**CHAPTER-5**

**SYSTEM DESIGN**

System design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

**5.1 SYSTEM ARCHITECTURE**

**5.1.1 AUTOMATED DEPLOYMENT AND MONITORING**

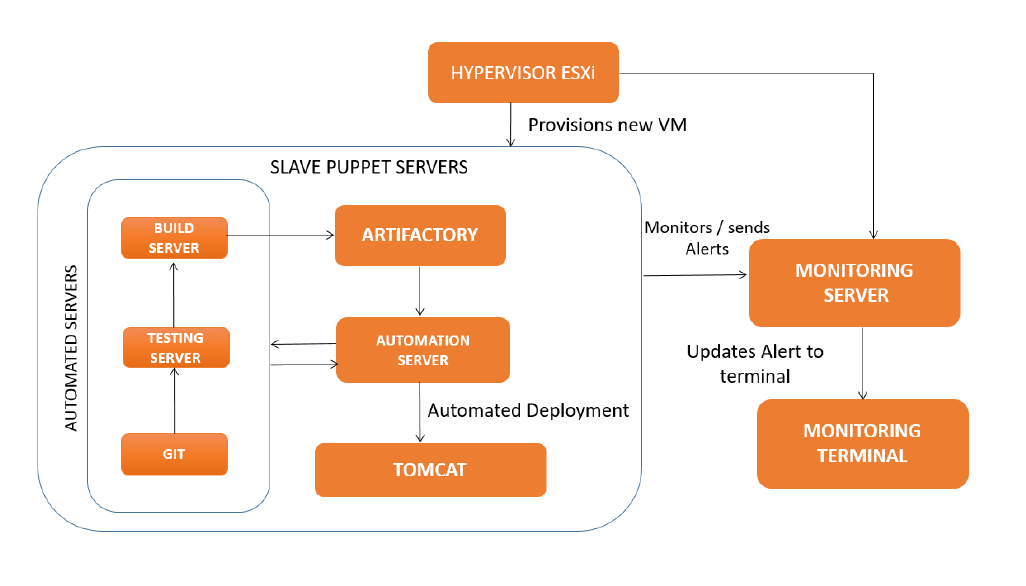


Figure 5.1: AUTOMATED DEPLOYMENT AND MONITORING

**5.1.2 DESCRIPTION OF ARCHITECTURE:**

Figure 5.1 depicts the architecture of the system and particularly concerned with the aspects of system monitoring and automated deployment.

The code is taken from the Git labs server which contains the bulk of the application code and it is tested using various testing tools. It is then built using build tools like Maven and Jenkins and it is forwarded and stored in the artifactory after the build process is completed.

**ARTIFACTORY**: It is a repository that contains all the packages needed to run the application. It contains the war and jar packages that are needed to run the system. Any number of packages can be stored in an artifactory and all the packages can be integrated and deployed as one application. The packages stored in the artifactory are taken and deployed automatically by the automated deployment server.

**AUTOMATION SERVER**: It is an automation server. It uses a tool called puppet for the purpose of automation of the entire system. The packages that are stored in the artifactory are deployed automatically into the live environment using puppet. It uses the puppet scripts to automate all the tools and services that are running in the entire system. This automation server can automate the installation and configuration of any tool or service on any number of hosts. A large number of services can be configured simultaneously on a large number of hosts with zero human intervention.

Once the slave host initiates the pull request from the manifests, all the configuration files of that particular node starts running and automatically calls all the module files that are stored in the master puppet server. These module files contain the script for the installation of all the services. We can split the modules files based on which service is to be installed on which server and accordingly can specify the list of module files under that particular node in the site.pp file.

**MONITORING SERVER**: The monitoring server monitors all the components of the system and keeps track of the system status and uptime and downtime. It monitors the behaviour of the components based on certain rules and an alert is sent to the monitoring terminal wherever an issue arises. The alert is addressed by . A tool called Nagios is used for monitoring the entire system. It is an open source tool that allows the administrator to write custom scripts to monitor specific services.

**MONITORING TERMINAL**: The monitoring terminal is the console from where it is possible to monitor all the hosts, tools and services that are configured in the system. The terminal is part of the Nagios master server and displays the state and run time messages of all the services and tools in the proposed system.

**HYPERVISOR**: The Hypervisor creates a new instance of a machine. The hypervisor is used to create virtual machines in the system. Based on the requirements, a machine is created. It can either be a linux based or a windows based machine. In the proposed system most of the machines are linux based except for one machine. One machine is windows based which is used for installation of testing tools like LeanFT, UFT and monitoring tool AGM.

This was part of the requirements from the testing team. The operations team executive creates these virtual machine using the EXSi hypervisor and provides the platform to install, configure and run all other services.

The beauty of these tools involved in this system is that they can be integrated together and automated to do complex tasks in a short span of time.

The puppet scripts that are written will take care of all the automation operations needed to run the system in an automated way. The scripts that are written for monitoring system will take care of all the components in the system and will send alerts about any changes being made in the system.

**5.2 Data Flow Diagram:**

A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes and how it gets stored.

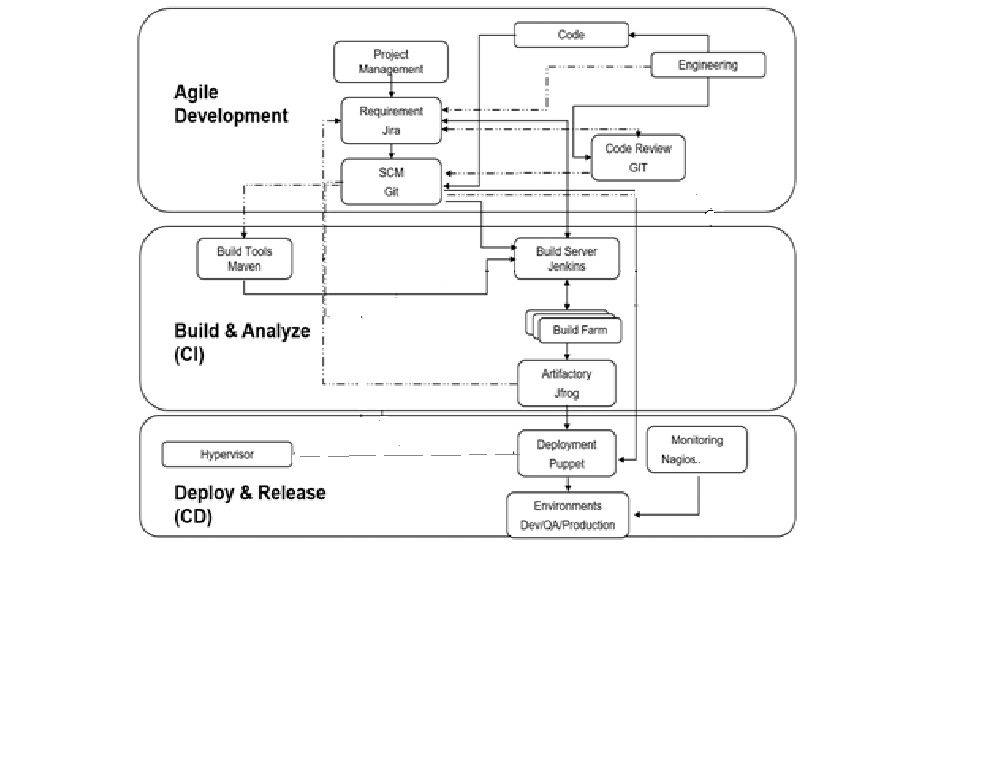
****

Figure 5.2 Data Flow Diagram.

Figure 5.2 will Depicts the Dataflow diagram of our projects.

**5.3 USE-CASE DIAGRAM**

The explanation behind use-case diagram is to get the dynamic part of a system. In any case, this definition is unreasonably non particular, making it difficult to delineate the reason. In a matter of seconds when the basic task is done use case layouts are shown to demonstrate the outside point of view. The inspirations driving use case charts can be according to the accompanying:

* Used to amass essentials of a structure.
* Used to get an outside point of view of a structure.
* Recognize external and inside components affecting the system.
* Exhibit the interfacing among the essentials are performing craftsmen.

The Use case diagram of the DevOps office model is given in Figure 5.3

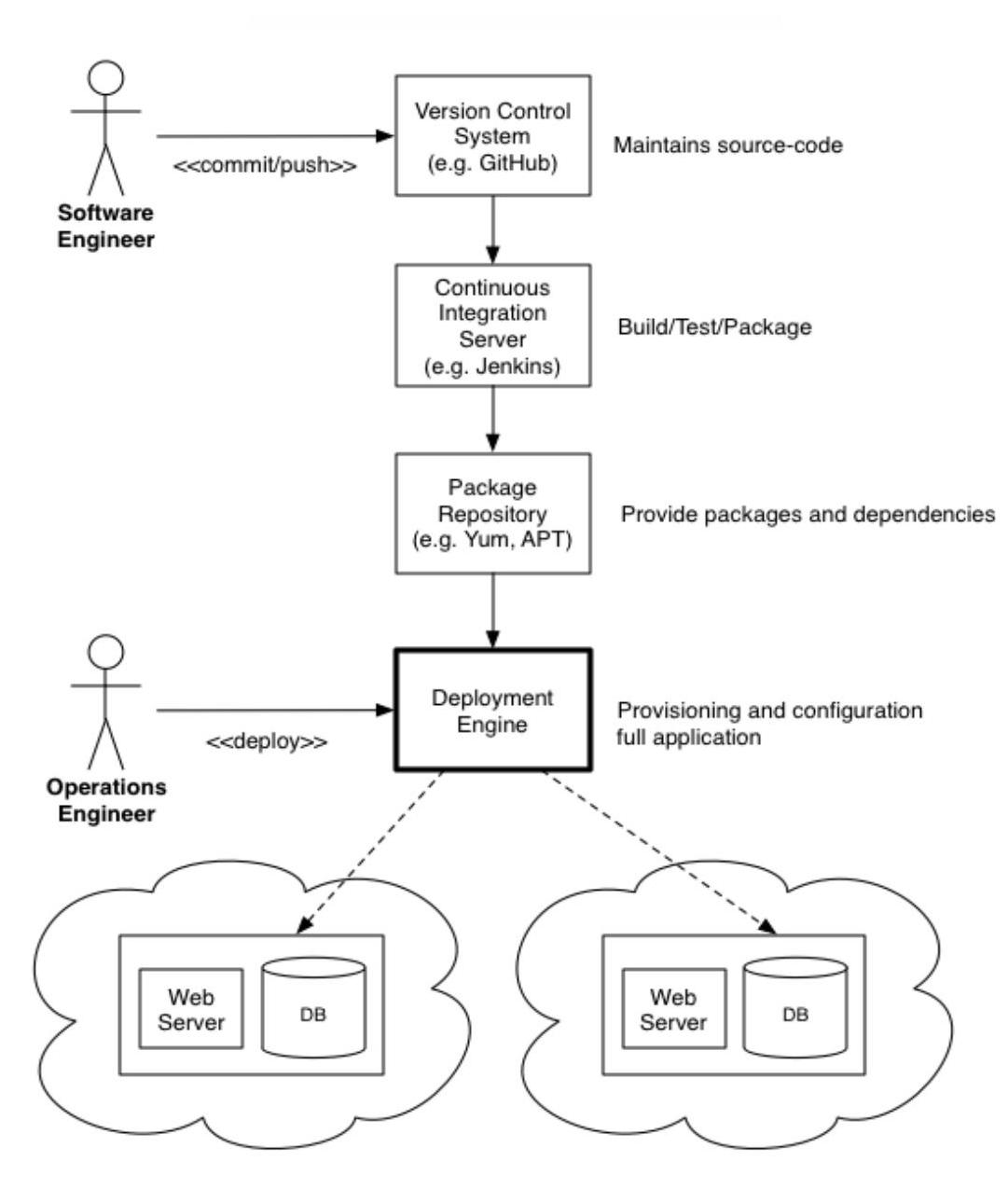


Figure 5.3: Devops Office Model Usecase Diagram

**5.4 SEQUENCE DIAGRAM**

This sections show the sequence diagram for the DevOps Office Model in Figure 4.1 where the sequence flows like the code is being pulled from the Git repository. Then it is build using the maven and pushed to git and the application is being tested with SonarQube and these tools are integrated using the integration server called Jenkins.

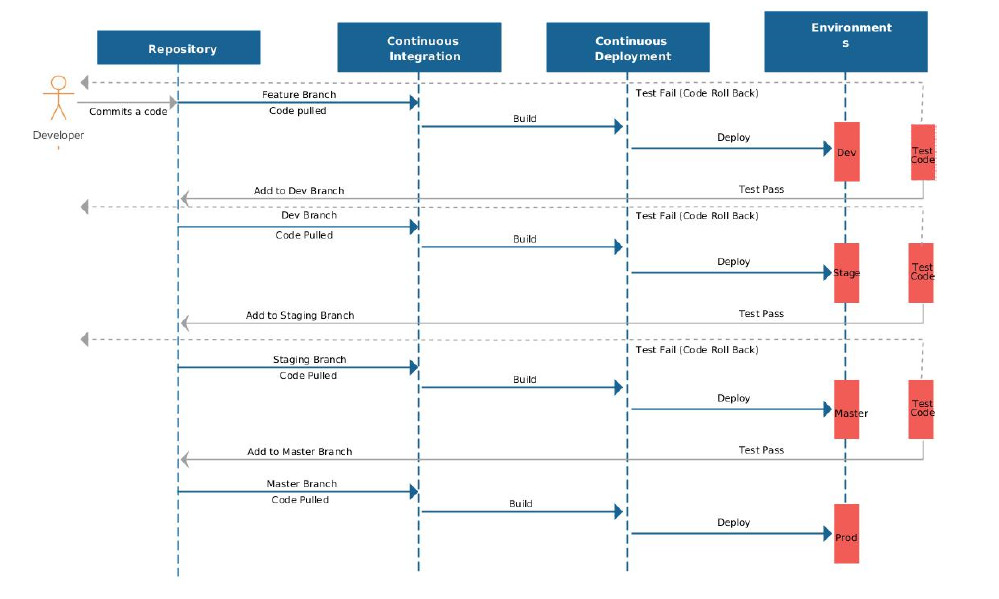


Figure 5.4: Sequence Diagram-Dev ops office Model

**5.5 ALGORITHMS**

The pseudocode for the nagios plugins is written here in this chapter.

**5.5.1 ALGORITHM TO MONITOR USED SPACE**

Input: Command to view memory statistics of host

Ouput: The amount of memory used and status of host

1: GET the memory details

2: SELECT filesystem

3: SELECT value of column5

SET column5 to spaceused

SET limit1 to 75

SET limit2 to 85

4: if spaceused less than limit1 then

5: spaceused used is under limit OK

elseif

6: spaceused greater than limit1 and spaceused less than limit2

7: memory needs to be monitored warning

8: else

9: memory out of bound CRITICAL

10: end if

The above algorithm is used to calculate the amount of memory space used. It takes the input from the command df -h. This command will return the statistics of all the memory space. Then from the over all statistics the file system is selected. From that column 5 which contains the used space is selected. Then it is compared with the limit levels. It returns three statuses. Ok, if the space is within limit. Warning, if the limit is between minimum and maximum limit, Critical, if it has exceeded the maximum limit.

**5.5.2 ALGORITHM TO MONITOR NO OF PROCESSES**

Input: List of active processes

Output: Count of active processes and status

1: GET the process list psaux

2: SELECT wordcount

3: SET wordcount to a

4: PRINT a SET lowerlimit to 500

SET middlelimit to 600

5: if a less than lowerlimit then

6: process count is under limit OK

return 0

elseif

7: a greater than lowerlimit and a less than middlelimit

8: process count needs to be monitored warning

9: return 1

10: else

11: process count out of bound CRITICAL

12: return 2

13: end if

In the above algorithm, the number of running processes are taken as input and displayed. Based on a certain threshold limit, specific warning are returned to the terminal. If the limit is below 500,it is ok. If it is between 500 and 600 then it is in warning stage. If it is above 600, it is in warning stage.

The following pseudocode has the same design because the scripts follow the same template. These services can be monitored in the same way.

**5.5.3 ALGORITHM TO MONITOR PUPPET MASTER**

In the below algorithm, the service query of puppet master is done.

If it returns a value as inactive then the service is down and an exit code 2 is returned to the terminal. If the service returns an active status then the service is running and it returns an exit code 0 to the terminal.

Input: Service query of puppet master

Output: Service status of puppet master

1: GET servicestatus

2: SELECT inactive

3: SET servicestatus to a

4: if a == true then

5: master server is not running CRITICAL

return 2

6: else

7: master is running OK

return 0

8: end if

**5.5.4 ALGORITHM TO MONITOR JENKINS**

In the below algorithm, the service query of jenkins master is done.

If it returns a value as inactive then the service is down and an exit code 2 is returned to the terminal. If the service returns an active status then the service is running and it returns an exit code 0 to the terminal.

Input: Service query of jenkins master

Output: Service status of Jenkins master

1: GET servicestatus

2: SELECT inactive

3: SET servicestatus to a

4: if a == true then

5: master server is not running CRITICAL

return 2

6: else

7: master is running OK

return 0

8: end if

**5.5.5 ALGORITHM TO MONITOR XINETD**

In the below algorithm, the service query of xinetd service is done.

If it returns a value as inactive then the service is down and an exit code 2 is returned to the terminal. If the service returns an active status then the service is running and it returns an exit code 0 to the terminal.

Input: Service query of xinetd service

Output: Service status of xinetd service

1: GET servicestatus

2: SELECT inactive

3: SET servicestatus to a

4: if a == true then

5: master server is not running CRITICAL

return 2

6: else

7: master is running OK

return 0

8: end if

**5.5.6 ALGORITHM TO AUTOMATE JAVA INSTALLATION**

In this algorithm, the java tar file is being extracted and stored in a file. It is then moved to a different directory where all the softwares are stored.

A directory is created to hold the jdk file. The file’s permission is set to root.This means that only the root user can access it. Any user apart from the root user cannot access the file. Then the file is executed using the exec command it is moved into the java path in the bin directory. The java compiler path is set to the javac file. All the files are set to their appropriate file locations from where java will execute effectively. The path set in the .bashrc file will make sure that all the services access java from the correct file locations if there are multiple copies or versions of java. It is possible to have multiple versions of java in the same machine under different users. Hence setting the path in the .bashrc file is very important so that there is no anomaly when others services that require java are executed.

Input: Jdk file

Output: Java installed required host

1: GET jdk f ile

2: SET jdk to f ile

3: SET owner to root

SET pathtoexecute

4: GET f ile

EXECUTE tar f ile

MOVE f ile to =opt

SET :bashrc to pathtoexecute

SET java path to =opt/ file/ java

SET compiler path to =opt/ file/ javac