Automated capturing and reporting of build and deployment Success and Failures

Submitted By

Bridgeeta R Rathod

16MCEC15

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (CSE)

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2018

Automated capturing and reporting of build and deployment Success and Failures

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering

Submitted By

Bridgeeta R Rathod

(16MCEC15)

Guided By

Prof. Monika Shah

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2018

Certificate

This is to certify that the major project entitled **"Automated capturing and reporting of build and deployment Success and Failures"** submitted by **Bridgeeta R Rathod (Roll No: 16MCEC15)**, towards the fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad, is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project part-II, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Monika Shah
Guide & Assistant Professor,
CE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Priyanka Sharma
Professor,
Coordinator M.Tech - CSE
Institute of Technology,
Nirma University, Ahmedabad

Dr. Sanjay Garg
Professor and Head,
CE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr Alka Mahajan
Director,
Institute of Technology,
Nirma University, Ahmedabad

Statement of Originality

I, **Bridgeeta R Rathod**, **Roll No: 16MCEC15**, give undertaking that the Major Project entitled "**Automated capturing and reporting of build and deployment Success and Failures**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made.It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

_____

Signature of Student

Date:

Place:

Endorsed by

Guide Name

(Signature of Guide)

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Prof. Monika Shah**, Assistant Professor, Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support she has imparted has been a great motivation to me in reaching a higher goal. Her guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr. Alka Mahajan**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation he has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

- Bridgeeta R Rathod

16MCEC15

Abstract

Capturing and reporting of build and deploy failures and success is a measure for Company to match results obtained by its analysis with preset thresholds to decide the stability of product and its release status to external customers. In this project, supervised learning method is used to data mine the existing records and build a training set for future automated data mining. This project is also taken to overcome the barriers of architecture change during move from Hudson to Jenkins. This project will be deployed on Jenkins as a test job in replacement to Hudson. If successful all the Build and Deploy jobs will be run in automation environment using Jenkins. Job analysis automation is a project that will reduce everyday 4 hours effort of a employee. Also it will increase the precision of output delivered and hence increasing the quality of product delivered to external customer ensuring the stability of product. Project will be developing scripts to capture the success and failure of jobs triggered and reporting and classifying their cause of failure.

Abbreviations

DSS	Development Support and Services.
SSO	Single Sign-On.
SVN	Sub-Version.
C/I	Check-in
C/O	Check-out
Dev	Development team
QA	Quality Assurance team
OID	Oracle installation Directory
OBIEE	Oracle Business Intelligence Enterprise Edition

Contents

List of Figures

Chapter 1

Introduction

1.1 Team Overview

Development Support Services(DSS) is an integral part of every software company, including Oracle Retail. It deals with Build and Packaging of products, Database and Server Management and monitoring, and Server Maintenance. The aim of this project was to reduce the tremendous amount of workload of the DSS, by contributing to some aspects of the responsibilities of the team.

1.2 Project Definition

Analyzing and automating the manual work of capturing and reporting success and failure rates of build and deploy triggered by the internal team of Oracle Retail.

Project will be developing scripts to:

- Capture success and failures of particular job.
- Reporting number of success and failure.
- Classify the cause of error.

Also as a part of DSS team, Studying the barriers caused for scripts while building them and deploying them on Hudson using central repository SVN tool.

1.3 Objective

Job analysis automation is a project that will reduce everyday 4 hours effort of a employee. Hence, it will save 126 man-days of Oracle Retail annually if this project is automated on Hudson using SVN.

Training set will be automatically trained for new entries of data. It will increase the precision of output delivered by the automation developed than the manual work done giving better stability analysis of the product to be delivered.

1.4 Work sections

1.4.1 Build

Build is the task of running code scripts to generate artifacts that can be used to launch an application for customer on their machine. Building a product's artifacts involves various distinct functions:

- 1. Version Control : The version control function creates storage medium where actual code is checked out and processed upon and artifacts are built there.
- 2. Code Quality : Code analysis is done to make sure the quality of code has matched the organization coding standards. Once code matches the standards it is sent out as production code.
- 3. Compilation : Compilation gives executable code after compiling thousands of code files using their dependent library.

1.4.2 Deploy

Deploy is the task of deploying the build of a product on application server to make product usage accessible to customers.[1]. It involves many distinct functions[1] :

- 1. Build : Build of a product provides necessary artifacts for the deployment of product that includes the code, dependent library functions, and property files to run the deployment.

- 2. Application Server : Server used to deploy the application using a Weblogic server configurations and made available to customers for running the application forms on that server.

1.4.3 Automation

Automation here can be defined as a set of interdependent scripts that will run using tools SVN and Hudson to to perform the defined task.

It will calculate the number of times job was triggered for every single job in Hudson. Also script will data mine through the logs of jobs ran to find the number of success and failures. Then it data mines each log to find the cause of failure and classify them according to the reason for failure.

Chapter 2

Literature Survey

2.1 Overview

Project is about knowing the success rate of deployments of application and also knowing the reason of failure if at all present. By knowing the reason of failure, we classify the failure types according to their reasons and the ownership of production type error.

This also gives clearance on which team owns what percentage of error causes. Frequent errors can be given personal attention by higher authorities for a longer term solution as well as better quality of delivery.

This data mining will output the following fields in excel.

- Build number
- Date of build completion
- Environment name
- Failure cause
- Team name
- Team code
- Production error code

2.2 Flow of Work

Here we studied the entire product installation life cycle to its working after coding and testing is completed for a product. This is a generic flow graph for product installa-

tion and checking the environment setting so that product gets deployed and running. Environment setting and dependent setting are taken care of in pre-installation for the required product. Also it tests the jars, ears and dependent libraries packaged for the product before delivering to customer. This includes the installer testing and environment testing. Environment includes weblogic server, database, deployments, and property files.

- It checks for the pre-installation tasks. Like setting paths to JAVA HOME, WEBLOGIC HOME, DATABASE HOME etc.

- If database installation exists for that product, it installs database by creating tables, users, schemas and tablespaces. Also it enters data according to requirements to default values set by dev for running first time. If any errors are detected during installation then it tries to resolve by itself. For example, there is a error caused which is resolved by running a DIANA query. This is done by a automated script.

- It checks for if weblogic installation is required, if so it installs weblogic using Oracle Fusion Middleware.

- [1]It then installs OID, Oracle Internet Directory is a LDAP v3 compliant directory with meta-directory capabilities. It is built on the industry leading Oracle database and is fully integrated into Oracle Fusion Middleware and Oracle Applications. Thus, it is ideally suited for Oracle environments or enterprises with Oracle database expertise. Similar to ODSEE, OID is also proven with large deployments in carrier and enterprise environments.

- [2]Oracle Business Intelligence Enterprise Edition (OBIEE) is a Business Intelligence (BI) tool by Oracle Corporation. Its proven architecture and common infrastructure producing and delivering enterprise reports, scorecards, dashboards, ad-hoc analysis, and OLAP analysis provides a rich end-user experience. This tutorial explains all the fundamental aspects of OBIEE.

- RCU creation is repository creation tool that creates schema and prefixes for creating weblogic domain. It basically allocates memory for as metadata space for weblogic creation in those repositories.

- Running installer takes few steps and user inputs. After pressing install button, application installation happens, if that goes successful we check it by launching

the URL deployed on weblogic server and logging in successfully implies successful installation of application.

- At end, it stores all this installation summary in a log file. For each step, there is a different log file created to easily debug the error.

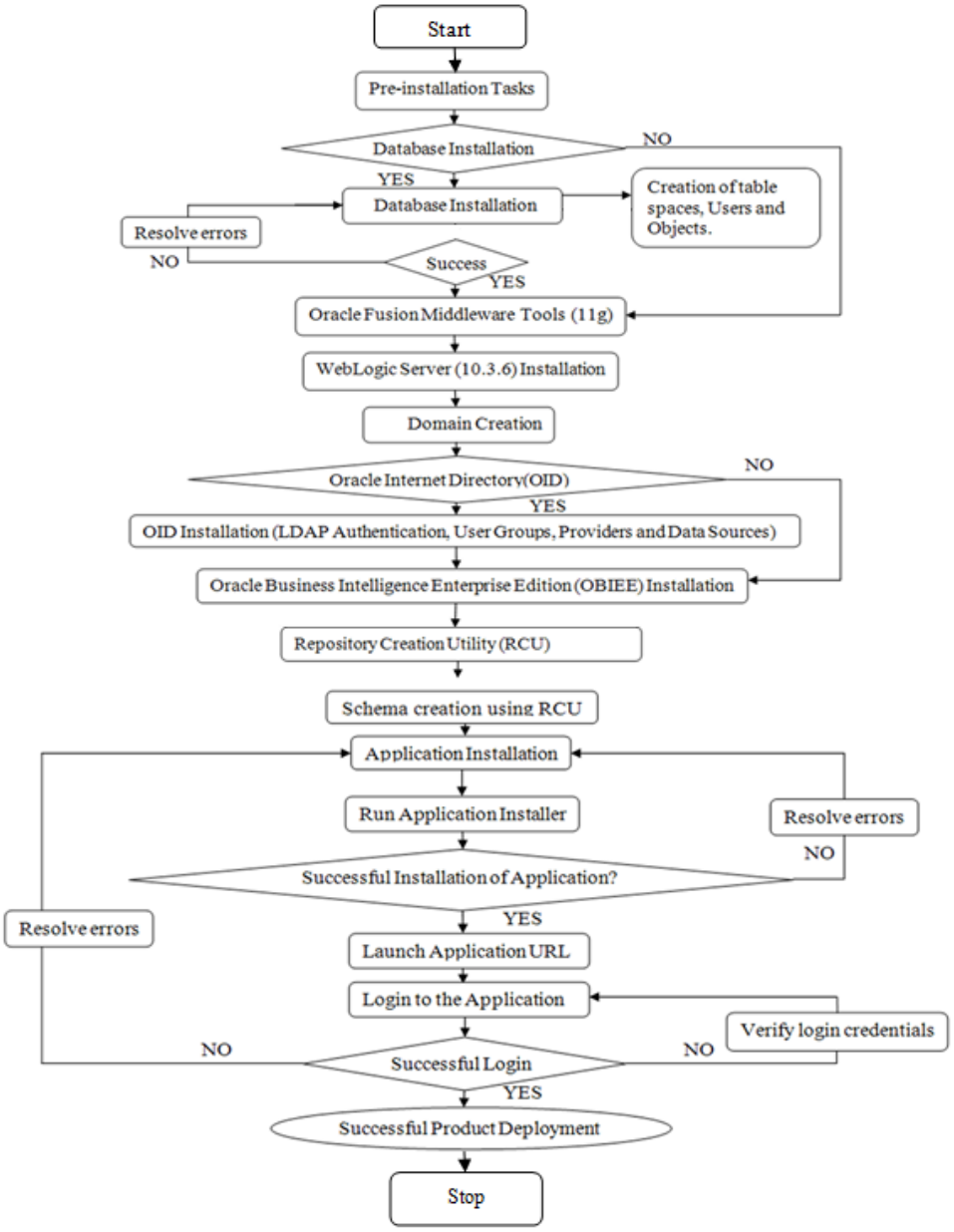


Figure 2.1: Flow of DSS work

2.3 Existing Approach

Existing approach is to make this data sheet in excel manually. We open Hudson URL where all jobs are present. We open each job's URL and check the section of webpage where the time and date along with build number are shown as in image here:

Figure 2.2: Hudson job console

From this we get total number of triggers each day, the date of trigger, the environment name and console output button. Also if the button near the build is green it means the build was successful but if it is read it means the build failed in execution. When we click console output button the log is displayed on Hudson screen. In existing approach, manually go through the log and find the error from that. Take that error and note it in excel. Also classify that error to get team name, team code and production error code from pre-defined criteria.

Chapter 3

Tools

3.1 Hudson Continuous Integration System

3.1.1 Hudson Introduction

[3]The term "Continuous Integration system" means a system that can integrate changes into the presently available code and reflect its changes dynamically. It can also notify all the users of presently available system about the changes. The main job of a continuous integration system is to monitor the execution of repeated job. Hudson performs two basic jobs as follows[5]:

- Testing and Building : Building and testing the code that has changes can be easily integrated for Dev by this tool. Hudson increases productivity by allowing user to obtain a fresh build every time required.
- Monitoring execution : External executed jobs can be easily monitors and regular notifications can be generated via emails about the log of output created by triggering the build. This notifications can be customized for sending for successful completion or failure as well.

Features of Hudson:

1. [4] Installation : Installing Hudson is as easy as downloading the latest jar from Hudson website and following command in windows command prompt: java – jar ¡absolutepathtojar¿ – httpPort ¡PortNo¿ The default port used by Hudson is "8080" but we manually enter an unused port number as per our requirement using the "httpPort" option.

2. Configuration: Hudson uses GUI for configuration making it more likeable for usage due to its ease of configuring complex tasks easily using a GUI. Hudson provides number of options to complete tasks like create/configure a job, add/configure slave, configure nodes, add plugins for variety of functionalists etc.

3. Summary: Hudson summarizes all the configuration changes made and also provides a functionality to revert back those changes. It also shows the difference between changes in different versions of that job changes.

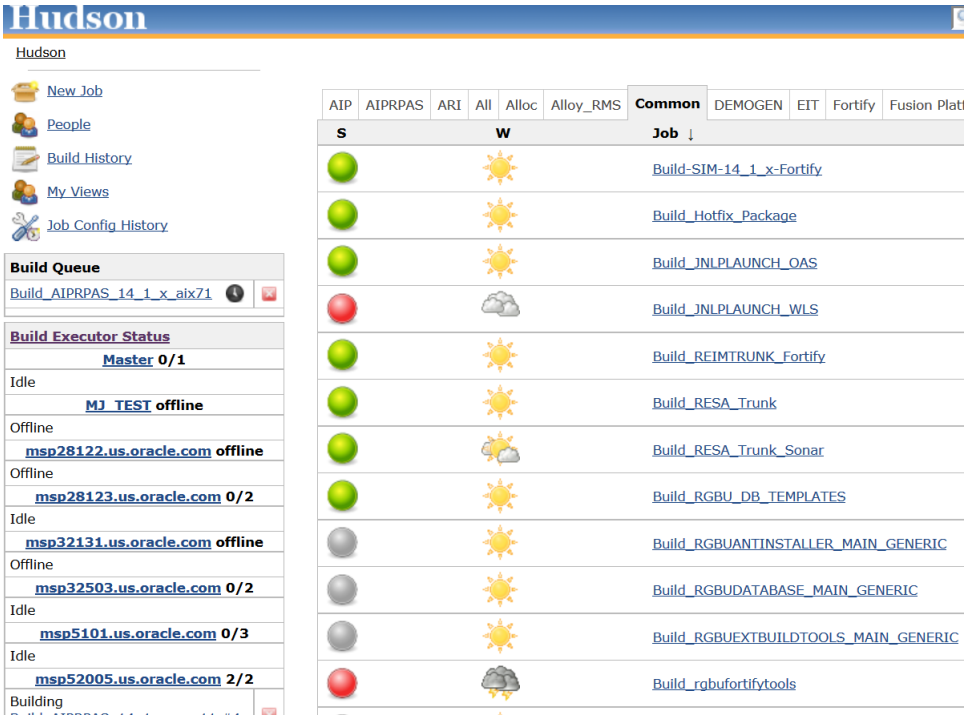


Figure 3.1: Hudson GUI Snapshot

4. Links to builds: It creates links to artifacts built, last successful build, last stable build, etc. This links are useful for restoration and easy access to artifacts directly from GUI without going to the server.

5. Plugins: Hudson provides it's in-build plugins and support to third party plugins for different functions to be carried out automatically using this plugins.

6. Customized environments: Allows to set customized paths, variables, executable code and checkout URLs.

3.1.2 Master/Slave Configuration

Hudson is capable of implementing master-slave configuration using following steps.

1. Java JDK version should be 1.6 or higher version. Check using command "java -version" in windows cmd before going further.

2. Use following command to install Hudson master: "java -jar".

3. A message "successfully installed" is displayed after installation of Hudson master iuf it is a success. Open "http://localhost:(port no)" in browser. Default port number is 8080.

Install Hudson master as a Windows service

Hudson master can be installed as a Windows Startup service if onw does not want to type URL everytime to access Hudson. The steps are:

1. After starting Hudson, click manage Hudson link and then click the "Install as Windows Service" link on the management page.

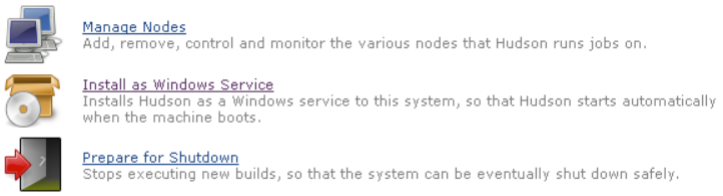


Figure 3.2: Hudson GUI Screen

2. After that installation screen opens as follows:

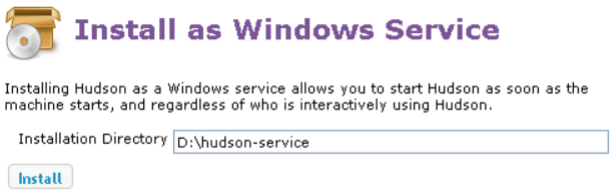


Figure 3.3: Hudson GUI Screen

3. Set HUDSON HOME by selecting an existing directory on local machine where Hudson is to be installed. This directory will be used as a storage medium for all Hudson tasks.

4. After installing Hudson successfully as a windows service, admin will need to restart Hudson master server to use windows service.

5. To check successful execution of Windows service, go to "start panel -¿ Services". A page like below appears:

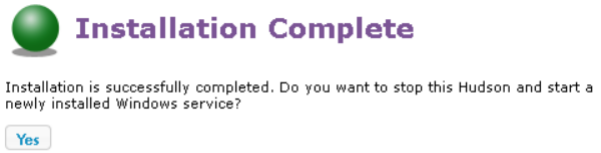


Figure 3.4: Restart screen

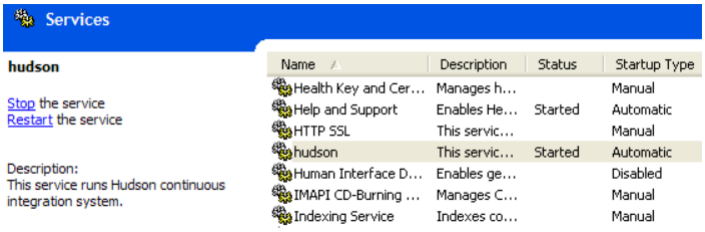


Figure 3.5: Windows Start up service

Configure/Install Hudson Slave

[5] Hudson jobs are mostly ran on slave nodes in case of product companies. Configuring such slaves is explained here:

1. On master Hudson screen, click Manage Jenkins -¿ Manage Nodes.

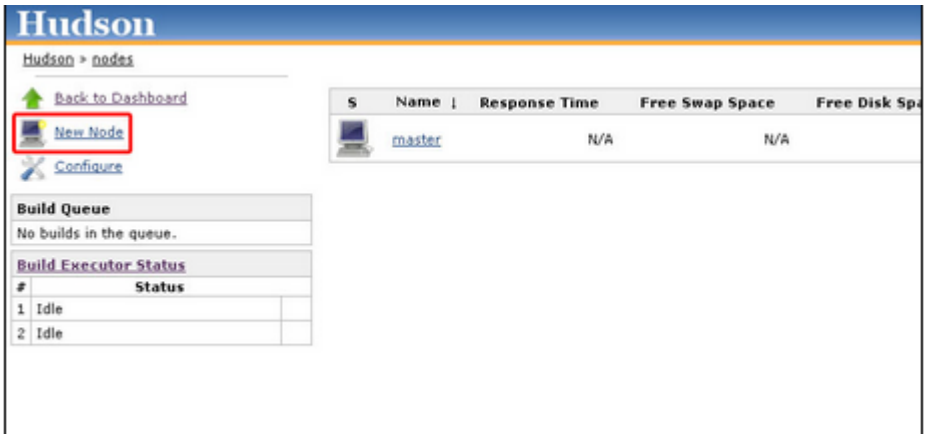


Figure 3.6: Configuring Hudson Slave

2. Click New Node-¿Enter Node Name.

3. Click "Dumb Slave" and press "OK" on screen as follows:

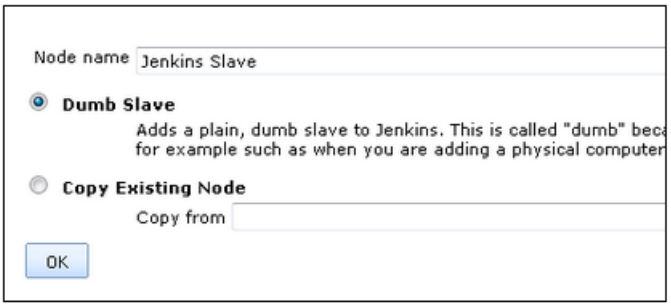


Figure 3.7: Slave configuration

4. Fill in the details. Here, Remote FS Root is "a slave directory on master Hudson".Give path on a local machine that already exists and is being configured as a slave of master Hudson.

Figure 3.8: Slave configuration

5. This simply creates a machine and it can be connected to master as follows:

   - Launch "http://localhost:(portno)" in browser on slave machine.
   - Go to "Manage Jenkins" -¿ "Manage Nodes".
   - Click on the newly created Slave:
   - Copy URL and run in cmd on Slave machine or click the launch button.
   - A jnlp agent gets downloaded. Running it following slave window appears:

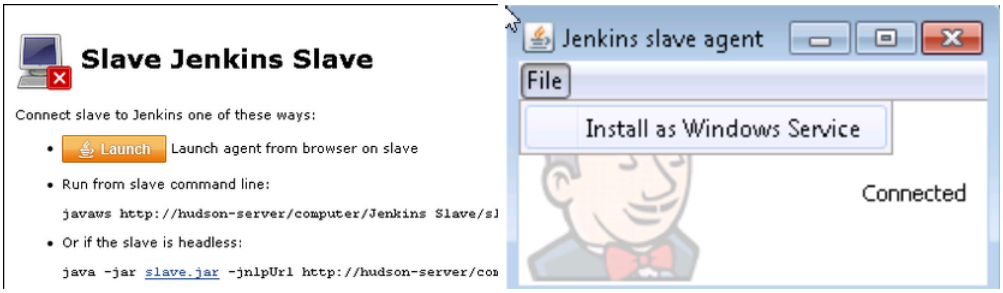


Figure 3.9: Hudson Slave launch window

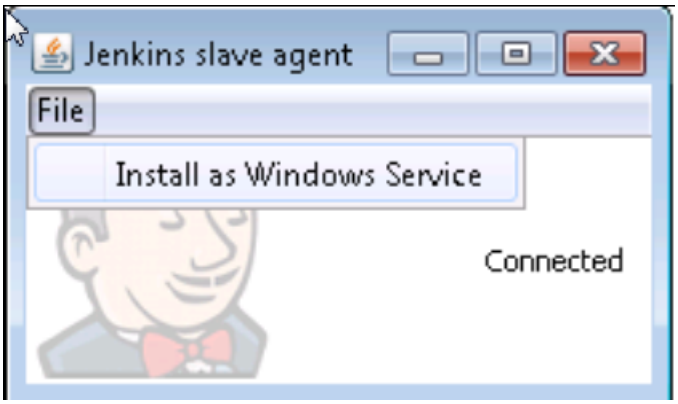


Figure 3.10: Hudson Slave

6. Eve Hudson slave can also be used as a windows service, "Slave as a service" by clicking on "Install as Windows Service".

3.1.3 New job configuration

A Hudson job on slave is configured set of mechanism for running scripts or code snippets. It provides a GUI for running complex set of scripts sequentially. Configuration steps:

1. Launch URL "http://localhost:(portno)"

2. Click on "New Job".

3. fill in this detail "name of the job" and click "build a free style project".Click "save". It displays us to a "Configuration page" as follows:

4. Select "Restrict where project is to be run". All slave node addresses are present in a drop down menu. Select one of them and the job will run on that node.

5. In "Execute build" option, we can specify the code we need Hudson to run or the location of script which is to be run as follows:

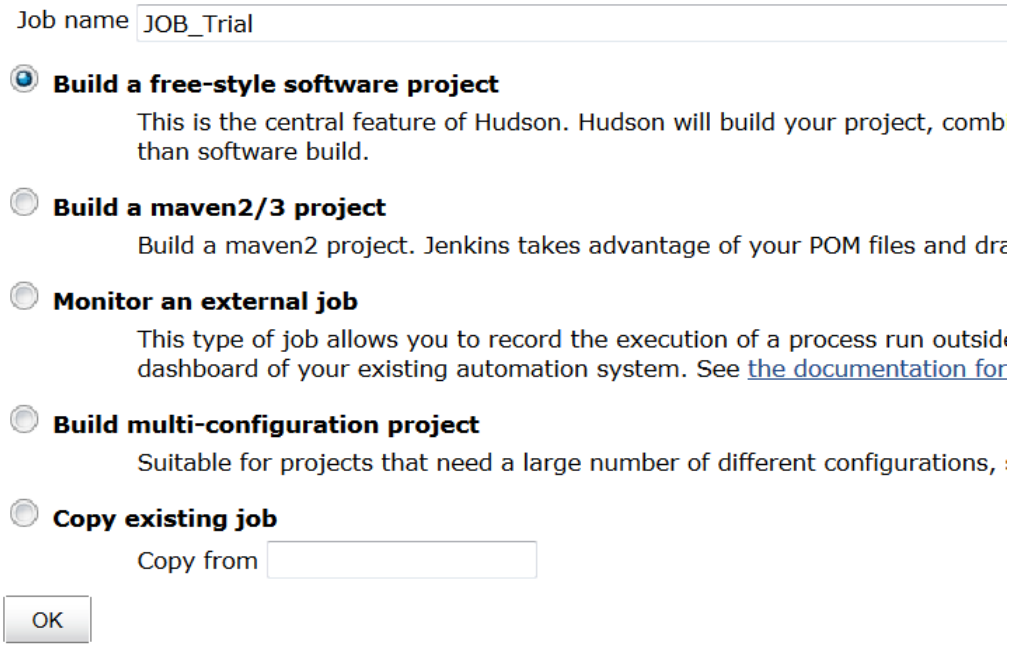


Figure 3.11: New job on Hudson

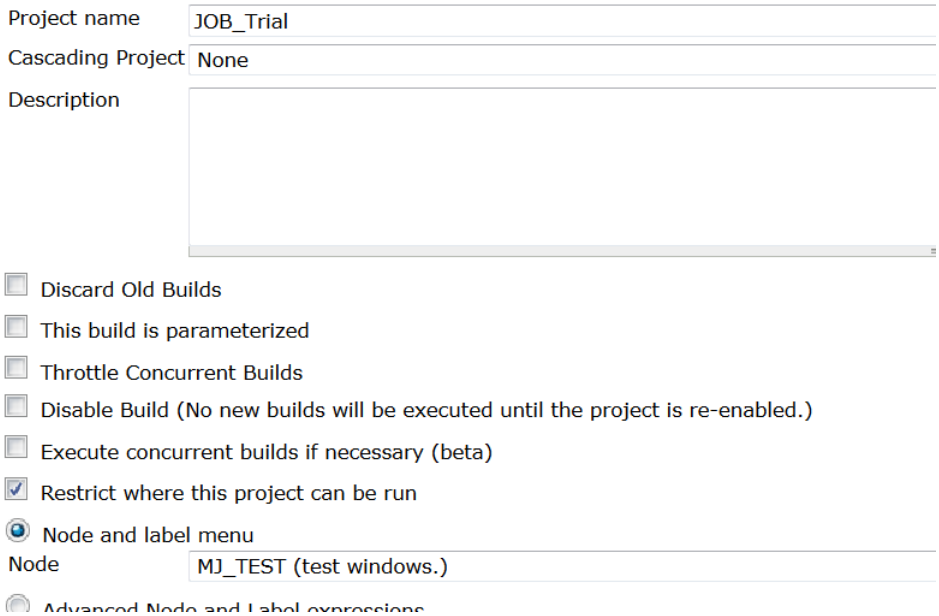


Figure 3.12: Job configurtaion Screen

6. Click "Save" finally after above configuration steps. A job, if successfully configured displays the following screen:

7. "Workspace" option shows all the stored artifacts when expanded that are inside "Remote FS Root" on Hudson Slave machine.

8. "Recent changes" shows summary of changes and the difference between last stable

Figure 3.13: Build specification Screen

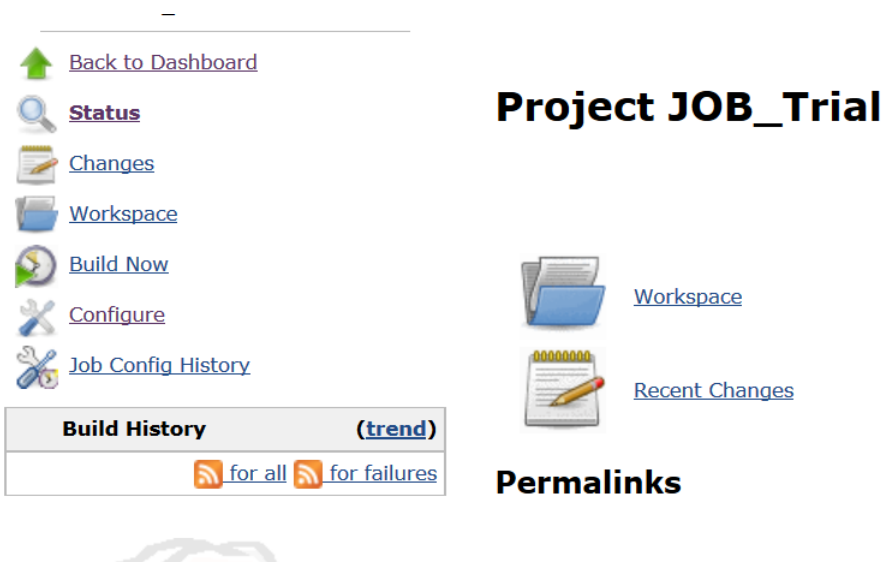


Figure 3.14: Build specification Screen

build and the current build changes in the configuration of build since last successful build was triggered.

9. "Permanent links" are links like "Last unsuccessful build", "Last stable build", "Last successful build" and links to artifacts of last successful build.

3.2 Tortoise SVN

3.2.1 Definition

SVN is centralized repository system that allows storage and access at a central location and also a revisiong control system.

To trigger a build or to modify any file that can be code, script, property file etc., we need to download it from a central repository called a subversion(SVN) repository. This process is called "Checkout". Basically a "check-out" is downloading a latest version of the required script from central database using a URL into local machine for mod-

ifications. Similarly, "check-in" is submitting your code from local machine to central repository after modifications. A software used for this purpose is "Tortoise SVN".

3.2.2 Features

- Maintains central copy for easy access over network.
- Secured access with credential authorization
- Allows customized access to different repositories to different group of users.
- Maintains logs for changes made along with credentials of user.
- Allows to revert back changes to older versions.

3.2.3 Installation

Prerequisite: Runs on any Windows higher in version than Windows Vista and is available for 32-bit and 64-bit systems.

Download installer with .exe extension and Install SVN. Restart the system and right click on any folder will show "SVN Check-out" option which will allow to check-out code in any directory specified.

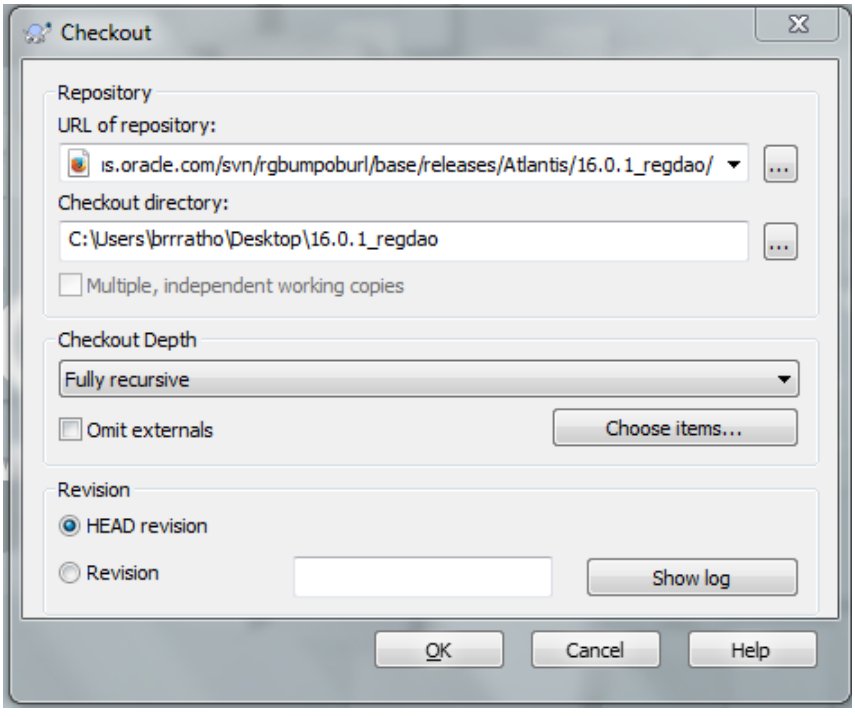


Figure 3.15: SVN Check-out Screen

- Mention URL from where code needs to be downloaded in "URL Repository".
- Mention local path to save the file in "Checkout Directory".
- If one needs to checkout folders and sub-folders then click "Recursive depth".

3.3 SSH Login

To have access to log files and other artifacts on any server we have a secure way to log in into those servers and access data. SSH Login has unique username and password set for each user of that machine.

If a person wants to access any data on a machine, it will first require a valid login with credentials and person should belong to valid group to access that data owned by a particular group.

Since this is an automation project where we are trying to avoid human interference and interactive programming, SSH provides a secured way out of entering username and password manually.

SSH setup in this project is necessary as there are thousands of log files to be checked and they might not be necessarily present on same machine. Hence to access logs and builds on different machines, it is unfeasible to run script on each machine as it will lead to asynchronization. So program will run on a single machine but will fetch data from multiple machines.

SSH connection is setup between the machines where this script will be executed and machine where data and artifacts are present.

Hence, summarizing SSH connection is used for following purposes and features:

- Secured way to access a machine with entering username and password manually.
- SSH mechanism created a pair of public key and private key. Public key will be on machine that will execute script and try to connect to other machine for data where private key will be present.
- When a machine tries to connect to other machine it sends it public key which second machine tries to match with its private key, and as it matches a SSH connection is setup between the two machines that allows password-less data transfer and connection.

- Also this SSH connection between two machines can be used to create processes on other machine with proper permissions for same.

[6]Steps to set-up SSH keys:

- Creation of RSA key pair: To set up a connection between two machines, we need a pair of public key and private key. For this we run following command on machine

  ssh – keygen – trsa This will create two files key.pub and key.private

- Key storage: Keys are stored on machines that require SSH connection between them. These keys are to be stored in a folder .ssh that should not have any more permissions than 600.

  Public key will be located at /home/user/.ssh/id-rsa.pub on machine that tries to connect to other machine. Private key will be at /home/user/.ssh/id-rsa on machine to which connection is to be made.

- Public key: To place public key into second machine we copy public key using following command ssh – copy– iduser@hostnameorhostipaddress This command puts public key to authorizedkeys on second machine.

  Output on running this command will be

  After this setup any machine can accessdata over any other machine that has SSH setup.

3.4 Scripting Languages

3.4.1 Shell Scripting

Scripting languages are the most important block of automation. Scripting language is the body of automation while logic is brain to the script. One of the most powerful scripting language is Shell Script which is used widely with the most powerful OS Linux and the modern day language Python are used for this project.

3.4.2 Python

Basic coding for data mining and classification done using Python.

```
ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/demo/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/demo/.ssh/id_rsa.
Your public key has been saved in /home/demo/.ssh/id_rsa.pub.
The key fingerprint is:
4a:dd:0a:c6:35:4e:3f:ed:27:38:8c:74:44:4d:93:67 demo@a
The key's randomart image is:
+--[ RSA 2048]----+
|          .oo.   |
|         .  o.E  |
|        + .  o   |
|     . = = .     |
|      = S = .    |
|     o + = +     |
|      . o + o .  |
|           . o   |
|                 |
+-----------------+
```

Figure 3.16: SSH key generation

```
The authenticity of host '12.34.56.78 (12.34.56.78)' can't be established.
RSA key fingerprint is b1:2d:33:67:ce:35:4d:5f:f3:a8:cd:c0:c4:48:86:12.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '12.34.56.78' (RSA) to the list of known hosts.
user@12.34.56.78's password:
Now try logging into the machine, with "ssh 'user@12.34.56.78'", and check in:

  ~/.ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

Figure 3.17: SSH key setup

Python Data library Panda is used to get the required format of output in Excel sheet. It is open source data analysis tool used to get result in Excel format. Pandas is an open source and BSD-licensed library. This data library provides high-performance data ananlysis and it has more features like easy-to-use data structures and data analysis. Pandas is compatible with Python programming language with any version greater than Python3

Below are some features of :

[7]

- A quick and proficient DataFrame question for information control with incorporated ordering;
- Apparatuses for reading and writing information between in-memory data structures and diverse organizations: CSV and documents, Microsoft Excel, SQL databases, and the quick HDF5 design;
- Astute data arrangement and coordinated treatment of missing data: increase programmed mark based arrangement in calculations and effortlessly control chaotic information into a systematic shape;
- Adaptable reshaping and rotating of informational indexes;
- Savvy name based slicing, fancy indexing, and subsetting of expansive informational indexes;
- Sections can be embedded and erased from information structures for size mutability;
- Elite merging and joining of data sets;
- Various leveled pivot ordering furnishes an instinctive method for working with high-dimensional information in a lower-dimensional information structure;
- Time arrangement usefulness: date go age and recurrence transformation, moving window measurements, moving window straight relapses, date moving and slacking. Indeed, even make area particular time balances and join time arrangement without losing information;
- upgraded for execution, with basic code ways written in Cython or C.
- Python with pandas is being used in a wide assortment of scholarly and business spaces, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and then some.

Chapter 4

Project Details

4.1 Basic Flow

Basic idea behind the project is to automate the manual work of making this data records in excel about the total number of trigger of each job every day. Out of which what are the number of failures and then finding the cause of failure from logs and other details of that build triggered.

Both this outputs are used to generate statistical report by analyzing them and making pictorial reports from them.

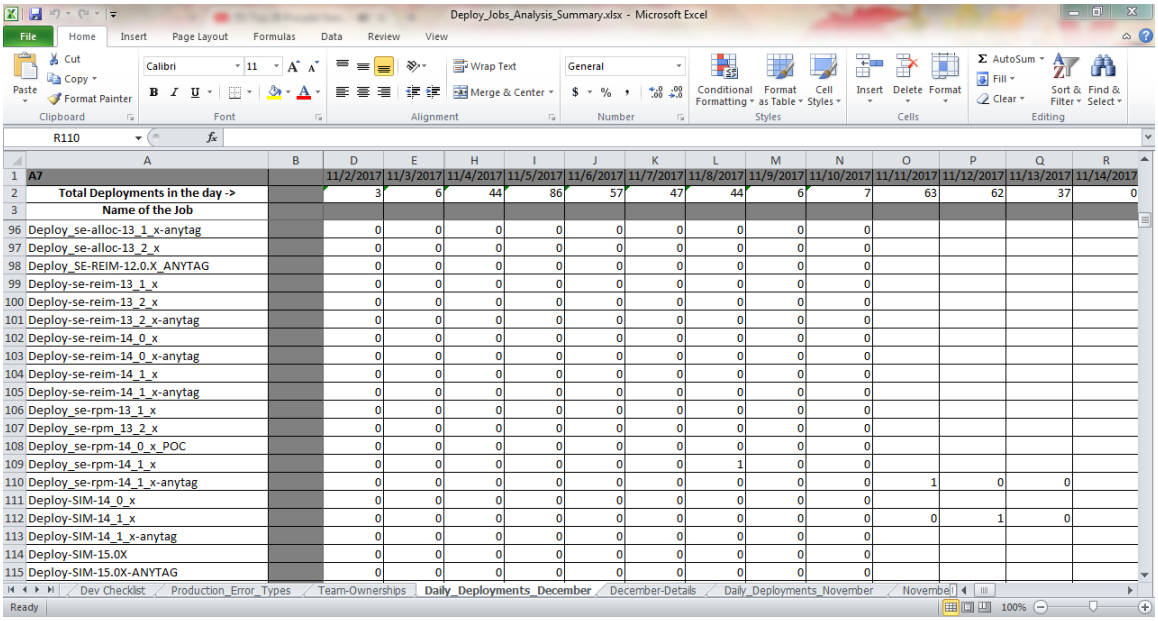
4.2 Problem Definition

4.2.1 Count total deploys/builds(Reporting)

As per this image, we have to count total number of triggers each day for each deploy. And there are such 350+ deploy jobs having there own directory as a workspace on Hudson master server. Inside a directory there are different directories for each job.

In each job's directory we can find different directory for each deploy triggered. Hence we will count number of directories for each particular date using command "ls -ltr". Count number of entries found and enter into this sheet against the Job name.

This count includes both failures and successes. It excludes the aborted builds though they are given sequential number explained below but are not counted in here. This count helps in finding the statistics and analyzing the rate of success as per the following chart.



Deploy_Jobs_Analysis_Summary.xlsx - Microsoft Excel

	A	B	D	E	H	I	J	K	L	M	N	O	P	Q	R
1	A7		11/2/2017	11/3/2017	11/4/2017	11/5/2017	11/6/2017	11/7/2017	11/8/2017	11/9/2017	11/10/2017	11/11/2017	11/12/2017	11/13/2017	11/14/2017
2	Total Deployments in the day ->		3	6	44	86	57	47	44	6	7	63	62	37	0
3	Name of the Job														
96	Deploy_se-alloc-13_1_x-anytag		0	0	0	0	0	0	0	0	0				
97	Deploy_se-alloc-13_2_x		0	0	0	0	0	0	0	0	0				
98	Deploy_SE-REIM-12.0.X_ANYTAG		0	0	0	0	0	0	0	0	0				
99	Deploy-se-reim-13_1_x		0	0	0	0	0	0	0	0	0				
100	Deploy-se-reim-13_2_x		0	0	0	0	0	0	0	0	0				
101	Deploy-se-reim-13_2_x-anytag		0	0	0	0	0	0	0	0	0				
102	Deploy-se-reim-14_0_x		0	0	0	0	0	0	0	0	0				
103	Deploy-se-reim-14_0_x-anytag		0	0	0	0	0	0	0	0	0				
104	Deploy-se-reim-14_1_x		0	0	0	0	0	0	0	0	0				
105	Deploy-se-reim-14_1_x-anytag		0	0	0	0	0	0	0	0	0				
106	Deploy_se-rpm-13_1_x		0	0	0	0	0	0	0	0	0				
107	Deploy_se-rpm_13_2_x		0	0	0	0	0	0	0	0	0				
108	Deploy_se-rpm-14_0_x_POC		0	0	0	0	0	0	0	0	0				
109	Deploy_se-rpm-14_1_x		0	0	0	0	0	0	1	0	0				
110	Deploy_se-rpm-14_1_x-anytag		0	0	0	0	0	0	0	0	0	1	0	0	
111	Deploy-SIM-14_0_x		0	0	0	0	0	0	0	0	0				
112	Deploy-SIM-14_1_x		0	0	0	0	0	0	0	0	0	0	1	0	
113	Deploy-SIM-14_1_x-anytag		0	0	0	0	0	0	0	0	0				
114	Deploy-SIM-15.0X		0	0	0	0	0	0	0	0	0				
115	Deploy-SIM-15.0X-ANYTAG		0	0	0	0	0	0	0	0	0				

Figure 4.1: Total number of triggers each day

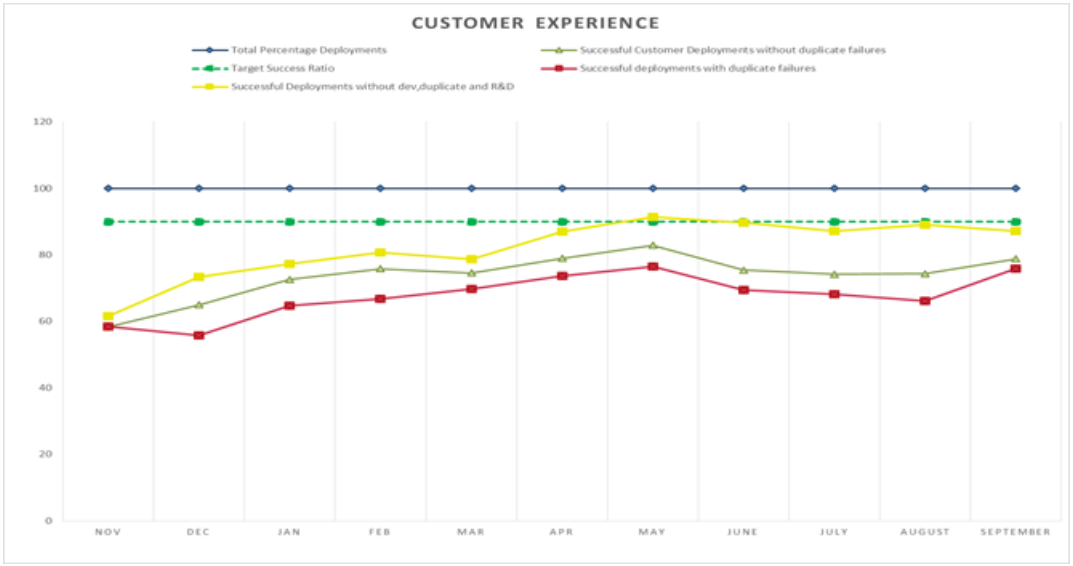


Figure 4.2: Statistics

4.2.2 Giving details of failures(Capturing)

This sheet shows the format for capturing failures and classifying them. Columns in order are:

- Job Name: This is the name of job on Hudson that usually comprises of product name and it's version number.
- Build No. : Build number is the number of build triggered for that job. For every trigger of a build starting from one it is assigned a number sequentially.

- Date: Date of build completed
- Environment Name: It is the name of environment on which application is to be deployed. Environment holds the details of application server, database, RCU, properties etc.
- Failure Log: This is the line from build log which caused failure.
- Team Name: This is the name of team because of which this failure was caused.
- Team code: Code of team responsible for failure. This field is introduced for ease of filtering data and easy access.
- Production error code: This code represents the error category.

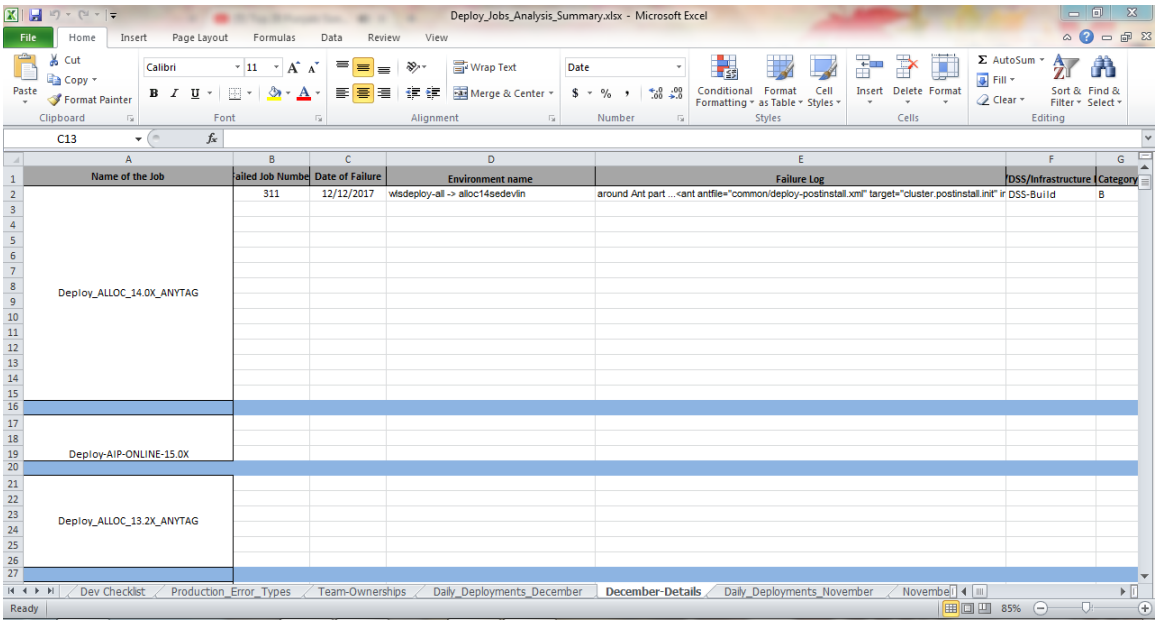
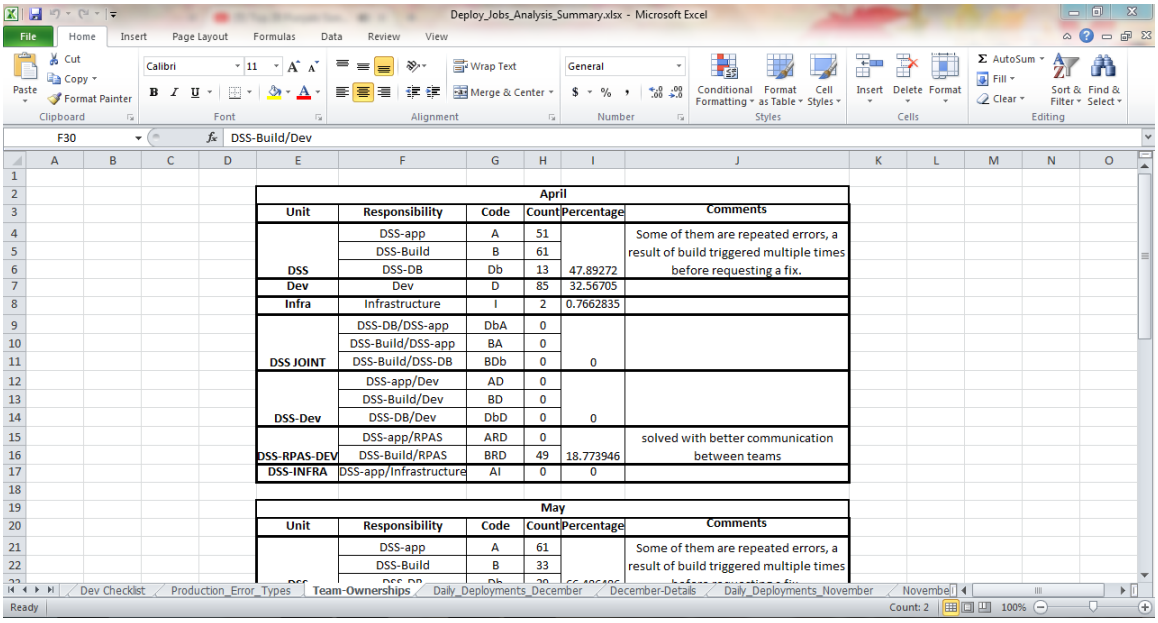


Figure 4.3: Error capturing

Team codes and Team names are classified from the past records of 1.5 years available. Teams are DSS, Dev, and DSS/Dev as broad classification. QA errors are noted down on Dev code. Also Environment creation team failure are listed on DSS's name since these teams are closely related on a broader aspect.

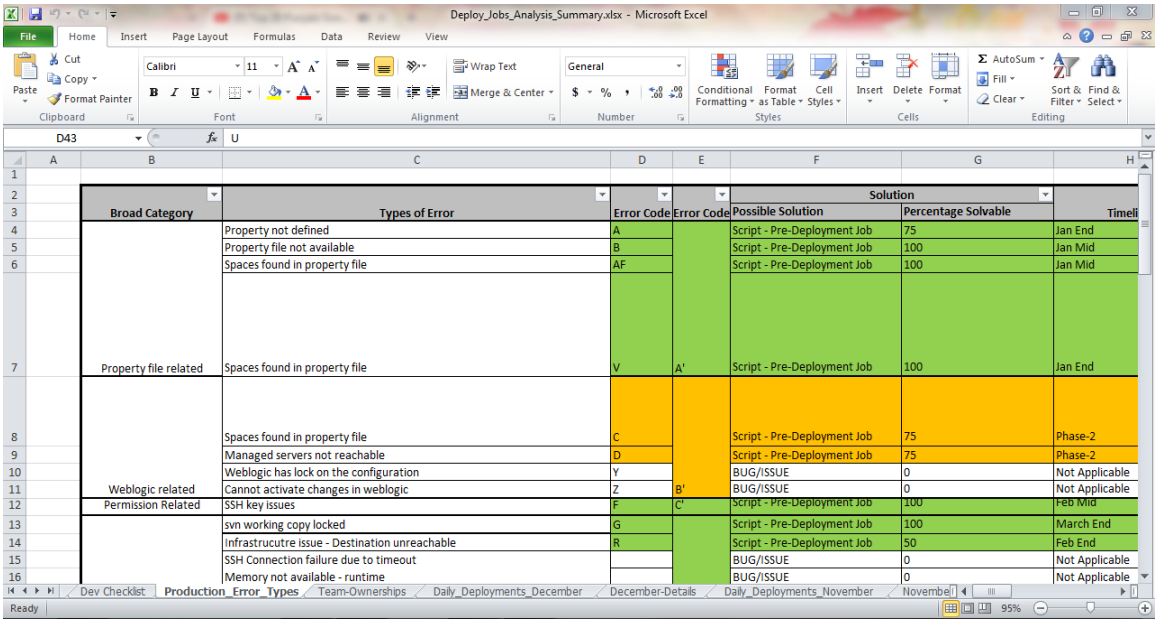


April					
Unit	Responsibility	Code	Count	Percentage	Comments
	DSS-app	A	51		Some of them are repeated errors, a result of build triggered multiple times before requesting a fix.
	DSS-Build	B	61		
DSS	DSS-DB	Db	13	47.89272	
Dev	Dev	D	85	32.56705	
Infra	Infrastructure	I	2	0.7662835	
	DSS-DB/DSS-app	DbA	0		
	DSS-Build/DSS-app	BA	0		
DSS JOINT	DSS-Build/DSS-DB	BDb	0	0	
	DSS-app/Dev	AD	0		
	DSS-Build/Dev	BD	0		
DSS-Dev	DSS-DB/Dev	DbD	0	0	
	DSS-app/RPAS	ARD	0		solved with better communication between teams
DSS-RPAS-DEV	DSS-Build/RPAS	BRD	49	18.773946	
DSS-INFRA	DSS-app/Infrastructure	AI	0	0	

May					
Unit	Responsibility	Code	Count	Percentage	Comments
	DSS-app	A	61		Some of them are repeated errors, a result of build triggered multiple times
	DSS-Build	B	33		

Figure 4.4: Team codes and names

Production error classification is done as in figure. Errors are classified according to their error types and not according to teams. Also there is a code for repeated same errors that is called as duplicate errors. Similarly since DSS handles Hudson and tools, configuring new job is done by DSS and also configuration changes are handles by DSS so if builds triggered by DSS member are classified under a code of research and development. This makes necessary to also data mine the name of user who triggered this build. This name of user can be obtained from the first line of the build log.



Broad Category	Types of Error	Error Code	Error Code	Solution: Possible Solution	Solution: Percentage Solvable	Timeli
	Property not defined	A		Script - Pre-Deployment Job	75	Jan End
	Property file not available	B		Script - Pre-Deployment Job	100	Jan Mid
	Spaces found in property file	AF		Script - Pre-Deployment Job	100	Jan Mid
Property file related	Spaces found in property file	V	A'	Script - Pre-Deployment Job	100	Jan End
	Spaces found in property file	C		Script - Pre-Deployment Job	75	Phase-2
	Managed servers not reachable	D		Script - Pre-Deployment Job	75	Phase-2
	Weblogic has lock on the configuration	Y		BUG/ISSUE	0	Not Applicable
Weblogic related	Cannot activate changes in weblogic	Z	B'	BUG/ISSUE	0	Not Applicable
Permission Related	SSH key issues	F	C'	Script - Pre-Deployment Job	100	Feb Mid
	svn working copy locked	G		Script - Pre-Deployment Job	100	March End
	Infrastrucutre issue - Destination unreachable	R		Script - Pre-Deployment Job	50	Feb End
	SSH Connection failure due to timeout			BUG/ISSUE	0	Not Applicable
	Memory not available - runtime			BUG/ISSUE	0	Not Applicable

Figure 4.5: Production Error Classification

4.3 Brief Solution on problem definition

Here is the description to how each value of column for each job will be fetched from where:

- Job Name: will be fetched from the directory name while recurring through the Hudson workspace.
- Build No. : will be fetched from directory name while recurring through each job's directory.
- Date: Date of build completed can be got but long listing the directory structure of each job.
- Environment Name: can be found at line containing words "Description set: ENV NAME" where ENV NAME is the environment name.
- Failure Log: Training set needs to be prepared for recognizing this error from the log since log might contain many error statements but we need to find the real cause for error. For this we will train a dataset with all possible errors from existing data. Also, if no error is detected which is a case when a new error is encountered, script will put a code and highlight it so that can be filled in manually also add that error to the trained set against which the logs are mined for not facing this same unavailability of error.
- Team Name: Along with this trained dataset we will add team name and team code against them.
- Production error code: This code is classified using the user triggering the build and the team code fro failure. Combination of this two values will decide the Production error code classification.

Some of these fields are filled in after clustering and classification and hold values of cluster and class IDs.

This basic fields fetched will be then clustered, classified and analyzed to generate graphs.

4.4 Solution

The end result is a simple graph depicting the ownership of failures by various teams. But to get this single simple graph it requires extensive data mining and data analysis strategies.

Heuristic approach is used for data mining error logs and their classification. There are many algorithms used for heuristic data mining like

- HILL CLIMBING
- SIMULATED ANNEALING
- TABU SEARCH
- GENETIC ALGORITHM

We will be using genetic algorithm as it provided a global optima, i.e. globally optimal solution.

Below is generic Genetic Algorithm for heuristic approach.
Result: Classification of error logs(cause of job failure)

$LetCdenoteallpossiblesolutionsandTdenoteteamsofsizeM, andx^lthel^thclassinteamsT.$

initialization;

$initializetheinitialteamsT_{C-0} = x^1_{C-0}, ..., x^M_{C-0};$

evaluate every $x^l \epsilon T_{C-0}, l = 1, ..., M, k = 1$

while *more Job exists to be looped in* **do**

instructions;

select $T^i(anintermediaterandomteamcode) fromT_{C=k-1}\ T_{C=k} <--$

$crossoverelementsinT^i\ mutateelementsinT_{C=k}\ evaluateeveryx^i \epsilon P^{G=k}_{G=0} l =$

$1, ..., M\ k = k + 1$

end

Algorithm 1: Genetic Algorithm

This algorithm is applied to classify the failure cause found from error log.

C is the clusters and T is the team to take ownership for failure and M is the number of possibilities of classification.

This algorithm applies heuristic approach to classify the failure cause found from log and it calssifies to 2 levels, one is cluster ID and one will be class ID.

Heuristic approach is advantageous as it provides a global optima since logs come from a wide range of variety of possibilities for clssification.

Since in our case, we first have 3 clusters,

- Dev
- DSS
- Anomaly

Then each of these clusters have classification of its own.

Dev:

- Syntax Error.
- Missing dependencies
- Missing properties.

DSS:

- APP
- DB
- Build and Tools

Each of DSS class has further classification of each teams relative errors.

But since each cluster has its own set of further classification, we donot need to match each class to each cluster. Each cluster will have mapping to some of specific classes as showen above.

For example, taking a bottom-up approach to understand working: If error is say missing column in a table. Here the approach of classification will work as:

- First it will search what cluster it falls in, that will be DSS cluster.
- Then it will only search in classes that are mapped to DSS cluster, i.e APP DB and Build
- Class it will match to is DB, then it will search for error code inside DB class's sub-classes.

We will discuss how training set will be created and worked upon.
Training set:

- Anomaly detection : Anomalies are basically cases that don't match any pattern or show expected behaviour. Such anomalies detected during data mining will be marked as abnormal and during classification, a special class has been assigned for such cases.

  Anomalies will be treated manually, after getting result we manually check this anomalies and try to classify them into one of the existing classes. This will update training set to detect wider range of errors from log and hence training set will be updated for more precision and accurate classification.

- Clustering : Clustering is a method of grouping data into number of clusters having similar characteristics. Each error log with have a specific error line that will fall into one of the clusters.

  There is a different cluster and class for anomalies.

- Classification : Different errors are classified into different classes according to the reason of failure and ownership of the code that caused failure

  Codes will be owed by different teams like Dev, DSS-Applications, DSS-Database, DSS-Build and deploy teams.

- Summarization : It is a process of representing data into a more compact and re-presentable form that gives a overall idea of categorization in one look.

  Pictorial summarization is the bet format for this as it is easiest manner to judge a scenario. Many formats are available for this presentation like, graphs, charts, etc.

- Make Decision tree(CHAID algorithm)

  This is a rough diagram of decision tree for this project

CHAID Algorithm and its extended version.

- Getting ready predictors. The initial step is to make clear cut predictors out of any persistent predictors by separating the individual constant dissemination into various classifications with a roughly break even with number of perceptions. For clear cut predictors, the classifications (classes) are "normally" characterized.

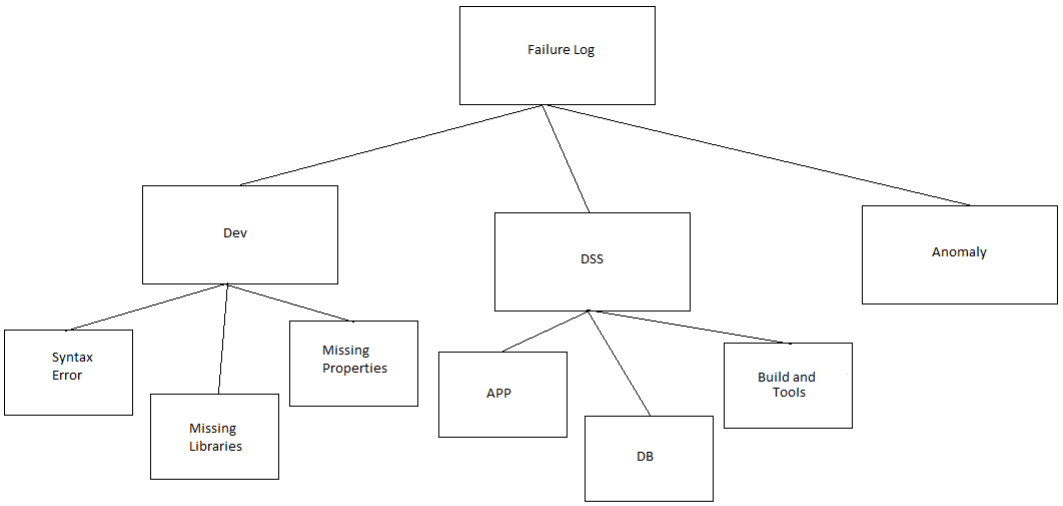


Figure 4.6: Decision Tree

- Merging classifications. The following stage is to spin through the predictors to decide for every predictor the match of (predictor) classifications that is slightest altogether extraordinary regarding the dependent variable; for order issues (where the dependent variable is categorical also), it will figure a Chi-square test (Pearson Chi-square); for regression issues (where the dependent variable is in continuity), F tests.

  In the event that the particular test for a given combine of predictor classifications isn't measurably noteworthy as characterized by an alpha-to-blend esteem, at that point it will consolidate the individual predictor classifications and rehash this progression (i.e., locate the following pair of classes, which now may incorporate already combined classes).

  In the event that the measurable criticalness for the particular combine of indicator classifications is huge (not as much as the individual alpha-to-blend esteem), at that point (alternatively) it will figure a Bonferroni balanced p-esteem for the arrangement of classifications for the separate indicator.

- Choosing the split variable. The subsequent stage is to pick the split the predictor variable with the littlest balanced p-esteem, i.e., the predictor variable that will yield the most huge split; if the littlest (Bonferroni) balanced p-esteem for any

indicator is more noteworthy than some alpha-to-part esteem, at that point no further parts will be performed, and the individual hub is a terminal hub.

Proceed with this procedure until the point when no further parts can be performed (given the alpha-to-consolidation and alpha-to-part esteems).

- CHAID and Exhaustive CHAID Algorithms. A change to the fundamental CHAID calculation, called Exhaustive CHAID, plays out a more intensive consolidating and testing of predictor factors, and henceforth requires all the more figuring time.

  In particular, the converging of classes proceeds (without reference to any alpha-to-consolidate esteem) until the point that lone two classifications stay for every predictor.

  The calculation at that point continues as depicted above in the Selecting the split variable advance, and chooses among the indicators the one that yields the most huge split. For vast datasets, and with numerous ceaseless indicator factors, this change of the less complex CHAID calculation may require huge figuring time.

4.5 Training undertaken for Project

- Oracle Database 12c
- Oracle Weblogic Server 12c
- Oracle Fusion Middleware 12c
- Shell Scripting Basics
- Python: Basic Programming
- Python: Data Mining smart Approaches

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This project accomplishes requirement of automating reporting and capturing of success and failure rates of jobs ran on Hudson that includes build jobs and deploy jobs. Execution of scripts results into a pictorial representation of analysis of data obtained from thousands of logs.

Along with this, we obtain a excel sheet with these data for precise and detailed observation of failure cause and reports of jobs triggered.

5.2 Future Work

This job is to be used as a test job to be moved on Jenkins. If successful, team will try to write a yaml script to configure job automatically on Jenkins continuous integration tool. If successful, more yaml scripts will be written for all 2000+ jobs present on Hudson to be migrated on Jenkins.

Here, we will also try to research on barriers encountered during migration from Hudson to Jenkins due to drastic architecture change between Hudson and Jenkins.

Bibliography

[1] "http://www.oracle.com/technetwork/middleware/id-mgmt/overview/index-082035.html."

[2] "http://www.tutorialspoint.com/obiee/."

[3] A. Shah, "Automation of software production line through continuous code, build, deployment and test integration for rpas products," 2014.

[4] "http://wiki.eclipse.org/hudson-ci/using$_h$udson/installing$_h$udsonwindowsinstallation."

[5] "https://wiki.jenkins.io/display/jenkins/step+by+step+guide+to+set+up+master+and+slave+mach

[6] "https:/www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys-2."

[7] "https://pandas.pydata.org/."