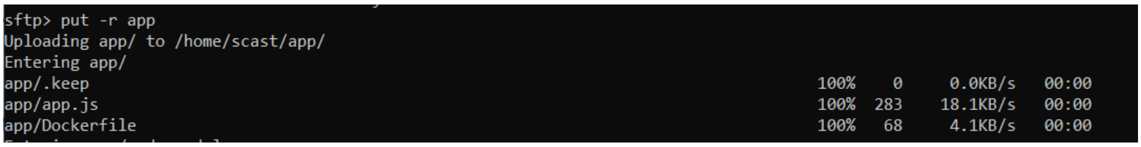
NIE-VCC Tutorial 5.

1. Build an image with a web application using the Dockerfile method.

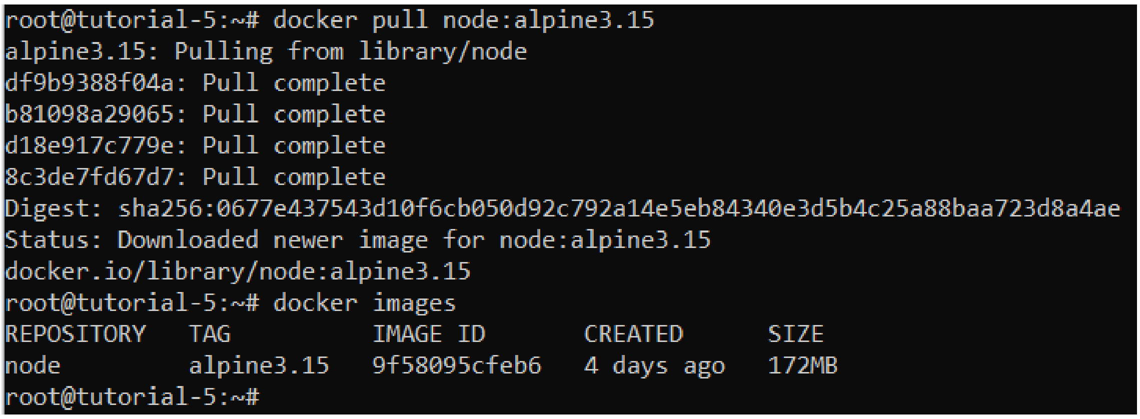
I created an image which for a web application which shows “Hola {name}! depending on the given URL (This work is taken from the course NIE-AM2, most specifically the homework 3) the needed files are on the folder “application from another subject”.

Let’s upload the folder to the virtual machine by sftp.



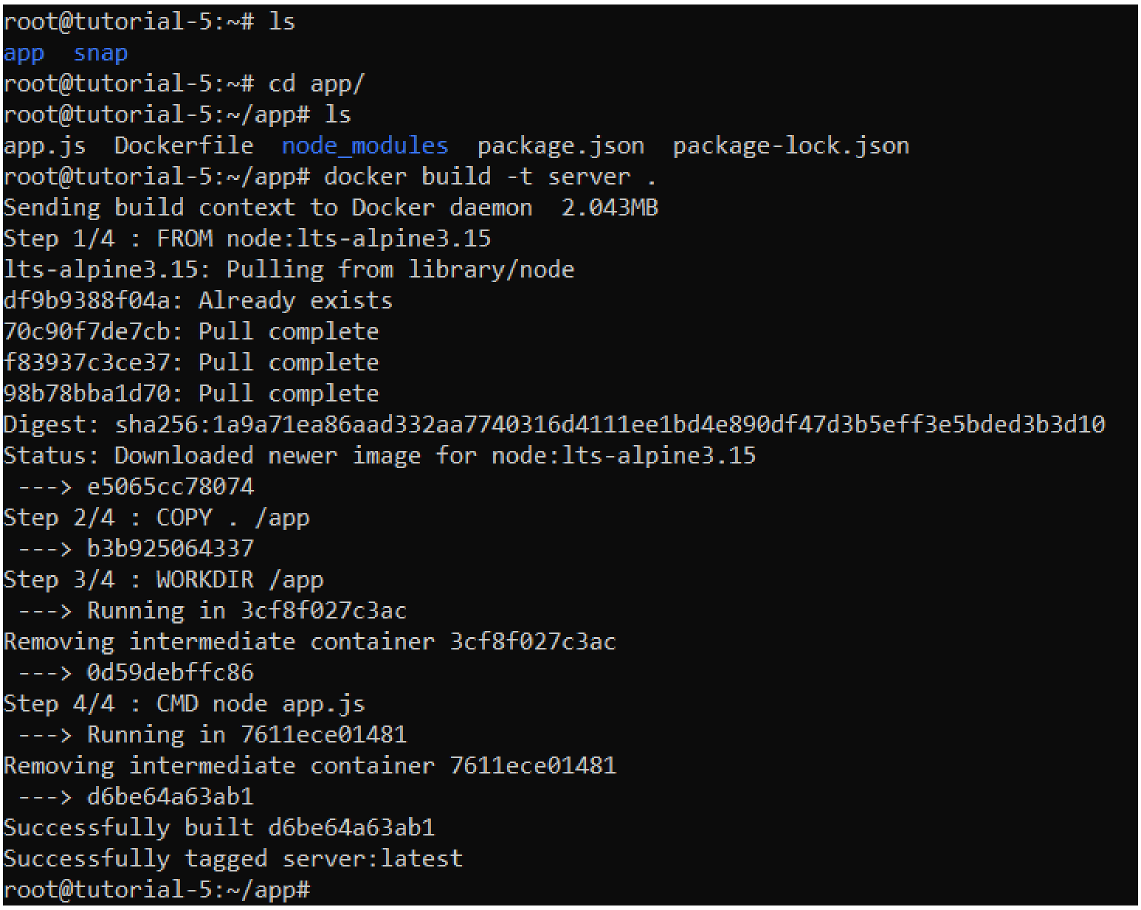
```
sftp> put -r app
Uploading app/ to /home/scast/app/
Entering app/
app/.keep                                                       100%    0     0.0KB/s   00:00
app/app.js                                                      100%  283    18.1KB/s   00:00
app/Dockerfile                                                  100%   68     4.1KB/s   00:00
```

Now let’s install the node:apline3.15 image because it is a simple image which consumes not so many resources and it is the image in which our server is created.



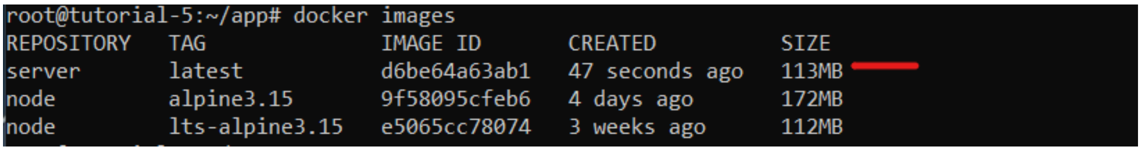
```
root@tutorial-5:~# docker pull node:alpine3.15
alpine3.15: Pulling from library/node
df9b9388f04a: Pull complete
b81098a29065: Pull complete
d18e917c779e: Pull complete
8c3de7fd67d7: Pull complete
Digest: sha256:0677e437543d10f6cb050d92c792a14e5eb84340e3d5b4c25a88baa723d8a4ae
Status: Downloaded newer image for node:alpine3.15
docker.io/library/node:alpine3.15
root@tutorial-5:~# docker images
REPOSITORY    TAG          IMAGE ID       CREATED      SIZE
node          alpine3.15   9f58095cfeb6   4 days ago   172MB
root@tutorial-5:~#
```

Now inside our app folder let’s execute the command “docker build -t server .” to create our new image which will have the implementation of the server.



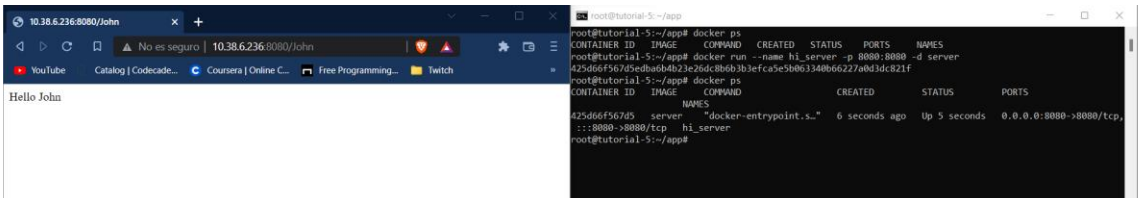
```
root@tutorial-5:~# ls
app  snap
root@tutorial-5:~# cd app/
root@tutorial-5:~/app# ls
app.js  Dockerfile  node_modules  package.json  package-lock.json
root@tutorial-5:~/app# docker build -t server .
Sending build context to Docker daemon  2.043MB
Step 1/4 : FROM node:lts-alpine3.15
lts-alpine3.15: Pulling from library/node
df9b9388f04a: Already exists
70c90f7de7cb: Pull complete
f83937c3ce37: Pull complete
98b78bba1d70: Pull complete
Digest: sha256:1a9a71ea86aad332aa7740316d4111ee1bd4e890df47d3b5eff3e5bded3b3d10
Status: Downloaded newer image for node:lts-alpine3.15
 ---> e5065cc78074
Step 2/4 : COPY . /app
 ---> b3b925064337
Step 3/4 : WORKDIR /app
 ---> Running in 3cf8f027c3ac
Removing intermediate container 3cf8f027c3ac
 ---> 0d59debffc86
Step 4/4 : CMD node app.js
 ---> Running in 7611ece01481
Removing intermediate container 7611ece01481
 ---> d6be64a63ab1
Successfully built d6be64a63ab1
Successfully tagged server:latest
root@tutorial-5:~/app#
```

Find our new docker image.



```
root@tutorial-5:~/app# docker images
REPOSITORY   TAG              IMAGE ID       CREATED          SIZE
server       latest           d6be64a63ab1   47 seconds ago   113MB
node         alpine3.15       9f58095cfeb6   4 days ago       172MB
node         lts-alpine3.15   e5065cc78074   3 weeks ago      112MB
```

Let’s build our server and create a container which will run it.



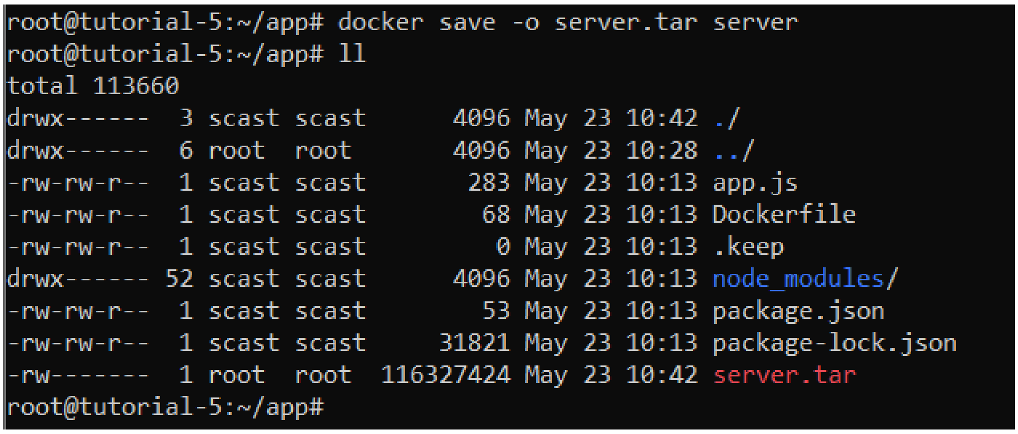
Hello John

```
root@tutorial-5:~/app# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@tutorial-5:~/app# docker run --name hi_server -p 8080:8080 -d server
425d66f567d5edba6b4b23e26dc8b6b3b3efca5e5b063340b66227a0d3dc821f
root@tutorial-5:~/app# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
                  NAMES
425d66f567d5   server    "docker-entrypoint.s…"   6 seconds ago   Up 5 seconds   0.0.0.0:8080->8080/tcp,
 :::8080->8080/tcp   hi_server
root@tutorial-5:~/app#
```

In the previous picture we can see how we create and run the container and how by specifying the name on the URL we get printed the greetings.

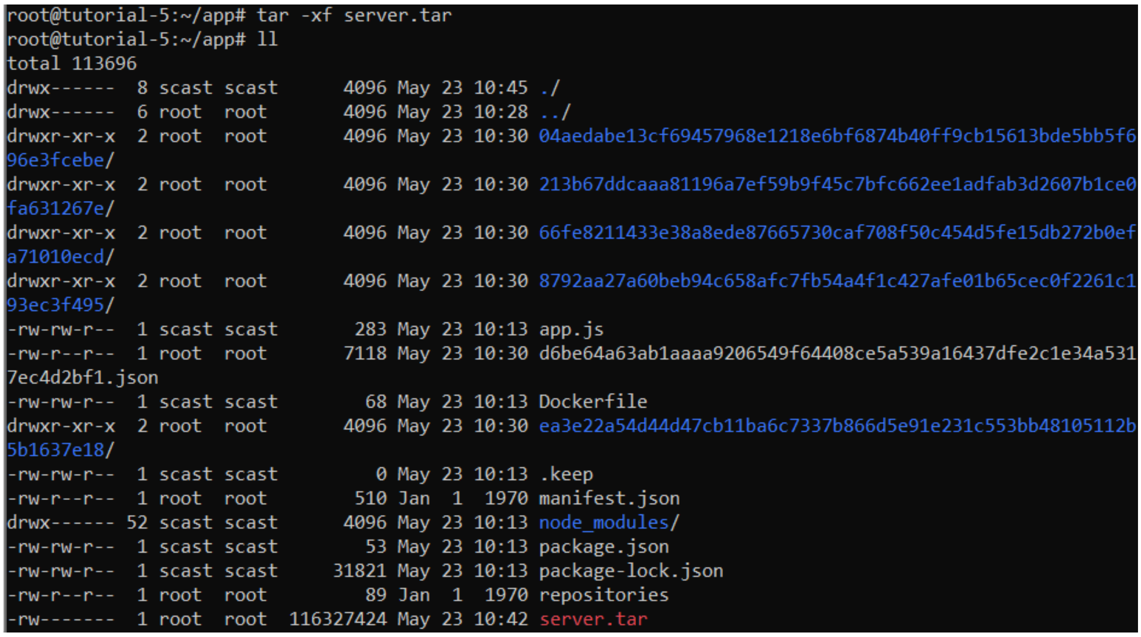
2. Analysis of the OCI / Docker image obtained from the previous step using "docker save"

I have saved the image of my server which is “server” in a tar file to analyse it by using the command “docker save -o server.tar server”



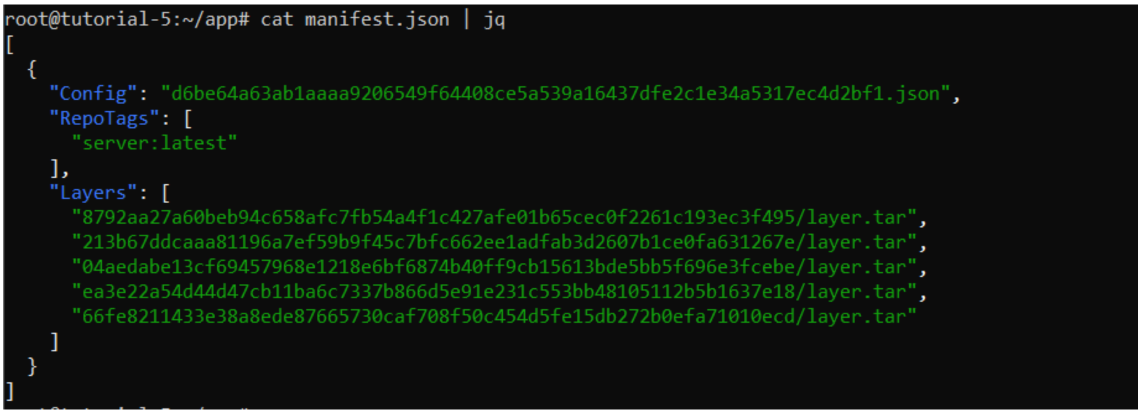
```
root@tutorial-5:~/app# docker save -o server.tar server
root@tutorial-5:~/app# ll
total 113660
drwx------  3 scast scast      4096 May 23 10:42 ./
drwx------  6 root  root       4096 May 23 10:28 ../
-rw-rw-r--  1 scast scast       283 May 23 10:13 app.js
-rw-rw-r--  1 scast scast        68 May 23 10:13 Dockerfile
-rw-rw-r--  1 scast scast         0 May 23 10:13 .keep
drwx------ 52 scast scast      4096 May 23 10:13 node_modules/
-rw-rw-r--  1 scast scast        53 May 23 10:13 package.json
-rw-rw-r--  1 scast scast     31821 May 23 10:13 package-lock.json
-rw-------  1 root  root  116327424 May 23 10:42 server.tar
root@tutorial-5:~/app#
```

Now let’s decompress the server.tar file and see the content with the command “tar -xf server.tar”



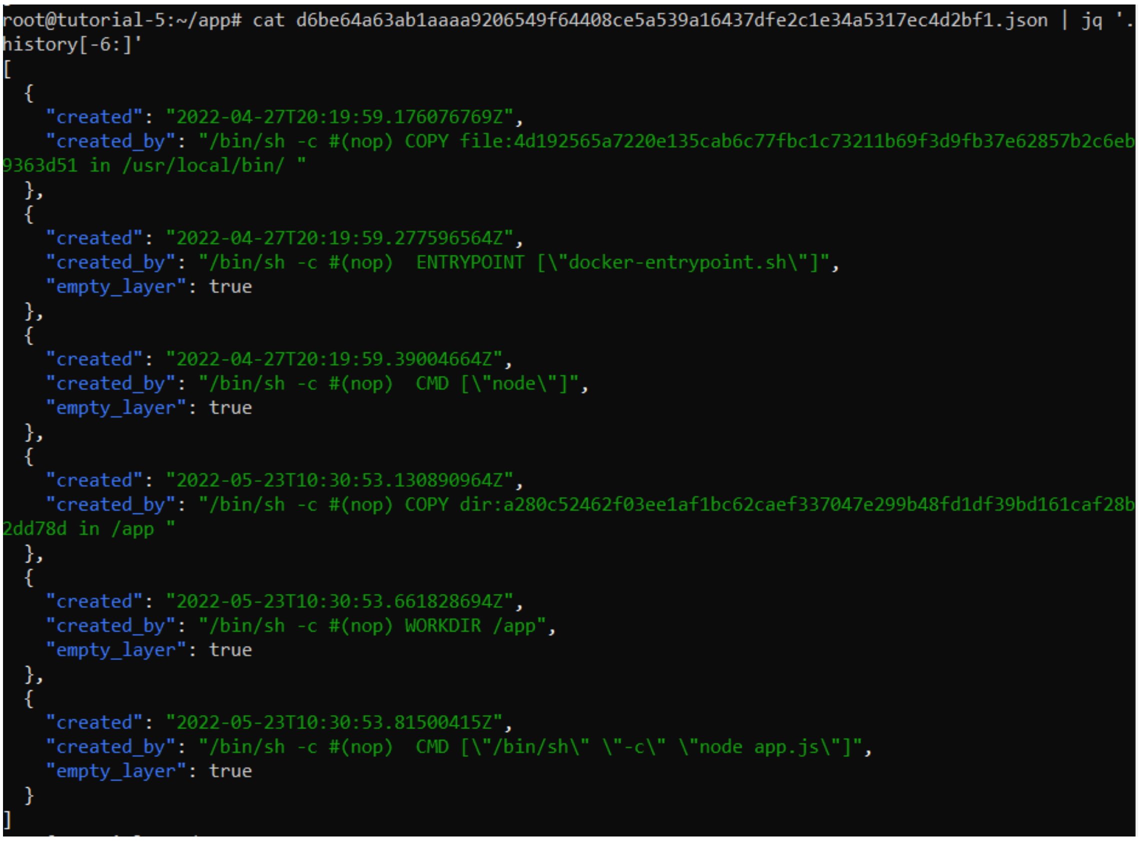
```
root@tutorial-5:~/app# tar -xf server.tar
root@tutorial-5:~/app# ll
total 113696
drwx------  8 scast scast      4096 May 23 10:45 ./
drwx------  6 root  root       4096 May 23 10:28 ../
drwxr-xr-x  2 root  root       4096 May 23 10:30 04aedabe13cf69457968e1218e6bf6874b40ff9cb15613bde5bb5f6
96e3fcebe/
drwxr-xr-x  2 root  root       4096 May 23 10:30 213b67ddcaaa81196a7ef59b9f45c7bfc662ee1adfab3d2607b1ce0
fa631267e/
drwxr-xr-x  2 root  root       4096 May 23 10:30 66fe8211433e38a8ede87665730caf708f50c454d5fe15db272b0ef
a71010ecd/
drwxr-xr-x  2 root  root       4096 May 23 10:30 8792aa27a60beb94c658afc7fb54a4f1c427afe01b65cec0f2261c1
93ec3f495/
-rw-rw-r--  1 scast scast       283 May 23 10:13 app.js
-rw-r--r--  1 root  root       7118 May 23 10:30 d6be64a63ab1aaaa9206549f64408ce5a539a16437dfe2c1e34a531
7ec4d2bf1.json
-rw-rw-r--  1 scast scast        68 May 23 10:13 Dockerfile
drwxr-xr-x  2 root  root       4096 May 23 10:30 ea3e22a54d44d47cb11ba6c7337b866d5e91e231c553bb48105112b
5b1637e18/
-rw-rw-r--  1 scast scast         0 May 23 10:13 .keep
-rw-r--r--  1 root  root        510 Jan  1  1970 manifest.json
drwx------ 52 scast scast      4096 May 23 10:13 node_modules/
-rw-rw-r--  1 scast scast        53 May 23 10:13 package.json
-rw-rw-r--  1 scast scast     31821 May 23 10:13 package-lock.json
-rw-r--r--  1 root  root         89 Jan  1  1970 repositories
-rw-------  1 root  root  116327424 May 23 10:42 server.tar
```

As we can see apart from the new directories we can see two json files: “manifest.json” file and “d6be64a63ab1aaaa9206549f64408ce5a539a16437dfe2c1e34a5317ec4d2bf1.json”



```
root@tutorial-5:~/app# cat manifest.json | jq
[
  {
    "Config": "d6be64a63ab1aaaa9206549f64408ce5a539a16437dfe2c1e34a5317ec4d2bf1.json",
    "RepoTags": [
      "server:latest"
    ],
    "Layers": [
      "8792aa27a60beb94c658afc7fb54a4f1c427afe01b65cec0f2261c193ec3f495/layer.tar",
      "213b67ddcaaa81196a7ef59b9f45c7bfc662ee1adfab3d2607b1ce0fa631267e/layer.tar",
      "04aedabe13cf69457968e1218e6bf6874b40ff9cb15613bde5bb5f696e3fcebe/layer.tar",
      "ea3e22a54d44d47cb11ba6c7337b866d5e91e231c553bb48105112b5b1637e18/layer.tar",
      "66fe8211433e38a8ede87665730caf708f50c454d5fe15db272b0efa71010ecd/layer.tar"
    ]
  }
]
```

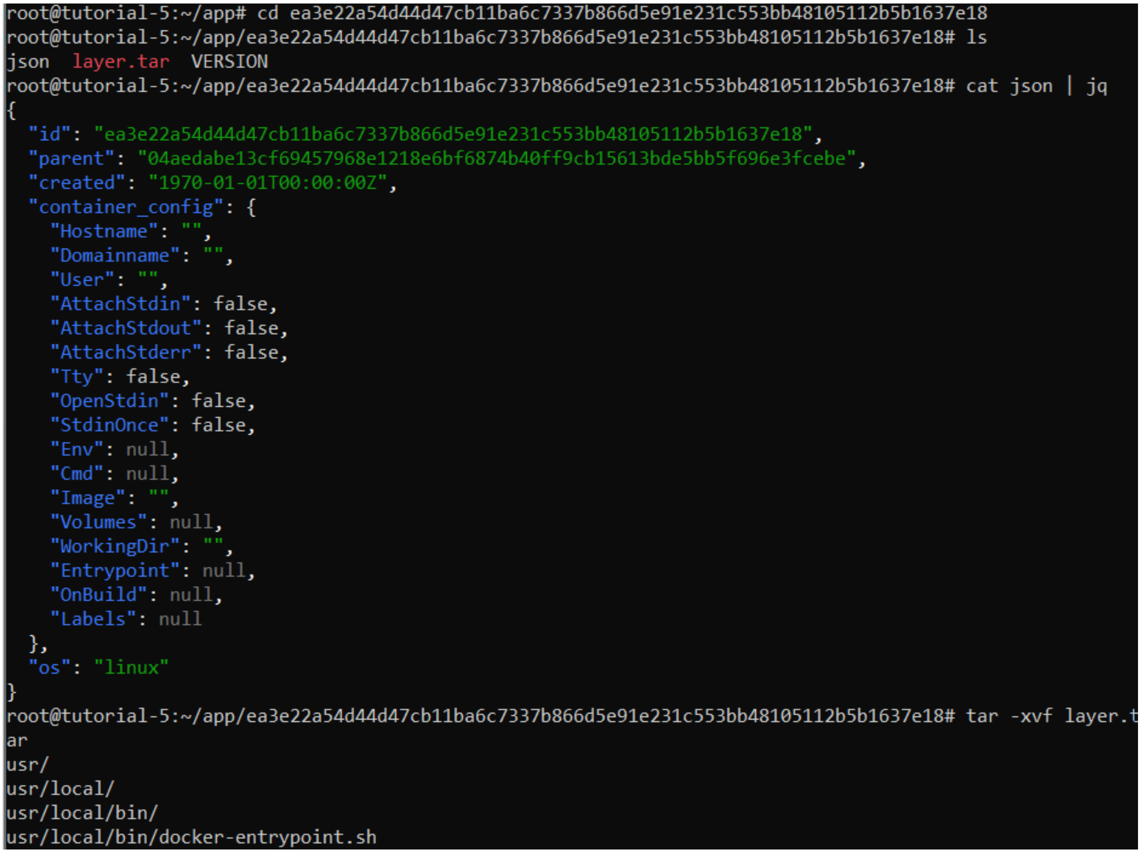
This the result of inspecting the manifest.json file.



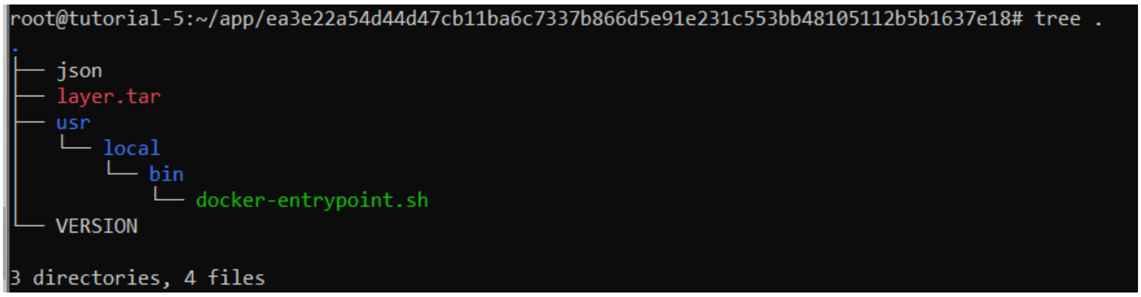
```
root@tutorial-5:~/app# cat d6be64a63ab1aaaa9206549f64408ce5a539a16437dfe2c1e34a5317ec4d2bf1.json | jq '.
history[-6:]'
[
  {
    "created": "2022-04-27T20:19:59.176076769Z",
    "created_by": "/bin/sh -c #(nop) COPY file:4d192565a7220e135cab6c77fbc1c73211b69f3d9fb37e62857b2c6eb
9363d51 in /usr/local/bin/ "
  },
  {
    "created": "2022-04-27T20:19:59.277596564Z",
    "created_by": "/bin/sh -c #(nop)  ENTRYPOINT [\"docker-entrypoint.sh\"]",
    "empty_layer": true
  },
  {
    "created": "2022-04-27T20:19:59.39004664Z",
    "created_by": "/bin/sh -c #(nop)  CMD [\"node\"]",
    "empty_layer": true
  },
  {
    "created": "2022-05-23T10:30:53.130890964Z",
    "created_by": "/bin/sh -c #(nop) COPY dir:a280c52462f03ee1af1bc62caef337047e299b48fd1df39bd161caf28b
2dd78d in /app "
  },
  {
    "created": "2022-05-23T10:30:53.661828694Z",
    "created_by": "/bin/sh -c #(nop) WORKDIR /app",
    "empty_layer": true
  },
  {
    "created": "2022-05-23T10:30:53.81500415Z",
    "created_by": "/bin/sh -c #(nop)  CMD [\"/bin/sh\" \"-c\" \"node app.js\"]",
    "empty_layer": true
  }
]
```

Here we can see the commands which has been executed these commands correspond with the Dockerfile.

If we inspect the layer before the last layer, should match with our work directory.



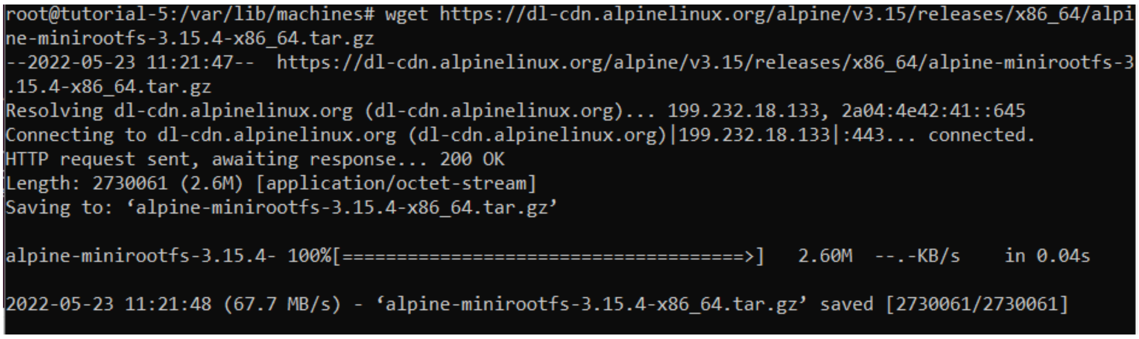
```
root@tutorial-5:~/app# cd ea3e22a54d44d47cb11ba6c7337b866d5e91e231c553bb48105112b5b1637e18
root@tutorial-5:~/app/ea3e22a54d44d47cb11ba6c7337b866d5e91e231c553bb48105112b5b1637e18# ls
json  layer.tar  VERSION
root@tutorial-5:~/app/ea3e22a54d44d47cb11ba6c7337b866d5e91e231c553bb48105112b5b1637e18# cat json | jq
{
  "id": "ea3e22a54d44d47cb11ba6c7337b866d5e91e231c553bb48105112b5b1637e18",
  "parent": "04aedabe13cf69457968e1218e6bf6874b40ff9cb15613bde5bb5f696e3fcebe",
  "created": "1970-01-01T00:00:00Z",
  "container_config": {
    "Hostname": "",
    "Domainname": "",
    "User": "",
    "AttachStdin": false,
    "AttachStdout": false,
    "AttachStderr": false,
    "Tty": false,
    "OpenStdin": false,
    "StdinOnce": false,
    "Env": null,
    "Cmd": null,
    "Image": "",
    "Volumes": null,
    "WorkingDir": "",
    "Entrypoint": null,
    "OnBuild": null,
    "Labels": null
  },
  "os": "linux"
}
root@tutorial-5:~/app/ea3e22a54d44d47cb11ba6c7337b866d5e91e231c553bb48105112b5b1637e18# tar -xvf layer.t
ar
usr/
usr/local/
usr/local/bin/
usr/local/bin/docker-entrypoint.sh
```



```
root@tutorial-5:~/app/ea3e22a54d44d47cb11ba6c7337b866d5e91e231c553bb48105112b5b1637e18# tree .
.
├── json
├── layer.tar
├── usr
│   └── local
│       └── bin
│           └── docker-entrypoint.sh
└── VERSION

3 directories, 4 files
```

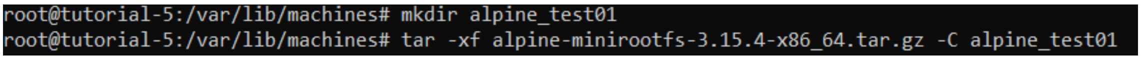
3. Creating a system container with a distribution other than Debian and Ubuntu

I have used the alpine distribution and the variant Mini root Filesystem.

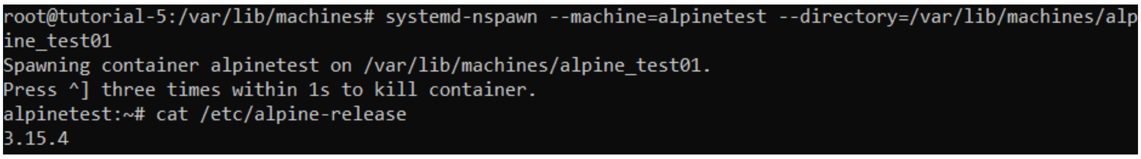


```
root@tutorial-5:/var/lib/machines# wget https://dl-cdn.alpinelinux.org/alpine/v3.15/releases/x86_64/alpi
ne-minirootfs-3.15.4-x86_64.tar.gz
--2022-05-23 11:21:47--  https://dl-cdn.alpinelinux.org/alpine/v3.15/releases/x86_64/alpine-minirootfs-3
.15.4-x86_64.tar.gz
Resolving dl-cdn.alpinelinux.org (dl-cdn.alpinelinux.org)... 199.232.18.133, 2a04:4e42:41::645
Connecting to dl-cdn.alpinelinux.org (dl-cdn.alpinelinux.org)|199.232.18.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2730061 (2.6M) [application/octet-stream]
Saving to: ‘alpine-minirootfs-3.15.4-x86_64.tar.gz’

alpine-minirootfs-3.15.4- 100%[===================================>]   2.60M  --.-KB/s    in 0.04s

2022-05-23 11:21:48 (67.7 MB/s) - ‘alpine-minirootfs-3.15.4-x86_64.tar.gz’ saved [2730061/2730061]
```

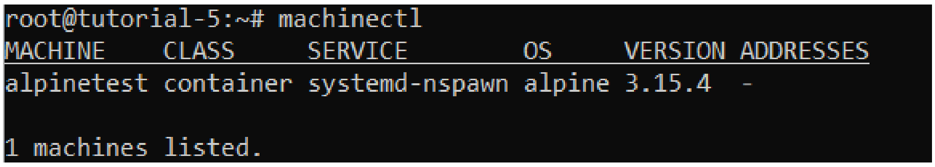
Here we can see the installation process and now let’s see how we start the container



```
root@tutorial-5:/var/lib/machines# mkdir alpine_test01
root@tutorial-5:/var/lib/machines# tar -xf alpine-minirootfs-3.15.4-x86_64.tar.gz -C alpine_test01
```



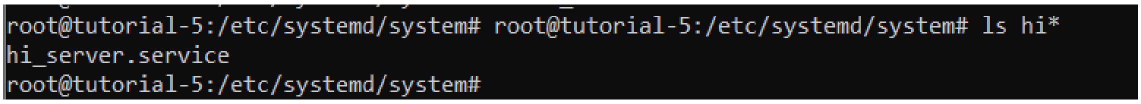
```
root@tutorial-5:/var/lib/machines# systemd-nspawn --machine=alpinetest --directory=/var/lib/machines/alp
ine_test01
Spawning container alpinetest on /var/lib/machines/alpine_test01.
Press ^] three times within 1s to kill container.
alpinetest:~# cat /etc/alpine-release
3.15.4
```



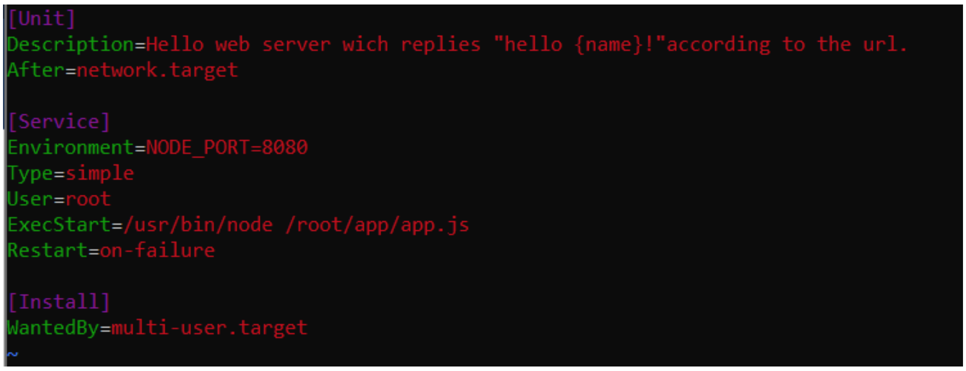
```
root@tutorial-5:~# machinectl
MACHINE    CLASS     SERVICE        OS     VERSION ADDRESSES
alpinetest container systemd-nspawn alpine 3.15.4  -

1 machines listed.
```

4. Writing a unit file for Systemd, which in "systemd-analysis security" will have at least a yellow security score.

Let’s create the file:

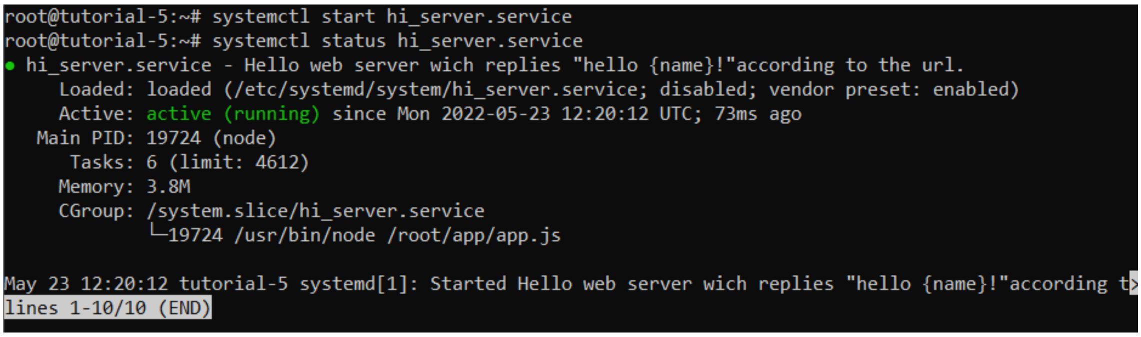


```
root@tutorial-5:/etc/systemd/system# root@tutorial-5:/etc/systemd/system# ls hi*
hi_server.service
root@tutorial-5:/etc/systemd/system#
```



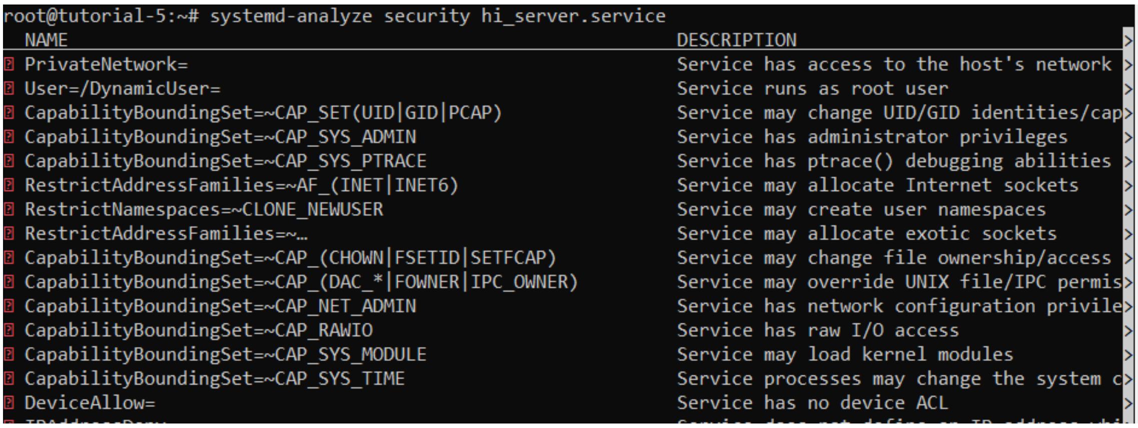
```
[Unit]
Description=Hello web server wich replies "hello {name}!"according to the url.
After=network.target

[Service]
Environment=NODE_PORT=8080
Type=simple
User=root
ExecStart=/usr/bin/node /root/app/app.js
Restart=on-failure

[Install]
WantedBy=multi-user.target
~
```

For this service to be able to run have had to install nodejs and pm with the command “apt install nodejs” and “apt install npm”



```
root@tutorial-5:~# systemctl start hi_server.service
root@tutorial-5:~# systemctl status hi_server.service
● hi_server.service - Hello web server wich replies "hello {name}!"according to the url.
     Loaded: loaded (/etc/systemd/system/hi_server.service; disabled; vendor preset: enabled)
     Active: active (running) since Mon 2022-05-23 12:20:12 UTC; 73ms ago
   Main PID: 19724 (node)
      Tasks: 6 (limit: 4612)
     Memory: 3.8M
     CGroup: /system.slice/hi_server.service
             └─19724 /usr/bin/node /root/app/app.js

May 23 12:20:12 tutorial-5 systemd[1]: Started Hello web server wich replies "hello {name}!"according t>
lines 1-10/10 (END)
```

Here we have our service running, after this if we analyse the security of our service, we will see that it is unsafe.

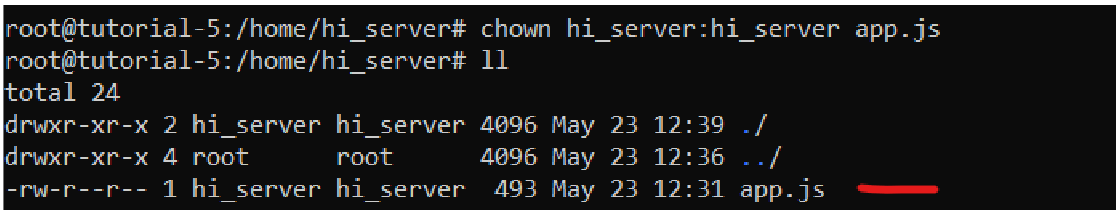


```
root@tutorial-5:~# systemd-analyze security hi_server.service
  NAME                                                        DESCRIPTION                                  >
▯ PrivateNetwork=                                             Service has access to the host's network     >
▯ User=/DynamicUser=                                          Service runs as root user                    >
▯ CapabilityBoundingSet=~CAP_SET(UID|GID|PCAP)                Service may change UID/GID identities/cap>
▯ CapabilityBoundingSet=~CAP_SYS_ADMIN                        Service has administrator privileges         >
▯ CapabilityBoundingSet=~CAP_SYS_PTRACE                       Service has ptrace() debugging abilities     >
▯ RestrictAddressFamilies=~AF_(INET|INET6)                    Service may allocate Internet sockets        >
▯ RestrictNamespaces=~CLONE_NEWUSER                           Service may create user namespaces           >
▯ RestrictAddressFamilies=~…                                  Service may allocate exotic sockets          >
▯ CapabilityBoundingSet=~CAP_(CHOWN|FSETID|SETFCAP)           Service may change file ownership/access     >
▯ CapabilityBoundingSet=~CAP_(DAC_*|FOWNER|IPC_OWNER)         Service may override UNIX file/IPC permis>
▯ CapabilityBoundingSet=~CAP_NET_ADMIN                        Service has network configuration privile>
▯ CapabilityBoundingSet=~CAP_RAWIO                            Service has raw I/O access                   >
▯ CapabilityBoundingSet=~CAP_SYS_MODULE                       Service may load kernel modules              >
▯ CapabilityBoundingSet=~CAP_SYS_TIME                         Service processes may change the system c>
▯ DeviceAllow=                                                Service has no device ACL                    >
```

Let’s start with the changes:

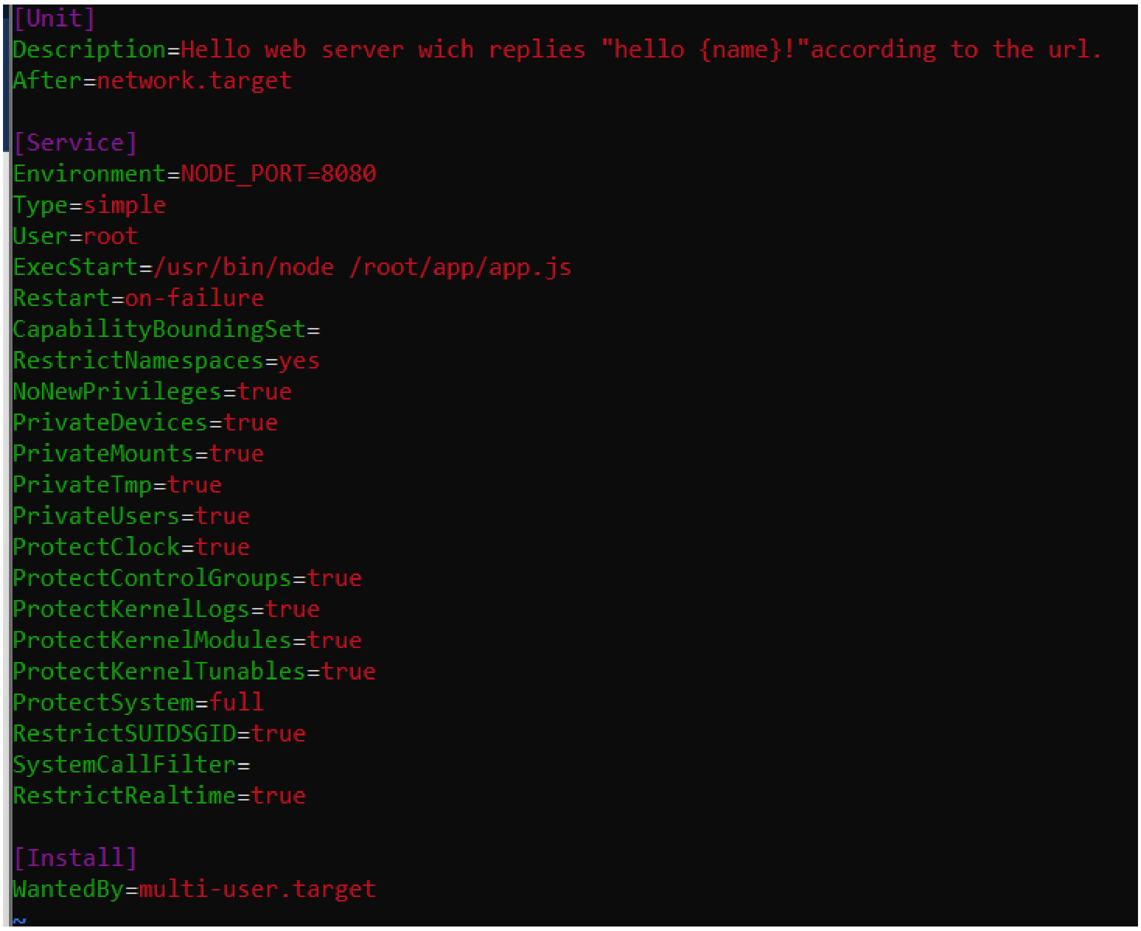
- Service run as a root user.

> Let’s add a user with the name of the service with the command “adduser hi_server” let’s now move the app.js file which rund our server to the home directory of this new user and change the owner of the file to this user.



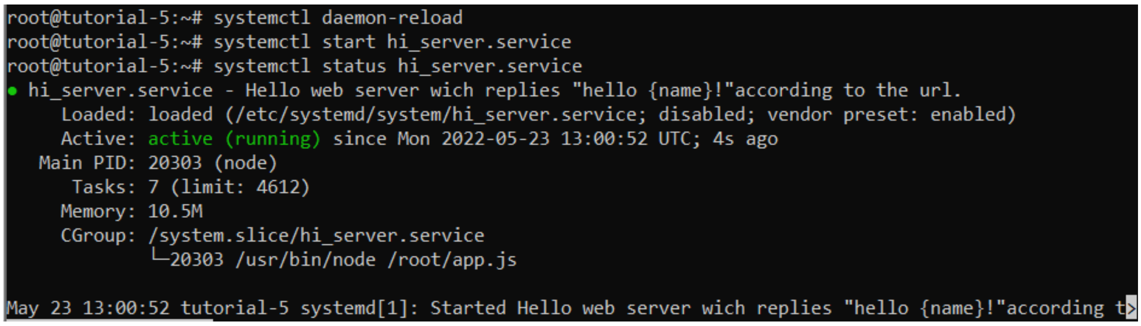
```
root@tutorial-5:/home/hi_server# chown hi_server:hi_server app.js
root@tutorial-5:/home/hi_server# ll
total 24
drwxr-xr-x 2 hi_server hi_server 4096 May 23 12:39 ./
drwxr-xr-x 4 root      root      4096 May 23 12:36 ../
-rw-r--r-- 1 hi_server hi_server  493 May 23 12:31 app.js
```

- Changes applied to the “hi_server.service” file.



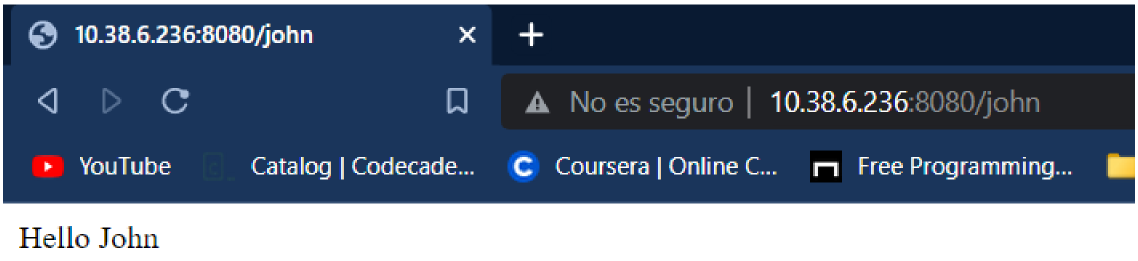
```
[Unit]
Description=Hello web server wich replies "hello {name}!"according to the url.
After=network.target

[Service]
Environment=NODE_PORT=8080
Type=simple
User=root
ExecStart=/usr/bin/node /root/app/app.js
Restart=on-failure
CapabilityBoundingSet=
RestrictNamespaces=yes
NoNewPrivileges=true
PrivateDevices=true
PrivateMounts=true
PrivateTmp=true
PrivateUsers=true
ProtectClock=true
ProtectControlGroups=true
ProtectKernelLogs=true
ProtectKernelModules=true
ProtectKernelTunables=true
ProtectSystem=full
RestrictSUIDSGID=true
SystemCallFilter=
RestrictRealtime=true

[Install]
WantedBy=multi-user.target
~
```

Now let’s restart the systemctl daemon with “systemct daemon-reload” and then restrart the server service.

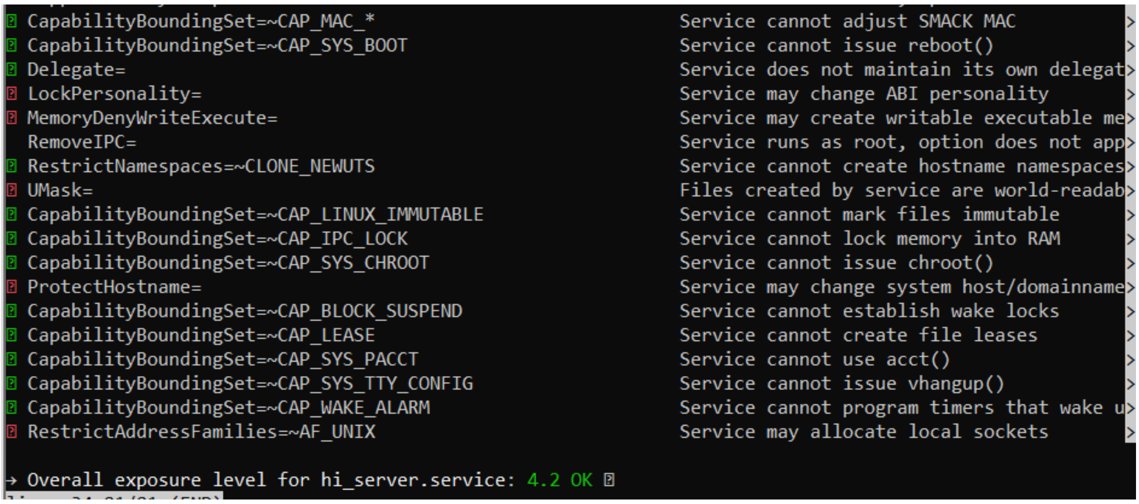


```
root@tutorial-5:~# systemctl daemon-reload
root@tutorial-5:~# systemctl start hi_server.service
root@tutorial-5:~# systemctl status hi_server.service
● hi_server.service - Hello web server wich replies "hello {name}!"according to the url.
     Loaded: loaded (/etc/systemd/system/hi_server.service; disabled; vendor preset: enabled)
     Active: active (running) since Mon 2022-05-23 13:00:52 UTC; 4s ago
   Main PID: 20303 (node)
      Tasks: 7 (limit: 4612)
     Memory: 10.5M
     CGroup: /system.slice/hi_server.service
             └─20303 /usr/bin/node /root/app.js

May 23 13:00:52 tutorial-5 systemd[1]: Started Hello web server wich replies "hello {name}!"according t>
```

Now if we inspect the security of our service we get a better score.



Hello John



```
□ CapabilityBoundingSet=~CAP_MAC_*                          Service cannot adjust SMACK MAC              >
□ CapabilityBoundingSet=~CAP_SYS_BOOT                       Service cannot issue reboot()                >
□ Delegate=                                                 Service does not maintain its own delegat>
□ LockPersonality=                                          Service may change ABI personality          >
□ MemoryDenyWriteExecute=                                   Service may create writable executable me>
  RemoveIPC=                                                Service runs as root, option does not app>
□ RestrictNamespaces=~CLONE_NEWUTS                          Service cannot create hostname namespaces>
□ UMask=                                                    Files created by service are world-readab>
□ CapabilityBoundingSet=~CAP_LINUX_IMMUTABLE                Service cannot mark files immutable          >
□ CapabilityBoundingSet=~CAP_IPC_LOCK                       Service cannot lock memory into RAM          >
□ CapabilityBoundingSet=~CAP_SYS_CHROOT                     Service cannot issue chroot()                >
□ ProtectHostname=                                          Service may change system host/domainname>
□ CapabilityBoundingSet=~CAP_BLOCK_SUSPEND                  Service cannot establish wake locks          >
□ CapabilityBoundingSet=~CAP_LEASE                          Service cannot create file leases            >
□ CapabilityBoundingSet=~CAP_SYS_PACCT                      Service cannot use acct()                    >
□ CapabilityBoundingSet=~CAP_SYS_TTY_CONFIG                 Service cannot issue vhangup()               >
□ CapabilityBoundingSet=~CAP_WAKE_ALARM                     Service cannot program timers that wake u>
□ RestrictAddressFamilies=~AF_UNIX                          Service may allocate local sockets           >

→ Overall exposure level for hi_server.service: 4.2 OK □
```