CHAPTER 2

LITERATURE SURVEY

**LITERATURE REVIEW**

**2.1 EXISTING SYSTEM**

The traditional approach where a development team develops a feature and passes it 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.

In existing system they are still using tradition 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 man handled and installed. If we are downloading the source from internet then we 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 we are doing this 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. We will have to physically carry out the process in a big network and there is a 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 we will have to repeat the process of installation for the entire system yet again 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 that need rapid changes and modifications
* 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 the 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 systemis 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 we can achieve high availability of the services that we render. Automate hand-offs/provisions to increase 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. We will be using various devops tools that serve the purpose of software production and deployment. The following are the objectives of the project that will help us in accomplishing the automation of delivery process.

Devops helps people in research, design, sales, and production. Everybody must work as a team in order to solve problems of production and usage that may be encountered with the product or service. It is important to maintain a single source repository to achieve stability. In this system everyone commits to the mainline every day. Every commit should build the mainline on an Integration machine.

In a non-DevOps environment, there is often tension between releasing new features and stability. The development team is measured on the updates they deliver to users while the operations team is measured on the health of the system.

In a 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 isn’t “thrown over the wall” to operations at the end of coding. Problems tend to be less complex because change sets are smaller. DevOps engineers can 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 don’t need to wait for a different team to troubleshoot and fix the problem.

The key advantage for adopting DevOps in the current business environment is business agility. As the rate of change for business accelerates, companies are less able to predict where business is heading. The top strategic imperative becomes responding rapidly via agility and modularity through DevOps and adaptive IT. When you don't know what the future holds, it's important to have the infrastructure in place that can respond to business needs on the fly. When you use the DevOps agile methodology, IT works directly with the business users and delivers exactly what they need – and nothing more. By focusing on business needs first we see that projects get finished and move into production fast.

Ultimately DevOps allows companies to deliver software that is mission critical to production much faster – sometimes 10X to 100X faster. For any business that captures advantage through any type of software, this kind of improvement could mean success or failure. In the future, when more companies are successfully deploying DevOps, there will no longer be any room for failure to adopt DevOps. While there are other advantages to DevOps, a 10x to 100x speed advantage is the only one that counts.

If DevOps is adopted properly by the organization, IT's role changes from being a business cost centre to being that of one that leads the innovation for the business. Product and service offerings can be introduced, tested, and rolled out much more quickly to the market place keeping up with the pace of change in the market place. The principles of DevOps allow IT teams to produce and deliver value quickly, test out hypotheses with real users, and roll out actual services and products that bring in revenue to help grow the business and make an impact to the top line

**2.4 OBJECTIVES OF THE PROJECT**

* Installation and setup of DevOps tools
* Integration of multiple tools within the DevOps chain
* Deployment Automation of the DevOps tools
* Automation of the bare metal provisioning and management

The project aims at automation of the processes involving software testing, building, deployment and monitoring. We will be initially carrying out all the installations and setup of multiple DevOps tools that range along different aspects such as testing tools, build tools, configuration tools, deployment tools, packaging tools, provisioning tools and monitoring tools. All the installations are done manually in the beginning.

Then we move into the integration phase where we will be integrating multiple tools within the DevOps chain using a tool called Jenkins. We will be integrating a lot of tools across different types like development, software testing, building, packaging, deployment, delivery and monitoring tools. Each tools is different and needs to be integrated in a different manner based on the requirements of the tool and the purpose of the tool.

The Deployment Automation of the DevOps tools is carried out by using a tool called Puppet. This tool automates the installation of multiple tools and sets up the system automatically within seconds for any live environment. It has a master and a slave puppet that manage the server client processes between the puppet master and all the slave puppets configured on all the multiple DevOps tools in the system.

We use a tool called razor for the Automation of the bare metal provisioning and management of the servers. This tool automates the provisioning of servers (nodes) to the services. It manages them and in case of a node crash, it automatically identifies any reserve node that is capable of handling the service that crashed. It then migrates the service onto that reserve node and starts the service within seconds of the original crash.

The puppet master controls the installation and configuration of all the nodes containing all the different types of tools. Each node will be having its own setup and OS requirements. Everything is described in the puppet master using a declarative language which is own of its kind to the puppet and it is used to describe the system configurations of all the nodes in the system. The puppet slaves are configured in all the existing nodes of the system and they get the instructions from the master and configure the tools or the services running on that particular node accordingly.

All the details related to a service crash is stored in a database along with the details of the system that went down. The razor will check for updates in the database and as soon as an update comes in the database it will fetch the details of the crash. It will compare the details of the crashed system using the template that is already stored in the razor and it will forward the orders to an EXSi hypervisor to create another machine and migrate all the operations of the crashed system onto the new machine. All these operations take place within seconds of the crash and hence the aspect of high availability is guarded here through the help of automated bare metal provisioning.