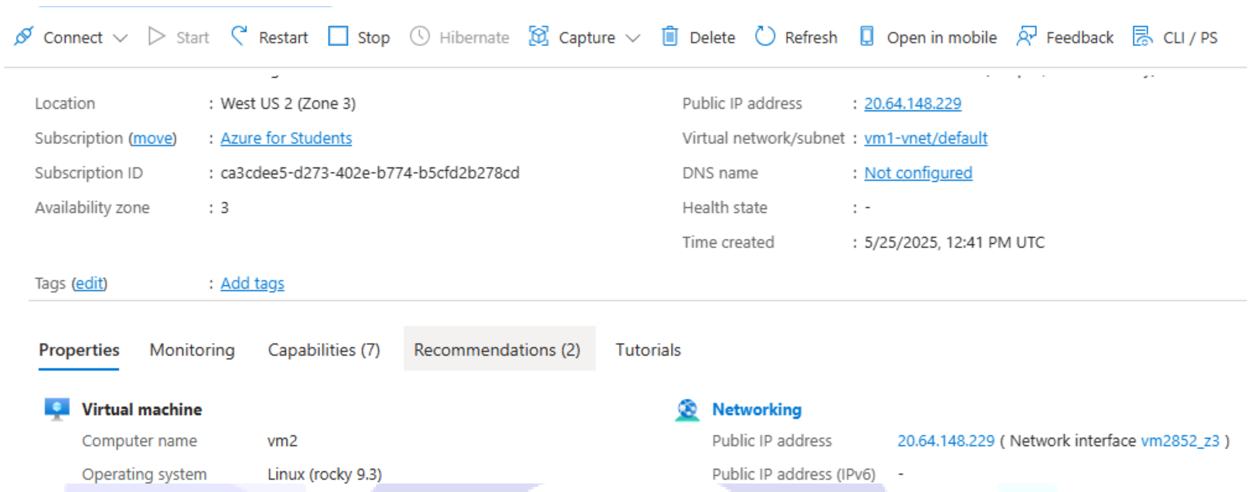
Hosting MERN on server – (Project BitBucket)

Plan

1. Create a Rocky Linux VM (or any RHEL based)

In this example, I have used Rocky Linux 9.3 machine.



Location	: West US 2 (Zone 3)	Public IP address	: 20.64.148.229
Subscription (move)	: Azure for Students	Virtual network/subnet	: vm1-vnet/default
Subscription ID	: ca3cdee5-d273-402e-b774-b5cfd2b278cd	DNS name	: Not configured
Availability zone	: 3	Health state	: -
		Time created	: 5/25/2025, 12:41 PM UTC
Tags (edit)	: Add tags		

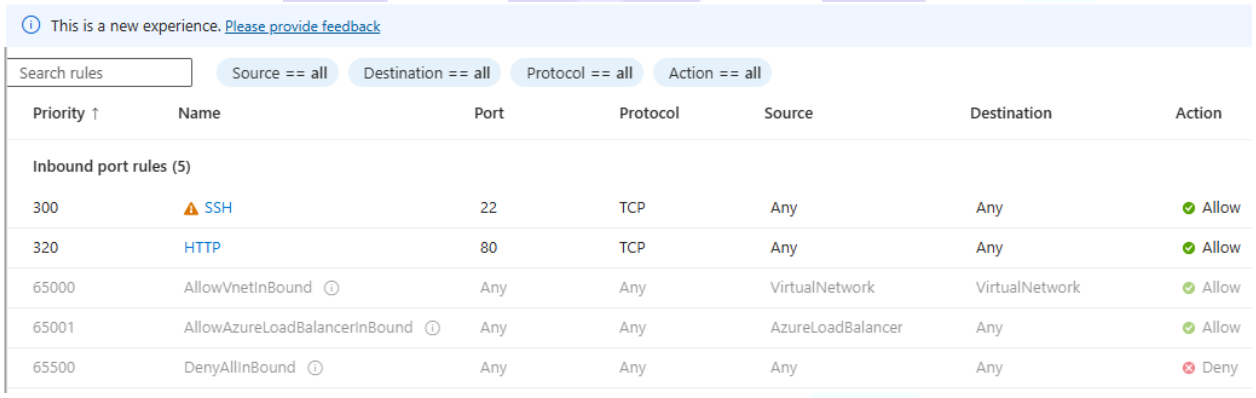
Virtual machine

Computer name	vm2
Operating system	Linux (rocky 9.3)

Networking

Public IP address	20.64.148.229 (Network interface vm2852_z3)
Public IP address (IPv6)	-

2. Open ports 22 and 80 (SSH & HTTP)



This is a new experience. Please provide feedback

Priority ↑	Name	Port	Protocol	Source	Destination	Action
Inbound port rules (5)						
300	SSH	22	TCP	Any	Any	Allow
320	HTTP	80	TCP	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

3. Install Nodejs, nginx, nano editor, pm2

- Install nginx

```
sudo dnf update && sudo dnf install nginx -y
```

- Install nodejs

```
curl -o- https://raw.githubusercontent.com/nvm-
sh/nvm/v0.40.2/install.sh | bash
\. "$HOME/.nvm/nvm.sh"
nvm install 22
```

- Install nano editor

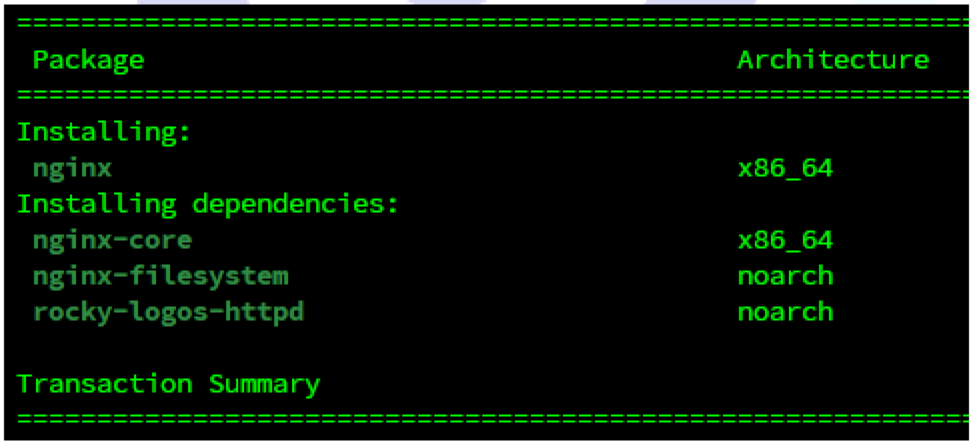
```
sudo dnf install nano -y
```

- Install pm2

```
npm install pm2 -g
```

- Start and enable nginx

```
sudo systemctl start nginx
sudo systemctl enable nginx
```



```
==============================================================
 Package                                            Architecture
==============================================================
Installing:
 nginx                                              x86_64
Installing dependencies:
 nginx-core                                         x86_64
 nginx-filesystem                                   noarch
 rocky-logos-httpd                                  noarch

Transaction Summary
==============================================================
```

4. **Check the nginx is working.**

```
nginx -v
```

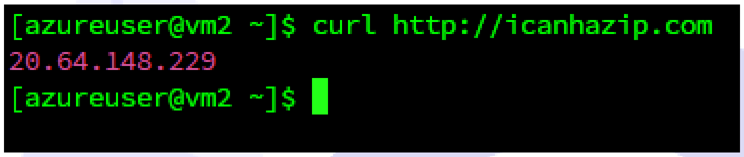
Enter IP address of the server in the web browser.

http://<server_ip>

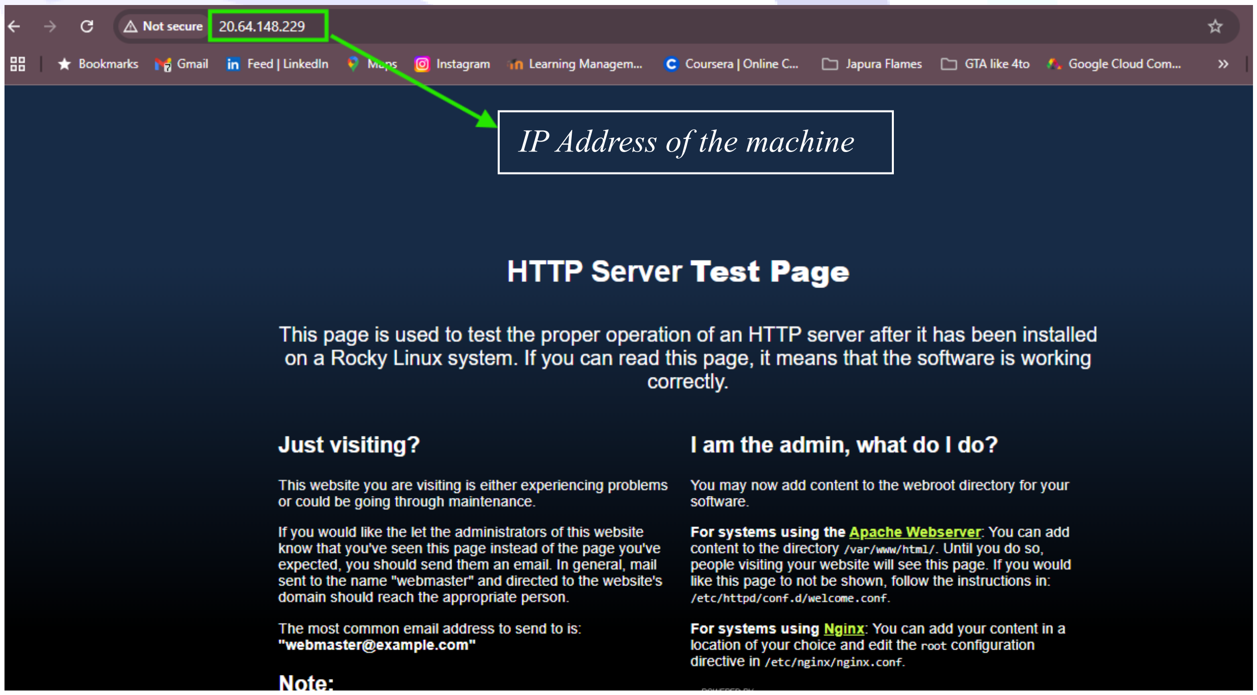
** Default nginx web page will be shown **

Check the public IP address of the server using following command.

```
curl http://icanhazip.com
```



```
[azureuser@vm2 ~]$ curl http://icanhazip.com
20.64.148.229
[azureuser@vm2 ~]$
```



HTTP Server Test Page

This page is used to test the proper operation of an HTTP server after it has been installed on a Rocky Linux system. If you can read this page, it means that the software is working correctly.

Just visiting?

This website you are visiting is either experiencing problems or could be going through maintenance.

If you would like the let the administrators of this website know that you've seen this page instead of the page you've expected, you should send them an email. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

The most common email address to send to is: **"webmaster@example.com"**

Note:

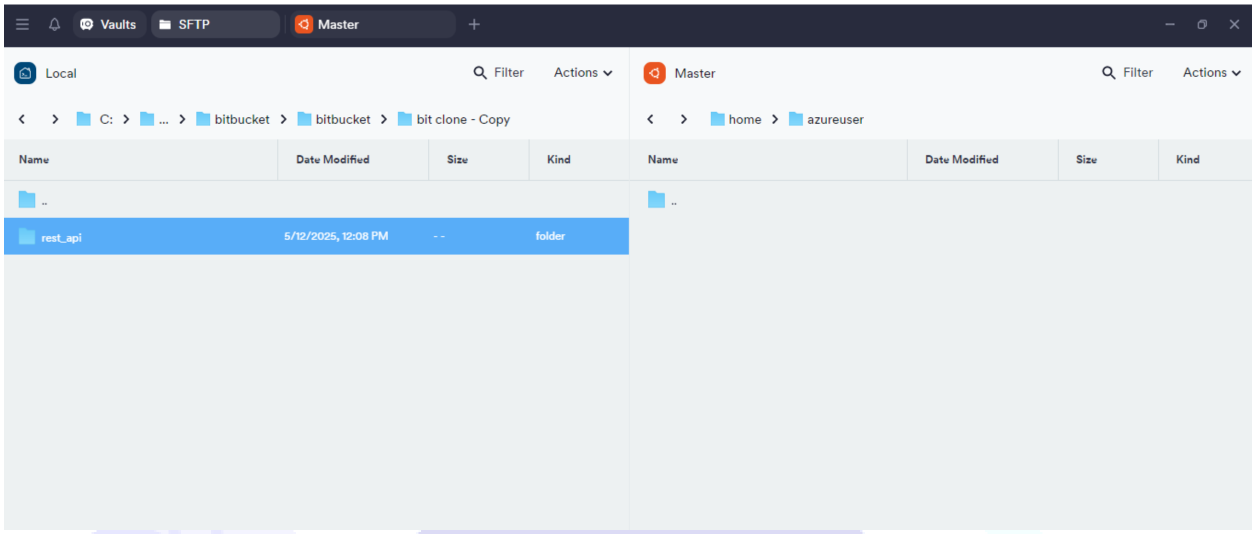
I am the admin, what do I do?

You may now add content to the webroot directory for your software.

For systems using the Apache Webserver: You can add content to the directory `/var/www/html/`. Until you do so, people visiting your website will see this page. If you would like this page to not be shown, follow the instructions in: `/etc/httpd/conf.d/welcome.conf`.

For systems using Nginx: You can add your content in a location of your choice and edit the `root` configuration directive in `/etc/nginx/nginx.conf`.

5. Transfer the project to the VM (via SFTP using Termius)

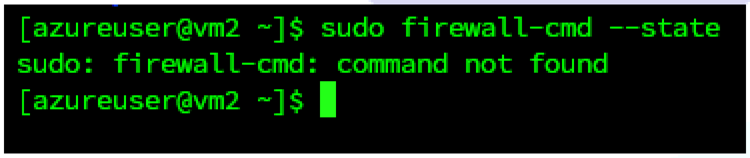


Drag and drop the project folder to the target machine (in the Right side).

6. Check for any firewalls are active

```
sudo firewall-cmd --state
```

If firewall-cmd is not installed, following message will be appeared.



```
[azureuser@vm2 ~]$ sudo firewall-cmd --state
sudo: firewall-cmd: command not found
[azureuser@vm2 ~]$
```

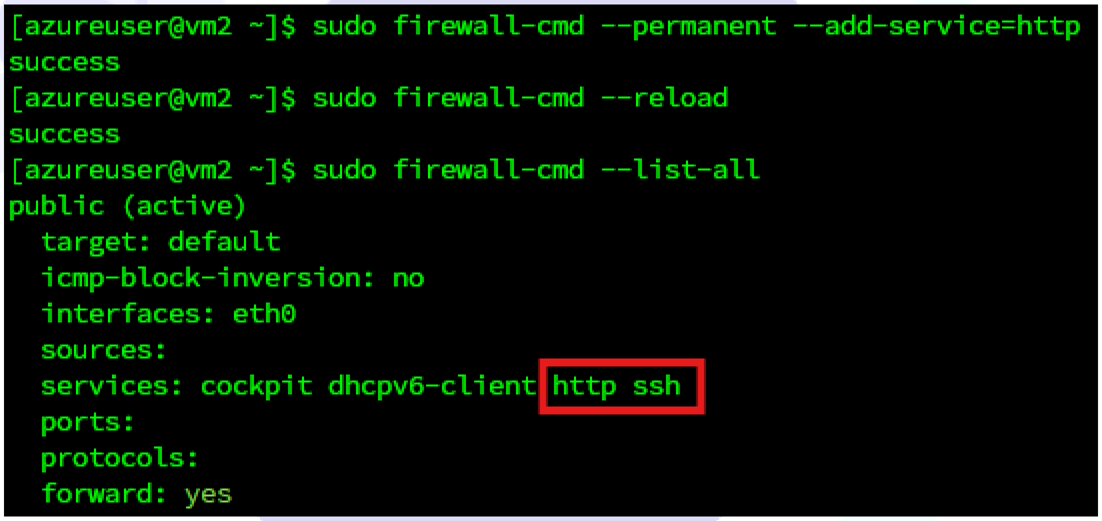
7. Install firewall and add rules

- Install and activate firewall-cmd

```
sudo dnf install firewalld -y
sudo systemctl enable firewalld
sudo systemctl start firewalld
```

- Add rules to open ports

```
sudo firewall-cmd --permanent --add-service=ssh
sudo firewall-cmd --permanent --add-service=http
```



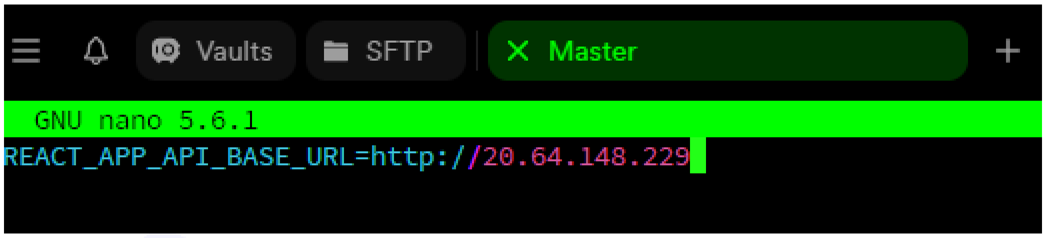
```
[azureuser@vm2 ~]$ sudo firewall-cmd --permanent --add-service=http
success
[azureuser@vm2 ~]$ sudo firewall-cmd --reload
success
[azureuser@vm2 ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0
  sources:
  services: cockpit dhcpv6-client http ssh
  ports:
  protocols:
  forward: yes
```

Learn more about firewall-cmd

👉 Firewall-cmd

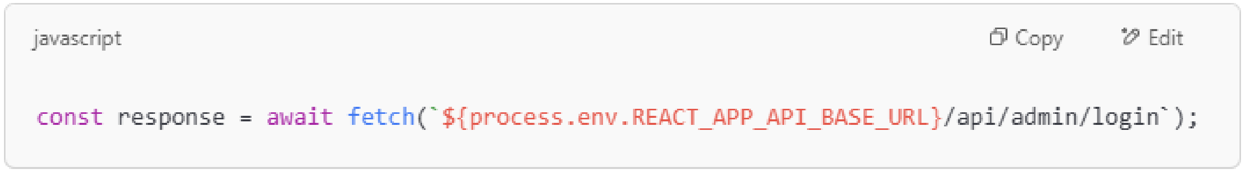
8. Change the .env file in frontend

Change the BASE_URL in the .env file to IP address of the machine. (Do not enter any port number. Just the IP address only.)



```
  GNU nano 5.6.1
REACT_APP_API_BASE_URL=http://20.64.148.229
```

9. Note down the API prefix to configure nginx for backend API calls



```javascript
const response = await fetch(`${process.env.REACT_APP_API_BASE_URL}/api/admin/login`);
```

From this, you know the prefix is `/api/`.

10. Install dependencies and create a build folder

```
npm install
npm run build
```

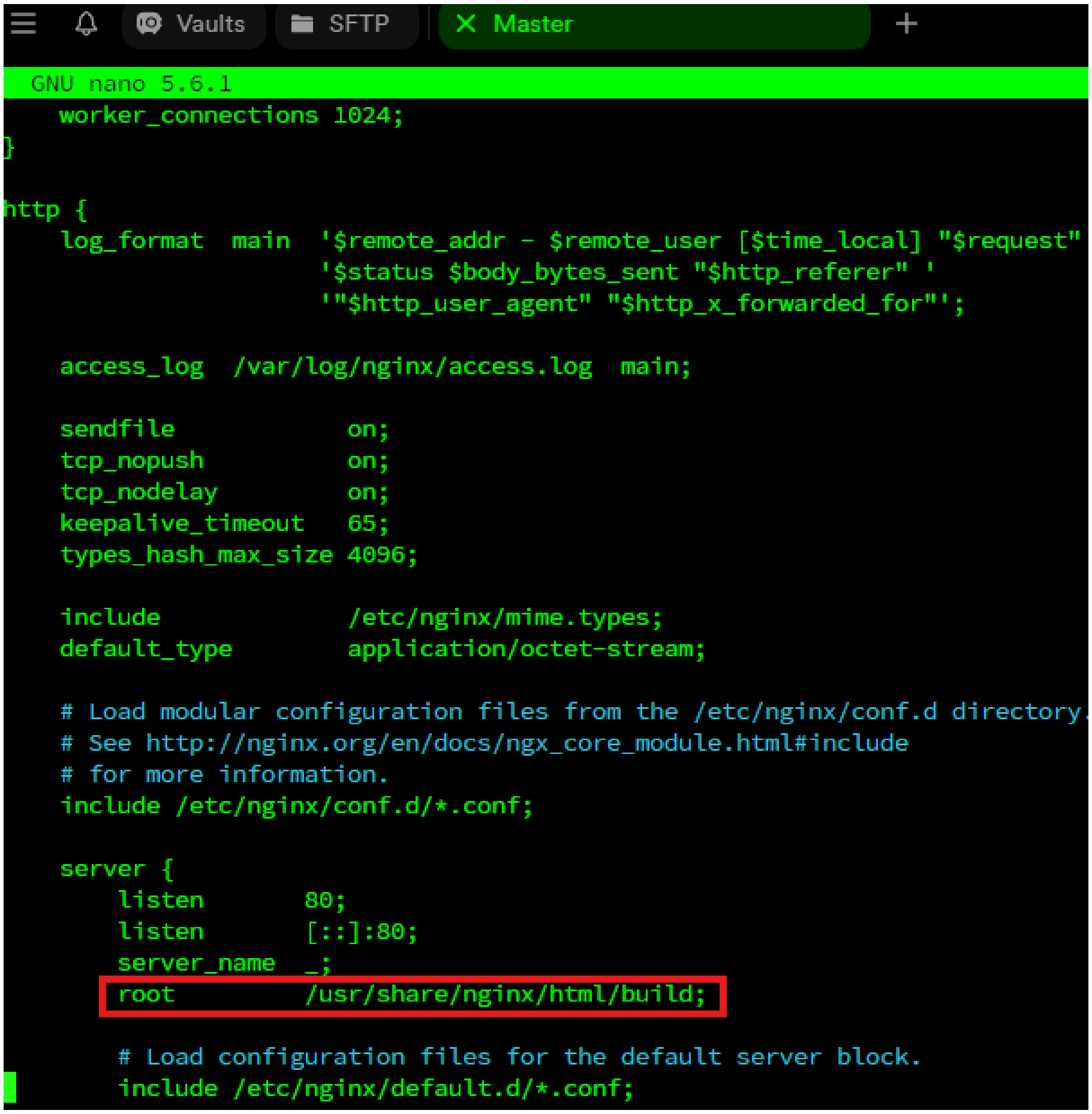
11. Copy the build folder to the nginx directory

```
sudo cp -r build /usr/share/nginx/html
```

12. Change the root directory of nginx that matches the location of the new build folder

Go to nginx configuration file and edit it.

```
sudo nano /etc/nginx/nginx.conf
```



```
 GNU nano 5.6.1
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 4096;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;

    server {
        listen       80;
        listen       [::]:80;
        server_name  _;
        root         /usr/share/nginx/html/build;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;
```

After changing the configuration file, reload nginx to apply changes.

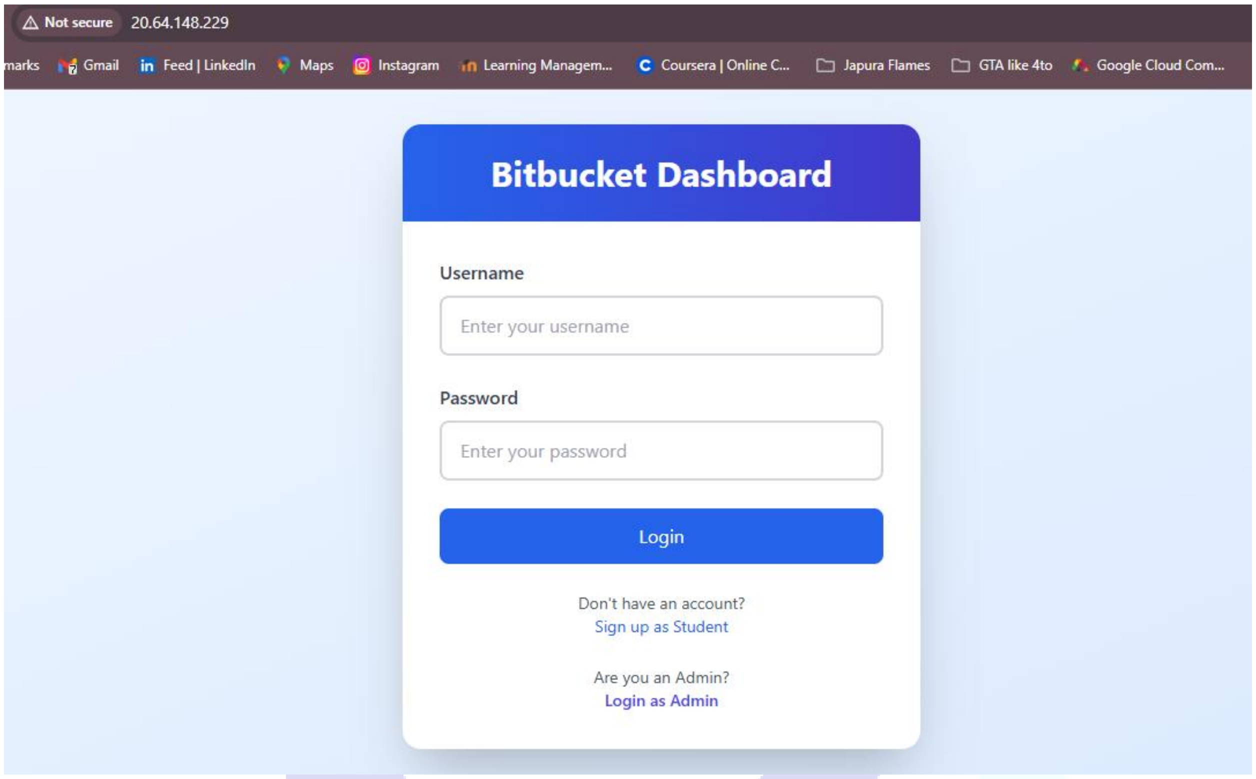
```
sudo nginx -s reload
```

As a best practice, it is advisable to test the syntax errors in nginx.conf file using following command. `sudo nginx -t`

13. Check the frontend working correctly

Enter the public ip address of the machine in the browser and check whether the frontend is accessible.

http://<server_ip>



14. Go to backend and install dependencies

```
npm install
```

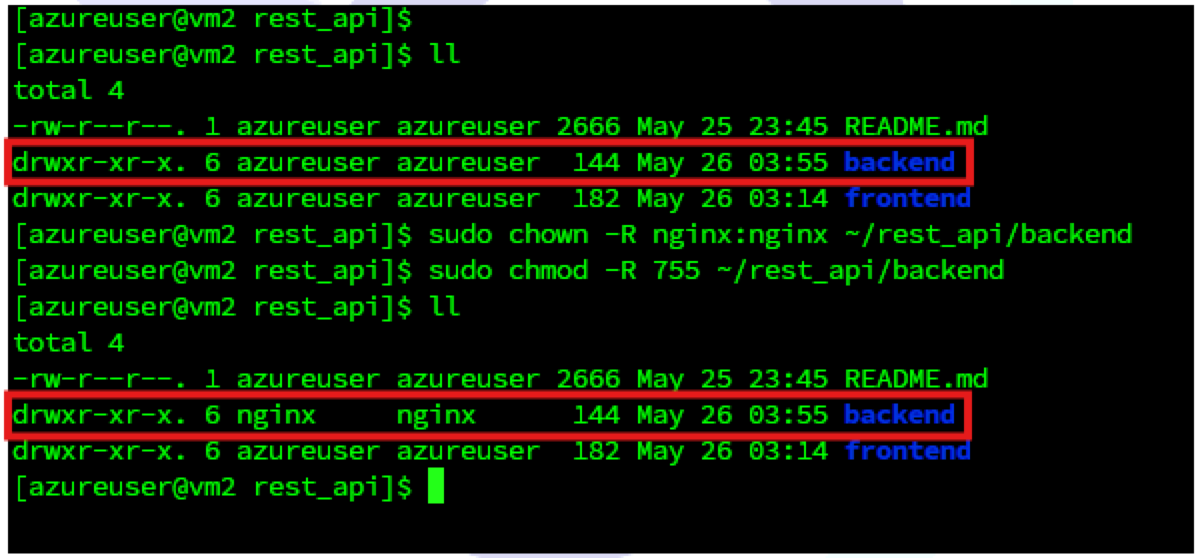
15. Give the permission to nginx to the backend directory

- Check the permissions

```
ll
```

- Give the permission to nginx to backend directory. (files inside backend inherit them)

```
sudo chown -R nginx:nginx ~/path/to/backend
sudo chmod -R 755 ~/path/to/backend
```



```
[azureuser@vm2 rest_api]$
[azureuser@vm2 rest_api]$ ll
total 4
-rw-r--r--. 1 azureuser azureuser 2666 May 25 23:45 README.md
drwxr-xr-x. 6 azureuser azureuser  144 May 26 03:55 backend
drwxr-xr-x. 6 azureuser azureuser  182 May 26 03:14 frontend
[azureuser@vm2 rest_api]$ sudo chown -R nginx:nginx ~/rest_api/backend
[azureuser@vm2 rest_api]$ sudo chmod -R 755 ~/rest_api/backend
[azureuser@vm2 rest_api]$ ll
total 4
-rw-r--r--. 1 azureuser azureuser 2666 May 25 23:45 README.md
drwxr-xr-x. 6 nginx     nginx      144 May 26 03:55 backend
drwxr-xr-x. 6 azureuser azureuser  182 May 26 03:14 frontend
[azureuser@vm2 rest_api]$
```

16. Change the nginx configuration file to set as a reverse proxy, and reload nginx

Before configuring nginx, we need to check backend files to write the server block correctly.

- Look at common entry points like server.js or app.js.
  If the backend defines routes like:

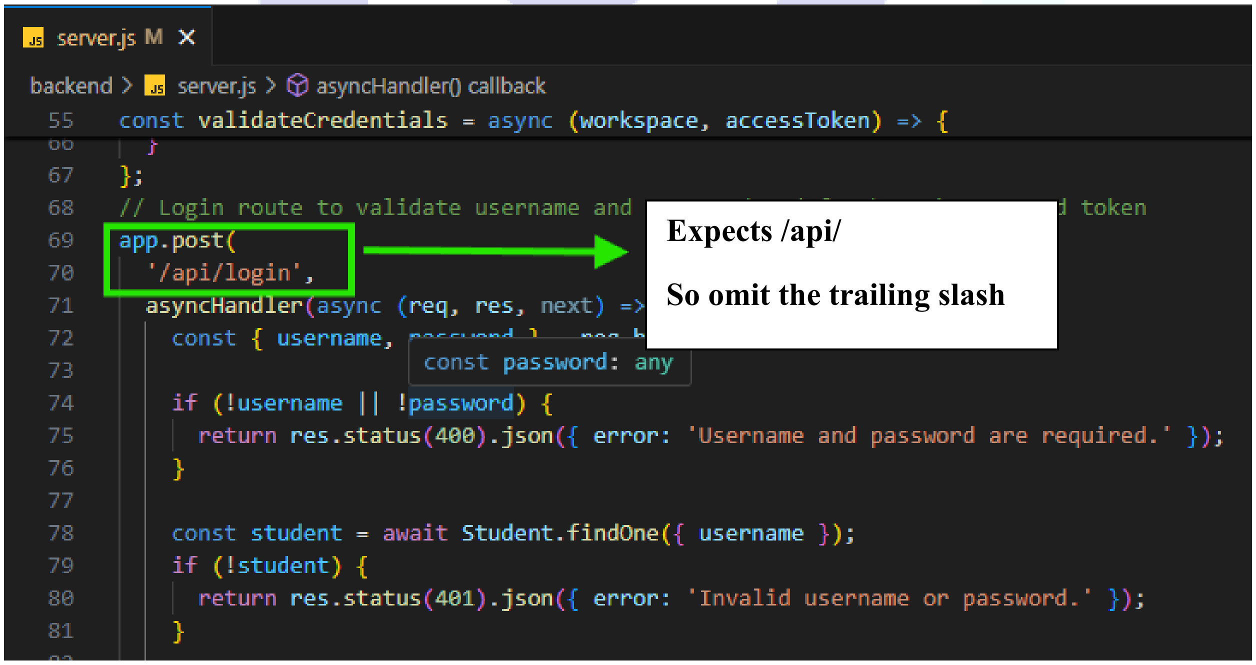
```javascript
app.get('/admin/login', handler);
```

→ The backend **does not expect** `/api/`, so use the trailing slash (`proxy_pass http://localhost:4000/;`).

If the backend defines routes like:

```javascript
app.get('/api/admin/login', handler);
```

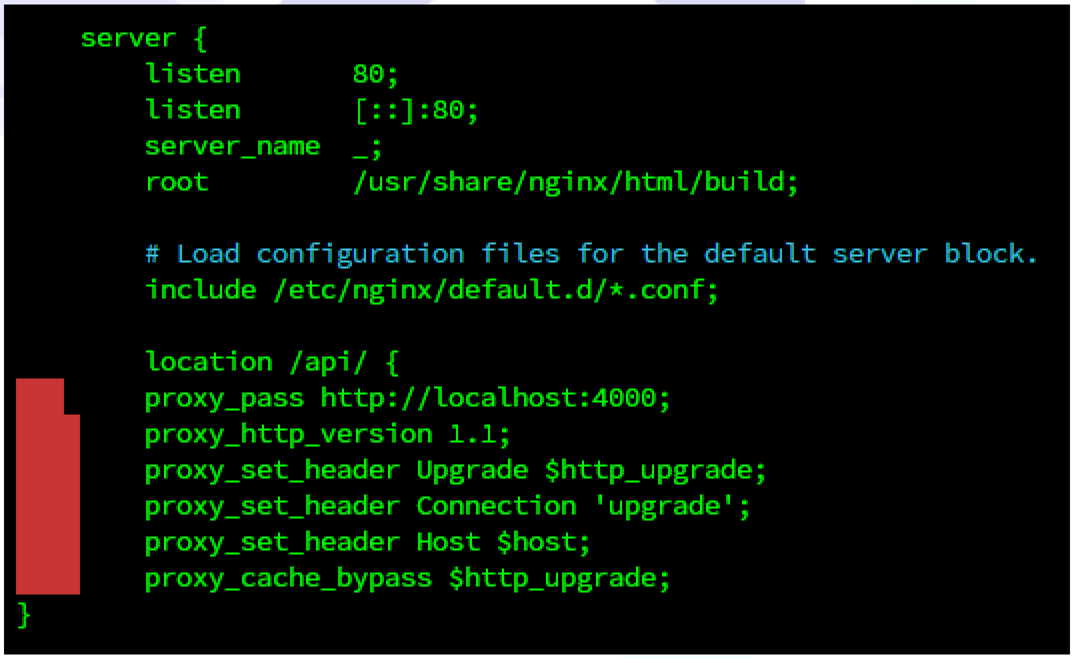
→ The backend **expects** `/api/`, so do not use the trailing slash (`proxy_pass http://localhost:4000;`).


server.js M
backend > server.js > asyncHandler() callback
Expects /api/
So omit the trailing slash
const password: any

```javascript
const validateCredentials = async (workspace, accessToken) => {
  }
};
// Login route to validate username and                                   token
app.post(
  '/api/login',
  asyncHandler(async (req, res, next) =>
    const { username, password } = req.b

    if (!username || !password) {
      return res.status(400).json({ error: 'Username and password are required.' });
    }

    const student = await Student.findOne({ username });
    if (!student) {
      return res.status(401).json({ error: 'Invalid username or password.' });
    }
```

```
sudo nano /etc/nginx/nginx.conf
```

```
location /api/ {
    proxy_pass http://localhost:4000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
}
```



```
    server {
        listen       80;
        listen       [::]:80;
        server_name  _;
        root         /usr/share/nginx/html/build;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        location /api/ {
        proxy_pass http://localhost:4000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
}
```

Now check for any inconsistencies in the configuration file and reload the nginx to apply changes.

```
sudo nginx -t
sudo nginx -s reload
```

17. Change SELinux policies to prevent restrictions

- Check the current SELinux status:

```
getenforce
```

- If SELinux is enforcing, allow Nginx to connect to the backend:

```
sudo setsebool -P httpd_can_network_connect 1
```

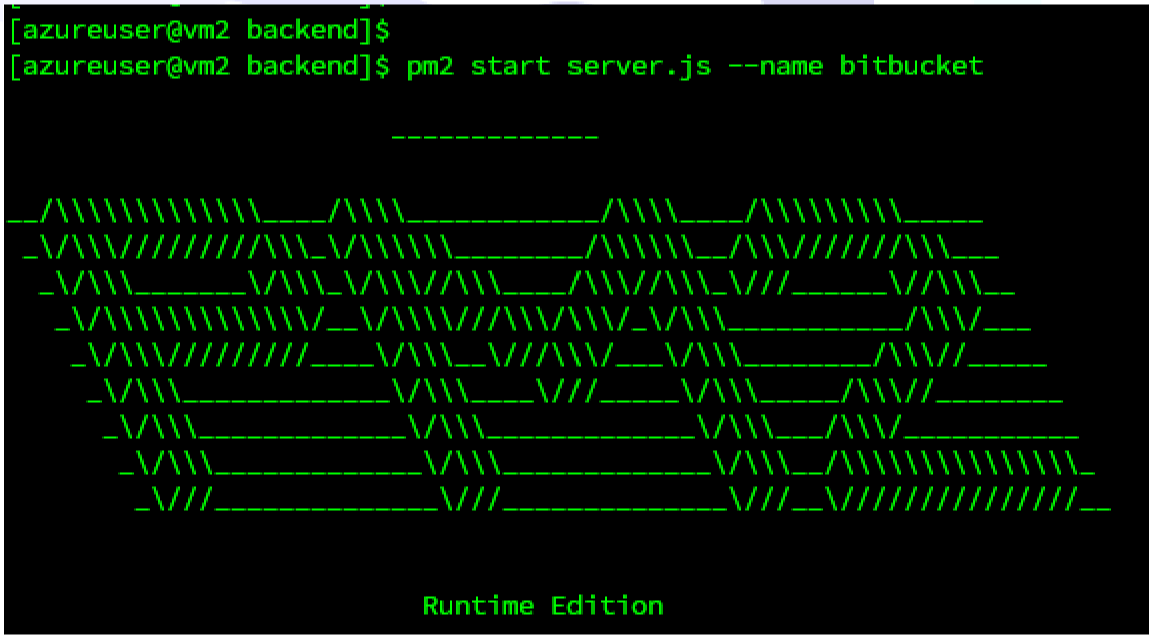
18. Run the main file in the backend using pm2

Go to backend folder.

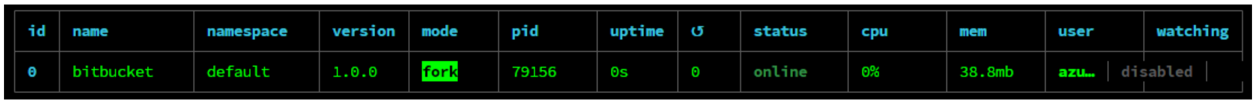
```
pm2 start <entry file> --name <process_name>
ex: pm2 start server.js --name bitbucket
```

Entry file can be: main.js/server.js/app.js/index.js

Process name can be any name : backend/node-process1

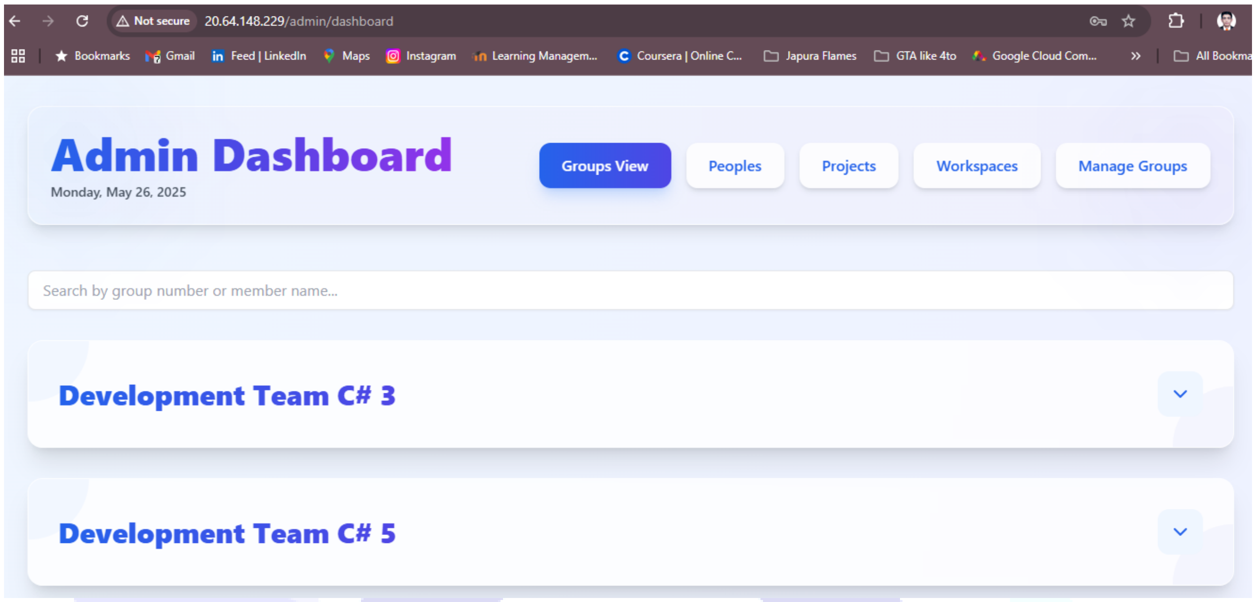


```
[azureuser@vm2 backend]$
[azureuser@vm2 backend]$ pm2 start server.js --name bitbucket

                        -------------

__/\\\\\\\\\\\\\____/\\\\____________/\\\\____/\\\\\\\\\_____
 _\/\\\/////////\\\_\/\\\\\\________/\\\\\\__/\\\///////\\\___
  _\/\\\_______\/\\\_\/\\\//\\\____/\\\//\\\_\///______\//\\\__
   _\/\\\\\\\\\\\\\/__\/\\\\///\\\/\\\/_\/\\\___________/\\\/___
    _\/\\\/////////____\/\\\__\///\\\/___\/\\\________/\\\//_____
     _\/\\\_____________\/\\\____\///_____\/\\\_____/\\\//________
      _\/\\\_____________\/\\\_____________\/\\\___/\\\/___________
       _\/\\\_____________\/\\\_____________\/\\\__/\\\\\\\\\\\\\\\_
        _\///______________\///______________\///__\///////////////__


                          Runtime Edition
```



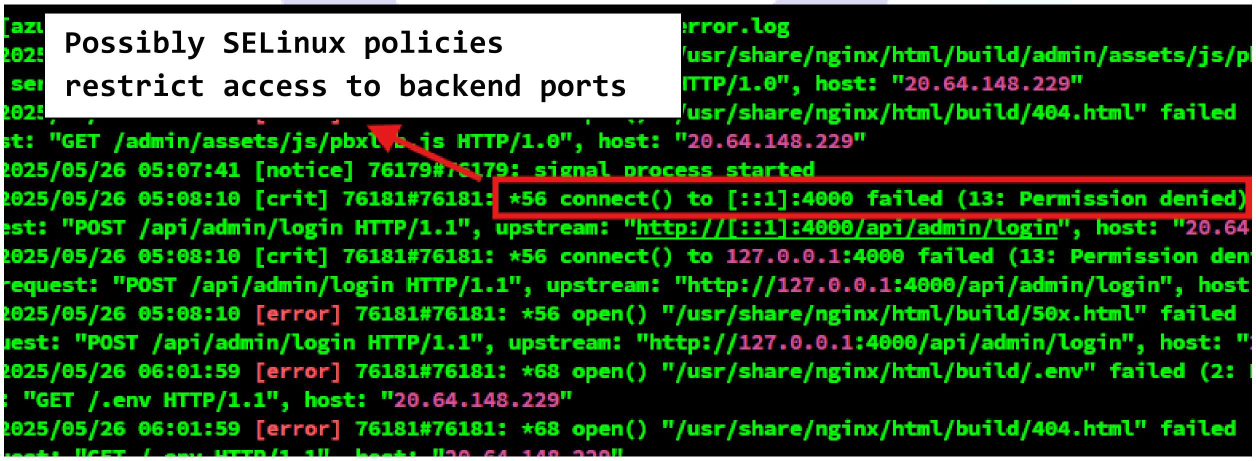
id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
0	bitbucket	default	1.0.0	fork	79156	0s	0	online	0%	38.8mb	azu…	disabled

19. Access the application from the frontend



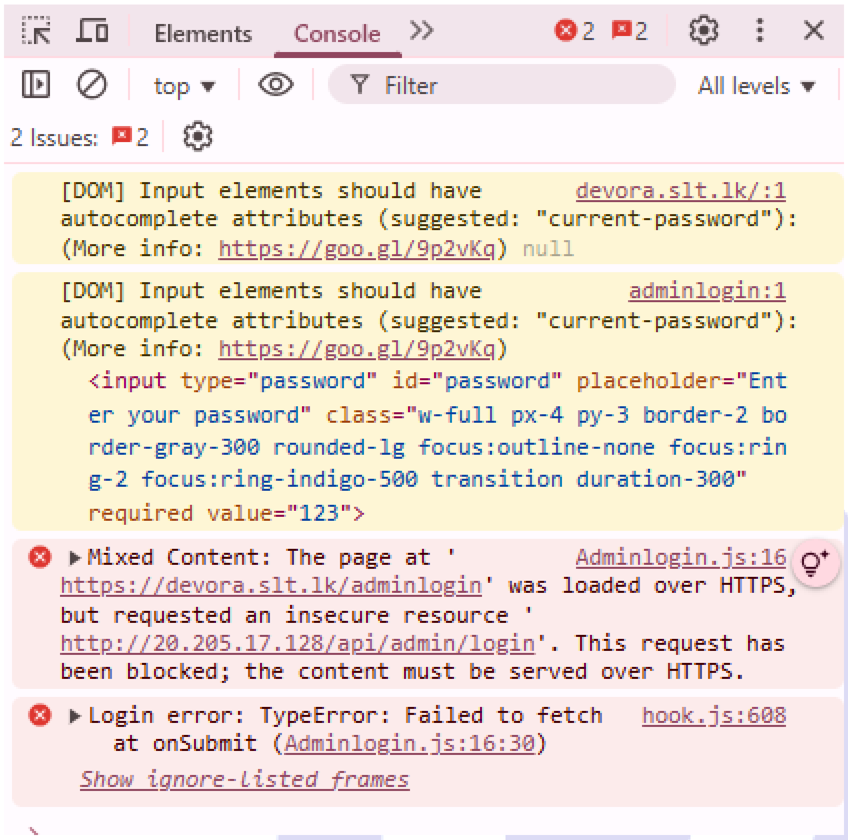
20. Check for any errors using nginx log files

```
sudo tail -f /var/log/nginx/error.log
sudo tail -f /var/log/nginx/access.log
```



```
[azu                                              rror.log
2025                                              /usr/share/nginx/html/build/admin/assets/js/p
 ser                                              TTP/1.0", host: "20.64.148.229"
2025                                              /usr/share/nginx/html/build/404.html" failed
st: "GET /admin/assets/js/pbxl   js HTTP/1.0", host: "20.64.148.229"
2025/05/26 05:07:41 [notice] 76179#76179: signal process started
2025/05/26 05:08:10 [crit] 76181#76181: *56 connect() to [::1]:4000 failed (13: Permission denied)
est: "POST /api/admin/login HTTP/1.1", upstream: "http://[::1]:4000/api/admin/login", host: "20.64
2025/05/26 05:08:10 [crit] 76181#76181: *56 connect() to 127.0.0.1:4000 failed (13: Permission den
request: "POST /api/admin/login HTTP/1.1", upstream: "http://127.0.0.1:4000/api/admin/login", host
2025/05/26 05:08:10 [error] 76181#76181: *56 open() "/usr/share/nginx/html/build/50x.html" failed
uest: "POST /api/admin/login HTTP/1.1", upstream: "http://127.0.0.1:4000/api/admin/login", host: "
2025/05/26 06:01:59 [error] 76181#76181: *68 open() "/usr/share/nginx/html/build/.env" failed (2:
: "GET /.env HTTP/1.1", host: "20.64.148.229"
2025/05/26 06:01:59 [error] 76181#76181: *68 open() "/usr/share/nginx/html/build/404.html" failed
```

21. Common Errors



```
[DOM] Input elements should have                devora.slt.lk/:1
autocomplete attributes (suggested: "current-password"):
(More info: https://goo.gl/9p2vKq) null

[DOM] Input elements should have                    adminlogin:1
autocomplete attributes (suggested: "current-password"):
(More info: https://goo.gl/9p2vKq)
  <input type="password" id="password" placeholder="Ent
  er your password" class="w-full px-4 py-3 border-2 bo
  rder-gray-300 rounded-lg focus:outline-none focus:rin
  g-2 focus:ring-indigo-500 transition duration-300"
  required value="123">

▶Mixed Content: The page at '             Adminlogin.js:16
https://devora.slt.lk/adminlogin' was loaded over HTTPS,
but requested an insecure resource '
http://20.205.17.128/api/admin/login'. This request has
been blocked; the content must be served over HTTPS.

▶Login error: TypeError: Failed to fetch      hook.js:608
    at onSubmit (Adminlogin.js:16:30)
  Show ignore-listed frames
```

The above error shows that content was requested over HTTP but served via HTTPS and has been blocked the request.

This happens often due to incorrect frontend URL. The frontend Base URL might be http://<server-ip>

We have to change the .env file or frontend Base URL and rebuild the frontend.