**CI/CD**

1. **Continuous Integration (CI)**

In a nutshell, CI is all about **automating the build steps**.

A picture containing text, screenshot, design

Description automatically generatedThe typical flow is to create a new branch for each feature/bug fix. The CI pipeline will be triggered as soon as the code is pushed into the branch. It’s vital to ensure the build passes as fast as possible as it will be triggered throughout the day. Traditionally, the build will perform the following steps automatically:

* Check out the code
* Build (i.e. docker build) if required. This step is about compiling the code or ensuring it can fetch all of the required dependencies it needs to run the application.
* Run static code analysis tools such as black or pylint for Python or DeepSource for C# for instance. The static code analysis will check for whitespace, naming conventions of methods, classes, namespaces, variables and so on.
* If the static code analysis succeeds then we’ll run the fast-running unit tests
* Once the unit tests stage passes then dependent on the build configuration, we can run integration or end-to-end tests.
* We can also create separate builds where slow-running unit tests and integration tests run once a day instead of running them at every check-in.
* Test reports can be uploaded to the reporting systems.
* Then the code will be merged into the master. The result of the build will be a package (code artefacts) that can be deployed.
* The code is now ready to be deployed.

A picture containing text, screenshot, font, yellow

Description automatically generated

1. **Continuous Deployment (CD)**

The key is to ensure the application is available to the end user whilst the code is being deployed.

The usual steps of the CD pipeline are:

* Take the code artefacts and the target environment
* Deploy the software without affecting the availability of the end user. This is possible with Kubernetes, ECS, ELK, NGINX and/or AWS lambdas where the new applications are started and the load balancer switches over to the newly deployed application. Subsequently, the user does not experience any downtime.
* We should perform end-to-end tests that do not change the state of the application/storage including running smoke and health status tests to ensure the application is healthy.