**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 EXISTING SYSTEM**

In traditional approach, a development team develops features and passes them on to a separate operations team may not address the needs of the frequent release scenario. The delays involved in acknowledging, testing and deploying the applications in the traditional manner increase the time to deliver the feature.

CONVENTIONAL SOFTWARE DEVELOPMENT, the existing system uses traditional Manual Installation methods and Deployment methods that create a lag in time, cost and efforts. The use of waterfall model was not very efficient in the software development process since at each stage the modules were fixed and any changes needed took a lot of time and resources and hence was very inefficient in many ways. In a similar manner the process of installations being done in a manual manner results in the same aspect of water fall model with a lot of time being spent on installations, upgrades and changes in the system, all being done manually. This does not help any organisation since the customer is always expecting immediate results and is ready to pay any amount to the organisation who can deliver that kind of results in a sort span of time.

In the existing system all the components need to be man handled and installed. If the downloading is done using source installation method then the system will have to search for the specific packages and download them and then we have to follow the steps and execute the steps of installations. And if this is done in a distributed environment then it gets very difficult since the number of machines is huge and accomplishing the installation on each and every machine is a very tedious and time consuming task. Physically carrying out the process in a big network involves the risk of human error. If a person handling the system performs a wrong step then there is a risk of the whole system crashing. This happens especially in the process of integration of various components in the system. In that case the process of installation for the entire system should be repeated and this will result in wastage of time and resources for the organisation.

The major difference occurs in the testing phase. Testing usually takes a lot of time and resources and it usually delays or extends the project completion date. If there is a system that can achieve the tasks of manual testing then a lot of time can be saved and we can complete the project at a very rapid pace. Thus the manuals operations related to the software delivery process need to shift to a new era of automation that will solve the problem of efficiency and consistency in regards to time, cost and resources.

**2.2 LIMITATIONS OF EXISTING SYSTEM**

* The existing system can work only within a certain scope and it cannot be extended to systems needing rapid change and modification.
* The existing system requires constant human interaction and resources
* In this system the administration of a distributed system is difficult since it will take a lot of time to manually attend each and every component across a distributed system that is very large in size.
* Existing system does not encourage constant change in requirements.
* The management of the system is mainly too time consuming.
* In the existing system addition of new components is an overhead since a lot of configuration changes need to be made.

**2.3 PROPOSED SYSTEM**

The proposed system is used to automate all the manual installations of the tools for the release. Automating the process of software building, deployment, delivery and maintenance are some of the goals of the proposed system. It also aims at automatic testing of the components and code that is used to create the application that will be running in the live environment. Automated monitoring and dash boarding of quality and performance against service level agreements at multiple stages is also one of the objectives of the system. By using advanced monitoring tools, high availability of the services can be achieved. Automate hand-offs/provisions to increase the velocity. It will automate all the manual installations for the release and deployment of the software. It will also automate tests that run fast and have good coverage on code. The system includes various devops tools that serve the purpose of the software production and deployment.

In this DevOps environment, on the other hand, the entire team is responsible for delivering both new features and stability. The combination of a shared code base, continuous integration, test-driven techniques and automated deploys, among other things, exposes problems in application code, infrastructure, or configuration earlier in the process, because the code is not thrown over the wall to operations at the end of coding. Problems tend to be less complex because change sets are smaller. DevOps engineers exploit real-time data into the performance of their systems to quickly understand the impact of application changes. And resolution times are faster because team members do not need to wait for a different team to troubleshoot and fix the problem.

The system requires a lot of installations and configurations of all the tools and services that will be used in the software project. The system will automate all the manual installations for the release and deployment of the software. There will be the usage of various DEVOPS tools like Git (Repository), Jfrog (Binary repository), Jenkins, Jira, Maven, Cobertura, Nagios (Monitoring tool). These tools cover the aspects of development, storage, integration, project management, system monitoring and product testing.

As far as automation is concerned integration of all these tools is done manually and then a tool called Puppet serves the purpose of automation of the entire system. Puppet is used to provide all the automation of all the tools and services that are running in the entire system and make them run in an order and correct manner just with the click of a single button.

Since the aim of any company is to make sure that the product or service is stable and available all the time we need to create a mechanism that take care of the concept of high availability. High availability means that the service is available to the customer at all times and to any number of customers.

Due to an overload the service should not go down and should manage the load and cater to all the customer who are requesting for the service.

To achieve high availability nagios is used that will monitor the system at all times. In order to make sure that a service crash is reported properly, custom scripts are written in Nagios to monitor each and every service running on the system. All the alerts are reported to the administrator in the form of alerts to the terminal or mails to the inbox of the administrator. This makes sure that any service crash doesn’t go unnoticed and all the services are tracked at all times.

**2.4 DRAWBACKS OVERCOME BY PROPOSED SYSTEM**

* The proposed system can work in a wide range and it can be extended to systems that need rapid changes and modifications.
* The proposed system requires only very minimal human interaction
* The proposed system enables the easy and efficient administration of a distributed system since there is minimal manual intervention in the system.
* The proposed system can incorporate constant change in requirements.
* The management of the system is easy and time efficient.
* The proposed system can incorporate addition of new components and a lot of configuration changes can be made.

**2.5 TOOLS USED**

* Nagios : It is open source monitoring tool which monitors sites.
* Git : It is open source version control system
* Github : Public open source Repository which is used store codes
* Maven : It is building tool which build the codes into binary files or packages
* Firewall : used to implement persistent network traffic rules.
* DNS : provides visitors access to websites using domain names rather than IP addresses.
* Jenkins : Continuous Integration tool which is used to integrate the all other tools.
* Jfrog : It's a single gateway through which you access external artifacts, and store your own build artifacts.
* Corbertura : is a free Java tool that calculates the percentage of code accessed by tests. It can be used to identify which parts of your Java program are lacking test coverage.
* Puppet : is an open-source software configuration management tool. It runs on many Unix-like systems as well as on Microsoft Windows, and includes its own declarative language to describe system configuration.