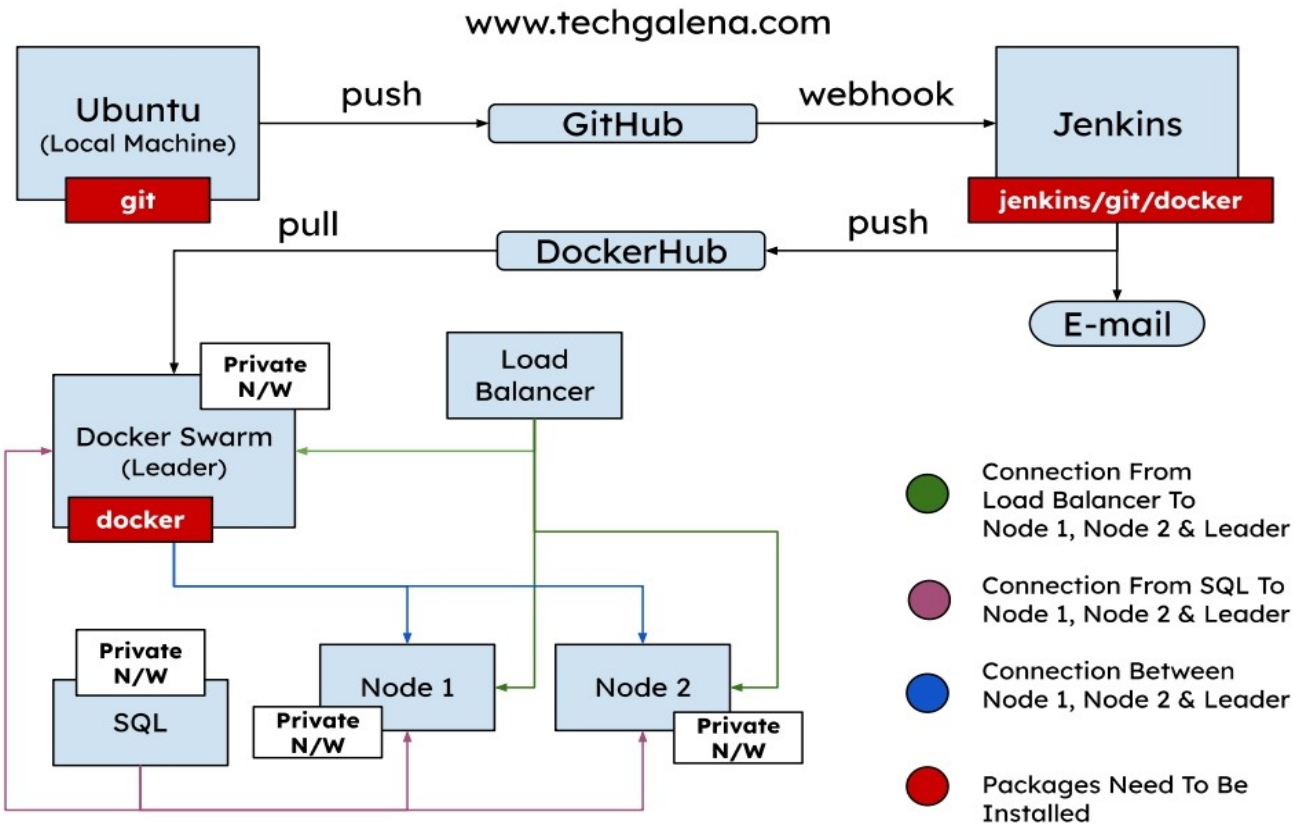
Devops project 1



Deploy local machine (ubuntu) + Launch all instance ensure all are in same subnet in our case it is us -east 1a .

1st instance : Jenkins

Default vpc / Subnet us -east 1a /Enable public ip

2nd instance : Docker swarm leader

Default vpc /Subnet us -east 1a /Disbale public ip

3rd instance : Docker swarm node 1

Default vpc /Subnet us-east 1a/Disable public ip

4th instance : Docker swarm node 2

Default vpc/Subnet us-east 1a/Disable public ip

5th instance : Sql server

Default vpc/ Subnet us -east 1/Disable public ip

6th instance : load balancer

Default vpc /Subnet us-east 1/Enable public ip

Step 1:

Now linking githhub pvt repo to jenkins :

Reference :Integrate Git Pvt repo to Jenkins

The public and pvt key doesn't work in it as github has removed this feature on 31st august 2021 .

Now we shall use other method : access token .

Git hub -> left hand setting -> developer setting -> personal access token -> token classic -> generate new token

Select scope (repo , user.email and admin read.org)

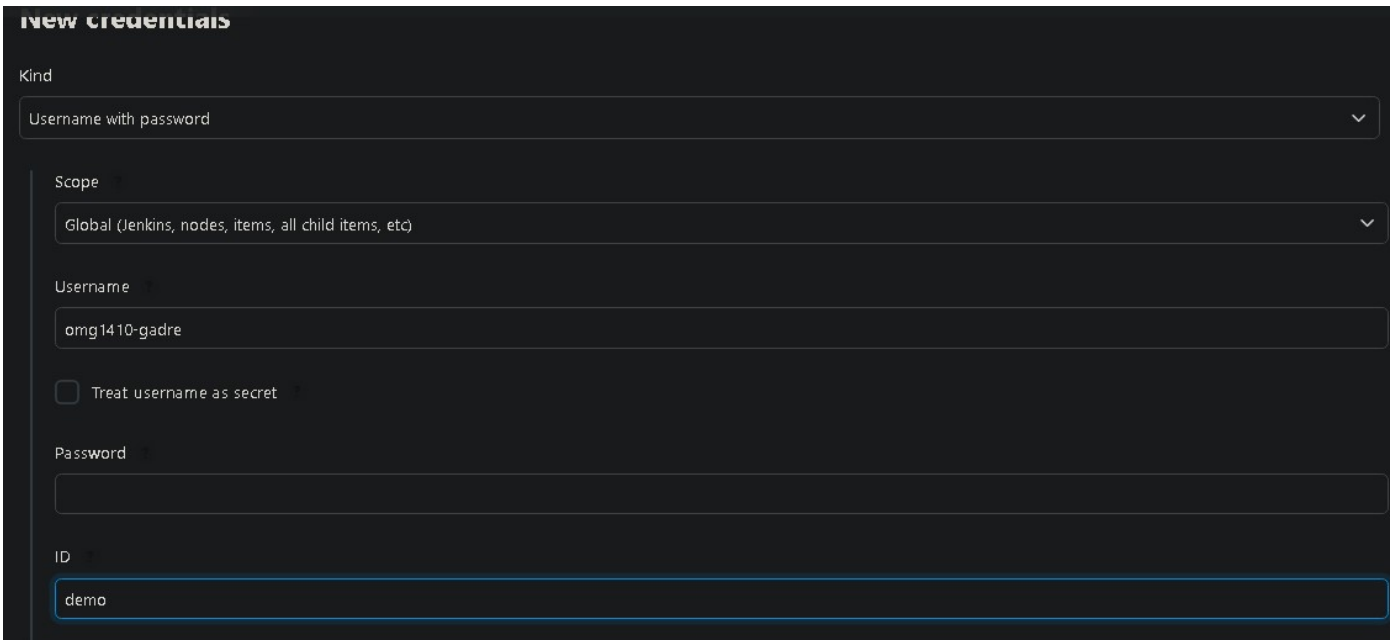
Generate token : copy that token

Go to jenkins -> credentials -> add credentials ->

Username : add your username of git

Id : assign id

Password : enter token

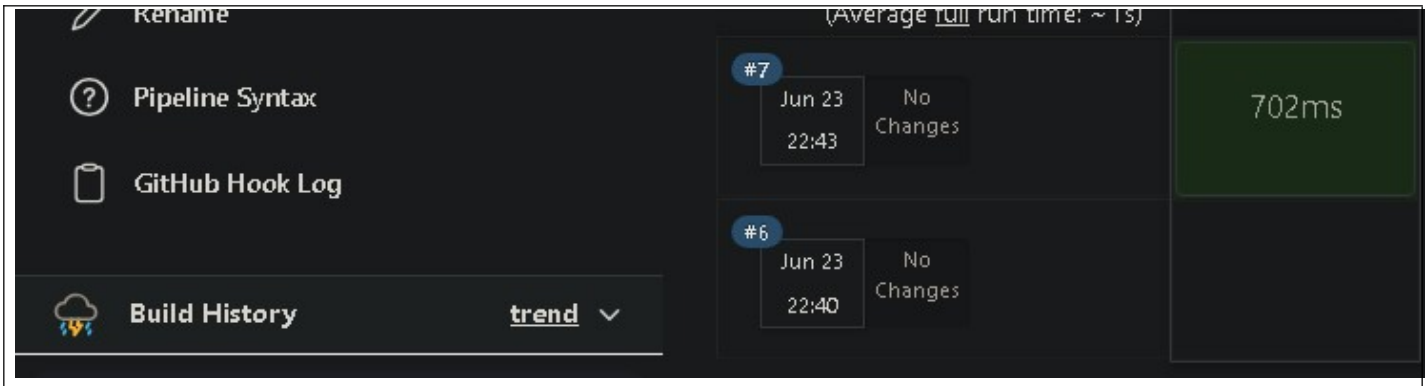


Add Webhook as well : Git -> repo -> setting -> add webhook

To convert into pipeline : in git credentials id : insert id you have assigned and https url .

```
pipeline {
   agent any

   stages {
      stage('git checkout ') {
         steps {
            git credentialsId: 'Jenkins_Token', url: 'https://github.com/omg1410-gadre/Project1.git'
            echo 'git login'
         }
      }
```

```
pipeline {
    agent any

    stages {
        stage('git checkout ') {
            steps {
                git credentialsId: 'Jenkins_Token', url: 'https://github.com/omg1410-gadre/Project1.git'
                echo 'git login'
            }
        }
```



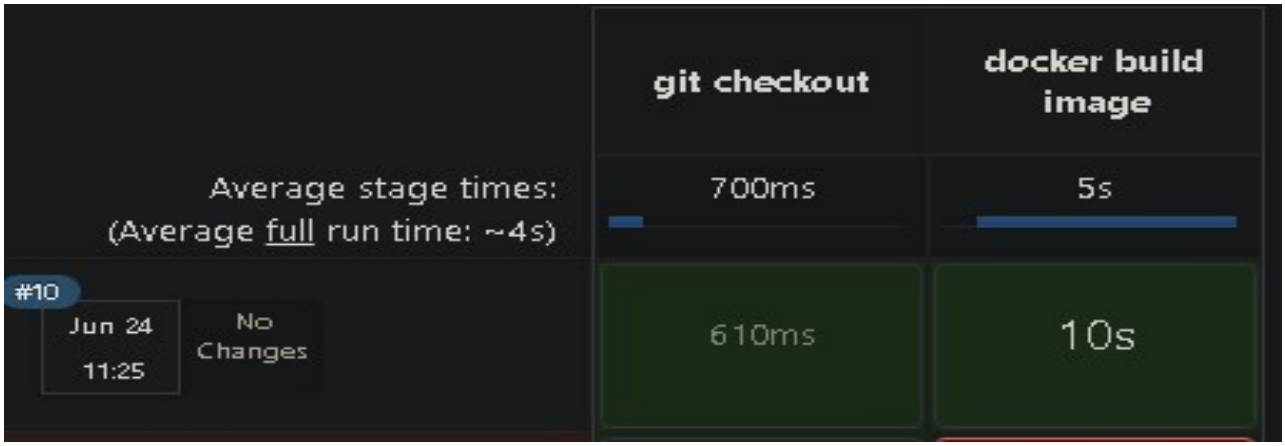
Step 2 :

Building image :

At Jenkins EC2 :

```
$sudo chown nobody:nogroup /var/run/docker.sock
$sudo chown nobody:nogroup /var/lib/docker
$sudo chmod 7777 /var/run/docker.sock
$sudo chmod 7777 -R /var/lib/docker
```

```
stage('docker build image  ') {
        steps {
           sh 'docker build -t project1 .'
           echo 'Build image completed'
        }
    }
```



Step 3:

Docker Login :with variable no uname and pwd to be vissible in script

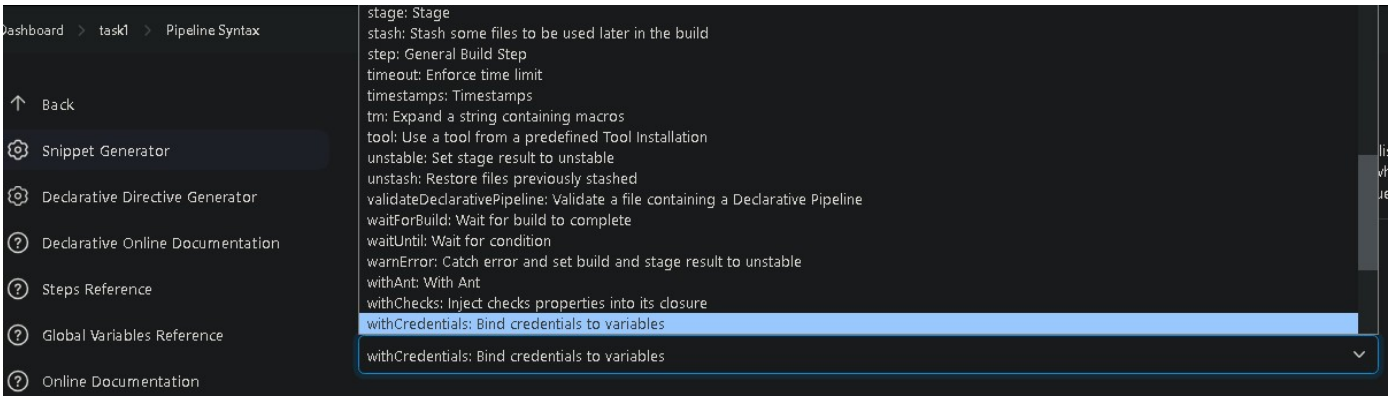
Dockerhub : setting - > security -> generate access token

Jenkins -> manage jenkins -> credentials ->

Username : your uname of docker hub

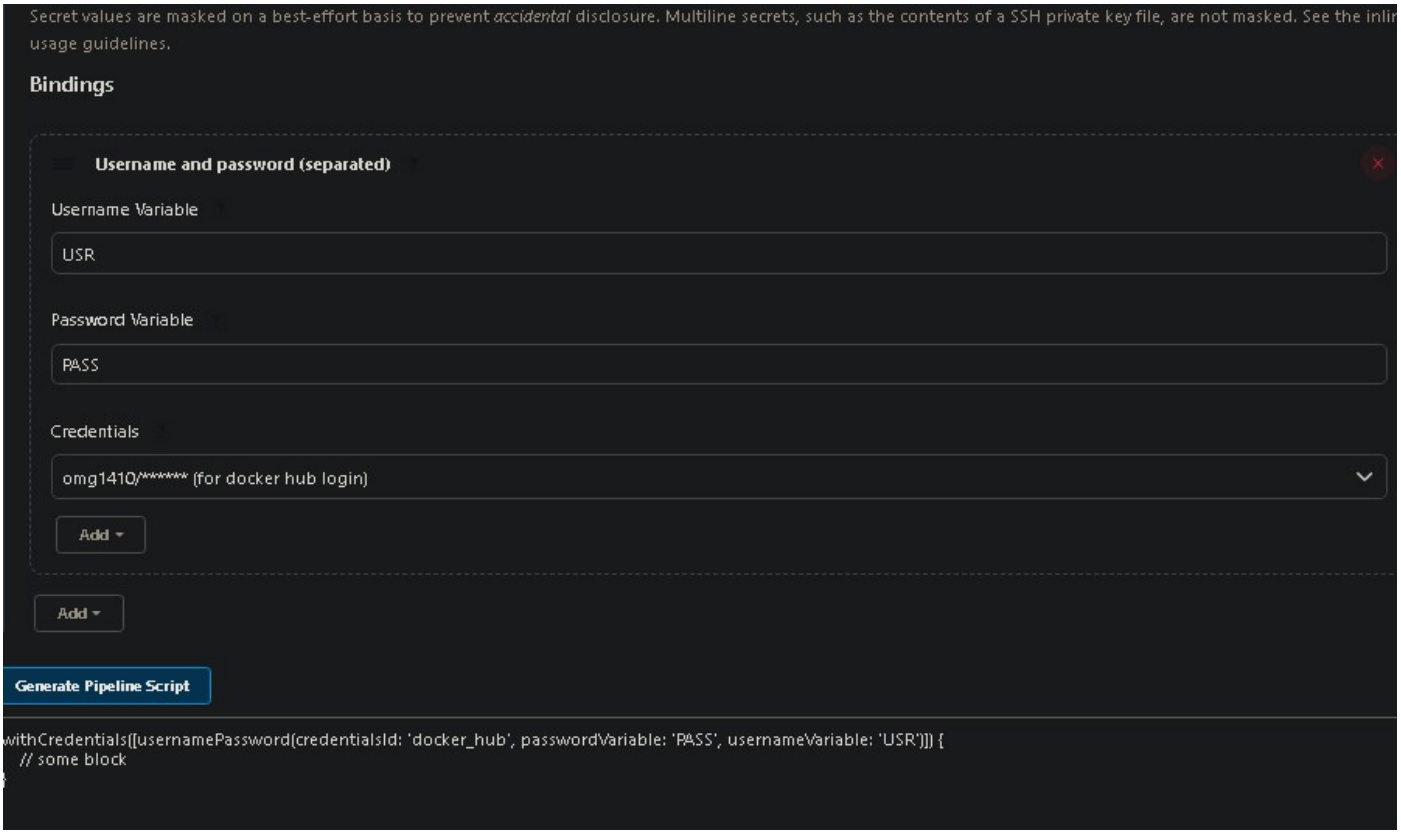
Pwd : token of docker hub
Id : assign any

Go to pipeline : generate pipeline scrip : *select option Bind credentials to variable*



Select uname and pwd (separated) option

Provide variables and in credentials select which you have created id which you have assigned before.



Generate pipeline script : copy as it is in pipeline

```
stage('docker login') {
        steps {
            withCredentials([usernamePassword(credentialsId: 'docker_hub',
passwordVariable: 'PASS', usernameVariable: 'USR')]) {
                echo 'docker login complete '
```

```
        }
    }
}
```

Step 4 :

Push image : Install docker and docker pipeline plugin.

Pipeline :

```
 stage('docker push image') {
        steps {
          sh 'docker push omg1410/project1:latest'
           echo 'docker push complete '
        }
    }
```

```
    }
    stage('docker push image') {
        steps {
          sh 'docker push omg1410/project1:latest'
           echo 'docker push complete '
        }
    }
}
```

Step 5 :

Step : Do openssh from public n/w to pvt n/w Jenkins to docker swarm leader

Caution already install pem file while launching instance if not then : use puttygen

Open putty gen ->Load pvt key I.e ppk file -> conversion -> export open ssh -> save

Go to jenkins putty :

$passwd root

$su root

#cd /home/admin

#nano leader.pem ***(Insert putty gen key here)***

#chmod 7777 leader.pem
#chown nobody:nogroup leader.pem
#su jenkins

$ssh -i leader.pem admin@pvt ip (*trial from putty)*

Pipeline :

```
stage('jenkins login to leader') {
        steps {
          sh 'ssh -i /home/admin/leader.pem admin@172.31.24.151'
           echo 'jenkins login to leader complete '
        }
    }
```

```
    }
    stage('jenkins login to leader') {
        steps {
          sh 'ssh -i /home/admin/leader.pem admin@172.31.24.151'
           echo 'jenkins login to leader complete '
        }
    }
}
}
```

	git checkout	docker build image	docker login	docker push image	jenkins login to leader
Average stage times: (Average full run time: ~4s)	943ms	1s	206ms	892ms	475ms
#23 Jun 24 14:34 No Changes	919ms	1s	148ms	1s	593ms
#22 Jun 24 14:33 No Changes	1s	1s	131ms	1s	357ms failed

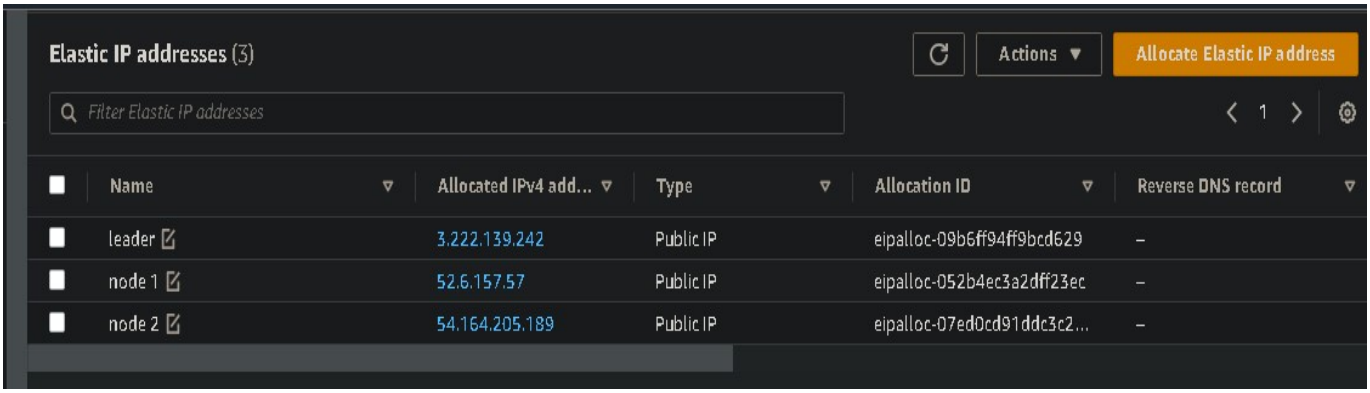
Alternative way :

Same step of above the only difference is manually login from putty to swarm leader by
$ ssh -I filename admin@pvt ip
And then go to .ssh and paste key in authorised key file .

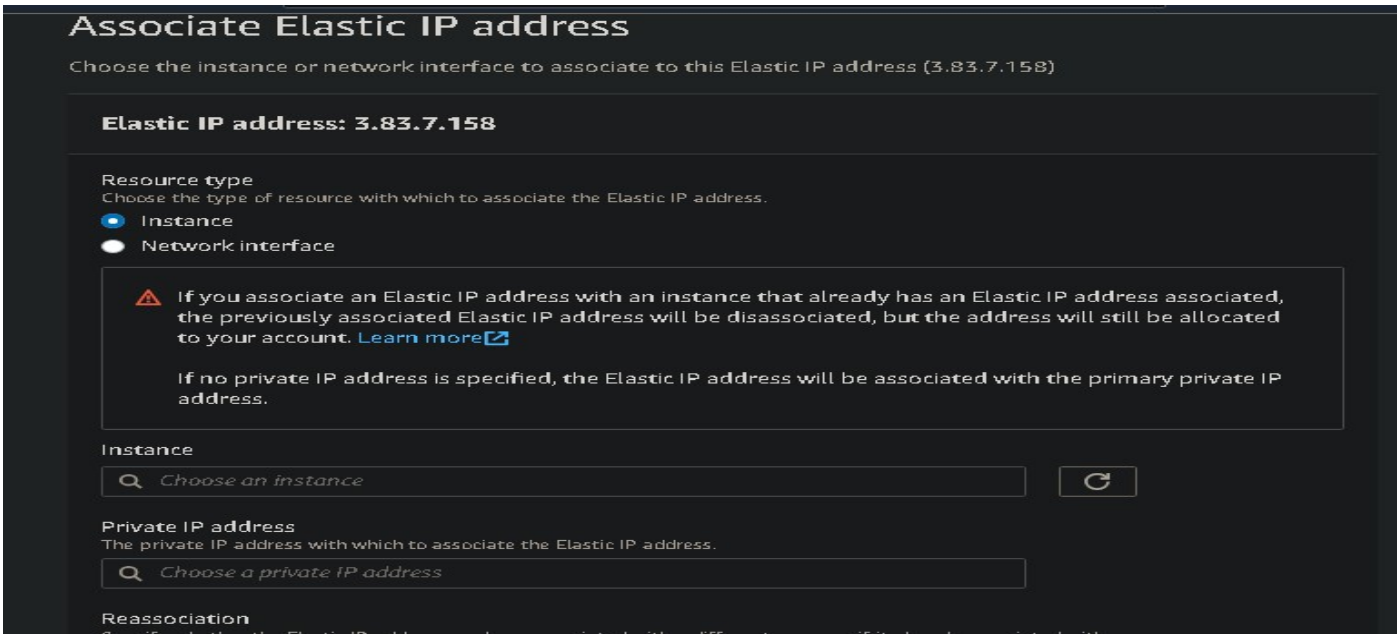
Step 6: Manual setup of docker swarm :

As we want to run commands on pvt n/w we need internet gateway so we need to create a elastic ip :

Your VPC -> Elastic IP -> generate elastic IP .and allocate that ip address to pvt n/w.



To bind elastic ip to pvt n/w -> click on elastic ip -> associate elastic ip



Leader.pen , node1.pem and node2.pem are files in jenkins which containes openssh-key of leader ,node1 and node 2
Change file permission as well before using (chown and chmod)

Jenkins :
$ssh -i leader.pem admin@leader.pem :
$sudo apt-get update
$sudo apt-get instal docker.io
$sudo docker swarm init (save token in file)

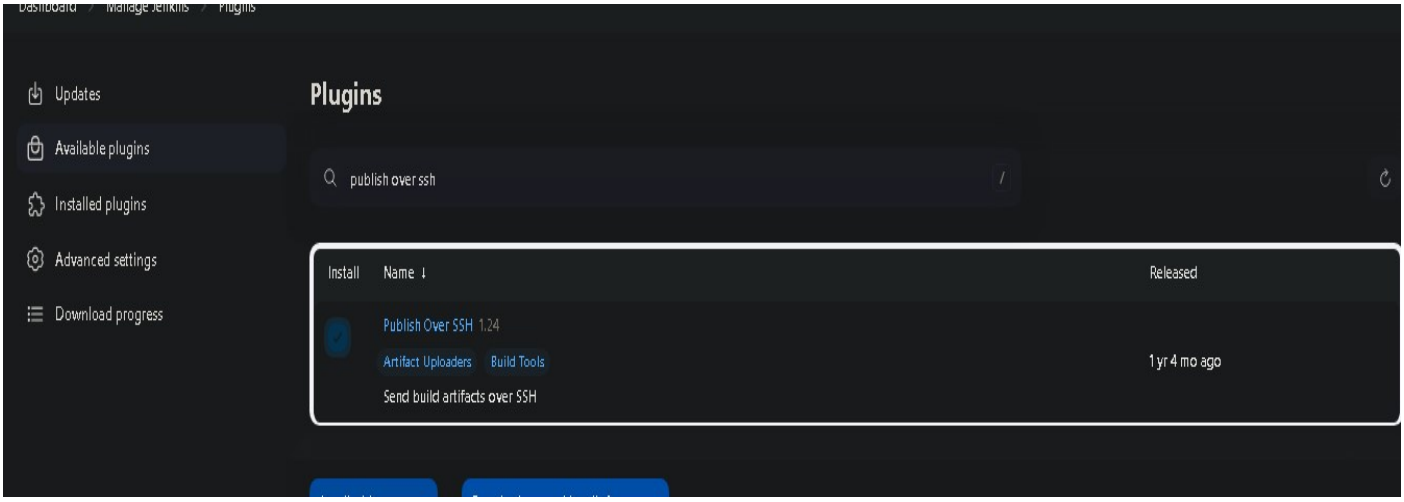
Jenkins :
$ssh -i node1.pem admin@node 1 ip
$sudo apt-get update
$sudo apt-get install docker.io
$sudo docker --swarm token…….

Jenkins :
$ssh 0i node2.pem admin@node 2 ip

$sudo apt-get update
$sudo apt-get install docker.io
$sudo docker --swarm token……..

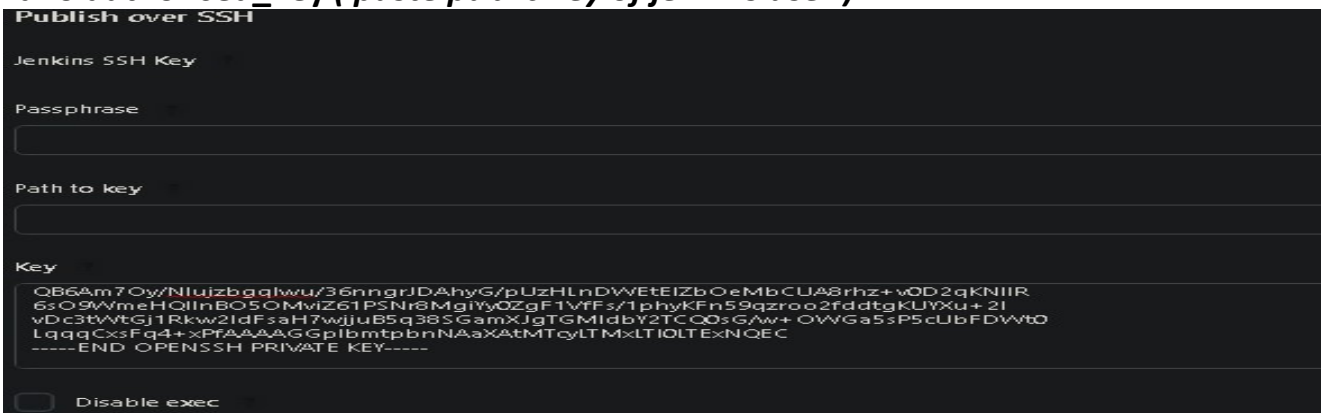
Step 7: Publish over ssh :

Install publish over ssh plugin :
Manage jenkins -> available plugin ->
After installation of plugin restart the jenkins as well

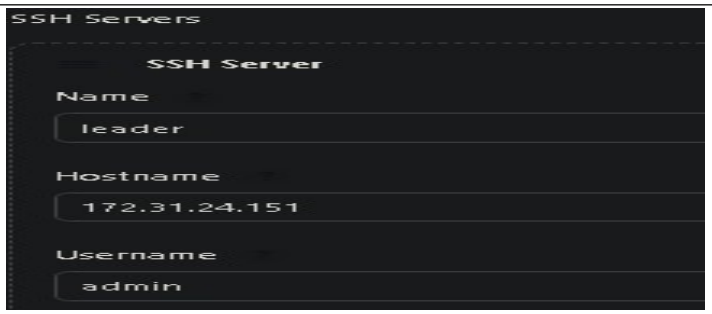


Manage jenkins -> system -> publish over ssh section .

Add pvt key of jenkins here and before that add public key of jenkins in authorised key file of where you want to connect .
Leader :
$ cd .sshh
$nano authorised_key ***(paste public key of jenkins user)***



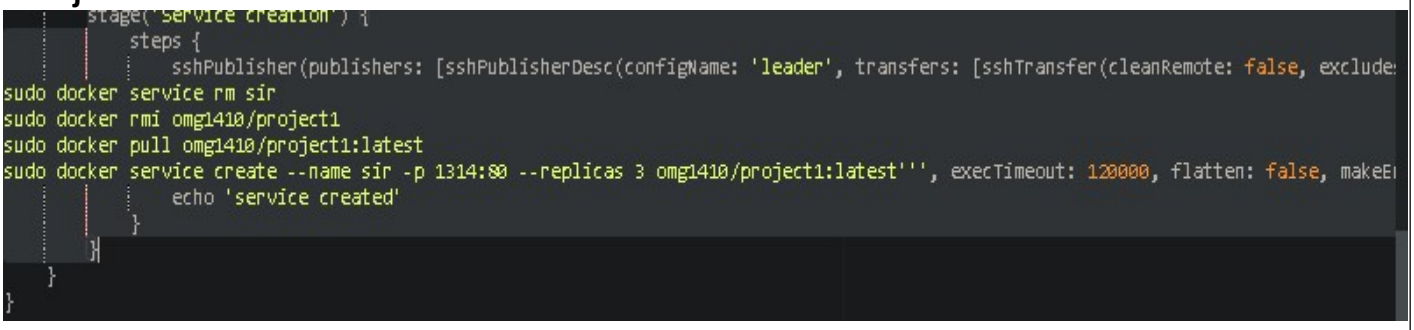
Name : xyz | hostname : pvt ip |username : admin

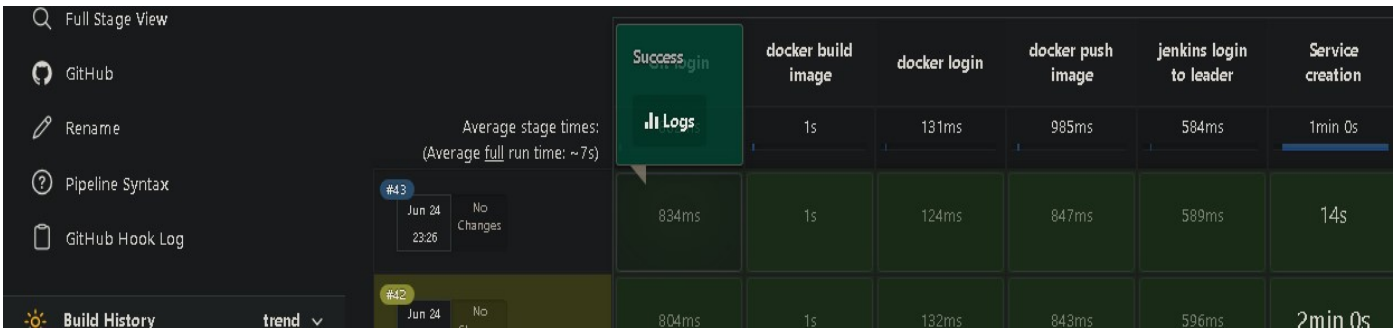


Command to run after nodes have joined leader .

Jenkins dashboard -> Pipeline - >Pipeline syntax : publish over ssh

```
stage('Service creation') {
        stage('Service creation') {
         steps {
            sshPublisher(publishers: [sshPublisherDesc(configName: 'leader', transfers:
[sshTransfer(cleanRemote: false, excludes: '', execCommand: '''sudo docker login -u
omg1410 -p Omkar1410@
sudo docker service rm sir
sudo docker rmi omg1410/project1
sudo docker pull omg1410/project1:latest
sudo docker service create --name sir -p 1314:80 --replicas 3 omg1410/project1:latest''',
execTimeout: 120000, flatten: false, makeEmptyDirs: false, noDefaultExcludes: false,
patternSeparator: '[, ]+', remoteDirectory: '', remoteDirectorySDF: false, removePrefix: '',
sourceFiles: '')], usePromotionTimestamp: false, useWorkspaceInPromotion: false,
verbose: false)])
            echo 'service created'
         }
      }
```





Step 8 : creation of load balancer :

Manuall
Take instance :
$sudo apt-get install squid
$sudo nano /etc/squid/squid.conf

1st change : 1200 :
acl project_users dst domain <u>public ip of load balancer</u>

```
acl localnet src 192.168.0.0/16        # RFC 1918 local private network (LAN)
acl localnet src fc00::/7              # RFC 4193 local private network range
acl localnet src fe80::/10             # RFC 4291 link-local (directly plugged) machines
acl project_users dstdomain 54.234.86.153
acl SSL_ports port 443
acl Safe_ports port 80          # http
```

2nd Change : Line 1400
http_access allow project-users

```
# Adapt localnet in the ACL section to its
# from where browsing should be allowed
#http_access allow localnet
http_access allow localhost
http_access allow project_users
```

3rd Change : line 1900
http_port *port where you have created service* vhost

```
# Squid normally listens to port 3128
http_port 3128
http_port 80 vhost
http_port 1314 vhost
```

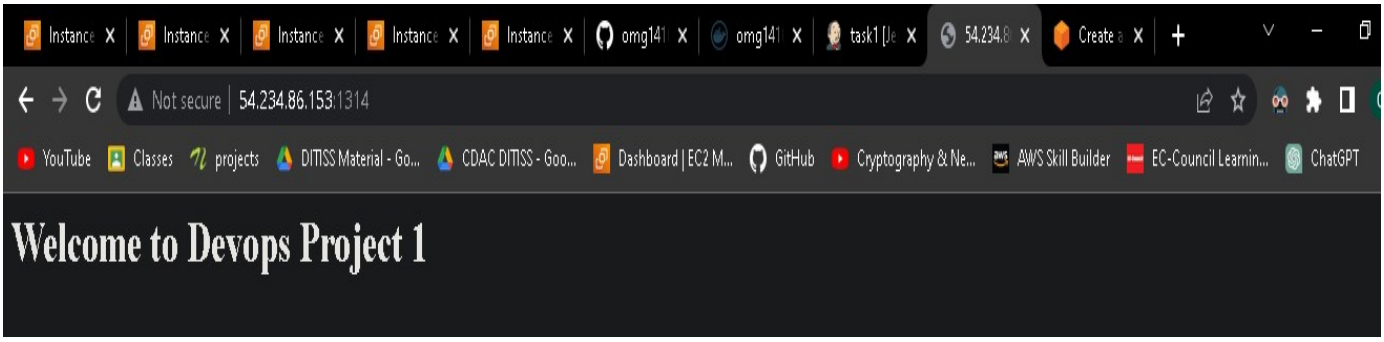
4th Change : cache_peer

Cache_peer pvt ip of leader/node1/node2 parent port 0 no-query originserver round-robin weight 1 name=

```
admin@ip-172-31-17-170: /etc/squid
  GNU nano 5.4                                    squid.conf
#       proxy-only      objects fetched from the peer will not be stored locally.
#
#Default:
# none
cache_peer 172.31.24.151  parent 1314 0 no-query originserver round-robin weight=1 name=one
cache_peer 172.31.20.193  parent 1314 0 no-query originserver round-robin weight=1 name=two
cache_peer 172.31.21.89  parent 1314 0 no-query originserver round-robin weight=1 name=three
#  TAG: cache_peer_access
#       Restricts usage of cache_peer proxies.
```

Semi Test 1 :

Push index.html from ubuntu it should be visible on load balancer public ip



Step 9 : Setting up SQL server :

$sudo apt-get update
$sudo apt-get install mariadb-server
$cd /etc/mysql/mariadb.conf.d
$sudo nano 50-server.conf

Bind-address = ip of databse server I.e own ip

```
admin@ip-172-31-30-93: /etc/mysql/mariadb.conf.d
  GNU nano 5.4                                    50-server.cnf
# * Basic Settings
#

user                    = mysql
pid-file                = /run/mysqld/mysqld.pid
basedir                 = /usr
datadir                 = /var/lib/mysql
tmpdir                  = /tmp
lc-messages-dir         = /usr/share/mysql
lc-messages             = en_US
skip-external-locking

# Broken reverse DNS slows down connections considerably and name resc
# safe to skip if there are no "host by domain name" access grants
#skip-name-resolve

# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 172.31.30.93
```

$sudo mysql -u root -p

Press enter when asked password

MariaDB [(none)]> CREATE DATABASE userdb;

MariaDB [(none)]>CREATE USER 'admin'@'pvt ip of leader' IDENTIFIED BY 'password';

MariaDB [(none)]>CREATE USER ‘admin’@’pvt ip of node 1’ IDENTIFIED BY ‘password’;
MariaDB [(none)]>CREATE USER ‘admin’@’pvt ip of node 2’ IDENTIFIED BY ‘password’;
MariaDB [(none)]>SHOW GRANTS;
MariaDB [(none)]>GRANT ALL ON *.* TO ‘admin’@’pvt ip of leader’;
MariaDB [(none)]>GRANT ALL ON *.* TO ‘admin’@’pvt ip node 1’;
MariaDB [(none)]>GRANT ALL ON *.* TO ‘admin’@’pvt ip of node 2’;

```
Aborted
admin@ip-172-31-30-93:/etc/mysql/mariadb.conf.d$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 33
Server version: 10.5.19-MariaDB-0+deb11u2 Debian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SHOW GRANTS
    -> ;
+-----------------------------------------------------------------------------------------------------------------------------------+
| Grants for root@localhost                                                                                                         |
+-----------------------------------------------------------------------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO `root`@`localhost` IDENTIFIED VIA mysql_native_password USING 'invalid' OR unix_socket WITH GRANT OPTION |
| GRANT PROXY ON ''@'%' TO 'root'@'localhost' WITH GRANT OPTION                                                                     |
+-----------------------------------------------------------------------------------------------------------------------------------+
```

MariaDB [(none)]>use userdb;
MariaDB [userdb]>SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 1; ***Change this to 1 from 0 because as client will insert data it will be in staging area so it automatically gets commit to final area***
START TRANSACTION;
SET time_zone = "+00:00";
CREATE TABLE `user_table` (
`id` int(11) NOT NULL,
`username` varchar(30) NOT NULL,
`email` varchar(30) NOT NULL,
`password` varchar(30) NOT NULL,
`image` longblob NULL ***change this to null then even if user don’t insert image he will be accepted.***
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
ALTER TABLE `user_table`
ADD PRIMARY KEY (`id`);
ALTER TABLE `user_table`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;
COMMIT;

It is availabel in aveyBD repo .

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";
CREATE TABLE `user_table` (
  `id` int(11) NOT NULL,
  `username` varchar(30) NOT NULL,
  `email` varchar(30) NOT NULL,
  `password` varchar(30) NOT NULL,
  `image` longblob NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
ALTER TABLE `user_table`
  ADD PRIMARY KEY (`id`);
ALTER TABLE `user_table`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;
COMMIT;
```

MariaDB [userdb]>SHOW tables;
MariaDB [userdb]>DESC user_table;

```
Database changed
MariaDB [userdb]> SHOW TABLES
    -> ;
+-------------------+
| Tables_in_userdb  |
+-------------------+
| user_table        |
+-------------------+
1 row in set (0.000 sec)

MariaDB [userdb]> DESC user_table
    -> ;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| id       | int(11)      | NO   | PRI | NULL    | auto_increment |
| username | varchar(30)  | NO   |     | NULL    |                |
| email    | varchar(30)  | NO   |     | NULL    |                |
| password | varchar(30)  | NO   |     | NULL    |                |
| image    | longblob     | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
5 rows in set (0.001 sec)
```

$sudo systemctl restart mariadb-server

STEP 10 : Local machine : Ubuntu

Remove existing index.html and Dockerfile which we created before now as well clone this repo there we already have existing dockerfile and index.html

#mkdir clone *(to clone the repo aveyBD)*
#cd clone
#git clone https://github.com/AveyBD/php-login-registration-using-php-mysql.git
#cp -r php-login-registration-using-php-mysql/* ../projectg/ (*it will copy all contents of dir into our project dir)*
#nano Dockerfile

DB_HOST=ip of database server
DB_USER='DB user'
DB_PASSWORD='rootroot'
DB_NAME='databse name'

```
 GNU nano 6.2                                                    Dockerfile
FROM php:7.4-apache

# Install necessary PHP extensions and dependencies
RUN docker-php-ext-install mysqli pdo pdo_mysql

# Copy your PHP code to the container
COPY . /var/www/html

# Set the appropriate permissions
RUN chown -R www-data:www-data /var/www/html

# Set the database connection details as environment variables #insert databse server ip ,username and passwor
ENV DB_HOST=172.31.30.93
ENV DB_PORT=3306
ENV DB_USER='admin'
ENV DB_PASSWORD='rootroot'
ENV DB_NAME='userdb'

# Expose port 80
EXPOSE 80

# Start Apache web server
CMD ["apache2-foreground"]
```

#cd /server/classes

#nano db.php

Edit : "insert ip of databse server" ,"databse user" , "password"

```
 GNU nano 6.2                                                    db.php
<?php
// connection oriented
class db{
   public function connect(){
       $connectionResource=mysqli_connect("172.31.30.93","admin","rootroot");
       mysqli_select_db($connectionResource,"userdb");
       if(!$connectionResource)
       {
           die("connection failed".mysqli_connect_error());
       }
       else
       echo "Connection Successful";
       return $connectionResource;
   }
   function __constructor(){
```

#git add .

#git commit -a -m '.....'

#git push origin master

Testing 1 : Go to leader / node1/ node2

$sudo apt-get install mariadb-client

$sudo mysql -u admin -h ip of databse server -p rootroot

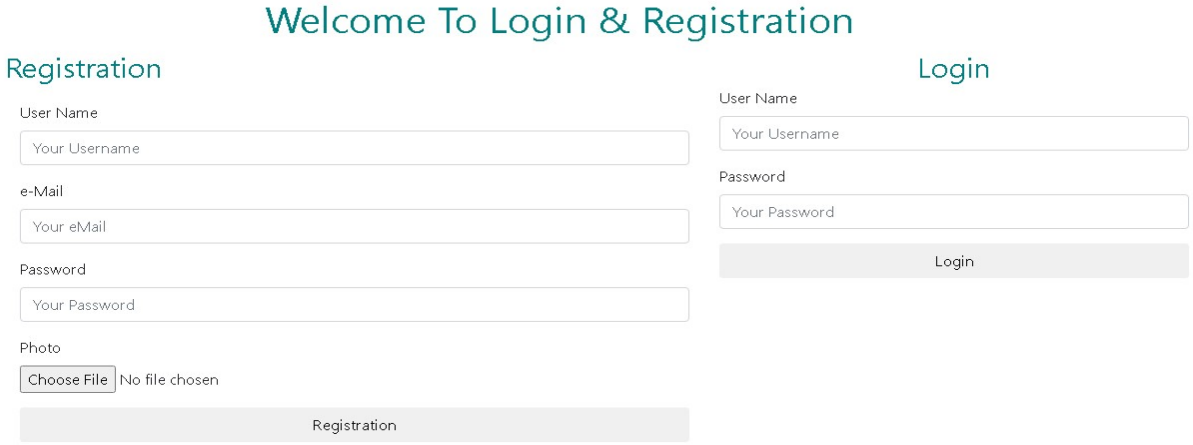
Connection should be successful.

```
admin@ip-172-31-24-151:~$ mysql -u admin -p -h 172.31.30.93
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 36
Server version: 10.5.19-MariaDB-0+deb11u2 Debian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

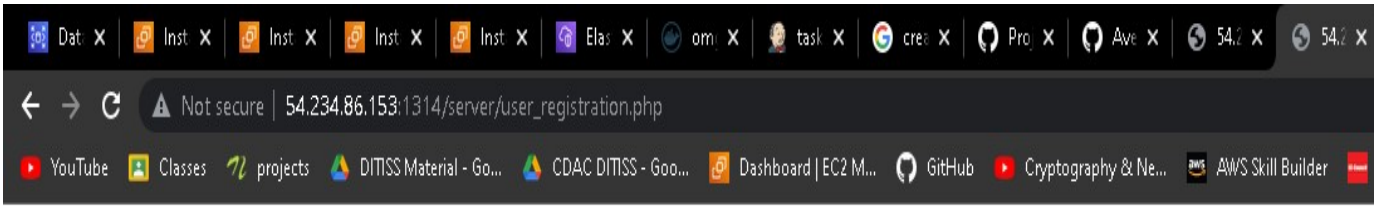
Testing 2: **Go to browser :**

Public ip of load balancer:port /client

http://54.234.86.153:1314/client



Insert data :



Connection SuccessfulInserted Successfully

Databse Check :

Databse server :

MariaDB [userdb]>SELECT * FROM user_table

```
MariaDB [userdb]> SELECT * FROM user_table;
+----+----------+--------------------------+-------------+-------+
| id | username | email                    | password    | image |
+----+----------+--------------------------+-------------+-------+
|  2 | fgsdfg   | dfsadfas                 | asdfsfsa    | NULL  |
|  3 | Ojas     | ojasjawale1010@gmail.com | Ojas1234    | NULL  |
|  4 | komal    | komal@baramati.com       | komal123    | NULL  |
|  5 | omkar    | omkar@gmail.com          | omkar123    | NULL  |
|  6 | kaustubh | kaustubh@gmail.com       | kaustubh123 | NULL  |
|  7 | nitin    | nitin@gmail.com          | nitin123    | NULL  |
|  8 | OMG      | omkargadre1410@gmail.com | Omkargadre  | NULL  |
+----+----------+--------------------------+-------------+-------+
```