**Scalable Continuous integration and Continuous deployment using DevOps**

**Phase I**

**Reference:** ScalableCI-CD\_PlanningDoc.docx

1. **Objective:** This document describes what and how to achieve the phase I of the solution.

Goal of this phase is to:

1. ✓ Understand various tools which will be part of the initial product.
2. ✓ Setup of Git server, Jenkins server (using docker),
3. Understand the roles of the management tools.
4. ✓Basic developer environment setup.
5. Basic pipeline setup. Which includes:

Git

MS-scripts

Jenkins

VS codebase setup

TestRunner (with Nunit tests)

DB setup (Optional).

1. Jenkins scripts are used as “Infrastructure as code” to trigger the build, run tests on different machines and push the passed builds to shared location.
2. **Way of working:**
3. There will be two teams. (you decide that).
4. Two teams works simultaneously on assigned tasks.
5. They share with complete technical details at end of every week (or the specific date you guys decided on).
6. **Teams and work:**
   1. **Both the teams:Below tasks to be performed using batch script (4 hours). We need to as most the first level invocation happens with batch scripts as which does not requires any security restrictions and can be edited from any machine as it is in script format. (DLL’s cannot be debugged so easily but a script can be).**

* How to invoke exe or another batch.
* Process input arguments. Ex: when TestArgs.Bat –Source-Directory c:\temp1 –Target-Directory D:\temp1 is provided, then script shall check both arguments are provided and throws error in case any extra argument provided or one of the input argument is missing.
* Check execution status of the command being executed. (using ERRORLEVEL argument).
* Return the exit value with proper error code (could be any integer value).
  1. **Both the teams: Below tasks to be performed in powershell. (version 5 is good enough). Powershell script is used at various levels including communicating with cloud, Harbor and many other tools. Which is rich with .net libraries and almost anything can be done with powershellcommandlets.**
* String operations: Concat, string extraction
* Read-write into file, connect to DB and run some queries.
* Functions
  1. **Both the teams: C# will be the main programming language for the product development in our case. We are going to develop mock product with various tests using C#.**
* Basics of C#, OOPS, String operation.
* NUNIT test case development using C#.
* Connect to DB and run query using C#.
  1. **Both the teams: Basic MSBuild script. This is required to build the application.**
* Simple hello word MSbuild script
* Property group and item group.
* How to import other MSbuild scripts into a MSbuild script.
* Invoke EXE using MSBuild.
  1. **Team 1:**
* **How to perform source code repository using GIT. Which includes creating a repository at GitHub.**
* **Basic git commands. Create repo, Clone repo, Pull, commit, push etc.**

1. **Basic pipeline development: TBD:**

**Test Runner requirements:**

1. Use C# , exe file.
2. Shall contain unit tests.
3. Should be able to execute “Nunit” test assemblies, execute the same and produce HTML results.
4. Results are consolidated in nature, shall have:
5. Summary: Total executed, passed, failed, no run
6. Details of each test being executed:

Test name, Status (pass/fail), Time consumption, Nodename, result file names

**Input/output:**

Input: Location where test assemblies are available. Tool shall recursively search for Nunit assemblies.

Outout: Html page with described information being available.

Tool returns number of tests processed, in case of all assemblies are addressed. Else negative number in case execution error happened.