

International Institute of Information Technology, Bangalore

Software Production Engineering Mini Project

Outgoing Management Portal

TA : Vaibhav Aggarwal

Guide : Prof. B. Thangaraju



Archit Semwal
MT2019026

Mohit Bansal
MT2019065

Shashank Agarwal
MT2019100

CONTENTS

1. Abstract
2. Introduction
3. Workflow Diagram
4. System Configuration
5. Software Development Life Cycle
 - 5.1 Source Code Management
 - 5.2 Build
 - 5.3 Testing
 - 5.4 Docker Artifact
 - 5.5 Continuous Integration
 - 5.6 Continuous Deployment
 - 5.7 Continuous Monitoring
 - 5.8 Webhook
6. Experimental Setup
7. Results and Discussion
8. Future work
9. Conclusion
10. References

1. ABSTRACT

Most institutes in our country become home to a large number of scholars every year. These widespread universities often have several entry and exit points which become the gates of influx to a variety of crowd including hostellers, day-scholars, faculty, staff and visitors. In such scenario, it becomes utterly important to have a record about the passage of people within the college premises. Outgoing Management Portal is a robust webapp which aims to bring the college authorities, students and their guardians under the same roof about the whereabouts of their ward. The portal is primarily developed using DevOps toolchain to store the incoming and outgoing activities of the students and provide timely notification to its respective users.

2. INTRODUCTION

About the web application

'Outgoing Management Portal' is a webapp which majorly capitalizes on bringing the college authorities, students and their guardians together to some extent of information. The webapp provides a common ground for students to apply for a leave or an outpass while it lays out an effective interface for the college/hostel incharge to look through the leave requests and permit accordingly. The web app also serves as a platform for the college security staff to record the incoming/outgoing of the students as well as visitors through it. The application has been programmed to implement an automatically triggered e-mail notifier too.

Scope of the project and features:

- The project efficiently solves the problem statement of managing and storing student outdoor visits along with time details.
- Allows a student to apply for leave and get real time updates on it. As soon as the student files for a leave from his account, a request card is generated at the warden's dashboard for approval. The response is automatically sent via e-mail to the student in real-time.
- Automates a warning e-mail notification whenever a student exceeds the local out pass time limits.
- The portal successfully replaces all kinds of manual entry by the security staff at any gate by providing a common platform to register leaves, record local outings and maintain visitor entries.

Why DevOps?

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.

This speed enables organizations to better serve their customers and compete more effectively in the market. Under a DevOps model, development and operations teams are no longer “siloed.”

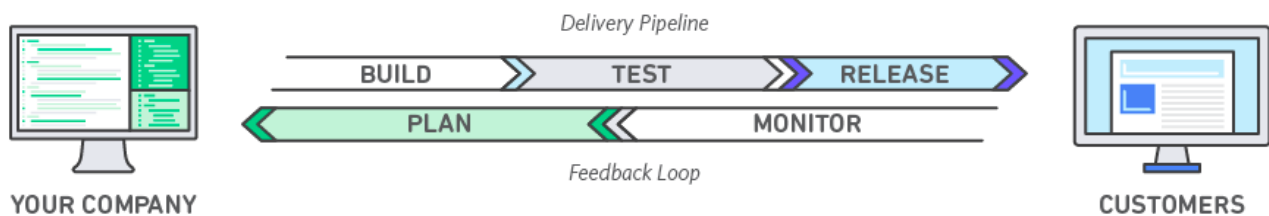


Fig 1: Basic devops model

We used DevOps toolchain in our model due to its various advantages as elaborated ahead.

Speed - Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results.

Rapid Delivery - Increase the frequency and pace of releases so you can innovate and improve your product faster. The quicker you can release new features and fix bugs, the faster you can respond to your customers’ needs and build competitive advantage.

Reliability - Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for end users.

Scale - Operate and manage your infrastructure and development processes at scale. For example, infrastructure as code helps you manage your development, testing, and production environments in a repeatable and more efficient manner.

Improved Collaboration - Build more effective teams under a DevOps cultural model, which emphasizes values such as ownership and accountability. Developers and operations teams collaborate closely, share many responsibilities, and combine their workflows. This reduces inefficiencies and saves time.

3. Workflow Diagram

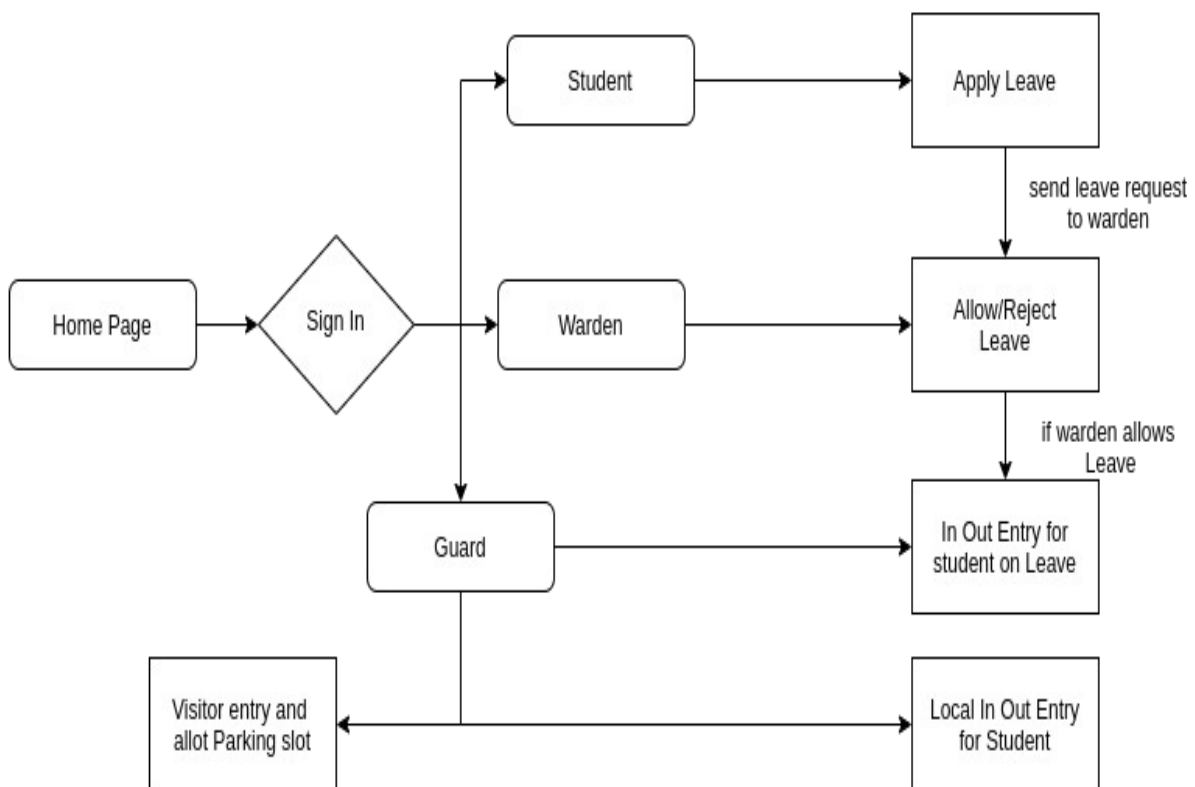


Fig 2: Workflow diagram of Portal

4.SYSTEM CONFIGURATION

System Architecture:

Ubuntu version – 16:04 LTS

CPU configuration – 4 cores

RAM – 4GB

HDD –120GB

kernel version –4.15.0-96-generic

Project Architecture:

Frontend Framework – HTML/CSS, Bootstrap

Backend Framework – Nodejs

Database – MySql

Server – Docker Container

Github Repo – <https://github.com/m0hitbansal/Outgoing-Managment-Portal.git>

DockerHub Repo – <https://hub.docker.com/repository/docker/m0hitbansal/outgoing-webapp>

DevOps Toolchain:

SCM – Git

Build – npm

Continuous Integration – Jenkins

Containerization Tool – Docker

Continuous Delivery – Rundeck

Continuous Testing – Chai, Mocha

Monitoring & Logging – ELK Stack (Winston ,Elasticsearch, Kibana)

5. Software development Life Cycle

5.1 Source Code Management

GitHub is used for SCM. It is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

GIT has various benefits as VCS (Version Control System) listed as follows :

- 1) Revert the code files back to their previous state.
- 2) Recall and revert the entire project back to its previous state.
- 3) Compare code changes over specific durations of time.
- 4) Find who last modified a piece of code that might be causing an issue or a problem. Who introduced a particular issue and when.

Advantages of using git:

- When multiple people collaborate on a project, it's hard to keep track revisions who changed what, when, and where those files are stored. GitHub takes care of this problem by keeping track of all the changes that have been pushed to the repository.
- Git branching model lets you have multiple local branches which are independent of each other. Having this also enables you to have friction-less context switching.

URL of the GIT repository:

GitHub: <https://github.com/m0hitbansal/Outgoing-Managment-Portal.git>

Steps used to add code to my git repository:

- i) `git init ./`
- ii) `git remote add origin https://github.com/m0hitbansal/Outgoing-Managment-Portal.git`
- iii) `git push -f origin <branch name>`

To create a new Project, we can do a git clone:

```
$ git clone https://github.com/m0hitbansal/Outgoing-Managment-Portal.git
```

To push the code on to repository follow following commands:

- i) git add .
- ii) git commit -m "Commit message name"
- iii) git push origin <branch name>

The screenshot shows the GitHub interface for the repository 'm0hitbansal / Outgoing-Managment-Portal'. At the top, there are buttons for 'Unwatch', 'Star' (0), and 'Fork' (0). Below this is a navigation bar with links to 'Code', 'Issues' (0), 'Pull requests' (0), 'Actions', 'Projects' (0), 'Wiki', 'Security' (0), 'Insights', and 'Settings'. The main content area displays the repository name and a description: 'College Gate In-Out Portal and leave from home application portal'. Below this, there are statistics: '55 commits', '4 branches', '0 packages', '0 releases', and '2 contributors'. A section for 'Branch: master' includes a 'New pull request' button and a 'Clone or download' button. The file list shows various files and folders with their commit messages and timestamps. The files listed are: 'db' (visitor page working, 3 days ago), 'node_modules' (elk and logs, yesterday), 'public' (html changes, 18 hours ago), 'test' (server file for local sql add, 4 days ago), '.gitattributes' (Update .gitattributes, 9 days ago), 'Dockerfile' (dockerfile update, 6 days ago), 'README.md' (Initial commit, 26 days ago), 'combined.log' (html changes, 18 hours ago), 'follow_comands.txt' (server file changes, 22 hours ago), 'future_work.txt' (docker file add and docker integration, 6 days ago), and 'jenkinsfile' (jenkins file add, 4 days ago).

File/Folder	Commit Message	Time Ago
db	visitor page working	3 days ago
node_modules	elk and logs	yesterday
public	html changes	18 hours ago
test	server file for local sql add	4 days ago
.gitattributes	Update .gitattributes	9 days ago
Dockerfile	dockerfile update	6 days ago
README.md	Initial commit	26 days ago
combined.log	html changes	18 hours ago
follow_comands.txt	server file changes	22 hours ago
future_work.txt	docker file add and docker integration	6 days ago
jenkinsfile	jenkins file add	4 days ago

Fig 3: Github Repository

5.2 Build

NPM with Node.js

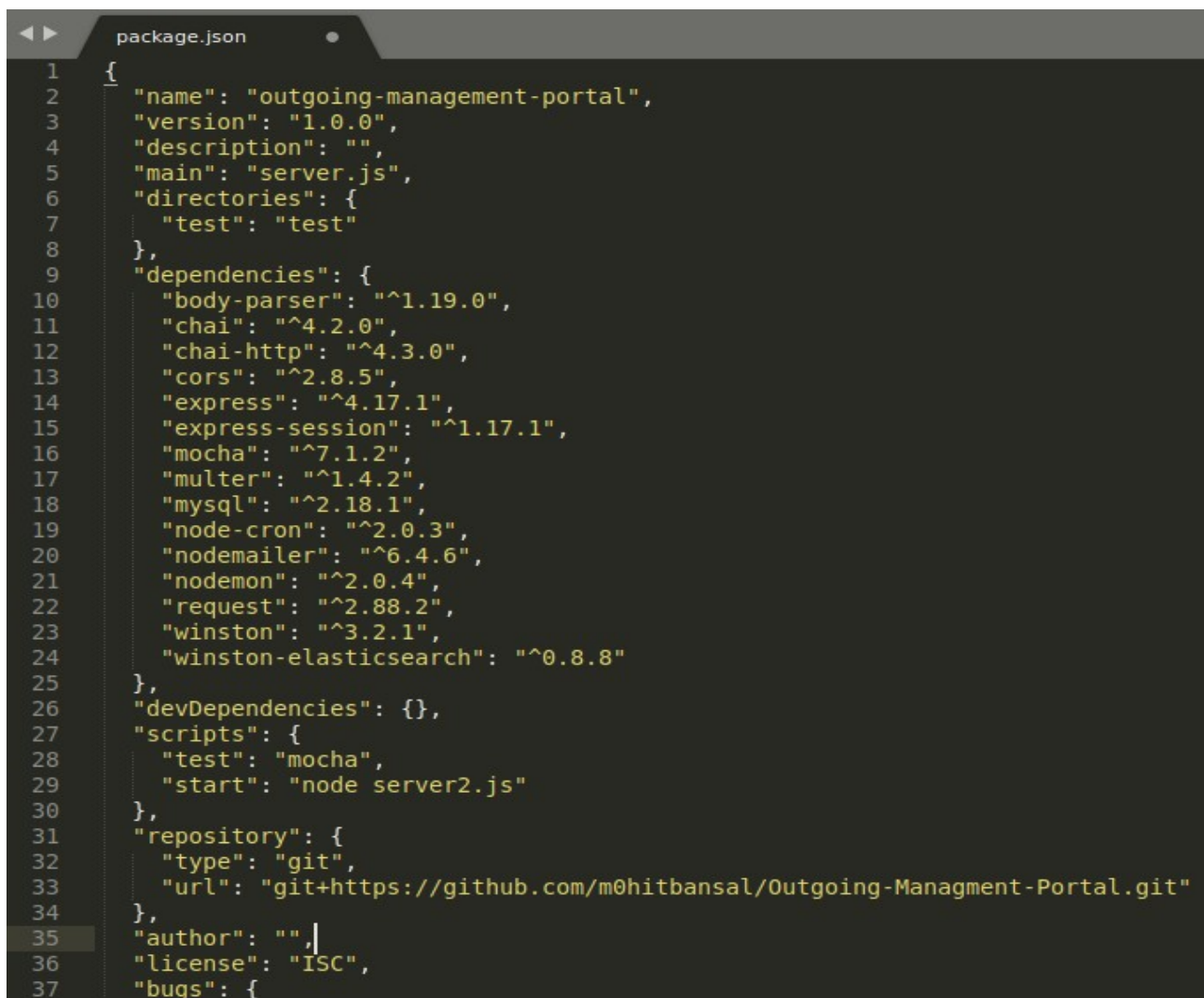
NPM is the default dependency management tool for node applications and is often used as a simple build tool as well. It works with all the node modules and with a single command, can fetch all the dependencies

`$ npm install`

and then build the project using

`$ npm build`

Again, it is extremely flexible and can easily be setup with 'package.json' file. It can be easily configured to work with express.

A screenshot of a code editor showing a package.json file. The file is named 'package.json' and is located in a directory. The content of the file is as follows:

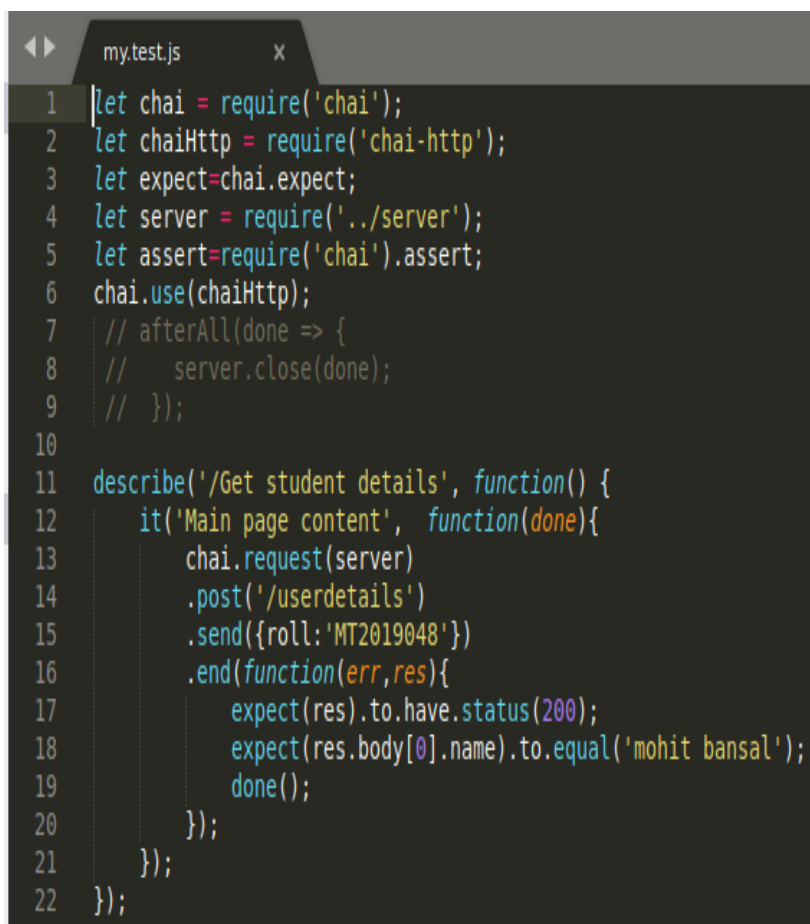
```
1 {
2   "name": "outgoing-management-portal",
3   "version": "1.0.0",
4   "description": "",
5   "main": "server.js",
6   "directories": {
7     "test": "test"
8   },
9   "dependencies": {
10    "body-parser": "^1.19.0",
11    "chai": "^4.2.0",
12    "chai-http": "^4.3.0",
13    "cors": "^2.8.5",
14    "express": "^4.17.1",
15    "express-session": "^1.17.1",
16    "mocha": "^7.1.2",
17    "multer": "^1.4.2",
18    "mysql": "^2.18.1",
19    "node-cron": "^2.0.3",
20    "nodemailer": "^6.4.6",
21    "nodemon": "^2.0.4",
22    "request": "^2.88.2",
23    "winston": "^3.2.1",
24    "winston-elasticsearch": "^0.8.8"
25  },
26  "devDependencies": {},
27  "scripts": {
28    "test": "mocha",
29    "start": "node server2.js"
30  },
31  "repository": {
32    "type": "git",
33    "url": "git+https://github.com/m0hitbansal/Outgoing-Management-Portal.git"
34  },
35  "author": "",
36  "license": "ISC",
37  "bugs": {
```

Fig 4: package.json file

5.3 Testing

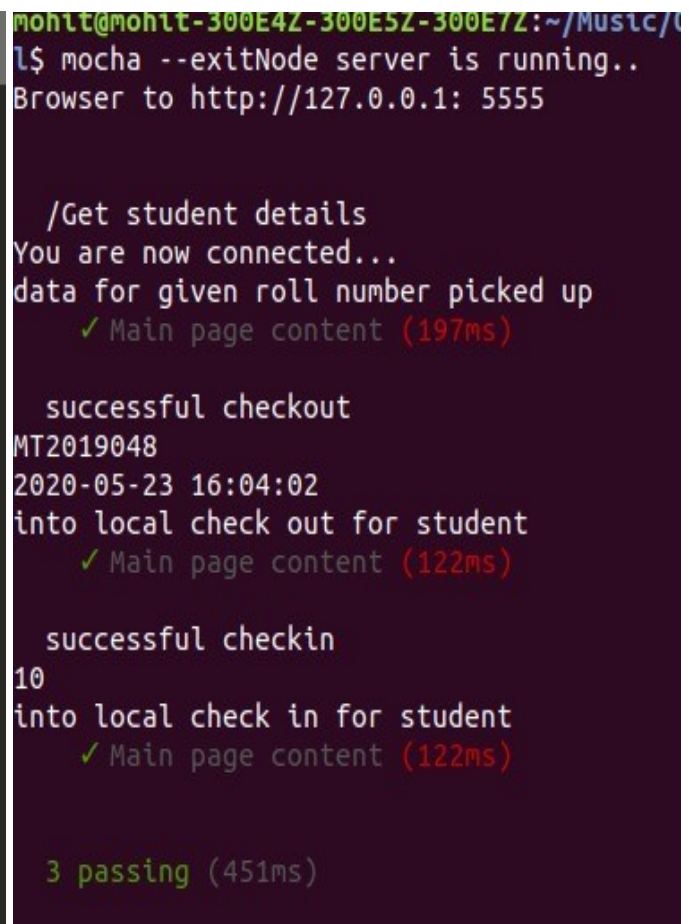
Mocha

Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple and fun. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases. The main advantage with Mocha is that it can seamlessly integrate with your node application, and supports various assertion libraries.



```
1 let chai = require('chai');
2 let chaiHttp = require('chai-http');
3 let expect = chai.expect;
4 let server = require('../server');
5 let assert = require('chai').assert;
6 chai.use(chaiHttp);
7 // afterAll(done => {
8 //   server.close(done);
9 // });
10
11 describe('/Get student details', function() {
12   it('Main page content', function(done) {
13     chai.request(server)
14       .post('/userdetails')
15       .send({roll: 'MT2019048'})
16       .end(function(err, res) {
17         expect(res).to.have.status(200);
18         expect(res.body[0].name).to.equal('mohit bansal');
19         done();
20       });
21   });
22 });
```

Fig 5: Test file



```
mohit@mohit-300E42-300E52-300E72:~/Music/
l$ mocha --exitNode server is running..
Browser to http://127.0.0.1: 5555

/Get student details
You are now connected...
data for given roll number picked up
  ✓ Main page content (197ms)

successful checkout
MT2019048
2020-05-23 16:04:02
into local check out for student
  ✓ Main page content (122ms)

successful checkin
10
into local check in for student
  ✓ Main page content (122ms)

3 passing (451ms)
```

Fig 6: Test file execution

To run this test file we use command

```
$mocha --exit
```

Chai

Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any javascript testing framework.

Why chai?

Actually Mocha provide the environment for making our test, But we need to test our API and our API using http calls like GET, PUT, DELETE, POST etc. So we need a assertion library to fix this challenge. Chai helps us to determine the output of this test case.

5.4 Docker Artifact

Docker is officially defined as “A set of platform as a service products that uses OS-level virtualization to deliver software in packages called containers.” A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

A Dockerfile is a text file written in an easy-to-understand syntax that includes the instructions to build a Docker *image*.

We used Docker for setting up Mysql, Frontend and Backend (NodeJS) of our application.

```
1 FROM node:latest
2
3 WORKDIR /Outgoing
4
5 ADD . /Outgoing
6
7 EXPOSE 5555
8 ENTRYPOINT ["node","server2.js"]
9
```

Fig 7: Docker file for webapp

```
FROM mysql:5.7
ADD wsl.sql ./
ENV MYSQL_ROOT_PASSWORD root
ENV MYSQL_DATABASE Outgoing
COPY ./wsl.sql /docker-entrypoint-initdb.d/
EXPOSE 3306
```

Fig 8: Docker file for mysql database

Create an account on docker hub and create a repository and the image's name will be <dockerhub-username>/<repository-name>for example, m0hitbansal/outgoing-webapp The following commands are used to create a docker image using dockerfile. (keep the Dockerfile in the same directory)

```
$ su
$ docker build -t m0hitbansal/outgoing-webapp .
$ docker build -t m0hitbansal/outgoing-sql .
$ docker login
```

```
$ docker push m0hitbansal/outgoing-sql
$ docker push m0hitbansal/outgoing-webapp
$ docker run -it --name webserv -d m0hitbansal/outgoing-sql
$ docker run --link webserv:db -e DATABASE_HOST=db --name webapp -p
5555:5555 m0hitbansal/outgoing-webapp
```

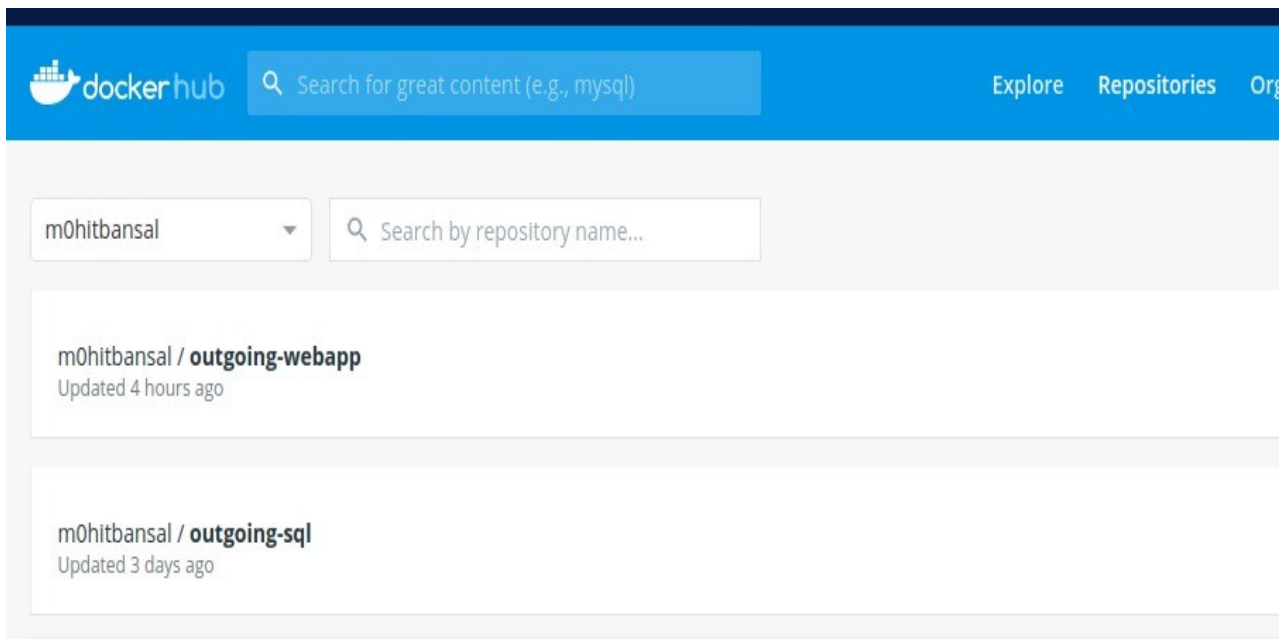


Fig 9: Dockerhub-Repository

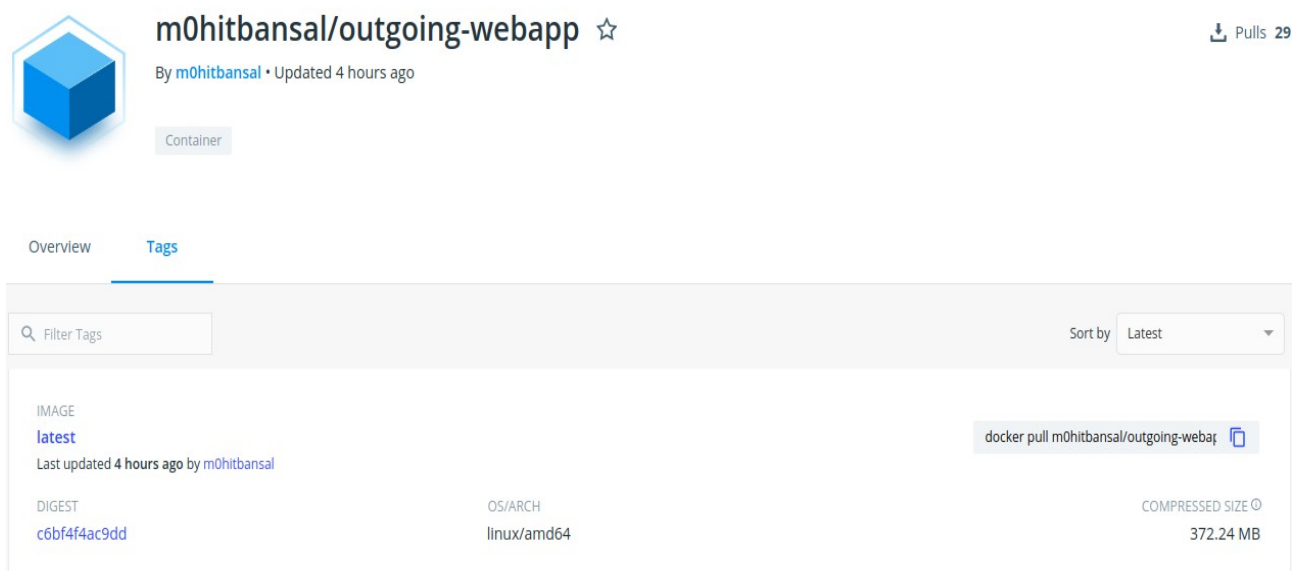


Fig 10: Image on dockerhub Repository

5.5 Continuous Integration

Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated tests. One of the key benefits of integrating regularly is that you can detect errors quickly and locate them more easily.

JENKINS

Jenkins is an open source CI tool written in Java, platform independent and easy to use. The user interface is simple, intuitive and visually appealing. It has a very low learning curve. It is extremely flexible and hundreds of open source plugins are available with more coming out every week. These plugins cover everything from version control systems, build tools, code quality metrics, build notifiers, integration with external systems, UI customizations, and much more.

Advantages of using Jenkins

- Jenkins is being managed by the community which is very open. Every month, they hold public meetings and take inputs from the public for the development of Jenkins project.
- As technology grows, so does Jenkins. So far Jenkins has around 320 plugins published in its plugins database.
- Jenkins also supports cloud-based architecture so that you can deploy Jenkins in cloud-based platforms.

Environment setup - Jenkins

Now we setup Jenkins and other plugins of it.

Make sure to add Jenkins to docker group, so that Jenkins can use docker for build procedure.

```
$ sudo usermod -aG docker Jenkins, you can verify it with
```

```
$ sudo grep jenkins/etc/gshadow
```

Rundeck Configuration in Jenkins

Go to Manage Jenkins → configure system

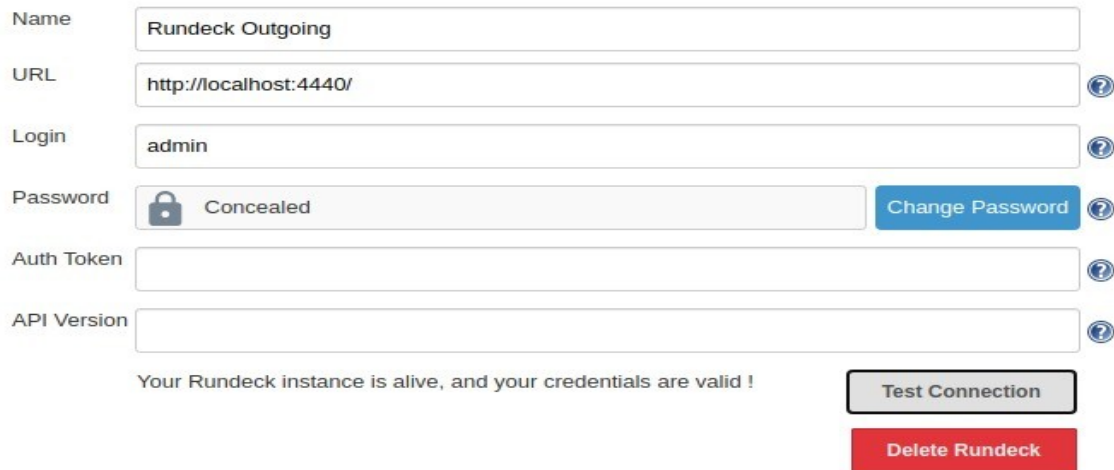
Under Rundeck enter the following configuration:

Name: Rundeck Outgoing

Url: http://localhost:4440

Login: admin

Password: admin



Name:

URL:

Login:

Password: [Change Password](#)

Auth Token:

API Version:

Your RunDeck instance is alive, and your credentials are valid !
 [Test Connection](#)
[Delete RunDeck](#)

Fig 11: RunDeck configuration in jenkins

Docker Configuration

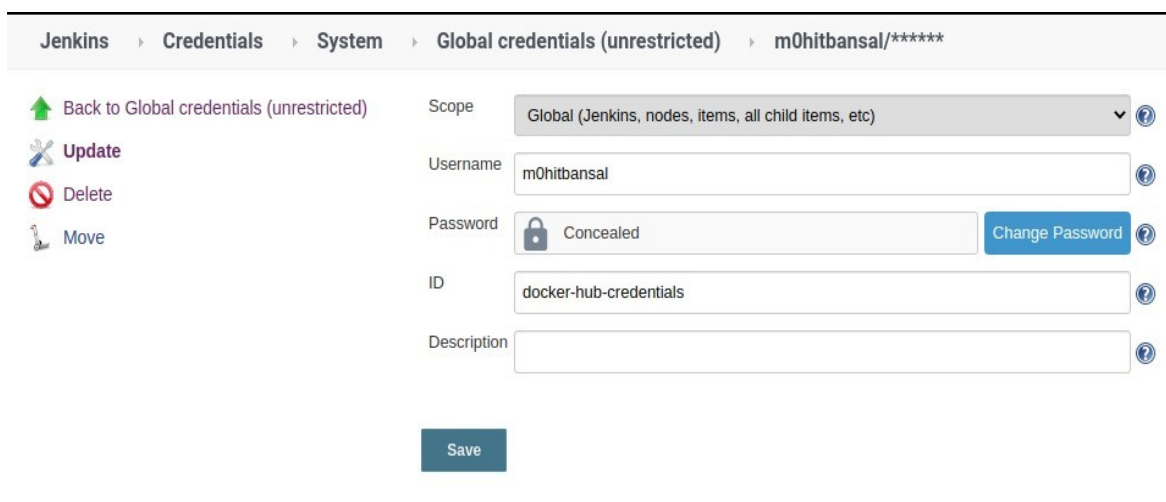
Go to Jenkins → credentials → System → Global Credentials

Under Global Credentials:

Username: m0hitbansal

Password: *****

ID: docker-hub-credentials



Jenkins > Credentials > System > Global credentials (unrestricted) > m0hitbansal/*****

[Back to Global credentials \(unrestricted\)](#)

[Update](#)
[Delete](#)
[Move](#)

Scope:

Username:

Password: [Change Password](#)

ID:

Description:

[Save](#)

Fig 12: Docker configuration in jenkins

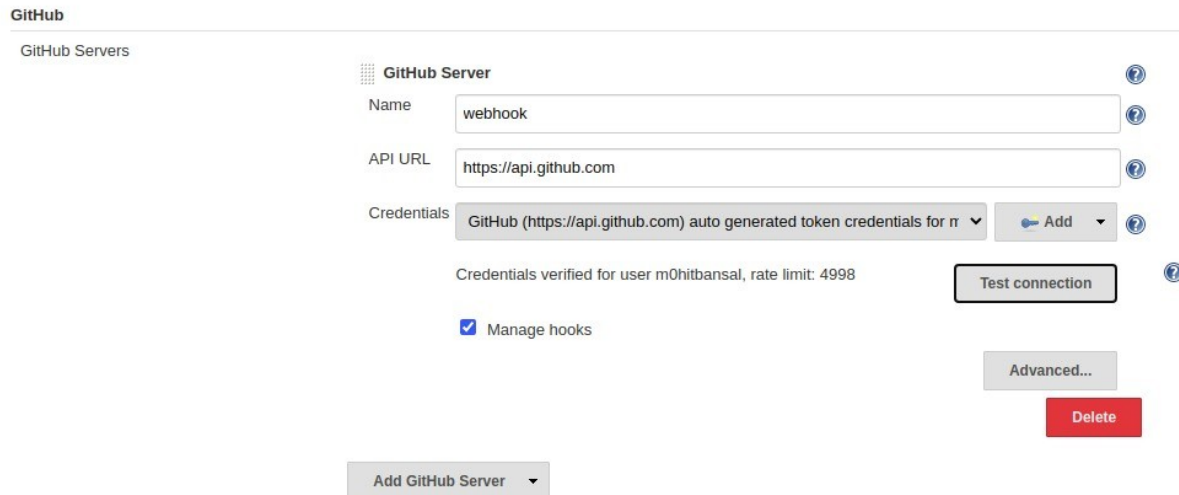
Github webhook Configuration in Jenkins

Go to Manage Jenkins → configure system

Under Github Server:

Name: webhook

API Url: http://api.github.com



The screenshot shows the Jenkins configuration page for a GitHub Server. The page has a sidebar with 'GitHub' and 'GitHub Servers'. The main content area is titled 'GitHub Server' and contains the following fields:

- Name:** A text input field containing 'webhook'.
- API URL:** A text input field containing 'https://api.github.com'.
- Credentials:** A dropdown menu showing 'GitHub (https://api.github.com) auto generated token credentials for m' with an 'Add' button.
- Credentials verified:** A message stating 'Credentials verified for user m0hitbansal, rate limit: 4998'.
- Test connection:** A button to test the connection.
- Manage hooks:** A checkbox that is checked.
- Advanced...** A button to expand advanced options.
- Delete:** A red button to delete the server.
- Add GitHub Server:** A button to add a new server.

Fig 13: Github webhook Configuration in Jenkins

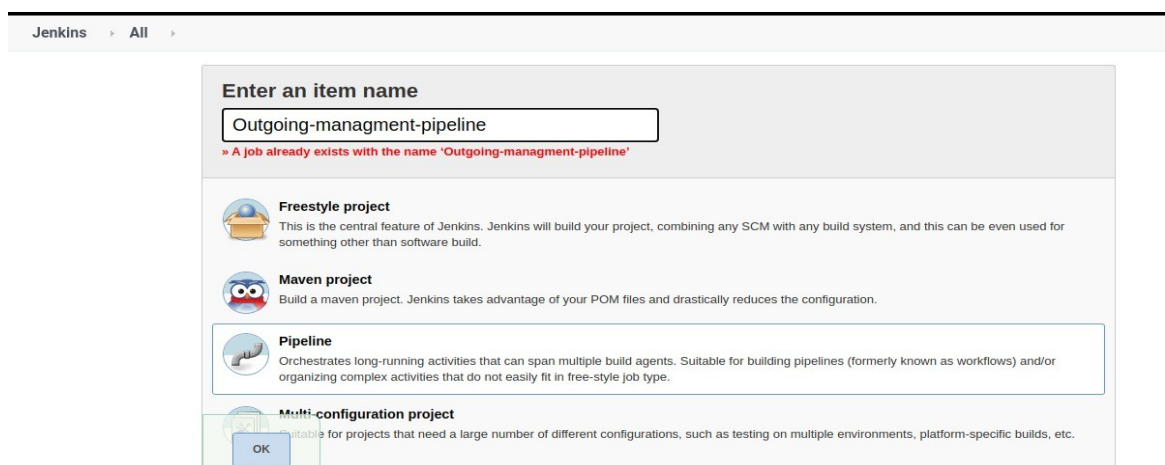
Setting up Jenkins Pipeline

Click on new item

Enter project Name: Outgoing-management-pipeline

Select pipeline project

Click Ok



The screenshot shows the Jenkins 'Enter an item name' dialog. The dialog has a text input field containing 'Outgoing-managment-pipeline'. Below the input field, there is a red error message: '» A job already exists with the name 'Outgoing-managment-pipeline''. Below the error message, there are three project types listed:

- Freestyle project:** This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Maven project:** Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline:** Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project:** Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

At the bottom of the dialog, there is an 'OK' button.

Fig 14: Create New Pipeline project

In project configuration we check github project to pull latest code from github as shown in the figure below.

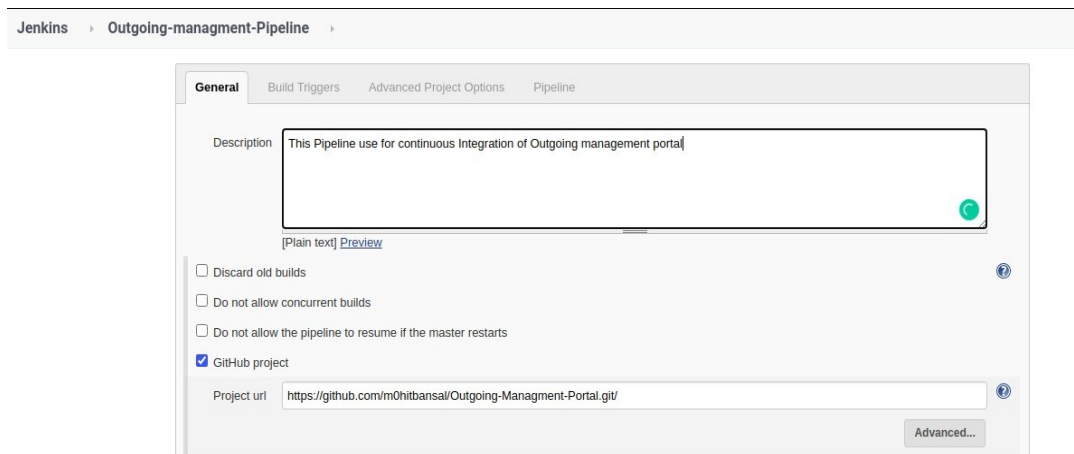


Fig 15: Add Github project

In Build Triggers
Check github hook trigger GITScm Polling

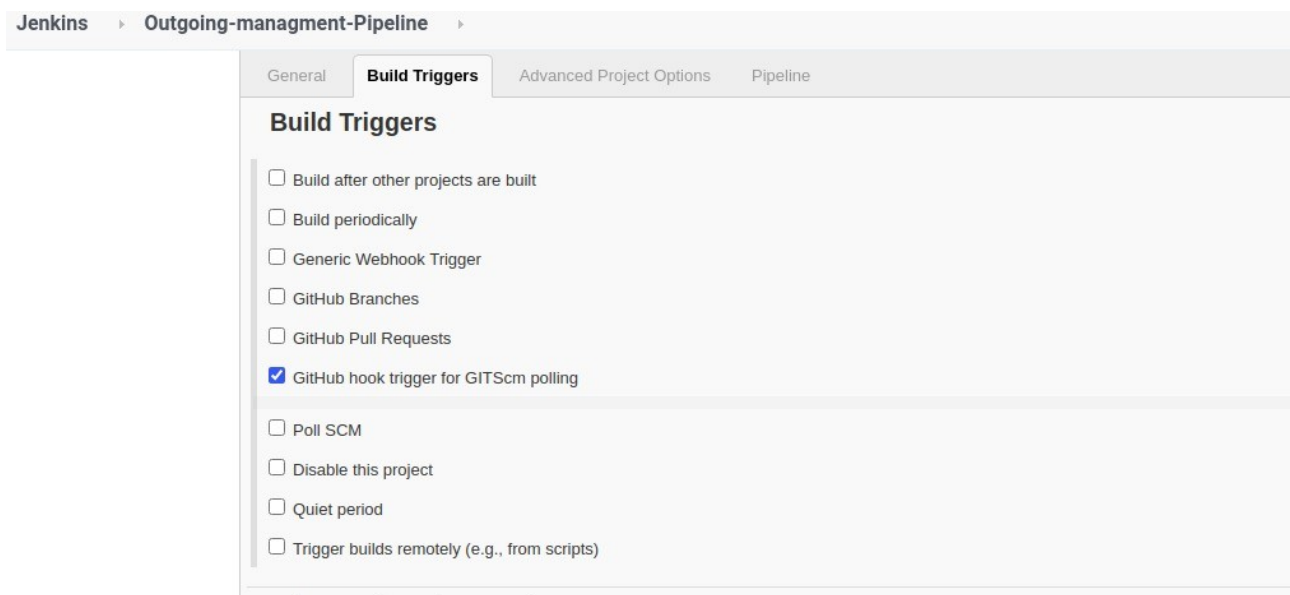


Fig 16: Set Build trigger

In Pipeline stage we write pipeline script as shown in the figure.

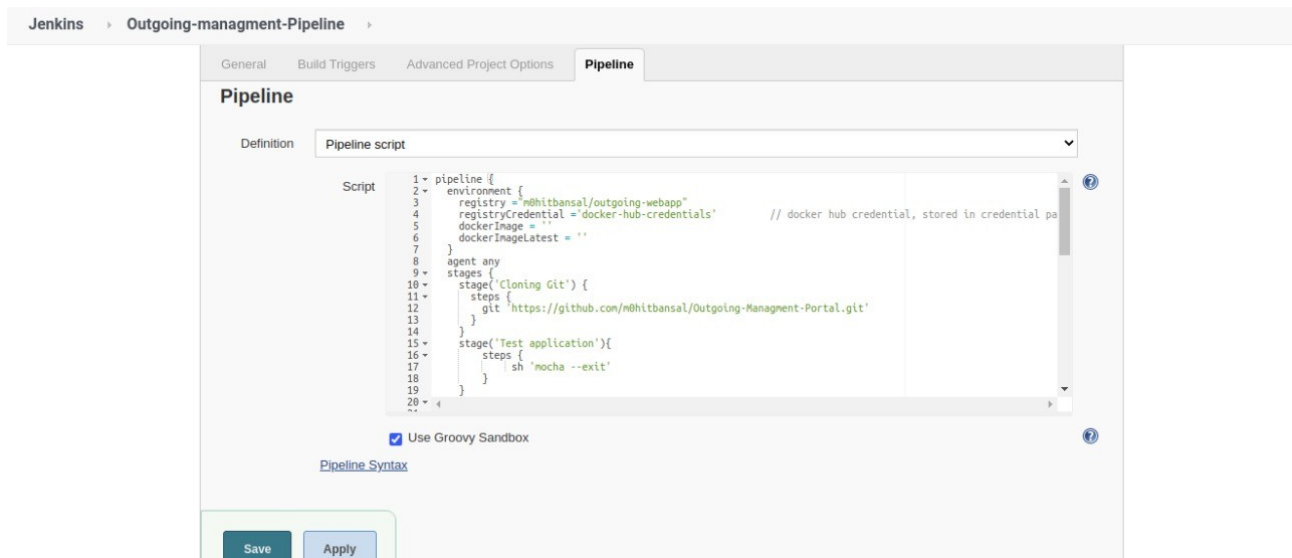


Fig 17: Add Pipeline script

```
1 pipeline {
2   environment {
3     registry="m0hitbansal/outgoing-webapp"
4     registryCredential='docker-hub-credentials' // docker hub crede
5     dockerImage = ''
6     dockerImageLatest = ''
7   }
8   agent any
9   stages {
10    stage('Cloning Git') {
11      steps {
12        git 'https://github.com/m0hitbansal/Outgoing-Managment-Portal.git'
13      }
14    }
15    stage('Test application'){
16      steps {
17        sh 'mocha --exit'
18      }
19    }
20    stage('Building image') {
21      steps{
22        script {
23          dockerImageLatest = docker.build registry + ":latest"
24        }
25      }
26    }
```

Fig 18: Pipeline script-1

```
27    stage('Deploy Image') {
28      steps{
29        script {
30          docker.withRegistry( '', registryCredential ) {
31            dockerImageLatest.push()
32          }
33        }
34      }
35    }
36    stage('Remove Unused docker image') {
37      steps{
38        sh "docker rmi $registry:latest"
39      }
40    }
41    stage('Execute Rundeck job') {
42      steps {
43        script {
44          step([$class: "RundeckNotifier",
45            includeRundeckLogs: true,
46            jobId: "8eb64841-d14e-4c86-9a7e-a3ea08dfd0cd",
47            rundeckInstance: "Rundeck Outgoing",
48            shouldFailTheBuild: true,
49            shouldWaitForRundeckJob: true,
50            tailLog: true])
51        }
52      }
53    }
54  }
55 }
```

Fig 19: Pipeline script-2

Explanation of the pipeline script

- 1) We initially declare the environment variable which contains dockerhub credentials and dockerhub repository name.
- 2) In the first stage we add the github project url to clone the project.
- 3) In the second stage we build project and run the test file using mocha.
- 4) In the third stage the docker image is build. The name of the image can be generic or declared in environment variables.
- 5) In the fourth stage the image is deployed on DockerHub. Here, we use the *registryCredential* we configured in the previous slide and then push the image to the repository <dockerHub Username>/<repository name>.
- 6) In the fifth stage we remove the same unused image from the docker.
- 7) Jenkins can handle the builds for the continuous integration cycle of development and triggering of Rundeck is required to control distributed orchestration across the deployment. In the sixth stage we called rundeck job using job-id as shown in the pipeline script.
- 8) After writing the script click on save and apply button.
- 9) Click on build now.

```
nt-Pipeline  #20
> git --version # timeout=10
> git fetch --tags --progress -- https://github.com/m0hitbansal/Outgoing-Mi
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision d3af1ff2be352493c60f24a5ac23b0401d3c5037 (refs/remote:
> git config core.sparsecheckout # timeout=10
> git checkout -f d3af1ff2be352493c60f24a5ac23b0401d3c5037 # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git branch -D master # timeout=10
> git checkout -b master d3af1ff2be352493c60f24a5ac23b0401d3c5037 # timeout=
Commit message: "Merge pull request #14 from m0hitbansal/archit"
> git rev-list --no-walk 16dfee49ale0b49668a1d921383a4bb550d6c0cf # timeout=
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test application)
[Pipeline] sh
+ mocha --exit
Node server is running..
Browser to http://127.0.0.1: 5555

/Get student details
You are now connected...
data for given roll number picked up
  ✓ Main page content (268ms)

successful checkout
MT2019048
2020-05-23 12:28:47
into local check out for student
  ✓ Main page content (116ms)

successful checkin
9
into local check in for student
  ✓ Main page content (69ms)

3 passing (466ms)
```

Fig 21: Console Output-1

```
[12:37:09] [NORMAL] Error: No such image: m0hitbansal/outgo
[12:37:09] [NORMAL] Using default tag: latest
[12:37:13] [NORMAL] latest: Pulling from m0hitbansal/outgoi
[12:37:13] [NORMAL] Digest: sha256:3032612e13b51d9b0f88c573
[12:37:13] [NORMAL] Status: Downloaded newer image for m0hi
[12:37:13] [NORMAL] docker.io/m0hitbansal/outgoing-sql:late
[12:37:14] [NORMAL] Using default tag: latest
[12:37:17] [NORMAL] latest: Pulling from m0hitbansal/outgoi
[12:37:17] [NORMAL] 1c6172af85ee: Already exists
[12:37:17] [NORMAL] b194b0e3c928: Already exists
[12:37:17] [NORMAL] 1f5ec00f35d5: Already exists
[12:37:17] [NORMAL] 93b1353672b6: Already exists
[12:37:18] [NORMAL] 3d7f38db3cca: Already exists
[12:37:18] [NORMAL] 21e102f9fe89: Already exists
[12:37:18] [NORMAL] e63dac87cb8e: Already exists
[12:37:18] [NORMAL] 2e15f38664d9: Already exists
[12:37:18] [NORMAL] 7639f95d3d43: Already exists
[12:37:18] [NORMAL] ba53c21961fb: Pulling fs layer
[12:37:18] [NORMAL] 8aeb23a843f1: Pulling fs layer
[12:37:20] [NORMAL] ba53c21961fb: Verifying Checksum
[12:37:20] [NORMAL] ba53c21961fb: Download complete
[12:37:21] [NORMAL] ba53c21961fb: Pull complete
[12:37:42] [NORMAL] 8aeb23a843f1: Verifying Checksum
[12:37:42] [NORMAL] 8aeb23a843f1: Download complete
[12:37:47] [NORMAL] 8aeb23a843f1: Pull complete
[12:37:47] [NORMAL] Digest: sha256:c6bf4f4ac9dd4be879729578
[12:37:47] [NORMAL] Status: Downloaded newer image for m0hi
[12:37:47] [NORMAL] docker.io/m0hitbansal/outgoing-webapp:l
END RUNDECK TAILED LOG OUTPUT
Rundeck execution #110 finished in 40 seconds, with status
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Fig 20: Console Output-2

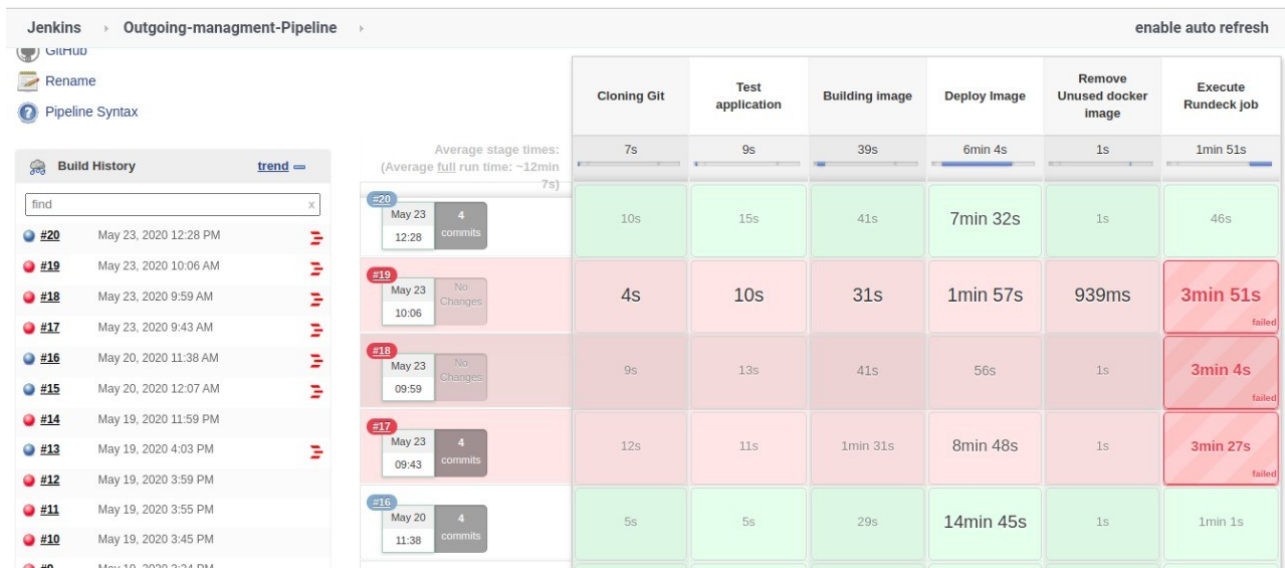


Fig 22: Pipeline Stage View

5.6 Continous Deployment

Continuous deployment is a strategy for software releases wherein any code commit that passes the automated testing phase is automatically released into the production environment, making changes that are visible to the software's users. Continuous deployment eliminates the human safeguards against unproven code in live software.

Rundeck

Rundeck is an open source automation service with a web console, command line tools and a WebAPI. It lets you easily run automation tasks across a set of nodes. RunDeck is cross-platform open source software that helps you automate ad-hoc and routine procedures in data center or cloud environments. RunDeck allows you to run tasks on any number of nodes from a web-based or command-line interface. RunDeck also includes other features that make it easy to scale up your scripting efforts including: access control, workflow building, scheduling, logging, and integration with external sources for node and option data.

Environment setup for Rundeck

To run docker on Rundeck, we need to add Rundeck to the root group.

```
$ usermod -aG docker rundeck
```

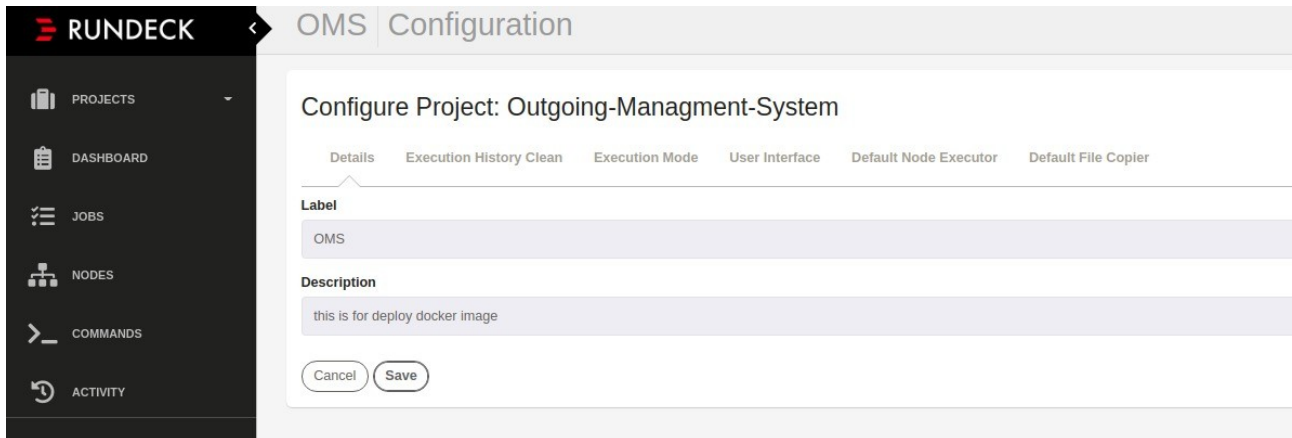
Restart Rundeck i.e.

```
$ systemctl restart rundeckd
```

This way, we can see how rundeck belongs to the same group and now we can run *root* commands in rundeck without using *sudo*.

Creating new Project and job in Rundeck:

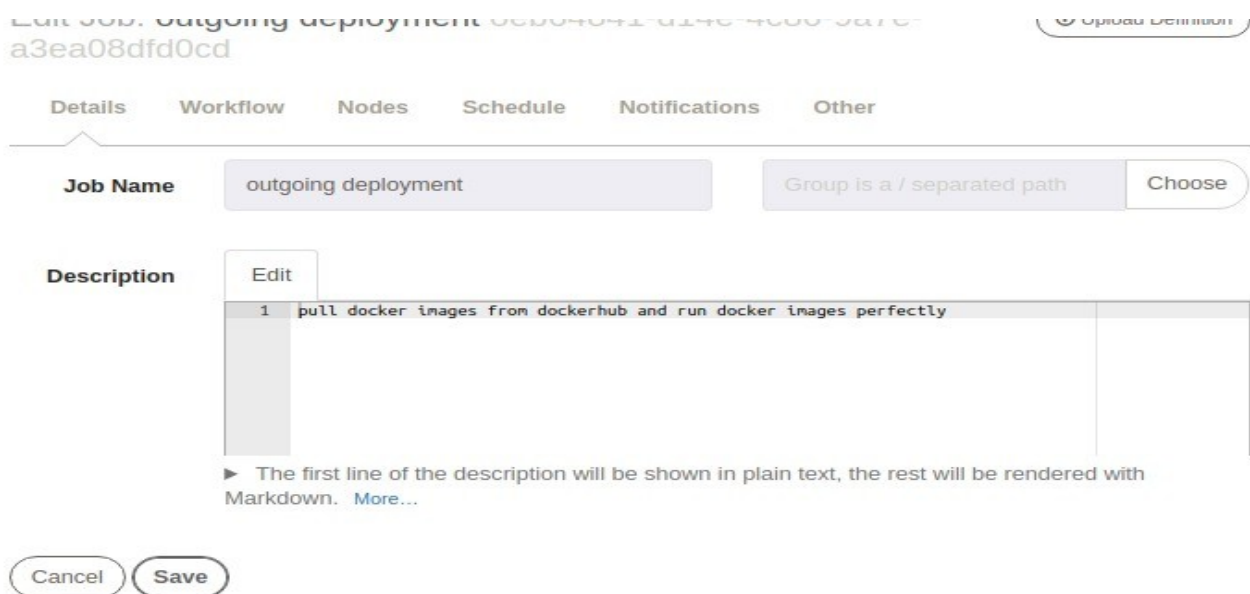
1) go to url and create a project a new project name it Outgoing-management-system and save it.



The screenshot shows the Rundeck web interface. On the left is a dark sidebar with navigation links: PROJECTS, DASHBOARD, JOBS, NODES, COMMANDS, and ACTIVITY. The main area is titled 'Configure Project: Outgoing-Management-System'. Below the title are tabs: Details (selected), Execution History Clean, Execution Mode, User Interface, Default Node Executor, and Default File Copier. The 'Details' tab contains a 'Label' field with the value 'OMS' and a 'Description' field with the text 'this is for deploy docker image'. At the bottom of the form are 'Cancel' and 'Save' buttons.

Fig 23: Create New Rundeck Project

2) Click on add new node source, select file and then select resource.xml as the format of the file. Add file path as `var/lib/rundeck/outgoing.xml` and check generate, include servernode and writable. Click on save button.



The screenshot shows the 'Edit Job' interface for a job named 'outgoing deployment'. The job ID is 'a3ea08dfd0cd'. There are tabs for Details, Workflow, Nodes, Schedule, Notifications, and Other. The 'Details' tab is active. It shows the 'Job Name' as 'outgoing deployment' and a 'Group' field with the value 'Group is a / separated path' and a 'Choose' button. Below this is the 'Description' section, which has an 'Edit' button and a text area containing the text 'pull docker images from dockerhub and run docker images perfectly'. At the bottom of the form are 'Cancel' and 'Save' buttons.

24: Create New Job

Fig

- 3) Create a new job, enter job name and description.
- 4) Go to workflow and add the script to execute on the registered node.

The screenshot displays a web-based interface for defining a job. It is divided into several sections:

- Options:** Includes 'Undo' and 'Redo' buttons, a 'No Options' label, and an '+ Add an option' button.
- Workflow:** Contains a section 'If a step fails:' with two radio buttons: 'Stop at the failed step.' (selected) and 'Run remaining steps before failing.'. Below this is a 'Strategy:' dropdown menu set to 'Node First', followed by the text 'Execute all steps on a node before proceeding to the next node.' and an 'Explain >' link.
- Global Log Filters:** Includes a '+ add' button.
- Script Editor:** A text area with a light blue border and a dark blue header 'Enter the entire script to execute'. It contains a bash script with line numbers 1 through 7:

```
1 #!/bin/bash
2 docker stop $(docker ps -aq)
3 docker rm $(docker ps -aq)
4 docker rmi -f m0hitbansal/outgoing-webapp
5 docker pull m0hitbansal/outgoing-sql
6 docker pull m0hitbansal/outgoing-webapp
7
```

Fig

25: Job definition

- 4) Under Nodes select execute locally because all these commands will be execute on local system. We can specify the specific nodes on which we want commands to execute using dispatch to nodes in node option. Make note of UUID for future reference and save.

5.7 Continuous Monitoring

Elastic Search

Elastic search is a indexing querying over apache's lucene engine.

```
$ cd elasticsearch
```

```
$ ./bin/elasticsearch
```

you can check if elasticsearch is up and running at <http://localhost:9200>

open another terminal and proceed for kibana.

Kibana helps visualizing data using after querying using elastic search.

Kibana

```
$ cd kibana
```

```
$ ./bin/kibana
```

You can access kibana GUI at <http://localhost:5601>

Logstash helps normazling data from various data soruces such as apache, log events, sql and other data sources.

Also, logstash enhances data us varius filters such as geoip, grok, matcher and many more. Open another terminal and proceed for logstash.

Winston

Winston is designed to be a simple and universal logging library with support for multiple transports. A transport is essentially a storage device for your logs.

Advantages of Winston

1) Distributed platforms are fantastic for solving a lot of troubles, such as scaling, high availability, even maintainability of a big code base. But for all the great benefits they provide, they also come with some added baggage you need to take into account when working on one.

2) A scalable logging strategy is exactly what the name implies: you're able to log as much as you need. Just like you can (and should) scale your processing power or your bandwidth when your platform is experiencing a spike on traffic, your logging capabilities should have a similar elasticity.

```

const winston = require('winston');
const Elasticsearch = require('winston-elasticsearch');
var esTransportOpts = {
  level: 'info'
};
const logger = winston.createLogger({
  level: 'info',
  format: winston.format.combine(
    winston.format.timestamp({
      format: 'YYYY-MM-DD HH:mm:ss'
    }),
    winston.format.json()
  ),
  transports: [
    //
    // - Write all logs with level `error` and below to `error.log`
    // - Write all logs with level `info` and below to `combined.log`
    new winston.transports.File({ filename: 'combined.log' }),
    new Elasticsearch(esTransportOpts)
  ]
});

function logstore(caller,method,text){
  id = id +1;
  logger.info({"index":{"index":"Outgoing", "_id":id}, "level":'info', 'message':""});
  logger.info({"type":'api-call', "call_name":caller, "method":method, "text_entry":text});
}

module.exports = app; // for testing

```

Fig 26: Winston logs generate code

The screenshot displays the Kibana Logs Dashboard. On the left, there is a sidebar with a search bar and a list of available fields. The main area shows a list of log entries with 59 hits. The log entries are displayed in a table format with the following fields: @timestamp, _id, _index, message.index.index, message.index._id, message.level, message.message, severity, and _score.

@timestamp	_id	_index	message.index.index	message.index._id	message.level	message.message	severity	_score
2020-05-22T03:37:24.866Z	yQh10nIB1dTLx8dqA8qy	logs-2020.05.22	Outgoing	2	info		info	0
2020-05-22T03:37:24.876Z	ygh10nIB1dTLx8dqA8qy	logs-2020.05.22	api-call	/checkrole	POST	warden login	info	0
2020-05-22T03:37:25.645Z	ywh10nIB1dTLx8dqA8qy	logs-2020.05.22	Outgoing	3	info		info	0
2020-05-22T03:37:25.645Z	zAh10nIB1dTLx8dqA8qy	logs-2020.05.22	api-call	/getLeaves	GET	Fetch all leaves which status is 0	info	0
2020-05-22T03:42:24.334Z	zQh50nIB1dTLx8dqkspN	logs-2020.05.22	Outgoing	2	info		info	0
2020-05-22T03:42:24.347Z	zgh50nIB1dTLx8dqkspN	logs-2020.05.22	api-call	/checkrole	POST	Student login Mohit.Bansal@iiitb.org	info	0
2020-05-22T03:42:25.059Z	zwh50nIB1dTLx8dqkspN	logs-2020.05.22	Outgoing	3	info		info	0

Fig 27: Kibana -1 Logs Dashboard

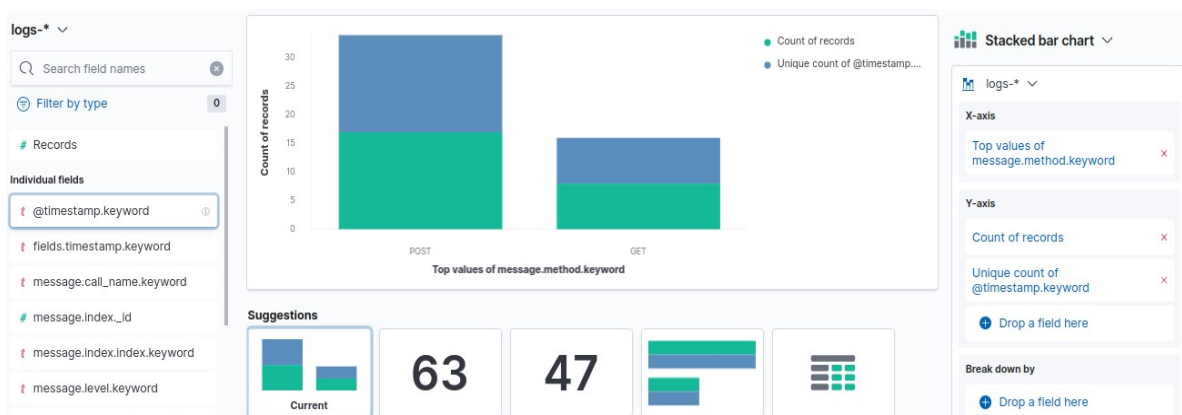


Fig 28: Kibana-2 GET/POST Count logs

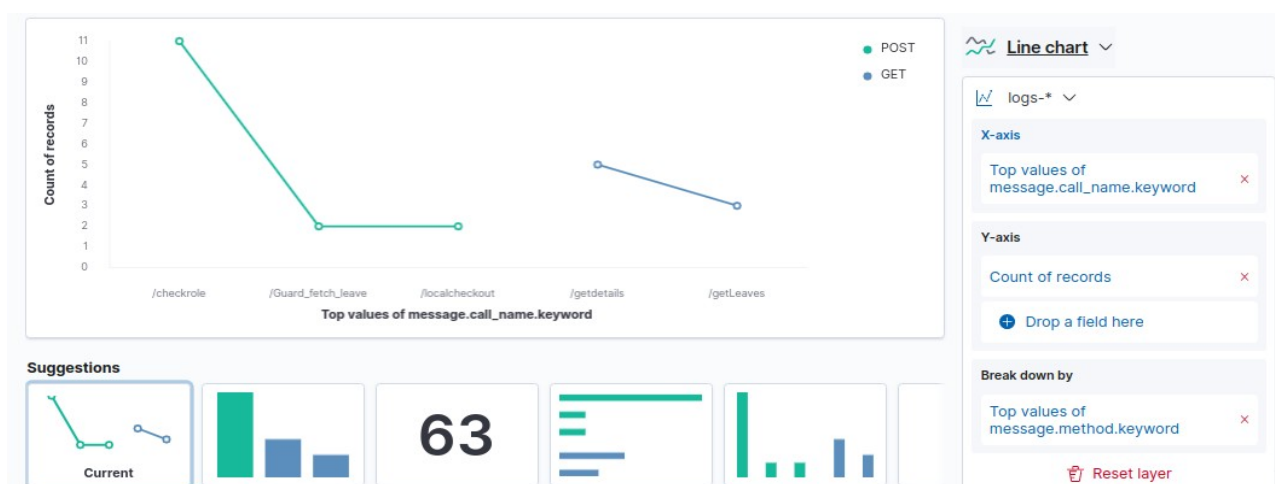


Fig 29: Kibana-3 Get/Post With timestamp

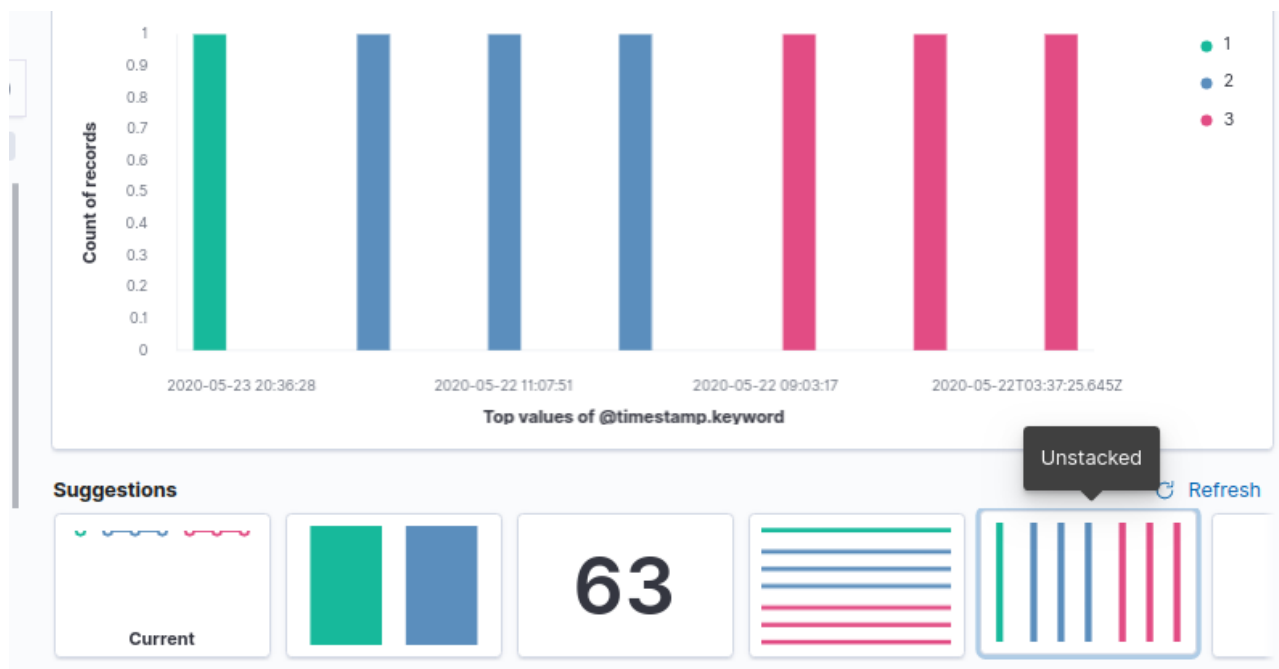


Fig 30: Kibana-4 Index id with time stamp

5.8 Webhooks

- The next problem in the pipeline was that each time, we manually needed to build the pipeline in Jenkins. We can automate this using webhooks which is a feature provided by GitHub.
- Thus, webhooks is a feature, where after each new commit/change in the repository, it calls the Jenkins for a build.
- Jenkins runs on localhost. To make it available, we need a tunneling application to make local ports publically available and we use **ngrok** for that purpose. Install it using
`$ sudo snap install ngrok`
- Run ngrok on the same port on which your Jenkins is running i.e.
`$ ngrok http 8080`

```
ngrok by @inconshreveable (Ctrl+C to quit)

Session Status      online
Account             Mohit Bansal (Plan: Free)
Version             2.3.35
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://ffe5aa71.ngrok.io -> http://localhost:8080
Forwarding           https://ffe5aa71.ngrok.io -> http://localhost:8080

Connections         ttl      opn      rt1      rt5      p50      p90
0                0        0        0.00     0.00     0.00     0.00
```

Fig 31: Ngrok start

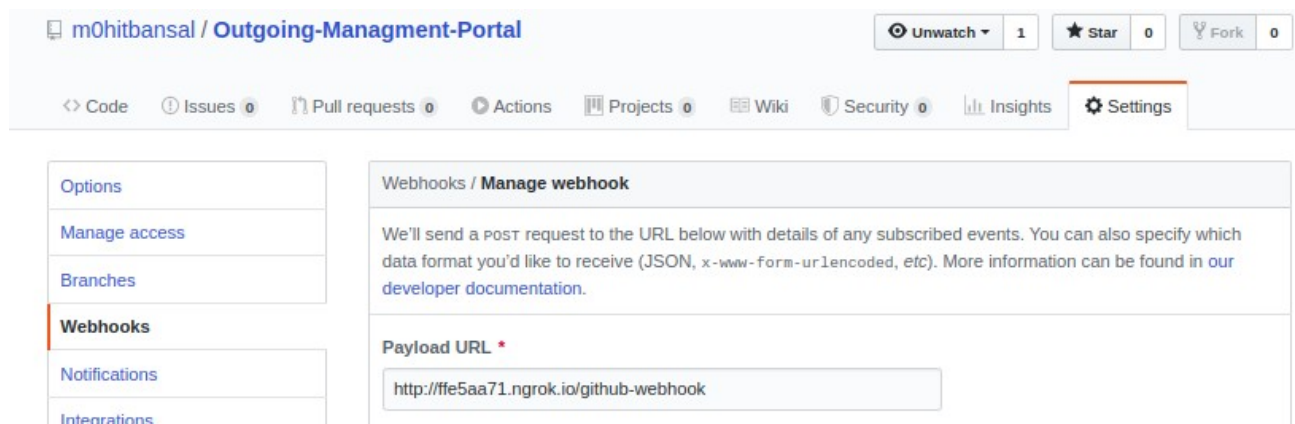


Fig 32: Github webhook setup

6.Experimental Setup

Installation

Install Nodejs and its libraries.

```
$ npm install nodejs
```

* Following Packages are also required:

express, express-session, mysql, body-parser, formidable, cors, nodemon, multer, chai, mocha, chai-http, node-cron, nodemailer, winston, winston-elasticsearch, nodemon

Git:

Install git and configure it using username and password.

```
$ sudo apt-get install git
```

```
$ git config --global user.name "m0hitbansal"
```

```
$ git config --global user.email "mohit.bansal@iiitb.org"
```

Install Docker:

Find resources at <https://docs.docker.com/engine/install/ubuntu/>

Docker Hub: Create a new account and create a new repository of the required name.

It can be pushed/pulled using

```
$ docker pull/push <DockerHub Username>/<DockerHub Repository Name>
```

Install Jenkins:

Jenkins can be installed from the official documentation. In manage-jenkins, find the plugins below, download and install them. Plugins – maven, pipeline, rundeck, Nodejs, GitHub Pull Request Builder, Build Pipeline, Dashboard & Email extension.

```
$ java -jar jenkins.war
```

Install Rundeck :

Install Rundeck using the official documentation. The default port for web-interface is 4440 and default username and password is admin.

```
$ dpkg -i rundeck_2.10.8-1-GA_all.deb
```

Install ELK Stack:

ELK Stack can be downloaded from <https://www.elastic.co/downloads/>

7.Result and Discussion

The screenshot shows the login interface of the 'OUTGOING MANAGEMENT PORTAL'. At the top left is the 'mtb' logo. The title 'OUTGOING MANAGEMENT PORTAL' is centered at the top. Below it, the word 'LOGIN' is displayed in a large, bold, purple font. The login form consists of two light gray input fields: 'Email' and 'Password'. Below these fields are two links: 'Remember me' (with an unchecked checkbox) and 'Forgot?'. A prominent purple 'LOGIN' button is centered below the links. At the bottom of the page, a dark blue footer contains the text 'Copyright @ 2020 IIT-Bangalore'.

Fig 33: Login Page

The screenshot displays the 'Request Leave' form. On the left is a sidebar with three menu items: 'Dashboard', 'Request for Leave', and 'Share Location'. The main content area is titled 'Request Leave' and includes the instruction 'Kindly fill the leave request form'. The form contains several input fields arranged in a grid: 'Roll Number:' (filled with 'MT2019026'), 'Name:' (filled with 'archit semwal'), 'Hostel:' (filled with 'Bhaskara'), 'Room Number:' (filled with '470'), 'Ticket Number:' (empty), 'Mode of Travel:' (empty), 'Destination:' (empty), 'Guardian's contact:' (empty), 'Departure date from college:' (with a date picker set to 'dd/mm/yyyy'), and a large text area for 'Reason for leave :'. A green 'Send' button is located at the bottom right of the form.

Fig 34: Student Leave Form Page

Leave Request Column Cards

Click on Accept/Reject...

mohit bansal

Roll Number :

MT2019048

Hostel :

Bhaskara

Room Number :

628

Ticket Number :

111

Mode of Travel :

bus

Destination :

pali

Guardian conatct :

9414989746

Departure Date :

23/05/2020

Reason for Leave :

hello

Allow

Reject

Fig 35: Warden's Approval Page

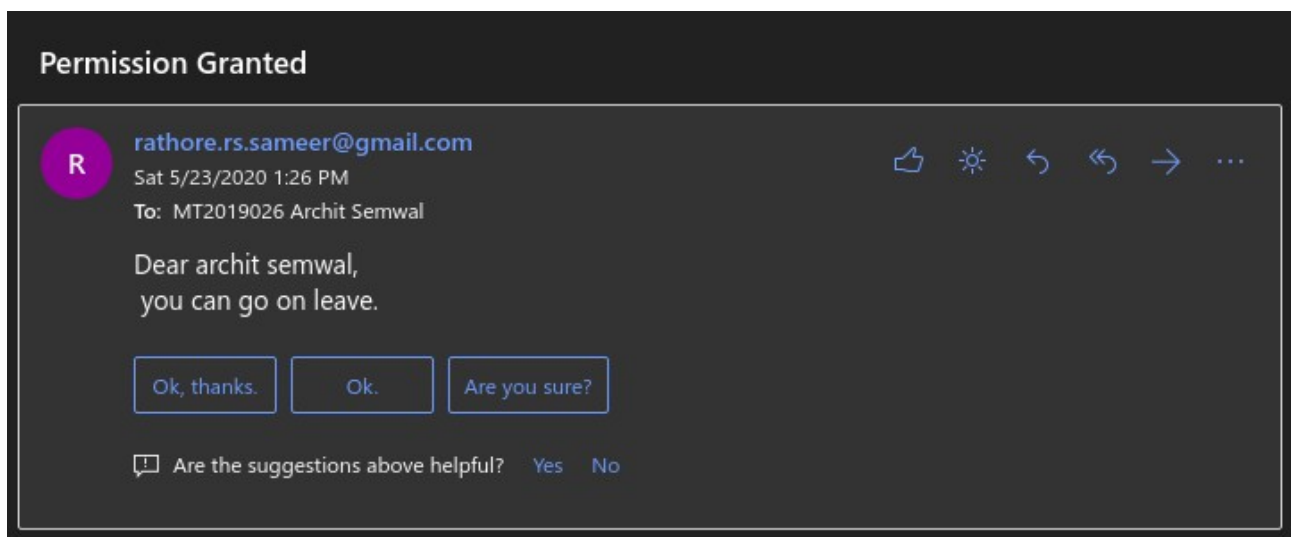



Fig 36: Autogenerated email for permission granted



OUTGOING MANAGEMENT PORTAL

MENU

Toggle Menu

GUARD ▾

Dashboard

Incoming-Outgoing

Outpass for Leave

Visitors and Parking

MT2019026

Go

Roll : MT2019026

Name : archit semwal

Hostel : Bhaskara

Room No : 470


In

Out

Fig 37: Local Outpass

Guard Login

localhost:5555/checkrole#



MENU

Toggle Menu

OK

Dashboard

Incoming-Outgoing

Outpass for Leave

Visitors and Parking

MT2019100

Go

localhost:5555 says
Request Leave not found

Fig 38: Leave not found alert

The screenshot shows the 'OUTGOING MANAGEMENT PORTAL' interface. On the left is a 'MENU' with options: Dashboard, Incoming-Outgoing, Outpass for Leave, and Visitors and Parking. A 'Toggle Menu' button is next to it. A 'GUARD' dropdown is in the top right. A 'Leave Outpass' modal is open in the center, containing the following fields:

MT2019026	archit semwal
Bhaskara	470
INDIGO-QR45Zn2X	flight
Dehradun	Approved By Warden
23/05/2020	

At the bottom of the modal are two buttons: 'Departure' (blue) and 'Return' (yellow).

Fig 39: Outstation outpass Page

The notification is titled 'Your Ward is leaving for home'. It is from 'rathore.rs.sameer@gmail.com' and dated 'Sat 5/23/2020 1:27 PM'. The subject is 'To: MT2019026 Archit Semwal'. The message reads: 'Dear Parent, Your ward archit semwal, has left for home. Please be informed.' Below the message are three buttons: 'Thank you for letting us know.', 'Thanks for the heads up.', and 'Thank you for informing me.' At the bottom, there is a feedback question: 'Are the suggestions above helpful?' with 'Yes' and 'No' options.

Fig 40: Guardian's Notification

The notification is titled 'Your Ward has arrived college'. It is from 'rathore.rs.sameer@gmail.com' and dated 'Sat 5/23/2020 1:27 PM'. The subject is 'To: MT2019026 Archit Semwal'. The message reads: 'Dear Parent, Your ward archit semwal, has arrived in college. Please be informed.' Below the message are three buttons: 'Thank you for letting us know.', 'Thank you!', and 'Thank you for the update.' At the bottom, there is a feedback question: 'Are the suggestions above helpful?' with 'Yes' and 'No' options.

Fig 41: Guardian's Notification

Dashboard

Incoming-Outgoing

Outpass for Leave

Visitors and Parking

Visitor Exit

Enter ID like visitor_xxExit

Visitor Details

Kindly fill the visitor form

Name :

Mohito Bansal

Email Id. :

Mohit.Bansal@iiitb.org

Meeting with Prof. G. N. Prasanna

G

Vehicle Entry : ☐

Enter

Fig 42: Visitor entry without vehicle

Outpass for Leave

Visitors and Parking

Visitor Details

Kindly fill the visitor form

Name :

Mohito Bansal

Email Id. :

Mohit.Bansal@iiitb.org

Meeting with Prof. G. N. Prasanna

G

Vehicle Entry : ☒

Vehicle Number :

DEL08A5534

Parking Slot :

1

Enter

Fig 43: Visitor entry with vehicle

From: rathore.rs.sameer@gmail.com <rathore.rs.sameer@gmail.com>
Sent: Saturday, May 23, 2020 1:36:20 PM
To: MT2019065 Mohit Bansal <Mohit.Bansal@iiitb.org>
Subject: Your visiting entry details

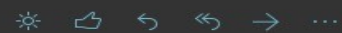
hello Mohito Bansal,
Your visiting id :visitor_1
Your parking no :1

Fig 44: Allot unique id and parking slot to Visitor

Hostel Reminder

R

rathore.rs.sameer@gmail.com
Sat 23-05-2020 11:57
To: MT2019065 Mohit Bansal



These students are late for today after 10pm

MT2019100
MT2019048
MT2019026

Thank you for letting us know.

Ok, thank you.

Ok, thanks.

Are the suggestions above helpful? Yes No

Fig 45: Autogenerated email to warden about late student's

Hostel Reminder

R

rathore.rs.sameer@gmail.com
Sat 23-05-2020 11:57
To: MT2019065 Mohit Bansal

Dear mohit bansal,
you are late, kindly reach hostel ASAP.

Fig 46: Student's late reminder email

8.Future Work

1. Warden can find list of students on leave for a specific date. warden when login can see students on leave on that day he logged in.
2. Student can see past records of his leave.
3. Student can share his locational co-ordinates when he is late on the web-portal.
4. We can append student id card scanner with the portal.

9.Conclusion

Deployment time: Avg 7.2 min.

We were successfully able to achieve the required results, and were able to integrate all the tools. However one issue is the time taken to deploy the app. The npm command to fetch the dependencies takes a lot of time due to the high number of dependencies being used.

Overall, it was great experience to learn and implement the DevOps tools as it will give us a headstart in the corporate world where these tools are used on a daily basis. DevOps tools are making the task quite easier manual interaction is getting reduced. Once all the things are set up, these tools will run automatically and building, integration, deployment, monitoring, testing will take place in a continuous manner.

10.References

[1] <https://github.com/Alakazam03>

[2] <https://dev.to/deleteman123/logging-at-scale-done-right-3m0e>

[3] <https://semaphoreci.com/community/tutorials/getting-started-with-node-js-and-mocha>

[4]<https://medium.com/@sece.cosmin/docker-logs-with-elastic-stack-elk-filebeat-50e2b20a27c6>