

**B8IT122 Team 2**

***Continuous Integration***

***Cloud Infrastructure and Virilisation***

Document Version No.v1.1

Presented by:

**Daniel Tittello,**

**Richard Burke,**

**Neil Hogarty**

Table of Contents

[Technical Document 3](#_Toc41135723)

[Overview 3](#_Toc41135724)

[Purpose 4](#_Toc41135725)

[Scope 5](#_Toc41135726)

[In Scope: 5](#_Toc41135727)

[Out of Scope: 5](#_Toc41135728)

[Environment Information 6](#_Toc41135729)

[Architecture 8](#_Toc41135730)

[Access Procedure 9](#_Toc41135731)

[Lesson’s Learned 10](#_Toc41135732)

[Conclusion 11](#_Toc41135733)

[Bibliography / References: 12](#_Toc41135734)

# Technical Document

## Overview

The purpose of this project is to test the extent to which the team have achieved a specific learning objective of the course. The team have been commissioned to design, build and deploy a continuous integration system, using a cloud service provider, taking into account all the learning objectives and minimum requirements for the project.

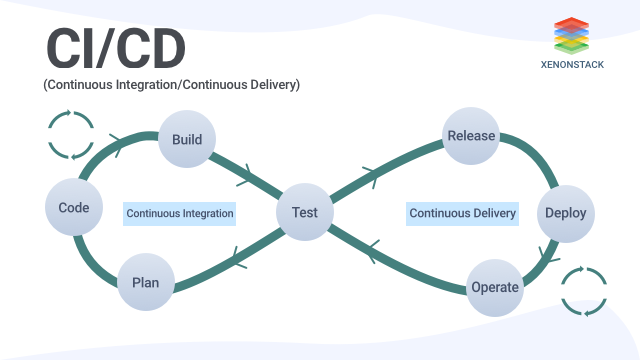
With increasing competition within the software development market, organisations pay special attention to and will allocated resources (both human and technical) to develop high-quality software at a faster pace. Continuous integration (a subset of continuous practices), offers organisations a more accelerated path to release a product into the market that may not be fully optimised but with the goal of continually deploying new versions getting closer to the end goal with each version and also embodying the agile DevOps model, i.e. continuous deployments influenced by customer needs, feedback, changing landscapes and disruptive technologies.

The product will be hosted on a web server, and will input data into a MySQL database from the use of a HTML form through and Apache server. The final project will create a pipeline for continuous integration (CI) utilising a Jenkins automation server and a Git Server to hold the code repository.

## Purpose

The primary purposes of this document are;

* To define scope objectives for all participants
* To ensure that all objectives outlined in the scope are achieved and that all parties have a high-level understanding of the overall objective.
* To ensure all parties have a high-level understanding of the technical processes within the project.
* To gain a more wholistic and practical understanding of the Continuous Integration process and how it can be applied to larger organisations.
* To critically analyse any decisions made in the planning phase (technical or strategic) and to reflect on any decision that could be made different in the future, depending on what was learned throughout the project life cycle.



***Fig 1, a sample continuous integrations / development loop.***

## Scope

The scope of the project is defined as the technical design and operation of a working continuous integration pipeline. This will incorporate the use of 4 severs (mentioned below in the “In Scope Critical Success Factors”.

## In Scope:

|  |  |
| --- | --- |
| Objective | Critical Success Factor |
| * Continuous Integration | All necessary components of the CI pipeline have been met and function as expected |
| * Web Server (Apache / JBoss) | Allow users to access to Apache Server |
| * MySQL Server | Allow users to access to MySQL Server |
| * Git Server | Allow users to access to Git Server |
| * Jenkins Server | Allow users to access to Jenkins Server |

## Out of Scope:

* Disaster Recovery
* Auto Scaling
* Load Balancing
* Performance Metrics
* Capacity
* Availability

## Environment Information

**Cloud Provider:**

The project was completed using Microsoft Azure Cloud Computing Services, a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through Microsoft-managed data centres. This cloud service provider was chosen;

* + As it was a common platform that all three developers had full access.
  + As it is one of the largest commercial cloud providers.
  + As all resources are familiar with the platform and technology, thus ensuring that the primary focus was on value adding activities (development and deployment) and would reduce the time taken on learning any new systems.

The development and deployment are both using **MS East US** region. The rationale behind this was to look at the most cost optimised option within MS Azure that would still have full functionality and services, i.e. full suite of MS resources available in this region.

|  |  |  |
| --- | --- | --- |
| Region | VM Ubuntu LTS 18.04 (cost per month) | Disc size |
| US East | € 6.40 | Standard SSD |
| (Europe) North | € 6.96 | Standard SSD |
| (Europe) UK West | € 7.25 | Standard SSD |

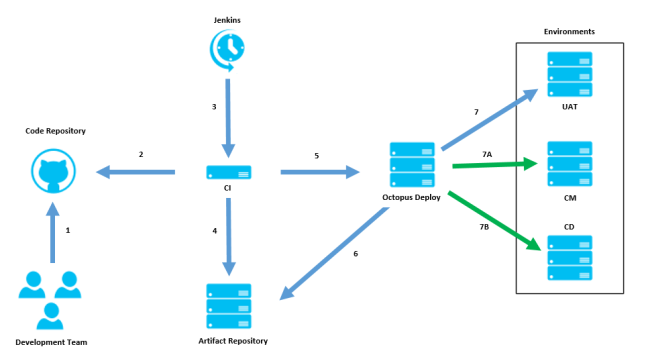
**Virtual Machines:**

As the main scope of the project was to develop a continuous integration, the focus was on developing a working application that will show the continuous integration “pipeline”. It is this justification that the virtual machines created were created with minimum specifications. It is this rationale that;

* The smallest size VM’s were chosen – Ubuntu Server 18.04 LTS (long term support) B1s / B1ls
* All VM’s created were single instances and not redundant, i.e. they were only in one Cloud / Data center. In a real-life scenario this wouldn’t be ideal, however as this project is a POC (proof of concept), the decision was taken to focus on the primary objective of continuous integration and not on cloud architecture principles (i.e Scalability etc)
* Note: for encapsulate configuration details and host more than one domain from a single server.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| VM Name | VM type | Size | Function | Software | Comment |
| B8IT122-CA | Ubtuntu 18.04 LTS B1s | 1GB ram | To develop and Host the web application | LAMP Stack  PhpAdmin | A full Lamp stack was installed on this machine, using Apache web server, MySQL and phpadmin to create the HTML application and insert data into MySql. |
| gitserverca | Ubtuntu 18.04 LTS B1s | 1GB ram | To configure and host the GIT server | GIT Server | A git server was installed on this machine. |
| Jenkins | Ubtuntu 18.04 LTS B1s | 1GB ram | To configure and host the Jenkins server | Jenkins Server | A Jenkins server was installed on this machine. |

## Architecture



## Access Procedure

Test Pages:

|  |  |
| --- | --- |
| Functionality form | <http://40.71.87.225/add-record-form.php> |
| Continuous integration form | <http://40.71.87.225/test-continuous-integration.php> |

## Lesson’s Learned

**Naming convention:**

To adhere to best practice, from a development and cloud architecture point of view, if this project was to be repeated, typical naming conventions would be adhered to. These would include having a tag on each VM to ensure best practice and also to have all VMs consistently named. (i.e VM0001-Apache / VM0002-GitServer). In a real-life scenario this would be adhered to as systems would be more complex and could include legacy systems and multiple VMs on the same estate, however as this was a POC, the focus was on functionality.

**Containerization:**

For this project, the decision was taken to use a series of Virtual Machines and not use a container service like Docker. If this project was to be repeated, or the approach was to be repeated, it would be more efficient (both economically and environmentally) to use containers. Containers are more lightweight and would share the operating systems resources of the VM it sits on. Containers take milliseconds to boot up. In larger organizations and environments, containerization is ideal for running multiple instances of the same operating system.

**Git Server:**

For this project,the decision was taken to install a git server onto a VM. This server hosts a git master that initially holds an empty folder. To access this repository, we downloaded git in our local machines (ex: Windows, mac, Linux, etc) which gives us opportunity to clone the master repository, add our changes and push it back to git.

Git is an open-source tool that developers use to manage source code and GitHub is an online service that connects to Git to manage resources with a more friendly UI. In this project, we are using Git and GitHub to manage our code changes and import it to Jenkins.

**Security:**

There were some security principles that could be improved should this project be repeated or continued. Firstly, all VM’s were set up on MS Azure secured with a password. This decision was taken for convenience of use and to have the ability to ssh into each VM from any location. Should this be repeated, our recommendation would be to introduce a private key (.pem file) with read only access to increase security of the VM.

## Conclusion

From review of the scope and critical success factor, this project has been marked as successful. All servers were accessible and configurable.

The Continuous Integration pipeline has been functional as stated previous in the report. Changes that were made on the development server were inserted into git and worked through the “pipeline” through GIT and Jenkins to deploy to the production server.

Throughout careful analysis and eliminating non value adding tasks (e.g. using an Apache Tomcat server when Java wasn’t a requirement) it gave the developers more time to focus on value adding tasks and enabled more efficient cross collaboration.

|  |  |  |
| --- | --- | --- |
| Objective | Critical Success Factor | Images |
| * Continuous Integration | All necessary components of the CI pipeline have been met and function as expected |  |
| * Web Server (Apache / JBoss) | Allow users to access to Apache Server | Apache:  PhpAdmin: |
| * MySQL Server | Allow users to access to MySQL Server | MySql tables |
| * Git Server | Allow users to access to Git Server |  |
| * Jenkins Server | Allow users to access to Jenkins Server |  |

Should this project be repeated, there are a few design decision that could be improved upon (e.g. using containerisation as opposed to VM’s) however the fundamental functionality (Pipeline activity) was deemed to be successful so the proof of concept has been proven.

## 

## Bibliography / References:

* Pahl, Claus & Jamshidi, Pooyan & Zimmermann, Olaf. (2017). Architectural Principles for Cloud Software. ACM Transactions on Internet Technology. 18.
* Bauer, R. and Bauer, R., 2020. *Docker Containers Vs. Vms: Pros And Cons Of Containers And Virtual Machines*. [online] Backblaze Blog | Cloud Storage & Cloud Backup. Available at: <https://www.backblaze.com/blog/vm-vs-containers/> [Accessed 23 May 2020].
* XenonStack. 2020. *A Quick Guide To Continuous Integration And Continuous Delivery*. [online] Available at: <https://www.xenonstack.com/blog/continuous-integration-and-continuous-delivery/> [Accessed 23 May 2020].