International Institute of Information Technology, Bangalore

IPL Auction

Prof. B. Thangaraju,

Teaching Assistant: Vaibhav

A.Harsha Vardhan
IMT2016101

V.Bhanu Prakash
IMT2016016

Index

1. Abstract

Online Cricket Management is a project aimed to provide an application that enables buyers to auction players through a portal. This application makes things easy for an auctioneer to conduct an auction without any flaws and conduct matches in round-robin fashion. We used DevOps tools and approaches to develop and deploy the above-mentioned application.

2. Introduction

2.1 Work Plan

The plan is to create a simple applications that is easy to use where players fill their data using google form and all the players data are maintained in sql database.on the day of auction all the buyers can participate in it, irrespective of their location.the application contains two interfaces one for buyers and the other is for auctioneer. The buyers interface will allow them to bid on the player and all statistics of their money and position of bidding. The auctioneer interface allows him to start the auction slot and select the highest bidder for the player.we use embedded js for multiple simultaneous actions on budding.

2.2 Why DevOps

As the project needs to be built in an incremental manner, automated pipelines make things easier . Devops helps us to decrease the deployment failures using agile programming principles since both of us are working from different locations, an automated pipeline will not only make our work easier, it also makes it more efficient. Increased efficiency helps to speed the development process and make it less prone to error. The amount of communication that has to happen between us will reduce. Build acceleration tools helps to compile code much faster

3. System configuration

NAME	SPECIFICATIONS
OS-UBUNTU	18.04 LTS
CPU	Intel core I7(8th gen)
RAM	8GB RAM
Languages used	HTMl,CSS,Javascript,Embedded JS
Kernel Version	5.3.0-53-generic

4. Software Development Life Cycle(SDLC)

The basic process of SDLC involves source code management, building, testing, deployment, and monitoring. Doing these processes individually takes time and resources. Using DevOps tools we will be able to automate them. The tools we will be using for each stage are:-

1. Source Code Management(SCM): **Git**
2. Building : **Docker & npm**
3. Testing : **Selenium**
4. Deployment : **Rundeck**
5. Monitoring : **ELK stack**

4.1 Source Code Management(SCM):

SCM is a software tool used by programmers to manage source code.We can keep track of all the changes made on a project regularly.It maintains the timestamp and maintains the names of the person who is responsible for those changes. The tool we used for SCM is git.

Why are we using git?

- Git is a distributed version control system
- Easy to push and merge
- Makes easy to track changes in the project
- In case of error/lesser efficiency, reversion of code files to previous state is also possible.

Commands used for code management are:

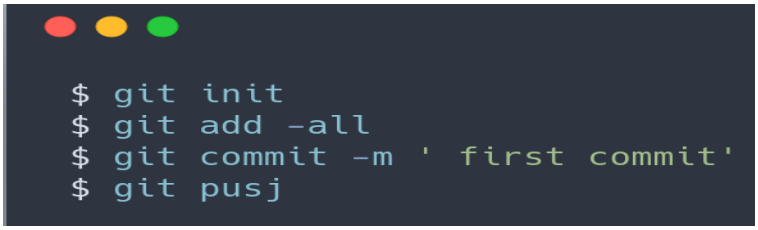


Figure:1

4.2 Building

We used npm(Node Package Manager) as a build automation tool for our project. It has an online database for public packages. The available packages can be browsed and searched via the npm website. For deployment of website there are some dependencies that need to be installed.installing these dependencies is the build stage of our project

Npm install

4.3 Testing

Testing is done using an automation tool called selenium. It is a open-source tool used for automating the test carried on web browsers.
Before testing with selenium, we installed a chrome extension of selenium for chrome browser. Once it's installed we need to launch the runner. From the chrome extension, all the tests can be downloaded to the local repository. We have downloaded the test suites and placed it in the './testing/' directory.

4.4 Artifact

We are using docker as an artifact for deploying our project. For this, we are creating three docker containers

- ➔ DOCKER-1 : docker image containing the code of our web application
- ➔ DOCKER-2 : docker image containing the database for our web application
- ➔ DOCKER-3 : docker image containing the redis for our web application

4.5 Deploy

For continuous deployment of our application, we are using RUNDECK.
Install RUNDECK and login with admin. Following below steps helps us to configure nodes for deploying projects successfully.

4.6 Monitor

ELK Stack: ElasticSearch Logstash Kibana Stack is a collection of open source s/w provided by Elastic. Using this we can search, analyze and visualize logs generated from any application/system sources.

The ELK has 4 main components:

- ElasticSearch: A distributed search engine which stores all the collected data.
- Logstash: This component sends the incoming data to ElasticSearch. This is mostly a backend like component.
- Kibana: A frontend like platform, (a web application) used for searching and visualizing logs.
- Beats: They are used as data shippers that can send loads of data from machines to Elasticsearch or Logstash.
- We use Filebeat and Metricbeat to analyze and visualize the logs.

5. Installation Procedure

We can do Installations,one by downloading the executables and the other is using commands. We mostly follow installations using commands because it is easy to use.

5.1 NodeJs

To install nodejs use following commands:

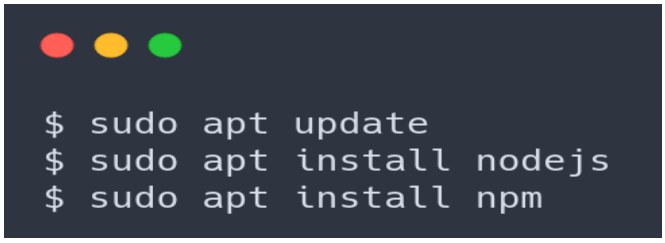


Figure 2

To check their version use the following commands. If your installation is successful you can see their version as below.

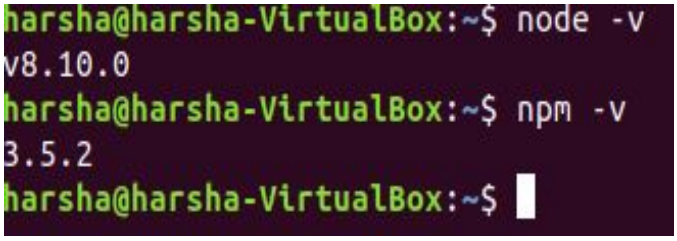


Figure:3

5.2 MySQL

To install, use the following commands:

```
$ sudo apt-get update

$ sudo apt-get install mysql-server

Here you asked to create a root password for mysql.Create your own mysql and password

$ systemctl status mysql.service
```

Figure:4

After installation and running the status command the output will look something like

```
harsha@harsha-VirtualBox:~$ systemctl status mysql.service
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: en
   Active: active (running) since Thu 2020-06-04 10:21:01 IST; 11h ago
 Main PID: 1620 (mysqld)
    Tasks: 28 (limit: 4915)
   CGroup: /system.slice/mysql.service
           └─1620 /usr/sbin/mysqld --daemonize --pid-file=/run/mysqld/mysqld.pid
```

Figure:5

5.3 Redis installation

We will use apt to install it from official ubuntu repositories

```
$ sudo apt update
$ sudo apt install redis-server
```

Figure: 6

We need to make a change in redis configuration file.inside the file, find the supervised directive. This directive allows you to declare an init system to manage Redis as a service.we need to change the directive from no to systemd

```
$ sudo nano /etc/redis/redis.conf
```

Figure:7

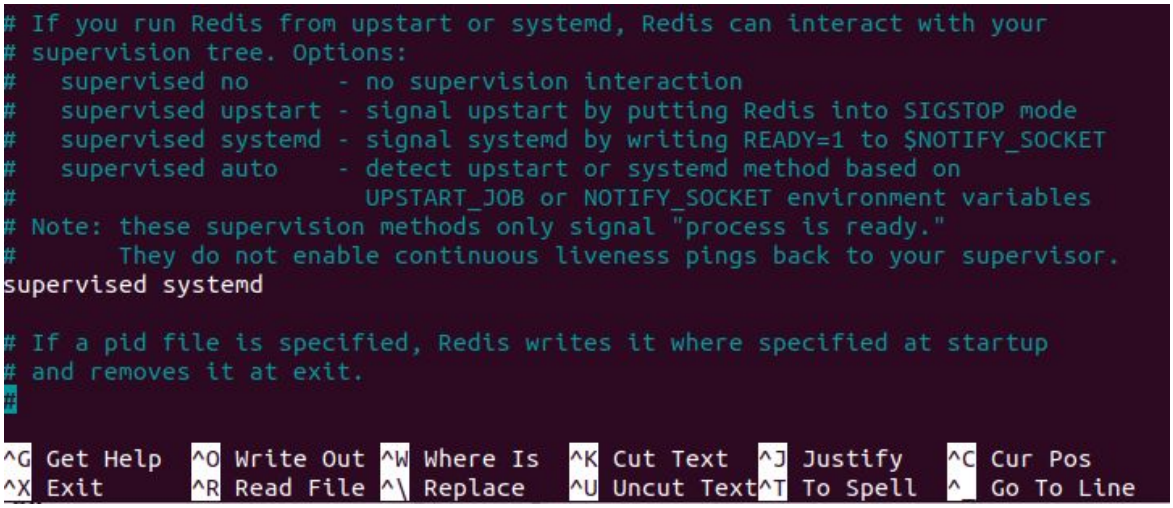


Figure:8

To reflect your changes restart redis using following command

5.3 SCM -git

1. First we need to run a general update. we can update by the following command in Figure 9 from installation procedure file.

Figure: 9

2. After you have run the general update on the server, start installing git use following command

Figure:10

3. You may be asked to confirm the download and installation; simply enter y/Y to confirm. It's that simple, Git should be installed and ready to use!
4. Confirm git installation with following command

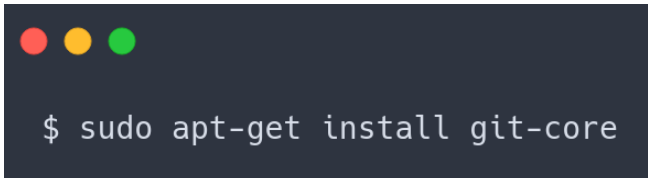


Figure:11

5.4 Build - Selenium

Test automation is performed using Selenium.

Before, testing with selenium, we installed a chrome extension of selenium for chrome browser.

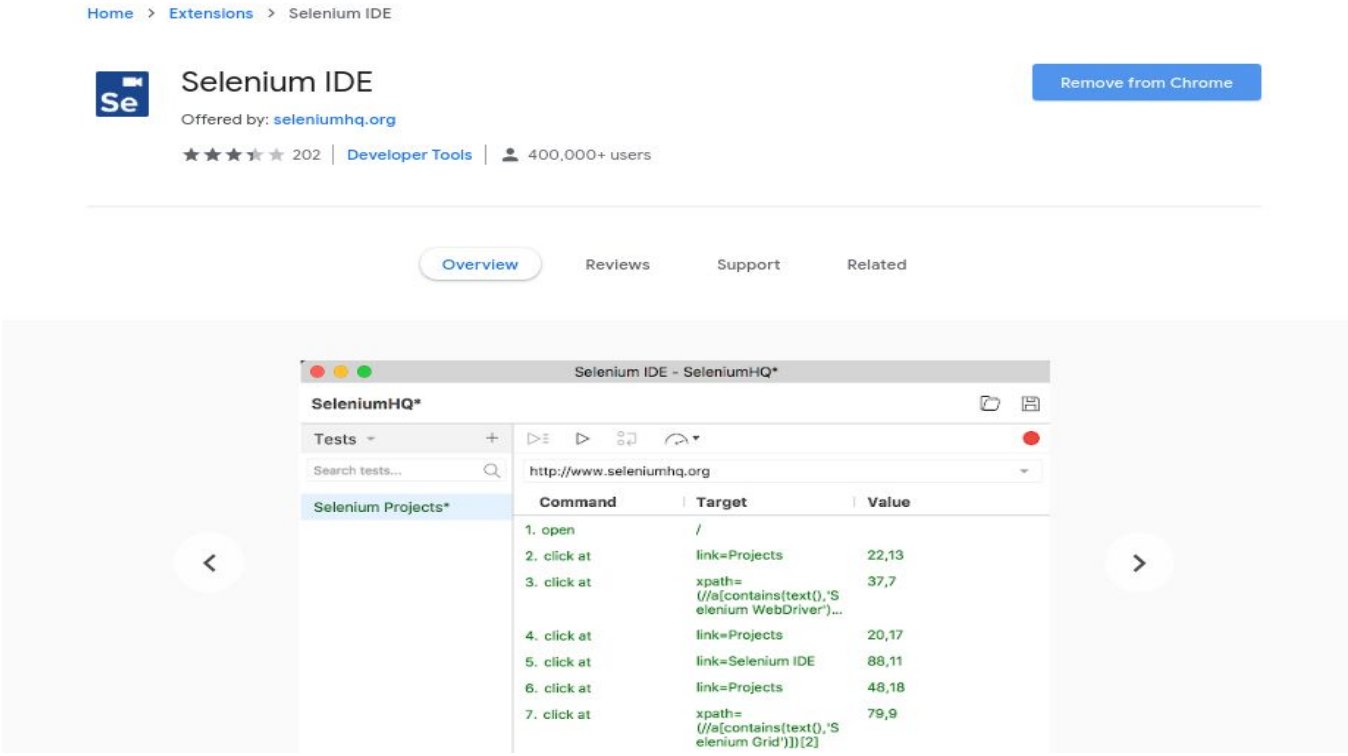


Figure:12

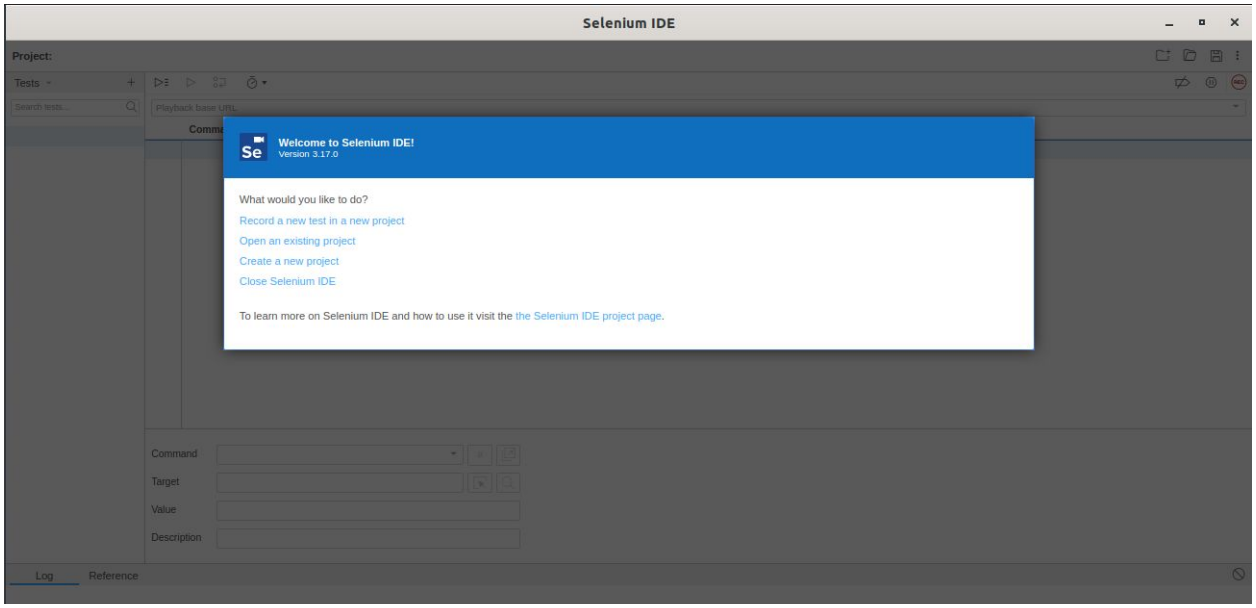


Figure:13

The tests we performed are visible in the left pane of the extension window can be seen above Using this extension we recorded different tests. For this application we performed a few tests. For each test, we can run it by clicking the play button. On clicking this the extension automatically pops up a window that shows what are all the events/clicks that were performed as shown below

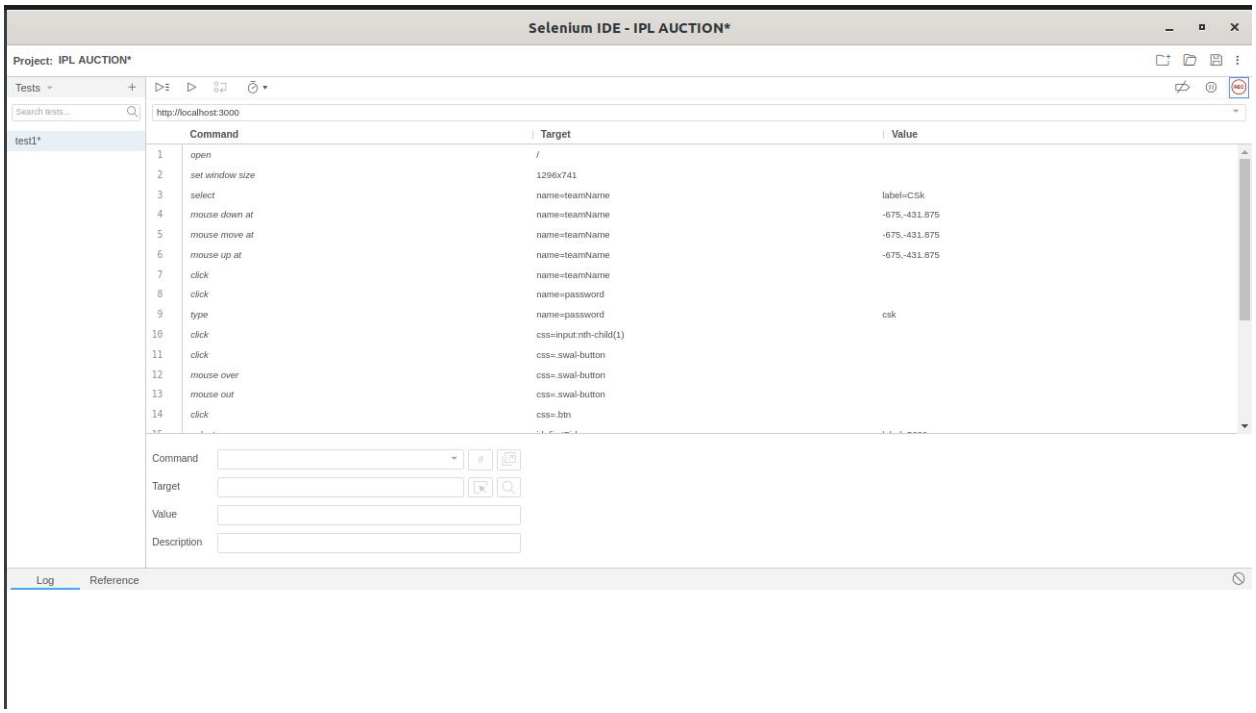


Figure:14

5.5 Docker

Docker is an application that helps packing and running an application process easily. It provides OS level virtualization to deliver the software in packages known as containers .
Create a Docker Hub account to create and push images to docker hub. Use this link to create account
The Docker installation packages available on ubuntu 18.04 may not be the latest version. To get latest version we will install Docker from Docker's official repository.
First, in order to ensure the downloads are valid, add the GPG key for the official Docker repository to your system.

```
$ sudo apt update
```

We will start with updating existing list of packages:

Figure:15

Now, lets install a few prerequisite packages which let apt use packages over HTTPS,

```
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

Figure:16

Add the GPG key for the official Docker repository to your system

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Figure:17

Update the package database from newly added repository

```
$ sudo apt update
```

Figure:18

To make sure we are installing from official docker repository run the following command :

```
$ apt-cache policy docker-ce
```

Figure:19

You will get output like this, version number for Docker can be different

```
harsha@harsha-VirtualBox:~$ apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:19.03.11~3-0~ubuntu-bionic
  Version table:
```

Figure:20

Now, install Docker

```
$ sudo apt-get install -y docker-ce
```

Figure:21

Docker is installed. To check that it is running use the following command

```
$ sudo systemctl status docker
```

Figure:22

Output should look like:

```
harsha@harsha-VirtualBox:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset:
   Active: active (running) since Wed 2020-06-03 17:18:08 IST; 43s ago
     Docs: https://docs.docker.com
 Main PID: 11325 (dockerd)
    Tasks: 10
   CGroup: /system.slice/docker.service
           └─11325 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/conta
```

Fig 23

5.6 Deploy- rundeck

Rundeck is an open source software. It is used to automate routine procedures in the data centres or cloud environments

Rundeck does not support 1.11 java version make sure u are on 1.8 version to verify java version use following command

```
$ java -version
```

Fig 24

Output looks like this

```
harsha@harsha-VirtualBox:~$ java -version
openjdk version "1.8.0_252"
OpenJDK Runtime Environment (build 1.8.0_252-8u252-b09-1~18.04-b09)
OpenJDK 64-Bit Server VM (build 25.252-b09, mixed mode)
```

Fig 25

To install rundeck, run the following commands

Figure 26

To start rundeck services use the command

Figure 27

To check the status of the rundeck use the following command

Figure 28

5.6.1 For logging to the rundeck

- Browse http://localhost:4440/
- Login in with user name as admin and password as admin

Rundeck is now running!

5.7 Monitor -ELK

5.7.1 Installation

Default ubuntu packages does not contain any Elasticsearch components .We can install with APT after adding Elastic's package source list.

Inorder to protect your system from package spoofing, all packages are signed with Elasticsearch key. First we will import the Elasticsearch public GPG key and then add the Elastic package source list to install Elasticsearch.

-fsSL is used to silence all progress and possible errors to allow cURL to make a request on a new location if it is redirected.

Run the following Commands

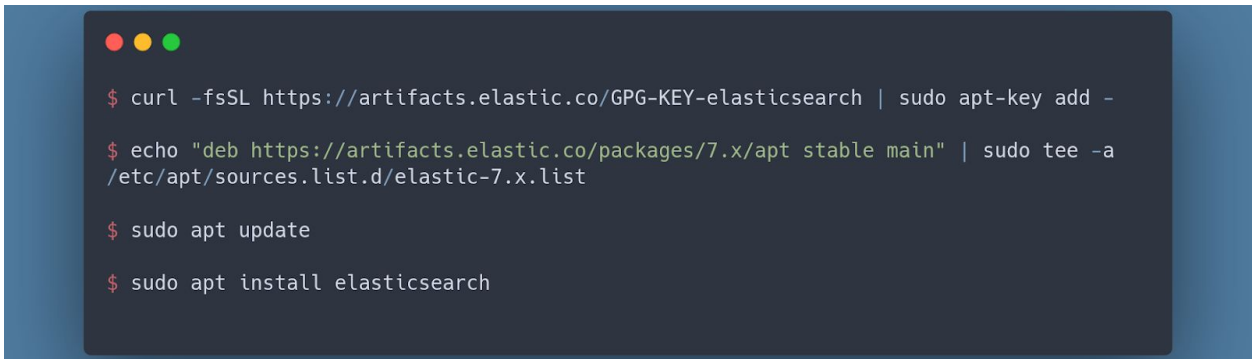
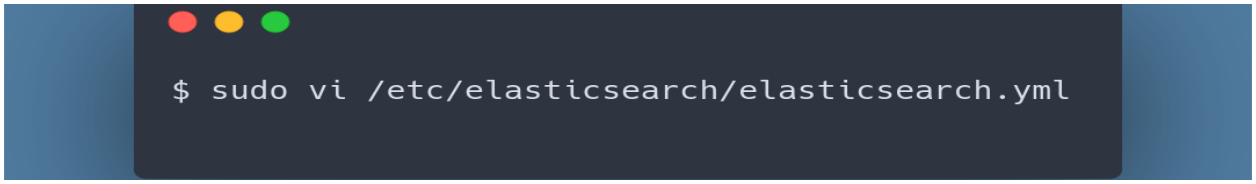


Figure 29

5.8 Configuration of Elasticsearch

To configure Elasticsearch we open ealsticsearch.yml file to edit in command line use following command

Fig 31



Elasticsearch running in localhost:9200

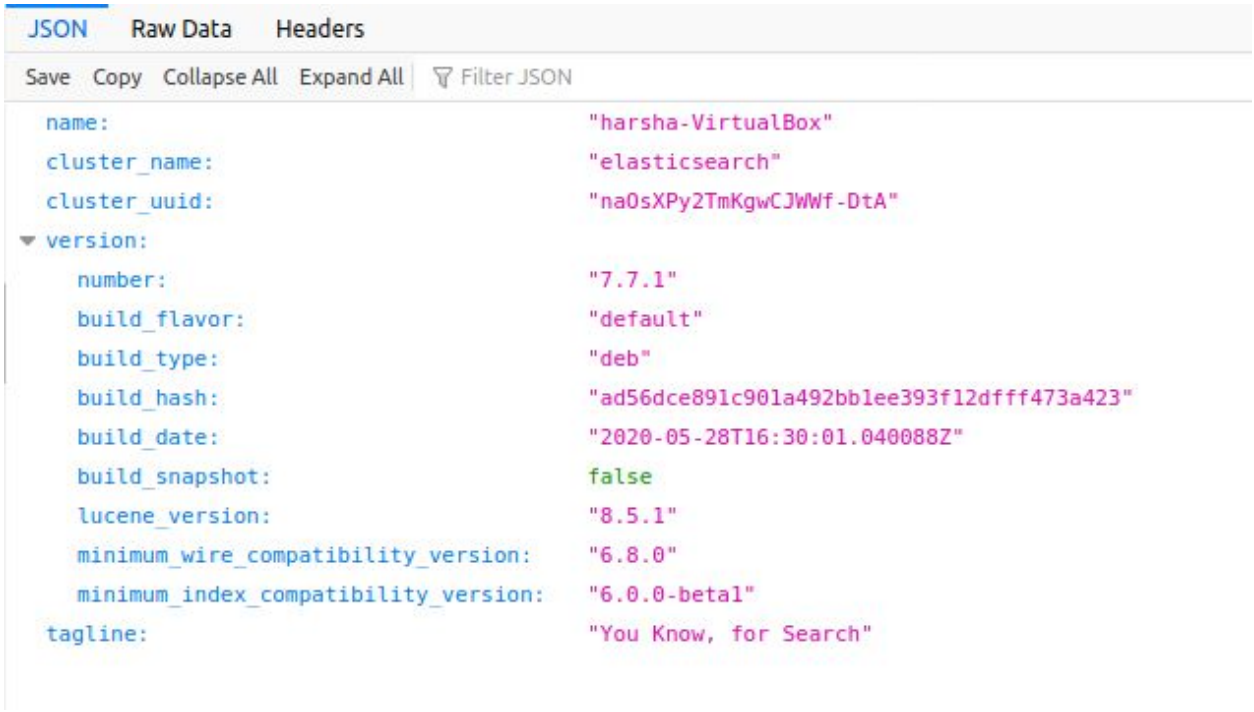


Fig 31

5.9 Installing Kibana

To Install Kibana use following command

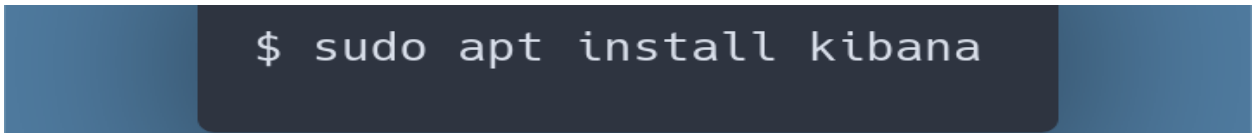


Fig 32

To start and enable the Kibana use below commands:

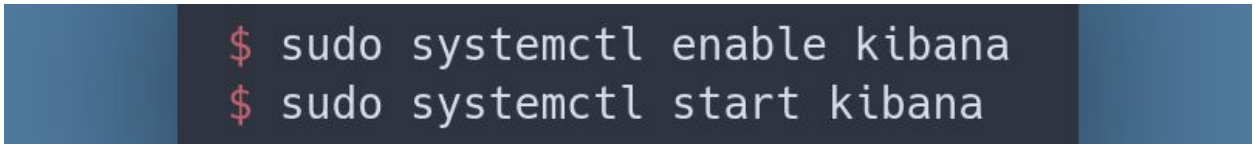


Fig 33

5.10 Configuration of Kibana

TO create a administrative Kibana username and password: use the following command

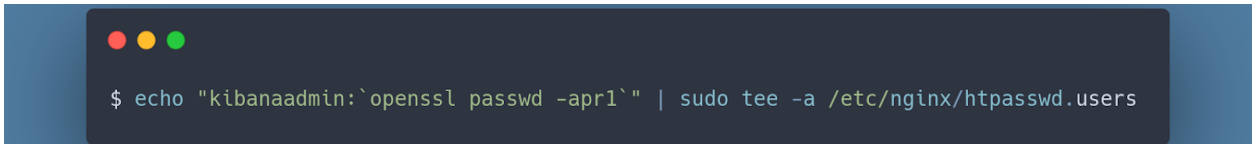


Fig 34

To configure the server we use nginx. To create nginx server block use the below command

Fig 35

We update the iplauction.com code as follows:

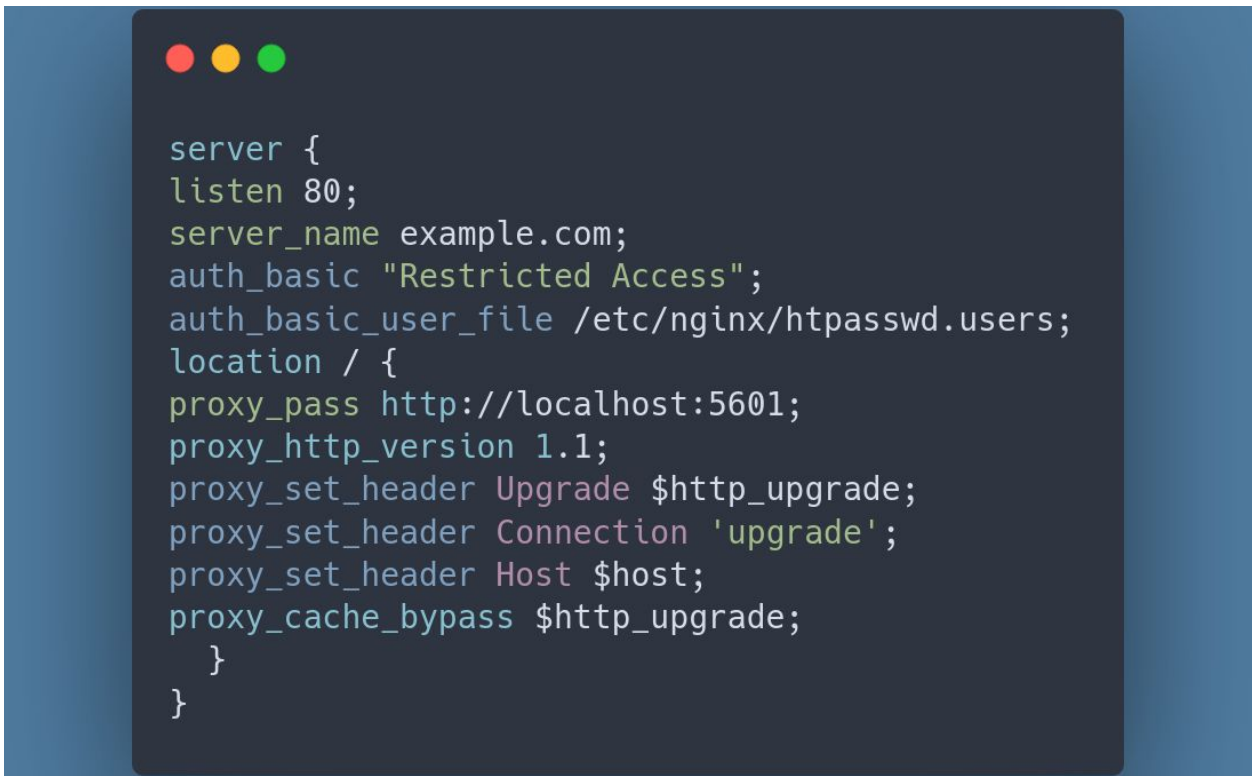


Fig 36

This code configures Nginx to direct the traffic to port localhost:5601

To create a symbolic link to the sites-enabled directory using the below command:

Fig 37

Run the below command to check for any errors

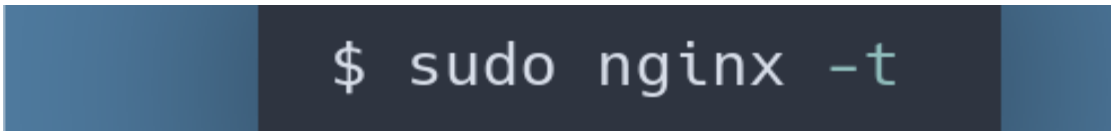


Fig 38

To restart nginx :

Fig 39

To see kibana dashboard go to http://localhost:5601/status

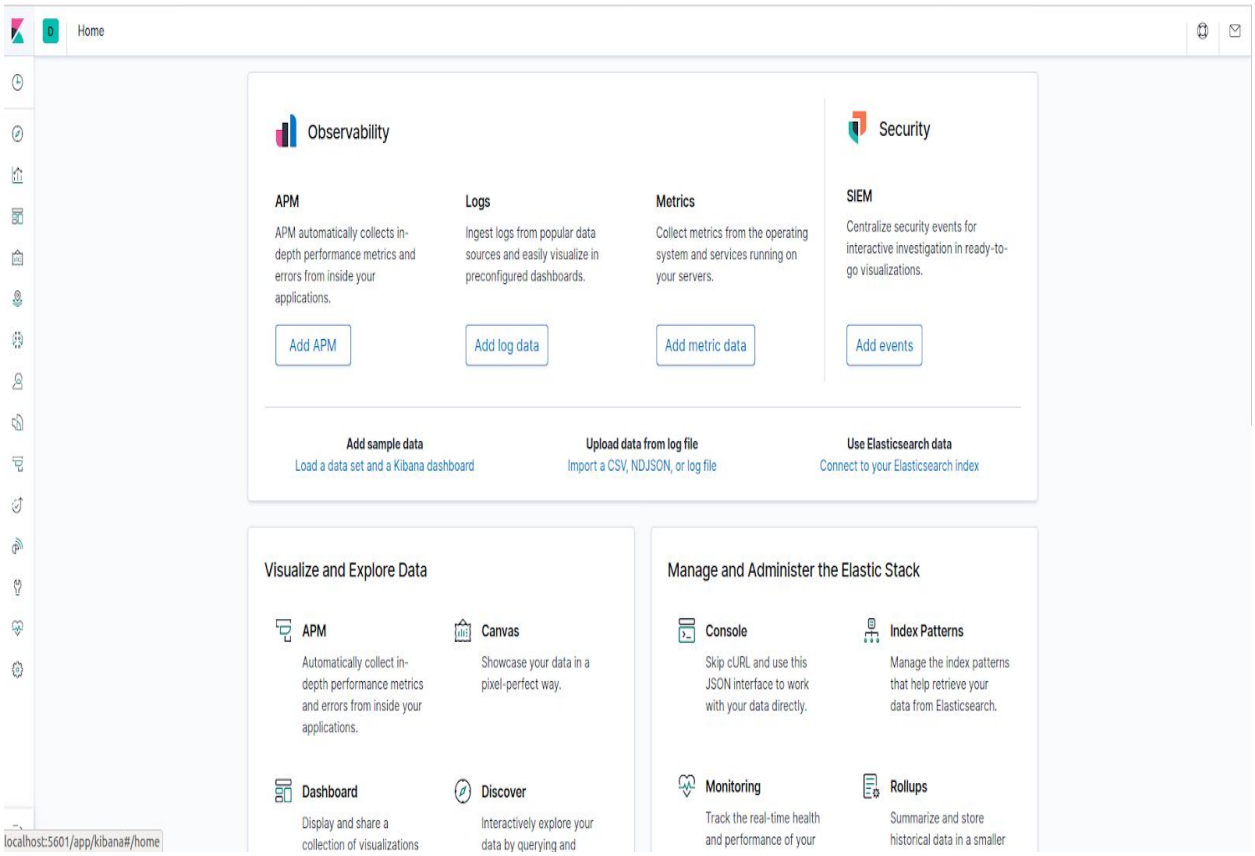


Fig 40

5.11 Installation of Logstash

We can install logstash using this command:

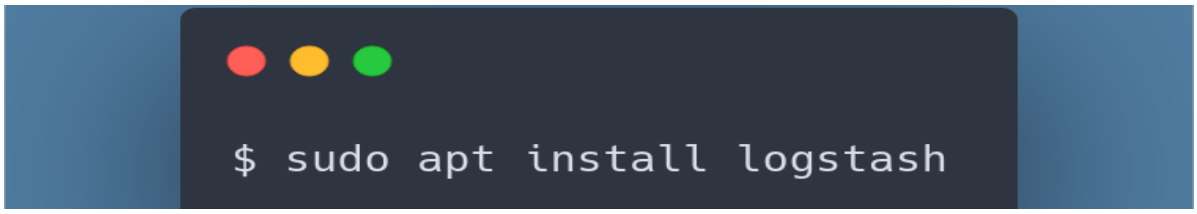


Figure 41

Logstash at one end takes the data and processes it and sends it to destination ElasticSearch.A logstash pipeline requires three elements input,output and filter.

The input plugin takes the data as input, and the filter plugin processes the data, and finally the

output plugin outputs it to ElasticSearch. Let's now look how to configure these plugins:

To Configure input plugin we create a new file and add the code below:

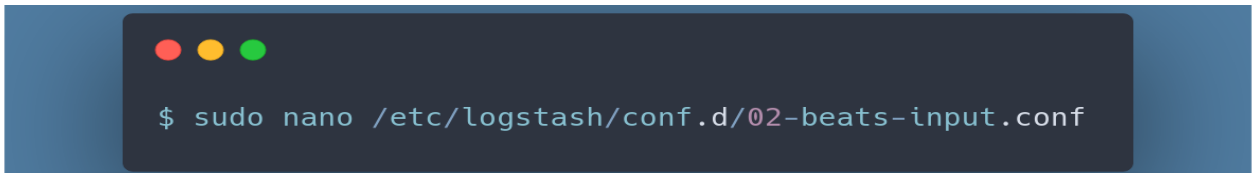


Figure 42

Figure 43

This specifies that beats to listen on port 5044

To configure filter plugin we create a new file and add the code below;

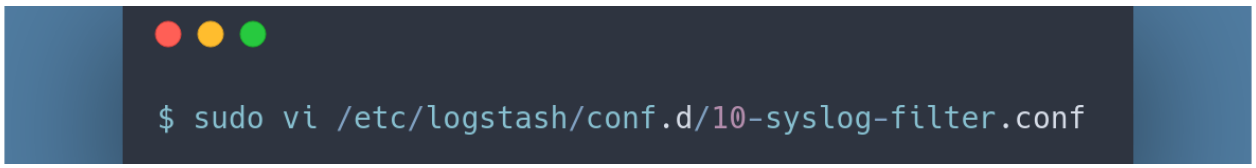


Fig 44

```
filter {
 if [fileset][module] == "system" {
  if [fileset][name] == "auth" {
   grok {
    match => { "message" => ["%{SYSLOGTIMESTAMP:[system][auth][timestamp]}
%{SYSLOGHOST:[system][auth][hostname]} sshd(?:\[%{POSINT:[system][auth][pid]}\])?:
%{DATA:[system][auth][ssh][event]} %{DATA:[system][auth][ssh][method]} for (invalid
user
)?%{DATA:[system][auth][user]} from %{IPORHOST:[system][auth][ssh][ip]} port
%{NUMBER:[system][auth][ssh][port]} ssh2(:
%{GREEDYDATA:[system][auth][ssh][signature]})?",
"%{SYSLOGTIMESTAMP:[system][auth][timestamp]}
%{SYSLOGHOST:[system][auth][hostname]} sshd(?:\[%{POSINT:[system][auth][pid]}\])?:
%{DATA:[system][auth][ssh][event]} user %{DATA:[system][auth][user]} from
%{IPORHOST:[system][auth][ssh][ip]}",
"%{SYSLOGTIMESTAMP:[system][auth][timestamp]}
%{SYSLOGHOST:[system][auth][hostname]} sshd(?:\[%{POSINT:[system][auth][pid]}\])?:
Did not receive identification string from %{IPORHOST:[system][auth]
[ssh][dropped_ip]}",
"%{SYSLOGTIMESTAMP:[system][auth][timestamp]}
%{SYSLOGHOST:[system][auth][hostname]} sudo(?:\[%{POSINT:[system][auth][pid]}\])?:
\s*%{DATA:[system][auth][user]} :( %{DATA:[system][auth][sudo][error]} ;)?
TTY=%{DATA:[system][auth][sudo][tty]} ; PWD=%{DATA:[system][auth][sudo][pwd]} ;
USER=%{DATA:[system][auth][sudo][user]} ;
COMMAND=%{GREEDYDATA:[system][auth][sudo][command]}",
"%{SYSLOGTIMESTAMP:[system][auth][timestamp]}
%{SYSLOGHOST:[system][auth][hostname]}
groupadd(?:\[%{POSINT:[system][auth][pid]}\])?: new group:
name=%{DATA:system.auth.groupadd.name}, GID=%{NUMBER:system.auth.groupadd.gid}",
"%{SYSLOGTIMESTAMP:[system][auth][timestamp]}
%{SYSLOGHOST:[system][auth][hostname]} useradd(?:\[%{POSINT:[system][auth][pid]}\])?:
new user: name=%{DATA:[system][auth][user][add][name]},
UID=%{NUMBER:[system][auth][user][add][uid]},
GID=%{NUMBER:[system][auth][user][add][gid]},
home=%{DATA:[system][auth][user][add][home]},
shell=%{DATA:[system][auth][user][add][shell]}$",
"%{SYSLOGTIMESTAMP:[system][auth][timestamp]}
%{SYSLOGHOST:[system][auth][hostname]}
%{DATA:[system][auth][program]}(?:\[%{POSINT:[system][auth][pid]}\])?:
%{GREEDYMULTILINE:[system][auth][message]}"] }
   pattern_definitions => {
    "GREEDYMULTILINE"=> "(.|\n)*"
    }
    remove_field => "message"
    }
    date {
    match => [ "[system][auth][timestamp]", "MMM d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
    geoip {
     source => "[system][auth][ssh][ip]"
     target => "[system][auth][ssh][geoip]"
    }
   }
   else if [fileset][name] == "syslog" {
    grok {
     match => { "message" => ["%{SYSLOGTIMESTAMP:[system][syslog][timestamp]}
%{SYSLOGHOST:[system][syslog][hostname]}
%{DATA:[system][syslog][program]}(?:\[%{POSINT:[system][syslog][pid]}\])?:
%{GREEDYMULTILINE:[system][syslog][message]}"] }
    pattern_definitions => { "GREEDYMULTILINE" => "(.|\n)*" }
    remove_field => "message"
    }
    date {
    match => [ "[system][syslog][timestamp]", "MMM d HH:mm:ss", "MMM dd HH:mm:ss" ]
   }
  }
```

Fig 45

In this code we are adding a filter for system logs. This makes the incoming syslogs structured and usable by the kibana dashboard.

Now let's configure output plugin: lets create a new file and add the code below:

Fig 46

Fig 47
This output configures logstat to store the Beats data in ElasticSearch running on 9200

5.12 Filebeat installation and configuration

Filebeat is used for collecting and shipping the log files.
Metric is used for collecting metrics from system and services

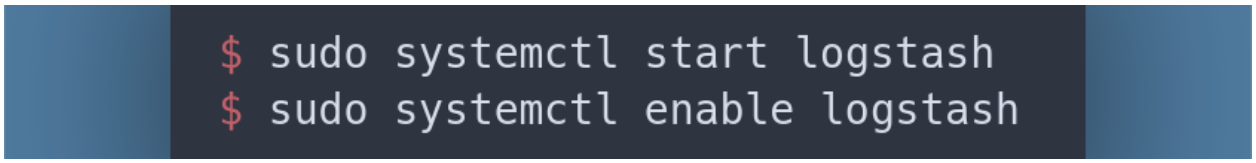


Fig 48
Lets install Filebeat by using below command:

Fig 49

To configure the .yml file inside /etc/filebeat:

```
$ sudo vi /etc/filebeat/filebeat.yml
```

Fig 50

Inorder to send the logs directly to logstassh, we uncomment the below code:
output .logstash:

Fig 51

```
# The Logstash hosts
hosts: ["localhost:5044"]
```

Since our application runs on mysql, and in order to collect syslogs and the logs from mysql we enable the following services.

```
$ sudo filebeat modules enable system
$ sudo filebeat modules enable mysql
```

Fig 52

To check what services are enabled for filebeat use below command:

```
$ sudo filebeat modules list
```

Fig 53

In order to direct the logs to a certain outputs destination, we load the below template:

```
$ sudo filebeat setup --template -E output.logstash.enabled=false -E
output.elasticsearch.hosts=["localhost:9200"] -E setup.kibana.host=localhost:5601
```

Fig 54
To enable and start filebeat:

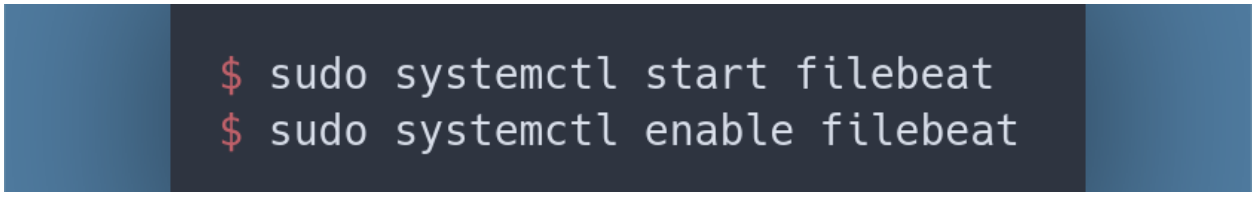


Fig 55

6. Continuous Integration and Deployment Pipeline

6.1 Jenkins

Installation and setting up Jenkins

Fig 56 installation of jenkins

Starting Jenkins

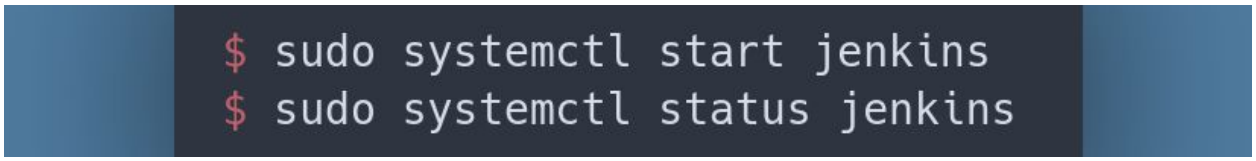


Fig 57 starting of jenkins

Status of jenkins looks like this

```
harsha@harsha-VirtualBox:~$ sudo systemctl status jenkins
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; generated)
   Active: active (exited) since Thu 2020-06-04 10:20:58 IST; 44min ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 0 (limit: 4915)
   CGroup: /system.slice/jenkins.service
```

Fig 58

6.2 Setup

Go to http://localhost:8080/

To unlock jenkins use the password from typing bellow command line

```
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Fig 59

Install required plugins.

Jenkins installation is successful. Lets configure our pipeline for continuous integration and deployment

We will go in stepwise order (SCM integration -> Build integration -> Docker integration -> Rundeck integration for continuous deployment -> testing -> ELK stack for monitoring)

Before starting the pipeline we need to make sure we have required configuration tools.
First go to **Manage Jenkins** → **Global tool Configuration** then select **Add git** and provide the location of where git has been installed in your system

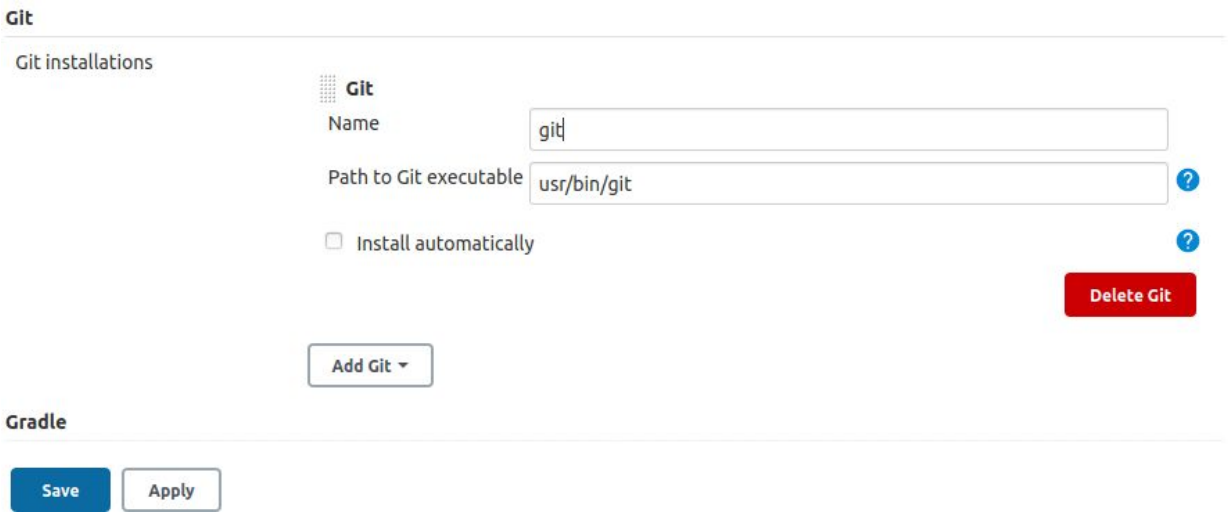


Fig 60

Now jenkins has the location of git.

Now, in the same Global Tool Configuration, we can also provide docker location so that jenkins can use it.

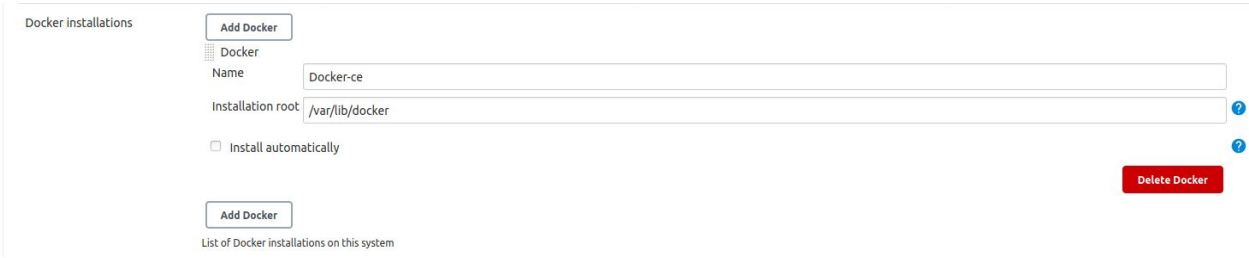


Fig 61

For creating the pipeline we will be using Blue Ocean. To install Blue ocean go to manage jenkins and go to manage plugins and search in available tab and install blue ocean after installation you can see blue ocean as shown in

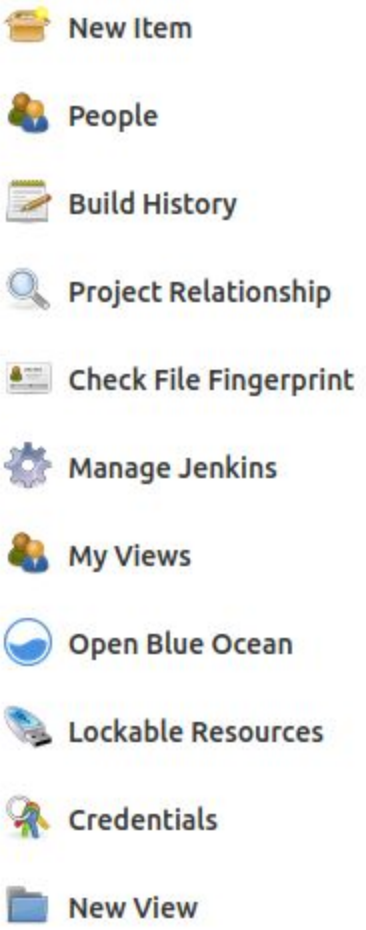


Fig 62

After installation of Blue Ocean, let us create a new project to automate the process of building,testing and Deployment.

In Jenkins dashboard select **New Item** as shown in

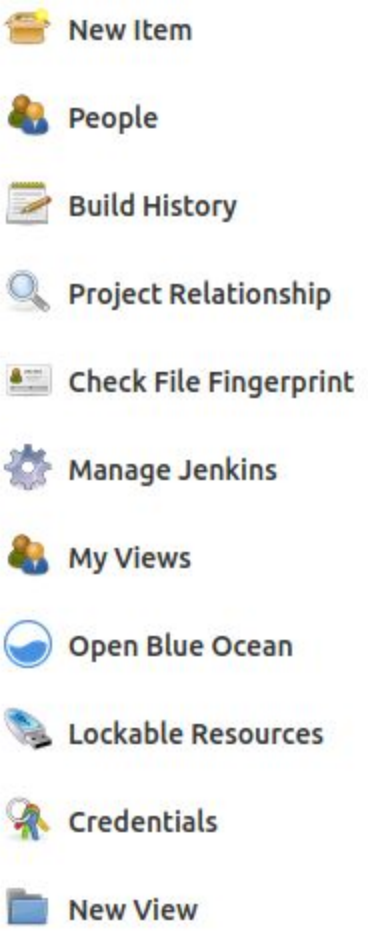


Fig 62

Fill the name of the item and select the type of project. As our main aim is to automate all stages of SDLC pipeline is the perfect choice then select ok

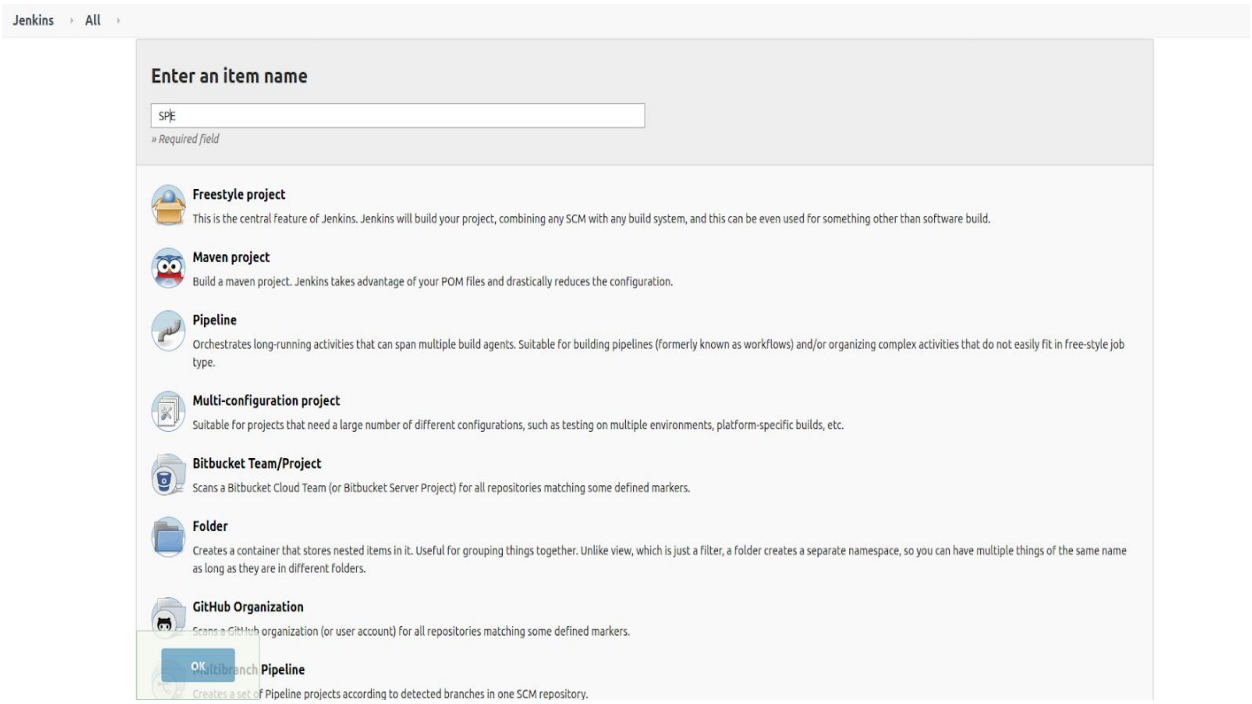


Fig 63

Step 1

We need to make Github inform Jenkins to pull the latest code and build it after every push into the github for that go to **Webhook** and tick **just the push event (for triggering** webhook**)**

Step 2

Now let's set the build step of our pipeline.we are using node js for our backend server. But for our server to properly function we have to have all the dependencies installed properly along with their versions as well. To do that node creates a file called packages.json that contains all the information of all the modules installed and their corresponding versions as shown below

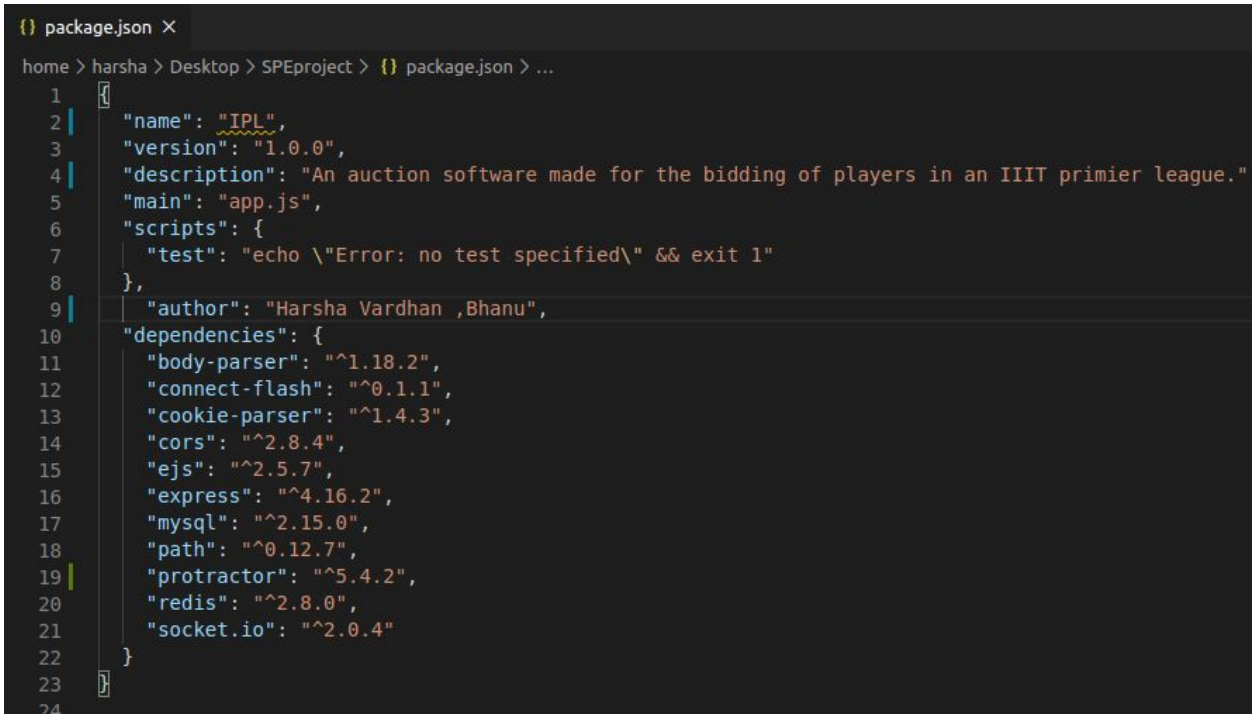


Fig 64

npm directly writes the details of every package use after installing them with the command npm i followed by the name of the package. But all the modules installed can't be transported from one system to another. So we can use the command " npm install" that installs all our dependencies by looking into the package.json.So our build step now involves running the command " npm install" as a step in the Jenkinsfile as shown below

```
stages {
  stage('Build'){
  steps{
    script{
    sh 'npm install'
    }
  }
  }
```

Fig 65

Step 3:
In this step we are building the docker images of our database,redis and website. We are using docker-compose to generate two images and connect their respective containers.But the problem arises when the container starts. The website container actually starts faster when compared to the database container. So whenever the website container sends a connection request to the database container, it is getting a connection refused error. To resolve that we have added a new check file that doesn't start the website container until the database container starts. To do that we will modify our Dockerfile as shown below

home > harsha > Desktop > SPEproject > Dockerfile

```
1  FROM node:8-alpine
2
3  EXPOSE 3000
4
5  WORKDIR /web
6
7  COPY . /web/
8
9  RUN apk add --no-cache bash
10 RUN chmod +x wait_for_it.sh
11
12
13 CMD ["./wait_for_it.sh","-t","20","database:3306", "--", "node", "app.js"]
14
```

Fig 66

The changes we're doing from the previous Dockerfile are:-
○ Installing bash in our image and making the run.sh file as an executable
○ And in the CMD part, we're running the run.sh file with a timeout of 20 secs and also to run the app.js only after connection to db is successful.

Now we can build the images using the command "docker-compose build" as shown in below

```
stage('Docker Image Build') {
  steps {
    script {
      sh 'docker-compose build'
    }

  }
}
```

Fig 67

This creates three images namey **harsha9199/db:latest** , **harsha9199/redis:latest** and **harsha9199/ipl:latest** all these images need to be pushed into our docker hub. We need to provide our DockerHub credentials to Jenkins so that it can use it to automate the processing of pushing the images. Go to credentials option available in the Dashboard page of Jenkins

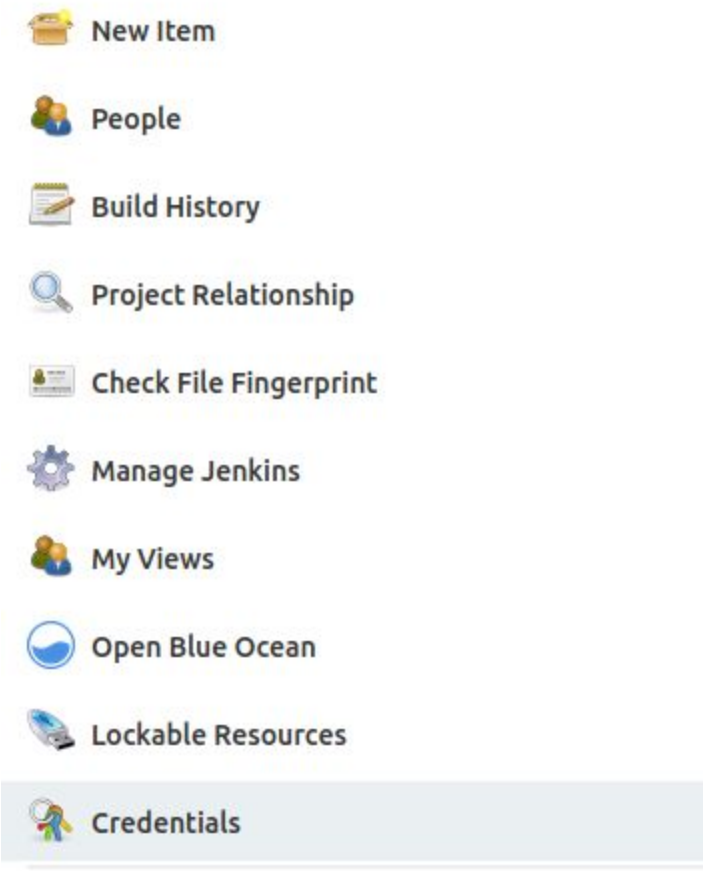


Fig 68

Add your Docker Hub credentials

Fig 69

Now we need to push our images created in the Jenkins file.with the help of following code we can do it

```
stage('Docker push') {
  steps {
    script {
      withDockerRegistry([ credentialsId: "DockerHub", url: "" ])
      {
        sh 'docker push harsha9199/db:latest'
        sh 'docker push harsha9199/redis:latest'
        sh 'docker push harsha9199/ipl:latest'

      }
    }

  }
}
```

Fig 70

Here harsha9199/db:latest , harsha9199/redis:latest and harsha9199/ipl:latest are the images created in the previous stage of Jenkinsfile.After successful push we can check if our images are successfully pushed by going to docker hub and checking them where after successful push we can find the images as below

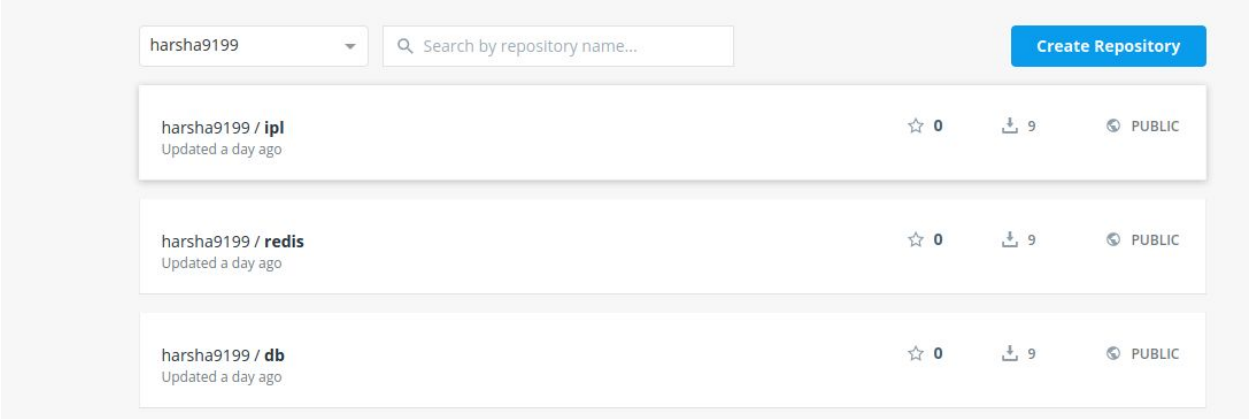


Fig 71

Now all that is left is to pull the image from the DockerHub and run the containers which will deploy the website in our local systems.

Step 4

For deploying our website to any machine we will be using Rundeck.
➢ Before that we have to set up a connection between Jenkins and Rundeck. To do that we have to perform these steps:-

- There is a plugin in Jenkins named “Rundeck” which will be useful when connecting to Rundeck.
- Go to Manage Jenkins and go to Configure System where there is an option of Rundeck which should filled as below

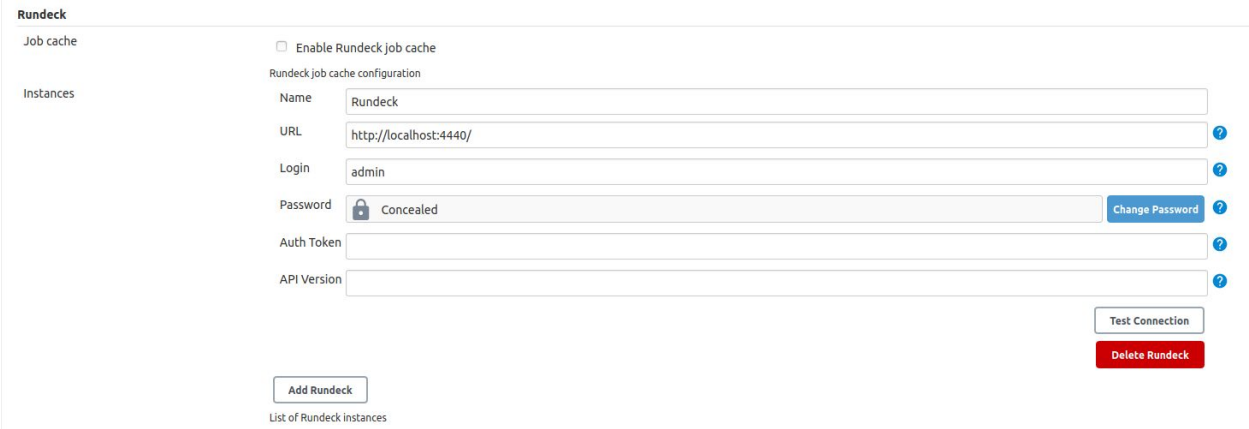


Fig 72

In the Rundeck option we have to provide the URL of where Rundeck is accessible and also with the username and password of Rundeck so that Jenkins can automatically authenticate and run a job in it.

Now in the Jenkins file we have to inform jenkins to use the Rundeck information which we have provided. So we have to set our Jenkinsfile for Rundeck as shown below

```
stage('Deploy with Rundeck') {
  agent any
  steps {
    script {
      step([$class: "RundeckNotifier",
      rundeckInstance: "Rundeck",
      shouldFailTheBuild: true,
      shouldWaitForRundeckJob: true,
      options: """
        BUILD_VERSION=$BUILD_NUMBER
      """,
      jobId: "b0a4af41-b909-43be-b39f-1db2b3c82b04"])
    }
  }
}
```

Fig 73

What we're telling Jenkins using this stage is that Jenkins has to trigger a rundeck process with Job Id mentioned in the jobId variable and has to wait until a reply comes from Rundeck confirming if the deployment is successful or not. And also we're Informing Jenkins to use the credentials provided previously in the rundeckInstance Variable. So whenever Jenkins runs this rundeck instance, execution of the Rundeck job takes place automatically which can be seen in the Activity for Jobs section for our Job which will look as

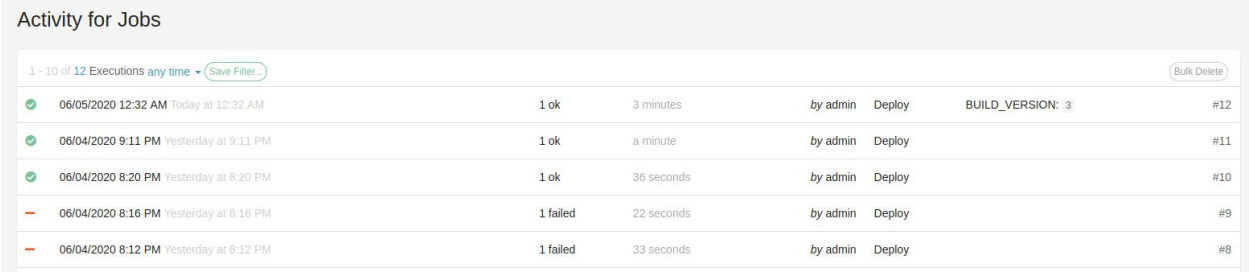


Fig 74

We can also check the individual execution status of each command and also output of each command of our Job

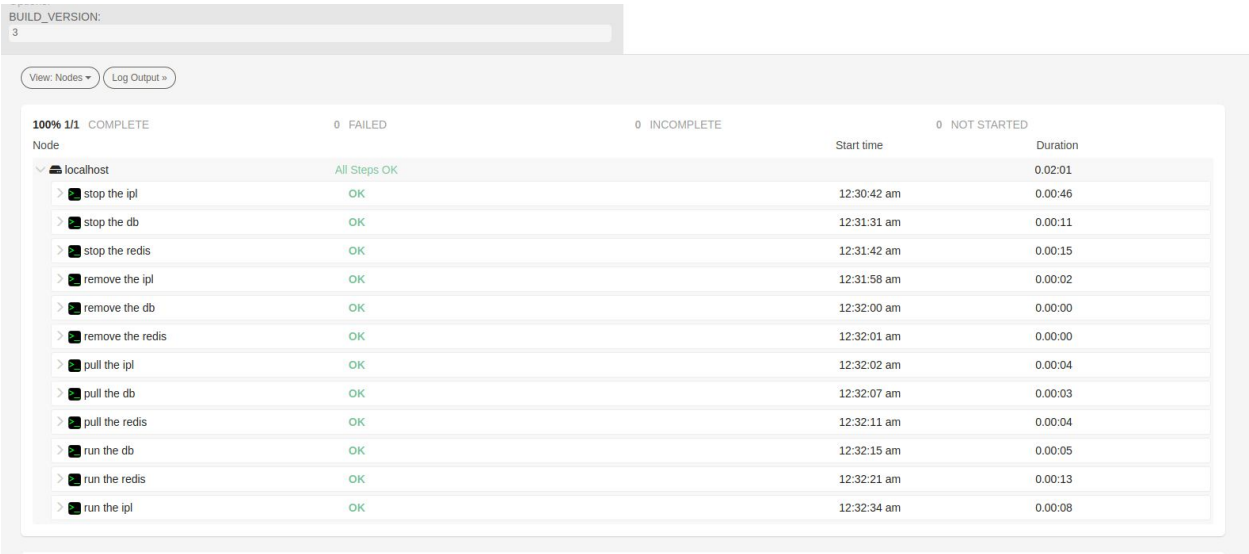


Fig 75

We can check if the deployment is successful or not in various ways. In our case as we're trying to deploy a website, it can be accessed in the http://localhost:3000 shown as

Step 5:
For running the test sites, we apply the below command

```
        steps {

        script {
            sh 'selenium-side-runner --output-directory=./testing/results -c
"browserName=chrome goog:chromeOptions.args=[headless]" --output-format=junit
./testing/devops.side'

                }
         }
         }
```

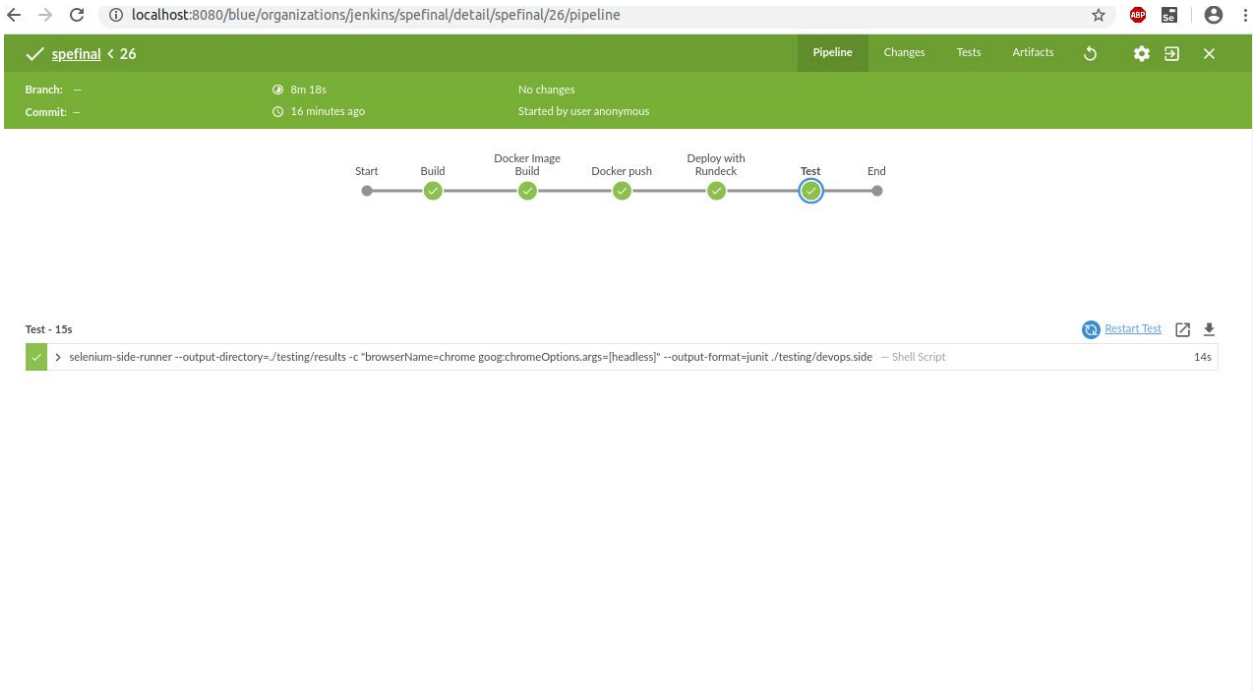


Fig 78

Now after successful deployment of our website, we also need to monitor the website. So for that we will be using ELK stack.

6.3 Monitoring

Here are some few visualizations from Filebeat and Metricbeat applications

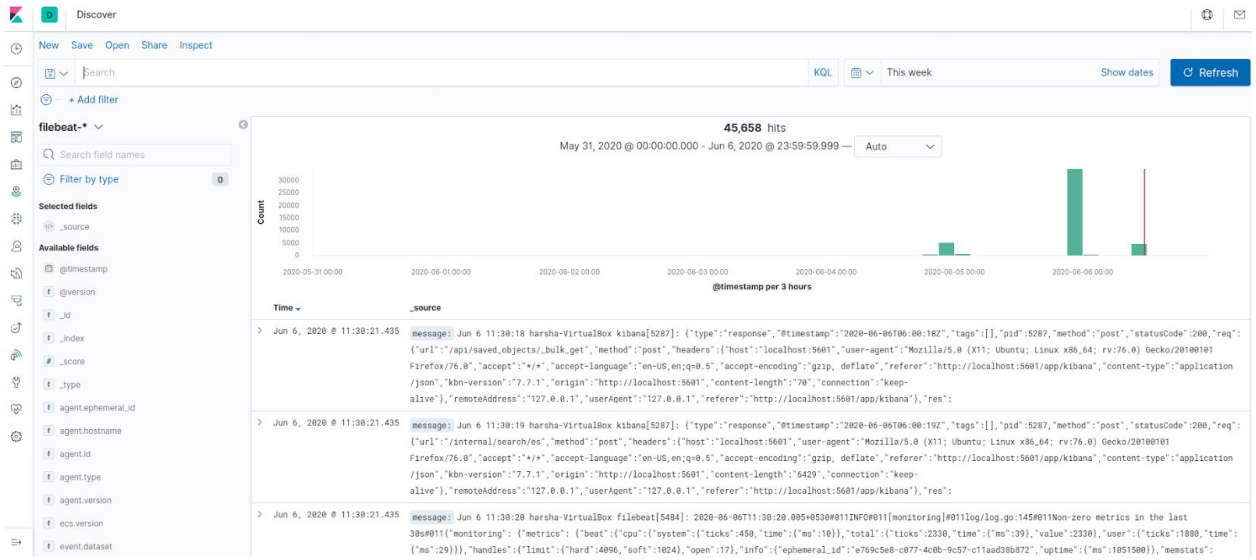


Fig 80

Filebeat logs

It is a bar data of logs from our application. This visualization is obtained from filebeat, where x-axis is a plot of log timestamps and y-axis is the frequency of logs at different Timestamps.

Below are Visualizations of various system logs and their frequency.

Figure 81 filebeat syslog

Figure 82 Filebeat, system logs

The following images fig 83-86 show the overview, host overview and container overviews. These visualizations are obtained from Metricbeat. They show various information such as CPU usage, RAM usage, Inbound and Outbound traffic of our application.

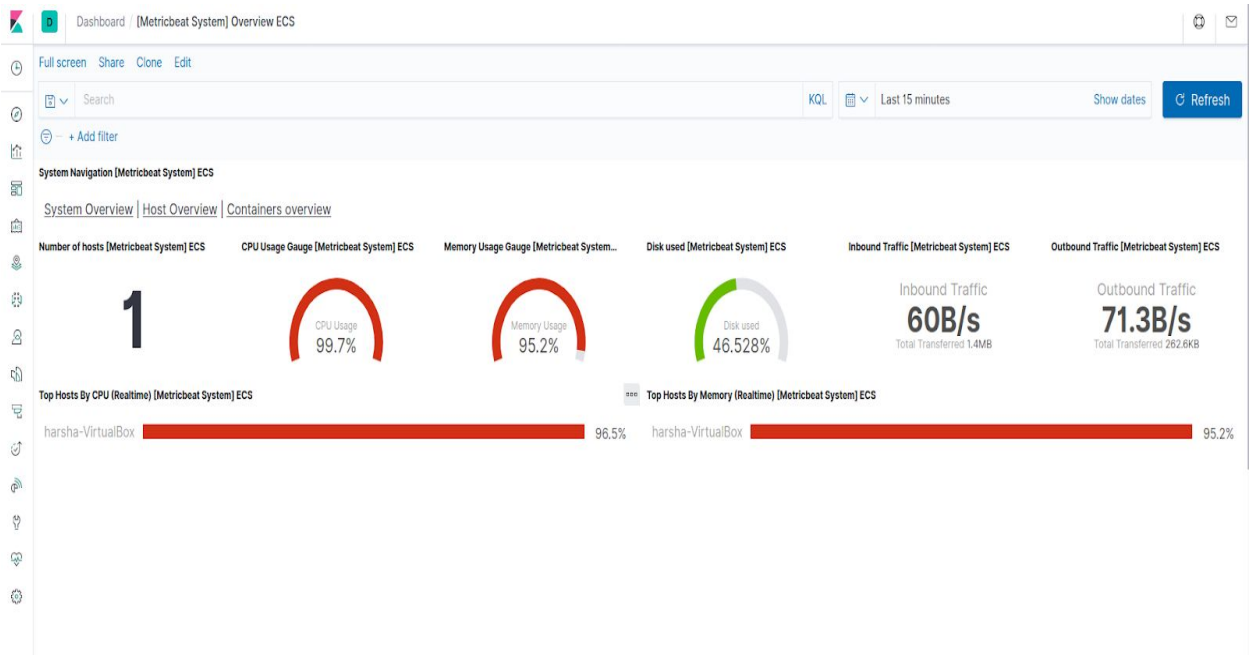


Figure 83 Metricbeat system overview ECS

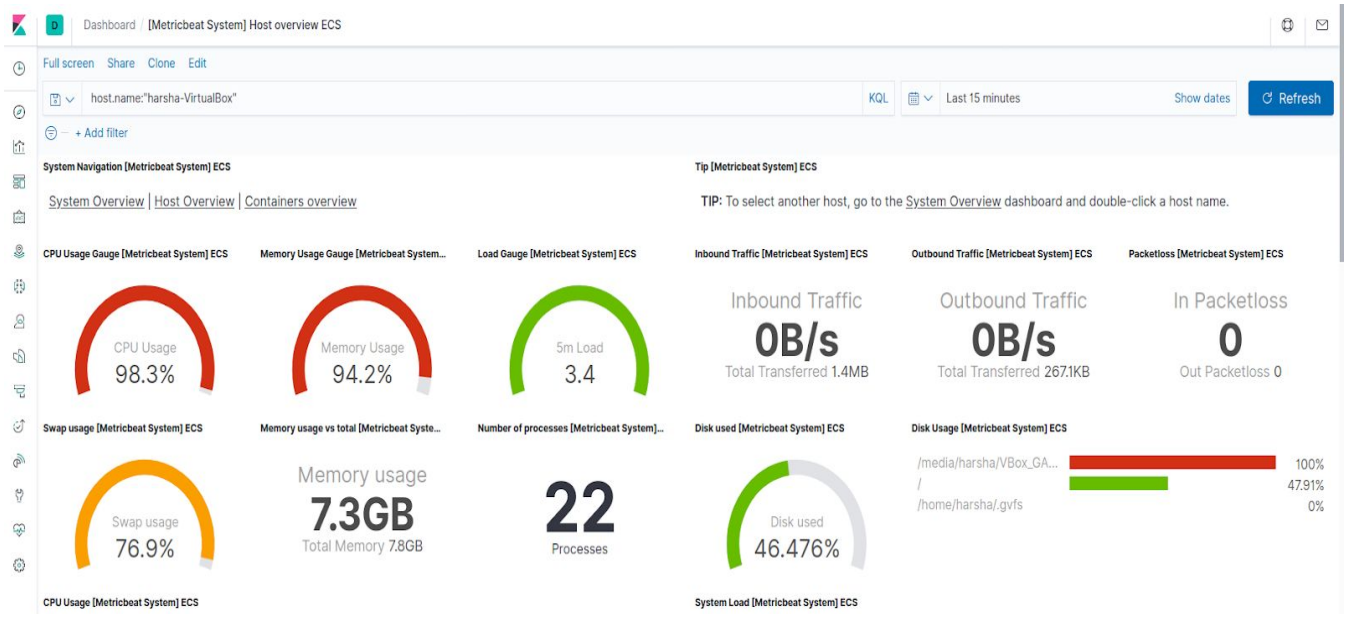


Figure 84 Metricbeat host overview ECS

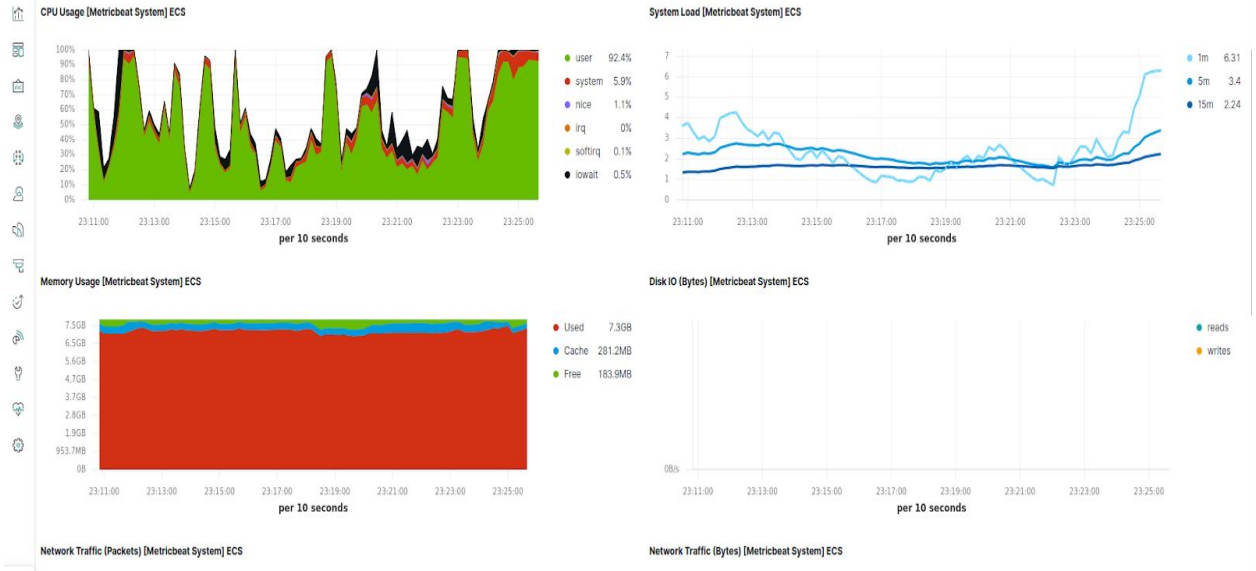


Figure 85 CPU usage Metricbeat System ECS

Figure 86

7. Experimental Setup

7.1 Functional requirements

Two kind of users : Buyers and Auctioneer

- Ability for auctioneer to start the bidding slot for buyers to bid on players.
- Auctioneer is able to decide the highest bidder and give the player to the highest bidder
- Auctioneer should be able to begin and end slot for bidding.
- Once the bidding of one player is done actionner can start new slot for next player
- Buyers should get log in credentials before starting of bidding slot
- Buyers should be able to bid once the auction starts
- Buyers should be able see others buyers bidding value along with bidder team name

7.2 Architecture and Design

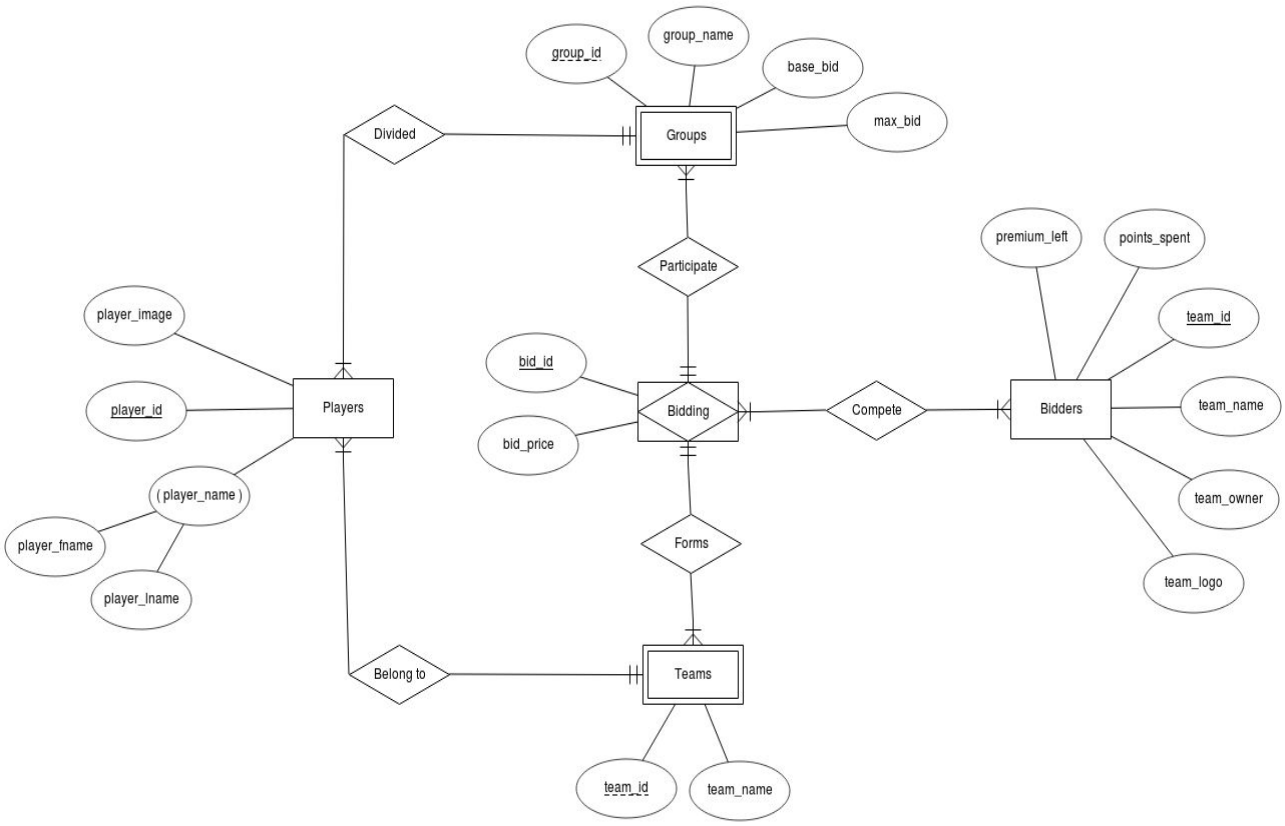


Figure 87 Design

8. Results

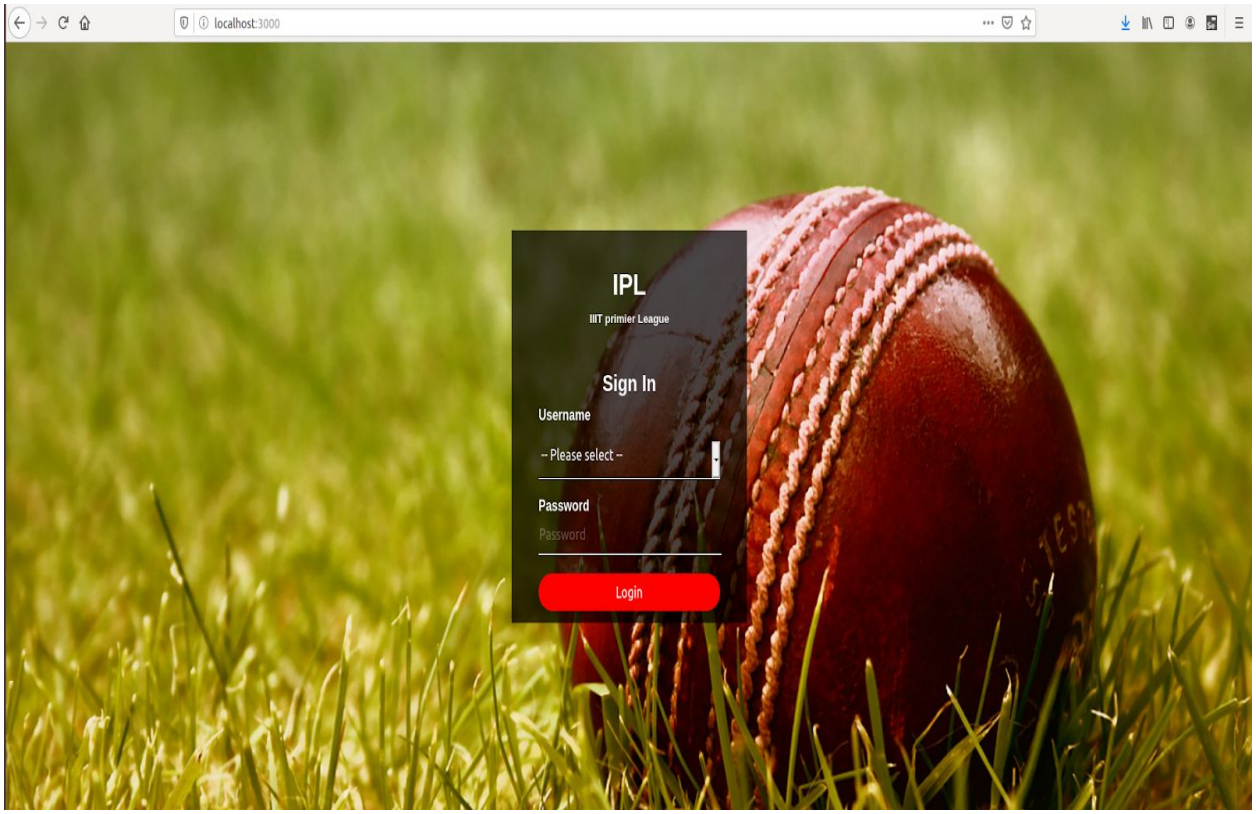


Figure 88: login portal for buyers

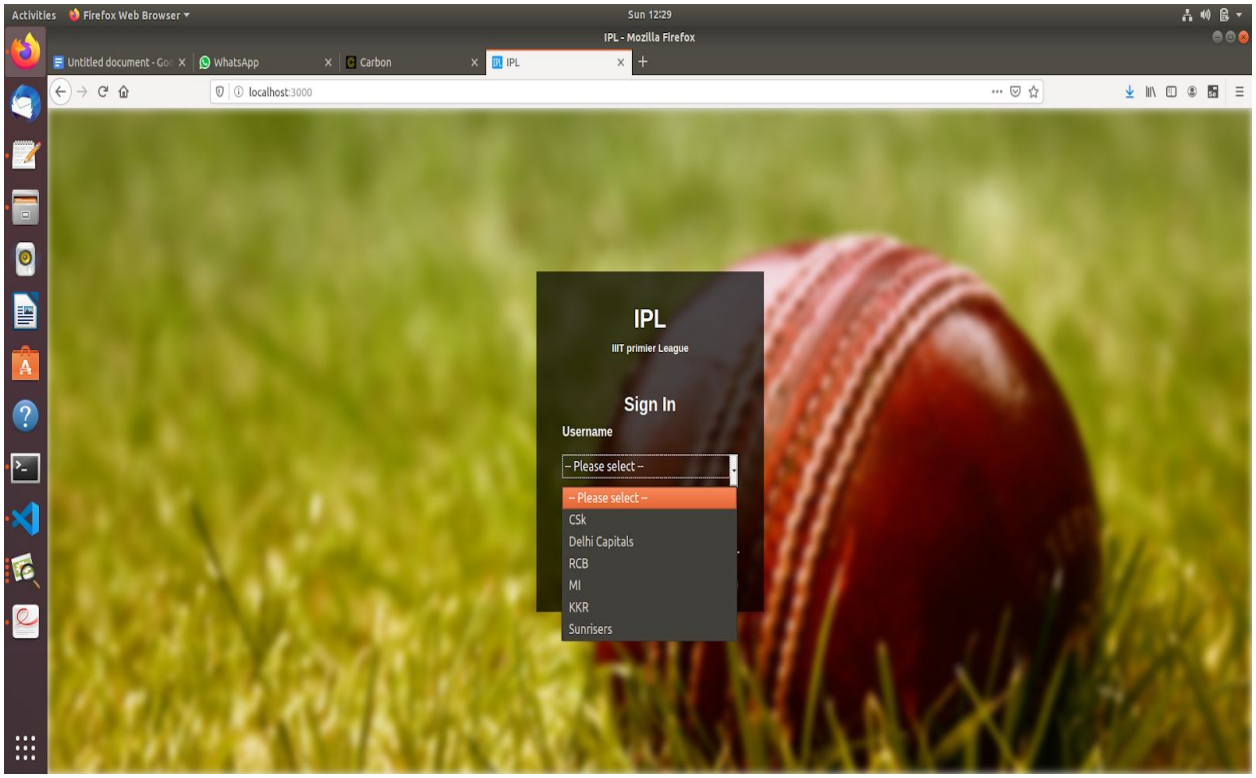


Figure 89 : buyers login with their credentials

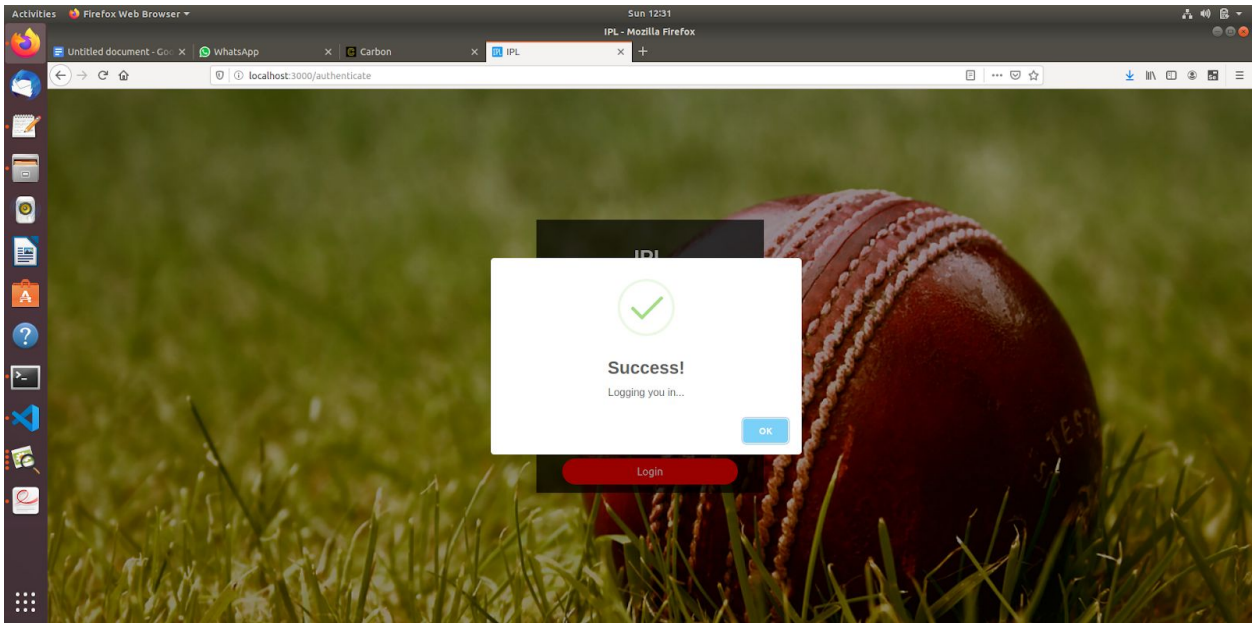


Fig 90 : success full login

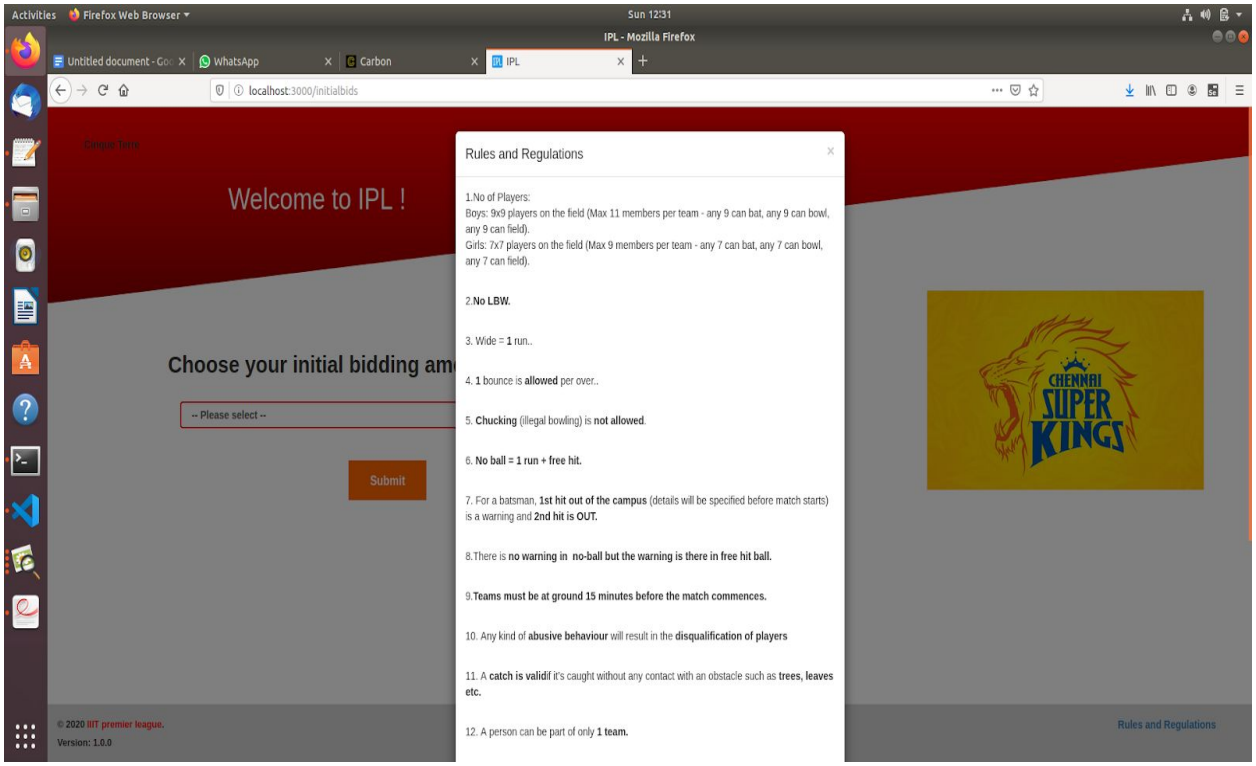


Figure 91 Rule and regulations

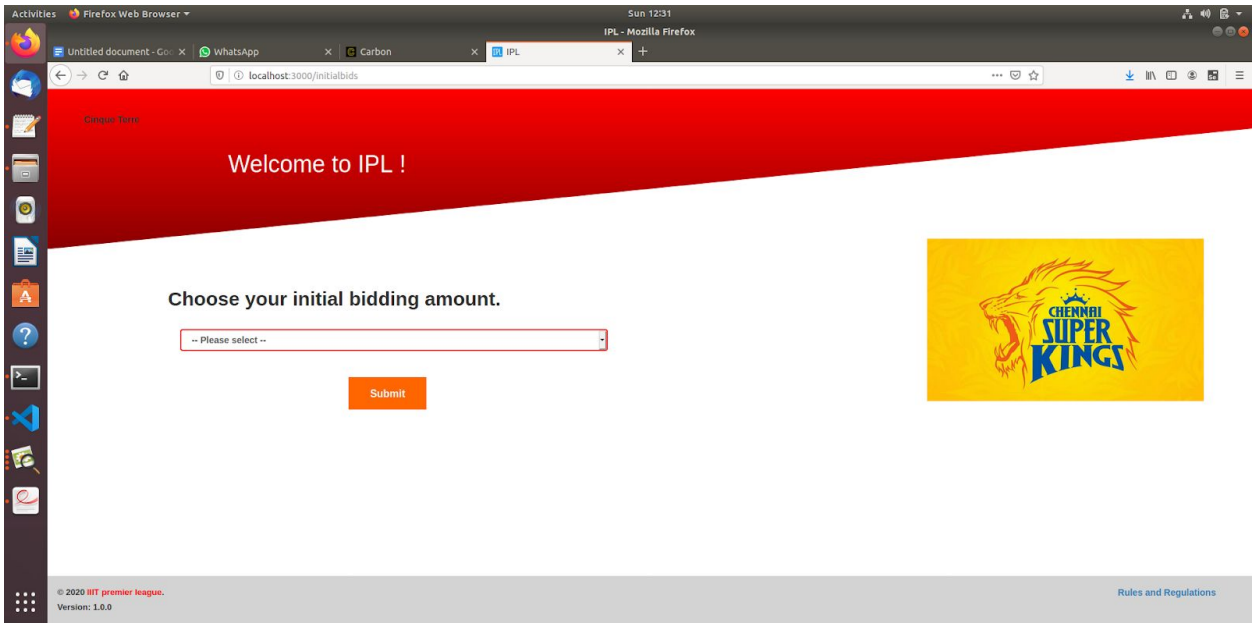


Figure 92 CSK Team initial bidding

Each team needs to select a bidding amount before entering the auction. There will be a base price

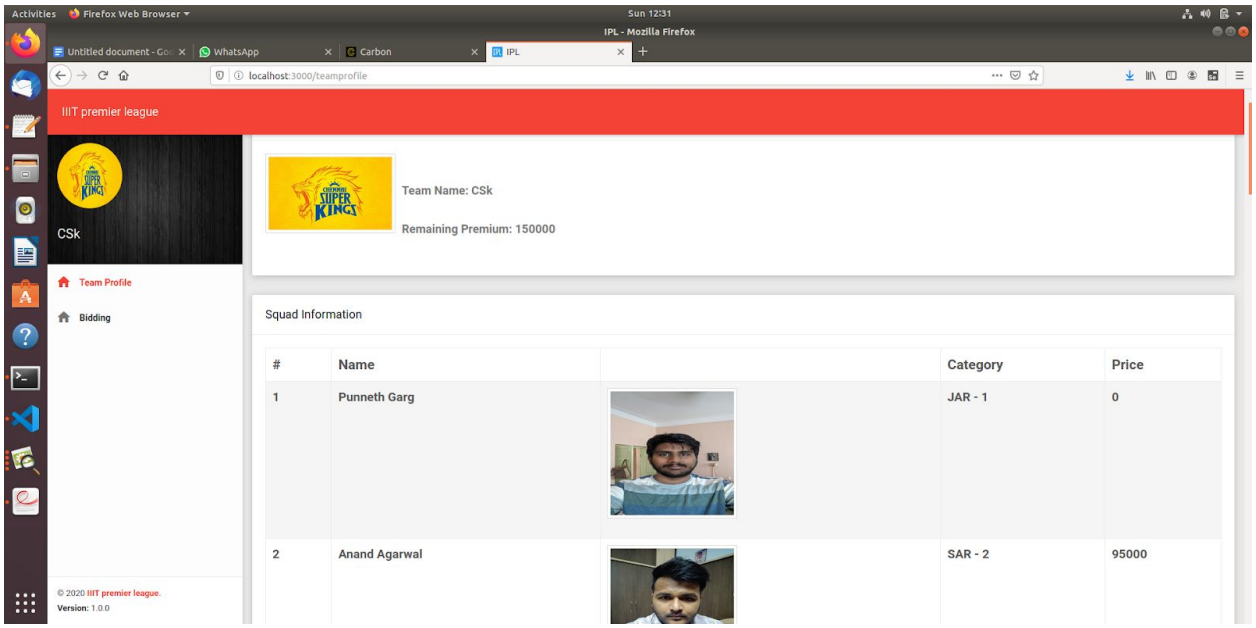


Figure 93 CSK Team Profile

The auction will be conducted by the auctioneer. Auctioneer need to login first with his credential at http://localhost:3000/gamemaster and he can start auction by starting slot.

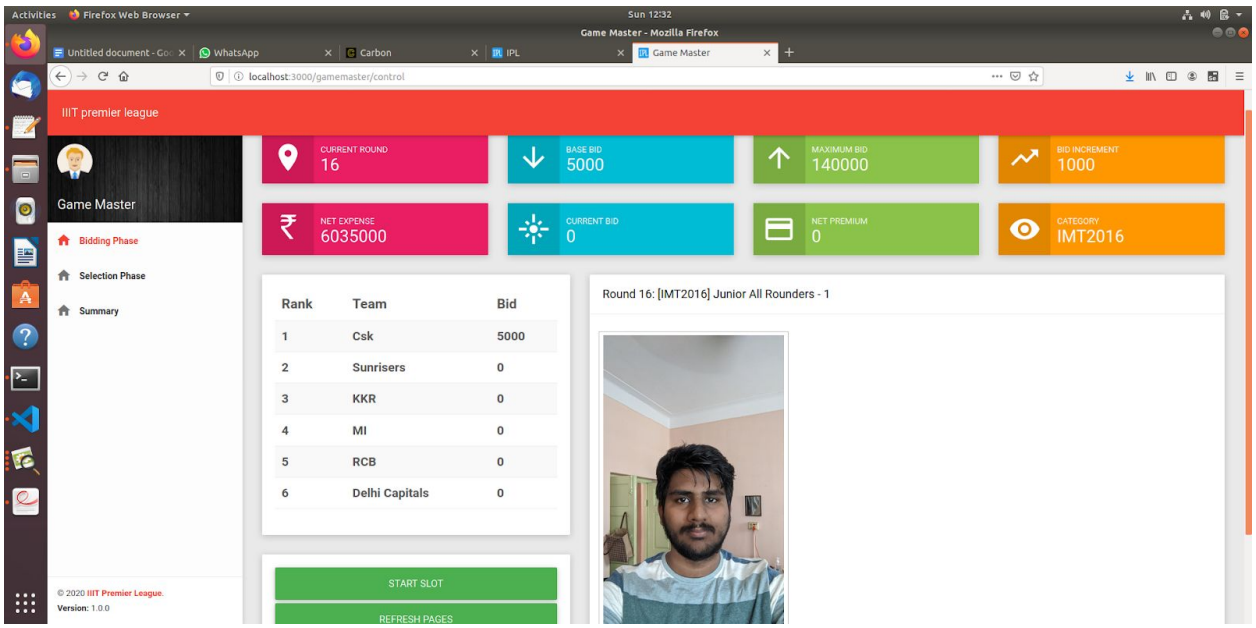


Figure 94 : auction for the player

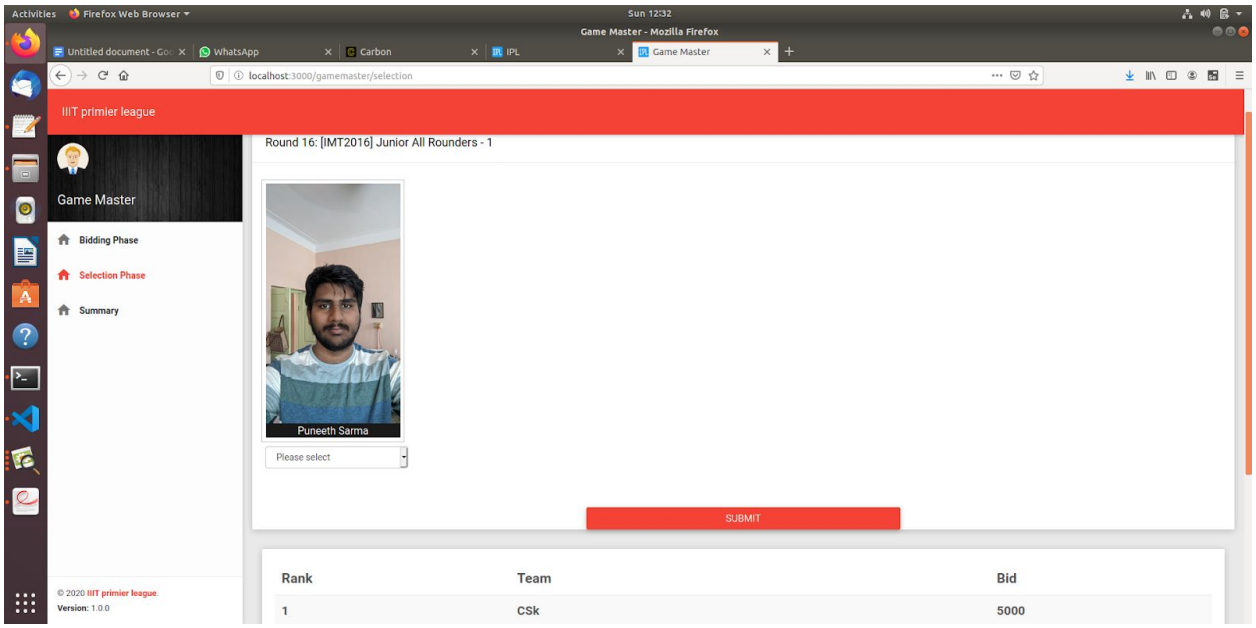
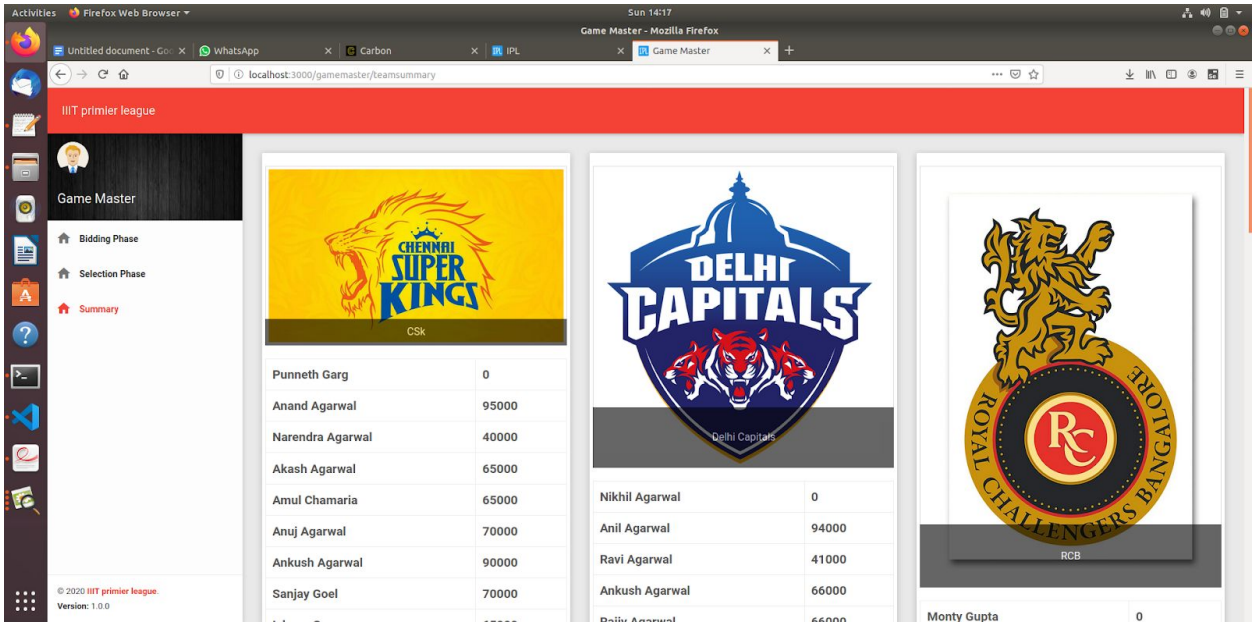


Figure 95: Once the auction is done player goes to the highest bidder

Figure 96: once all the auction is done team and their player info can be seen in summary



Once the auctioneer starts the slot buyer are able to bid. The bid tab only works when auctioneer starts the slot.

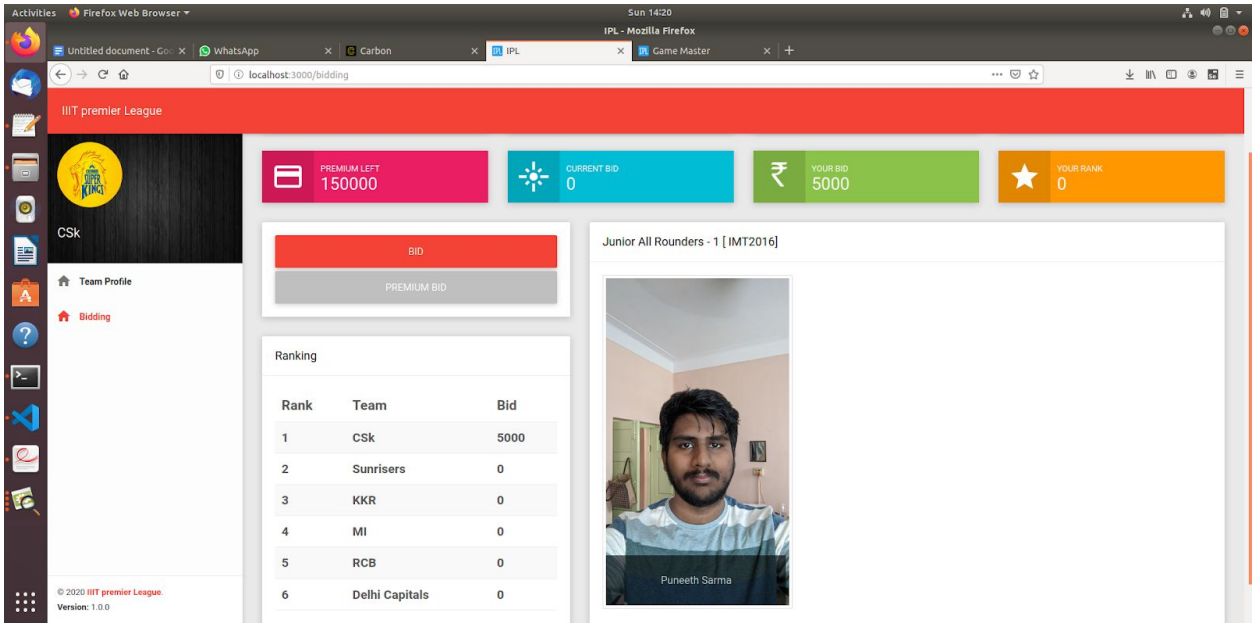


Figure 97 :

9. Conclusion

By using Devops tools to automate the SDLC lifecycle, we are able to successfully deploy our project. We can deploy new changes from anywhere just pushing new changes into github and there by triggering the jenkins

10. Future Works

The present application does not maintain statistics of any match it only contains statistics of players and match schedules so we can create another database for maintaining match statistics.we can also use the same application for other games with slight changes.

11. References

[1] what is devops :
https://www.toptal.com/insights/innovation/what-is-devops
[2] Dockerizing and pythonizing redis:
https://medium.com/better-programming/dockerizing-and-pythonizing-redis-41b1340979de
[3] Installation of Docker:
https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04
[4] Installations of Elasticsearch:
https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-elasticsearch-on-ubuntu-18-04
[5] Tutorial - How To Install Elasticsearch, Logstash, and Kibana (Elastic Stack) on Ubuntu 18.04,
https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-ubuntu-18-04
[6] Selenium Installation:
https://tecadmin.net/setup-selenium-chromedriver-on-ubuntu/