| MSc in Web Technologies – Part-time Year 1- MSCWEBTECH 1 National College of Ireland (NCI) |
| --- |
| Deployment Project |
| Submitted by: Jonathan Harris (Student Id: x13118901) |

|  |
| --- |
| Jonathan Harris  January 22, 2014 |

Contents

[Introduction 3](#_Toc378603710)

[Approach 3](#_Toc378603711)

[Delivery Process 6](#_Toc378603712)

[Specification of the Delivery process 7](#_Toc378603713)

[Requirements Analysis 7](#_Toc378603714)

[User Requirements 7](#_Toc378603715)

[Function requirements 7](#_Toc378603716)

[Use Cases 8](#_Toc378603717)

[Risks and Controls 10](#_Toc378603718)

[Design of Delivery process 10](#_Toc378603719)

[Detailed design 10](#_Toc378603720)

[Process Flow 10](#_Toc378603721)

[Architecture 10](#_Toc378603722)

[Build/ Integration/Test process 10](#_Toc378603723)

[Sub Functions 11](#_Toc378603724)

[Deployment process 13](#_Toc378603725)

[Deployment sub functions 13](#_Toc378603726)

[Monitoring process 15](#_Toc378603727)

[Crontab command: 15](#_Toc378603728)

[Contents of crontab\_mon.arg parameter file: 15](#_Toc378603729)

[crontab\_contents: 15](#_Toc378603730)

[Email Notification (using sendmail.pl): 15](#_Toc378603731)

[Mailer Utility (sendmail.sh) 15](#_Toc378603732)

[Gmail 16](#_Toc378603733)

[Monitoring script (SystemMonitor.sh) 16](#_Toc378603734)

[Level 0 function (SystemStatusFuncLevel\_0.sh) 18](#_Toc378603735)

[Level 1 function (SystemStatusFuncLevel\_1.sh) 20](#_Toc378603736)

[Test Plan 21](#_Toc378603737)

[Unit tests (UnitTesting.sh) 21](#_Toc378603738)

[Integration testing (UnitTesting.out) 22](#_Toc378603739)

[Execution plan 23](#_Toc378603740)

[Record of demonstration 23](#_Toc378603741)

# Introduction

Deployment in large hosting services companies is now treated as part of a development process that supports the Operations of the company and its customers (DevOps). Where before deployment required building of physical servers and configuration of routers, these days, deployment is much about controlling the process programmatically.

In this document I will describe the requirements gathering, design and implementation of the automated delivery process used to manage the build, integration, test, deployment and monitoring of a web site application to an Oracle VM VirtualBox. The content of the website will be a very simplistic site with mix of static content hosted via an Apache webserver with content dynamically generated through a MySQL database. The content and functionality of the web-site will not be described in detail as the purpose of this document is demonstrate the deployment of an generic web application not to test the functionality of the web application itself.

## Approach

My approach will be to apply the Information Technology Infrastructure Library (ITIL®) framework throughout the project as it specifically applies to the deployment process being demonstrated.

ITIL® is a set of practices for IT service management (ITSM) that focuses on aligning IT services with the needs of a business. ITIL® describes an exhaustive set of processes, procedures, tasks and checklists that are not organisation-specific but can be integrated within any organisation's existing IT strategy. It is used to establish a baseline from which the organisation can plan, implement and measure improvements thus delivering value to the business through its IT services. Ultimately by maintain a minimum level of IT competency across the whole organisation ITIL® demonstrates compliance with industry best practices.

A discussion of the entire ITIL® is outside the scope of this project so I will concentrate on the specific (ITIL®)& ITSM practices (shown in Fig. 1) and highlight how they apply to the deployment process.

ITIL

Service

Design

Service

Transition

Service

Operation

ITIL

ITIL

ITIL

Service

Level

Management

ITIL

ITIL

ITIL

Release

and

Deployment

ITIL

ITIL

ITIL

Problem

Management

Figure 1. ITIL® framework as applied to the topic of deployment.

In order to discuss the topic of deployment in the context of best practice for the management of IT infrastructure I will concentrate on the following subject matter areas as described by ITIL:

###### Service Design

Service Design provides good-practice guidance on the design of IT services, processes, and other aspects of the service management effort. Service design addresses how a planned service solution interacts with the larger business and technical environments, service management systems required to support the service, processes which interact with the service, technology, and architecture required to support the service, and the supply chain required to support the planned service. The areas of service design are addresses as follows:

**Service Level Management** -Ensuring that the agreed IT services are delivered when and where they are supposed to be, this is addressed at a high level by the delivery process specification which defines the service will be delivered on a PC with Oracle VM VirtualBox on the project demonstration date.

**Availability Management** - Addresses the ability of an IT component to perform at an agreed level over a period of time. Availability gives a clear overview of the end-to-end availability of the system which can be further detailed under the following headings:

Reliability: This is the ability of an IT component to perform at an agreed level at described conditions. The proscribed level of reliability for the deployment project will within the normal PC/Laptop operating reliability conditions only.

Maintainability: This is the ability of an IT component to remain in, or be restored to an operational state. The maintainability of the deployment project is assured by the provision of both the build and deployment scripts to restore the deployment component at any time and along with the monitoring script which will verify the components will remain in an operational state for the duration of the project demonstration.

Serviceability: The ability for an external supplier to maintain the availability of component or function under a third-party contract. There will be no provision for external supplier or third-party contract for the deployment project.

Resilience: This measure of freedom from operational failure and a method of keeping services reliable. One popular method of resilience is redundancy. This is addressed in the build and unit testing process through both error reporting and redundancy provided by the Git software repository.

Security: Where a service may have associated data, security refers to the confidentiality, integrity, and availability of that data. For the deployment project security will be provided through the PC/GitHub and Oracle VM VirtualBox logging and authorisation process.

**Capacity Management -** Capacity management supports the optimum and cost-effective provision of IT services by helping organisations match their IT resources to business demands. Capacity management is focused on strategic capacity, including capacity of personnel (e.g., human resources, staffing and training), system capacity, and component (or tactical) capacity. There will be no provision for this area in the deployment project as there is no identifiable costs associated with the project.

**Service Transition**

Service transition relates to the delivery of services required by a business into live/operational use, and often encompasses the "project" side of IT rather than business as usual (BAU). This area also covers topics such as managing changes to the BAU environment.

List of ITIL processes in service transition:

Transition planning and support – N/A to this project

Change management – N/A to this project

Service asset and configuration management – – N/A to this project

Release and deployment management – Addresses in the next section

Service validation and testing – N/A to this project

Change evaluation – N/A to this project

Knowledge management – N/A to this project

**Release and Deployment Management**

Release and deployment management is used by the software migration team for platform-independent and automated distribution of software and hardware, including license controls across the entire IT infrastructure.

Proper software and hardware control ensures the availability of licensed, tested, and version-certified software and hardware, which functions as intended when introduced into existing infrastructure.

Quality control during the development and implementation of new hardware and software is also the responsibility of Release Management. This guarantees that all software meets the demands of the business processes.

The goals of release management include:

* Planning the rollout of software – Addressed in the Design of Delivery Process.
* Designing and implementing procedures for the distribution and installation of changes to IT systems – Addressed in the Design of Delivery Process.
* Effectively communicating and managing expectations of the customer during the planning and rollout of new releases – Partially address in End User Testing
* Controlling the distribution and installation of changes to IT systems - Partially covered by the Execution plan.

Release management focuses on the protection of the live environment and its services through the use of formal procedures and checks. The deployment project Integration testing, System testing, and End user testing documentation will provide the formal procedures and checks for Release and Deployment in line with these high level ITIL process categories.

## Delivery Process

The delivery process for the deployment project will be to us the V-Model System Development Life Cycle (SLDC) which describes a development methodology which may be considered an extension of the waterfall model. Instead of moving down in a linear way as in the waterfall model, the V-Model process steps are bent upwards after the coding phase, to form the typical V shape. The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.

Starting with Requirements Analysis on the left hand side falling through the subsequent verification phases to the coding phase. The model then steps upward through a series of validation phases through to User Acceptance Testing. Each step of the model is an iterative process as more detail is added then the subsequent step may need to be revisited as the output of each step of the verification process on the way down the V-model act as input to the corresponding phases on the way up the validation arm of the V-Model as outlined in Figure 2.

The deployment project delivery process will be to apply the V-Model System Development Life Cycle (SLDC) as outlined below.

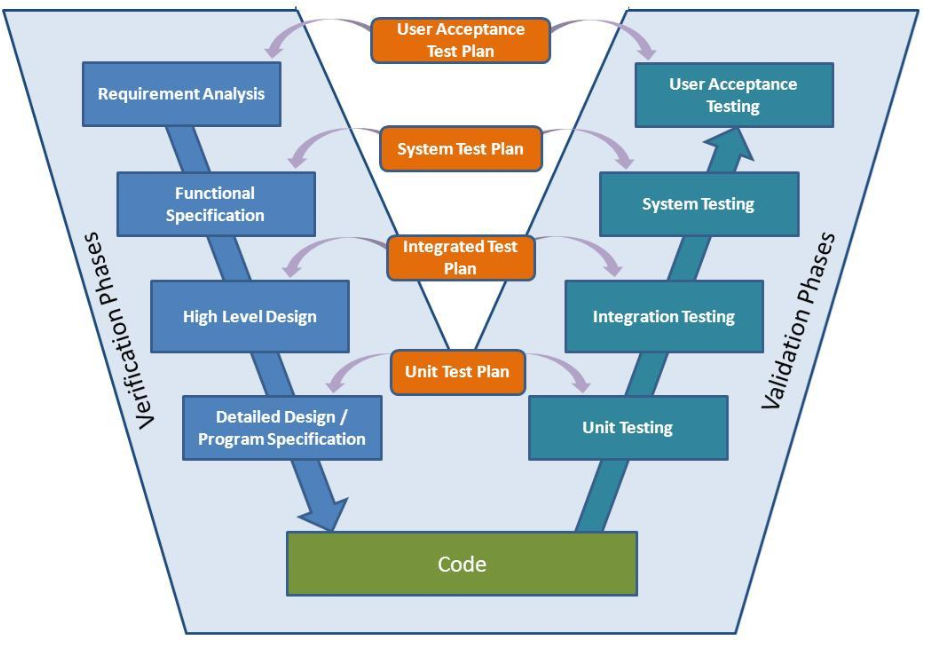


Figure 2. V-Model System Development Life Cycle (SLDC).

# Specification of the Delivery process

## Requirements Analysis

The specific user requirements for this project are taken from the “Deployment module of MSc in Web Development” document as provided by the course instructor.

### User Requirements

I will outline the high level project requirements using the MoSoCoW definitions in order to determine what needs to be done (requirements) and therefore evolve a design plan. MoSoCoW is an abbreviation of **M**ust haves, **S**hould haves **C**ould have and **W**ouldn’t have as described below:

* **M** - MUST: Describes a requirement that must be satisfied in the final solution for the solution to be considered a success.
* **S** - SHOULD: Represents a high-priority item that should be included in the solution if it is possible. This is often a critical requirement but one which can be satisfied in other ways if strictly necessary.
* **C** - COULD: Describes a requirement which is considered desirable but not necessary. This will be included if time and resources permit.
* **W** - WON’T: Represents a requirement that stakeholders have agreed will not be implemented in a given release, but may be considered for the future.

The specific MoSoCoW’s as apply to this deployment project are as follows:

|  |  |  |
| --- | --- | --- |
| No. | MoSoCoW | Requirements |
| 1 | Must Have | It must be possible to build the deployment components with a single command onto a defined environment. |
| 2 | Must Have | It must be possible to deploy a website and its content and components with a single command onto a defined webserver environment. |
| 3 | Should Have | Error logging should be available in one place. |
| 4 | Should Have | A mailer system to alert users and system admins of potential system problems or events. |
| 5 | Could Have | Additional security features other than those provided by the native VM logging access and authorisation services |
| 6 | Wouldnt Have | Deployed across multiple servers |

## Function requirements

Taking the above MoSoCoW table we can use the Use Cases model to describe how an operator of the Deployment process can achieve these requirements and thus define the functional requirements as follows:

## Use Cases

##### Use Case for Deployment Process debug trigger:

|  |  |
| --- | --- |
| Use Id | UC1 |
| Use Case Title | Triggering the build process to start |
| MoSoCoW Ref: | MSCW1, MSCW 3, MSCW 4, MSCW 5 |
| Precondition | The user’s PC must have a Unix terminal window open on a Oracle VM VirtualBox |
| Actor | User |
| Scenario | 1. User logs into terminal window 2. From the command prompt user runs bash script to start debug process 3. Bash script completes the build process with no further user input and prompts the user when the build process has completed |
| Postcondition | If the bash script encounters errors during build process then it prompt the user and provides further information via a log file location |

##### Use Case for Deployment Process Trigger:

|  |  |
| --- | --- |
| Use Id | UC2 |
| Use Case Title | Triggering the deployment process to start |
| MoSoCoW Ref: | MSCW2, MSCW 3, MSCW 4, MSCW 5 |
| Precondition | The users PC must have a Unix terminal window open on a Oracle VM VirtualBox |
| Actor | User |
| Scenario | 1. User logs into terminal window 2. From the command prompt user runs bash script to start deployment. 3. Bash script completes deployment with no further user input and prompts the user when the deployment has completed |
| Postcondition | If the bash script encounters errors during deployment then it prompts the user and provides further information via a log file location. |

##### Use Case for Monitor Trigger:

|  |  |
| --- | --- |
| Use Id | UC3 |
| Use Case Title | Triggering the monitoring process to start |
| MoSoCoW Ref: | MSCW 3, MSCW 4, MSCW 5 |
| Precondition | The user’s PC must have a Unix terminal window open on a Oracle VM VirtualBox |
| Actor | User |
| Scenario | 1. User logs into terminal window 2. From the command prompt user runs bash script to start monitor. 3. Bash script executes a script which establishes a crontab job which checks the deployment environment every minute with no further user input from the user and prompts the user when the crontab job started executing. |
| Postcondition | If the bash script encounters errors then it prompts the user and provides further information via a log file location. |

##### Use case of generate report:

|  |  |
| --- | --- |
| Use Id | UC4 |
| Use Case Title | Triggering the generate report process to start |
| MoSoCoW Ref: | MSCW 3, MSCW 4, MSCW 5 |
| Precondition | The user’s PC must have a Unix terminal window open on a Oracle VM VirtualBox |
| Actor | User |
| Scenario | 1. User logs into terminal window 2. From the command prompt user runs bash script to start the report generation process 3. Bash script combines the logfiles together and displays the report on screen |
| Postcondition | If the bash script encounters errors during report generation then it prompt the user and provides further information via a log file |

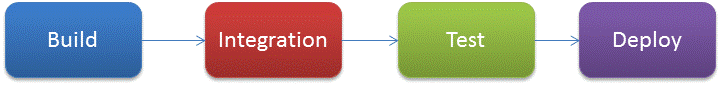
## Risks and Controls

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Impact | Risk Statement | Control |
| 1 | High | Corrupted content could replace valid working content on the live platform | Content is backed up prior to deployment of new content. If the deployment fails, the old site is kept in place. |
| 2 | Medium | Incorrect Apache or mySQL installation could cause the site to not function | Apache & MySQL are tested during Integration and the re-installed during the deployment process |
| 3 | Low | Monitor logfile running for extended period could cause the system to run out of disk space | Log files backed up to a zip file ever hour and deleted on a rolling daily/weekly basis |

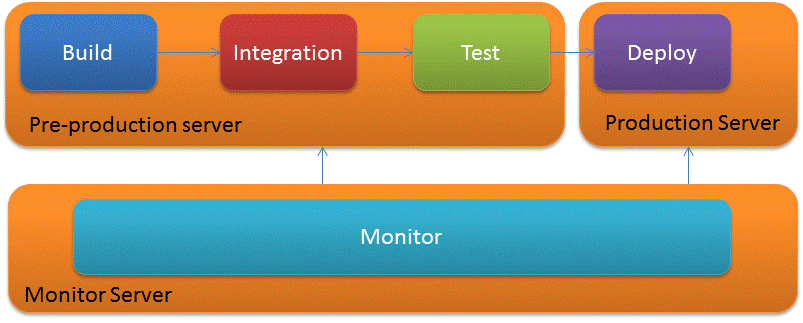
# Design of Delivery process

## Detailed design

### Process Flow



### Architecture



## Build/ Integration/Test process

The build process will download content from a repository (i.e GitHub). It checks that all components and resources are in place for testing. It integrates the deployment and monitoring scripts from 2 or more files, into one. The build process makes sure that the environment is clean and revisions of necessary components are at the right level.

### Sub Functions

Create local sandbox

# Create Sandbox

cd /tmp

SANDBOX=sandbox\_$RANDOM

mkdir $SANDBOX

Download from Github

cd $SANDBOX/

git clone https://github.com/JonnyHarris/NCIRL.git

cd NCIRL/

Unpack package

tar -zcvf webpackage\_preBuild.tgz webpackage

Perform tests

MD5SUM=$(md5sum webpackage\_preBuild.tgz | cut -f 1 -d' ')

PREVMD5SUM=$(cat /tmp/md5sum)

FILECHANGE=0

if [[ "$MD5SUM" != "$PREVMD5SUM" ]]

then

FILECHANGE=1

echo $MD5SUM not equal to $PREVMD5SUM

else

FILECHANGE=0

echo $MD5SUM equal to $PREVMD5SUM

fi

echo $MD5SUM > /tmp/md5sum

if [ $FILECHANGE -eq 0 ]

then

echo no change in files, doing nothing and exiting

exit

fi

Move files

cp Apache/www/\* /var/www/

cp Apache/cgi-bin/\* /usr/lib/cgi-bin/

chmod a+x /usr/lib/cgi-bin/\*

Check Dependencies

# Perfrom Test on Web Page contents using md5sum to

# compute and check MD5 message digest

MD5SUM=$(md5sum webpackage\_preBuild.tgz | cut -f 1 -d' ')

# save previous md5sum value

PREVMD5SUM=$(cat /tmp/md5sum)

# Test if the value has changes

# i.e. the Web contents has changed

FILECHANGE=0

if [[ "$MD5SUM" != "$PREVMD5SUM" ]]

then

FILECHANGE=1

echo $MD5SUM not equal to $PREVMD5SUM

else

FILECHANGE=0

echo $MD5SUM equal to $PREVMD5SUM

fi

# Save the new md5sum to disk

echo $MD5SUM > /tmp/md5sum

# Only coninue with BUILD if the files have changed

if [ $FILECHANGE -eq 0 ]

then

echo no change in files, doing nothing and exiting

exit

fi

Create zip/tar file (package)

# BUILD Process

# Move build package

mv webpackage\_preBuild.tgz build

rm -rf webpackage

cd build

tar -zxvf webpackage\_preBuild.tgz

#

Move package to next step

tar -zcvf webpackage\_preIntegrate.tgz webpackage

ERRORCHECK=0

# INTEGRATE

mv webpackage\_preIntegrate.tgz ../integrate

rm -rf webpackage

cd ../integrate

#

tar -zxvf webpackage\_preIntegrate.tgz

###

tar -zcvf webpackage\_preTest.tgz webpackage

Clean-up

ERRORCHECK=0

# TEST

mv webpackage\_preTest.tgz ../test

rm -rf webpackage

cd ../test

#

tar -zxvf webpackage\_preTest.tgz

Report/Log into BuildInt.out

Cloning into 'Project'...

webpackage/

webpackage/script1.plx

webpackage/script2.plx

webpackage/form.htm

webpackage/index.htm

4c35c39059c70384f192b9b283a936f4 not equal to a95687e6c9c74b40ae06d53d41f7f492

webpackage/

webpackage/script1.plx

webpackage/script2.plx

webpackage/form.htm

webpackage/index.htm

webpackage/

webpackage/script1.plx

webpackage/script2.plx

webpackage/form.htm

webpackage/index.htm

webpackage/

webpackage/script1.plx

webpackage/script2.plx

webpackage/form.htm

webpackage/index.htm

webpackage/

webpackage/script1.plx

webpackage/script2.plx

webpackage/form.htm

webpackage/index.htm

webpackage/

webpackage/script1.plx

webpackage/script2.plx

webpackage/form.htm

webpackage/index.htm

webpackage/

webpackage/script1.plx

webpackage/script2.plx

webpackage/form.htm

webpackage/index.htm

webpackage/

webpackage/script1.plx

webpackage/script2.plx

webpackage/form.htm

webpackage/index.htm

## Deployment process

Deployment ensures that all components (content, packages etc) and resources (memory, disk, I/O etc) are in place for production. It unpacks the content and move it to its proper location on the production server.

It backs up the content prior to deployment of new content. If the deployment fails, the old site is kept in place.

### Deployment sub functions

Create local sandbox

# Create Sandbox

cd /tmp

SANDBOX=sandbox\_$RANDOM

mkdir $SANDBOX

cd $SANDBOX

Unpack package

# Initalise the error counter used in report

ERRORCHECK=0

# Make empty test pages in webpackage dir

mkdir webpackage

touch webpackage/index.htm

touch webpackage/form.htm

touch webpackage/script1.plx

touch webpackage/script2.plx

#

# Make the Architecture process directories

mkdir build

mkdir integrate

mkdir test

mkdir deploy

#

# copy webpackage from git and move webpackage

#

git clone https://github.com/JonnyHarris/Project.git

Checksum function to check for changes in files etc./Check Dependencies/Perform tests

Perfrom Test on Web Page contents using md5sum to

# compute and check MD5 message digest

MD5SUM=$(md5sum webpackage\_preBuild.tgz | cut -f 1 -d' ')

# save previous md5sum value

PREVMD5SUM=$(cat /tmp/md5sum)

# Test if the value has changes

# i.e. the Web contents has changed

FILECHANGE=0

if [[ "$MD5SUM" != "$PREVMD5SUM" ]]

then

FILECHANGE=1

echo $MD5SUM not equal to $PREVMD5SUM

else

FILECHANGE=0

echo $MD5SUM equal to $PREVMD5SUM

fi

# Save the new md5sum to disk

echo $MD5SUM > /tmp/md5sum

# Only coninue with BUILD if the files have changed

if [ $FILECHANGE -eq 0 ]

then

echo no change in files, doing nothing and exiting

exit

fi

Backup current files/Move files

# Move build package

mv webpackage\_preBuild.tgz build

rm -rf webpackage

cd build

tar -zxvf webpackage\_preBuild.tgz

#

tar -zcvf webpackage\_preIntegrate.tgz webpackage

ERRORCHECK=0

# INTEGRATE

mv webpackage\_preIntegrate.tgz ../integrate

rm -rf webpackage

cd ../integrate

#

tar -zxvf webpackage\_preIntegrate.tgz

###

tar -zcvf webpackage\_preTest.tgz webpackage

ERRORCHECK=0

# TEST

mv webpackage\_preTest.tgz ../test

rm -rf webpackage

cd ../test

#

tar -zxvf webpackage\_preTest.tgz

###

tar -zcvf webpackage\_preDeploy.tgz webpackage

ERRORCHECK=0

# DEPLOY

if [ $ERRORCHECK -eq 0 ]

then

mv webpackage\_preDeploy.tgz ../deploy

rm -rf webpackage

cd ../deploy

tar -zxvf webpackage\_preDeploy.tgz

fi

## Monitoring process

Monitoring ensures that the site is functioning from a HTML, HTTP and other socket layer perspective. It makes sure that the key parameters (e.g. memory, I/O etc. ) are within thresholds. It report errors.

The log and monitor utility performs a sanity check of the web app environment, and when executed in a crontab it can be made to run periodically and if there is a problem then it produces both a logfile and also sends an email to the system administrator. Its primary function is to ensure that network and server parameters are in bound and that functions are available as follows.

### Crontab command:

testuser@ubuntu:~$ crontab crontab\_mon.arg

### Contents of crontab\_mon.arg parameter file:

\*/1 \* \* \* \* bash SystemMonitor.sh >> SystemMonitor.out

### crontab\_contents:

testuser@ubuntu:~$ crontab -l

\*/1 \* \* \* \* bash SystemMonitor.sh >> SystemMonitor.out

### Email Notification (using sendmail.pl):

#### Resulting SystemMonitor.out contents:

2014-01-22 16:01:25 status

Apache process is Running

Apache is Listening

Remote Apache TCP port is up

Apache within limits

Mysql process is Running

Mysql is Listening

Remote Mysql TCP port is down

Remote IP is alive

There was a problem on the server email sent to System Admin

### Mailer Utility (sendmail.sh)

#Perl mailer utility

#!/usr/bin/perl

my $msg=shift @ARGV;

use strict;

use warnings;

use Email::Send;

use Email::Send::Gmail;

use Email::Simple::Creator;

my $email = Email::Simple->create(

header => [

From => 'jrhdeploy@gmail.com',

To => 'jrhdeploy@gmail.com',

Subject => 'Problems with the site',

],

body => $msg,

);

my $sender = Email::Send->new(

{ mailer => 'Gmail',

mailer\_args => [

username => 'jrhdeploy@gmail.com',

password => 'jrhdeploy353',

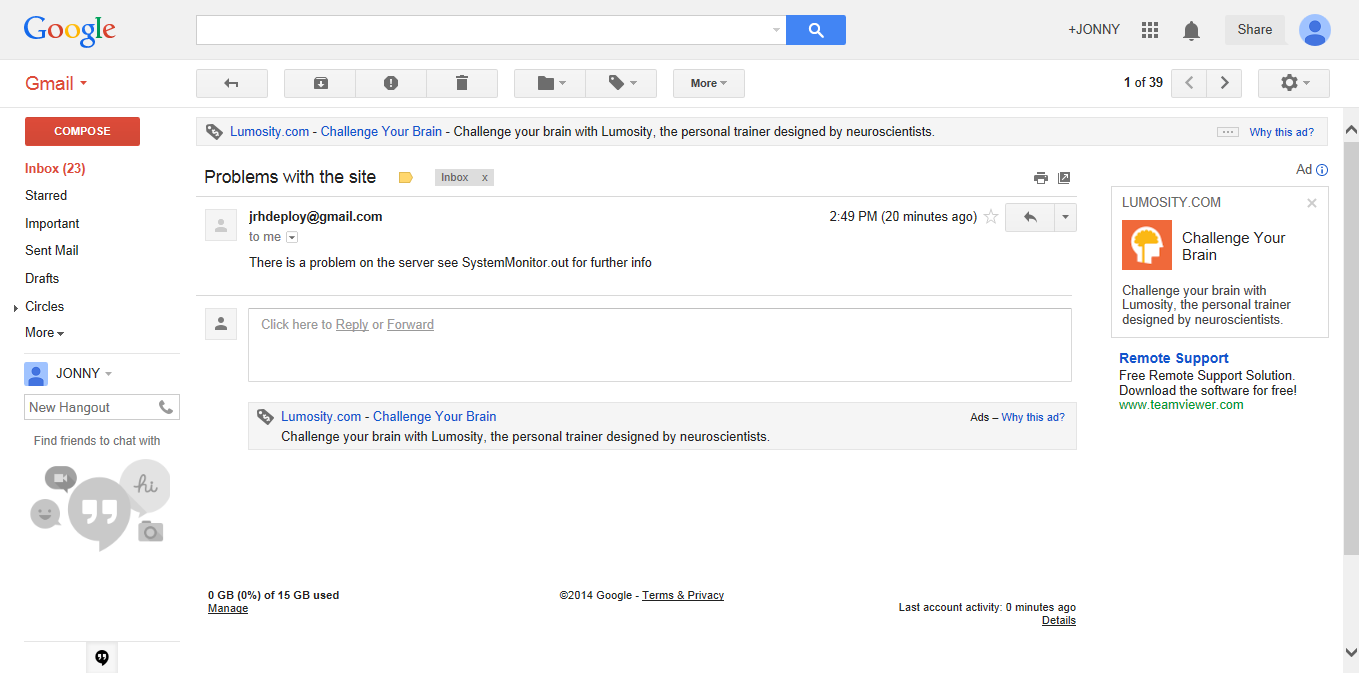
]

}

);

eval { $sender->send($email) };

### Gmail



### Monitoring script (SystemMonitor.sh)

It contains a number of functions (which interact with the wire, or the hardware), and as such execute utilities such as ping, netstat, top etc. These Level 0 functions are referenced by more abstract level 1 functions, which are then called to log, report and alert.

####################### SystemMonitor.sh ####################################

# Date: 01/01/2014

# Author: Jonathan Harris

# Student Id: x13118901

# Description: Level 1 Functions to monitor deployment environment dependancies:

# isApacheRunning

# isApacheListening

# isApacheRemoteUp

# isApacheUsageCPUwithinLimits

# isMysqlRunning

# isMyPinging

#

# Dependencies: SystemStatusFuncLevel\_0.sh - Level 0 System Functions

#

##############################################################################

#!/bin/bash

# Jonathan Harris 2014

source SystemStatusFuncLevel\_0.sh # Level 0 System Functions

source SystemStatusFuncLevel\_1.sh # Level 1 System Functions

# Functional Body of SystemMonitor script <----------------------------

ERRORCOUNT=0

isApacheRunning

if [ "$?" -eq 1 ]; then

echo Apache process is Running

else

echo Apache process is not Running

ERRORCOUNT=$((ERRORCOUNT+1))

fi

isApacheListening

if [ "$?" -eq 1 ]; then

echo Apache is Listening

else

echo Apache is not Listening

ERRORCOUNT=$((ERRORCOUNT+1))

fi

isApacheRemoteUp

if [ "$?" -eq 1 ]; then

echo Remote Apache TCP port is up

else

echo Remote Apache TCP port is down

ERRORCOUNT=$((ERRORCOUNT+1))

fi

isApacheCpuUsageWithinLimits

if [ "$?" -eq 1 ]; then

echo Apache within limits

else

echo Apache outside limits

ERRORCOUNT=$((ERRORCOUNT+1))

fi

isMysqlRunning

if [ "$?" -eq 1 ]; then

echo Mysql process is Running

else

echo Mysql process is not Running

ERRORCOUNT=$((ERRORCOUNT+1))

fi

isMysqlListening

if [ "$?" -eq 1 ]; then

echo Mysql is Listening

else

echo Mysql is not Listening

ERRORCOUNT=$((ERRORCOUNT+1))

fi

isMysqlRemoteUp

if [ "$?" -eq 1 ]; then

echo Remote Mysql TCP port is up

else

echo Remote Mysql TCP port is down

ERRORCOUNT=$((ERRORCOUNT+1))

fi

isMyPinging

if [ "$?" -eq 1 ]; then

echo Remote IP is alive

else

echo Remote IP is down

ERRORCOUNT=$((ERRORCOUNT+1))

fi

if [ $ERRORCOUNT -gt 0 ]

then

echo "There was a problem on the server email sent to System Admin"

perl sendmail.pl "There is a problem on the server see SystemMonitor.out for further info"

fi

### Level 0 function (SystemStatusFuncLevel\_0.sh)

####################### SystemStatusFuncLevel\_0.sh ###############################

# Date: 01/01/2014

# Author: Jonathan Harris

# Student Id: x13118901

# Description: Level 0 Functions to monitor deployment environment dependancies

# isRunning

# isTCPlisten

# isUDPlisten

# isIPalive

# isTCPlisten

# isTCPlisten

# Dependencies: /bin/bash

#

#######################################################################################

#!/bin/bash # script is to be interpreted and run by bash in Bourne Shell mode

#-------------------- Level 0 function block --------------------------

function isRunning {

PROCESS\_NUM=$(ps -ef | grep "$1" | grep -v "grep" | wc -l)

if [ $PROCESS\_NUM -gt 0 ] ; then

return 1

else

return 0

fi

}

function isTCPlisten {

TCPCOUNT=$(netstat -tupln | grep tcp | grep "$1" | wc -l)

if [ $TCPCOUNT -gt 0 ] ; then

return 1

else

return 0

fi

}

function isUDPlisten {

UDPCOUNT=$(netstat -tupln | grep udp | grep "$1" | wc -l)

if [ $UDPCOUNT -gt 0 ] ; then

return 1

else

return 0

fi

}

function isTCPremoteOpen {

timeout 1 bash -c "echo >/dev/tcp/$1/$2" && return 1 || return 0

}

function isIPalive {

PINGCOUNT=$(ping -c 1 "$1" | grep "1 received" | wc -l)

if [ $PINGCOUNT -gt 0 ] ; then

return 1

else

return 0

fi

}

function myCPUusage {

app\_name=$1

cpu\_limit=$2

app\_pid=`ps aux | grep $app\_name | grep -v grep | awk {'print $2'}`

app\_cpu=`ps aux | grep $app\_name | grep -v grep | awk {'print $3\*100'}`

if [[ $app\_cpu -gt $cpu\_limit ]]; then

return 0

else

return 1

fi

}

#\*\*\*\*\*\*\*\*\*\*\*\*\*\* END OF FILE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### Level 1 function (SystemStatusFuncLevel\_1.sh)

####################### SystemStatusFuncLevel\_1.sh ################################

# Date: 01/01/2014

# Author: Jonathan Harris

# Student Id: x13118901

# Description: Level 1 Functions to monitor deployment environment dependancies:

# isApacheRunning

# isApacheListening

# isApacheRemoteUp

# isApacheUsageCPUwithinLimits

# isMysqlRunning

# isMyPinging

#

# Dependencies: SystemStatusFuncLevel\_0.sh - Level 0 System Functions

#

###################################################################################

#!/bin/bash # script is to be interpreted and run by bash in Bourne Shell mode

source SystemStatusFuncLevel\_0.sh # Level 0 System Functions

#--------------------------> Level 1 function block <---------------------------------------

function isApacheRunning {

isRunning apache2

return $?

}

function isApacheListening {

isTCPlisten 80

return $?

}

function isApacheRemoteUp {

isTCPremoteOpen 127.0.0.1 80

return $?

}

function isApacheCpuUsageWithinLimits {

myCPUusage Apache 5000

return $?

}

function isMysqlListening {

isTCPlisten 3306

return $?

}

function isMysqlRunning {

isRunning mysqld

return $?

}

function isMysqlRemoteUp {

isTCPremoteOpen 127.0.0.1 3036

return $?

}

function isMyPinging {

isIPalive 127.0.0.1

return $?

}

#\*\*\*\*\*\*\*\*\*\*\*\*\*\* END OF FILE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Test Plan

## Unit tests (UnitTesting.sh)

######################### UnitTesting.sh ########################

# Date: 01/01/2014

# Author: Jonathan Harris

# Student Id: x13118901

# Description: Unit Test of Level 0 & Level 1 Functions to monitor deployment

# i.e. Apache2, MySql, Remote TCP Link

# Dependencies: ./scripts/SystemStatusFunc.sh

# Apache2, MySql, Remote TCP Link

##############################################################################

#!/bin/bash # script is to be interpreted and run by bash in Bourne Shell mode

source SystemStatusFuncLevel\_0.sh # Level 0 System Functions

################## Unit Test System Status Level 0 System Status Functions ##################

# Report Date and UT Result Title

echo $(date +"%F %T") status

echo Unit Test System Status Level 0 System Status Functions

# UT Level 0 Function isRunning with Parameter apache2

UTFAILED=0

UTCOUNT=0

function UnitTestAtom {

echo "Positive Unit Test of $1 using $2 $4 paramaeters"

UTFAIL=0

$1 $2 $4

if [ "$?" -gt 0 ]; then

echo "Positive UT Passed"

else

echo "Positive UT Failed"

UTFAIL=$((UTFAIL+1))

fi

echo "Negative Unit Test of $1 using $3 $5 paramaeters"

$1 $3 $5

if [ "$?" -eq 0 ]; then

echo "Negative UT Passed"

else

echo "Negative UT Failed"

UTFAIL=$((UTFAIL+1))

fi

return $UTFAIL

}

#UT isTCPlisten

UnitTestAtom isRunning apache2 RUBBISH

UTCOUNT=$((UTCOUNT+1))

#UT isTCPlisten

UnitTestAtom isTCPlisten 80 9999

UTCOUNT=$((UTCOUNT+1))

#UT isUDPlisten

UnitTestAtom isUDPlisten 68 9999

UTCOUNT=$((UTCOUNT+1))

#UT isIPalive

UnitTestAtom isIPalive "127.0.0.1" "256.0.0.9"

UTCOUNT=$((UTCOUNT+1))

#UT isTCPremoteOpen

#UnitTestAtom isTCPremoteOpen 53 55

#UTCOUNT=$((UTCOUNT+1))

#UT myCPUusage

UnitTestAtom myCPUusage Apache Rubbish 100 999

UTCOUNT=$((UTCOUNT+1))

if [ $UTFAIL -gt 0 ]; then

echo "\*\*\*\*\*\*\*\*\*\*\* Unit Testing failed \*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "No of Failed UT Total=" $((UTFAIL))

else

echo "\*\*\*\*\*\*\*\*\*\*\* Unit Testing Passed \*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "Unit Testing Passed $((UTCOUNT-UTFAIL))"

fi

echo Total No of UTests= $UTCOUNT

################### End of Script ########################

# Integration testing (UnitTesting.out)

2014-01-20 14:24:12 status

Unit Test System Status Level 0 System Status Functions

Positive Unit Test of isRunning using apache2 paramaeters

Positive UT Passed

Negative Unit Test of isRunning using RUBBISH paramaeters

Negative UT Passed

Positive Unit Test of isTCPlisten using 80 paramaeters

Positive UT Passed

Negative Unit Test of isTCPlisten using 9999 paramaeters

Negative UT Passed

Positive Unit Test of isUDPlisten using 68 paramaeters

Positive UT Passed

Negative Unit Test of isUDPlisten using 9999 paramaeters

Negative UT Passed

Positive Unit Test of isIPalive using 127.0.0.1 paramaeters

Positive UT Passed

Negative Unit Test of isIPalive using 256.0.0.9 paramaeters

Negative UT Passed

Positive Unit Test of myCPUusage using Apache 100 paramaeters

Positive UT Passed

Negative Unit Test of myCPUusage using Rubbish 999 paramaeters

Negative UT Failed

\*\*\*\*\*\*\*\*\*\*\* Unit Testing failed \*\*\*\*\*\*\*\*\*\*\*\*\*

No of Failed UT Total= 1

Total No of UTests= 5

# Execution plan

The instructions on how to test the process is shown in the Record of demonstration document below.

# Record of demonstration

The demonstration of execution on a virtualbox environment was recorded on the document on following:

To execute the deployment script, I enlisted the help of a third party, User X. User X was given a series of tasks to perform, beginning from running the deploy script to checking for errors.

The following is the execution plan. It conducted by User X. I observed the execution of the tasks with the following comments below.

|  |  |  |
| --- | --- | --- |
| Step | Topic | Verification Y/N |
| 1 | Clear Environment |  |
| 2 | Run Deployment scripting |  |
| 3 | Demonstrate static content is delivered through browser |  |
| 4 | Demonstrate dynamic content is delivered through browser |  |
| 5 | Demonstrate Logging is functioning correctly |  |
| 6 | Run Deployment scripting again |  |
| 7 | Demonstrate Content is delivered correctly and logging is functioning correctly |  |
| 8 | Stop  web server,  verify logging of event in log |  |
| 9 | Stop  Database, verify logging of event in log |  |
| 10 | Start web server, verify logging of even in log |  |
| 11 | Start Database, verify logging of event in log |  |
| 12 | Change the content in code repository. Run the deployment scripting. Verify that Content is delivered correctly and logging is functioning correctly. |  |

Executed by : \_\_\_\_\_\_\_\_\_\_\_Witnessed by : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_