blue is the current version with live traffic, usually blue environment refers to the

existing production deployment before the new changes.

* + 1. The Green environment is the idle one with the latest changes and is ready to be switched on. Green refer to the updated application with the new changes applied.

To release a new version, code is deployed to the environment with no traffic where final tests are performed. Once IT is confident the application is ready, all traffic is routed to the green environment. Green is now live, and the actual release executed.

But if something goes wrong, IT can quickly reroute the traffic back to the blue version.

Pros of Blue-Green Software Deployment Strategy

* Easy to implement
* Straightforward rollback
* Rapid release

Cons of Blue-Green Software Deployment Strategy

* Less Realtime testing
* Hard to track experimental behaviour of software
* **Canary deployment:** Canary deployment is a strategy that incrementally deploys the new version of an application (e.g.: 5%, 25%, 80%, 100%) to a subset of users. Unlike blue-green deployments, traffic is moved to the updated code gradually. It can start at 1%, then move to 10%, 25%, and so on, until it reaches 100%.

This strategy helps you to reduce the risk of introducing a new feature in production.

1. DevOps Pillars or Processes

* Continuous Development
* Contiguous Integration
* Continuous Deployment
* Continuous Testing
* Continuous Monitoring

1. DevOps Tools for Continuous Monitoring

Continuous Monitoring is an automated process that leverages specialized software tools to empower DevOps teams with enhanced visibility of application performance, security threats, and compliance concerns across the entire DevOps pipeline.

Continuous monitoring capabilities may be further divided into three types of continuous monitoring:

**Infrastructure Monitoring** - Tools and processes for monitoring the data centres, networks, hardware, and software needed to deliver products and services.

**Application Monitoring** - Tools and processes for monitoring the health and performance of released applications in a production environment.

**Network Monitoring** - Tools and processes for monitoring network activity and components, such as servers, firewalls, routers, and switches, as well as detecting security threats across the network.

1. DevOps Tools for Continuous Monitoring

* Akamai mPulse
* AppDynamics
* ChaosSearch
* Dynatrace
* Splunk
* Sumo Logic
* Sysdig Monitor
* BMC Helix Operations Management

1. Security in Continuous Integration:

When we talk about the security in continuous integration -> It start with the development stage



* **Unite** test has been written and tested for each Module created in the project. So that we ensure 100% **code coverage** (Code Coverage means -> the code that written are executed during application, no code available that are useless and not being executed)
* We also cover the **Code Matrix** -> Means what is the complexity for the code and what is maintainability of the code)
* Static code analysis tools -> scan the code and based on the scanning it can generate the reports of various issues, recommendation to improve the code.

Above security measures can be done into both (Integrated development and continuous Integration)

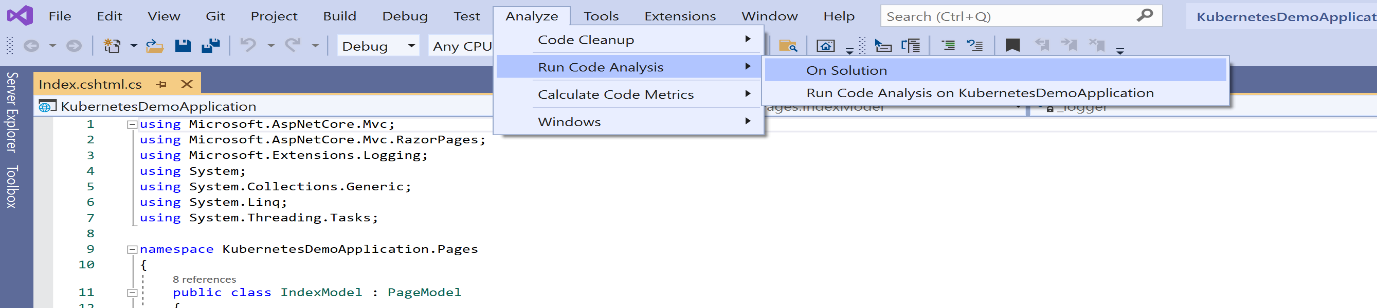
* Automation functional Testing (using Selenium we can do functional testing)
* We can also do the load testing -> very important to simulate the load testing like happen in Production environment)

**Static Code Analysis ->** We can use the various static code analysis tool via Visual Studio **->**

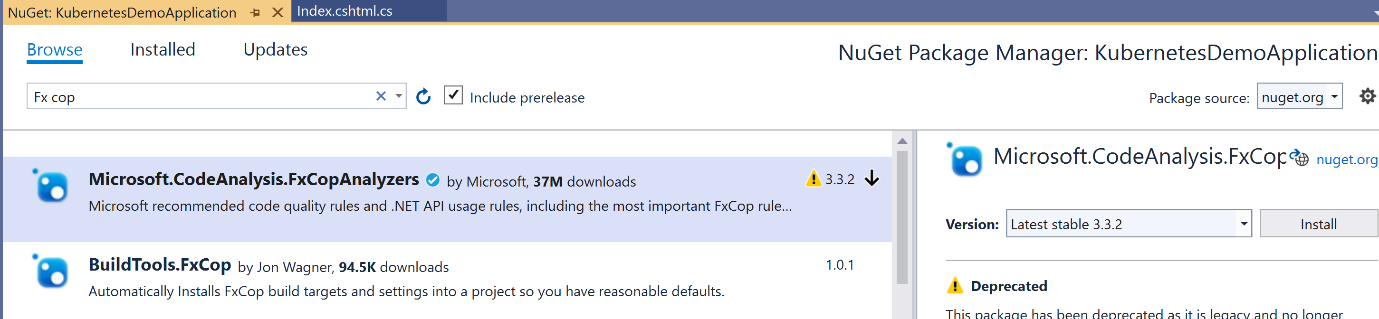
* this tool goes and look at the code, and based on the certain set of rules -> it can generates issues, defect or recommendation how to improve the code
* These rules are best on the best practice in the industry.
* Tools helps to build more secure code and high performance in nature.

We can run the code analysis by below ways.

**Step 1:** Already exist in Visual Studio

****

**Step 1:** Go to Nuget Package Manager -> Install FX Cop



**Note:** When we push the code into Azure Repo -> And created a Build Pipeline -> and Run the build pipeline -> It added one new file into Repo (azure-pipeline.yml)

So we need to Pull first then can do Push when going to push the changes into Repo

**White Source Bolt ->**

During the build phase -> developers might be using open source packages, but even if these open source components are free, it does not mean that they don't have any sort of security vulnerabilities because you are not responsible for those open source packages.

So in the continuous integration lifecycle -> We integrate some tools that detecting or scanning for such vulnerabilities.

**WhiteSource Bolt** is used for detecting any vulnerabilities in terms of licensing, in terms of security for all of your open source components.

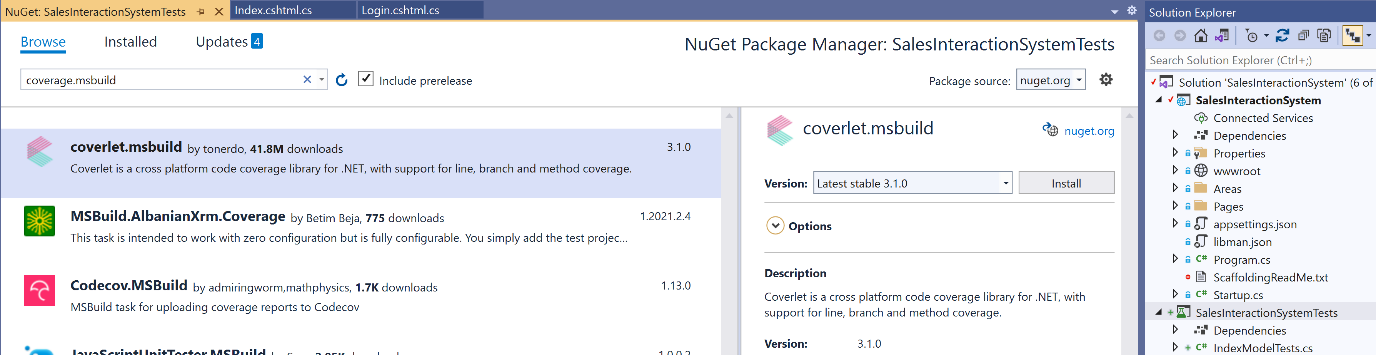
Go to DevOps Organisation Setting -> Go to Extension -> browse the marketplace -> search WhiteSource Bolt -> Install it

For Self-Hosted Machine -> Install the WhiteSource Bolt into machine

Add the task into YAML file -> Once Run -> Go to newly created Build -> go to WhiteSource Bolt Build Report -> See all the errors, defects and recommendations

**Code Coverage ->** So let's say that there is pieces of code in your application that are never run at all. Then the point comes whether these application codes is actually required because it is just an access code or excess baggage for your application if it's not being used.

* So, the best way to do code coverage is to ensure that the right unit test cases in place.
* On top of it we can add the coverlet.msbuild (Code Coverage tool) into your test project. So having this package in place will actually help us to get the code coverage results.



- task: DotNetCoreCLI@2

displayName: 'dotnet test'

inputs:

command: 'test'

projects: '\*\*/\*Test.csproj'

arguments: '/p:CollectCoverage=true /p:CoverletOutputFormat=cobertura /p:CoverletOutput=./MyCoverage/'

publishTestResults: true

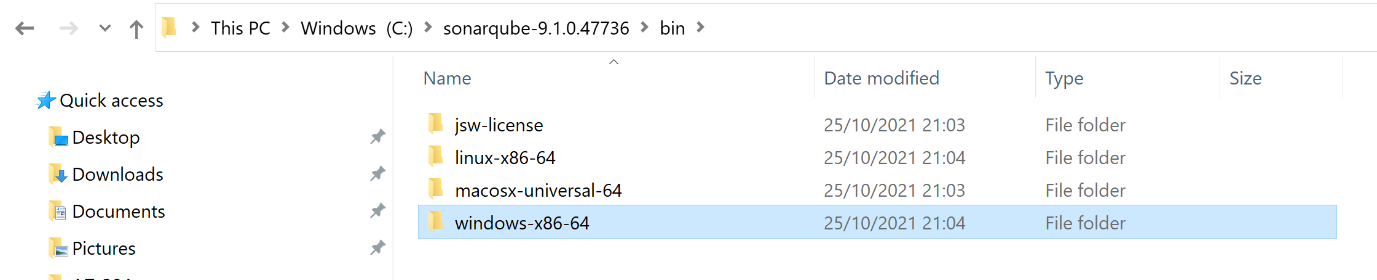
**SonarQube ->**

We can download the SonarQube on our VM and install it and then can use it in our CI/CD. In the cloud environment, we don’t need to install anything, we can use the SonarCloud

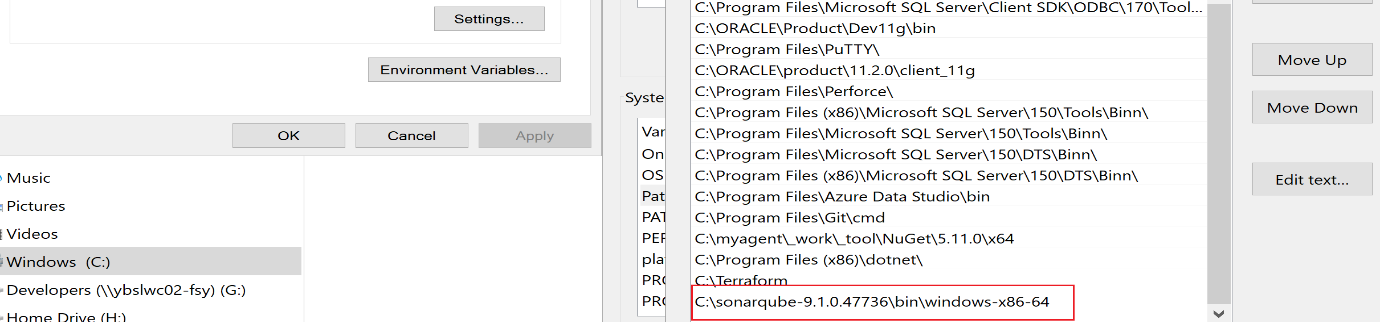
* Static code analysis for 15 languages -> Java, JavaScript, C#, TypeScript, Kotlin, Ruby, Go, Scala, Flex, Python, PHP, HTML, CSS, XML and VB.NET
* Detect Bugs & Vulnerabilities
* Review Security Hotspots
* Track Code Smells & fix your Technical Debt
* Code Quality Metrics & History
* CI/CD integration
* Extensible, with 50+ community plugins

Configure SonarQube on Premise VM - [How to Install SonarQube on Windows 10 - YouTube](https://www.youtube.com/watch?v=846xAH40oUM)

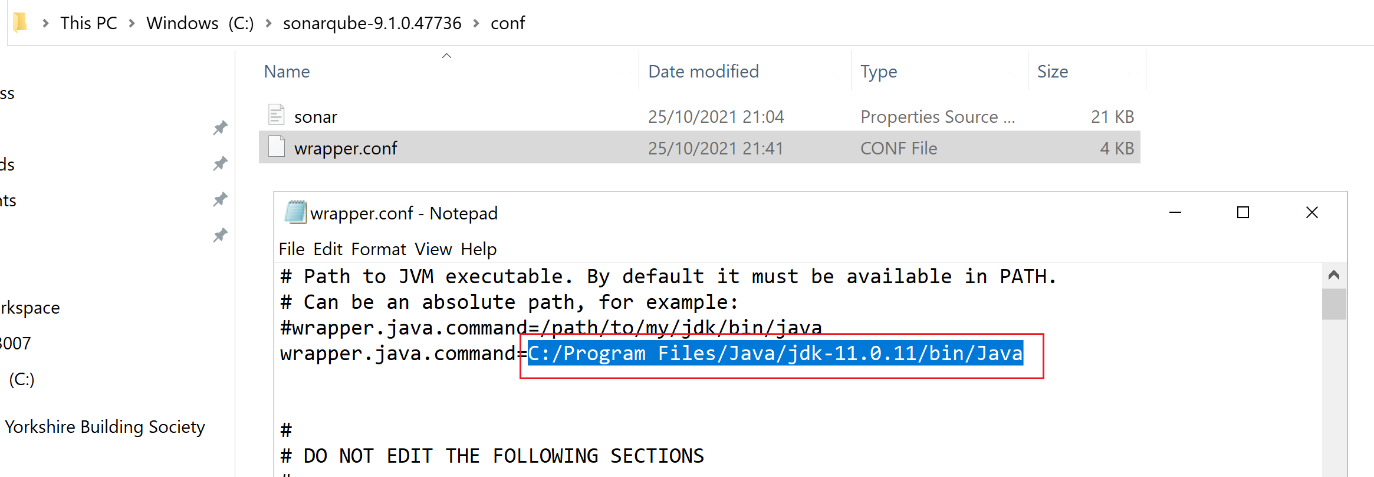
**Step 1:** Download SonarQube on the VM -> Extract the zip file and paste file into C drive



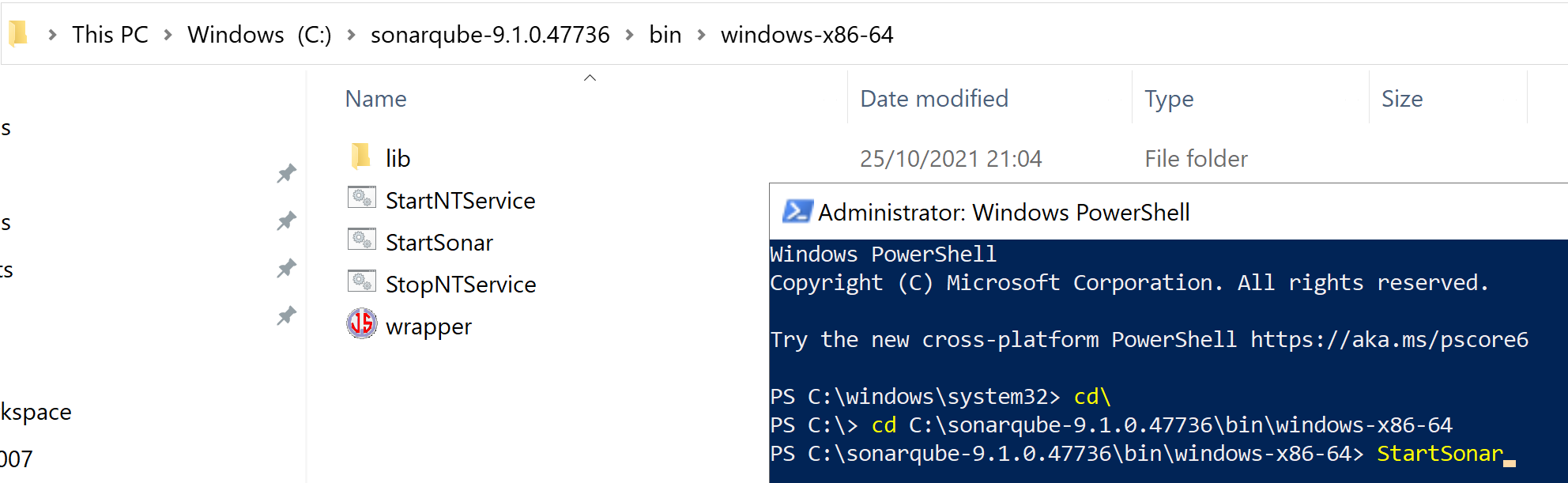
**Step 2:** Set the environment variable



**Step 3:** go to config folder -> open the wrapper.config -> Provide the JDK path

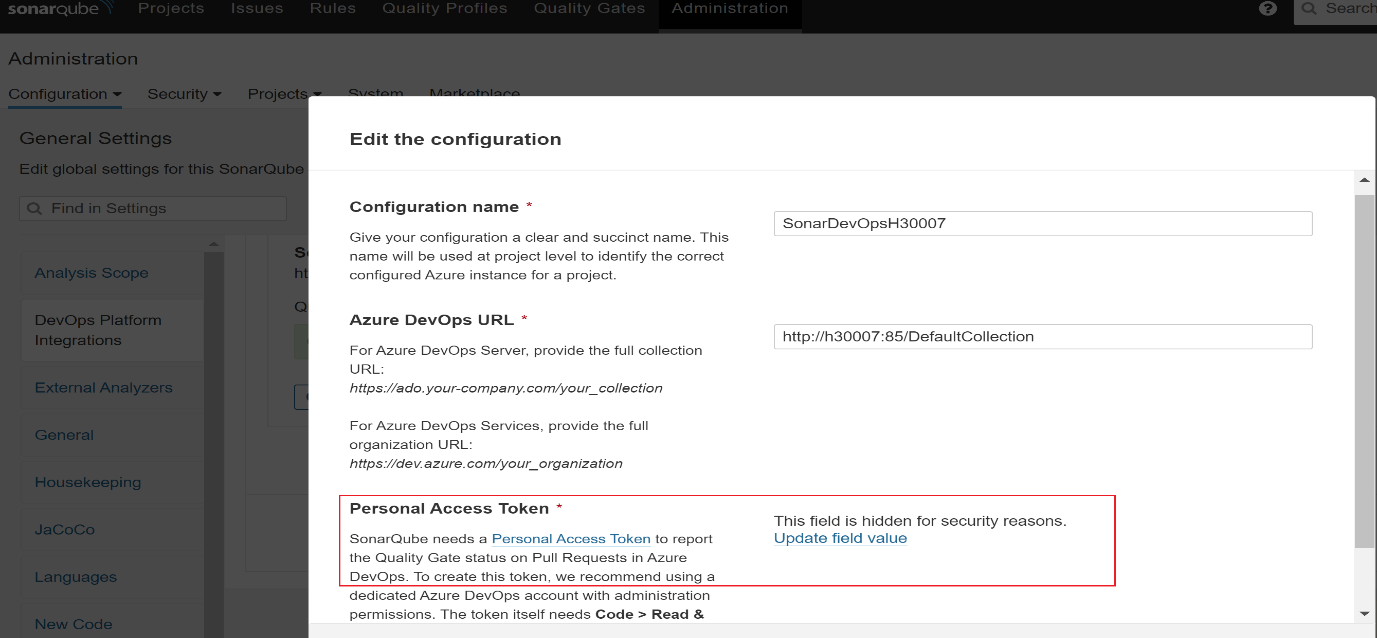


**Step 4: Open the PowerShell -> reach to the directory where StartSonar.bat file available-> run the command**

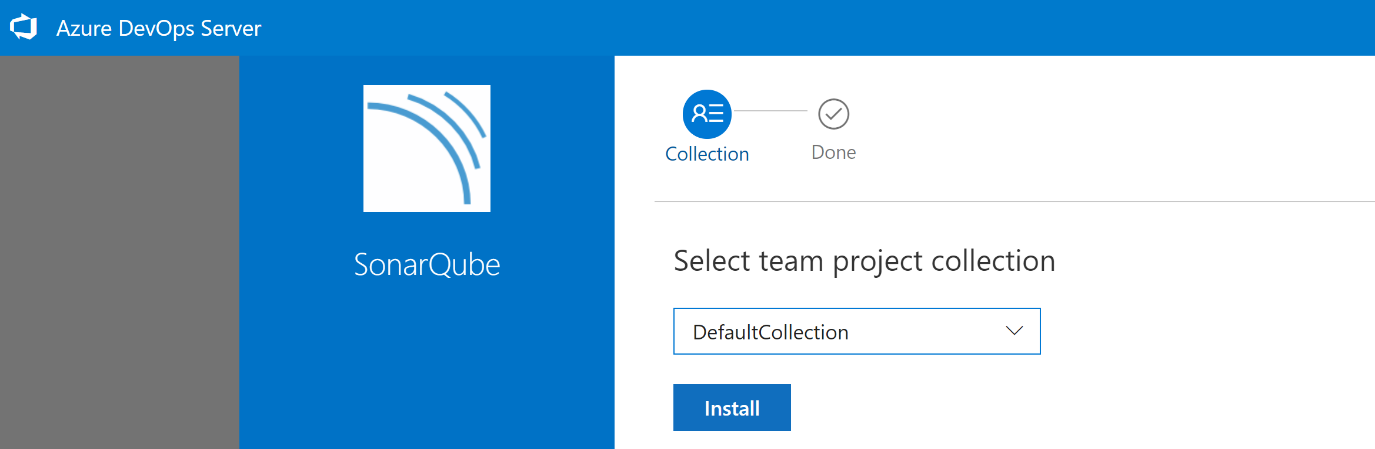


**Step 5:** Browse Sonar Cloud -> localhost:9000

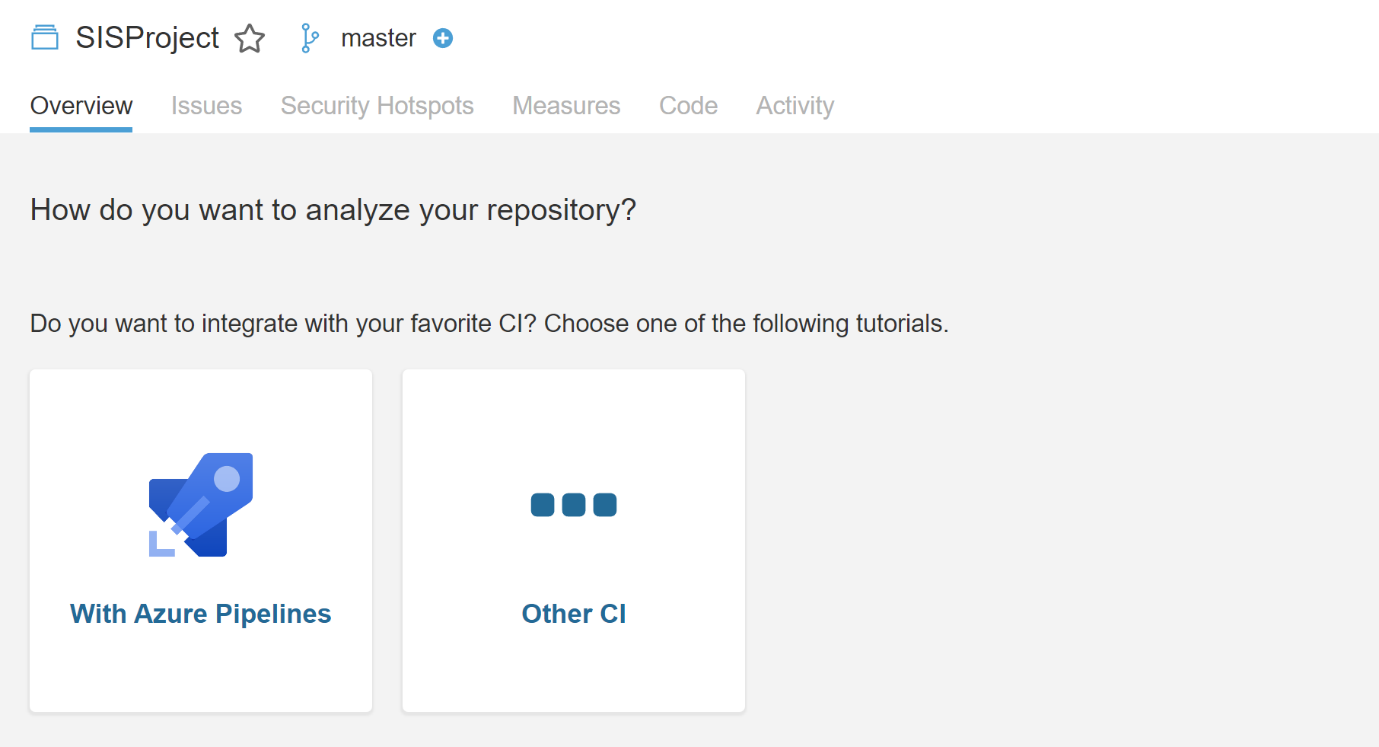
**Step 6: Configure Azure DevOps integration ->** Give configuration Name, Azure DevOps URL and Provide Personal Access Token



**Step 7:** Once Integration done -> go to Azure DevOps extension -> Browse Market place -> install SonarQube



**Step 8:** Configure the repository to analyse the code.



**Note:** For Azure DevOps on Cloud Platform ->

* We first go to website of Sonar Cloud -> register with same User credentials as in DevOps ->
* Then we create a service connection from Azure DevOps to Sonar Cloud -> It will ask Sonar Token
* Go to Sonar Cloud -> User Account -> Security -> Create Token -> Add token into service connection in DevOps

**Technical Debt:** Reasons that causes the technical debt

* When developers are under pressure to go ahead and deliver core, the application code may not be up to the mark. Hence sacrifice on the quality of the code being developed.
* Code developed that are tightly coupled and when in future it require changes in functionality it might become difficult.
* Sometimes the design is not done properly, the development will also not be done properly.

When we integrate our project with SonarQube or Sonar Cloud -> Go to Sonar portal -> Project -> Maintainability -> You will see the technical debt graph.

So the sonar cloud tool has the ability to go out and look into your code and based on certain factors, it can go out and give you a metric on the technical debt.

Question: Can we define approvals on YAML build pipeline?

Question: When we create build pipeline -what runs parallel and what runs sequentially?

1. List out some famous tools of DevOps?

Jenkin, Bamboo, JIRA, Slack, Git, Bitbucket, GitHub

2. Mention various phases of DevOps?

**Plan:** plan is essential for any application which requires the development, preparing plan for the process of development is the best practice.

**Code:** The code of application is designed according to the needs of users.

**Build:** with the help of different codes that are generated in earlier steps, the build of the application created.

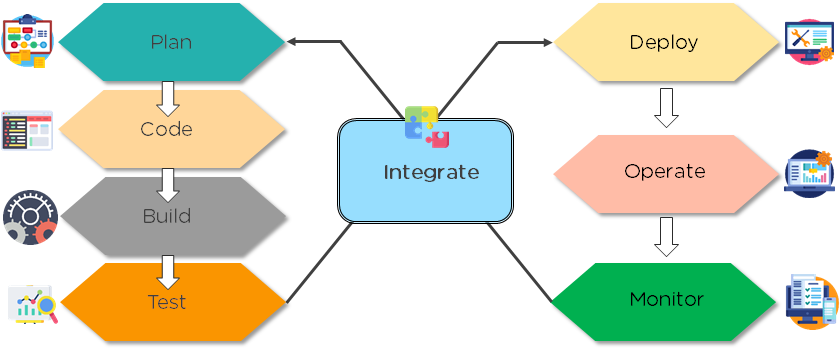
**Test:** Test plays an essential role in the development of the It application, which helps to test the application and it re-built the application when required.

**Integrate:** It is used to collaborate the various codes, which are from multiple programers.

**Deploy:** It helps for the future use by forming the environment of the cloud from the deployed codes, and it maintains continuous functioning even though the new changes take place in the website with a high traffic.

**Operate:** When it is necessary operations take place on code.

**Monitor:** For the customer needs, the performance of the application is monitored.



4. Explain the test plans of Azure?

Azure Test Plans is a service provided by Azure DevOps. It provides a browser-based test management solution along with crucial capabilities in user acceptance testing, exploratory testing, and planned manual testing.

Azure Test Plans, additionally, focus on DevOps on automated testing. It combines the contributions from developers, managers, testers, product owners, and user experience advocates and enhances the quality of a project.

5. What are Azure DevOps Projects?

Azure DevOps Project is a simplified way to effectively bring existing code and Git repository for the creation of CI and CD pipeline to Azure.

6. What is the difference between Azure DevOps Services and Azure DevOps Server?

This is one of the trickier Azure DevOps interview questions. Azure DevOps Services is the cloud service of Microsoft Azure and it offers a highly scalable and reliable hosted service that is globally available. DevOps Server, on the other hand, is an on-premise offering that is on a SQL Server back end.

7. Explain about break fix issues.

Break fix issues are nothing but the technical issues, that term belongs to industrial which includes work with technical support when it fails in normal functioning. This failure needs supportive organizations interruption for working order restoration.

8. What is continuous testing and the ideal DevOps tools for the same?

Many opensource tools for test automation can help in achieving the DevOps objective of continuous testing. Some of the notable DevOps tools for continuous testing are Selenium, JMeter, AntUnit, JUnit, SoapUI, and Cucumber.

9. What should you do to make a NuGet package available to anonymous users outside your organization alongside minimizing the number of publication points?

The solution to this question is the creation of a new feed for the package. Packages hosted in Azure Artifacts find storage in a feed. Setting up permissions on the feed enables sharing packages with higher scalability according to the scenario’s requirements. The multiple feeds on Azure Artifacts help in controlling access to packages across four levels of access. The four levels of access are owners, readers, contributors, and collaborators.

10. What are the necessary components for integrating Azure DevOps and Bitbucket?

The solution to this question refers to a self-hosted agent and an external Git service connection. GitLab CI/CD is compatible with GitHub and Git server like Bitbucket. Rather than shifting an entire project to GitLab, it is possible to connect external repository to obtain the benefits of GitLab CI/CD.

11. Explain the differences between git pull and Git fetch?

|  |  |
| --- | --- |
| **git fetch** | **git pull** |
| Fetches the changes from the remote repository but doesn't merge them with the local repository. | Fetches the changes of the branch from the remote repository and merges them with the local repository. pull = fetch + merge |
| The local repository is unchanged, but the central repository is updated. | The changes from the central repository are updated in the local repository. |
| Done at latter stages, since no merging is involved, there are no conflicts to be resolved. | There are chances of merge conflicts if two or more people are working on different copies of the same code/file. |

12. Git is a popular DevOps tool. Tell us how you will revert a commit that has already been pushed and made public.

Answer: There are two ways of doing so:

By creating a new commit to undo all changes made by the commit that has already been pushed and made public. Following command is used for doing so: git revert

By fixing or removing the bad file in a new commit and then pushing it to the remote repository. After making necessary changes to the file, commit it to the remote repository using the command: git commit -m “commit message”

Question: What is the basic premise of DevOps?

Answer: DevOps is a cultural shift wherein a collaboration and operation teams work together throughout the product or service life cycle.

Question: Which methodology is DevOps related to?

Answer: DevOps is related to Agile methodology.

Question: What are the priorities in DevOps?

Answer: The priorities in DevOps include resource management, teamwork, and communication.

Question: What are the different advantages of Git?

Answer: Git has the following advantages:

It helps in data redundancy and replication.

It is highly available.

It supports collaboration.

It can be used for a variety of projects.

It uses only one Git directory per repository.

It supports disk utilization.

It offers higher network performance.

Question: Can we handle merge conflict in Git?

Answer: Yes, we can handle merge conflict in the following three steps:

Step 1: Developing a clear understanding by checking everything using Git status.

Step 2: Mark and clean up the conflict by applying the merge tool.

Step 3: Performing commit and merging with the current branch along with the master branch.

Question: What is Forking Workflow?

Answer: Forking Workflow gives every developer with their service side repositories, thereby supporting open source projects.

Question: How is Forking Workflow better than Git Workflow?

Answer: Forking Workflow is better than Git Workflow because it helps in integrating the contributions of different developers without needing everyone to push to a single central repository for cleaning project history. Thus the developers are allowed to push their server-side repositories, and thereby on the project maintainer will push to the official repository.

Question: What is Git rebase?

Answer: Git rebase is a command which is designed to integrate the changes from one branch to another brand.

Question: How is Git rebase different from Git merge?

Answer: Git rebase is different from Git merge because, in the case of Git rebase, the feature branch is transferred to the master branch's ending point. However, in the case of Git merge, the merging adds to a new commit in the history wherein history does not change, but the master branch is changed.

Question: Can we move or copy Jenkins from one server to another?

Answer: Yes, we can move or copy the Jenkins from one server to other. For instance, by copying the Jenkins jobs directory can be moved from the older server to the new server. This way, installation can be moved from one installation to another by copying in the corresponding job directory.

Question: Can we make a new copy of an existing Jenkins job?

Answer: Yes, we can make a new copy of an existing Jenkins job by creating a clone of the directory in a different name.

Question: What is the difference between continuous testing and automation testing?

Answer: In continuous testing, the process of executing the automated test is part of the software delivery process. In contrast, automation testing is a process wherein the manual process of testing is applied wherein the separate testing tool helps the developers in creating test scripts that can be executed again and again without any kind of manual intervention.

How is DevOps different from agile methodology?

DevOps is a culture that allows the development and the operations team to work together. This results in continuous development, testing, integration, deployment, and monitoring of the software throughout the lifecycle.

Agile is a software development methodology that focuses on iterative, incremental, small, and rapid releases of software, along with customer feedback. It addresses gaps and conflicts between the customer and developers.

Question: Difference – Continuous Delivery & Deployment

Continuous delivery - when it comes to continuous delivery, so here the pipeline needs to be able to deploy the application on to a production-based environment. This entire thing might not be automated.

Continuous deployment - which is the next level of continuous delivery. This is where the entire process from the code commit on to the deployment is completely automated.

You need to have a lot of trust in the code changes because over here we are saying that as soon as the code changes are pushed on to the repository, they should be deployed onto the production-based environment.