International Journal of Computer and Internet Security.
ISSN 0974-2247 Volume 15, Number 1 (2023), pp. 7-19

https://dx.doi.org/10.37624/IJCIS/15.1.2023.7-19

Report on Automation of SDLC Using Devops in Cloud Environment

Ch. Sravanthi
Asst. Prof, Information Technology, (JNTUH)
G.Narayanamma Institute of Technology and science(for women), Hyderabad, India
E-mail: sravanthi.cvr@gnits.ac.in

G. Tanmayi, M. Hemika, N. Harshitha, P. Bahula and R. Sushma
Information Technology, (JNTUH)
G.Narayanamma Institute of Technology and science(for women), Hyderabad, India
E-mail: tanmayigajadi29@gmail.com, mamillahemika@gmail.com
nimmalaharshitha2002@gmail.com, bahulapatharlapalli25@gmail.com
sushma.ent2@gmail.com

Abstract

Lately, the efficiency of the work done by the project development team has been a major concern. To deliver the end product to the customers on time the internal functions between various teams involved in the project has to go smoothly without any miscommunications so that they won't blame on one another as the root cause for unsuccessful project delivery. Different knowledge basis present among various teams in the project should not affect the project progress. When the project is moving from one team to another there should be any discrepancy in understanding the state of the project and the successor team should be able to work without having to know the core details of how the project has been developed so far. The roots for agile software process or methodology is to provide the right track to develop the product or project as per the client's input. Agile process gives importance to the people which includes the teammates, clients and their interactions rather than the tools, equipment or technologies they've been or they will be using; it focuses on the work products and its fidelity against the customer requirements; its attention is towards the adaptability as per the situations rather than focusing on the predetermined plans which consumes so much time to come up with one and one such plan doesn't exists upon following which there won't be any hurdles thus the endurance towards changes and flexibility is must. Agile process is about developing the project in the incremental and iterative fashion which reduces the concept of "finished project".The two

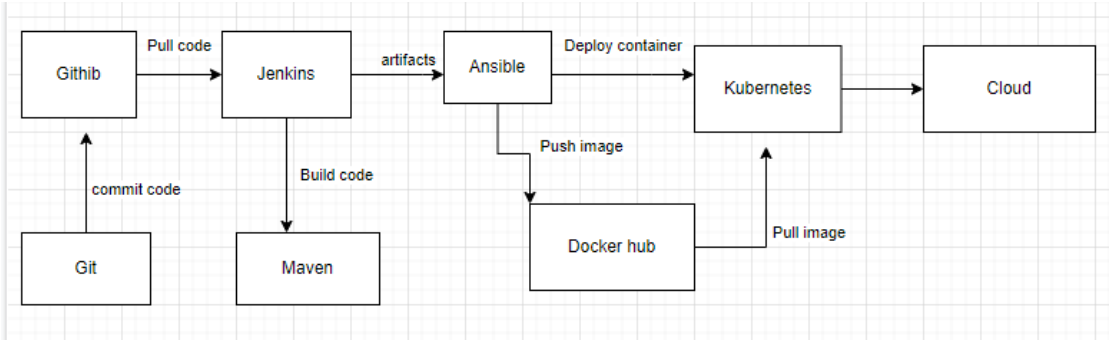
important areas of the software business is IT operations (ITOps) and Developmental operations (DevOps). ITOps focuses on the safety, reliability and compliance while DevOps focuses on designing the product and delivering it to the end users. DevOps work on pipeline optimisation and spotlights the continuous integration and installation of the software developed. Its major idea is on operational efficiency and end to end automation. Though the agile software development methodologies are more common many organisations realised that they have not reached a frequent release rate because of the gap between the development team and operations team. (thus, DevOps helps in automating the process from design to delivery of the product or process to the end user). DevOps removes the communication barriers among various teams involved and helps in constant delivery with continuous integration of developed with components with varying requirements.

Keywords: DevOps, Automation, software engineering, configuration management, agile methodology

Introduction

The business environment nowadays in terms of software & product design has evolved more rapidly. Project (a software project involves the execution of tasks to achieve added value outcomes) planning, evaluation, management (the planning, implementation, and monitoring of the tasks) of improvements, and quality control are some of the challenges that differentiate successful and failed projects. Agile methods (incremental and iterative approaches) are widely used and implemented worldwide to deal with these issues. Software developers concentrate on agile development, for increasing the productivity of their projects and meeting the competitive demands of their customers. Agile methodology has been developed to address the problems faced by the conventional model and to offer project teams multiple possibilities during the development process. But then here there is no end for the successful project completion there might be the chance that customer suggest at least one change at each increment due to which there will be unlimited time and resources consumed which leads to cost overrun; and as the outputs are generated in fragments at each increment they are difficult to track as well as the incremental development by different teams on different units of the project causes some teams to wait till the others do their work and if one team lags behind then all other must be on hold with causes misunderstandings and more time consumption to resolve internal conflicts thus management becomes hectic. Whereas when it comes to DevOps its main motto is to reduce the gap between the Development team and Operations team. The gap between both the teams arises because they come from different backgrounds (knowledge domain) that is development team knows about the software products and service whereas the team knows about the testing and production environments thus there will different business goals and priorities between both the teams. DevOps promotes automated continuous integration and deployment pipelines to enable frequent releases, to evaluate the work

products in sprints rather than focusing on the adaptability to change. DevOps reduces the complexity and helps in faster product delivery, issue resolution, resource utilisation, scalability, availability, stable operating environments, visibility into system outcomes and many more.



Literature Survey

Agile and DevOps are bombinate ideas that are commonly used in many organisations. There are many common features of both which overlap together and cause the ambiguity whereas the difference lies in the outcome and techniques followed though the core structure of providing the end product to the customer on time as per the requirements remains the same. DevOps focuses on the continuous integration, continuous deployment, continuous monitoring and continuous feedback and optimisation whereas the Agile methodology is all about the flexibility, adaptability, incremental and iterative development. DevOps is oriented towards the faster performance by proving various environments, tools and techniques to the allocated where the work products generated by predecessor teams will be clearly seen along with the configurable files and software which are required to run them.

Proposed System

The proposed system helps in automating the development process for building a project by provide the interface for clients, software client manager, software architect and developer to communicate with each other and it provides an integration point via Jenkins for developers to commit the code through git and develop the code through maven (which together provides the continuous integration) from there a pipeline is created from Jenkins to ansible(for handling the configurable files for deployment) and ansible to docker and Kubernetes (which provides the continuous deployment and delivery). Our system provides the communication media(chat application) for clients to pitch their ideas and once the client software manager likes the idea then he discusses with other teams about various factors(such as budget, resource allocation, technology used, tools and software required, time required to complete the project etc) and he may also convince the clients on what can be done based on the allocation to avoid the exception expectations from the clients. Then he forwards the reviewed requirements to the software architect via the proposed system so they start building suitable models

and once these models are built then they are forwarded to the software developers and then they start developing the code and commit them onto the git where there is ci/cd pipeline to the Jenkins and this code is checked against the code built by the Maven (which already has the pipeline to the Jenkins[central point]) automatically and the functionalities done till here from the code commit on git is known as the continuous integration. Then the built code is sent to the Ansible which generates the artefacts and these artefacts are given to the docker via pipeline which then generates images and these images are deployed onto the cloud with the help of Kubernetes. All the functionalities after the code generation to the image and container deployment is regarded as the Continuous Deployment (that is the code is automatically deployed into various environments as per the changes in the code committed on Git)

Implementation

The modules are implemented using AWS Management Console. The tools used are Jenkins, Git, Maven, Ansible, Docker and Kubernetes.

Creating CI/CD pipeline for Git, Jenkins and Maven:

In the continuous integration and continuous deployment process, we need to build our source code and play it in that target environment. The code should be already available in our GitHub account. Once the code is commited, we need to integrate GitHub with the Jenkins and Maven with the Jenkins so that we can pull the code onto Jenkins and build with the help of Maven.And if we need any changes to our source code with the help of Git from our local workstation station will mark play it and will commit that code on to GitHub.

Setup Jenkins Server:

STEP-1: Setup a Linux ec2 instance
STEP-2: Install Java
STEP-3: Install Jenkins
STEP-4: Start Jenkins
STEP-5: Access it on the web UI using port 8080

Setup an ec2 instance

STEP-1: Go to AWS Management Console click on "Launch Instance" and then **choose the AMI**(Amazon Machine Language) which is nothing but choosing the operating system for this project we'll be choosing Amazon Linux 2
STEP-2: Choose an **Instance type** we'll be choosing t2 micro
STEP-3: Then choose the **Configure instance details** as per the requirement but we'll be using the default details
STEP-4: Select the storage required we're going to use the default storage
STEP-5: Then we need to **Add tags** we'll give the Key as Name and Value as the Jeniks_Server

STEP-6: Then we need to **Configure security group** where we'll be creating an security group instead of using the existing one for which for the **Security group name** we'll be giving "Jenkins_Security_Group" and even **Description** we'll be giving the same and we'll give the **Port range** as 8080 as the Jenkins runs on the port 8080 and for **Type** we'll give Custom_IP
STEP-7: Then click on "Review and Instance" after reviewing what you've created
STEP-8: Then a window pops up saying for choosing the key_pair we'll be creating a new key pair and Give the Key Pair Name as DevOps_Project_key then download that key_pair and launch the instance

We'll be using MobaXterm to connect our ec2 Instances to connect to the Linux ec2 instance we'll use the **Session** there click on SSH and give the public IP address of the Jenkins ec2 instance which we've created and then check in the box "Use Private Key" and choose the downloaded key pair of Jenkins over there and give the Specific User Name as "ec2-user"

Installing Jenkins
For this we'll be writing the following commands in the ssh session we've created

```
sudo wget-O /etc/yum.repos.d/jenkins.repo \https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm--import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
sudo yum upgrade
# Add required dependencies for the jenkins package
sudo yum install java-11-openjdk
sudo yum install jenkins
sudo systemctl daemon-reload
```

To check the status of the Jenkins server use: "service jenkins status"
To start the Jenkins server use: "service Jenkins start"

Integrate Git with the Jenkins
STEP-1: Install Git on Jenkins Instance
STEP-2: Install GitHub plugin on Jenkins GUI
STEP-3: Configure Git on Jenkins GUI

Jenkins Job to pull the code from GitHub
For that goto Jenkins GUI and then click on "New Item" and then enter the name as "PullCodeFromGitHub" and click on "Freestyle Project" and then click on "OK"
Then give the Description same as the job name that is "Pull Code From GitHub" and then under "Source Code Management" section choose "Git" then under Repository URL section you need to give the URL of the GitHub where the code is present and if it is a Private Repository you need to provide the credentials for the Public repository

there is no need for that and then click on “Apply” then “Save” and By default, Jenkins is going to store all the build related information under /var/lib/Jenkins/workspace

Integrate Maven with Jenkins
STEP–1: Setup Maven on Jenkins Server
STEP–2: Setup Environment variables [JAVE_HOME, M2, M2_HOME]
STEP–3: Install Maven Plugin on Jenkins GUI
STEP–4: Configure Maven and Java on Jenkins GUI

Building the source code through Maven via Jenkins
For that goto Jenkins GUI then click on “New Item” then give the name “FirstMavenProject” then click on the “Maven Project” and then on “OK” and then give the description as the “First Maven Project” and under the “SourceCode Management” choose “git” and in the “Repository URL” give the URL of the source code and in the “Build” section give the “pom.xml” then if want to specify any “goals” you can do that and then click on “Apply” and “Save”
If we goto our FirstMavenProject under the workspace in the webapp under the target directory is where all our build artifacts gets stored. This build generates the.war file as the output

Deploying our code in the target environment
Here the target environment we’re using is Tomcat Server. For this we’ll be creating an ec2 instance on top of that we’ll be installing the tomcat and we’ll setup Jenkins job such for deploying the code onto the Jenkins

Setup a Tomcat Server
STEP-1: Setup a Linux ec2 Instance
STEP-2: Install java
STEP-3: Configure Tomcat
STEP-4: Start Tomcat Server
STEP-5: Access Web UI on port 8080

Setup Docker Environment
For this setup we need a Docker host upon which we’ll install “Docker” and create “Docker container” Setup of Docker Host includes

- Setup a Linux EC2 instance
- Install Docker
- Start Docker services
- Basic Docker Commands

For integration the steps include

- Create a Dockeradmin user
- Install "Publish Over SSH" plugin to deploy the artifacts from Jenkins to docker container
- Add Dockerhost to Jenkins "configure systems" so that the Jenkins will be able to communicate with the DockerHost
  "cat /etc/passwd" to list all the users present
  "cat /etc/group" to list all the groups present

Now we need to create a user and add it to the docker group

- "useradd dockeradmin" to create a user dockeradmin
- "passwd dockeradmin" to create a password for that user
- "id dockeradmin" to check the group which it belongs to
- "usermod-aG docker dockeradmin" to add the dockeradmin user to docker group
- "vi /etc/ssh/sshd_config" and then search "Password" by /Password and then decomment the "PasswordAuthentication yes" and comment out "PasswordAuthentication no" by this it enables the password based authentication
- "service sshd reload" to reload do not stop and start because once you stop it you're going to loose the connection to dockerhost

Deploying on a Container

We're going to create a new job where a artifacts from Jenkins are deployed onto the docker container For that goto "New Item" then enter the name of the job as "BuildAndDeployOnContainer" and at copy from choose "BuildAndDeployJob" and then click on "OK" and then at "General" section give some description and scroll down to "Post-Build Actions" and delete the existing one and add the new "Post-Build Action" by clicking on it and choose the option "Send build artifacts over SSH" which is enabled because we've installed "publish over ssh" plugin and then at name choose the ssh server from what we added so far then at "Source Files" give "webapp/target/*.war" and then at "Remove prefix" give "webapp/target" as we only want.war files to copied and then click on "Apply" and then "Save"

Updating the deployment process

For that login to SSH by the instance created for docker
"exit" to logout
"cd /opt"
"mkdir docker" creating the docker directory
"chown-R dockeradmin:dockeradmin docker" to change the ownership of "docker" from root to "dockeradmin"
"mv DockerFile /opt/docker" to move the DockerFile into docker directory
"chown-R dockeradmin:dockeradmin /opt/docker" to change the ownership

Automate the build on Docker Container

For this edit the configuration of “BuildAndDeployContainer” by clicking on it and then select the “Configure” and then scroll down to “Exec Command” under “Post-Build Actions” there give the commands “cd /opt/docker; docker build-t regappv1. ” ; “docker run-d--name registerapp-p 8087:8080 regapp:v1” and then click on “Apply” then “Save” and then for the execution click on “Build Now” and you can access your application on the port “8087” but the problem here will be that you cannot use same given name “registerapp” for different containers which will be created for every change in the code

Thus to overcome that we’ll go back to our job and then click on “Configure” and then under “Post Build Actions” at the “Exec command” add two more line above the last line they are “docker stop registerapp ; docker rm registerapp ” that is we’ll be stop the existing container and then we’ll be removing it before creating the new container wit the same for different artefact

To install the latest version of AWS cli run the following commands in the SSH session curl

"https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip"-o "awscliv2.zip" unzip awscliv2.zip
sudo./aws/install

Setup the Kubectl by the following four commands

curl-o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.21.2/2021-07-05/bin/linux/amd64/kubectl
chmod +x./kubectl
mv./kubectl /usr/local/bin
kubectl version--short –client

Setup the eksctl by the following

curl--silent--location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname-s)_amd64.tar.gz" | tar xz-C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version

Create IAM role and attach it an EC2 instance

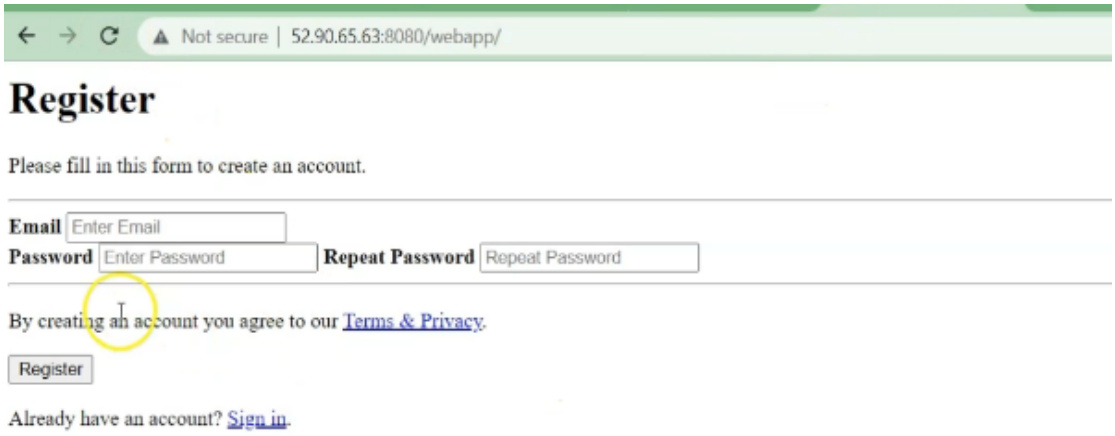
For that go to AWS Management console and search for IAM and from the IAM dashboard select “Roles” and then click on “Create Role” as we’re this role to EC2 select an EC2 and then click on Next Permission then again search for “Administrative Access” and click on next then ignore tags i.e., just click on nex and then give the role name as “eksctl_role” and then click on “create role”.

Setting up Kubernetes on EKS
Use the following commands to create a cluster
eksctl create cluster--name valaxy \
--region us-east-1 \
--node-type t2.small \

Create Jenkins deployment job for Kubernetes
Creating job so that artifacts from Jenkins are deployed on to the Kubernetes. For that login to your Jenkins and then click on "New Item" and then give the name as "Deploy_On_Kubernets" then choose "Freestyle Project" and click on "Ok" and after give some description and scroll down to "Post-build actions" there you choose "Send build artifacts over ssh" and then at the name section choose the ssh server then at "Exec Command" enter the necessary configurations required and then click on "Apply" and then "Save"

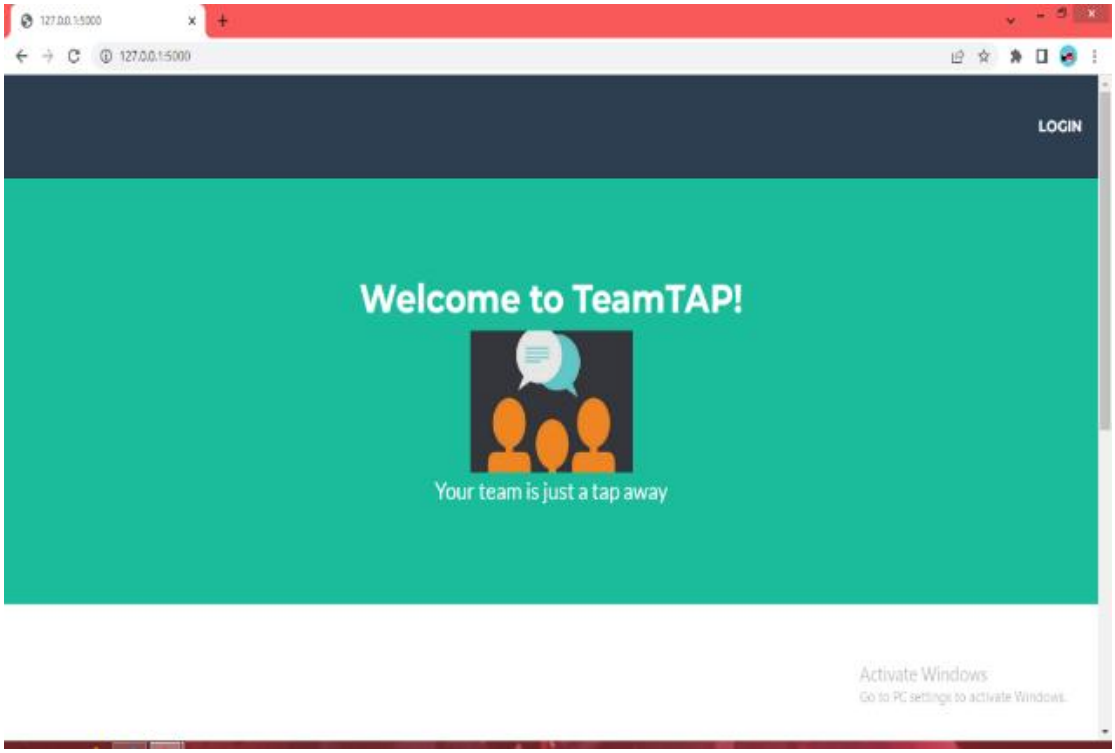
CI Job to create Image for Kubernetes
We've created a CD job which takes care of deployments now we'll create a CI job which will pull the code from GitHub and build the artifacts in Jenkins via Maven and we'll generate a new images for every artefact that is being generated and try to deploy that on the Kuberenetes to do this we'll create a new job for that go to Jenkins and click on "New Item" give it a name and copy from the "BuildAandDeployOntoContainer" and then click on "Ok" and then give the Description as "Build code with help of maven and create an image" then choose "git" and give the repository URL and then specify the branch as "master branch" and then under "Post-Build Actions" choose the SSH server and at the exec command give the following
"ansible-playbook /opt/docker/regapp.yml"
"sleep 10"
"ansible-playbook /opt/docker/deply_regapp.yml"
And then click on "Apply" and "Save"

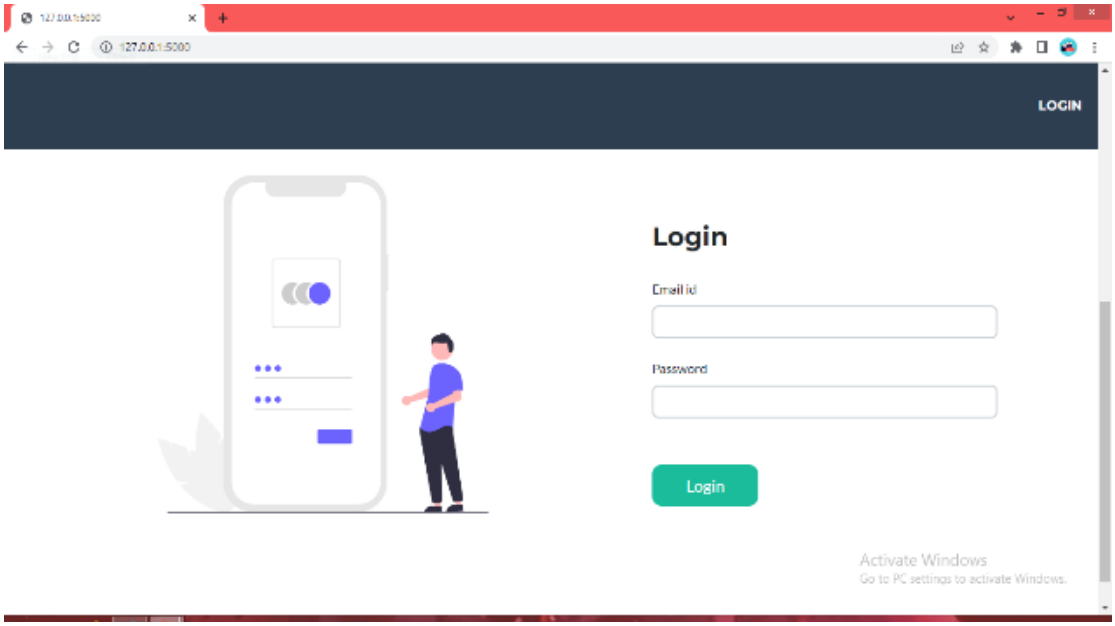
Results
The pipeline between the Jenkins and Docker is established to deploy every artifact that is generated for every code commit on the image which then containerised and the success of this job indicates the access of that container on the browser

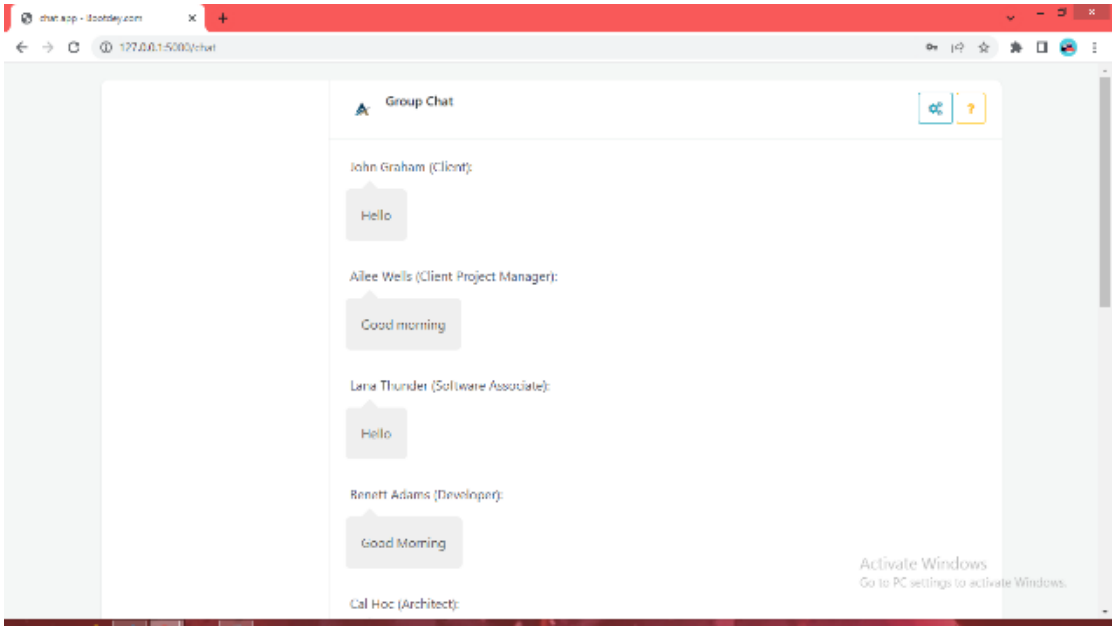


This pipeline is created to deploy the containers that are generated after every commit on to the Kubernetes the success of this job reflects in the deployment of container on Kubernetes successfully after every change in the code that has been committed and that right deployment is showcased as per the display on the browser and corresponding change in the code

Chat Application that serves as the communication medium between stake holders







Conclusion

DevOps integrates the efforts of the development team and operation team and turns their work progress lead to a successful project with atmost quality and customer satisfaction. Our system provides the cohesive way to develop a project right from the requirements stage to the deployment stage as all the required DevOps tools are connected via pipelines considering Jenkins as the central point which helps from committing the developed code to deploying the container onto different environments and the chat window provides the interface to gather all the requirements. This system enables frequent updates based on the feedback given customers to make the product more user friendly with the maximum utilisations of components which are already built (that is provides component based development). There will be no specific timings scheduled for a client to interact with the manager he/she will be able to theirs requirements whenever they want on the chat window and they can expect the

Future Scope

1. Customer support 24/7 when the system is down reasons for it alternatives that can be followed….
2. Video and audio conference so that diff teams can interact remotely
3. Taking this to a high scale tor handling diff teams on diff project with diff set of customers
4. Providing customer to try what has already been built and get the initial feedback but not so frequently as it delays but after major quarters of completion
5. As Maven and Jenkins are extensively used on java based applications to use various devops tools where there won't be any restriction on the specific language based development

Acknowledgement
It is our proud privilege to express the feelings of gratitude to several people who helped us in completing this work We express our heartfelt gratitude to Dr. I. Ravi Prakash Reddy, Head of the Department, IT department, GNITS for his constant guidance and moral support. We would like to express our sincere thanks to Dr. K. Ramesh Reddy, Principal, GNITS for providing working facilities in the college. Finally, we would like to thank all the faculty and staff of IT department who helped us during our project and also our parents and friends for their cooperation in completing the project.

References

[1] Moreira, M. The Agile Enterprise: Building and Running Agile Organizations, 1st ed.; Apress: Berkeley, CA, USA, [**Google Scholar**]
[2] Fitzgerald, B.; Stol, K.J. Continuous Software Engineering: A Roadmap and Agenda. J. Syst. Softw., 123, 176–189. [**Google Scholar**] [**CrossRef**]
[3] Bosch, J. Continuous Software Engineering: An Introduction. In Continuous Software Engineering; Springer: Berlin/Heidelberg, Germany, ; pp. 3–13. [**Google Scholar**]
[4] Humble, J. Continuous Delivery vs. Continuous Deployment. Available online: **https://continuousdelivery.com/2010/08/continuous-delivery-vs-continuous-deployment**
[5] Jenkins. Build Great Things at any Scale. Available online: **https://jenkins.io**
[6] Chacon, S.; Straub, B. Pro Git, 2nd ed.; Apress: Berkeley, CA, USA,. [**Google Scholar**]
[7] Christof, E.; Gallardo, G.; Hernantes, J.; Serrano, N. DevOps. IEEE Softw., 33, 94–100. [**Google Scholar**] CHALLENGES IN PRACTICE‖, NOV 2016