We are going to learn how to implement CI/CD pipeline using GITHUB ACTION in ASP.Net Core Application.

For the very beginning, we need to know what is **CI/CD**, what is P**ipeline** and what is **GITHUB ACTIONS.** There have three different things what we need to learn fast. If we don’t know about those things, then there has no meaning to learn.

1. **What is CI/CD???**

CI means Continuous Integrations and CD means Continuous Deployment or Continuous Delivery. It’s a very pretty much basic thing we all know very much well. This process we can call combined practice for a software development process. It’s a very essential thing for any kind of software development. Because, some developer using Mac, some are Windows, some are Linux etc. Different developer using different OS. Yes, It’s true that we can deploy directly from our local machine. But think, if your machine has some malware and when you deploy your code in cloud, the malware can go with this and everything screwed up.That’s why this thing is **more reliable, its repeatable & secure.**

1. **What is Pipeline???**

The steps that form a CI/CD pipeline are distinct subsets of tasks grouped. Typical pipeline stages include:

* **Build** - The stage where the application is compiled.
* **Test** - The stage where code is tested. Automation here can save both time and effort.
* **Release** - The stage where the application is delivered to the repository.
* **Deploy** - In this stage code is deployed to production.
* **Validation and compliance** - The steps to validate a build are determined by the needs of your organization. Image security scanning tools, like Clair, can ensure the quality of images by comparing them to known vulnerabilities (CVEs).

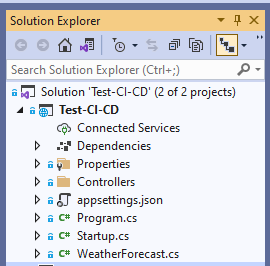
1. **What is GitHub Actions???**

Its compiled our application automatically using a build system. and the build system is GitHub Actions. It’s a feature from GitHub which allows you to automatically build your code into executable. That means, in the context of C#, the c# code is compiled into DLL with intermediate language without you having compiled the code in your local machine. If you are using for instance of C++ or Rust or GO this will compile into your machine language. If you are using typescript may be Angular, React, Vue then this workflow can turn your typescript code into compiled the minified the JavaScript code the you can use as your mobile app or web app or progressive app or so on.

So, all of these things we are learning theoretically. Now we are going to see practical example.

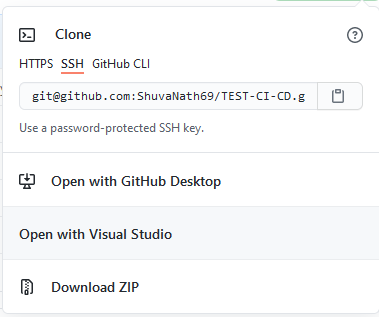
1. **Open Visual Studio 2019 => Create new project => Name Test CI-CD => Select .Net 5.0 => then Create.**

After creating the project, you will see all this file

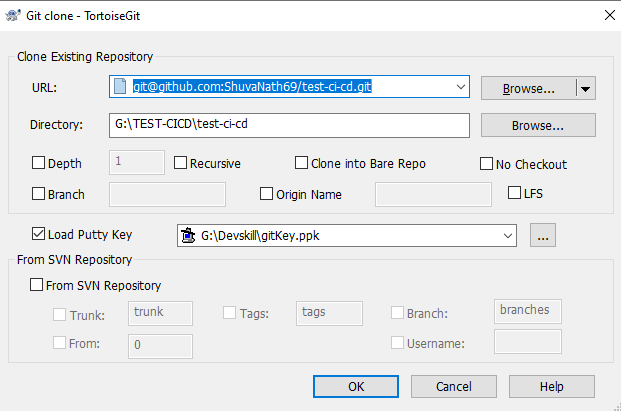


1. Go to your GitHub account and create a public repository. After creation copy the HTTP/SSH URL whatever you like and clone the repository. You can clone using command prompt/Tortoise Git.

**Step 1: Copy the HTTPS/SSH URL**

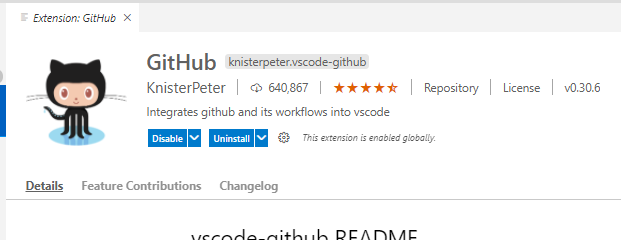


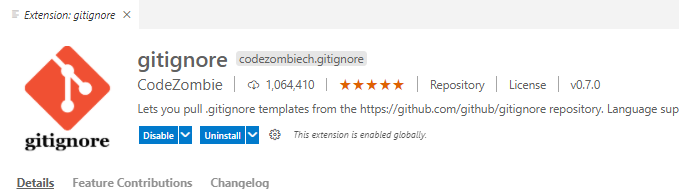
**Step 2: Clone the repository in your local machine**



**Step 3: After clone the repository, copy your project folder and paste into the repository folder. After paste, then will do Commit and push. Then you will see in your project in your repository.**

Now, Open the solution in Visual Studio Code. Install GitHub & GitIgnore extension

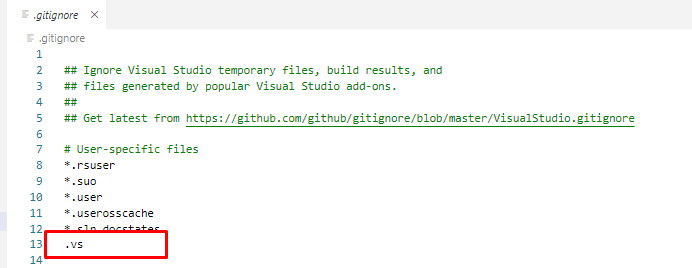




After installing those, add unnecessary extension into the gitignore file. You can add many ways.

* Go to the browser, and write gitignore.io and type visual studio then hit Enter. They will give you gitignore file.
* **Press F1 => Select add .gitignore => select visual studio => select append**. Then you will see that, an automated gitignore file will generate

**N.B: After adding gitinore file, please write .vs.**

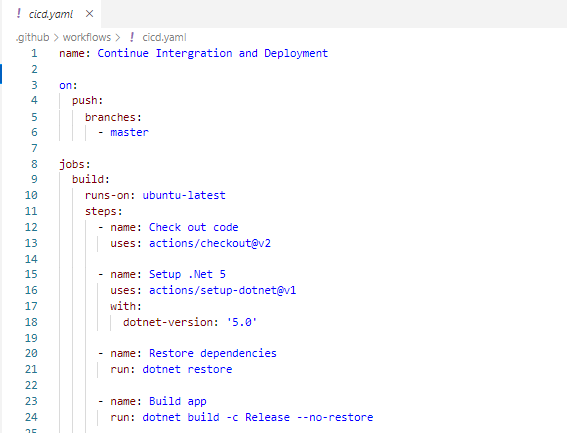


**Now, commit and push whatever we will change**

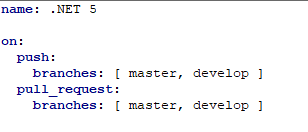
Now Follow the steps.

* create a new folder and name is. github.
* Under the .github folder create another subfolder which name is workflows.
* Then create a yaml (YAML aren’t markup language) file which name is ci-cd.yaml. Here, name is optional. You can use your own choice.

After open the yaml file, please write down step by step follow the attachment file below.

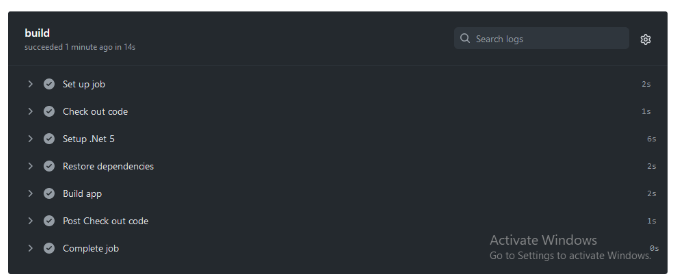


1. **Name:** This can be anything. Whatever you will prefer.
2. **Branches:** Here you see only master branch available in Push. But it can 2 branch. One is master branch another is develop branch (This one belongs to Gitflow). If we will maintain the Gitflow then it should look like this. About Gitflow please read the documentation about this. But here we will not discuss here about Gitflow.



1. **Jobs:** Now the important part begin. This job can multiple. But for learning purpose we will see one job. Under the job we will see build. Meaning is, where we want to build. It can Ubuntu or Windows or Linux.
2. **Steps:** There has multiple step we need to add here. Like GitHub Actions version, .Net version, install or Restore dependencies, application building and so on. So, very First beginning, we add GitHub Action version. Secondly, we add .Net version. We created a project using .Net 5 version. This is very important part. If your version mismatch, then you GitHub Action will show **build error.**
3. **Dependencies Restore:** This step, GitHub Actions restore all kind of essential dependencies for your projects.
4. **Build app:** Here GitHub Action build your application. –c stand for –configuration.

Now, commit the changes and push all file. Refresh the repository in your browser and you will see your yaml file will add. Then go Action Tab. You will see that, your application is building and after few second later you will see that, the application build in successfully.

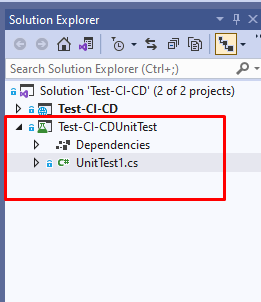


**So, Finally Continuous Integration (CI) progress is done. Isn’t it cool???**

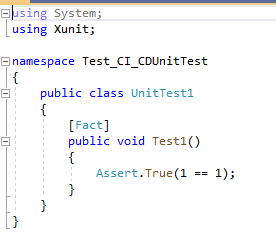
**Now we are going to see how to deploy in a cloud server using Microsoft Azure. This method is called Continuous Deployment.**

So, For Deployment you must do one thing first. It’s called Unit Test. Whatever your application it is; you must test your application. Most of case it will reduce your time and productivity will go higher. It’s highly recommended when an application will create.

For, testing our application you will create a new project which is XUnit.test. There has many more testing tool available. For example, NUnit Test, XUnit test etc. But here, we will not discuss about this.

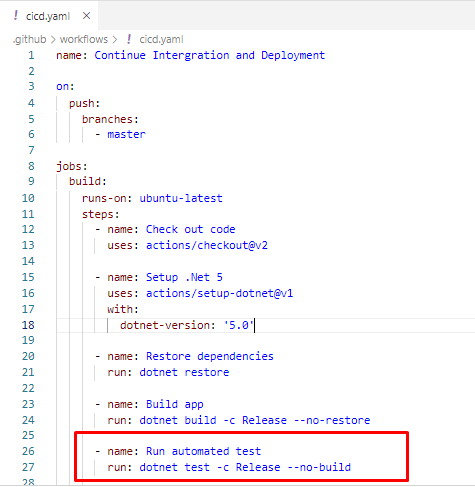


In UnitTest1 class we will write some code.



After writing this code, run the test explorer and will see that test run successful.

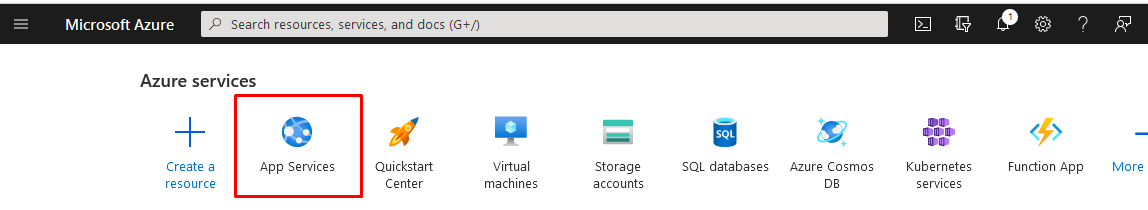
Now, again go to the Yaml file and add some code which has in image below.



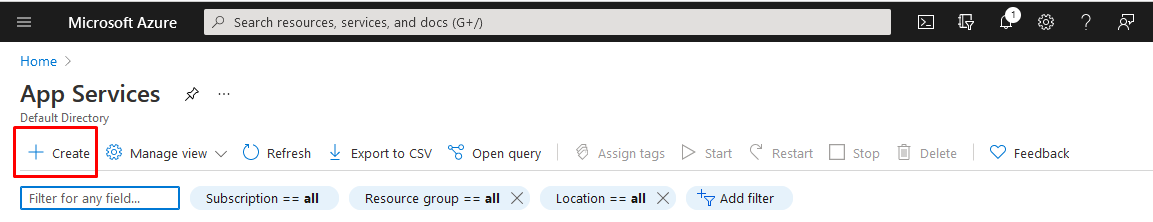
**Now, commit all the changes and push. Then go to the browser, refresh your repository, go to Action tab and after few second late we will see that build run successfully.**

After Unit Test, Update Yaml file, now we will go to publish our application in cloud server. For this, you need follow some steps.

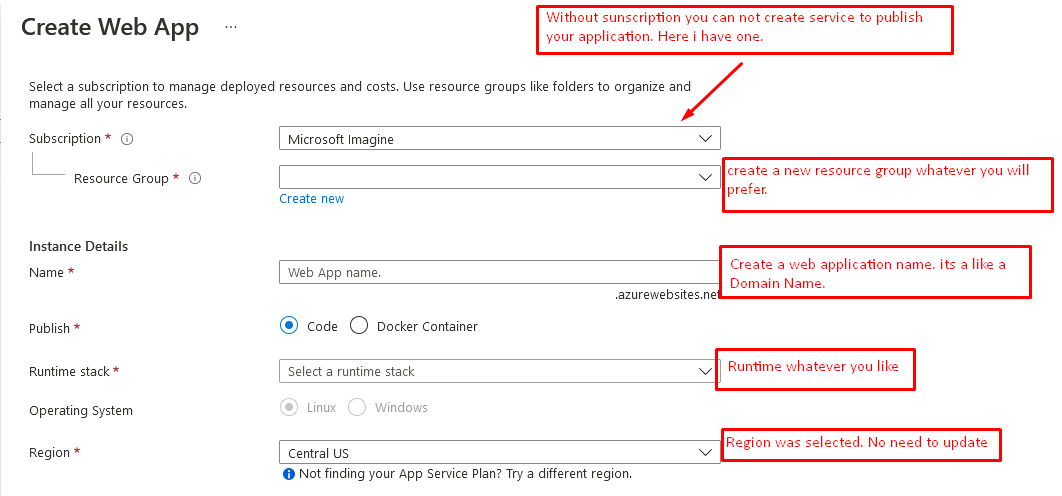
1. Go to browser and type portal.azure.com
2. Login using your Microsoft account.
3. Go to App Service



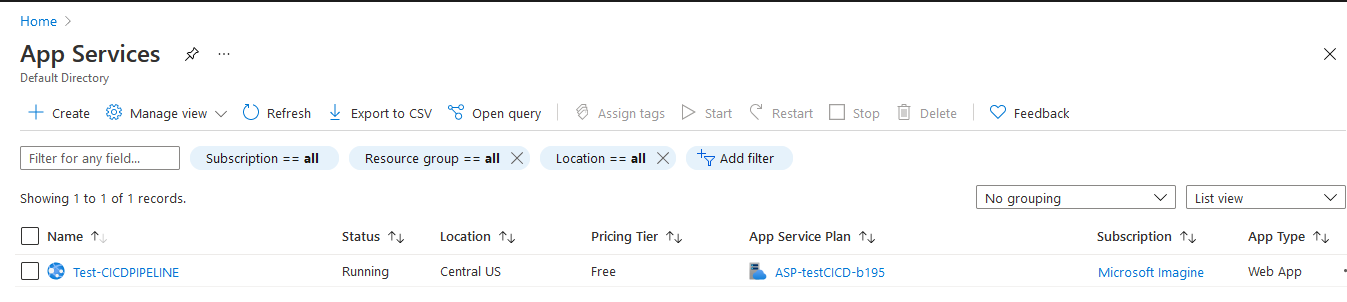
1. Go and click to Create Service (Here you will need Microsoft Subscription).



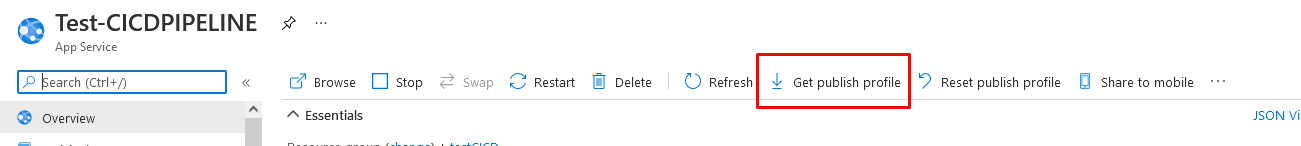
1. Then create a service



1. Then you will see, our app service will create in your dashboard



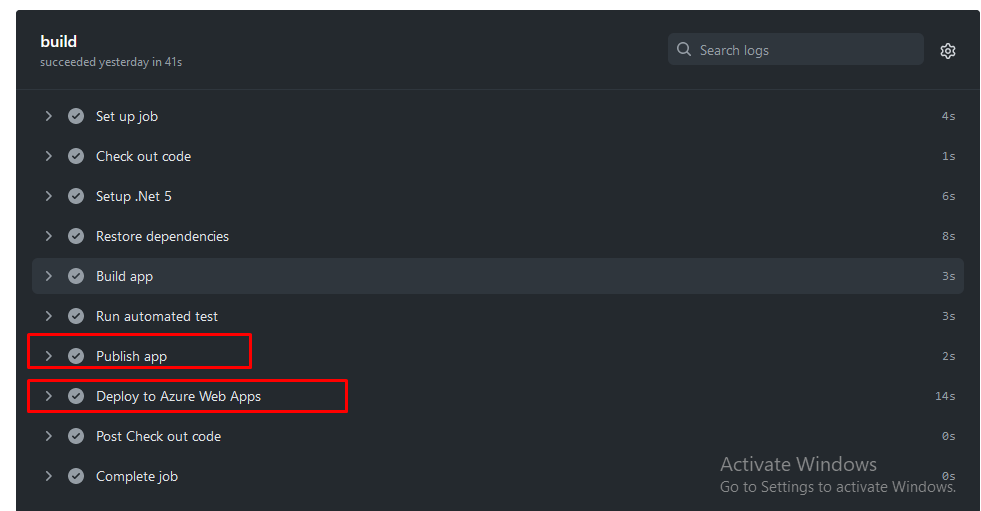
1. Click your service name & go into there. In your service, you will see a publish file download option.



1. **Download the publish file and open in notepad. Press Ctrl+A, and then Ctrl+C**
2. Go to the setting options in your repository. In **Setting** options, we will a **secret tab**. Enter this secret tab, click **New Repository Secret**, put the name **ZURE\_WEBAPP\_PUBLISH\_SECRET** (the name is optional), then Ctrl+V and publish. **Once you will publish; you can’t add again. Either it will update or delete.**
3. Again, go the YAML file and add some code again.



1. **Now Commit and push the all file. Go to the browser refresh your repository, go to the action tab you will Publish App and Deploy to Azure Web Apps run successfully.**



**Expand Deploy to azure Web Apps,** you will see **App Service Application URL.** Copy the URL and paste into the browser and you will see that, your application is running.

