**DEVELOPING CI/CD INFRASTRUCTURE**

The diagram (Figure 1) below depicts the entire process flow in a highly simplified manner.

Continuous Monitoring

Figure : CI//CD High-level

The key success factor for a CI/CD pipeline is to ensure that the entire process from the time a developer commits their code onto the version control system, a process must pick the change and build the code, and perform tests, right to the deployment of change into production, is fully automated.

Given our current scenario, and to be hosting environment agnostic, the process will require the following two processes:-

1. Development of the docker script file which creates the MySQL database
2. Creation of the schema scripting file which would change from time to time. The base data will only be loaded once

The developer will have to commit the changes to a version control system.

Based on the two points above, whenever there is a change either in the docker script file or the schema file, the changes will be compiled and pushed into the ***Development Environment***. Automated unit tests and integration tests will be defined and executed at this stage. This is a critical stage and the system must be rigorously tested as the next stage is so dependent on this stage. Depending on the outcome, the change will either be accepted and moved to the ***UAT/QA Environment*** or it will be rejected and feedback is immediately sent back to the developer. A docker definition file for each environment will be provided which will ensure that the environments are similar.

The ***Production Environment***, will also follow suit and have its own docker file, that will back up the docker image before applying any changes.

The proposed tool to assist with this orchestration is [Jenkins](https://jenkins.io/), which has an easy to use pipeline scripting tool that allows for the definition of the CI/CD process. This allows the flexibility and scalability of the entire life cycle and development of the three environments as discussed above. Jenkins will also handle the configuration, package deployments and migrations.

The proposed implementation uses declarative scripting, to improve performance and scalability, scripted pipeline must be used, which really is the future of CI/CD.

The one aspect that is usually neglected is security. For any CI/CD pipeline, security checks must be performed at every stage. There are now security pipeline tools which can be used to test every deployment in the pipeline and ensure that all checks are passed. The other important aspect is to ensure that the third party packages used are stable and that patches are applied systematically to ensure systems are running on the vendor recommended releases.