International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2454-9150 Vol-05, Issue-02, May 2019

Setting Up CICD Pipeline For Web Development Project in Cloud

Sushmitha M S, Student, PES College of Engineering,Mandya,India sushmithams95@gmail.com

Mr. Sanjay H M, Assistant Professor, PES College of Engineering, Mandya, India , sanjaypesce@gmail.com

Abstract Few years back when agile methodology was playing a major role in the industry, software was deployed in monthly, quarterly or annual basis which was time consuming. But now it's DevOps era! Where software can be deployed multiple times a day. In current era, delivering creative ideas in a rapid and steady manner is eminently significant for all organizations. In addition to that, organizations need to react to vigorous market requirements, faster time to market, decrease in failure rate and increase in customer interaction. This could be achieved with the help of DevOps methodology. DevOps methodology extends the agile to quickly produce software and automatically deploy them across various platforms/environment in order to gain high performance and quality assurance products. Continuous integration/Continuous deployment (CI/CD) is the backbone of DevOps environment. By automating the build, testing and deployment of software, CI/CD bridges the gap between development and operation teams. Git, Maven, Jenkins, Terraform, Docker, Kuberentes are the DevOps tools used in order to automate the entire environment.

Keywords — Agile, CICD, DevOps, Docker, Git, Jenkins, Kuberenetes, Maven, Terraform.

I. INTRODUCTION

Due to increasing competition in software industry, organizations play a major in assigning required resources to develop and deliver trustworthy and high quality products to consumers. Consumers expect to have continuous interaction with DevOps team so that they can provide their continuous feedback. DevOps is blending of two terms development and operations which aims to provide conjoin approach to industry's software development and operation team job in software development lifecycle. It provides a good communication between these two teams. DevOps describes the conformation of automation and programmable software development and infrastructure deployment and maintenance. Continuous integration, continuous deployment and continuous delivery are the important factors in software industry that helps organizations to constantly release new attributes and products that are trustworthy. Continuous integration focuses on integrating each developers work multiple times per day so that debugging of error is easy. Continuous delivery focuses on demoting discordance in deployment or release process and automating the build step so that code can be released securely at any time. CI/CD pipeline provides following benefits in software delivery lifecycle: obtaining rapid feedback from customers, rapid and steady release leads to have customer satisfaction and quality assured product, CD helps to automate tasks which was carried out manually.

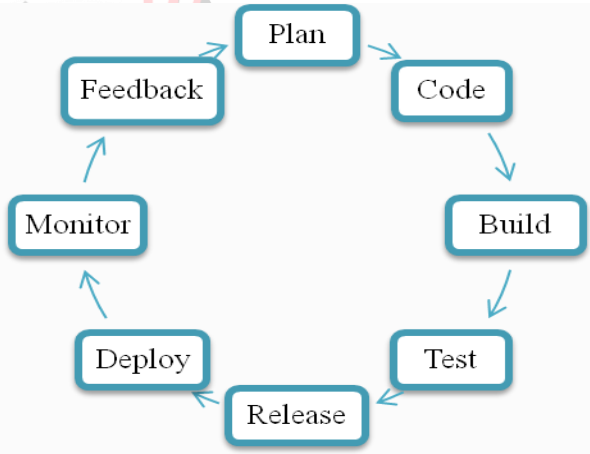

Figure 1: DevOps lifecycle

In our work, developers push their code to common repository every time they write the code and hence other developer in different location can access the code to know about current status of the project. Now the code in common repository should be built. The task of the building tool is to download the corresponding dependencies for the code and convert it into package. Hence there is no need for the testing team to download the dependencies again. If any error occurs, they are sent back to the developer. This process is manual. In order to automate the entire process, continuous integration process can be used. CI process frequently monitors the common repository to check for new arrival of code. If any new code arrives, build it. This entire process contributes to continuous integration. After

 DOI : 10.35291/2454-9150.2019.0091

building the code, it should deploy to a web server hosted with container technology. This process contributes to continuous deployment.

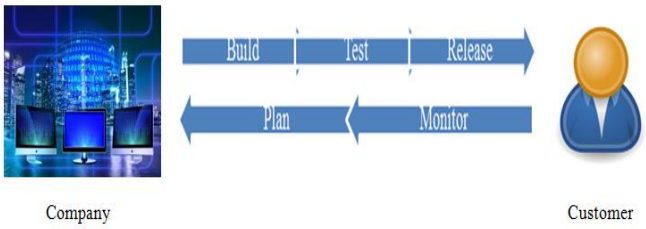

Figure 2: DevOps Working Overview

II. SURVEY SUMMARY

Project development consists of three phases like Development, testing and Production. Initially the software industry has adopted Waterfall methodology. In this methodology, if the project has to be completed within 12 months, the time required to complete the project is divided among three phases: say development phase requires six months and testing and production phase requires three months each. During six months of development phase, the developer writes the entire code of the project. After development of code, the code is then transferred to testing phase. In testing phase, the entire code is verified for error. If any error occurs, testing team send the error message to the development team. Now the developer starts to debug the error and again send the altered code to testing team. Again the testing team should verify the entire code for error. Thus it is a time consuming process. Hence the project cannot be completed in specified time, say 12 months. During these 12 months, many new technologies will emerge and they are unable to reach those technologies. Hence adaptation of new technologies in middle of the project is not possible. Also there is a lack of communication between developer and other teams.

To overcome the above challenges, industry started to adopt agile methodology. Here the project is divided into modules called sprints. It supports adaptation of new technologies. In this methodology, development and testing phases are collaborated. The code is written in each sprint and after completion of each sprint, it is given to testing phase. Thus it is easy to identify the error in single sprint and also fixing of error is easy and time consuming. After fixing all the errors, the code is then deployed to production phase. When compared to waterfall model, agile methodology is speedy. There are some drawbacks of this methodology. They are (i) it fails to deliver the product on deadline (ii) project getting deviated due to lack of documentation during discussions (iii) short time period to complete a task.

Hence these are the limitations of previous methodology which was followed by the software industry. To overcome these limitations, DevOps methodology era started.

III. PROPOSED METHODOLOGY

Devops is a software development procedure that combines both software development and software operation teams together. DevOps methodology is explained below using various DevOps tools. In this work, source code repository used is Git, building tool is Maven, CI tool is Jenkins, and CD tool is Docker and Kubernetes.

A. System Architecture

The below figure gives an idea about system architecture:

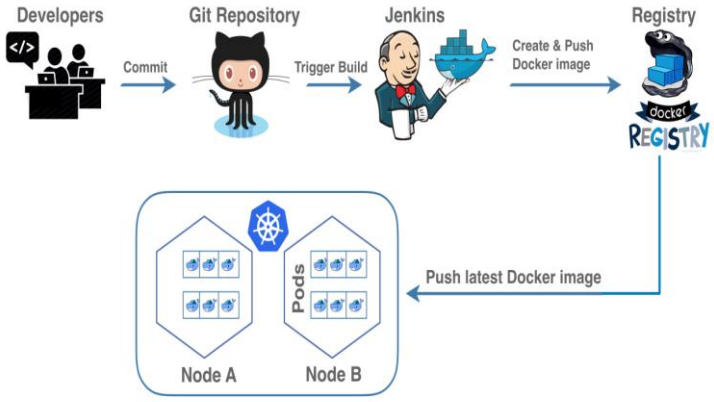

Figure 3: System Architecture

1. Different developer from different locations pushes their code from their local repository to remote repository called Github.
2. The code is then build using Maven resulting in creation of package called jar or war file.
3. This whole process is automated using Jenkins. Job is created in Jenkins and task of that job is to monitor Github for new code and build new code into package.
4. This jar file is deployed in common space say S3 with the help of Jenkins.
5. At this point, testing team checks for error (in Jenkins), if any error occurs, an error message occurs. Development team debug the error and again pushes their code to github.
6. Before deploying the jar file, testing team authenticate the production environment for quality assurance.
7. The production environment is set up using terraform tool by creating vpc, subnets, route table, and internet gateway.
8. After authentication, jar file is deployed in servers.
9. The place where we deploy the code could be either virtual machine or docker images. Here we use docker images. Containers are created in docker. Docker is used in order to overcome environmental change problem. Requirements are placed in images and these images can be transferred to all the phases.
10. Docker may contain ‘n’ number of containers which is called as micro service. In order to manage ‘n’ number of containers, a tool called Kubernetes is used.

 DOI : 10.35291/2454-9150.2019.0091

Kubernetes is an orchestration tool used to have co-ordination among containers. Kuberentes is also used for automating deployment, managing containerized application and scaling of resources.

B. *DevOps Tools*

1. Terraform

   Terraform helps to build, change and version infrastructure safely and efficiently. It is designed to support and manage the lifecycle of a vast resources, physical servers and SaaS products. It focuses on the automation of the infrastructure itself. It is also used to avoid code duplication. Install terraform using commands given in the link: https://medium.com/@anusha.sharma3010/terraform-installation-on-ec2-ubuntu-instance-10ccc023bca8

2. Git

   Git is a version control tool used to push code into remote repository i.e., Github.com during software development lifecycle. It is also used to monitor changes in file sets. Developers push their code to repository created in Github.com using git commands. Initially install git in the server using sudo apt-get install git command.

3. Maven

   Maven is project management and comprehension tool which provides complete build lifecycle framework for developers. Maven is based on Project Object Tool (POM) file. POM is used for project builds, dependency and documentation. POM is a XML file that is present in the base directory of project as pom.xml. POM file contains all the necessary information and configuration details of the project. Install maven using commands given in the link: https://qiita.com/dewaken/items/3f10c245b6780ad473bf

4. Jenkins

   Continuous integration (CI) process is carried out using Jenkins tool. Jenkins is an open source automation server helps to automate manual work of software development lifecycle. Install Jenkins using the commands given in the link: https://medium.com/@itsmattburgess/installing-jenkins-on-amazon-linux-16aaa02c369c

5. Docker

   Docker is a containerization platform that is used to create a package containing an application and all its dependencies altogether in the form of a docker container to make sure that the application works perfectly in all environments. Docker Container is a standardized unit which is created on the fly to deploy a specific application or environment. Consider a scenario where code running in one machine is not running in another machine. This is due to environmental change. To overcome this problem, Docker is used. Docker image is created. Install Docker using the commands given in the link: https://gist.github.com/brianz/8458fc666f5156fdbbc2

6. Kubernetes

   Kubernetes is a platform for deploying and managing containers. It is production-grade, open-source infrastructure for the deployment, scaling, management, and composition of application containers across clusters of hosts. It is primarily targeted at applications composed of multiple containers. It is therefore a group of containers using pods and labels into tightly coupled and loosely coupled formations for easy management and discovery. Install kubernetes using the commands given in the link: https://kubernetes.io/docs/tasks/tools/install-kubectl/

C. *Working Procedure*

- Install Git software (for windows). Create single container website called "Pet Mitra" using php. Initially launch two instances. Now connect to server_2 using "connect" option. Install Terraform and Kubernetes. After installing terraform, create a directory called "terraform" and move to that directory and create two files called "main.tf" and "variable.tf". Run the terraform script using commands using terraform init, terraform plan and terraform apply.
- It creates virtual private cloud (VPC), Subnets, Internet gateway and Route table. Here IP address is given externally by calling function “variable”, provide VPC name and enable DNS support and DNS host name externally. Then create four subnets out which two are public subnets (subnet1 and subnet2) and other two are private subnets (subnet3 and subnet4). Provide IP address of subnets within the function but give availability zone and vpc id by calling function “variable”. Create internet gateway and attach it to VPC. Create route table and attach route table to internet gateway. Now associate two public subnets to public route table and other two private route tables to private route table. Thus the whole infrastructure is automated. Thus creates vpc, subnets, route table and internet gateway. Exit from server. Create an account in github and create a public repository (say Pet-Mainproject-Kubernetes). Connect to server_1 using "connect" option. Install git, maven, Jenkins and docker. Create a folder called “sourcecode”. Inside that folder copy all the code of website pet mitra using “scp” command. Perform all the git commands (like init, add, push which are given while creating github repository) on the folder “sourcecode” to push code to git repository.

 DOI : 10.35291/2454-9150.2019.0091

- Create two bucket in S3 (AWS) with bucket name unique and enable versioning and set all the permissions. One bucket is to store output of Jenkins and other is to store kubernetes state information. Create user in IAM (AWS) with programmatic access and set Administartion access permission. Secret key and Access key will be created and use that in Jenkins to get access to AWS. Open Jenkins server using IP address of that server with port number 8080 exposed. A window Unlock Jenkins appears and prompts for "initialadminpassword". The path of initial admin password will be present in the window. Move to that path and copy the password and use it in the Unlock Jenkins window. Next another window appears to install suggested or specific plugins. Here "install suggested plugins" is chosen. After the installation of plugins, sign up page appears. Sign up using required credentials. Then click on "start using Jenkins". Then Jenkins dashboard appears. Here jobs can be created. Download git, maven and S3 plugins in manage plugins section. Click on create new job and create job (say "CI-petproject"). Then enter required credentials. Enter description of the project but it is optional. Select Source code management as "Git": enter repository URL and select "master" branch. Select required build trigger option. In Build field, enter "mvn clean package" command. In Post Build Action select Publish Artefacts to S3. Here enter S3 profile details obtained during S3 bucket and IAM user creation. Now the job will build the code and deploy to s3 bucket. Check S3 service in AWS. (Code is deployed). Now create an account in dockerhub using docker ID, password and email and create a repository (say "petproject"). Write a Dockerfile to create docker image. This Dockerfile contains commands to create docker image. It uses already build image 16.04, update the dependencies and install apache. Download php and mysql files. Also download the contents of Github repository to create an image and unzip. Expose port 80 and allow this website or code to run in foreground. After creating Dockerfile, build that file using the command "sudo docker build –t xyz/petproject: 10.0". This command build the Dockerfile and store that in Dockerhub repository specified in url. –t specifies the tag. Now run the Dockerfile using the command "docker run –d –p 80:80 b596f2c49e16". This command creates a container. b596f2c49e16 is the docker image id which is created when build command was executed. 80:80 is to map virtual machine and docker. After this push the docker image to repository using the command "sudo docker push petproject:10.0". Thus the code is dockerized. Use command "docker images" to view the docker images created. Use command "docker ps" to view the container created Check dockerhub repository for docker image with tag 10.0 Exit from server
- Now connect to server_2 using "connect" option. To set environmental variables and to store state information of kubernetes, create a bucket called project-kubernetescluster in S3. Give the name of cluster as "petproject.k8s.local". Now create cluster using the command "kops create cluster --node-count=1 --node-size=t2.small --master-size=t2.small --zones=us-west-1a --name=${KOPS_CLUSTER_NAME}**"**. It creates one worker node with size t2.small and one master node with size t2.small in zone us-west-1a. Then by giving following command kubernetes cluster will be created. "kops" is used to create cluster and "kubectl" is used for deployment.
- "kubectl cluster info" command gives complete information about the created cluster. Now deploy dockerized image inside kubernetes. Now run dockerized image using the command **"**kubectl run petproject010 --image=xyz/petproject:10.0 --replicas=1 --port=80". Here "petproject010" is a pod. Pod is a place where the docker image will run. Thus pod will create after executing this command. In order to know about the pod information, execute the command "kubectl get pods". Inside pod dockerized image is running. To know information about the deployment, execute the command "kubectl get deployment". Now code deployment should be exposed to load balancer. This is done by executing the command **"**kubectl expose deployment petproject010 --type=LoadBalancer --name=my-project010". To know the information of load balancer, execute the command "kubectl get service". Upon executing this command, external IP of the load balancer is obtained (a648de64b5e9a11e9a11906195251c80-138107180.us-west-1.elb.amazonaws.com). This is the end point. If this IP address is accessed, it will redirect to the pod where the website is running in a dockerized form. (Petmitra website code which is pushed as input to git repository). In real world, one cannot remember this load balancer link to access this website. So this link should be mapped to domain name. To create ease of access to this website, a domain name called "projectpet.tk" is purchased from "freenom". Then go to Route 53 service in AWS and create a public hosted zone called "projectpet.tk". Thus connection between freenom and AWS is created by updating NS Record obtained in Route 53 to "projectpet.tk". Also, "A Record" is created for load balancer link with some additional credentials. If anyone tries to access either "projectpet.tk" or load balancer link, it redirects to pet mitra website. Now the user request to "projectpet.tk" goes to freenom, from freenom it redirects to Route 53.

 DOI : 10.35291/2454-9150.2019.0091

From Route 53 it redirects to load balancer link and petmitra website appears.

IV. RESULTS

The below snapshots gives the result of each tool used.

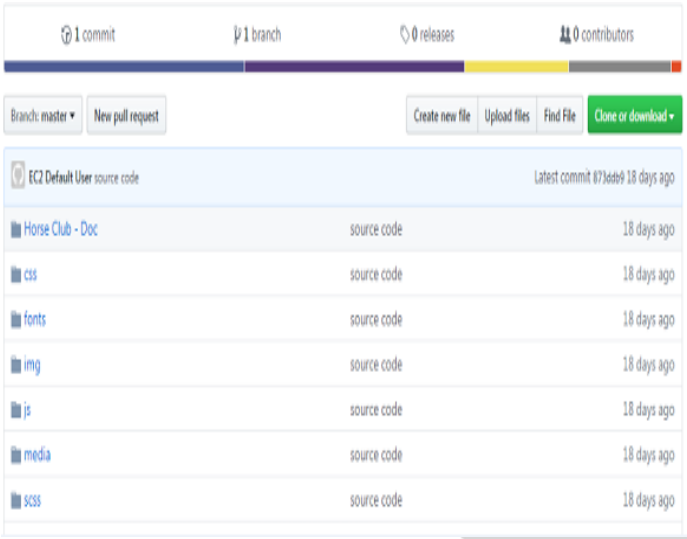

Figure 4: Code pushed from local to Github Repository after executing git commands

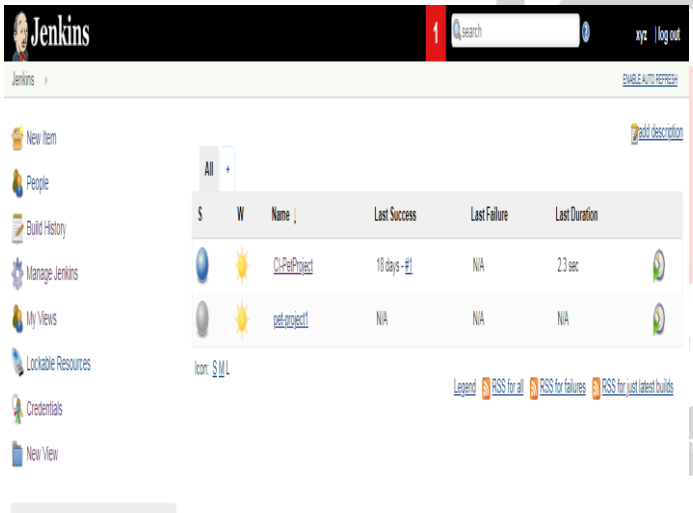

Figure 5: Job- CI-PetProject is created in Jenkins that build the code and deploy to S3

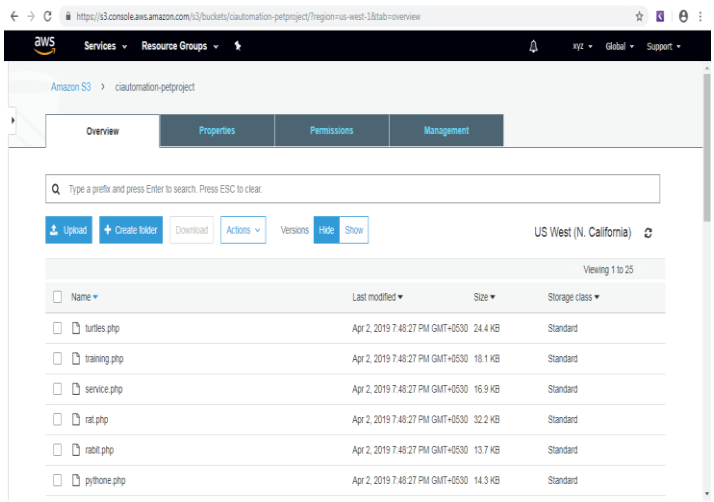

Figure 6: Code deployed to S3 by Jenkins job

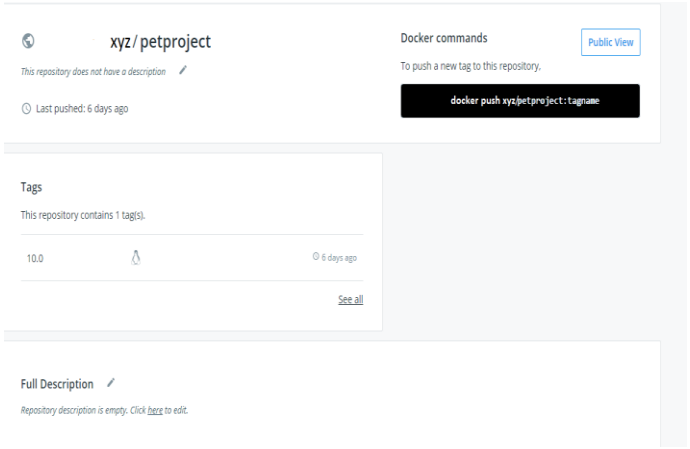

Figure 7: Docker image pushed into dockerhub repository petproject

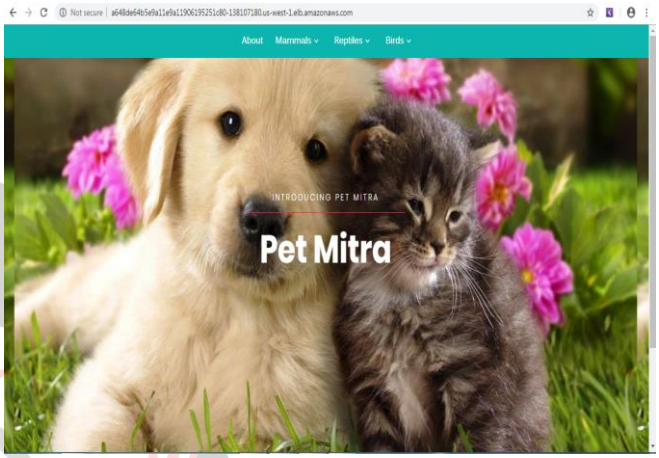

Figure 8: Docker image (contains code of input website) running in kubernetes cluster (pod) with the help of load balancer link

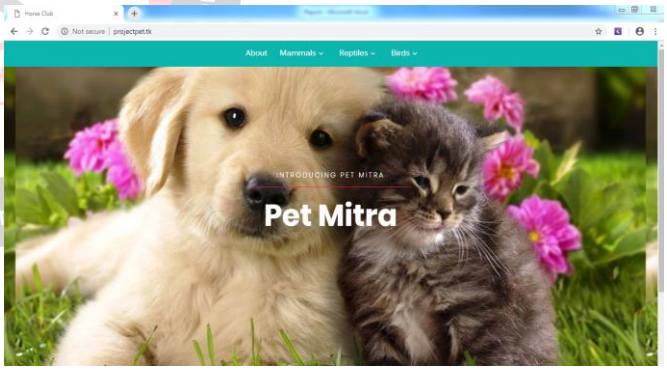

Figure 9: Website accessed via domain name "projectpet.tk"- the user request to "projectpet.tk" goes to freenom, from freenom it redirects to Route 53. From Route 53 it redirects to load balancer link and website appears.

V. CONCLUSION AND FUTURE WORK

DevOps is a methodology that improves the collaboration between Development and Operations teams. Enabling DevOps improves the speed of the delivery according to the business and customer needs. Especially automation in DevOps improves the productivity, reliability and allows standardizing the process, which in turn plays a major role in product delivery for organizations. Processes have to change with time as the market environment we operate in is continuously changing. Thus adaption of DevOps in the

 DOI : 10.35291/2454-9150.2019.0091

current era helps the industry to operate and deliver the products quickly. By using various tools like git, maven, Jenkins, docker, terraform and kubernetes this project tries to show the working of all these tools by developing single container website.

In this project single container application is used to show the working of all the tools. In future multiple container application can be developed to show the working of these tools along with the working of load balancer. Since the task of load balancer is to maintain the user request in turn called load on the application, it decides to which website access permission should be given to the user when there are multiple websites are running and that can be seen while using multiple container application.

REFERENCES

[1] Constantine Aaron Cois, Joseph Yankel, Anne Connell, "Modern DevOps: Optimizing software development through effective system interactions", IEEE International Professional Communication Conference (IPCC), 2014.

[2] Manish Virmani, "Understanding DevOps & Bridging the gap from Continuous Integration to Continuous Delivery", Fifth international conference on Innovative Computing Technology (INTECH 2015), 2015.

[3] Juhoon Kim, Catalin Meirosu, Ioanna Papafili, Rebecca Steinert, Sachin Sharma, Fritz-Joachim Westphal, Mario Kind, Apoorv Shukla, Felician Nemeth, "Service provider DevOps for large scale modern network services", IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015.

[4] Felicián Németh, Rebecca Steinert, Per Kreuger, Pontus Sköldström, "Roles of DevOps tools in an automated, dynamic service creation architecture", IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015.

[5] Hasan Yasar, Kiriakos Kontostathis, "Secure DevOps Process and Implementation", IEEE Cybersecurity Development (SecDev), 2016.

[6] Christof Ebert, Gorka Gallardo, Josune Hernantes, Nicolas Serrano, "DevOps", IEEE Software, 2016.

[7] Shahin, Muhammad Ali Babar, Liming Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices", IEEE 2016.

[8] Matt Callanan, Alexandra Spillane, "DevOps: Making It Easy to Do the Right Thing", IEEE Software, 2016.

[9] M Rajkumar, Anil Kumar Pole, Vittalraya Shenoy Adige, Prabal Mahanta, "DevOps culture and its impact on cloud delivery and software development", International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Spring), 2016

[10] Elisa Diel, Sabrina Marczak, Daniela S. Cruzes, "Communication Challenges and Strategies in Distributed DevOps", IEEE 11th International Conference on Global Software Engineering (ICGSE), 2016.

[11] Hui Kang, Michael Le, Shu Tao, "Container and Microservice Driven Design for Cloud Infrastructure DevOps", IEEE International Conference on Cloud Engineering (IC2E), 2016.

[12] S. Palihawadana, C. H. Wijeweera, M. G. T. N. Sanjitha, V. K. Liyanage, I. Perera, D. A. Meedeniya, "Tool support for traceability management of software artefacts with DevOps practices", Moratuwa Engineering Research Conference (MERCon), 2017.

[13] Wolfgang John, Guido Marchetto, Felician Nemeth, Pontus Skoldstrom, Rebecca Steinert, Catalin Meiros, Ioanna Papafili, Koastas Pentikousis, "Service Provider DevOps", IEEE Communications Magazine, 2017.

[14] Zhenhua Li, Yun Zhang, Yunhao Liu, "Towards a full-stack devops environment (platform-as-a-service) for cloud-hosted applications", Tsinghua Science and Technology, 2017.

[15] Arachchi, Indika Perera, "Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management", Moratuwa Engineering Research Conference (MERCon) 2018.

[16] Aayush Agarwal, Subhash Gupta, Tanupriya Choudhury, "Continuous and Integrated Software Development using DevOps", International Conference on Advances in Computing and Communication Engineering (ICACCE-2018) Paris, France 22-23 June 2018.

[17] Thomas F. Düllmann, Christina Paule, André van Hoorn, "Exploiting DevOps Practices for Dependable and Secure Continuous Delivery Pipelines", IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE), 2018.

[18] Lianping Chen, "Microservices: Architecting for Continuous Delivery and DevOps", IEEE International Conference on Software Architecture (ICSA), 2018.

[19] Aayush Agarwal, Subhash Gupta, Tanupriya Choudhury "Continuous and Integrated Software Development using DevOps", International Conference on Advances in Computing and Communication Engineering (ICACCE), 2018.

[20] Barry Snyder , Bill Curtis, "Using Analytics to Guide Improvement During an Agile/DevOps Transformation", IEEE Software, 2018.

[21] Len Bass, "The Software Architect and DevOps", IEEE SOFTWARE 2018.

 DOI : 10.35291/2454-9150.2019.0091