

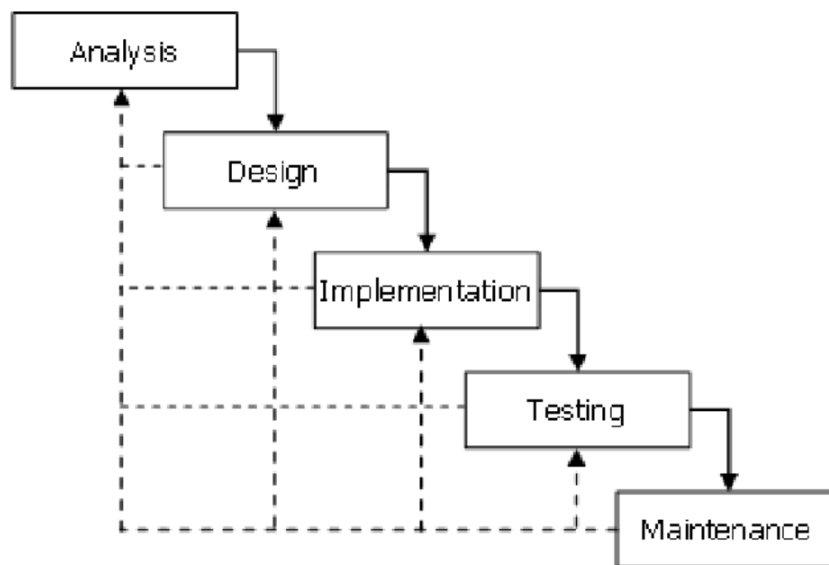
1. INTRODUCTION

Patrick Debois, who devised the name “DevOps” in 2009 and he is also called as “The Father of DevOps” devised the name “DevOps” in 2009. The word DevOps says itself that it formed by coalescing the two words “Development” and “Operations”. DevOps is the collaboration of development and deployment of software. DevOps is the portmanteau of development and operations. It is a software development method that escalates to the amalgamation between software development team and operations team. This is the time to change the old technology to new technology like DevOps. - “Time to stop wasting money, time to start delivering great software and building systems that scale and last” – Patrick Debois [15].

The new software delivery procedure is adopted by the organizations since the market needs are changing continuously, rapid change in technology to deliver quickly. Customer waiting for 6 months or 1 year for a version to be released and giving feedback after release cannot happen nowadays. Customers need continuous engagement with the project so that they can provide the feedback continuously. In order to face the challenges, the organization should be lean and follow the agile transformation in all phases of SDLC. Over the years the organizations have adopted the agile transformation for optimization, but the evolution takes place to change the technology to DevOps [8]. It is important to keep all the phases in pace so that the software delivery lifecycle will not be delayed. DevOps is the mechanism which bridges the gap between developer-operations and not only limited to developer-operations but also for the continuous development, continuous testing and continuous integration [5]. DevOps main goal is to deliver the software rapidly with continuous development, continuous integration, continuous feedback and communication with development and operations team [1]. DevOps is the extension of agile principles in software delivery pipeline. DevOps principle plays an important role in complete SDLC, but it makes sense if both the development team and operations team work in a same pace [8]. Mental fitness implies a state of psychological well-being. It denotes having a positive sense of how we feel, think, and act, which improves one’s ability to enjoy life. It contributes to one’s inner ability to be self-determined. It is a proactive, positive term and forsakes negative thoughts that may come to mind. The term mental fitness is increasingly being used by psychologists, mental health practitioners, schools, organisations, and the general population to denote logical thinking, clear comprehension, and reasoning ability.

1.1 Related Work

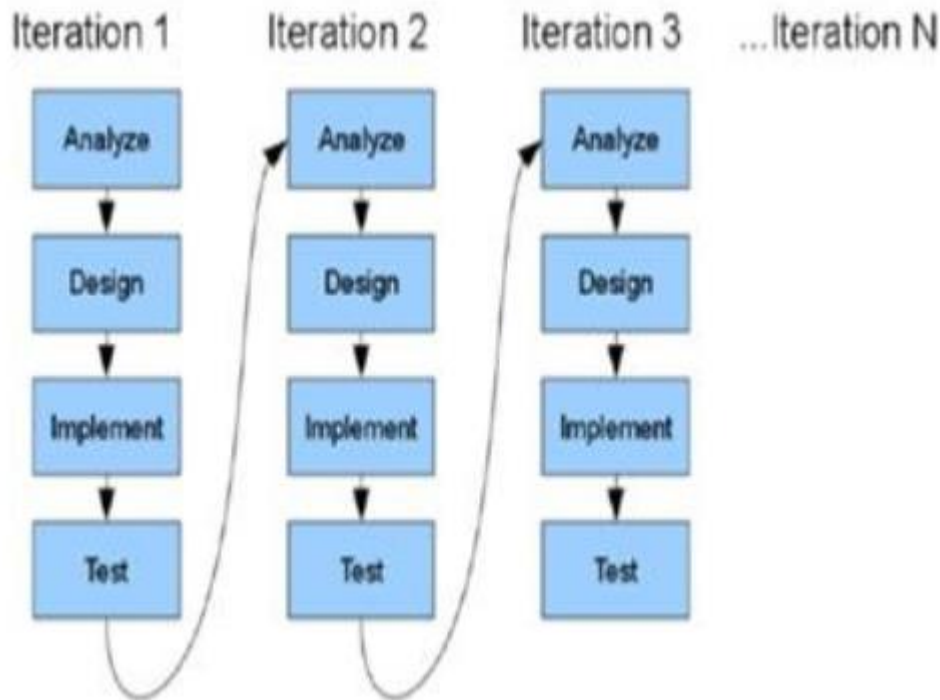
The first introduced Process Model is the Waterfall model. It is also referred to as a traditional model or linear-sequential life cycle model. It is very simple to understand and use. In this Waterfall model, every phase must be completed before going to the next phase and there is no issue of overlapping of phases. It is the first SDLC approach model for software development. The waterfall model illustrates the software development process flow in the linear sequential flow. The waterfall model process development is divided into separate phases and will start only after completion of the previous phase. The phases of Waterfall development model are as follows 1. Requirement gathering and Analysis 2. System Design 3. Implementation 4. Testing 5. Deployment 6. Maintenance This model is the first model developed for the software development and ensures success, but the disadvantage is that each phase should be done only after completion of the previous phase it results in a long duration and all the requirements are to be specified before starting the project [2]



Waterfall Model

1.2 Agile Model

Agile software development model is the process model which is the combination of both iterative and incremental model. The agile model focuses on process adaptability and customer satisfaction by rapid development and delivery of software product [12]. It breaks the complete product into small incremental builds and these builds are completed in iterations. These iterations last for one to three weeks and involve in cross-functional product development. The teams simultaneously work on these phases 1. Planning 2. Requirement Analysis 3. Design 4. Coding 5. Unit Testing 6. Acceptance Testing This agile model after completion of system testing for the product it will be deployed in the market. It helps to complete the project rapidly. It is also suitable for static or dynamic requirements, but it is not suitable for handling complex dependencies and a high risk of sustaining, maintaining and extending [3], [4].



1.3,1.4

DEVOPS AND ITS IMPORTANCE

Patrick Debois, who is called as “The Father of DevOps” devised the name “DevOps” in 2009. DevOps (a portmanteau of development and operations) is a software development method that escalates to the amalgamation between software developers and information technology (IT) operation professionals.

A. DO WE REALLY NEED DEVOPS?

Developers always want to deliver the changes in the product as soon as possible whereas the operation team want reliability and stability in the product. This situation was explained clearly in “wall of confusion” by Lee Thomson is shown in figures 4 & 5. This wall of confusion not only gives the mentalities of two teams but also the tools they practice.

Development team uses some tools and Operations team uses different tools to perform the same task. DevOps bridge the gap between the development and operations for better and faster results

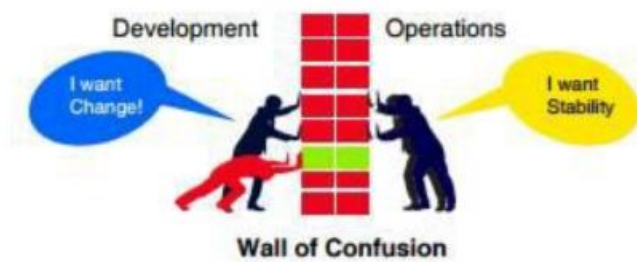


Figure 4: Wall of Confusion without DevOps

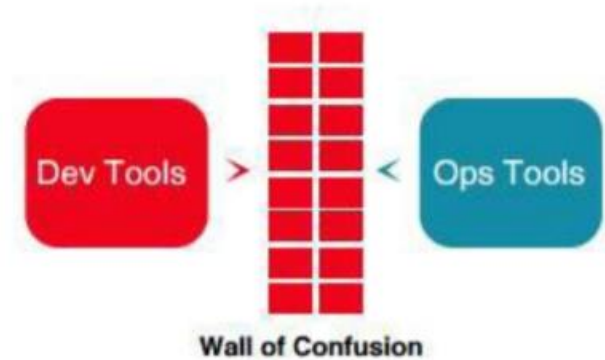


Figure 5: Wall of Confusion with DevOps

B. WHAT DRIVES THE NEED OF DEVOPS?

1. Strong collaboration between development and operation teams.
2. Synchronized deployment across multiple platforms.
3. Pressure to release applications to meet customer requirements or to enter into the new market.
4. Improving end user capability levels.
5. Vast usage of smart devices
6. Necessity to develop and deploy into cloud-based applications.
7. Increasingly complex IT infrastructure.
8. Need to reduce the cost for IT industry

C. RECOGNIZING BUSINESS VALUE OF DEVOPS

DevOps applies agile and lean principles in the complete software deployment process to enhance the speed of delivery of product or service from the initial release to the production release and to the feedback given by the client based on the release. DevOps return our investment in these three areas.

1. **Enhanced Customer Experience** Delivering an enhanced product for the customer leads to build loyalty and increase in market share. To deliver an enhanced product we need to continuously obtain and respond to the customer's feedback faster and perform required changes suggested by the customer.
2. **Increased capacity to Innovate** Lean thinking approaches are used in modern organizations to increase their innovation capacity. Their goals are to utilize the resources efficiently for other activities by reducing waste and rework. An example of lean thinking in organization is A-B testing in which large organizations asks a small group of users to test and rate two or more sets of software having different capabilities then the better capability set is picked up for the users and unsuccessful version is rolled back. This A-B testing is reliable only if efficient and automated mechanisms are adopted such as DevOps.

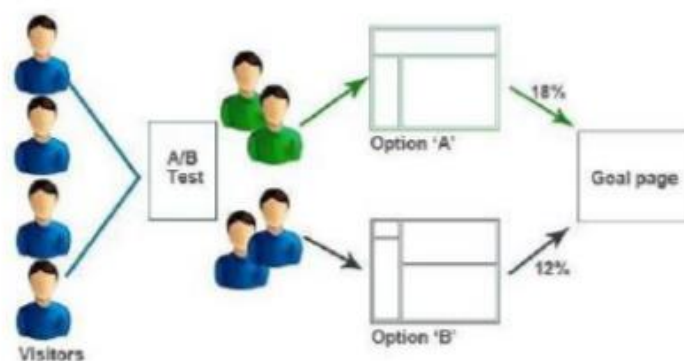


Figure 6: A-B Testing

A-B testing is the comparison of two web pages to know which will perform fast and efficient. It is also called as split testing. We compare two web pages by showing the two variants (let's call them A and B) to similar visitors at the same time. The one which has more conversion rate is accepted.

3. **Faster time to value:** This involves in development of new culture and practices and automating the project leads to fast and reliable delivery process throughout the production phase. This DevOps can be worked as a business capability with the tools for release planning, predictability and success. The DevOps main goal is to deliver the value faster and in efficient way and the value definition changes with organization or with the project.

IV. DEVOPS PHASES AND DELIVERY PIPELINE

A. Continuous Planning Business plans are already using agile methodologies to deliver quickly and change according to market conditions. It is better to have the checkpoints so that we can easily do the necessary changes given as feedback by customer. Dev / Test teams adapting to quick changes is not an easy task in business environments. DevOps allows us to prioritize the product backlogs and taking business perspective into consideration. This is the continuous process of planning, executing, getting feedback from the customer, the cycle continues [11].

B. Continuous Integration Continuous Integration means dynamically integrating the changes made to the project to the team and not restricted to our local machine and validates the behaviour of the code. Sharing with component teams but integrating beyond component boundaries at product integration level. Further the process optimization refers to automation as soon as the developer delivers the change build systems must detect the change and trigger a build taking sanity test and building repository. This must be a cyclic process across the development [9, 10].

C. Continuous Deployment Continuous Deployment is the heart of the DevOps and acts as the Centre point to the complete software delivery optimization. Most of the surveys said that in many organizations the reason for the delay in software delivery is the operations. Hardware setting in the development build may vary from days to weeks. These deployment processes are inconsistent and manual. DevOps principles recommend the automation of deployment and hardware provision and cloud play a vital role in this field. DevOps proposes a concept called Infrastructure as a code (IAAC) which says that complete infrastructure provision should be maintained in source code repository [6].

D. Continuous Testing Automation is the best option for continuous testing to test every test case. If any process need to do repeatedly for some constant time it is better to automate that process. They are humongous applications available in the market for do that type of testing process to meet the goal. There will be a chance of maximum to automate the manual testing process we need to evaluate on the same. Software delivery process must be able to execute the test suite automatically with the user intervention leading towards the goal reach easily. This kind of process not only makes testing process automate but also allows test cases to be carried out fast in production like system (deployment) [6].

E. Continuous Monitoring As discussed in all the above approaches we adopted, there is a chance to observe various parameters and react to them accordingly. The capability to test early and production like systems we can react to them in timely manner.

V. ARCHITECTURE OF PROPOSED SYSTEM

The proposed system architecture contains the three phases known as DEV, STG and PRD. These three phases are explained here with some of the tools required to work in these phases [14].

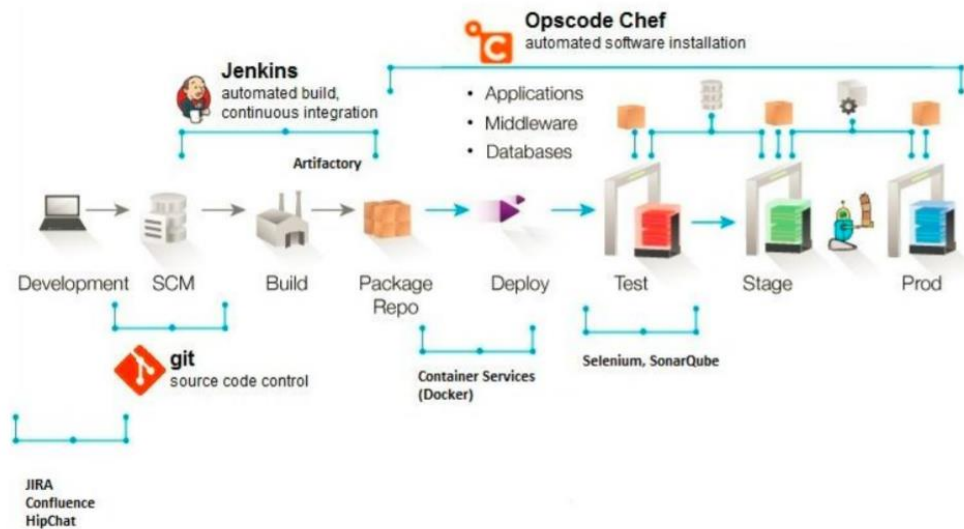


Figure 8: Architecture of Proposed DevOps System

A. DEV:

Dev is the kind of server in which the complete development of the project is done and uses various tools in this phase. After completion of the development process the project is sent for testing and this is tested in the test server. In some cases, Dev is also used for testing purposes like debugging [13].

B. TEST

Test is also a kind of server in which the testing takes place for the developed project. This testing will have some test cases written for testing of project. Testing can be manual or automated.

C. STG

STG also known as staging in which all the test cases passed and the project configuration and acceptance testing by the customer is done. If the customer agrees then it goes to deployment otherwise it will be changed according to the instructions given by the client.

D. PRD

PRD server is used for the production purpose. After accepting by the customer, the complete project after passing through all the tests will be deployed and released into the market.

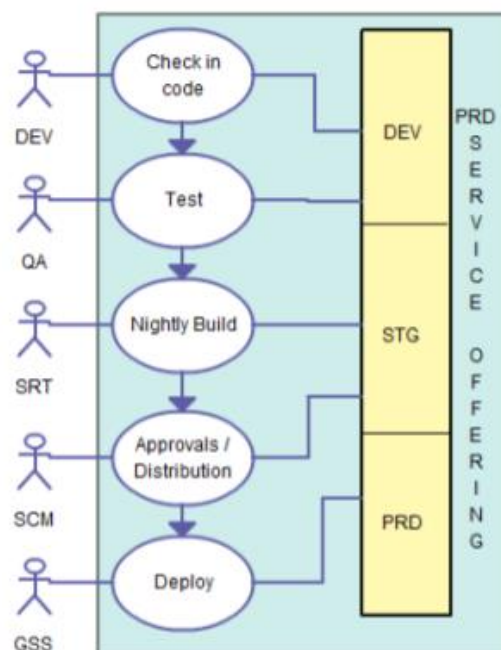


Figure 9: Dev, STG, PROD structure

PART A

TOOLS USED

1. NPM for Gradle Build

It is a Software Management tool works with Project Object Model (POM). It is used as a reporting, building, documenting from an informational central source point and can be used for building Java based projects.

2. GIT SOFTWARE

Git is a version control system for tracking the changes in the local machine or the distributed system among the multiple people in the project. Its primary use is to Source code management (SCM) in software development, but it is able to track the any number of changes done in the project. In distributed system its aim is to provide the speed, data integrity, availability and support for the distributed and nonlinear workflows.

3. JENKINS

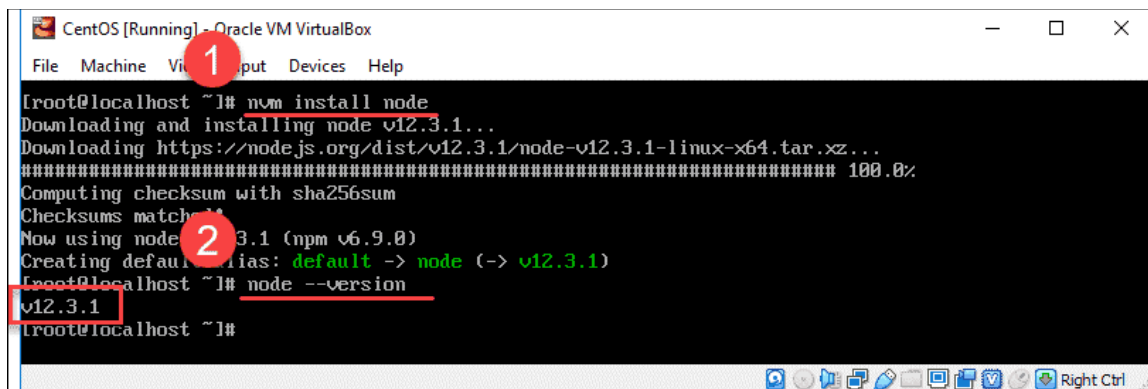
Jenkins is a continuous integration (CI) server, or a tool written in java language. It is open-source software for download. The continuous integration services can be provided for software development which can be done via command line or web application server.

4. ANSIBLE

Ansible is the automation software that provides configuration management, software provisioning and application deployment. A platform was created by the Michael Dehaan, who is the author of the provisioning server application cobbler and a co-author of the function framework for remote administration. It is the part of Fedora Linux which is owned by Red Hat Inc., and is available for the Red hat Enterprise Linux, Cent OS, Oracle Linux via extra packages for Enterprise Linux (EPEL), Scientific Linux and for remaining OSs also. It is used for automating the tasks like software installation etc.

VI. RESULTS AND DISCUSSION

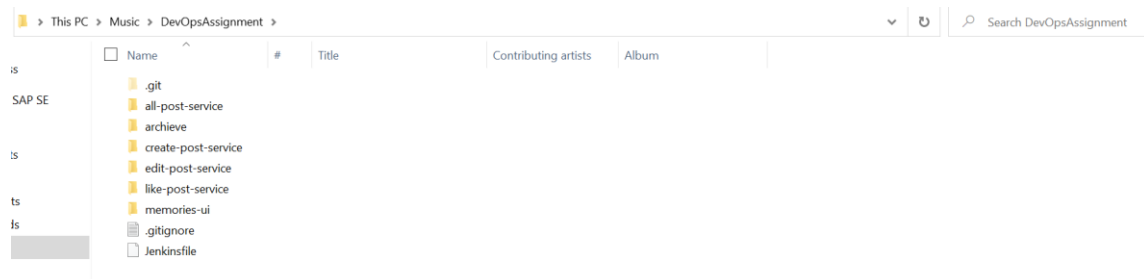
The results for the npm, Git, Jenkins are shown here with the detailed steps for implementation of the system.



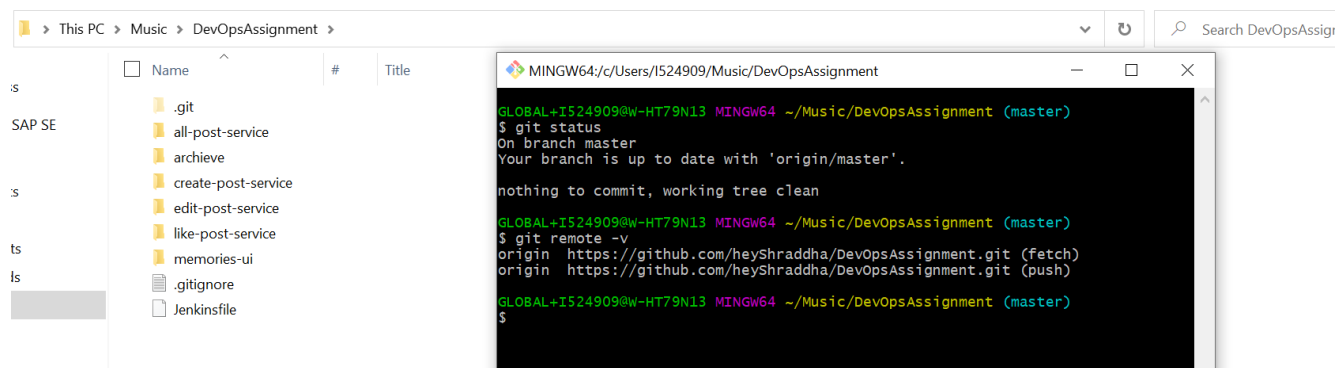
```
CentOS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@localhost ~]# npm install node
Downloading and installing node v12.3.1...
Downloading https://nodejs.org/dist/v12.3.1/node-v12.3.1-linux-x64.tar.xz...
##### 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v12.3.1 (npm v6.9.0)
Creating default alias: default -> node (-> v12.3.1)
root@localhost ~]# node --version
v12.3.1
root@localhost ~]#
```

```
C:\Users\I524909>npm -v
6.14.11
```

File Structure



Single project folder



Aligned with GitHub for Version Management

```

PS C:\Users\I524909\Music\DevOpsAssignment\all-post-service> npm i
> nodemon@2.0.15 postinstall C:\Users\I524909\Music\DevOpsAssignment\all-post-service\node_modules\nodemon
> node bin/postinstall || exit 0

npm WARN server@1.0.0 No description
npm WARN server@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"
os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

added 303 packages from 193 contributors in 31.39s

24 packages are looking for funding
  run `npm fund` for details

PS C:\Users\I524909\Music\DevOpsAssignment\all-post-service> cd ..\create-post-service\
PS C:\Users\I524909\Music\DevOpsAssignment\create-post-service> npm i

> nodemon@2.0.15 postinstall C:\Users\I524909\Music\DevOpsAssignment\create-post-service\node_modules\nodemon
> node bin/postinstall || exit 0

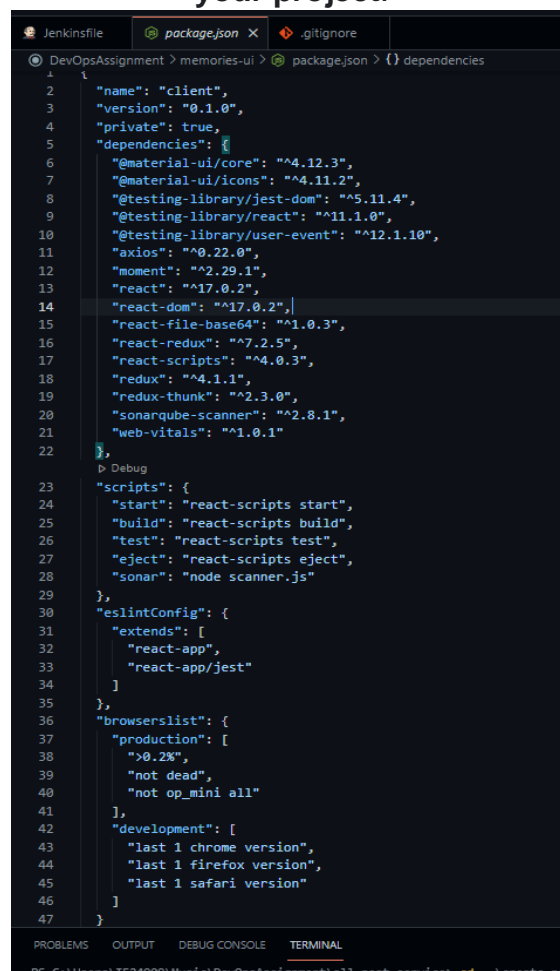
npm WARN server@1.0.0 No description
npm WARN server@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"
os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

added 303 packages from 193 contributors in 28.751s

24 packages are looking for funding

```

NPM is used defines build configurations that apply to all modules in your project.



```

Jenkinsfile package.json .gitignore
DevOpsAssignment > memories-ui > package.json > {} dependencies
1 {
2   "name": "client",
3   "version": "0.1.0",
4   "private": true,
5   "dependencies": {
6     "@material-ui/core": "^4.12.3",
7     "@material-ui/icons": "^4.11.2",
8     "@testing-library/jest-dom": "^5.11.4",
9     "@testing-library/react": "^11.1.0",
10    "@testing-library/user-event": "^12.1.10",
11    "axios": "^0.22.0",
12    "moment": "^2.29.1",
13    "react": "^17.0.2",
14    "react-dom": "^17.0.2",
15    "react-file-base64": "^1.0.3",
16    "react-redux": "^7.2.5",
17    "react-scripts": "^4.0.3",
18    "redux": "^4.1.1",
19    "redux-thunk": "^2.3.0",
20    "sonarqube-scanner": "^2.8.1",
21    "web-vitals": "^1.0.1"
22  },
23  "scripts": {
24    "start": "react-scripts start",
25    "build": "react-scripts build",
26    "test": "react-scripts test",
27    "eject": "react-scripts eject",
28    "sonar": "node scanner.js"
29  },
30  "eslintConfig": {
31    "extends": [
32      "react-app",
33      "react-app/jest"
34    ]
35  },
36  "browserslist": {
37    "production": [
38      ">0.2%",
39      "not dead",
40      "not op_mini all"
41    ],
42    "development": [
43      "last 1 chrome version",
44      "last 1 firefox version",
45      "last 1 safari version"
46    ]
47  }

```

SonarQube for continuous inspection of code quality

The screenshot displays the SonarQube interface for a project named 'memories-ui' on the 'master' branch. The top navigation bar includes links for Overview, Issues, Security Hotspots, Measures, Code, and Activity. A status bar at the top right indicates 'Last analysis had 1 warning' on January 11, 2022, at 10:18 PM, with a note that the version is not provided.

The main dashboard is divided into several sections:

- Failed:** A red box indicating '1 conditions failed'.
- On New Code:** A section showing 'Coverage on New Code is less than 80.0%'.
- New Code:** A section showing 'Since December 26, ... Started 16 days ago'.
- Overall Code:** A section showing various quality metrics:

Metric	Value	Quality Gate
New Bugs	0	Reliability (A)
New Vulnerabilities	0	Security (A)
New Security Hotspots	0	Security Review (A)
Added Debt	0	Maintainability (A)
New Code Smells	0	Maintainability (A)

The 'Issues' tab is selected, showing a list of issues. The first issue is a 'Code Smell' titled 'Default parameters should be last.' located in 'src/reducers/posts.js'. The issue is categorized as 'Major' and 'Open', with a '20min effort' estimate. The issue description states: 'Default parameters should be last. Why is this an issue?'. The issue was created 24 days ago by user 'es2015'.

The code snippet for 'src/reducers/posts.js' is displayed below the issue:

```
1 import { FETCH_ALL, CREATE, UPDATE, DELETE, LIKE } from '../constants/actionTypes';
2
3 export default (posts = [], action) => {
4
5   switch (action.type) {
6     case FETCH_ALL:
7       return action.payload;
8     case LIKE:
9       return posts.map(post => (post._id === action.payload_id ? action.payload : post));
10    case CREATE:
11      return [...posts, action.payload];
12    case UPDATE:
13      return posts.map(post => (post._id === action.payload_id ? action.payload : post));
14    case DELETE:
15      return posts.filter(post => post._id !== action.payload_id);
16    default:
17      return posts;
18   }
19 };
```

SELENIUM

```

memories-ui > JS test.js > driver
1  const {Builder, By, Key, Util} = require("selenium-webdriver")
2  require("chromedriver")
3  let driver = new Builder().forBrowser("chrome").build();
4
5  async function launchBrowser(){
6      driver.manage().window().maximize();
7      driver.manage().setTimeouts({implicit: 30000 })
8      await driver.get("http://localhost:3000/");
9  }
10
11  async function createMemory(){
12      await driver.findElement(By.name('creator')).sendKeys('abcd', Key.ENTER);
13      await driver.findElement(By.name('title')).sendKeys('memory-image', Key.ENTER);
14      await driver.findElement(By.name('message')).sendKeys('this is my memory', Key.ENTER);
15      await driver.findElement(By.name('tags')).sendKeys('def', Key.ENTER);
16  }
17
18  async function navigateToHome(){
19      driver.manage().setTimeouts({implicit: 4000 })
20      driver.navigate().to("http://localhost:3000/");
21  }
22
  
```

Jenkins for CI/CD

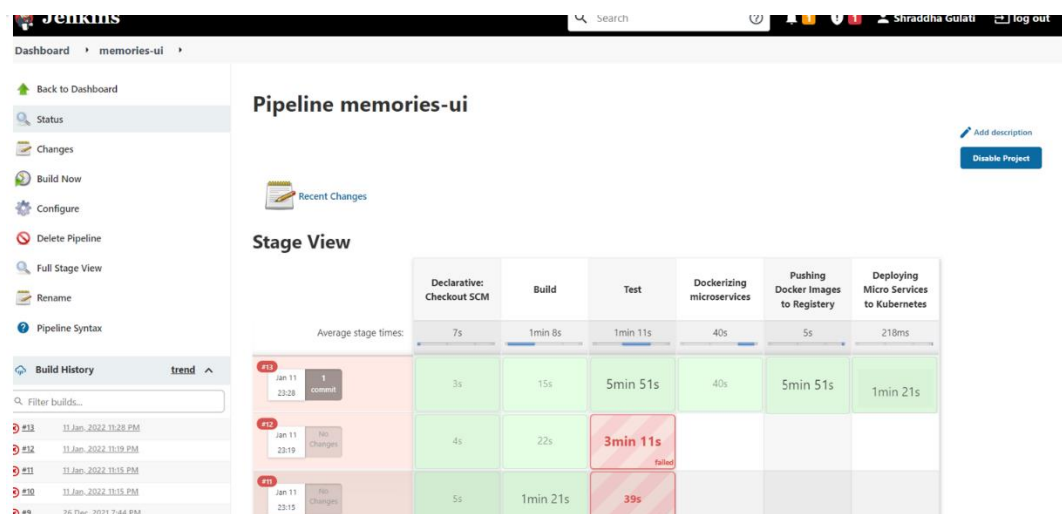


Welcome to Jenkins!

Username

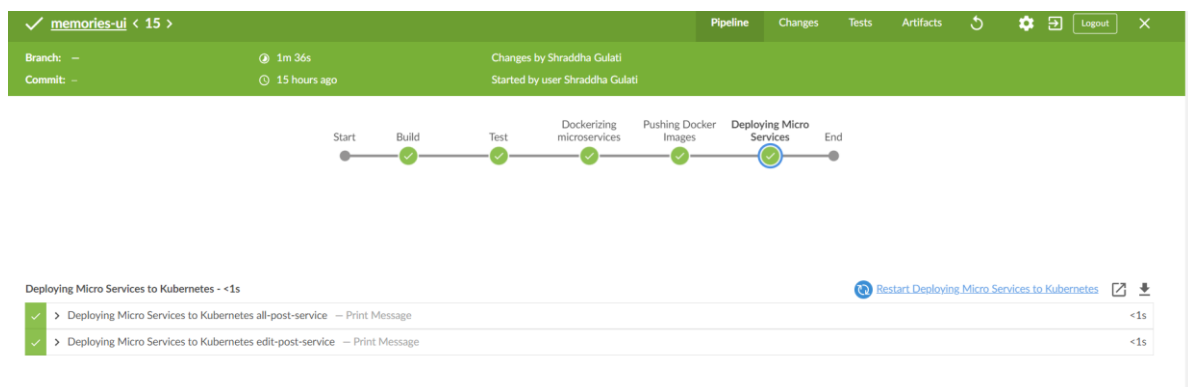
Password

☐ Keep me signed in

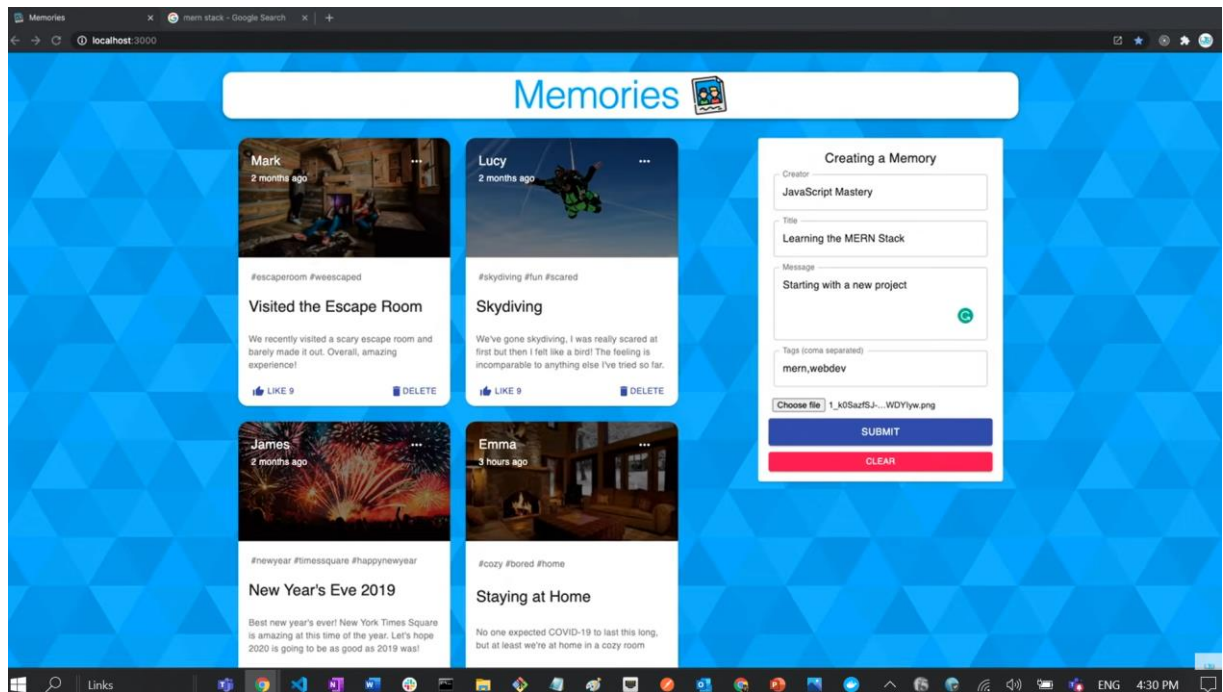


```
Jenkinsfile X package.json .gitignore
DevOpsAssignment > Jenkinsfile
1 def modules = ['all-post-service', 'edit-post-service', 'create-post-service', 'like-post-service']
2 pipeline {
3     agent any
4     stages {
5         stage('Build') {
6             steps {
7                 script {
8                     modules.each { item ->
9                         dir("${item}") {
10                            print("Building Code for $item")
11                            bat "npm install"
12                        }
13                    }
14                }
15            }
16        }
17
18        stage('Test') {
19            steps {
20                script {
21                    modules.each { item ->
22                        dir("${item}") {
23                            print("Testing Code for $item")
24                            bat "npm run sonar"
25                        }
26                    }
27                }
28            }
29        }
30        stage('Dockerizing'){
31            steps {
```

Downloaded Blue Ocean Plugin to get a better picture



Application



PART B

TOOLS USED

1. Build on CI CD pipeline

Jenkins is a continuous integration (CI) server, or a tool written in java language. It is open-source software for download. The continuous integration services can be provided for software development which can be done via command line or web application server.

2. Web Cloud Application with Micro Services

A microservice **attempts to address a single concern**, such as a data search, logging function, or web service function. This approach increases flexibility—for example, updating the code of a single function without having to refactor or even redeploy the rest of the microservices architecture.

3. Deploy using Docker

While most Go applications compile to a single binary, web applications also ship with templates, assets, and configuration files; these can get out of sync and cause faulty deployments. Docker lets us create a self-contained image with everything our application needs to work. In this tutorial, you will learn how to deploy a Go web application with Docker, and how Docker can help improve your development workflow and deployment process.

4. ELK

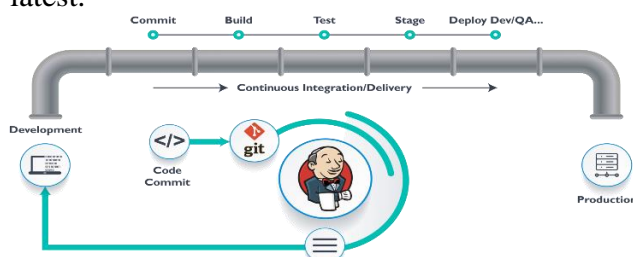
ELK is a **log management platform** that helps DevOps engineers in making better decisions for the company. ELK comprises of Elastic search, Logstash, and Kibana open-source software offered by the elastic company. Logstash collects different types of logs and sends them into a pipeline of events

5. Containerize with Docker

Docker is the containerization platform which is used to package your application and all its dependencies together in the form of containers so to make sure that your application works seamlessly in any environment which can be development or test or production.

6. Manage with Kubernetes

Docker Hub is a public registry managed by Docker, Inc. It centralizes information about organizations, user accounts, and images. It includes a web UI, authentication and authorization using organizations, CLI and API access using commands such as docker login, docker pull, and docker push, comments, stars, search, and more. The Docker Registry is a component of Docker's ecosystem. A registry is a storage and content delivery system, holding named Docker images, available in different tagged versions. For example, the image distribution/registry, with tags 2.0 and latest.



VI. RESULTS AND DISCUSSION

Deployment and Managing Images through Docker and Kubernetes

The image shows a Jenkinsfile configuration and the corresponding Jenkins Pipeline execution for a project named 'memories-ui'.

Jenkinsfile:

```

42     }
43     stage('Pushing Docker Images'){
44         steps {
45             script {
46                 bat "docker login -u heyshraddha -p QAZwsx@#123"
47                 modules.each { item ->
48                     dir("${item}") {
49                         print("Pushing Docker Images to Registry for $item")
50                         bat "docker push heyshraddha/$item "
51                     }
52                 }
53             }
54         }
55     }
56 }
57 stage('Deploying to Kubernetes') {
58     steps {
59         script {
60             bat "kubectl version"
61             modules.each { item ->
62                 print("Deploying Micro Services to Kubernetes $item")
63                 bat "kubectl create deployment $item-cluster --image heyshraddha/$item"
64                 bat "kubectl scale deployment $item-cluster --replicas 2"
65                 bat "kubectl expose deployment $item-cluster --type=NodePort --port 3000"
66             }
67             sleep(60);
68             bat "kubectl get services"
69             bat "kubectl get deployment"
70         }
71     }
72 }

```

Jenkins Pipeline Execution:

The pipeline is titled 'memories-ui < 15 >' and shows a successful build. The pipeline consists of the following stages:

- Start
- Build (Completed)
- Test (Completed)
- Dockerizing microservices (Completed)
- Pushing Docker Images to Registry (Completed)
- Deploying Micro Services to Kubernetes (Completed)
- End

The build log for 'Build - 15s' shows the following steps:

- Check out from version control (4s)
- Building Code for all-post-service - Print Message (<1s)
- npm install - Windows Batch Script (7s)
- Building Code for edit-post-service - Print Message (<1s)
- npm install - Windows Batch Script (7s)

```

all-post-service > Dockerfile > FROM
1 FROM node:13-alpine
2
3 WORKDIR /app
4
5 COPY package.json ./
6
7 RUN npm install --production
8
9 COPY . .
10
11 EXPOSE 3000
12
13 CMD node index.js

```

Managing images through Kubernetes and Kubectl

The screenshot shows the Jenkins dashboard on the left and a Windows PowerShell terminal on the right. The Jenkins dashboard displays the 'Build' status for the 'memories-ui' project. The PowerShell terminal shows the execution of 'minikube start' and 'kubectl get services' commands. The 'minikube start' command output includes details about the Kubernetes version (v1.22.3) and the Docker driver. The 'kubectl get services' command output shows a table of services in the 'minikube' namespace.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
all-post-service-cluster	LoadBalancer	10.103.112.97	127.0.0.1	80:31447/TCP	134m
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	4h7m

Minikube for setting up and getting the internal and external IP to access the application through image

```

$ C:\Test> minikube service all-post-service-cluster

```

NAMESPACE	NAME	TARGET PORT	URL
default	all-post-service-cluster	80	http://192.168.49.2:31447

```

Starting tunnel for service all-post-service-cluster.

```

NAMESPACE	NAME	TARGET PORT	URL
default	all-post-service-cluster		http://127.0.0.1:63041

```

Opening service default/all-post-service-cluster in default browser...
Because you are using a Docker driver on windows, the terminal needs to be open to run it.
Stopping tunnel for service all-post-service-cluster.
$ C:\Test> kubectl get services

```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
all-post-service-cluster	LoadBalancer	10.103.112.97	<pending>	80:31447/TCP	6m48s
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	119m

```

$ C:\Test> minikube tunnel
0111 21:47:04.767544 15500 ssh_tunnel.go:82] error listing ingresses: the server could not find the requested resource
Access to ports below 1024 may fail on Windows with OpenSSH clients older than v8.1. For more information, see: https://minikube.sigs.k8s.io/docs/handbook/accessing/#access-to-ports-1024-n-windows-requires-root-permission
Starting tunnel for service all-post-service-cluster.
0111 21:47:07.132486 15500 ssh_tunnel.go:82] error listing ingresses: the server could not find the requested resource

```

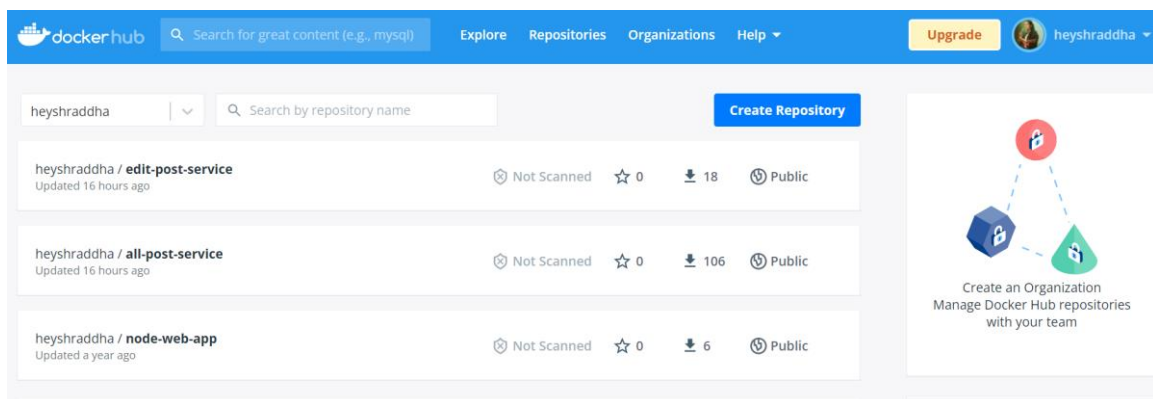
```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL
error: failed to create deployment: Deployment.apps "allPostServiceCluster" is invalid: metadata.name: Invalid value: "allPostServiceCluster": a DNS-1123 subdomain must consist of lower case alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character (e.g. 'example.com', regex used for validation is '[a-z0-9]([-a-z0-9]*[a-z0-9])?(\.[a-z0-9]([-a-z0-9]*[a-z0-9])?)*')
PS C:\Users\I524909\Music\Development\Assignment\DevOps\Backend-AllPostService> kubectl create deployment all-post-service-cluster --image shraddha/all-post-service
deployment.apps/all-post-service-cluster created
PS C:\Users\I524909\Music\Development\Assignment\DevOps\Backend-AllPostService> kubectl scale deployment all-post-service-cluster --replicas 2
deployment.apps/all-post-service-cluster scaled
PS C:\Users\I524909\Music\Development\Assignment\DevOps\Backend-AllPostService> kubectl expose deployment all-post-service-cluster --type=NodePort --port 5000
service/all-post-service-cluster exposed
PS C:\Users\I524909\Music\Development\Assignment\DevOps\Backend-AllPostService> kubectl get services
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP  PORT(S)          AGE
all-post-service-cluster  NodePort    10.96.191.212    <none>        5000:31986/TCP   14s
kubernetes            ClusterIP   10.96.0.1        <none>        443/TCP          9m42s
PS C:\Users\I524909\Music\Development\Assignment\DevOps\Backend-AllPostService> kubectl get nodes -o wide
NAME                STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION
minikube             Ready     master   10m   v1.18.3   192.168.49.2   <none>         Ubuntu 20.04 LTS     5.10.60.1-microsoft-standard-ws
l2                   docker://19.3.8
PS C:\Users\I524909\Music\Development\Assignment\DevOps\Backend-AllPostService> ^C
PS C:\Users\I524909\Music\Development\Assignment\DevOps\Backend-AllPostService> 

```



Images got pushed to docker hub



Learnings

- Overcoming the dev versus ops mentality
- Common understanding of Continuous Delivery practices
- Moving from legacy infrastructure & architecture to microservices
- Implementing a test automation strategy
- Dev and Ops toolset clashes
- How DevOps helps team to for a faster shipment
- Learnt about the tools and an area which is not likely to be much explored.

Challenges:

- There was an issue where the <external-ip> was not coming and was always <pending>. We resolved it by connecting to docker hub by “docker login” and hence it was able to pull our images and then it got successfully deployed.
- Minikube was not run the application and it was hard to find out the port on which the application is running. It was solved by running minikube service service-name command.
- Jenkins download was very slow. We resolved it by having Jenkins.msi and not the war file.

VII. CONCLUSION and Future Scope

This proposed work is successfully designed, implemented and tested. DevOps (a portmanteau of development and operations) is a software development methodology that escalates to the amalgamation between software developers and information technology (IT) operation professionals. Its focuses mainly on delivering software product faster and reducing the failure rate of releases to make the product efficient. This system will be helpful for the developers or testers who need to fix the bugs rapidly and want to add extra features to the existing product according to the client requirement. At present DevOps is the most advanced approach in IT industry than waterfall model and agile model. This system can be extended by on boarding the complete project into cloud services like Microsoft azure or Amazon Web Services (AWS) etc. to improve the efficiency. This can also be extended by generating the review reports of the project through Data visualization tools like Power BI or Tableau for better understanding of the project to the client

REFERENCES

- [1]Carmine Giardino, Nicolò Paternoster, Michael Unterkalmsteiner, Tony Gorschek and Pekka Abrahamsson, “Software Development in Startup Companies: The Greenfield Startup Model”, IEEE Transactions on Software Engineering, 2016
- [2]Youssef Bassil, “A simulation Model for the Waterfall Software Development Life Cycle”, International Journal of Engineering & Technology (iJET), ISSN: 2049-3444, Vol. 2, No. 5, 2012
- [3]Dimitris Karagiannis, “Agile Modeling Method Engineering”, Faculty of Computer Science, University of Vienna.
- [4]David P. Harvie, Arvin Agah, “Targeted Scrum: Applying mission command to Agile Software Development”, IEEE Transactions on Software Engineering, 2016.
- [5]Dan Hao, Lu Zhang, Lei Zang, Yanbo Wang, Xingxia Wu, Tao Xie, “To be optimal or not in Test-case prioritization”, IEEE Transactions on Software Engineering, 2016.
- [6]Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, Shin Yoo, “The Oracle Problem in Software Testing: A Survey”, IEEE Transactions on Software Engineering, 2015.
- [7]Lucy Ellen Lwakatare, Pasi Kuvaja and Markku Oivo, “Dimensions of Devops”, Springer International Publishing Switzerland 2015.
- [8]Manish Virmani, “Understanding DevOps & bridging the gap from continuous integration to continuous delivery”, Innovative Computing Technology (INTECH), Fifth International IEEE Conference, 2015.
- [9]S. R. Chidamber ; C. F. Kemerer, “A metric suite for Object Oriented Design”, IEEE Transactions on Software Engineering, 2015.
- [10]Maximilien de Bayser, Leonardo G. Azevedo, Renato Cerqueira, “The case for Devops in scientific applications”, IFIP/IEEE Internat