CI/CD

Our organisation should seriously consider implementing the usage of CI/CD to change the way our developers are delivering our software products to our consumers and customers.

Adopting the practice of CI/CD might appear to be a huge shift in the way our developers produce and deliver our products. However, virtually all of the process steps shown below can be fully automated. The result of all this would save our organisation resources, time, effort and more importantly costs for all future work produced and delivered to our customers.

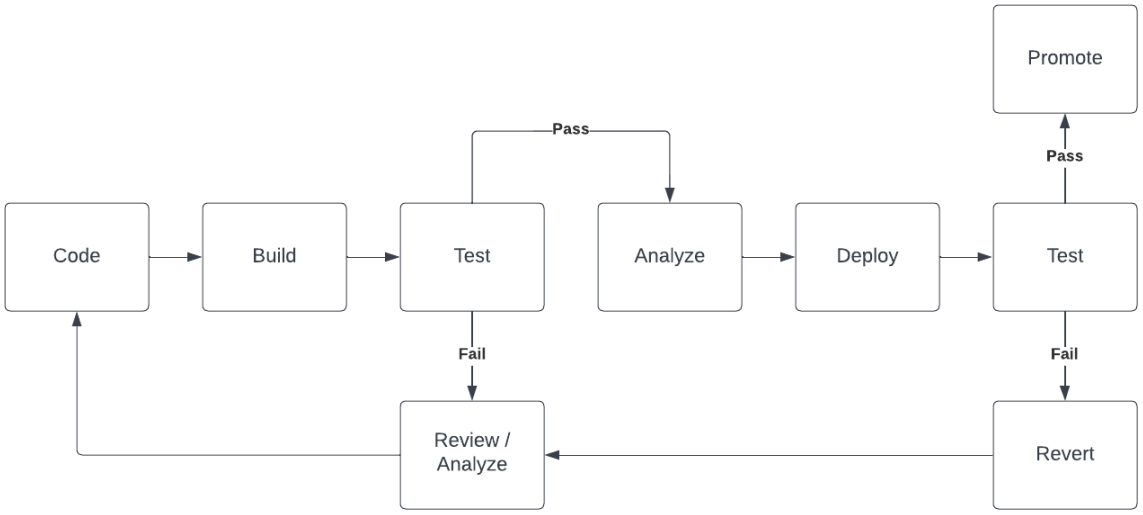
What is CI/CD ?

CI/CD is a combination of a number of different methods and practices which our developers can learn and start to use.

CI/CD is known as Continuous Integration and Continuous Deployment. This whole concept allows our developers to increase the speed of delivery of new products or product improvements and fixes.

This then also provides us with the idea of Continuous Delivery. What this means is that because of the time saving the automated steps provide us, we can then delivery new products and updates or fixes to our customers.

CI/CD works like shown in the following diagram:



Code & Build

Our developers can continue to work with the VSCode tool and develop our products as normal. But we can use tools such as Github to manage the builds. Github will allow us to maintain a master set of code for our products, our developers can work and deliver on modifications and improvements. Github then allows us to control and mange how all this work is combined and merged.

Test

For this stage we can use other systems available such as CircleCI. This is a CI/CD delivery platform and allows the build code to be automatically pulled and to create what is called a Pipeline.

This CircleCI pipeline allows us to fully automate and manage almost all the remaining steps in CI/CD. Tests can be created, deployed and then if the tests are successful the deployment would move to the next step.

Analyse

At this stage we can setup scans for vulnerabilities and any other such requirements we have before deploying new software.

Deploy

During this stage we can automatically deploy any required new infrastructure to services such as AWS. This includes servers, storage, networking, load balancing and others.

We are able to quickly create and deploy brand new AWS services specifically for this new build of our product.

Test

This is a key stage in CI/CD and once again we can fully automate it. We need to ensure that the new or updated product is acceptable to be delivered to the customer.

Promote

If all testing is successful, we can promote the new AWS services to be the new production service. We can also decide if we wish to decommission the previous production environment or leave it in-situ for an agreed period of time before decommissioning it.

Revert

Should any of the new production deployment tests fail, the automated CI/CD pipeline can simply destroy the new build deployment and any required AWS services created.

Review/Analyse

This is another key step where we could allow the failed deployment AWS environment to be retained for a short period of time to help with any troubleshooting of the failure and errors observed.

Why should we consider CI/CD ?

I have listed out each of the CI/CD stages and listed just some of the benefits which our organisation could see from adopting the usage of CI/CD:

Stage	Developer Benefits	Organisational Benefits
Code	Using a tool like VSCode as the standard development tool for our whole organisation.	Cost savings due to reduction in employee time spent in training and operations.
Build	Using a single system such as GitHub will streamline how we review and merge code changes.	Employee time saved and thus reduced costs to the organisation.
Test	They can use a shared standard set of tests which can be easily managed and updated.	By automating all testing, we can avoid any manual mistakes or failures. This would ensure we avoid deployment of any bad products to our customers.
Analyse	Time and effort saved by performing the vulnerability checks of new code.	Employee costs saved by the use of the new automated checks.
Deploy	Remove manual creation of the AWS services used to deliver the new product.	Further cost savings of employee time spent deploying AWS services.
Test	Use a set of standard production ready tests. This would be the last opportunity to catch any bugs before the new product is delivered.	Reputation with our customers could be impacted if there were any bugs missed because of manual testing.
Promote	Automated promotion to a production environment.	Our customers would be delivered their new/updated product quicker.
Revert	The developers can be given the new production environment to aid with identifying the reason the production tests failed.	If production testing failed this stage would ensure the faulty product is not delivered to our customer.
Review / Analyse	Once they have identified the root cause of the failure, they can delete the AWS services delivered for the new product. They can then go back to the code stage and make the required fixes to the code.	Our developers would be able to identify the reason for the failure and then plan to code the fix.