Configuring Linux Server 20.04 LTS as a web server to host a React application

This how-to assumes a bare-bones 20.04 LTS Ubuntu Server freshly installed and ready to be configured, and with SSH access, as well as a production version of your application, which by default sits in the build folder within your solution.

From the command line, open the ssh session with

\ssh -l username 10.0.0.4

, altering username and ip address apropriately.

I'm a Windows guy, so I dislike having to sudo everything and typing and retyping my password, so I accept the associated risks and enable the root account with

#sudo passwd root

, at least for the initial configuration; it may be disabled later. If you choose to do the same, be sure to pick a strong password.

After setting the new password, switch to the root account with

#su –

Good practice recommends the creation of a new user account with limited privileges for daily use, regular usage of root account being strongly discouraged. The new account can be granted temporary super user rights on an as-needed basis.

Next step, securing the server. Our chosen distro should come with ufw (Uncomplicated Firewall) pre-installed but inactive. We'll check it with

# ufw status

The firewall must be first configured to allow ssh connections, lest we lock ourselves out when enabling it.

# ufw app list

expected response:

Available applications:

OpenSSH

# ufw allow OpenSSH

expected response:

Rules updated

Rules updated (v6)

C

# ufw enable

expected response:

Command may disrupt existing ssh connections. Proceed with operation (y|n)?

y

Firewall is active and enabled on system startup

# ufw status

Status: active

To Action From

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

OpenSSH ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

Now that our basic firewall is up and running, we'll install the light-weight web server, Nginx. But first, since this is the first time we're running the beast, we'd better update the local package index with

# apt update

and then

# apt install nginx

We can't forget to punch a hole in the firewall for the server. Check the ufw again with

# ufw app list

expected response:

Available applications:

Nginx Full

Nginx HTTP

Nginx HTTPS

OpenSSH

The first option tantamounts to the following two combined, i.e, ports tcp 80 and 443 will be opened. Choose whatever suits your needs. I'll go ahead and open both:

# ufw allow 'Nginx Full'

expected response:

Rule added

Rule added (v6)

We can then check it

# ufw status

expected response:

Status: active

To Action From

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

OpenSSH ALLOW Anywhere

Nginx Full ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

Nginx Full (v6) ALLOW Anywhere (v6)

While we're at it, let's check nginx too:

# systemctl status nginx

expedted response:

● nginx.service - A high performance web server and a reverse proxy server

Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)

Active: active (running) since Fri 2023-02-03 15:01:28 UTC; 9min ago

Docs: man:nginx(8)

Main PID: 6574 (nginx)

Tasks: 2 (limit: 1066)

Memory: 4.6M

CGroup: /system.slice/nginx.service

├─6574 nginx: master process /usr/sbin/nginx -g daemon on; master\_process on;

└─6575 nginx: worker process

Feb 03 15:01:28 216.207.178.68.host.secureserver.net systemd[1]: Starting A high performance web server and a reverse p>

Feb 03 15:01:28 216.207.178.68.host.secureserver.net systemd[1]: Started A high performance web server and a reverse pr>

lines 1-13/13 (END)

Now, type

# networkctl status

expected response:

● State: routable

Address: <ext IP number> on eth0

fe80::5054:d2ff:fea5:e686 on eth0

Gateway: 169.254.0.1 on eth0

DNS: 10.255.250.30

10.255.251.30

Feb 03 13:30:46 temp.secureserver.net systemd[1]: Starting Network Service...

Feb 03 13:30:46 temp.secureserver.net systemd-networkd[388]: Enumeration completed

Feb 03 13:30:46 temp.secureserver.net systemd[1]: Started Network Service.

Feb 03 13:30:46 temp.secureserver.net systemd[1]: Starting Wait for Network to be Configured...

Feb 03 13:30:46 temp.secureserver.net systemd-networkd[388]: eth0: IPv6 successfully enabled

Feb 03 13:30:46 temp.secureserver.net systemd-networkd[388]: eth0: Link UP

Feb 03 13:30:46 temp.secureserver.net systemd-networkd[388]: eth0: Gained carrier

Feb 03 13:30:48 temp.secureserver.net systemd-networkd[388]: eth0: Gained IPv6LL

Feb 03 13:30:48 temp.secureserver.net systemd[1]: Finished Wait for Network to be Configured.

…and open your browser, type

http://<ext IP number>

as per the output above, and you should see the default nginx page:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamenteNext, we have to download an install node like so:

# apt install nodejs

After it's done, you can check the version with

# node -v

expected response:

v10.19.0

Now it's time to install the node package manager, as follows:

# apt install npm

Once that's done, we’re all set to deploy our app.

The easiest way I've found to do it was:

On Windows, add all the files from the build folder into a .rar file (not the build folder, but rather its contents);

Copy the file (let's call it build.rar) onto a ftp server;

# ufw disable

Install rar on your Linux distro with

# apt install rar

Change directory on the Linux server to /var/www/html

At this point the firewall must be disabled

# ufw disable

Run the ftp client by typing

# ftp

expected response:

ftp>

Type

ftp> open <ftp server ip number>

expected response:

Connected to 177.70.106.XXX.

220 Microsoft FTP Service

Name (177.70.106.102:user):

Type your username and <enter>, inform your password and hit <enter> again

expected output:

331 Password required

Password:

230 User logged in.

Remote system type is Windows\_NT.

ftp>

binary

ftp> get aam.rar

local: aam.rar remote: aam.rar

200 PORT command successful.

125 Data connection already open; Transfer starting.

226 Transfer complete.

60324 bytes received in 0.51 secs (115.8946 kB/s)

ftp> bye

221 Goodbye.

root@216:/var/www/html#

re-enable firewall

# ufw enable

decompress files with

# rar x -y build.rar

If you previously enabled the root user in Ubuntu and now you want to disable it, set the root password to expire.

To disable the root account password, use the following command:

# passwd -l root

And that's all there is to it.