Introduction of Jenkins

* Jenkins is the open-source automation server written in java
* It supports all continuous integration and continuous deployment-related things right from build, test, and deploy in the software development
* CI/CD is the main part of DevOps and many tools in the market are used by many organizations like Jenkins, TeamCity, Azure DevOps, and many more.
* This makes it easier for a developer to just focus on software development for better productivity by integrating changes rather than focusing on the build, testing, and deploying of the application

Features of Jenkins

* Open source
* Easy to configure and Installation
* Multiple plugins
* User-Friendly Interface

Jenkins Continuous Integration

* In the continuous integration process we integrate all the latest code changes which are committed by the developer into the git and any other.
* In real-time multiple developers are working on the same branch and EOD they will commit their work to the git.
* Later on, the Jenkins server will trigger whenever the new code is committed, run a test case, and create an error-free build for many environments like QA, DEV, and Production as per our need.
* If some build and unit test cases failed then the user will get notified about that and again the new code will come and this process is continuously work
* If you want to learn more about Jenkins then read the official documentation ([jenkins.io/doc](https://www.jenkins.io/doc/)).

Installation of Java and Jenkins

Java

**Step 1**

Install Java 11 for Jenkins

<https://www.oracle.com/java/technologies/downloads/#java11>

**Step 2**

Open the exe file and installed Java on your system

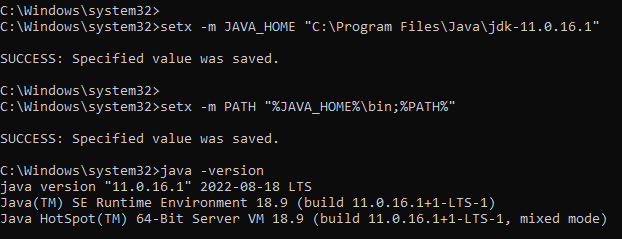
**Step 3**

Set the environment variable of Java Path using CMD administrator mode

*setx -m JAVA\_HOME "C:\Program Files\Java\jdk-11.0.16.1"  
setx -m PATH "%JAVA\_HOME%\bin;%PATH%"*

**Step 4**

Check whether the Java is properly installed or not as I showed below



Jenkins

**Step 1**

Install Jenkins

<https://www.jenkins.io/download/>

**Step 2**

Open the exe file and select local service as a logon type

**Step 3**

Configure port number (default port is 8080) but I set 8888 you can configure as per availability

**Step 4**

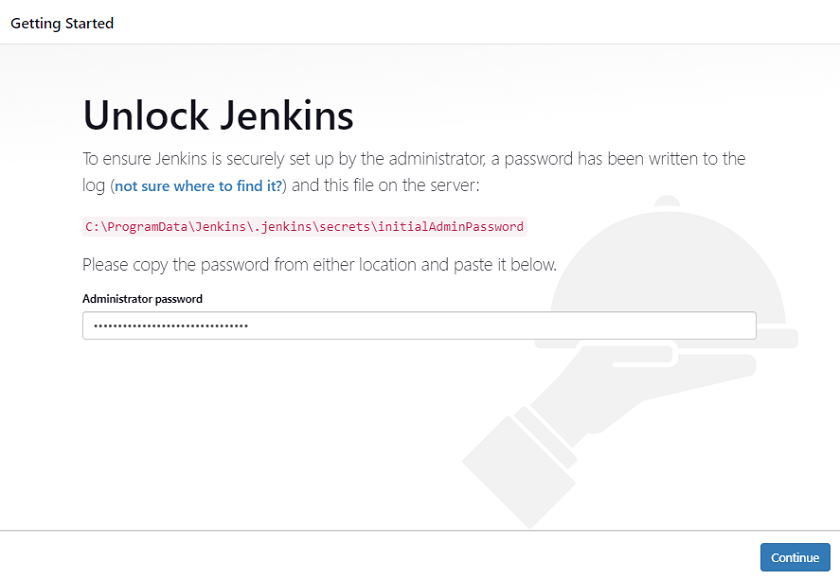
Provide Java JDK path of bin folder which are present in Java folder inside Program Data

**Step 5**

After successful installation of the Jenkins, the server opens the URL in the browser(*http://localhost:8888/*)

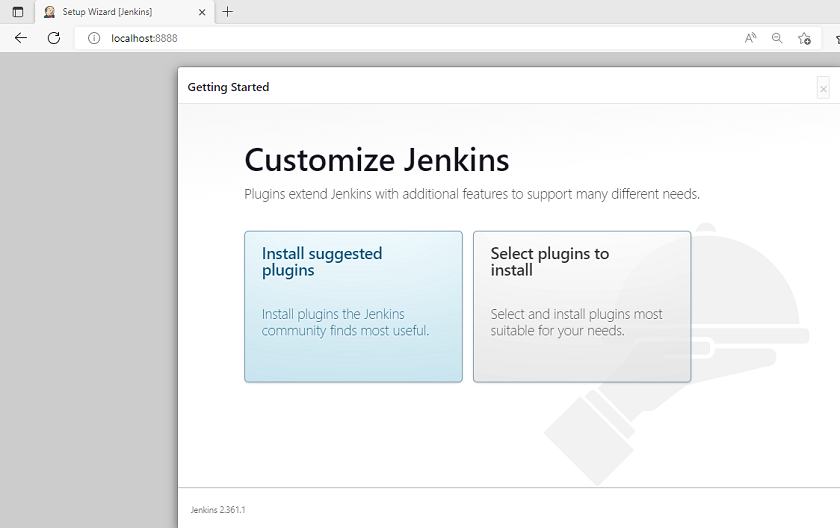
**Step 6**

You can see one inside that provides an admin key



**Step 7**

Customize Jenkins plugin as you wish. I installed Git, MS Build MS Test, and a few others for .NET Application

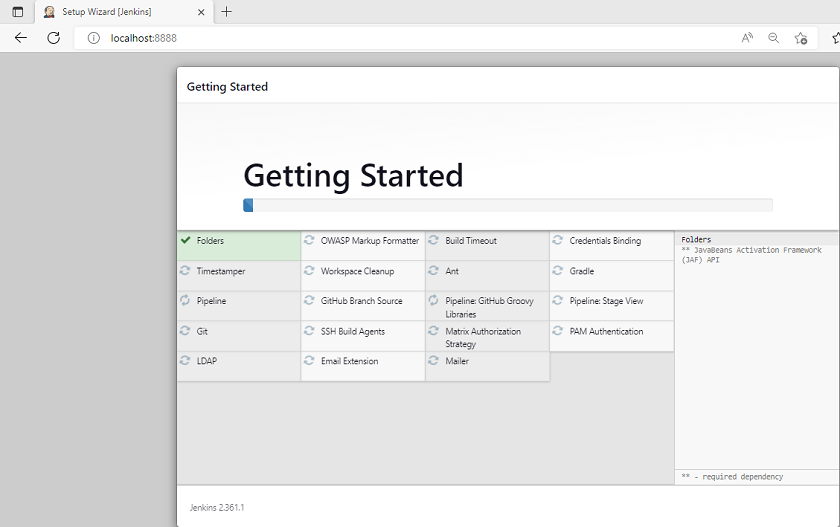


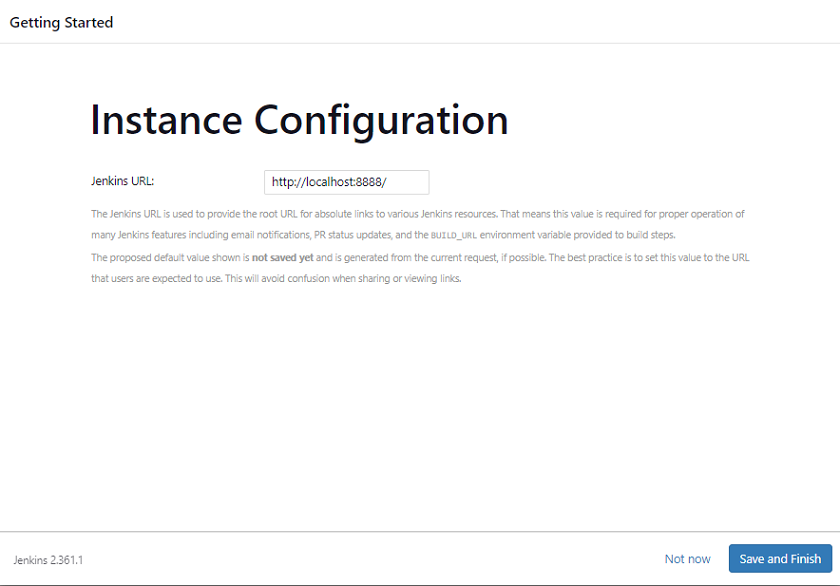
**Step 8**

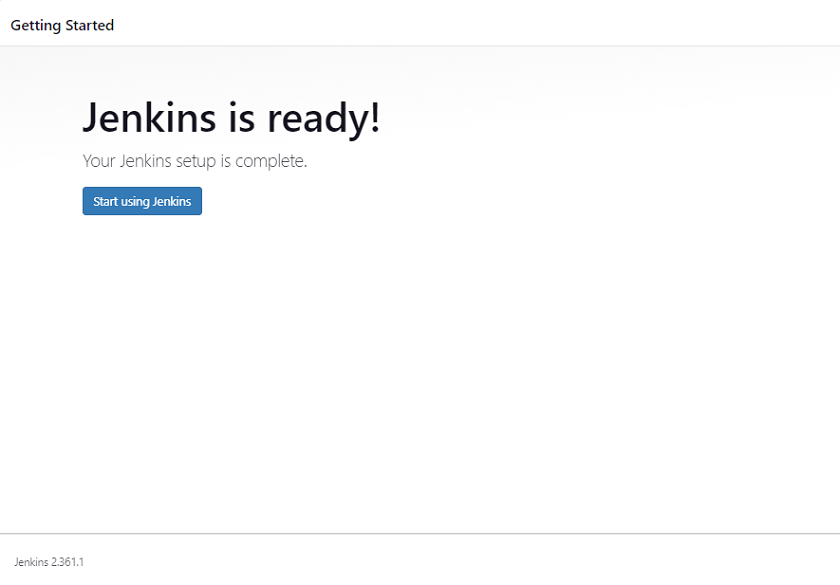
Set user credentials which you need while login each time on Jenkins Server.

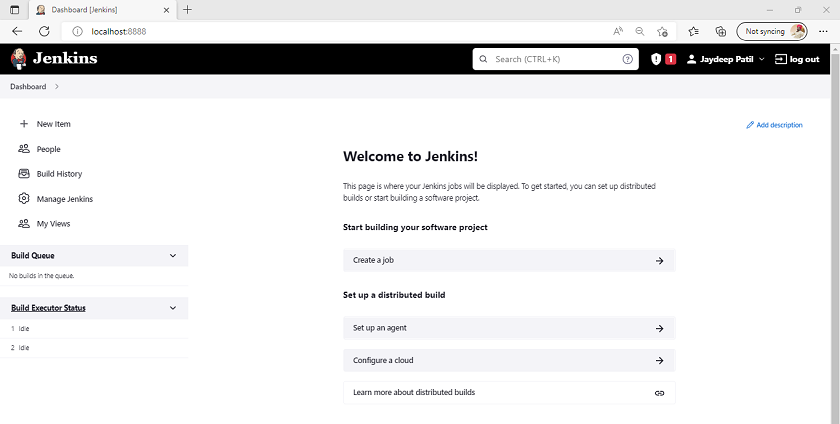
**Step 9**

It will take some time to install default plugins



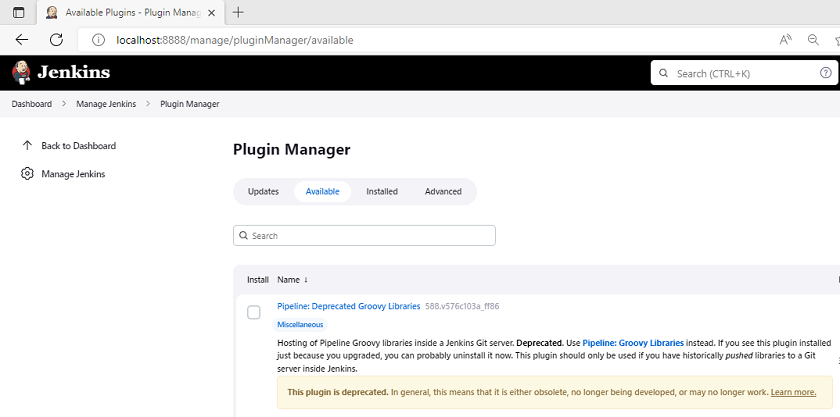






**Step 10**

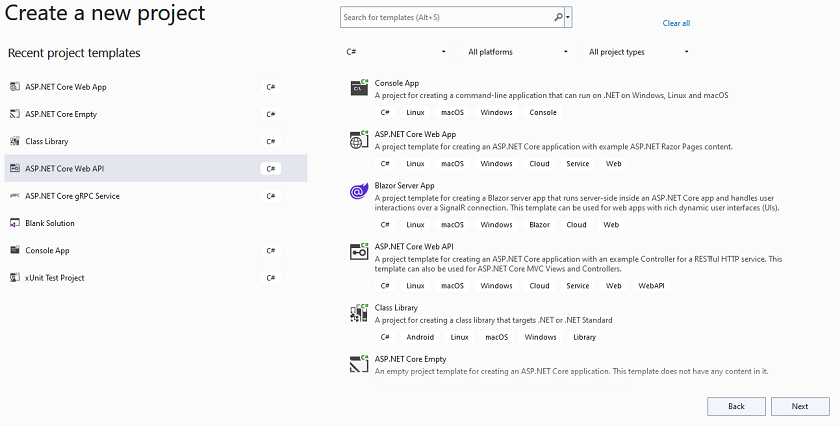
Go to the Manage Plugins inside Manage Jenkins and installed necessary plugins which you want for .NET Application I installed, MS Build, Git, MS Test, and a few others.



Step by Step Implementation of.NET Core Application

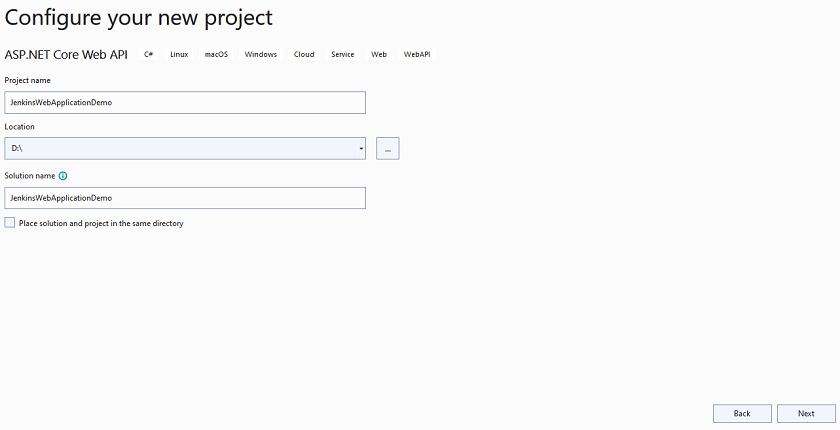
**Step 1**

Create a new Web API



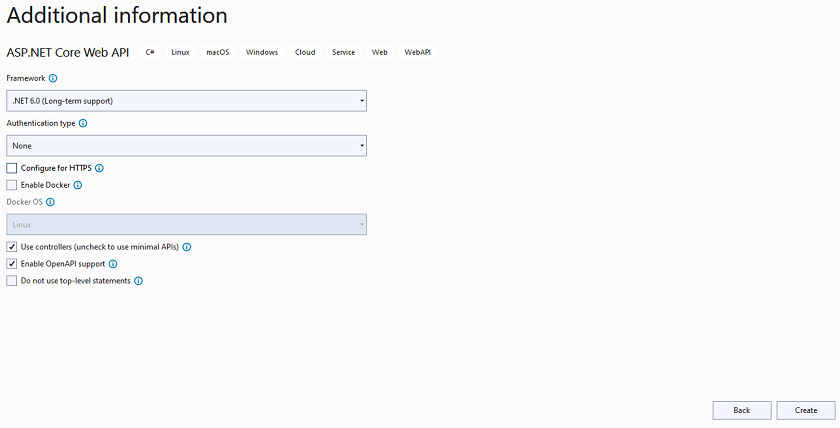
**Step 2**

Configure the application



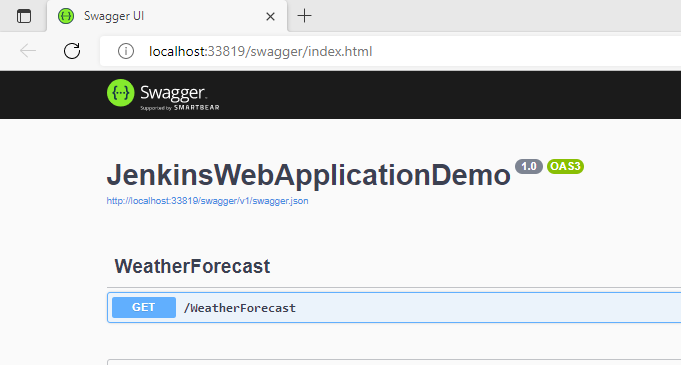
**Step 3**

Provide additional information



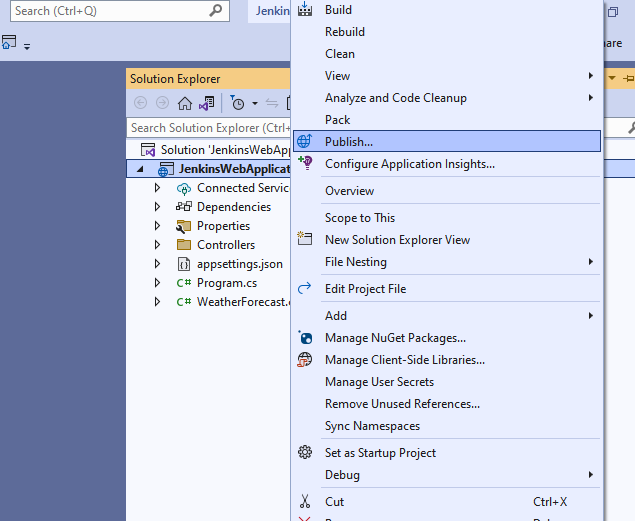
**Step 4**

Finally, run the application



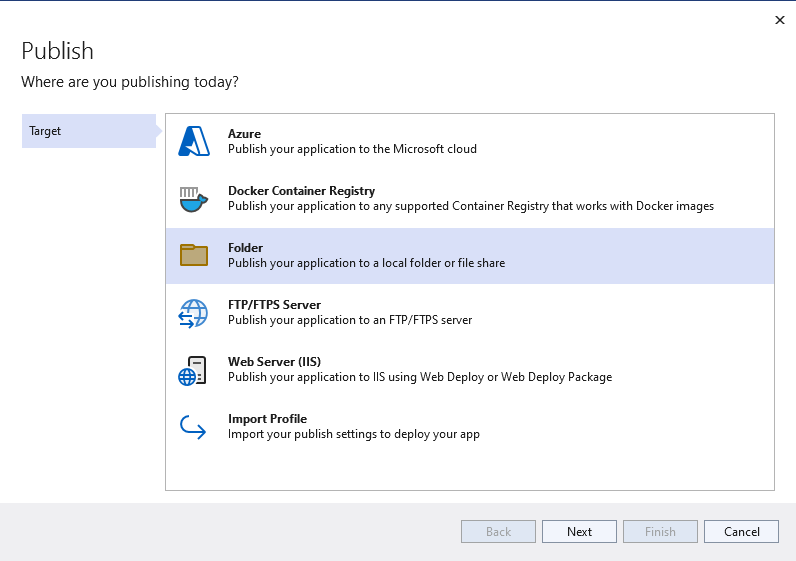
**Step 5**

Now we are going to publish this code



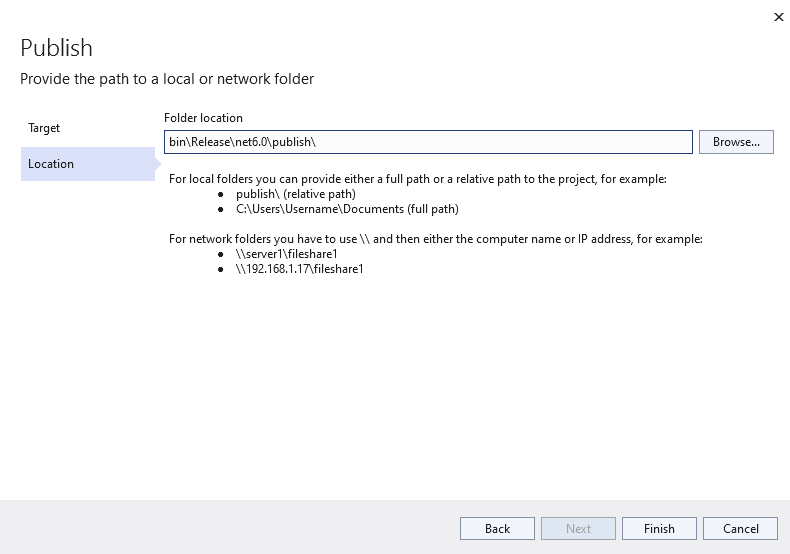
**Step 6**

Select folder



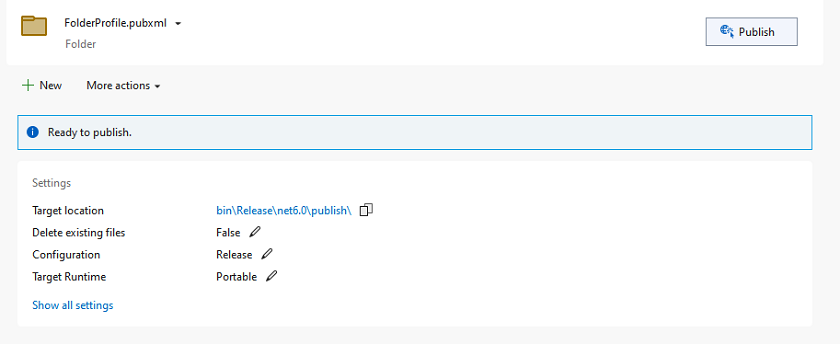
**Step 7**

It will take default publish path



**Step 8**

Finally, publish



**Step 9**

Go to the property section of project inside solution and edit FolderProfile.pubxml file and change the web publish method to Package



**Step 10**

We also need Microsoft Web Deploy, and most of time it will come with Visual Studio

**Step 11**

Later on, we just create an empty xUnit Test Case project just for demo purposes inside the same solution

**Step 11**

Run your project and confirm all things are working fine on your system before committing to git.

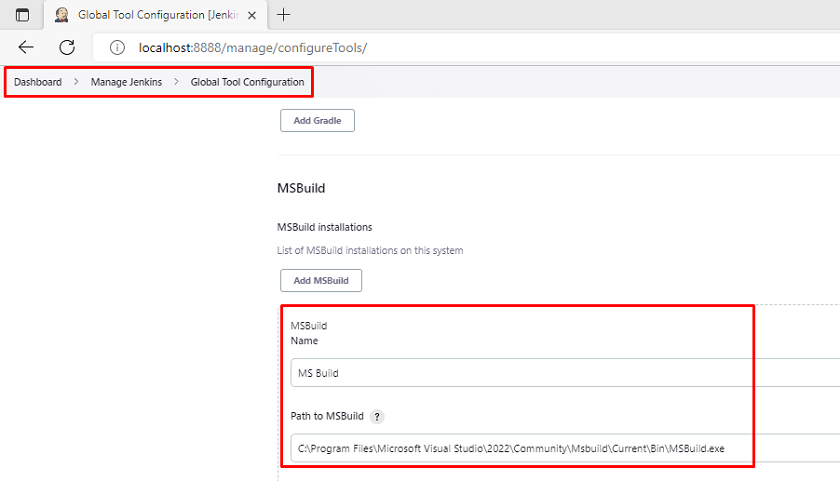
**Step 12**

Next, create a git repository and push your code into that

Jenkins Pipeline and Configuration

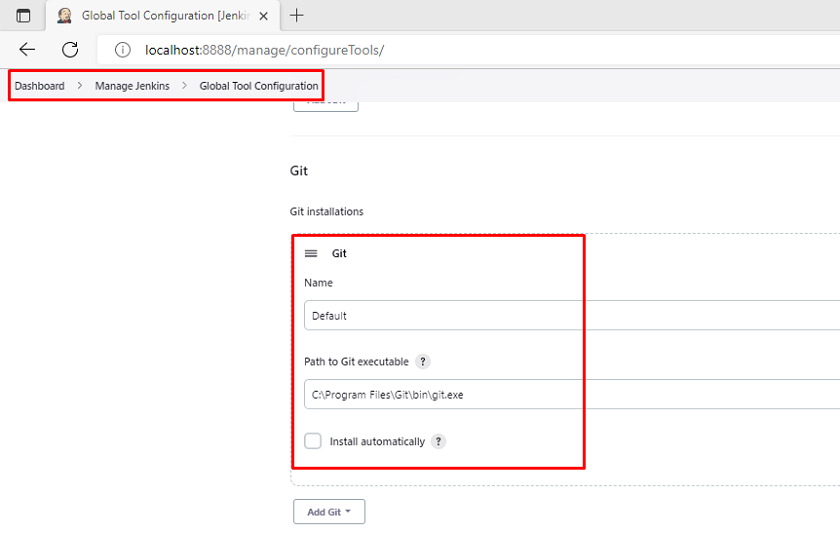
**Step 1**

First, we are going to add MS Build Path and Configuration for that go to the Global Tool Configuration inside the Manage Jenkins



**Step 2**

Next, we add the git file path inside the global tool configuration



**Step 3**

Now, we are going to add git credentials in Manage Credentials inside Manage Jenkins click and **global credentials(unrestricted)** add and your git credentials like username and password it will create credential id automatically once you click on save and that id you need to put inside pipeline configuration.

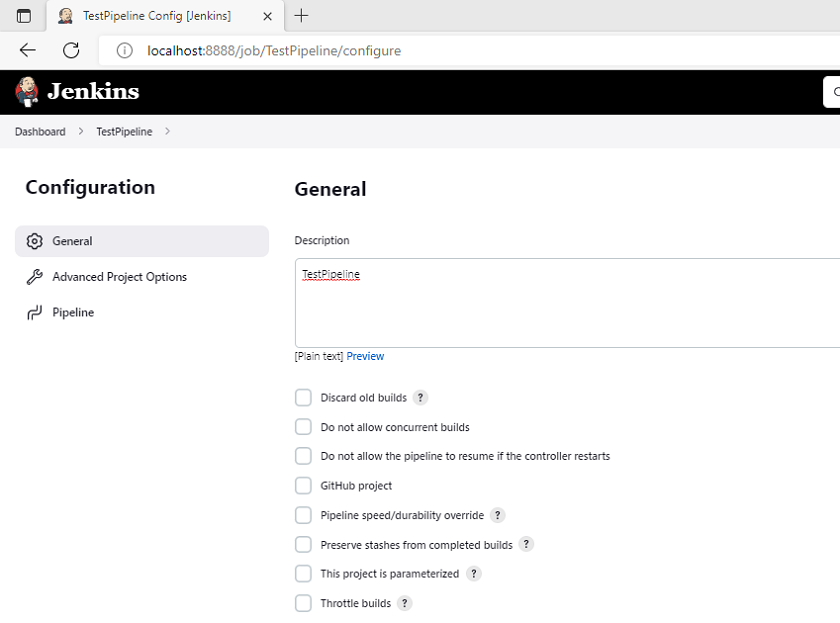
**Step 4**

Open Jenkins Dashboard and click on New Item to create a Pipeline

The pipeline is basically the set of steps that are going to execute before creating the build that includes unit test, integration test case, and many more as per need

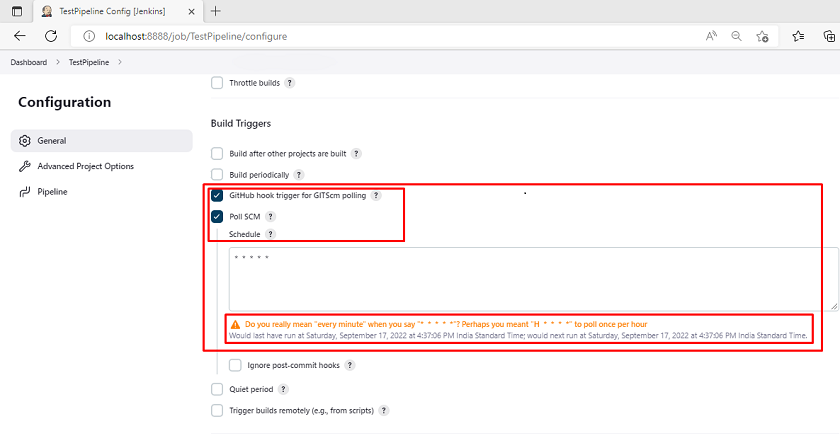
**Step 5**

Provide general information like a description



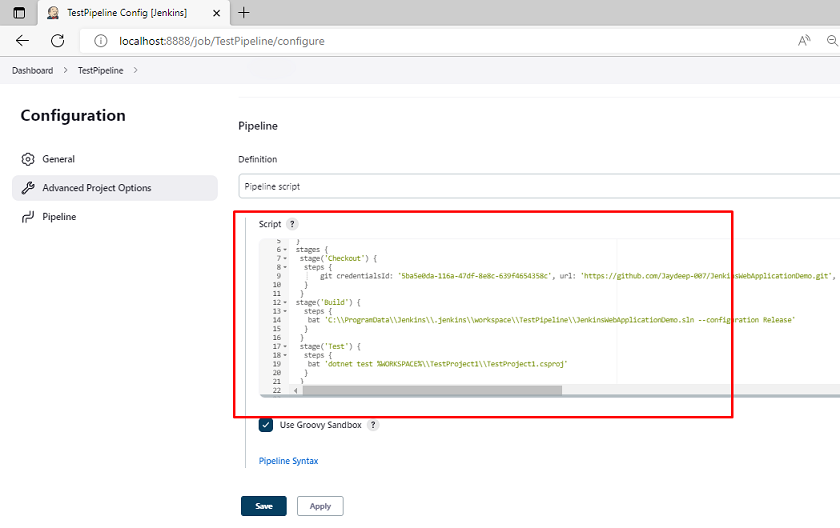
**Step 6**

Next, click on a few checkboxes which help us to automatically trigger the pipeline whenever we commit code inside git repositories and put five start separated by spaces that is us to trigger pipeline instantly whenever the developers commit the code



**Step 7**

Add the following pipeline script which is basically a set of steps that are going to executes before creating the build (Note- Please make sure all your paths are corrects otherwise it will throw errors in the build process)



pipeline {

agent any

environment {

dotnet = 'C:\\Program Files\\dotnet\\dotnet.exe'

}

stages {

stage('Checkout Stage') {

steps {

git credentialsId: '5ba5e0da-116a-47df-8e8c-639f4654358c', url: '<https://github.com/Jaydeep-007/JenkinsWebApplicationDemo.git'>, branch: 'main'

}

}

stage('Build Stage') {

steps {

bat 'C:\\ProgramData\\Jenkins\\.jenkins\\workspace\\TestPipeline\\JenkinsWebApplicationDemo.sln --configuration Release'

}

}

stage('Test Stage') {

steps {

bat 'dotnet test %WORKSPACE%\\TestProject1\\TestProject1.csproj'

}

}

stage("Release Stage") {

steps {

bat 'dotnet build %WORKSPACE%\\JenkinsWebApplicationDemo.sln /p:PublishProfile=" %WORKSPACE%\\JenkinsWebApplicationDemo\\Properties\\PublishProfiles\\FolderProfile.pubxml" /p:Platform="Any CPU" /p:DeployOnBuild=true /m'

}

}

stage('Deploy Stage') {

steps {

//Deploy application on IIS

bat 'net stop "w3svc"'

bat '"C:\\Program Files (x86)\\IIS\\Microsoft Web Deploy V3\\msdeploy.exe" -verb:sync -source:package="%WORKSPACE%\\JenkinsWebApplicationDemo\\bin\\Debug\\net6.0\\JenkinsWebApplicationDemo.zip" -dest:auto -setParam:"IIS Web Application Name"="Demo.Web" -skip:objectName=filePath,absolutePath=".\\\\PackagDemoeTmp\\\\Web.config$" -enableRule:DoNotDelete -allowUntrusted=true'

bat 'net start "w3svc"'

}

}

}

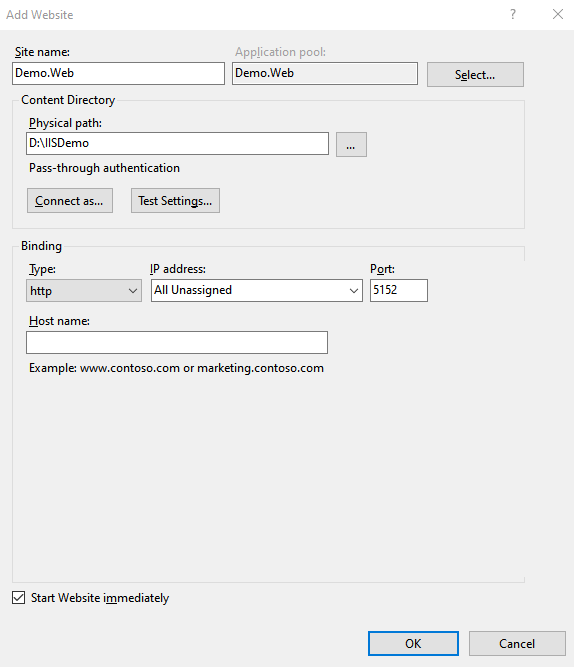
}

JavaScript

Copy

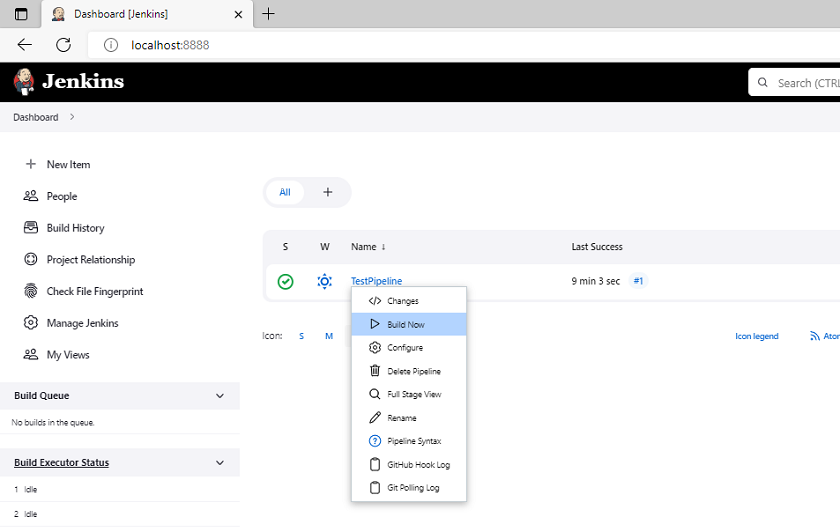
**Step 8**

Open the IIS and create empty website and point to the any empty folder in that our pipeline will publish and deploy our code after build is getting completed



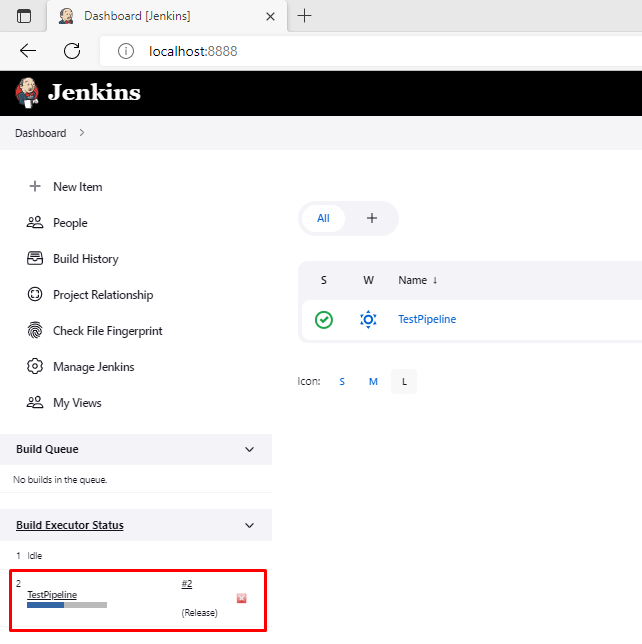
**Step 9**

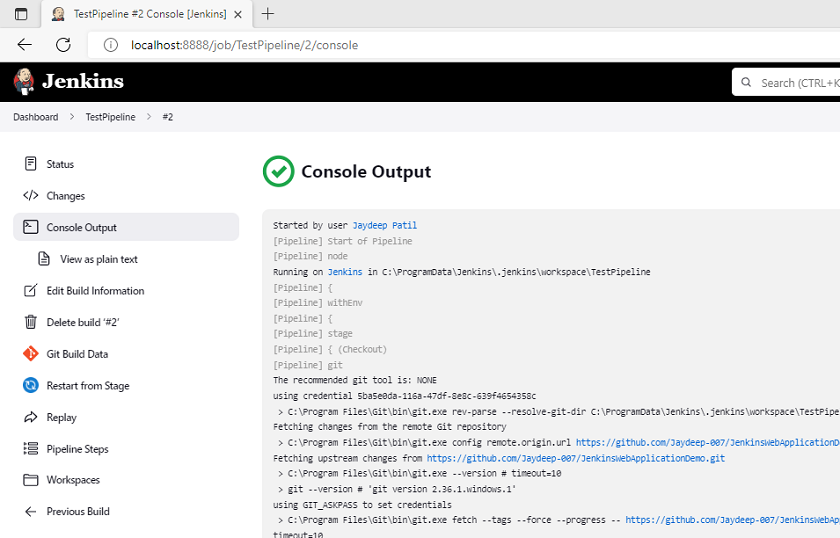
After that go to the dashboard and run your pipeline

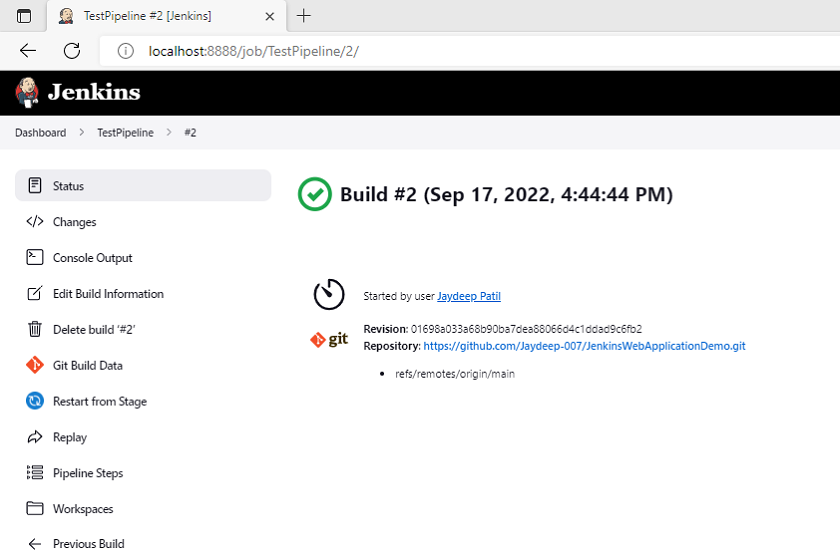


**Step 10**

Here you can see the build is started







Now whenever you change add commit the code then build is triggered and the pipeline is going to create a new build for us and deploy it inside the IIS. This process is continuously running. Also, if the build failed due to some test cases and something like that then you can also configure the email using SMTP protocol inside Jenkins that helps us build

**Step 11**

Finally, Open IIS and you can now access your application with the latest build

**Step 12**

Browse the URL from IIS and you can see the application is in running mode

