

Name - Aman Yadav
Roll No - 171210007 (CSE 3rd year)
Subject - Network programming

Summary -

I have created a google cloud account and created a vm instance

Apart from that I have made a simple blog writing web application using basic web technologies html,css,javascript ,nodejs and mongodb as its database .

so In this report I am going to write about three things

A) How I create vm instance on google cloud and run client sever program over there.

B) how I have deployed **database** of **Blog writing web application** on mongodb cloud server (**ATLAS**).

C) how I have deployed web Application on **HEROKU** cloud web service.

End results -

Publically accessible URL :

<https://fathomless-basin-80101.herokuapp.com/>

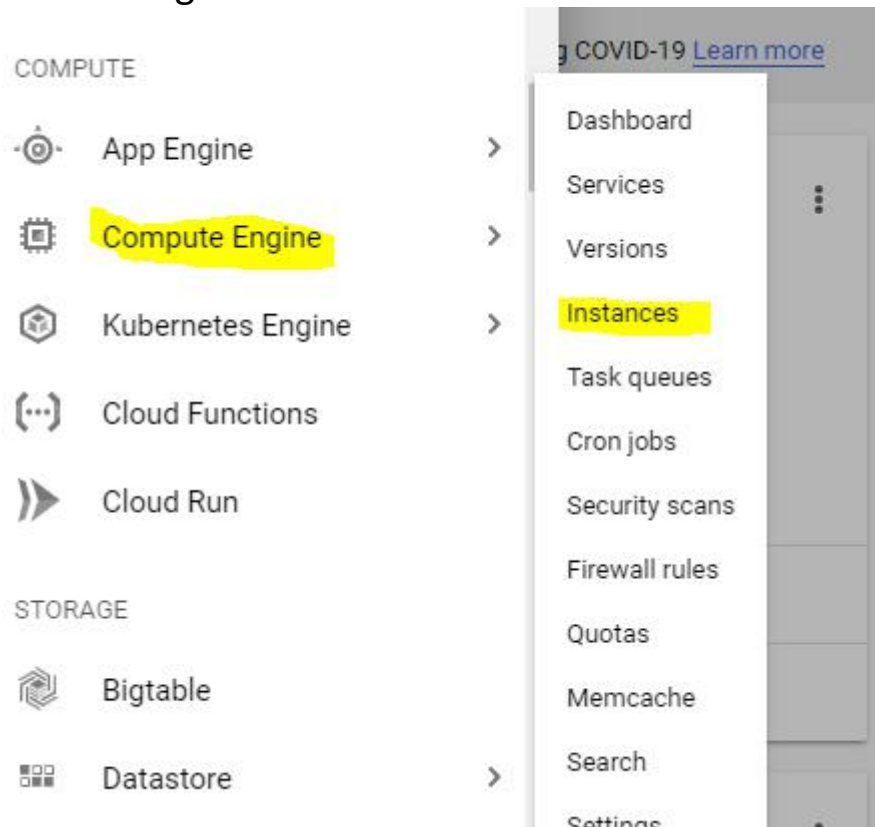
GitHub link for code :

<https://github.com/aman212yadav/assignment/tree/master/assignment3>

Creating VM instance on google cloud and running unix program on it.

Step1-> create google account.

Step2-> click on **Instances** option present in **compute engine** inside Navigation menu.



Step3-> Press create and fill the required details to create vm instance.

VM instances

Compute Engine
VM instances

Compute Engine lets you use virtual machines that run on Google's infrastructure. Create micro-VMs or larger instances running Debian, Windows or other standard images. Create your first VM instance, import it using a migration service or try the quickstart to build a sample app.

Create or Import or Take the quickstart

Create an instance

To create a VM instance, select one of the options:

New VM instance
Create a single VM instance from scratch

New VM instance from template
Create a single VM instance from an existing template

New VM instance from machine image
Create a single VM instance from an existing machine image

Marketplace
Deploy a ready-to-go solution onto a VM instance

You have a draft that wasn't submitted. Click Restore to keep working on it

Name ⓘ
Name is permanent
unix

Labels ⓘ (Optional)
+ Add label

Region ⓘ
Region is permanent
us-central1 (Iowa)

Zone ⓘ
Zone is permanent
us-central1-a

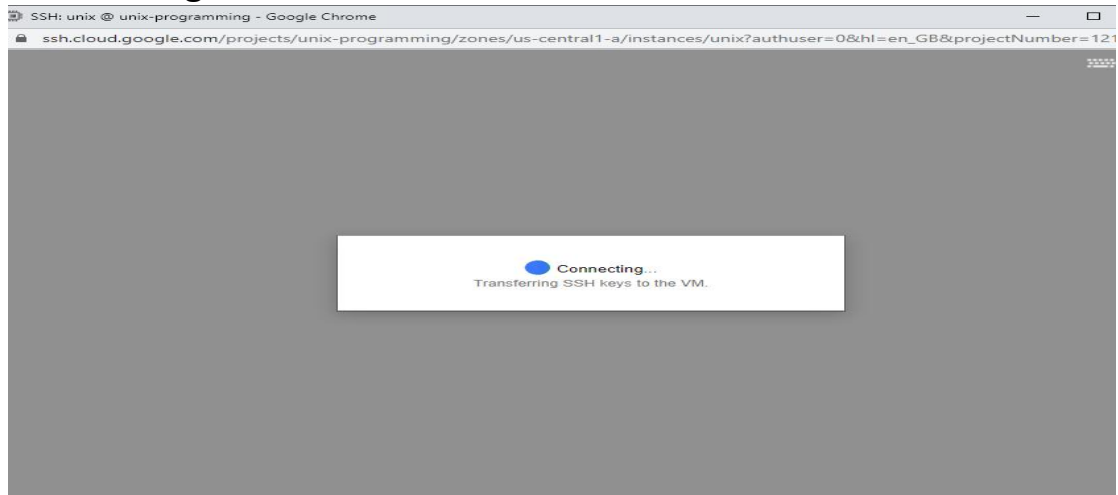
Machine configuration
Machine family
General-purpose Memory-optimised Compute-optimised
Machine types for common workloads, optimised for cost and flexibility
Series
N1
Powered by Intel Skylake CPU platform or one of its predecessors

You have ₹12,036.653977 free trial credits remaining
\$24.67 monthly estimate
That's about \$0.034 hourly
Pay for what you use: No upfront costs and per second billing
Details

VM instance is ready

Filter VM instances							
<input type="checkbox"/>	Name ^	Zone	Recommendation	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>	unix	us-central1-a			10.128.0.2 (nic0)	35.188.129.139 ↗	SSH ⌵ ⋮

Connecting to VM



Running VM

```
ssh.cloud.google.com/projects/unix-programming/zones/us-central1-a/instances/unix?authuser=0&hl=en_GB&projectNumber=121
Connected, host fingerprint: ssh-rsa 0 5E:8D:36:ED:31:EC:99:8A:B7:CF:43:3A:B4:C2
:88:B1:BE:DB:CD:ED:94:77:D9:9F:87:A8:DD:A5:22:B0:B1:95
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-1061-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

aman212yadav@unix:~$ ls
aman212yadav@unix:~$ pwd
/home/aman212yadav
aman212yadav@unix:~$ cd ..
aman212yadav@unix:/home$
```

Step4-> writing and executing hello world program on google cloud.

```

client.c helloWorld.c server.c
aman212yadav@unix:~$ g++ helloWorld.c
aman212yadav@unix:~$ ./a.out
hello worldaman212yadav@unix:~$ █

```

Step5-> writing and executing **server.c** file on google cloud.

```

hello worldaman212yadav@unix:~$ g++ server.c
server.c: In function 'int main()':
server.c:36:9: warning: 'char* gets(char*)' is deprecated [-Wdeprecated-declarations]
     gets(data);
     ^
In file included from server.c:2:0:
/usr/include/stdio.h:638:14: note: declared here
extern char *gets (char *__s) __wur __attribute_deprecated__;
               ^
server.c:36:9: warning: 'char* gets(char*)' is deprecated [-Wdeprecated-declarations]
     gets(data);
     ^
In file included from server.c:2:0:
/usr/include/stdio.h:638:14: note: declared here
extern char *gets (char *__s) __wur __attribute_deprecated__;
               ^
server.c:36:18: warning: 'char* gets(char*)' is deprecated [-Wdeprecated-declarations]
     gets(data);
     ^
In file included from server.c:2:0:
/usr/include/stdio.h:638:14: note: declared here
extern char *gets (char *__s) __wur __attribute_deprecated__;
               ^
/tmp/ccCKALdV.o: In function 'main':
server.c:(.text+0x148): warning: the 'gets' function is dangerous and should not be used.
aman212yadav@unix:~$ ./a.out
server is Running.
█

```

Step6->Opening separate vm terminal and running **client.c** file

```

aman212yadav@unix:~$ g++ client.c
client.c: In function 'int main()':
client.c:36:5: warning: 'char* gets(char*)' is deprecated [-Wdeprecated-declarations]
     gets(data);
     ^
In file included from client.c:5:0:
/usr/include/stdio.h:638:14: note: declared here
extern char *gets (char *__s) __wur __attribute_deprecated__;
               ^
client.c:36:5: warning: 'char* gets(char*)' is deprecated [-Wdeprecated-declarations]
     gets(data);
     ^
In file included from client.c:5:0:
/usr/include/stdio.h:638:14: note: declared here
extern char *gets (char *__s) __wur __attribute_deprecated__;
               ^
client.c:36:14: warning: 'char* gets(char*)' is deprecated [-Wdeprecated-declarations]
     gets(data);
     ^
In file included from client.c:5:0:
/usr/include/stdio.h:638:14: note: declared here
extern char *gets (char *__s) __wur __attribute_deprecated__;
               ^
/tmp/ccVNUoxD.o: In function 'main':
client.c:(.text+0x1fd): warning: the 'gets' function is dangerous and should not be used.

```

Connection between server and client is established and Message is exchanged successfully .

client

```
aman212yadav@unix:~$ ./a.out
client is running : Enter ur message for server-> this is client from cloud
message from server : hello client this is server from cloud
```

Server

```
aman212yadav@unix:~$ ./a.out
server is Running.
server : one client sent me a message and the message is -> this is client from cloud
server : enter response message for client -> hello client this is server from cloud
server: response message sent to the client
```

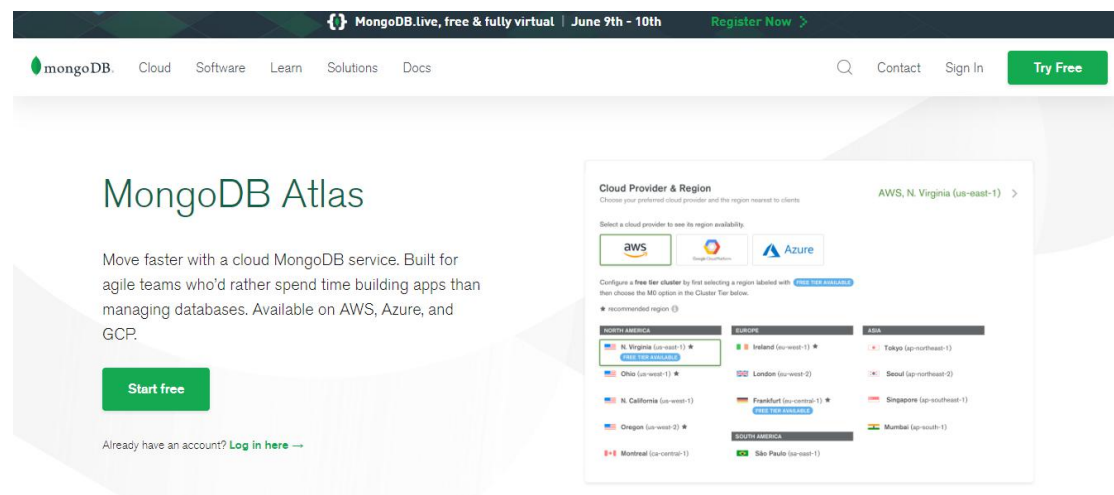
I have attached all the related code on my github account.

Link ->

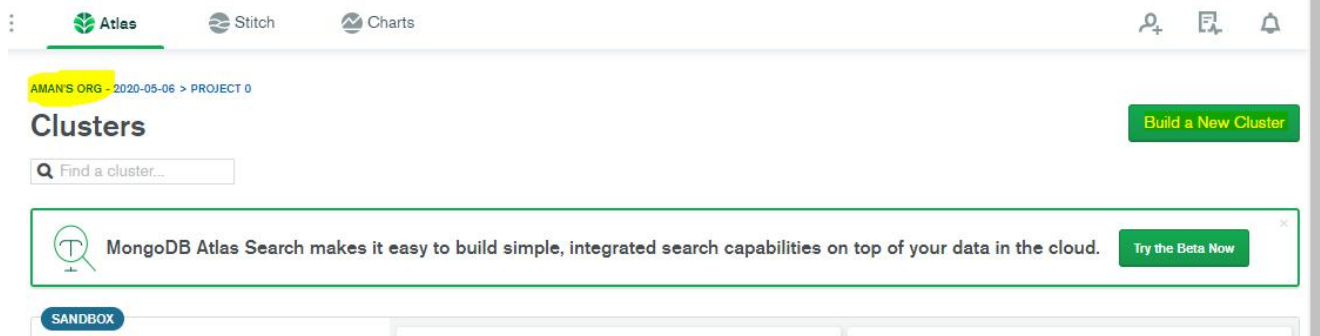
<https://github.com/aman212yadav/assignment/tree/master/assignment3>

Deploying database of the web application on Mongo Db cloud server (ATLAS).

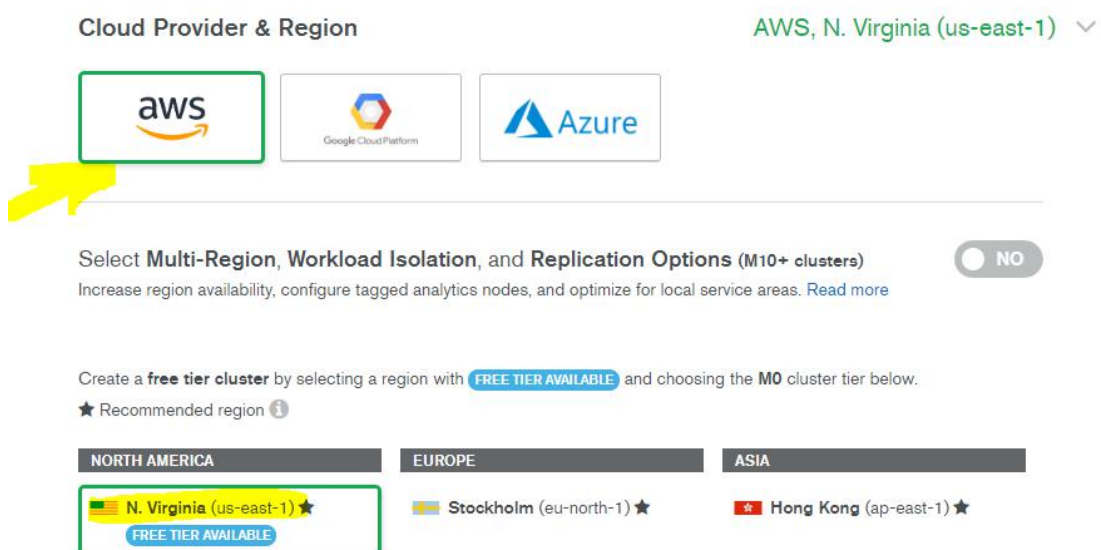
Step 1-> creating a mongoDB atlas account.



Step 2-> creating a new cluster



Step 3-> selecting cloud provider.



Step 4-> Getting access to database by providing credential.

Edit User: **admin-aman@admin**

Password Authentication

MongoDB uses **SCRAM** as its default authentication method.

admin-aman

.....

SHOW

Autogenerate Secure Password

Copy

Database User Privileges

Atlas admin

Read and write to
any database

Only read any
database

Select Custom
Role

Add Default Privileges

Cancel

Update User

Step 5-> Making it accessible from Anywhere by setting a global IP address (0.0.0.0) .

AMANS ORG - 2020-05-06 > PROJECT 0

Network Access

IP Whitelist

Peering

Private Endpoint



Whitelist an IP address

Configure which IP addresses can access your cluster.

Add IP Address

Add IP Whitelist Entry

Atlas only allows client connections to a cluster from entries in the project's whitelist. Each entry should either be a single IP address or a CIDR-notated range of addresses. [Learn more.](#)

ADD CURRENT IP ADDRESS

ALLOW ACCESS FROM ANYWHERE

Whitelist Entry:

0.0.0.0/0

Comment:

Optional comment describing this entry



This entry is temporary and will be deleted in

6 hours

Cancel

Confirm

Step 6-> Generating secure link to connect web application to the hosted database.

Connect to Cluster0

✓ Setup connection security

Choose a connection method

Connect

Choose a connection method [View documentation](#)

Get your pre-formatted connection string by selecting your tool below.



Connect with the mongo shell

Interact with your cluster using MongoDB's interactive Javascript interface



Connect your application

Connect your application to your cluster using MongoDB's native drivers



Connect using MongoDB Compass

Explore, modify, and visualize your data with MongoDB's GUI



Go Back

Close

Connect to Cluster0

✓ Setup connection security

✓ Choose a connection method

Connect

1

Select your driver and version

DRIVER

Node.js

VERSION

3.0 or later

2

Add your connection string into your application code

Connection String Only

Full Driver Example

mongodb+srv://admin-aman:<password>@cluster0-ydwhd.mongodb.net/test?r

Copy

Replace **<password>** with the password for the user, **admin-aman**, and ensure all special characters are [URL encoded](#).

Having trouble connecting? [View our troubleshooting documentation](#)

Step 7->Adding the generated link to the web application

```
25 mongoose.connect('mongodb+srv://admin-aman:<password>@cluster0-ydwhd.mongodb.net/blogDB');
```

(hiding some information for security reasons)

Step 8->Web application is now able to communicate with database hosted on cloud (ATLAS).

DAILY JOURNAL

HOME

COMPOSE

ABOUT US

CONTACT US

Home

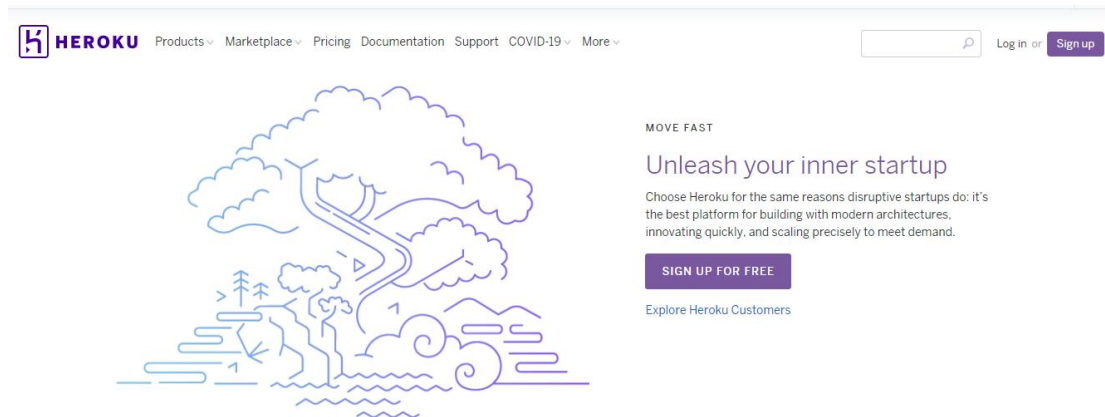
Lacus vel facilisis volutpat est velit egestas dui id ornare. Semper auctor neque vitae tempus quam. Sit amet cursus sit amet dictum sit amet justo. Viverra tellus in hac habitasse. Imperdiet proin fermentum leo vel orci porta. Donec ultrices tincidunt arcu non sodales neque sodales ut. Mattis molestie a iaculis at erat pellentesque adipiscing. Magnis dis parturient montes nascetur ridiculus mus mauris vitae ultricies. Adipiscing elit ut aliquam purus sit amet luctus venenatis lectus. Ultrices vitae auctor eu augue ut lectus arcu bibendum at. Odio euismod lacinia at quis risus sed vulputate odio ut. Cursus mattis molestie a iaculis at erat pellentesque adipiscing.

hosted database to mongo db atlas

check this report to know how I have hosted this web Application database on Mongo db cloud server ... [Read More](#)

Deploying the Blog writing web application on HEROKU cloud service.


Step 1->Creating a Heroku account.



Step 2->Installing Heroku CLI (command line interface).


In this step you'll install the Heroku Command Line Interface (CLI). You use the CLI to manage and scale your applications, provision add-ons, view your application logs, and run your application locally.

Download and run the installer for your platform:

 **macOS**
[Download the installer](#)

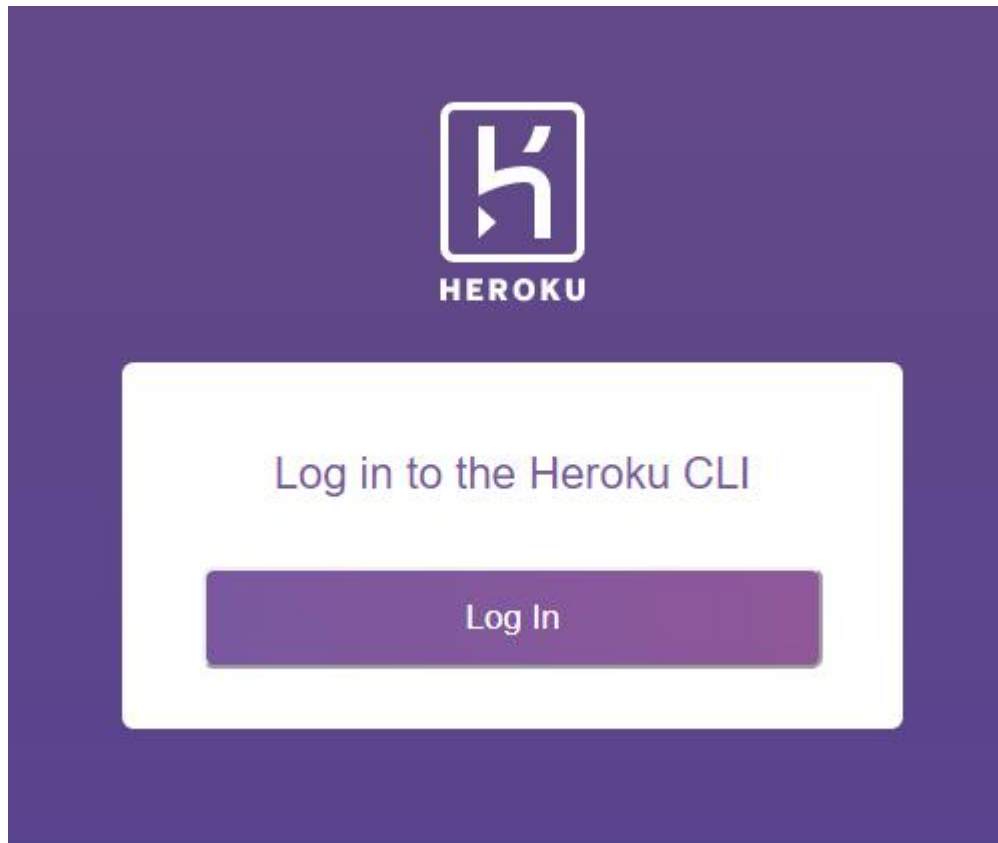
Also available via Homebrew:

```
$ brew install heroku/brew/heroku
```

 **Windows**
Download the appropriate installer for your Windows installation:
[64-bit installer](#)
[32-bit installer](#)

Step 3->Login to heroku by using **heroku login** command.

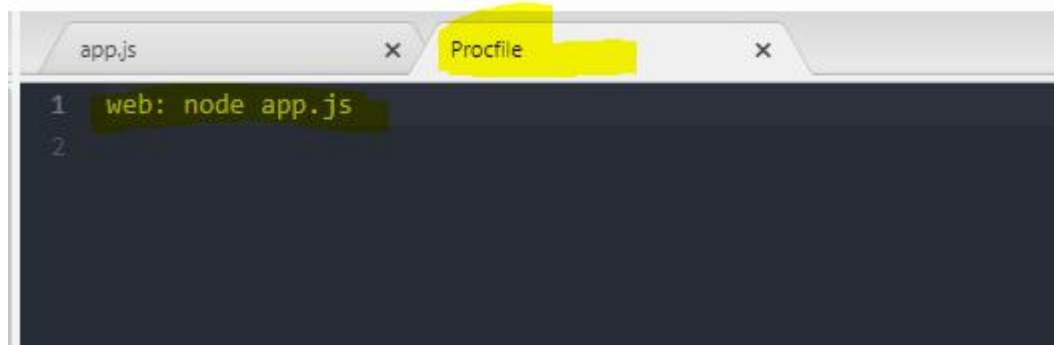
```
C:\Users\aman2>heroku login
» Warning: heroku update available from 7.35.1 to 7.40.0.
heroku: Press any key to open up the browser to login or q to exit:
```



Step 4->Creating an app on heroku which will prepare heroku to receive our source code.

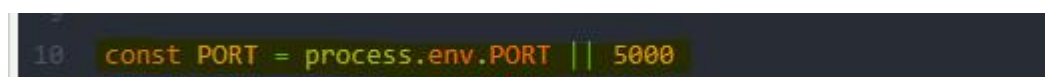
```
C:\Users\aman2>heroku create
» Warning: heroku update available from 7.35.1 to 7.40.0.
Creating app... done, ⬢ peaceful-wildwood-41117
https://peaceful-wildwood-41117.herokuapp.com/ | https://git.heroku.com/peaceful-wildwood-41117.git
```

Step 5->Defining a **procfile** to explicitly declare what command heroku should use to start the app.

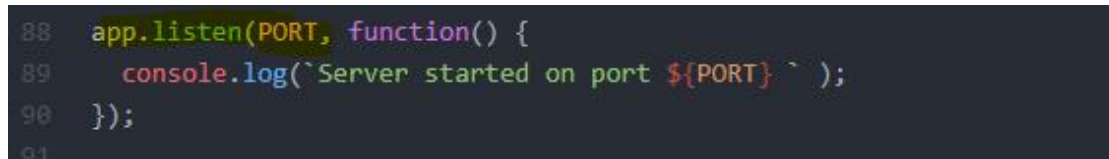


```
1 web: node app.js
2
```

Step 6 -> Defining **port** on which app should listen/start



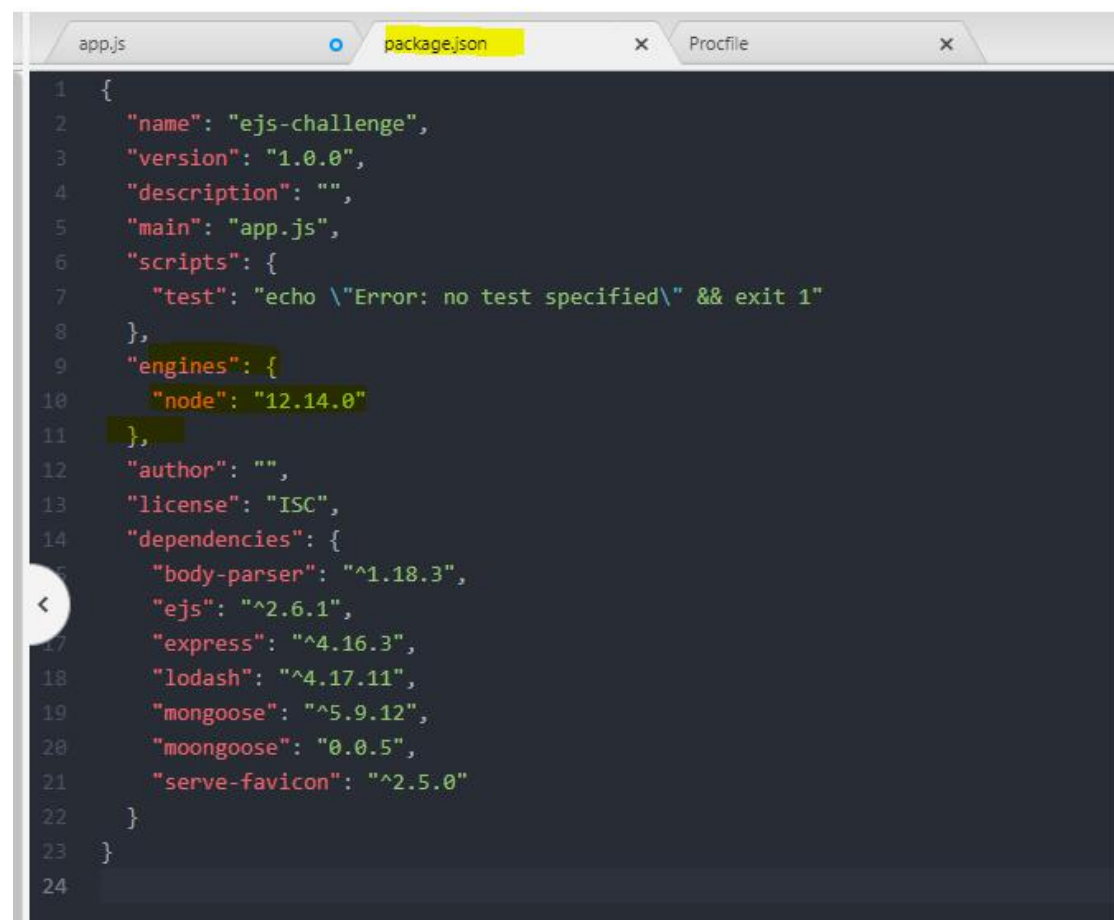
```
10 const PORT = process.env.PORT || 5000
```



```
88 app.listen(PORT, function() {
89   console.log(`Server started on port ${PORT} `);
90 });
91
```

Step 7 -> Finding version of node and adding it **package.json** file

```
C:\Users\aman2>node --version  
v12.14.0
```



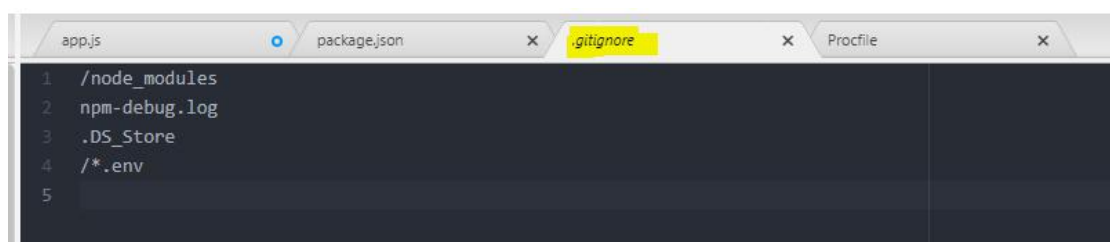
The screenshot shows a code editor with three tabs: 'app.js', 'package.json' (which is active and highlighted in yellow), and 'Procfile'. The 'package.json' file contains the following JSON structure:

```
1 {  
2   "name": "ejs-challenge",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "app.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1"  
8   },  
9   "engines": {  
10    "node": "12.14.0"  
11  },  
12  "author": "",  
13  "license": "ISC",  
14  "dependencies": {  
15    "body-parser": "^1.18.3",  
16    "ejs": "^2.6.1",  
17    "express": "^4.16.3",  
18    "lodash": "^4.17.11",  
19    "mongoose": "^5.9.12",  
20    "moongoose": "0.0.5",  
21    "serve-favicon": "^2.5.0"  
22  }  
23 }  
24
```


Step 7 ->Initialising git repository .

```
rcise\Blog-with-Database-Starting-Files>git init
Initialized empty Git repository in C:/Users/aman2
```

Step 8 ->Adding **.gitignore** file to avoid uploading unnecessary files/folder.



The screenshot shows a code editor with four tabs: 'app.js', 'package.json', '.gitignore', and 'Procfile'. The '.gitignore' tab is active and displays the following content:

```
1 /node_modules
2 npm-debug.log
3 .DS_Store
4 /*.env
5
```

Step 9 ->Adding all files to **staging area** and **committing** all changes.

```
C:\Users\aman2\Desktop\Blog-with-Database-Starting-Files>git add
warning: LF will be replaced by CRLF in app.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in package-lock.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in package.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in public/css/styles.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in views/about.ejs.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in views/compose.ejs.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in views/contact.ejs.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in views/home.ejs.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in views/partials/footer.ejs.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in views/partials/header.ejs.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in views/post.ejs.
The file will have its original line endings in your working directory
```

```
rcise\Blog-with-Database-Starting-Files>git commit -am "commit all changes"
[master (root-commit) 187b135] commit all changes
14 files changed, 1034 insertions(+)
create mode 100644 .gitignore
create mode 100644 Procfile
create mode 100644 app.js
create mode 100644 package-lock.json
create mode 100644 package.json
create mode 100644 public/css/styles.css
create mode 100644 public/favicon.ico
create mode 100644 views/about.ejs
create mode 100644 views/compose.ejs
create mode 100644 views/contact.ejs
create mode 100644 views/home.ejs
create mode 100644 views/partials/footer.ejs
create mode 100644 views/partials/header.ejs
create mode 100644 views/post.ejs
```

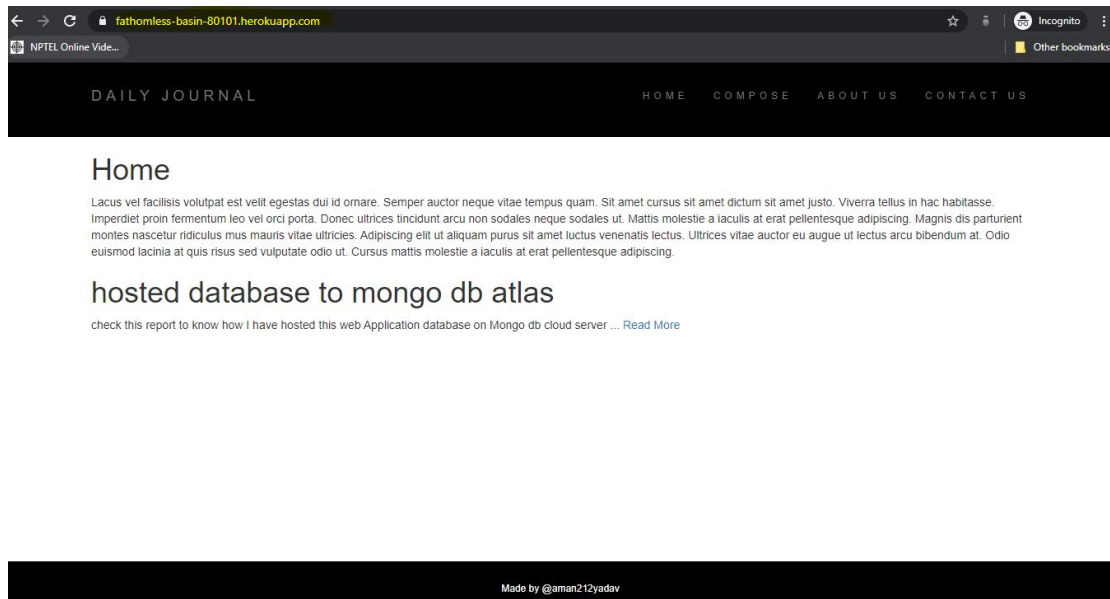
Step 10 ->Pushing all changes to remote repository heroku master.

```
rcise\Blog-with-Database-Starting-Files>git push heroku master
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 4 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (20/20), 16.64 KiB | 3.33 MiB/s, done.
Total 20 (delta 0), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Node.js app detected
remote:
remote: -----> Creating runtime environment
remote:
remote:       NPM_CONFIG_LOGLEVEL=error
remote:       NODE_ENV=production
remote:       NODE_MODULES_CACHE=true
remote:       NODE_VERBOSE=false
remote:
remote: -----> Installing binaries
remote:       engines.node (package.json): 12.14.0
remote:       engines.npm (package.json):  unspecified (use default)
remote:
remote:       Resolving node version 12.14.0...
remote:       Downloading and installing node 12.14.0...
remote:       Using default npm version: 6.13.4
remote:
remote: -----> Installing dependencies
remote:       Installing node modules
```

Step 11 ->Application deployed successfully on cloud .

```
remote: -----> Caching build
remote:       - node_modules
remote:
remote: -----> Pruning devDependencies
remote:       audited 219 packages in 1.178s
remote:
remote:       1 package is looking for funding
remote:       run `npm fund` for details
remote:
remote:       found 2 high severity vulnerabilities
remote:       run `npm audit fix` to fix them, or `npm audit` for details
remote: -----> Build succeeded!
remote: -----> Discovering process types
remote:       Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:       Done: 25.4M
remote: -----> Launching...
remote:       Released v3
remote:       https://fathomless-basin-80101.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/fathomless-basin-80101.git
 * [new branch]      master -> master
```

Got a publically accessible URL.



Conclusion -> both database and web application hosted successfully on the cloud web services (ATLAS, Heroku).

URL-> <https://fathomless-basin-80101.herokuapp.com/>