



# Containerized CRUD Application with Automation Scripts

By: Salma Salah

# Project's Description

This project is a containerized CRUD (Create, Read, Update, Delete) application that includes a frontend and backend, which interacts with a MongoDB database. The application is built using Docker, Docker Compose, and is deployed on an AWS EC2 instance running Ubuntu. Also, a daily backup of the MongoDB database is automated using a cron job.

# Steps

- I. Creating CRUD App
- II. Confirming that application is working locally
- III. Dockerize the application in three containers; the first for the application front end , the second for API and third for the database
- IV. Running the application using Docker-Compose
- V. Pushing the Application into git repo
- VI. Clon application into Main Server on AWS
- VII. Write scripts to automate building, running, and stopping the Docker containers
- VIII. Write scripts to transfer code to target server and build/run the Docker image on the VM
- IX. Write scripts to ensure successful deployment.
- X. Implement an Automated Daily Task by Automatically back up the application's data once per day.

# 1- Docker file for frontend

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "NODE-API-MONGO-APP". The "frontend" folder is expanded, showing "public", "src", ".dockerignore", ".env.development", ".gitignore", "Dockerfile", "package-lock.json", "package.json", "README.md", ".dockerignore", ".gitignore", "docker-compose.yml", and "README.md".
- Editor:** The "Dockerfile M" tab is active, displaying the following Dockerfile content:

```
FROM node:14-alpine
WORKDIR /app
# add '/app/node_modules/.bin' to $PATH
ENV PATH /app/node_modules/.bin:$PATH
# install application dependencies
COPY package*.json .
# RUN npm install react-scripts --force
RUN npm install
# copy app files
COPY . .
#Expose port 3000
EXPOSE 3000
CMD ["npm", "start"]
```

- Terminal:** The terminal tab is active, showing the command "PS F:\Node-API-Mongo-App\Node-API-Mongo-App>".
- Bottom Status Bar:** Shows the current file is "main\*", with 0 changes, and the status "Ln 22, Col 1 Spaces: 4 UTF-8 LF Dockerfile". It also displays system icons for battery, signal, and time (11:52 AM, 10/12/2024).

## 2- Docker file for Backend

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** On the left, it shows a project structure for "NODE-API-MONGO-APP" with a "backend" folder containing "models", "routes", "Dockerfile", "package-lock.json", "package.json", "server.js", "frontend", ".dockerignore", ".gitignore", "docker-compose.yml", and "README.md".
- Search Bar:** At the top center, it says "Node-API-Mongo-App".
- Tab Bar:** Shows ".env.development", "Dockerfile frontend M", "docker-compose.yml", and "Dockerfile backend M X".
- Dockerfile Content:** The "Dockerfile backend" tab is selected, displaying the following code:

```
1 FROM node:10-alpine
2
3 WORKDIR /usr/src/app
4
5 COPY package*.json .
6
7 RUN npm install
8
9 COPY .
10
11 EXPOSE 3001
12
13 CMD ["node", "server.js"]
```
- Terminal:** At the bottom, the terminal shows "PS F:\Node-API-Mongo-App\Node-API-Mongo-App>".
- Status Bar:** At the bottom right, it shows "Ln 15, Col 1 Spaces: 4 UTF-8 LF Dockerfile".
- System Tray:** At the very bottom, it shows a weather icon (26°C), a search bar, and various system icons.

## 3- Docker-Compose file

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure under "NODE-API-MONGO-APP". It includes folders for "backend" and "frontend", files like ".env.development", "Dockerfile", "package-lock.json", "package.json", "server.js", ".dockerignore", ".gitignore", and "README.md", and a selected "docker-compose.yml" file.
- Search Bar (Top):** Displays "Node-API-Mongo-App".
- Code Editor (Center):** The "docker-compose.yml" file is open, defining a multi-container application with three services: "web", "api", and "mongo". The "web" service depends on "api" and runs on port 3000. The "api" service depends on "mongo" and runs on port 3001. The "mongo" service uses the "mongo" image, restarts always, and maps the "/data/db" volume. Environment variables for MongoDB are defined, and a local access port is mapped to 27017.
- Bottom Status Bar:** Shows file status (main\*), line numbers (Ln 41, Col 1), and encoding (UTF-8). It also displays system icons for battery, signal, and time (11:55 AM, 10/12/2024).

# 4- Running app

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the title bar "Node-API-Mongo-App". The Explorer sidebar on the left displays the project structure:

- backend
- routes
- JS todos.js
- models
- JS todo.js
- index.js
- Dockerfile
- {} package-lock.json
- {} package.json
- JS server.js
- frontend
- public
- src
- .dockerrcignore
- .env.development
- .gitignore
- Dockerfile
- {} package-lock.json
- {} package.json
- README.md
- .dockerrcignore
- .gitignore
- docker-compose.yml
- README.md

The "Todos" tab in the editor is active, showing the following code:

```
const express = require("express");
const router = express.Router();
const Todo = require("../models/todo");

// GET all todos
router.get("/", async (req, res) => {
  const todos = await Todo.find({ is_complete: false });
  res.send(todos);
});

// GET todo based on ID
router.get("/:id", async (req, res) => {
```

The terminal tab shows the output of the "docker ps" command:

```
=> => writing image sha256:3862fb1ebcf82f90ecceb933933c0c7ac78bf063b619080 0.0s
=> => naming to docker.io/library/node-api-mongo-app-web 0.0s
[+] Running 5/5
✓ Network node-api-mongo-app_network-backend Created 0.3s
✓ Volume "node-api-mongo-app_mongodb_data" Created 0.0s
✓ Container node-api-mongo-app-mongo-1 Started 2.4s
✓ Container node-api-mongo-app-api-1 Started 3.9s
✓ Container node-api-mongo-app-web-1 Started 4.9s
PS F:\Node-API-Mongo-App\Node-API-Mongo-App> docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
699d599185c6 node-api-mongo-app-web "docker-entrypoint.s..." 3 minutes ago Up 3 minutes 0.0.0.0:3000->3000/tcp node-api-mongo-app-web-1
c7e80193ee06 node-api-mongo-app-api "docker-entrypoint.s..." 3 minutes ago Up 3 minutes 0.0.0.0:3001->3001/tcp node-api-mongo-app-api-1
bb2d4ca4c382 mongo "docker-entrypoint.s..." 3 minutes ago Up 3 minutes 27017/tcp node-api-mongo-app-mongo-1
PS F:\Node-API-Mongo-App\Node-API-Mongo-App>
```

The status bar at the bottom shows the following information:

- 23°C
- Safavi غالبا
- Search
- File Explorer
- Taskbar icons (File, Home, Recent, Task View)
- 2:30 AM
- 10/7/2024
- PURE logo

# 5- Application after running the three containers

The screenshot shows the Docker Desktop interface with the following details:

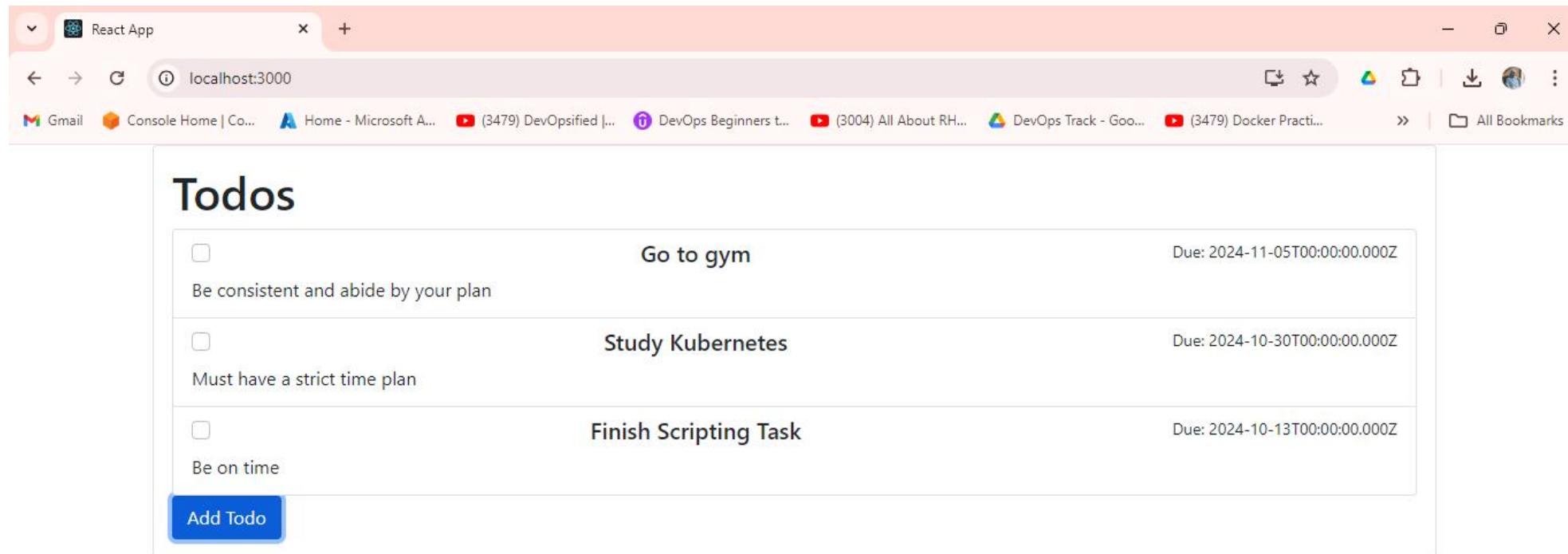
- Containers Tab:** Selected in the sidebar.
- Search Bar:** Shows "Search for images, containers, volumes, ext..." with a "Ctrl+K" hotkey.
- Metrics:** Container CPU usage: 1.09% / 800% (8 CPUs available) and Container memory usage: 285.33MB / 3.63GB. A "Show charts" link is present.
- Filter:** "Only show running containers" is selected.
- Table:** Displays the following container information:

Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<a href="#">node-api-mongo-app</a>		Running (3/3)		1.09%	5 minutes ago	[More]
<a href="#">api-1</a> c7e80193ee06	<a href="#">node-api-m</a>	Running	<a href="#">3001:3001</a>	0%	5 minutes ago	[More]
<a href="#">web-1</a> 699d599185c6	<a href="#">node-api-m</a>	Running	<a href="#">3000:3000</a>	0%	5 minutes ago	[More]
<a href="#">mongo-1</a> bb2d4ca4c382	<a href="#">mongo</a>	Running		1.09%	5 minutes ago	[More]

At the bottom, it says "Showing 4 items".

**System Tray:** Shows "Engine running", "23°C صاف غالباً", "RAM 2.27 GB CPU 0.25%", "New version available", "3 notifications", "2:29 AM 10/7/2024", and "ENG".

# 6- Frontend Api & DB working



# 7- Logging into db (todos)

The screenshot shows a VS Code interface with the title bar "Node-API-Mongo-App". The Explorer sidebar on the left shows a project structure for "NODE-API-MONGO-APP" with "backend" and "frontend" branches. The "backend" branch contains "models", "routes", "Dockerfile", "package-lock.json", "package.json", "server.js", ".dockerignore", and ".gitignore". The "frontend" branch contains "README.md". The "TERMINAL" tab is active, displaying the following MongoDB shell session:

```
2024-10-06T23:24:59.496+00:00: vm.max_map_count is too low
-----
test> show dbs
admin 40.00 KiB
config 92.00 KiB
local 40.00 KiB
todos 72.00 KiB
test> use todos
switched to db todos
todos> db.items.find().pretty()

todos> db.getCollectionInfos()
[
  {
    name: 'todos',
    type: 'collection',
    options: {},
    info: {
      readOnly: false,
      uuid: UUID('509c055d-cc63-4b1f-9f4a-442871f64848')
    },
    todos> db.products.find
```

The status bar at the bottom shows "Ln 39, Col 6 Spaces: 2 UTF-8 LF Compose". The system tray at the very bottom includes icons for weather (22°C), search, file explorer, file manager, phone, browser, and VS Code.

# 8- Adding more entries to our Todos

A screenshot of a web browser window titled "React App" displaying a todo list application at "localhost:3000". The page has a header "Todos" and a list of four items:

- Go to gym Due: 2024-11-05T00:00:00.000Z  
Be consistent and abide by your plan
- Study Kubernetes Due: 2024-10-30T00:00:00.000Z  
Must have a strict time plan
- Finish Scripting Task Due: 2024-10-13T00:00:00.000Z  
Be on time
- Off-Line Soft Skills Due: 2024-10-13T00:00:00.000Z  
Be on-time at 9:00 AM

A blue "Add Todo" button is located at the bottom left of the list.



# 9- Connecting to mongodb using MongoDB Compass (gui)

The screenshot shows the MongoDB Compass interface connected to the 'localhost:27017/todos.todos' database. The left sidebar lists connections, with 'localhost:27017' expanded to show 'admin', 'config', 'local', and 'todos'. The 'todos' connection is selected and highlighted with a blue border. The main pane displays the 'Todos' collection with three documents. The first document has the following fields:

```
_id: ObjectId('67031ebe99a347448ed10811')
title : "Study Kubernetes"
description : "Must have a strict time plan"
is_complete : false
due_date : 2024-10-30T00:00:00.000+00:00
__v : 0
```

The second document has the following fields:

```
_id: ObjectId('67031eea99a3474fe8d10812')
title : "Finish Scripting Task"
description : "Be on time"
is_complete : false
due_date : 2024-10-13T00:00:00.000+00:00
__v : 0
```

The third document has the following fields:

```
_id: ObjectId('670325ece3ccc50d261e5f12')
title : "Off-Line Soft Skills"
description : "Be on-time at 9:00 AM"
is_complete : false
due_date : 2024-10-13T00:00:00.000+00:00
__v : 0
```

The top navigation bar includes 'Connections', 'Edit', 'View', 'Collection', and 'Help'. The right side of the interface includes 'Open MongoDB shell' and various search and filter options.

# 10- Create local and remote git repositories and push all files to remote repo

The screenshot shows a GitHub repository page for 'Node-API-Mongo-App'. The repository is public and has 1 branch and 0 tags. The main commit is by 'salmasalam024' titled 'BackUp file after exexuting backup script now data on database will b...' with a timestamp of 'b2661e1 · 17 hours ago' and 8 commits. Below it is a commit for 'BackUp-Dir/mongodump\_20241011001034/to...'. The repository also contains folders for 'backend' and 'frontend', and files like '.dockerignore', '.gitignore', 'Automate-Daily-Backups.sh', and 'Copy\_Automate\_Deployment.sh'. The repository has 0 stars and 1 watching. The 'About' section describes the project as a containerized CRUD application with a frontend and backend interacting with a MongoDB database, built using Docker and Docker Compose, deployed on an AWS EC2 instance running Ubuntu, with daily backups automated via a cron job.

github.com/salmasalam024/Node-API-Mongo-App

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Node-API-Mongo-App Public

main 1 Branch 0 Tags

Go to file Add file Code

salmasalam024 BackUp file after exexuting backup script now data on database will b... b2661e1 · 17 hours ago 8 Commits

BackUp-Dir/mongodump\_20241011001034/to... BackUp file after exexuting backup script now data on datab... 17 hours ago

backend script to backup application data daily 2 days ago

frontend Bash scripts for Copying Deploying Testing and Stopping th... 3 days ago

.dockerignore Initial Commit 5 days ago

.gitignore Initial Commit 5 days ago

Automate-Daily-Backups.sh script to backup application data daily 2 days ago

Copy\_Automate\_Deployment.sh Bash scripts for Copying Deploying Testing and Stopping th... 3 days ago

About

This project is a containerized CRUD (Create, Read, Update, Delete) application that includes a frontend and backend, which interacts with a MongoDB database. The application is built using Docker, Docker Compose, and is deployed on an AWS EC2 instance running Ubuntu. Also a daily backup of the MongoDB database is automated using a cron job.

Readme

Activity

0 stars

1 watching

23°C صافي غالباً

Search

4:16 AM 10/13/2024

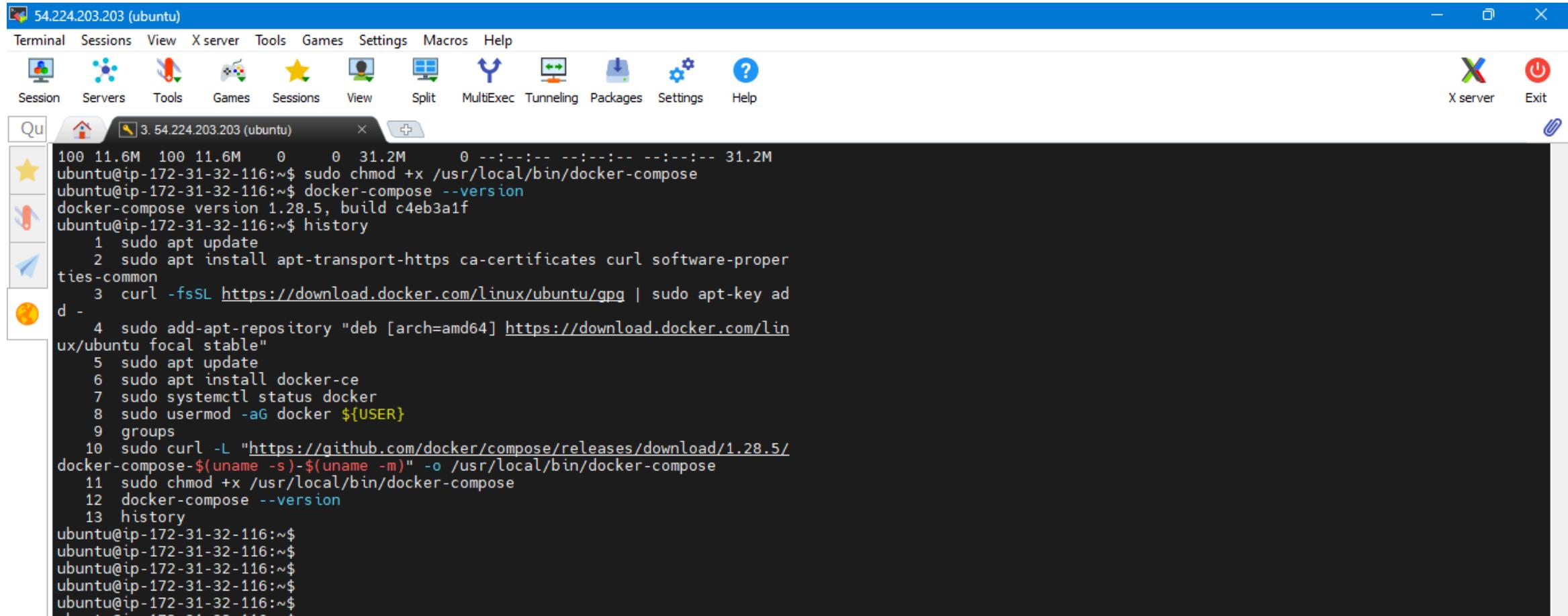
# 11- Creating Main Server on AWS (Ubuntu) and open required ports

The screenshot shows the AWS Management Console interface for the EC2 service. The main content area displays the 'Instance summary' for an instance named 'i-0fe9dc40a1294a815 (Main-Server)'. The instance is currently running. Key details shown include:

- Public IPv4 address:** 54.162.98.141 (with a link to 'open address')
- Private IPv4 addresses:** 172.31.32.116
- Public IPv4 DNS:** ec2-54-162-98-141.compute-1.amazonaws.com (with a link to 'open address')
- Instance state:** Running
- Private IP DNS name (IPv4 only):** ip-172-31-32-116.ec2.internal
- Instance type:** t2.micro
- VPC ID:** vpc-07e368fb8901161b4 (with a link)
- Subnet ID:** subnet-0b61a6f56cdd003f5 (with a link)
- Instance ARN:** arn:aws:ec2:us-east-1:821594020462:instance/i-0fe9dc40a1294a815

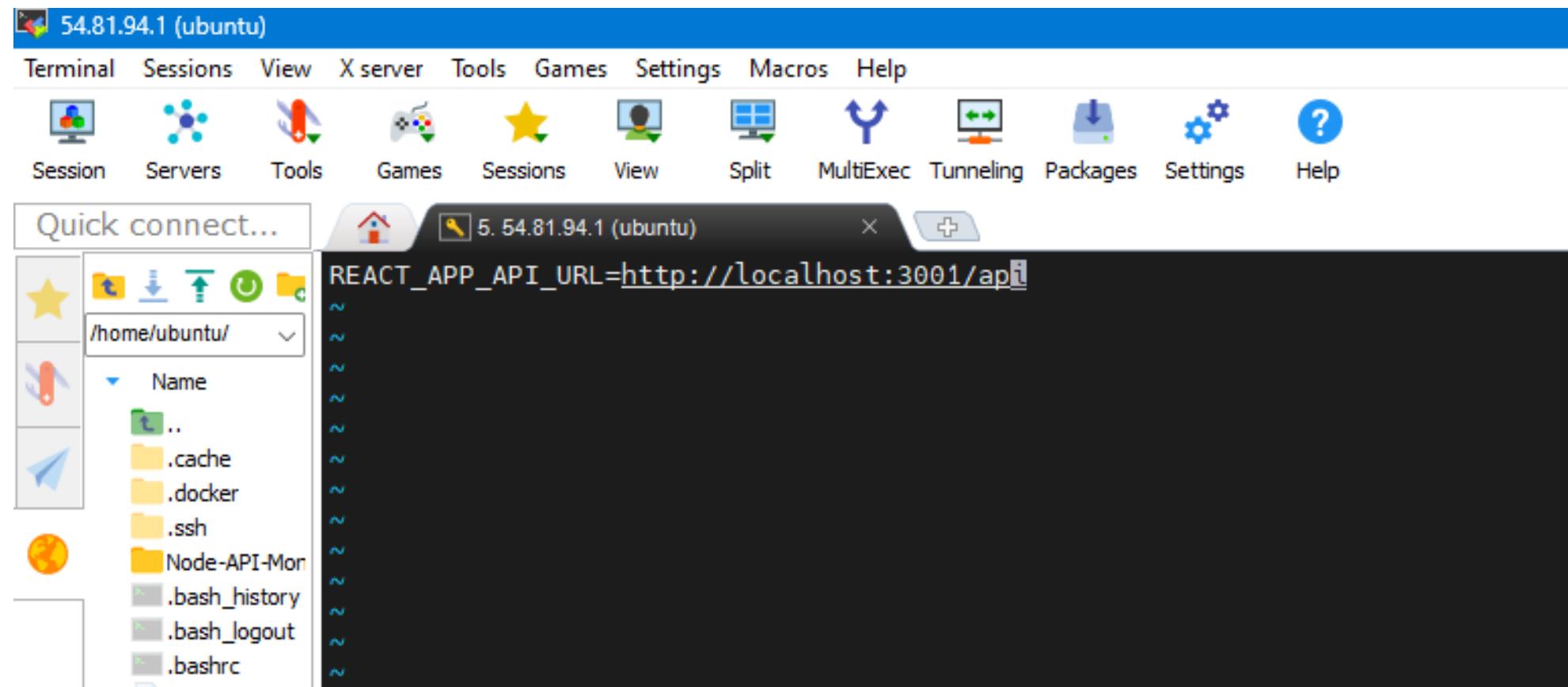
The left sidebar shows the navigation menu for the EC2 service, including 'Instances' (selected), 'Images', and 'Elastic Block Store'.

# 12- Install Docker and Docker-Compose



```
100 11.6M 100 11.6M 0 0 31.2M 0 --:--:-- --:--:-- --:--:-- 31.2M
ubuntu@ip-172-31-32-116:~$ sudo chmod +x /usr/local/bin/docker-compose
ubuntu@ip-172-31-32-116:~$ docker-compose --version
docker-compose version 1.28.5, build c4eb3a1f
ubuntu@ip-172-31-32-116:~$ history
 1 sudo apt update
 2 sudo apt install apt-transport-https ca-certificates curl software-proper
ties-common
 3 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key ad
d -
 4 sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/lin
ux/ubuntu focal stable"
 5 sudo apt update
 6 sudo apt install docker-ce
 7 sudo systemctl status docker
 8 sudo usermod -aG docker ${USER}
 9 groups
10 sudo curl -L "https://github.com/docker/compose/releases/download/1.28.5/
docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
11 sudo chmod +x /usr/local/bin/docker-compose
12 docker-compose --version
13 history
ubuntu@ip-172-31-32-116:~$
```

13- Note that at first the application was deployed on host machine so the API URL was <http://localhost:3001/api> but after deploying on the cloud vm (localhost) must be replaced by the public ip address of ec2 instance



# 13- API URL after modification with public ip address of ec2 (main server)

54.81.94.1 (ubuntu)

Terminal Sessions View X server Tools Games Settings Macros Help

Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages X server Exit

Quick connect... 5. 54.81.94.1 (ubuntu) +

Name

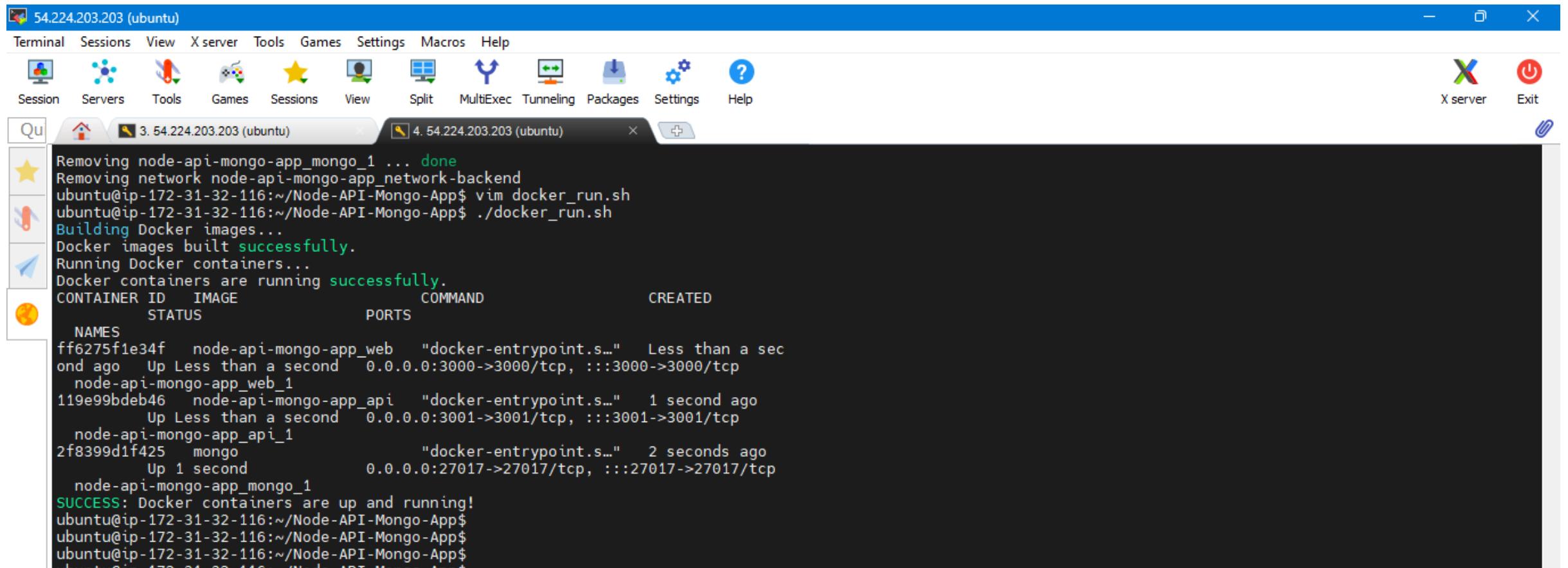
- ..
- .cache
- .docker
- .ssh
- Node-API-H
- .bash\_hist
- .bash\_logc
- .bashrc
- .gitconfig
- .profile
- .sudo\_as\_i
- .viminfo

REACT\_APP\_API\_URL=http://54.81.94.1:3001/api

:wq

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

# 14- Running application from main server on aws



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "3. 54.224.203.203 (ubuntu)". The terminal content shows the following steps to run a Docker application:

```
Removing node-api-mongo-app_mongo_1 ... done
Removing network node-api-mongo-app_network-backend
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ vim docker_run.sh
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ ./docker_run.sh
Building Docker images...
Docker images built successfully.
Running Docker containers...
Docker containers are running successfully.
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
 NAMES
ff6275f1e34f      node-api-mongo-app_web   "docker-entrypoint.s..."   Less than a sec
ond ago           Up Less than a second   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp
    node-api-mongo-app_web_1
119e99bdeb46      node-api-mongo-app_api   "docker-entrypoint.s..."   1 second ago
                Up Less than a second   0.0.0.0:3001->3001/tcp, :::3001->3001/tcp
    node-api-mongo-app_api_1
2f8399d1f425      mongo                 "docker-entrypoint.s..."   2 seconds ago
                Up 1 second            0.0.0.0:27017->27017/tcp, :::27017->27017/tcp
    node-api-mongo-app_mongo_1
SUCCESS: Docker containers are up and running!
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$
```

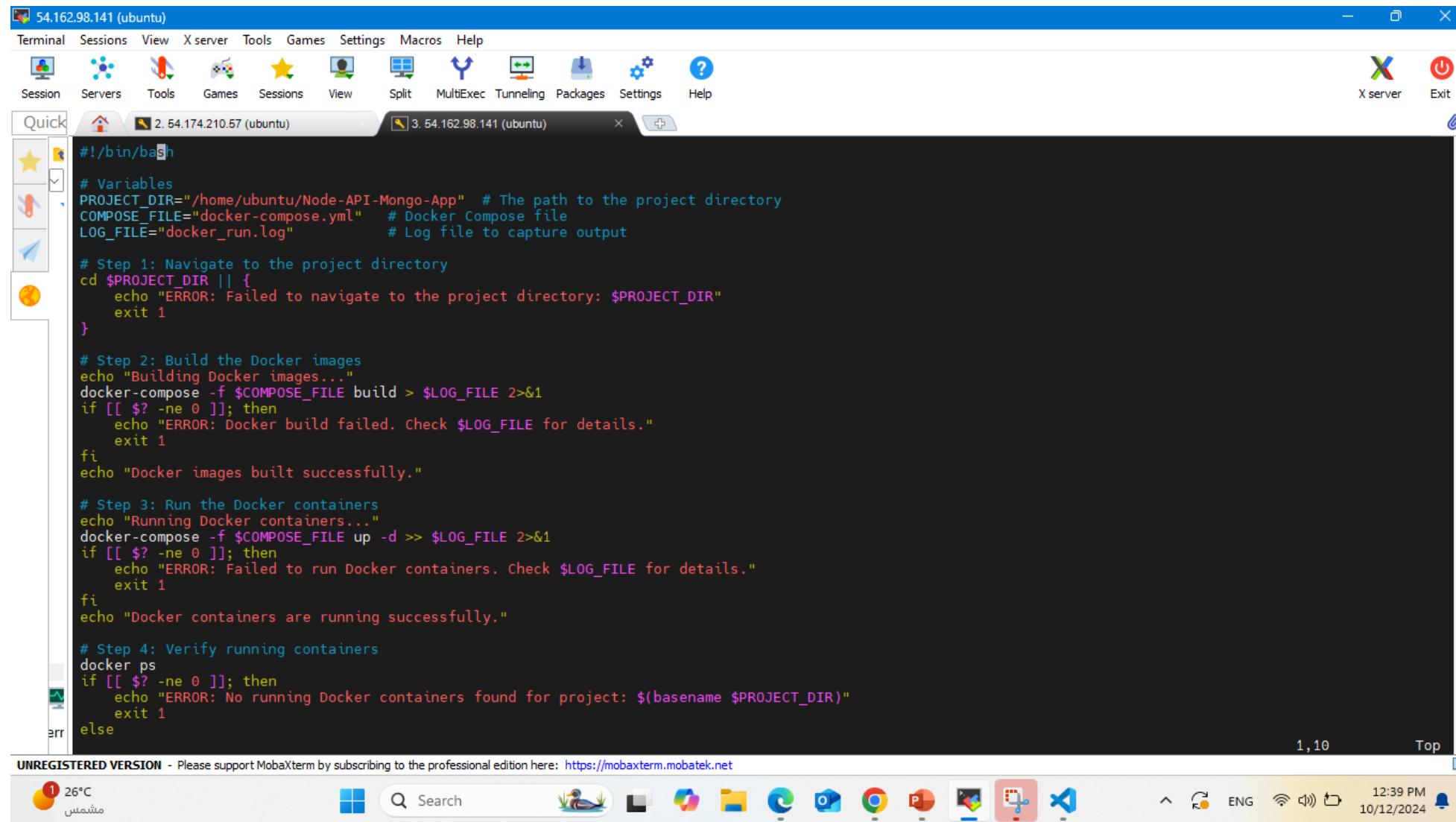
# 14- Application after deployment in main server on aws

The screenshot shows a web browser window with the following details:

- Title Bar:** React App
- Address Bar:** Not secure 54.81.94.1:3000
- Toolbar:** Back, Forward, Stop, Refresh, Home - Microsoft A..., YouTube, DevOpsified, DevOps Beginners t..., YouTube, All About RH..., DevOps Track - Goo..., Your Repositories, More, All Bookmarks.
- Content Area:**
  - Todos** heading
  - Problem Statement**:
    - 
    - Due: 2024-10-10T00:00:00.000Z
    - Text: Here is an issue I faced that first code was on my local machine , After pushing it to github and cloning on aws ec2 as the main-server; Only Frontend was working but not sending API calls to Backend
  - Solution**:
    - 
    - Due: 2024-10-10T00:00:00.000Z
    - Text: After troubleshooting, the issue was found that the React app API URL was set to http://localhost:3001/api and that was working on local machine and after running app on aws it has to be the Public IP address of the VM not localhost
- Buttons:** Add Todo



# 15- Writing Write scripts to automate building and running Docker containers



The screenshot shows a MobaXterm window titled "54.162.98.141 (ubuntu)". The terminal session displays a shell script for automating Docker container management. The script includes comments explaining the steps: navigating to the project directory, building Docker images, running Docker containers, and verifying their status. It uses docker-compose to handle the build and run processes.

```
#!/bin/bash

# Variables
PROJECT_DIR="/home/ubuntu/Node-API-Mongo-App" # The path to the project directory
COMPOSE_FILE="docker-compose.yml" # Docker Compose file
LOG_FILE="docker_run.log" # Log file to capture output

# Step 1: Navigate to the project directory
cd $PROJECT_DIR || {
    echo "ERROR: Failed to navigate to the project directory: $PROJECT_DIR"
    exit 1
}

# Step 2: Build the Docker images
echo "Building Docker images..."
docker-compose -f $COMPOSE_FILE build > $LOG_FILE 2>&1
if [[ $? -ne 0 ]]; then
    echo "ERROR: Docker build failed. Check $LOG_FILE for details."
    exit 1
fi
echo "Docker images built successfully."

# Step 3: Run the Docker containers
echo "Running Docker containers..."
docker-compose -f $COMPOSE_FILE up -d >> $LOG_FILE 2>&1
if [[ $? -ne 0 ]]; then
    echo "ERROR: Failed to run Docker containers. Check $LOG_FILE for details."
    exit 1
fi
echo "Docker containers are running successfully."

# Step 4: Verify running containers
docker ps
if [[ $? -ne 0 ]]; then
    echo "ERROR: No running Docker containers found for project: $(basename $PROJECT_DIR)"
    exit 1
else
    echo "Docker containers are running successfully."
fi
```

At the bottom of the terminal, a message reads: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>". The system tray at the bottom right shows the date and time as "10/12/2024 12:39 PM".

# 16- Writing a script to automate stopping the Docker containers

The screenshot shows a MobaXterm window with multiple sessions open. The current session is titled '54.162.98.141 (ubuntu)'. The terminal window displays a bash script for stopping Docker containers. The script sets variables for the project directory, Docker Compose file, and log file, then navigates to the project directory and uses docker-compose to stop the containers, logging the output to the specified log file.

```
#!/bin/bash

# Variables
PROJECT_DIR="/home/ubuntu/Node-API-Mongo-App" # Path to the project directory
COMPOSE_FILE="docker-compose.yml" # Docker Compose file
LOG_FILE="docker_stop.log" # Log file to capture output

# Step 1: Navigate to the project directory
cd $PROJECT_DIR || {
    echo "ERROR: Failed to navigate to the project directory: $PROJECT_DIR"
    exit 1
}

# Step 2: Stop Docker containers
echo "Stopping Docker containers..."
docker-compose -f $COMPOSE_FILE down > $LOG_FILE 2>&1
if [[ $? -ne 0 ]]; then
    echo "ERROR: Failed to stop Docker containers. Check $LOG_FILE for details."
    exit 1
fi

echo "Docker containers stopped and resources cleaned up successfully."
```

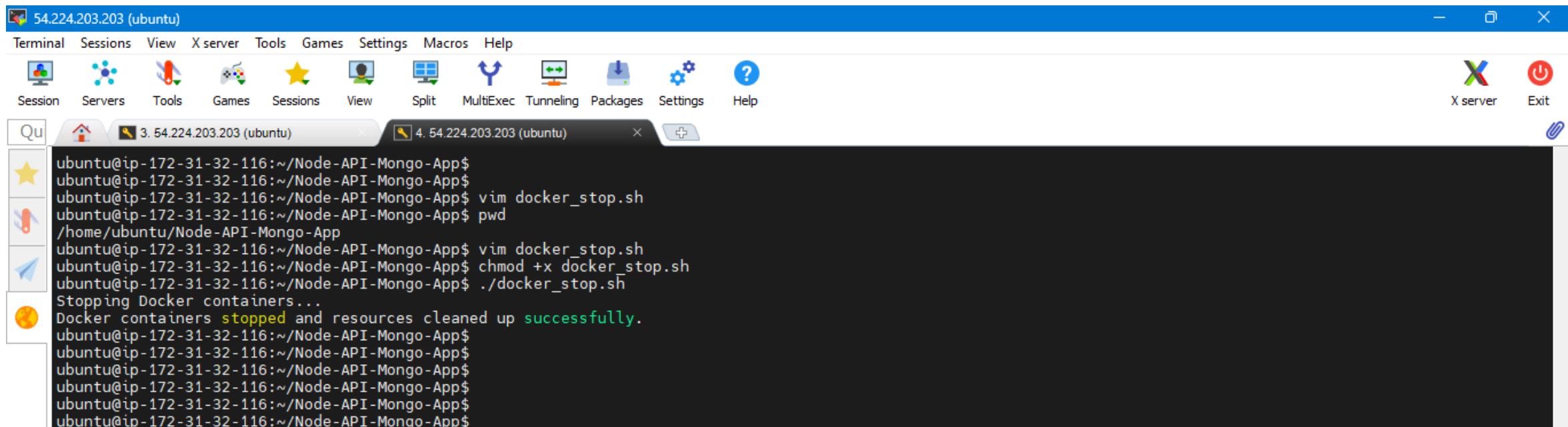
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

26°C مشرقي

Search

12:44 PM 10/12/2024

# 17- Give the running and stopping scripts execute permissions and testing their execution



The screenshot shows the Quassel IRC interface with a terminal window open. The terminal window title is "4. 54.224.203.203 (ubuntu)". The terminal content is as follows:

```
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ vim docker_stop.sh
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ chmod +x docker_stop.sh
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ ./docker_stop.sh
Stopping Docker containers...
Docker containers stopped and resources cleaned up successfully.
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$
```

# 18- Creating ssh keys to authenticate with Github repo to push the new scripts

The screenshot shows a MobaXterm window with three tabs open:

- Tab 3: 3. 54.224.203.203 (ubuntu) - Shows the command to generate an SSH key:

```
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ ssh-keygen -t ed25519 -C "salmasalam024@gmail.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_ed25519
Your public key has been saved in /home/ubuntu/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:8ewkedaRxzzDogEBZL0ycWEqco8yuo9b2iJ4/6jdK7M salmasalam024@gmail.com
The key's randomart image is:
+--[ED25519 256]--+
| .B |
| + = |
| . o + o.. |
| o + = =. |
| o . o S.= |
| . o B. ...o |
| o . . o....++ |
| +*..oo .o ..o |
| *+=oE=+. . |
+---[SHA256]----+
```

- Tab 4: 4. 54.224.203.203 (ubuntu) - Shows the command to eval the ssh-agent:

```
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ eval "$(ssh-agent -s)"
Agent pid 7600
```

- Tab 5: 5. 54.224.203.203 (ubuntu) - Shows the commands to add the key to the ssh-agent and then attempt to ssh to github.com:

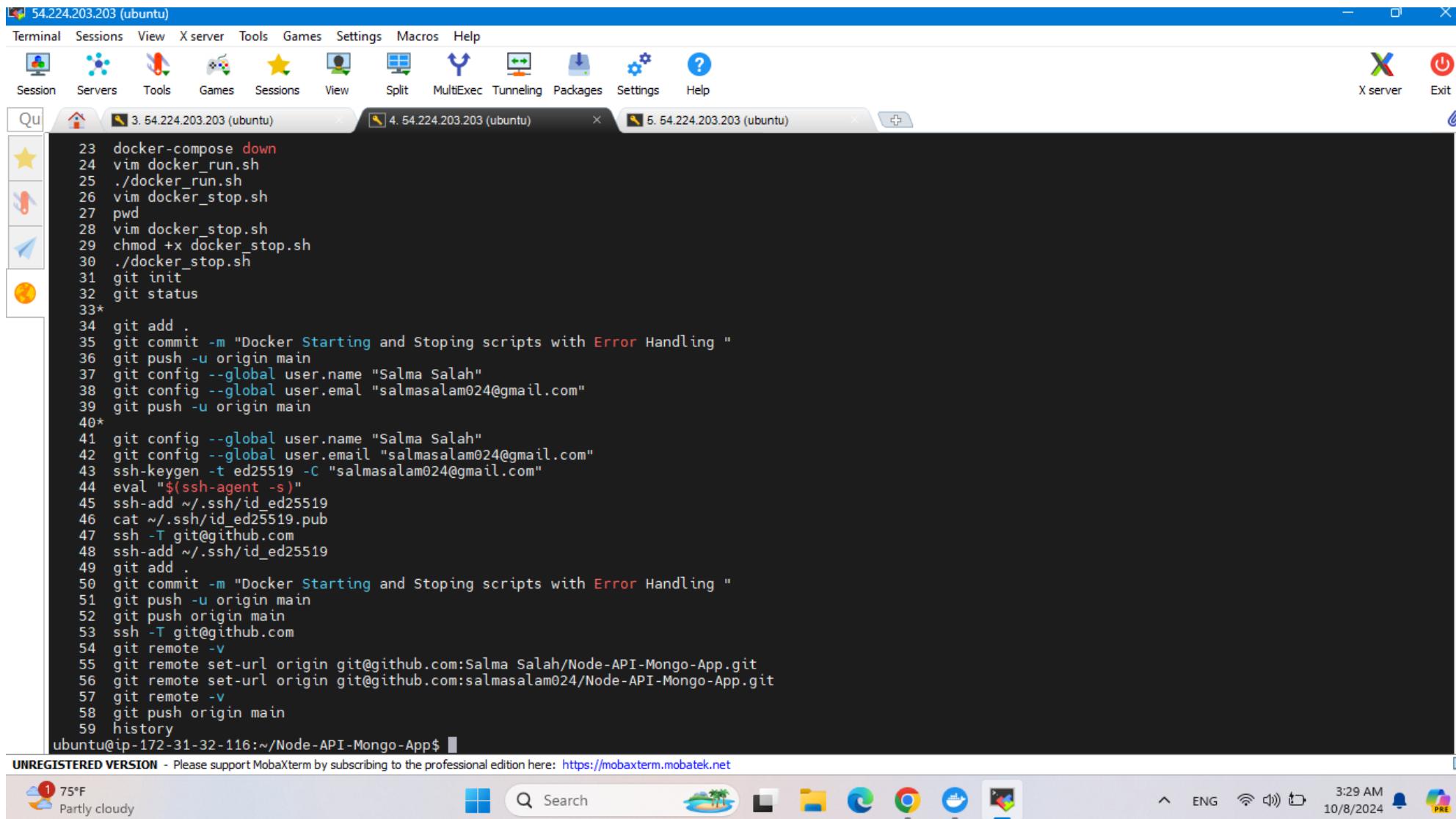
```
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ ssh-add ~/.ssh/id_ed25519
Identity added: /home/ubuntu/.ssh/id_ed25519 (salmasalam024@gmail.com)
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEMpfFQzDgclRglTB1CNT18JBisDirT32ZG/F1zRWCK salmasalam024@gmail.com
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ ssh -T git@github.com
The authenticity of host 'github.com (140.82.114.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMsvHdkr4UvC0qu.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Hi salmasalam024! You've successfully authenticated, but GitHub does not provide shell access.
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ ssh-add ~/.ssh/id_ed25519
Identity added: /home/ubuntu/.ssh/id_ed25519 (salmasalam024@gmail.com)
```

At the bottom of the terminal window, there is a message: UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

The taskbar at the bottom of the screen shows the following icons and information:

  - Cloud icon with '1' and '75°F' (Partly cloudy)
  - Search bar
  - File Explorer
  - Code Editor
  - Google Chrome
  - Terminal
  - MobaXterm icon
  - System tray with battery, signal, and date/time (3:17 AM, 10/8/2024)

# 19- Successfully connecting to github repo and push script



The screenshot shows a MobaXterm window titled "54.224.203.203 (ubuntu)". The terminal session contains the following command history:

```
23 docker-compose down
24 vim docker_run.sh
25 ./docker_run.sh
26 vim docker_stop.sh
27 pwd
28 vim docker_stop.sh
29 chmod +x docker_stop.sh
30 ./docker_stop.sh
31 git init
32 git status
33*
34 git add .
35 git commit -m "Docker Starting and Stoping scripts with Error Handling"
36 git push -u origin main
37 git config --global user.name "Salma Salah"
38 git config --global user.emal "salmasalam024@gmail.com"
39 git push -u origin main
40*
41 git config --global user.name "Salma Salah"
42 git config --global user.email "salmasalam024@gmail.com"
43 ssh-keygen -t ed25519 -C "salmasalam024@gmail.com"
44 eval "$(ssh-agent -s)"
45 ssh-add ~/.ssh/id_ed25519
46 cat ~/.ssh/id_ed25519.pub
47 ssh -T git@github.com
48 ssh-add ~/.ssh/id_ed25519
49 git add .
50 git commit -m "Docker Starting and Stoping scripts with Error Handling"
51 git push -u origin main
52 git push origin main
53 ssh -T git@github.com
54 git remote -v
55 git remote set-url origin git@github.com:Salma Salah/Node-API-Mongo-App.git
56 git remote set-url origin git@github.com:salmasalam024/Node-API-Mongo-App.git
57 git remote -v
58 git push origin main
59 history
```

The command "git push -u origin main" was run at line 36, and "git push origin main" was run again at line 52. The command "git remote -v" was run at line 54, and "git remote -v" was run again at line 57.

At the bottom of the terminal, the prompt "ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App\$" is visible. The status bar at the bottom of the window displays "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

# 20- Creating Destination Server on AWS (Ubuntu) and open required ports

The screenshot shows the AWS EC2 Instances details page for an instance named "Destination\_Server".

**Instance summary for i-05f4ccfa52c8ef2c1 (Destination\_Server)**

Updated less than a minute ago

Attribute	Value	Details
Instance ID	i-05f4ccfa52c8ef2c1 (Destination_Server)	Public IPv4 address: 54.174.210.57   open address
IPv6 address	-	Instance state: Running
Hostname type	IP name: ip-172-31-40-204.ec2.internal	Private IP DNS name (IPv4 only): ip-172-31-40-204.ec2.internal
Answer private resource DNS name	IPv4 (A)	Instance type: t2.micro
Auto-assigned IP address	It is taking a bit longer than usual to fetch your data	VPC ID: vpc-07e368fb8901161b4
IAM Role	-	Subnet ID: subnet-0b61a6f56cd003f5
IMDSv2	Required	Instance ARN: arn:aws:ec2:us-east-1:123456789012:instance/i-05f4ccfa52c8ef2c1

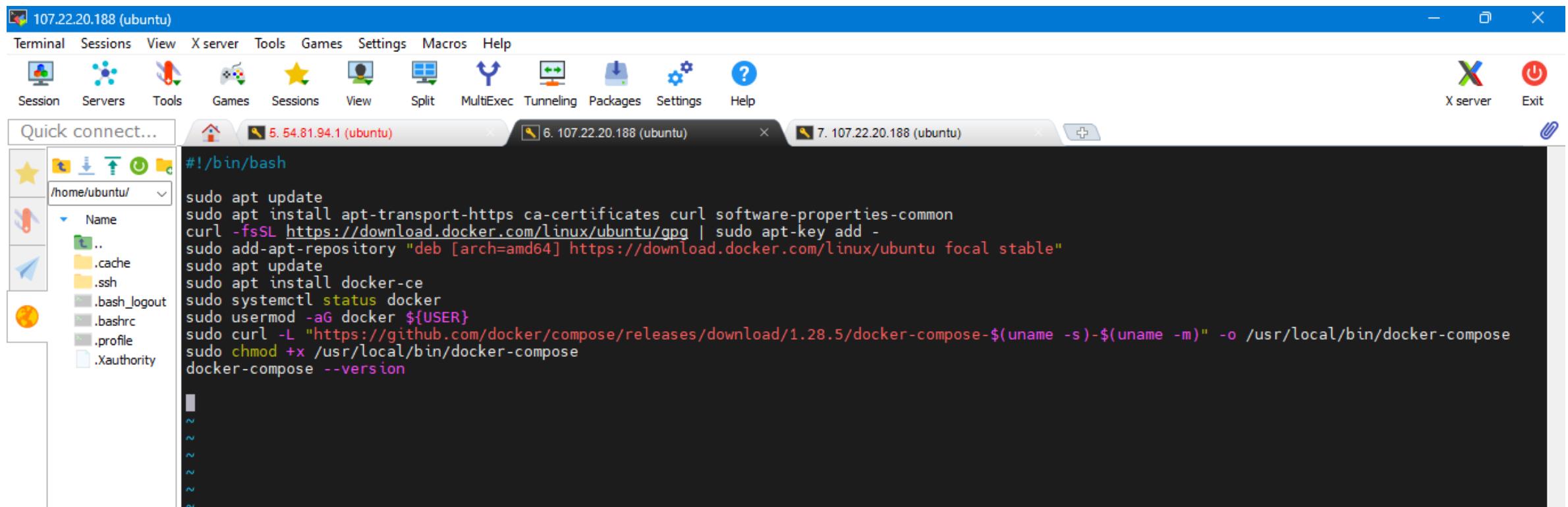
**Actions**: C, Connect, Instance state, Actions

**EC2 Dashboard**, **EC2 Global View**, **Events**, **Instances** (Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), **Images** (AMIs, AMI Catalog), **Elastic Block Store** (Volumes, Snapshots).

CloudShell, Feedback, © 2024, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, Cookie preferences.

29°C, 10:06 PM, 10/12/2024.

# 21- Writing bash script to automate installation if docker and docker-compose on destination server



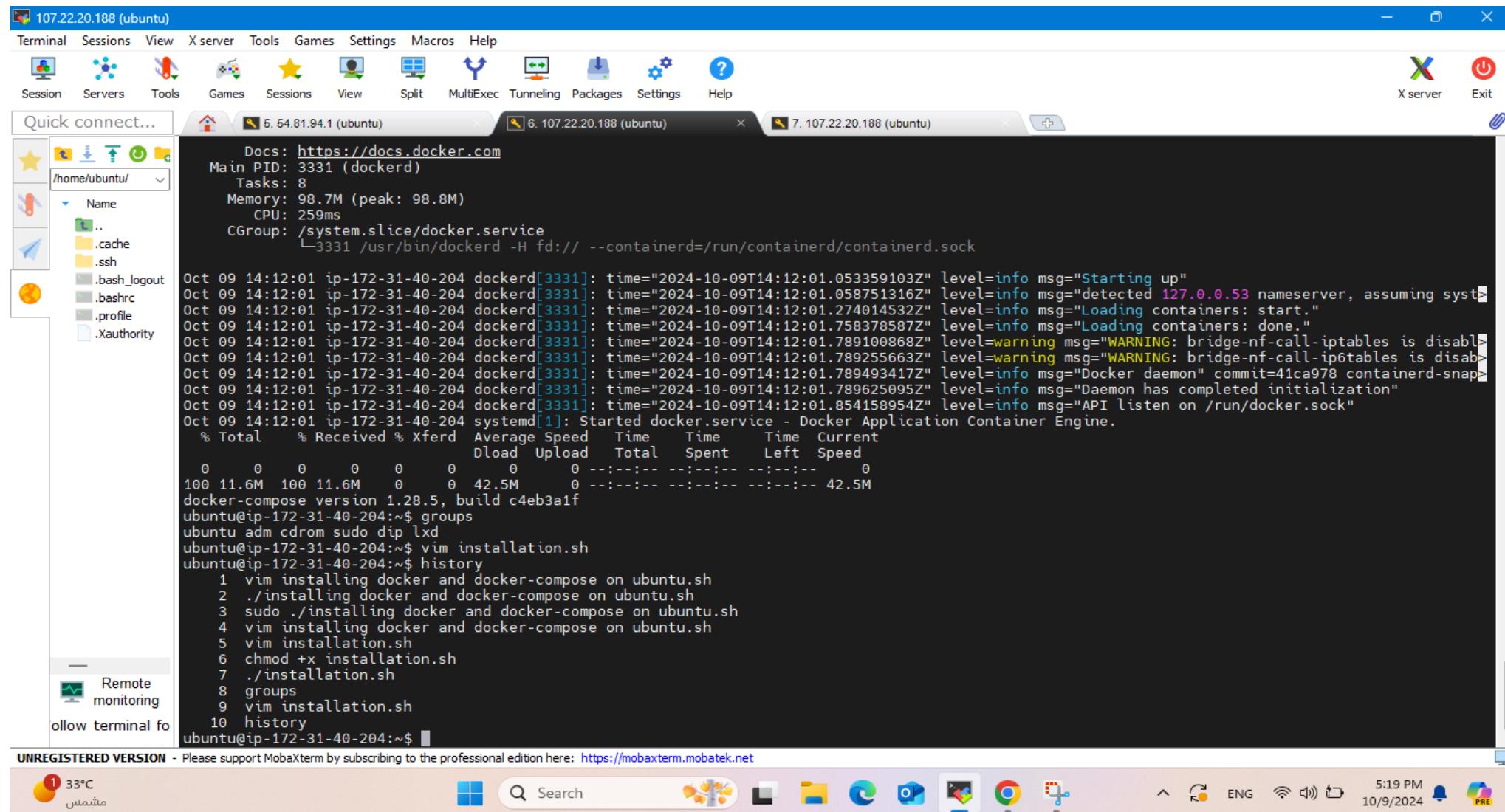
The screenshot shows a desktop environment with multiple windows open. The main window is a terminal titled "107.22.20.188 (ubuntu)". The terminal contains the following bash script:

```
#!/bin/bash

sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
sudo apt update
sudo apt install docker-ce
sudo systemctl status docker
sudo usermod -aG docker ${USER}
sudo curl -L "https://github.com/docker/compose/releases/download/1.28.5/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

The terminal window has a blue header bar with the title "107.22.20.188 (ubuntu)". Below the title is a menu bar with options: Terminal, Sessions, View, X server, Tools, Games, Settings, Macros, Help. To the right of the menu bar are icons for Session, Servers, Tools, Games, Sessions, View, Split, MultiExec, Tunneling, Packages, Settings, and Help. Further right are icons for X server and Exit. Below the menu bar, there are four tabs labeled "Quick connect...", "5. 54.81.94.1 (ubuntu)", "6. 107.22.20.188 (ubuntu)", and "7. 107.22.20.188 (ubuntu)". On the left side of the terminal window, there is a file browser sidebar with a tree view showing a directory structure under "/home/ubuntu/". The tree view includes entries for Name, .., .cache, .ssh, .bash\_logout, .bashrc, .profile, and .xauthority.

## 22- Successfully installing docker and docker-compose using script on destination server



The screenshot shows a MobaXterm window with three tabs open:

- Tab 5: 54.81.94.1 (ubuntu)
- Tab 6: 6. 107.22.20.188 (ubuntu)
- Tab 7: 7. 107.22.20.188 (ubuntu)

The content of Tab 7 (107.22.20.188) is as follows:

```
Docs: https://docs.docker.com
Main PID: 3331 (dockerd)
Tasks: 8
Memory: 98.7M (peak: 98.8M)
CPU: 259ms
CGroup: /system.slice/docker.service
└─3331 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Oct 09 14:12:01 ip-172-31-40-204 dockerd[3331]: time="2024-10-09T14:12:01.053359103Z" level=info msg="Starting up"
Oct 09 14:12:01 ip-172-31-40-204 dockerd[3331]: time="2024-10-09T14:12:01.058751316Z" level=info msg="detected 127.0.0.53 nameserver, assuming systemd"
Oct 09 14:12:01 ip-172-31-40-204 dockerd[3331]: time="2024-10-09T14:12:01.274014532Z" level=info msg="Loading containers: start."
Oct 09 14:12:01 ip-172-31-40-204 dockerd[3331]: time="2024-10-09T14:12:01.758378587Z" level=info msg="Loading containers: done."
Oct 09 14:12:01 ip-172-31-40-204 dockerd[3331]: time="2024-10-09T14:12:01.789100868Z" level=warning msg="WARNING: bridge-nf-call-iptables is disabled"
Oct 09 14:12:01 ip-172-31-40-204 dockerd[3331]: time="2024-10-09T14:12:01.789255663Z" level=warning msg="WARNING: bridge-nf-call-ip6tables is disabled"
Oct 09 14:12:01 ip-172-31-40-204 dockerd[3331]: time="2024-10-09T14:12:01.789493417Z" level=info msg="Docker daemon" commit=41ca978 containerd-snapshots=0
Oct 09 14:12:01 ip-172-31-40-204 dockerd[3331]: time="2024-10-09T14:12:01.789625095Z" level=info msg="Daemon has completed initialization"
Oct 09 14:12:01 ip-172-31-40-204 dockerd[3331]: time="2024-10-09T14:12:01.854158954Z" level=info msg="API listen on /run/docker.sock"
Oct 09 14:12:01 ip-172-31-40-204 systemd[1]: Started docker.service - Docker Application Container Engine.

% Total    % Received % Xferd  Average Speed   Time     Time  Current
          Dload  Upload   Total Spent    Left  Speed
0       0      0      0      0      0      0      0      0      0      0
100 11.6M 100 11.6M 0      0  42.5M 0      0      0      0      42.5M
docker-compose version 1.28.5, build c4eb3a1f
ubuntu@ip-172-31-40-204:~$ groups
ubuntu adm cdrom sudo dip lxd
ubuntu@ip-172-31-40-204:~$ vim installation.sh
ubuntu@ip-172-31-40-204:~$ history
 1 vim installing docker and docker-compose on ubuntu.sh
 2 ./installing docker and docker-compose on ubuntu.sh
 3 sudo ./installing docker and docker-compose on ubuntu.sh
 4 vim installing docker and docker-compose on ubuntu.sh
 5 vim installation.sh
 6 chmod +x installation.sh
 7 ./installation.sh
 8 groups
 9 vim installation.sh
10 history
ubuntu@ip-172-31-40-204:~$
```

At the bottom of the terminal window, there is a message: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

# 23- Writing bash script to transfer application code to target server

The screenshot shows a MobaXterm window titled "54.81.94.1 (ubuntu)". The window contains a terminal session with the following bash script:

```
#!/bin/bash

# Variables
EC2_PUBLIC_IP="107.22.20.188" # Destination server's public IP
EC2_USER= "ubuntu" # user id
PEM_FILE="/home/ubuntu/.ssh/id_rsa" # Path to your AWS .pem key
LOCAL_PROJECT_DIR="/home/ubuntu/Node-API-Mongo-App" # Path to local project directory
REMOTE_PROJECT_DIR="/home/ubuntu/Node-API-Mongo-Copy" # Directory on the destination server where the project will be transferred

# Transfer code to Destination server
echo "Transferring code to EC2 instance..."
scp -i $PEM_FILE -r $LOCAL_PROJECT_DIR $EC2_USER@$EC2_PUBLIC_IP:$REMOTE_PROJECT_DIR
```

The terminal window has two tabs: "5. 54.81.94.1 (ubuntu)" and "8. 107.22.20.188 (ubuntu)". The status bar at the bottom right shows "23.0-1" and "All". The bottom taskbar includes icons for weather (24°C), search, file explorer, and various system applications.

# 24- Writing bash script to transfer application code to target server

The screenshot shows a MobaXterm window with two tabs: "5. 54.81.94.1 (ubuntu)" and "8. 107.22.20.188 (ubuntu)". The left tab displays a command-line session where the user is executing a deployment script. The right tab shows the progress of file transfers using SCP. The transferred files include various Node.js modules and configuration files.

```
ubuntu@ip-172-31-32-116:~$ chmod 600 ~/.ssh/id_rsa
ubuntu@ip-172-31-32-116:~$ cd Node-API-Mongo-App
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ vim Copy_Automate_Deployment.sh
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ ./Copy_Automate_Deployment.sh
./Copy_Automate_Deployment.sh: line 5: ubuntu: command not found
Transferring code to EC2 instance...
Copy_Automate_Deployment.sh
docker_stop.log
docker_run.log
main
HEAD
main
d2e77b695daa401054016b68ed07406b5f989a
62cb1ce8b8a95306280799350aa722c016f9ed
ee2f65be83ba3c38ed81a180ce18d380b11a11
9e6b074e029aba03512e16d1b56fc8dd47c086
pack-3895397ddf14d6f779f6a034b372dd0feb4eed.rev
pack-3895397ddf14d6f779f6a034b372dd0feb4eed.pack
pack-3895397ddf14d6f779f6a034b372dd0feb4eed.idx
3538ec4961e76a52955fa7abb5b09ec74aa00e
fc4b33996ceed41ddf00e727161a56d0012e13
index
config
exclude
main
HEAD
main
HEAD
packed-refs
description
HEAD
prepare-commit-msg.sample
pre-applypatch.sample
post-update.sample
push-to-checkout.sample
pre-merge-commit.sample
fsmonitor-watchman.sample
```

100% 1126 953.5KB/s 00:00  
100% 580 293.2KB/s 00:00  
100% 2532 1.5MB/s 00:00  
100% 41 15.5KB/s 00:00  
100% 30 10.8KB/s 00:00  
100% 41 14.5KB/s 00:00  
100% 201 132.5KB/s 00:00  
100% 345 175.8KB/s 00:00  
100% 912 355.9KB/s 00:00  
100% 518 232.7KB/s 00:00  
100% 228 128.4KB/s 00:00  
100% 374KB 30.2MB/s 00:00  
100% 2304 1.0MB/s 00:00  
100% 138 60.5KB/s 00:00  
100% 397 197.1KB/s 00:00  
100% 3753 982.3KB/s 00:00  
100% 272 122.5KB/s 00:00  
100% 240 149.0KB/s 00:00  
100% 152 109.2KB/s 00:00  
100% 213 177.9KB/s 00:00  
100% 422 372.0KB/s 00:00  
100% 422 252.9KB/s 00:00  
100% 112 125.0KB/s 00:00  
100% 73 59.8KB/s 00:00  
100% 21 17.5KB/s 00:00  
100% 1492 765.3KB/s 00:00  
100% 424 154.9KB/s 00:00  
100% 189 70.9KB/s 00:00  
100% 2783 1.7MB/s 00:00  
100% 416 279.6KB/s 00:00  
100% 4726 3.0MB/s 00:00

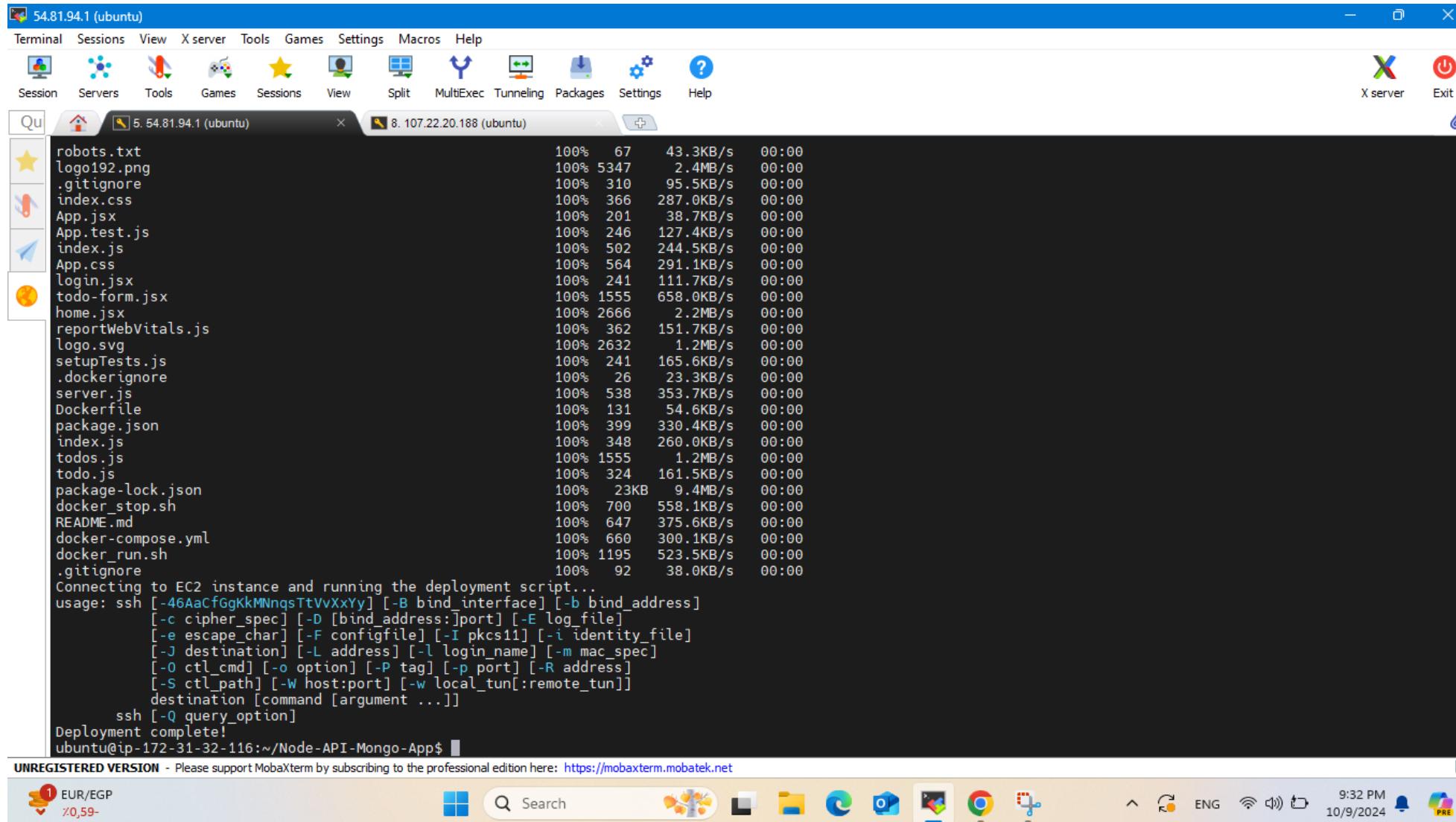
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

1 EUR/EGP  
X0,59-

Search

9:30 PM  
10/9/2024

# 25- Application code is successfully copied to target server



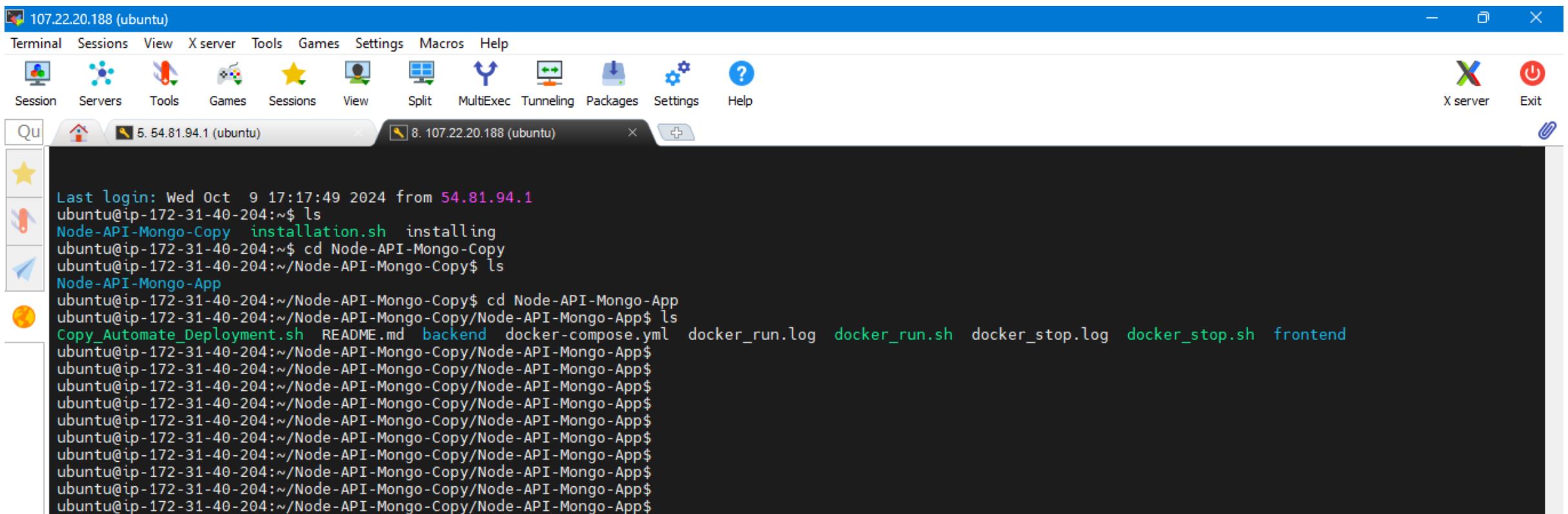
The screenshot shows a MobaXterm window titled "54.81.94.1 (ubuntu)". The interface includes a top menu bar with "Terminal", "Sessions", "View", "X server", "Tools", "Games", "Settings", "Macros", and "Help". Below the menu is a toolbar with icons for "Session", "Servers", "Tools", "Games", "Sessions", "View", "Split", "MultiExec", "Tunneling", "Packages", "Settings", and "Help". On the right side, there are "X server" and "Exit" buttons.

The main terminal window displays a file transfer log from "5. 54.81.94.1 (ubuntu)" to "8. 107.22.20.188 (ubuntu)". The log shows the progress of copying various files, including "robots.txt", "logo192.png", ".gitignore", "index.css", "App.jsx", "App.test.js", "index.js", "App.css", "login.jsx", "todo-form.jsx", "home.jsx", "reportWebVitals.js", "logo.svg", "setupTests.js", ".dockerignore", "server.js", "Dockerfile", "package.json", "index.js", "todos.js", "todo.js", "package-lock.json", "docker\_stop.sh", "README.md", "docker-compose.yml", "docker\_run.sh", and ".gitignore". The transfer speeds range from 38.0KB/s to 43.3KB/s.

```
robots.txt          100%   67    43.3KB/s  00:00
logo192.png        100%  5347   2.4MB/s  00:00
.gitignore          100%   310   95.5KB/s  00:00
index.css           100%   366   287.0KB/s  00:00
App.jsx             100%   201   38.7KB/s  00:00
App.test.js         100%   246   127.4KB/s  00:00
index.js            100%   502   244.5KB/s  00:00
App.css             100%   564   291.1KB/s  00:00
login.jsx           100%   241   111.7KB/s  00:00
todo-form.jsx       100%  1555   658.0KB/s  00:00
home.jsx            100%  2666   2.2MB/s  00:00
reportWebVitals.js 100%   362   151.7KB/s  00:00
logo.svg            100%  2632   1.2MB/s  00:00
setupTests.js       100%   241   165.6KB/s  00:00
.dockerignore        100%   26    23.3KB/s  00:00
server.js           100%   538   353.7KB/s  00:00
Dockerfile          100%   131   54.6KB/s  00:00
package.json         100%   399   330.4KB/s  00:00
index.js             100%   348   260.0KB/s  00:00
todos.js             100%  1555   1.2MB/s  00:00
todo.js              100%   324   161.5KB/s  00:00
package-lock.json    100%   23KB   9.4MB/s  00:00
docker_stop.sh       100%   700   558.1KB/s  00:00
README.md            100%   647   375.6KB/s  00:00
docker-compose.yml   100%   660   300.1KB/s  00:00
docker_run.sh         100%  1195   523.5KB/s  00:00
.gitignore           100%   92    38.0KB/s  00:00
Connecting to EC2 instance and running the deployment script...
usage: ssh [-46AaCfGgKkMNqsTtVvXxYy] [-B bind_interface] [-b bind_address]
           [-c cipher_spec] [-D [bind_address:]port] [-E log_file]
           [-e escape_char] [-F configfile] [-I pkcs11] [-i identity_file]
           [-J destination] [-L address] [-l login_name] [-m mac_spec]
           [-O ctl_cmd] [-o option] [-P tag] [-p port] [-R address]
           [-S ctl_path] [-W host:port] [-w local_tun[:remote_tun]]
           destination [command [argument ...]]
ssh [-Q query_option]
Deployment complete!
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$
```

At the bottom, a message reads "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>". The system tray at the very bottom shows icons for currency conversion (EUR/EGP), battery level (X0,59%), system status, network, volume, and date/time (9:32 PM, 10/9/2024).

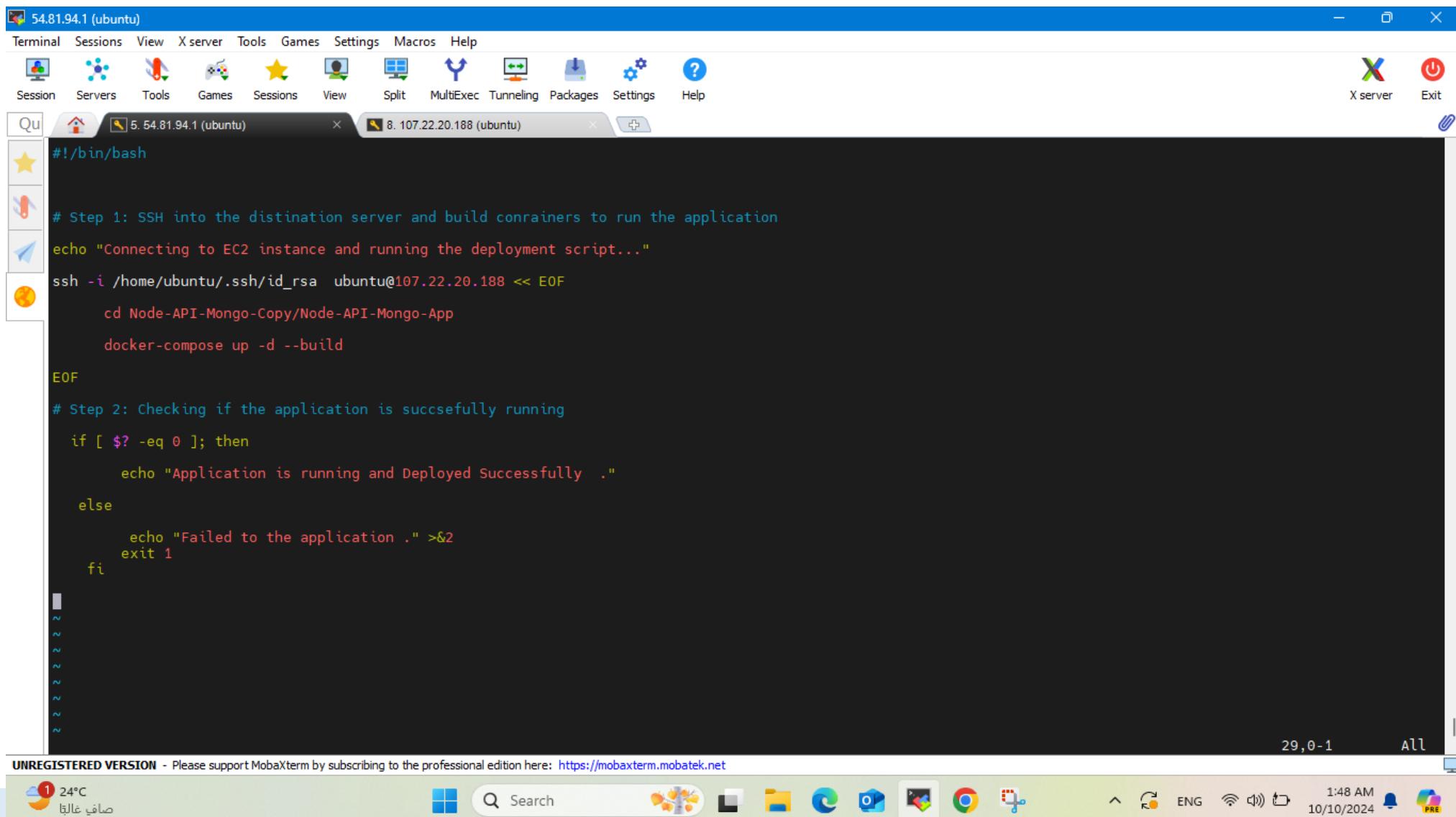
# 26- Application files in destination server



The screenshot shows a Linux desktop environment with a blue header bar. The title bar reads "107.22.20.188 (ubuntu)". The menu bar includes "Terminal", "Sessions", "View", "X server", "Tools", "Games", "Settings", "Macros", and "Help". Below the menu is a toolbar with icons for Session, Servers, Tools, Games, Sessions, View, Split, MultiExec, Tunneling, Packages, Settings, and Help. On the right side of the header are "X server" and "Exit" buttons. The main window contains a terminal session titled "Qu". The terminal output shows the following command history:

```
Last login: Wed Oct  9 17:17:49 2024 from 54.81.94.1
ubuntu@ip-172-31-40-204:~$ ls
Node-API-Mongo-Copy  installation.sh  installing
ubuntu@ip-172-31-40-204:~$ cd Node-API-Mongo-Copy
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy$ ls
Node-API-Mongo-App
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy$ cd Node-API-Mongo-App
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App$ ls
Copy_Automate_Deployment.sh  README.md  backend  docker-compose.yml  docker_run.log  docker_run.sh  docker_stop.log  docker_stop.sh  frontend
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App$
```

## 27- Writing bash script to ssh from main server to destination server and run the application using(docker-compose up – build –d )



The screenshot shows a terminal window titled "54.81.94.1 (ubuntu)" in MobaXterm. The window contains a bash script for deploying an application. The script uses SSH to connect to a destination server (IP 8.107.22.20.188) and runs a deployment script. It then checks if the application is running successfully. If it is, it prints a success message; if not, it prints a failure message and exits with code 1.

```
#!/bin/bash

# Step 1: SSH into the destination server and build containers to run the application
echo "Connecting to EC2 instance and running the deployment script..."
ssh -i /home/ubuntu/.ssh/id_rsa ubuntu@107.22.20.188 << EOF
    cd Node-API-Mongo-Copy/Node-API-Mongo-App
    docker-compose up -d --build
EOF

# Step 2: Checking if the application is successfully running
if [ $? -eq 0 ]; then
    echo "Application is running and Deployed Successfully ."
else
    echo "Failed to the application ." >&2
    exit 1
fi
```

At the bottom of the terminal, there is a note: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>". The system tray at the bottom right shows the date and time as "29, 0-1 All 1:48 AM 10/10/2024".

# 28- Deploying application in destination server by executing the bash script in main server

The screenshot shows a terminal window in MobaXterm with two tabs open:

- 54.81.94.1 (ubuntu)**: This tab displays the deployment command being run on the destination server. The output shows the user is connecting to an EC2 instance and running a deployment script. It also provides system information for Ubuntu 24.04.1 LTS.
- 8.107.22.20.188 (ubuntu)**: This tab is currently active and shows the deployment process. The user runs `./SSH_Build_Deploy. ssh`, which connects to the EC2 instance and runs the deployment script. The terminal then displays system information for the destination server, including load average, memory usage, and swap usage. It also checks for security updates and enables ESM Apps. Finally, it creates a network named "node-api-mongo-app\_network-backend" using Docker.

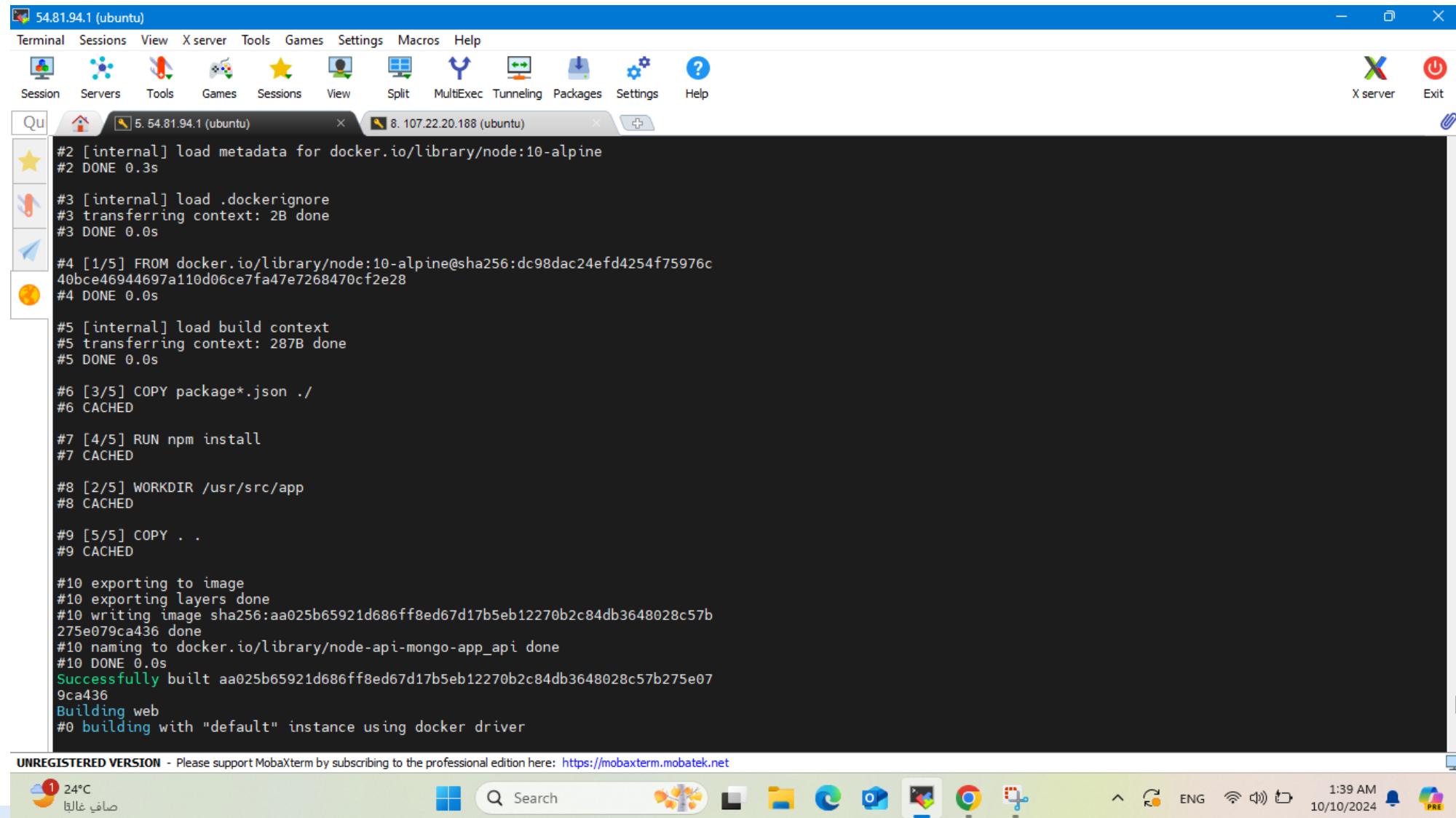
**UNREGISTERED VERSION** - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

1 24°C صافي غالباً

Search

1:38 AM ENG 10/10/2024 PRE

# 28- Deploying application in destination server by executing the bash script in main server



The screenshot shows a MobaXterm window titled "54.81.94.1 (ubuntu)". The terminal session displays the following Docker build logs:

```
#2 [internal] load metadata for docker.io/library/node:10-alpine
#2 DONE 0.3s

#3 [internal] load .dockerrcignore
#3 transferring context: 2B done
#3 DONE 0.0s

#4 [1/5] FROM docker.io/library/node:10-alpine@sha256:dc98dac24efd4254f75976c
40bce46944697a110d06ce7fa47e7268470cf2e28
#4 DONE 0.0s

#5 [internal] load build context
#5 transferring context: 287B done
#5 DONE 0.0s

#6 [3/5] COPY package*.json .
#6 CACHED

#7 [4/5] RUN npm install
#7 CACHED

#8 [2/5] WORKDIR /usr/src/app
#8 CACHED

#9 [5/5] COPY . .
#9 CACHED

#10 exporting to image
#10 exporting layers done
#10 writing image sha256:aa025b65921d686ff8ed67d17b5eb12270b2c84db3648028c57b
275e079ca436 done
#10 naming to docker.io/library/node-api-mongo-app_api done
#10 DONE 0.0s
Successfully built aa025b65921d686ff8ed67d17b5eb12270b2c84db3648028c57b275e07
9ca436
Building web
#0 building with "default" instance using docker driver
```

At the bottom of the terminal window, a message reads: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

28- Deploying application in destination server by executing the bash script in main server

The screenshot shows a MobaXterm window titled "5. 54.81.94.1 (ubuntu)". The terminal output displays the following Docker build logs:

```
Successfully built aa025b65921d686ff8ed67d17b5eb12270b2c84db3648028c57b275e07
9ca436
Building web
#0 building with "default" instance using docker driver

#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 320B done
#1 DONE 0.0s

#2 [internal] load metadata for docker.io/library/node:14-alpine
#2 DONE 0.1s

#3 [internal] load .dockerignore
#3 transferring context: 123B done
#3 DONE 0.0s

#4 [1/5] FROM docker.io/library/node:14-alpine@sha256:434215b487a329c9e867202
ff89e704d3a75e554822e07f3e0c0f9e606121b33
#4 DONE 0.0s

#5 [internal] load build context
#5 transferring context: 914B done
#5 DONE 0.0s

#6 [2/5] WORKDIR /app
#6 CACHED

#7 [3/5] COPY package*.json .
#7 CACHED

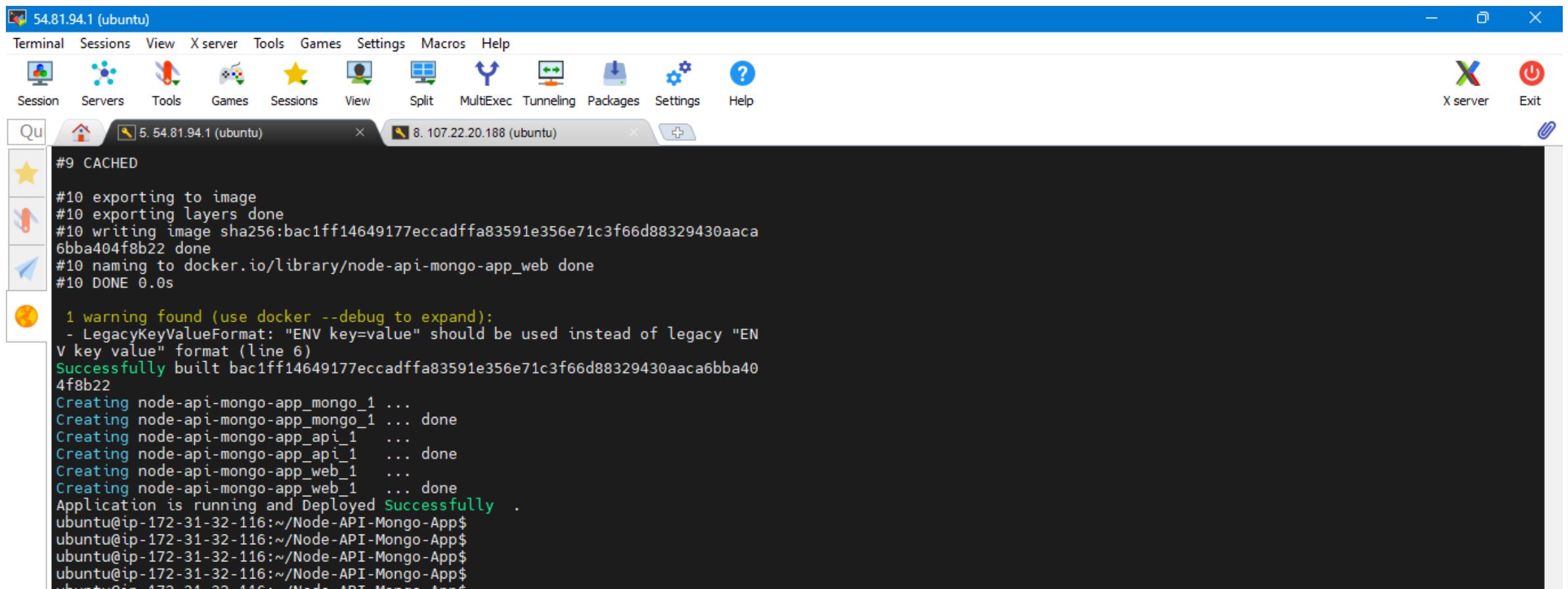
#8 [4/5] RUN npm install
#8 CACHED

#9 [5/5] COPY . .
#9 CACHED

#10 exporting to image
#10 exporting layers done
```

At the bottom of the terminal window, there is a message: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

# 29- Application successfully deployed in main server and the three containers are running by executing run bash script from main server



The screenshot shows a desktop environment with a terminal window open. The terminal window title is "54.81.94.1 (ubuntu)". The terminal content displays the output of a Docker build command:

```
#9 CACHED
#10 exporting to image
#10 exporting layers done
#10 writing image sha256:bac1ff14649177eccadffa83591e356e71c3f66d88329430aaca
6bba404f8b22 done
#10 naming to docker.io/library/node-api-mongo-app_web done
#10 DONE 0.0s

1 warning found (use docker --debug to expand):
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "EN
V key value" format (line 6)
Successfully built bac1ff14649177eccadffa83591e356e71c3f66d88329430aaca6bba40
4f8b22
Creating node-api-mongo-app_mongo_1 ...
Creating node-api-mongo-app_mongo_1 ... done
Creating node-api-mongo-app_api_1 ...
Creating node-api-mongo-app_api_1 ... done
Creating node-api-mongo-app_web_1 ...
Creating node-api-mongo-app_web_1 ... done
Application is running and Deployed Successfully .
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$
```

# 30- Successfully serving the application from destination server

The screenshot shows a browser window titled "React App" displaying a "Todos" application. The URL in the address bar is "Not secure 107.22.20.188:3000". The page lists two items:

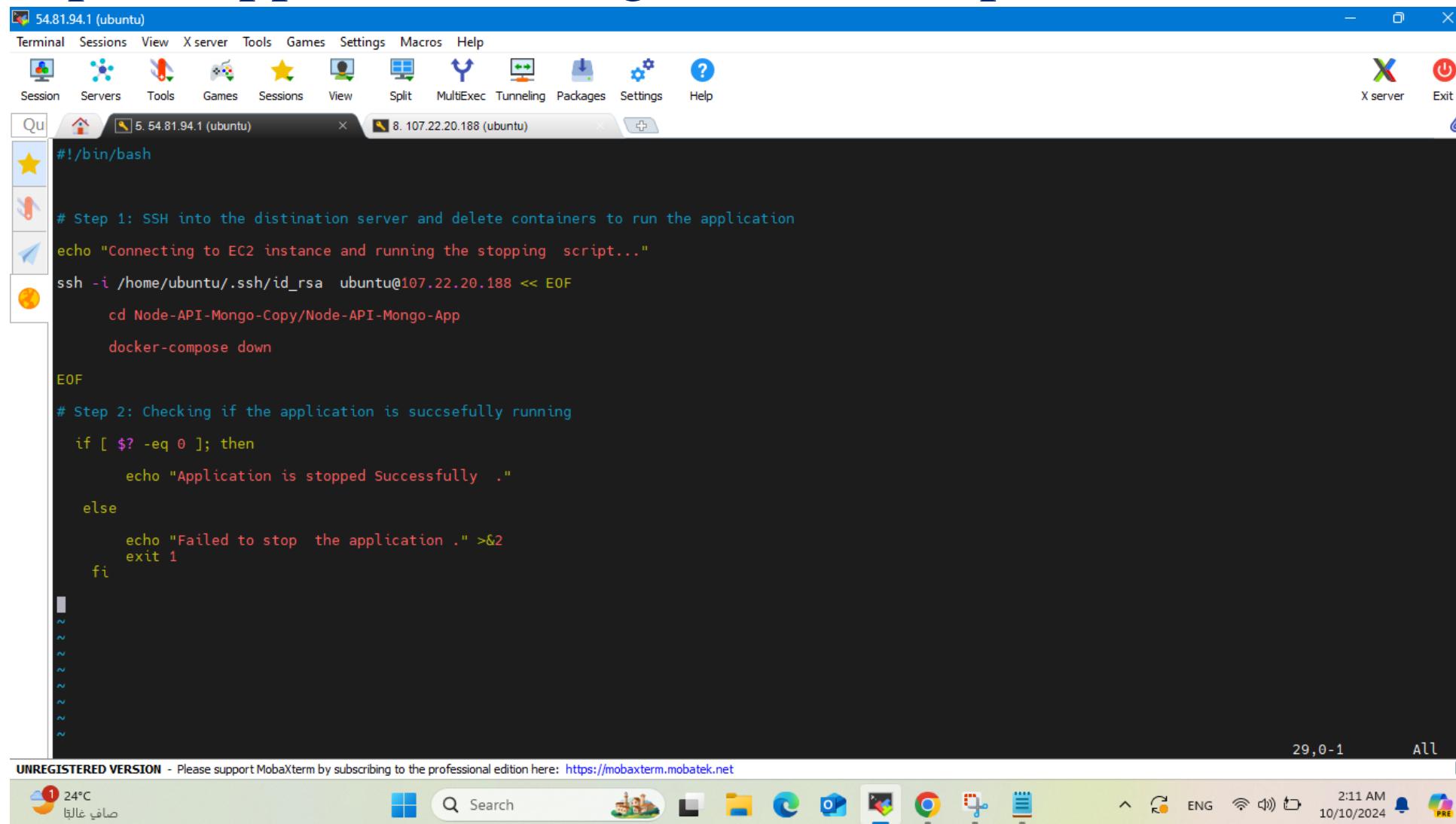
- Target Server on AWS** Due: 2024-10-10T00:00:00.000Z  
This is the application served from target server after copying the files from the main server using scp
- Steps for Deployment** Due: 2024-10-09T21:09:24.093Z  
1. Run the first script to copy application files to target server. 2. Run the second script to: 1) ssh into destination server 2) run docker-compose up to run application 3. Run the third script to send an HTTP request to the EC2 public IP to ensure the application is running and returning a 200 OK status. 4. Run the fourth script to stop all containers

A blue "Add Todo" button is located at the bottom left of the list.

At the bottom of the screen, a Windows taskbar is visible with various icons and information:

- Weather: 25°C صافي غالباً
- Search bar
- Icons for File Explorer, Edge, OneDrive, Paint, and Google Chrome
- Language: ENG
- Network: Wi-Fi signal
- Date and Time: 12:12 AM 10/10/2024
- Volume and Battery icons
- Pins icon

# 31- Writing bash script to ssh from main server to destination server and stop the application using(docker-compose down )



The screenshot shows a MobaXterm window titled "54.81.94.1 (ubuntu)". The terminal session contains the following bash script:

```
#!/bin/bash

# Step 1: SSH into the destination server and delete containers to run the application
echo "Connecting to EC2 instance and running the stopping script..."
ssh -i /home/ubuntu/.ssh/id_rsa ubuntu@107.22.20.188 << EOF
cd Node-API-Mongo-Copy/Node-API-Mongo-App
docker-compose down
EOF

# Step 2: Checking if the application is successfully running
if [ $? -eq 0 ]; then
    echo "Application is stopped Successfully ."
else
    echo "Failed to stop the application ." >&2
    exit 1
fi
```

The terminal window has two tabs: "54.81.94.1 (ubuntu)" and "8.107.22.20.188 (ubuntu)". The status bar at the bottom right shows "29,0-1 All". The taskbar at the bottom includes icons for File Explorer, Edge, and other system tools.

# 32- Stopping the application in destination server by executing the bash script in main server

The screenshot shows a MobaXterm window titled "54.81.94.1 (ubuntu)". The terminal session is connected to port 54.81.94.1 (ubuntu). The user has run a script named "Delete\_Stop-Containers.sh" which connects to an EC2 instance and runs a stopping script. The terminal output shows the script's execution and the stopping of several containers:

```
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ vim Delete_Stop-Containers.sh
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ chmod +x Delete_Stop-Containers.sh
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ ./Delete_Stop-Containers.sh
Connecting to EC2 instance and running the stopping script...
Pseudo-terminal will not be allocated because stdin is not a terminal.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

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

System information as of Wed Oct 9 23:11:28 UTC 2024

System load: 0.0      Processes: 130
Usage of /: 56.3% of 6.71GB  Users logged in: 1
Memory usage: 62%          IPv4 address for enX0: 172.31.40.204
Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.

8 updates can be applied immediately.
5 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

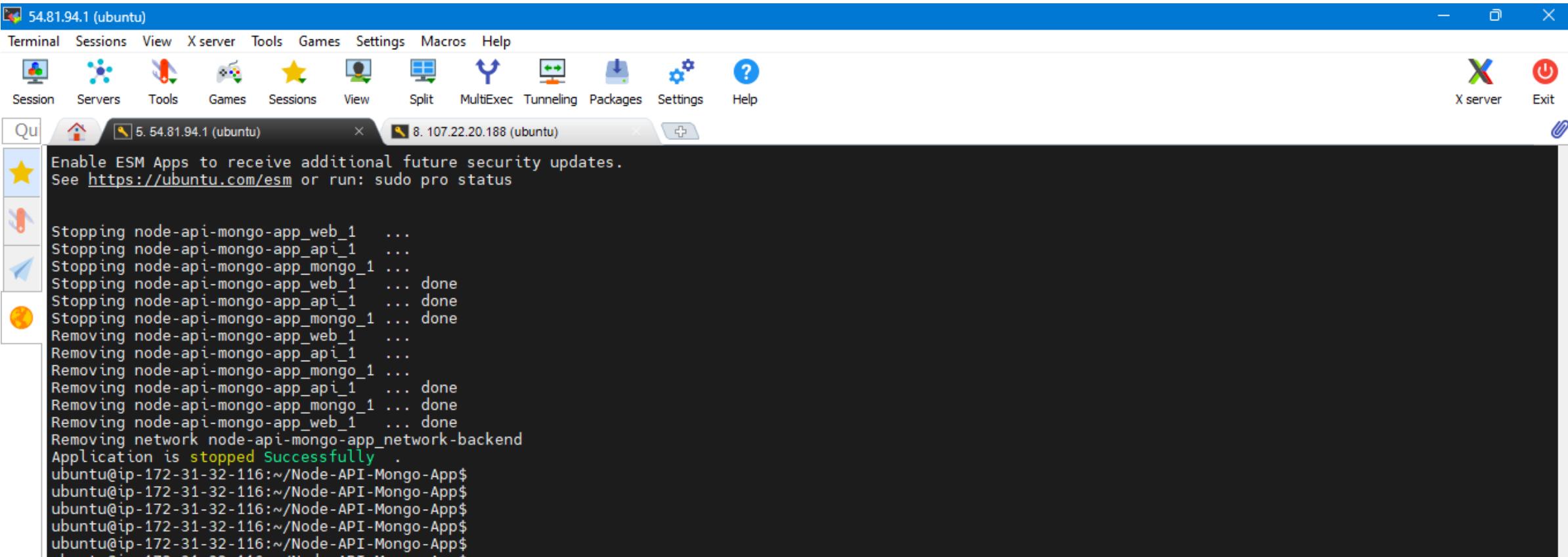
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Stopping node-api-mongo-app_web_1 ...
Stopping node-api-mongo-app_api_1 ...
Stopping node-api-mongo-app_mongo_1 ...
Stopping node-api-mongo-app_web_1 ... done
Stopping node-api-mongo-app_api_1 ... done
Stopping node-api-mongo-app_mongo_1 ... done
Removing node-api-mongo-app_web_1 ...
Removing node-api-mongo-app_api_1 ...
Removing node-api-mongo-app_mongo_1 ...
```

At the bottom of the terminal, there is a message: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

The taskbar at the bottom of the screen shows various icons for system monitoring and productivity tools.

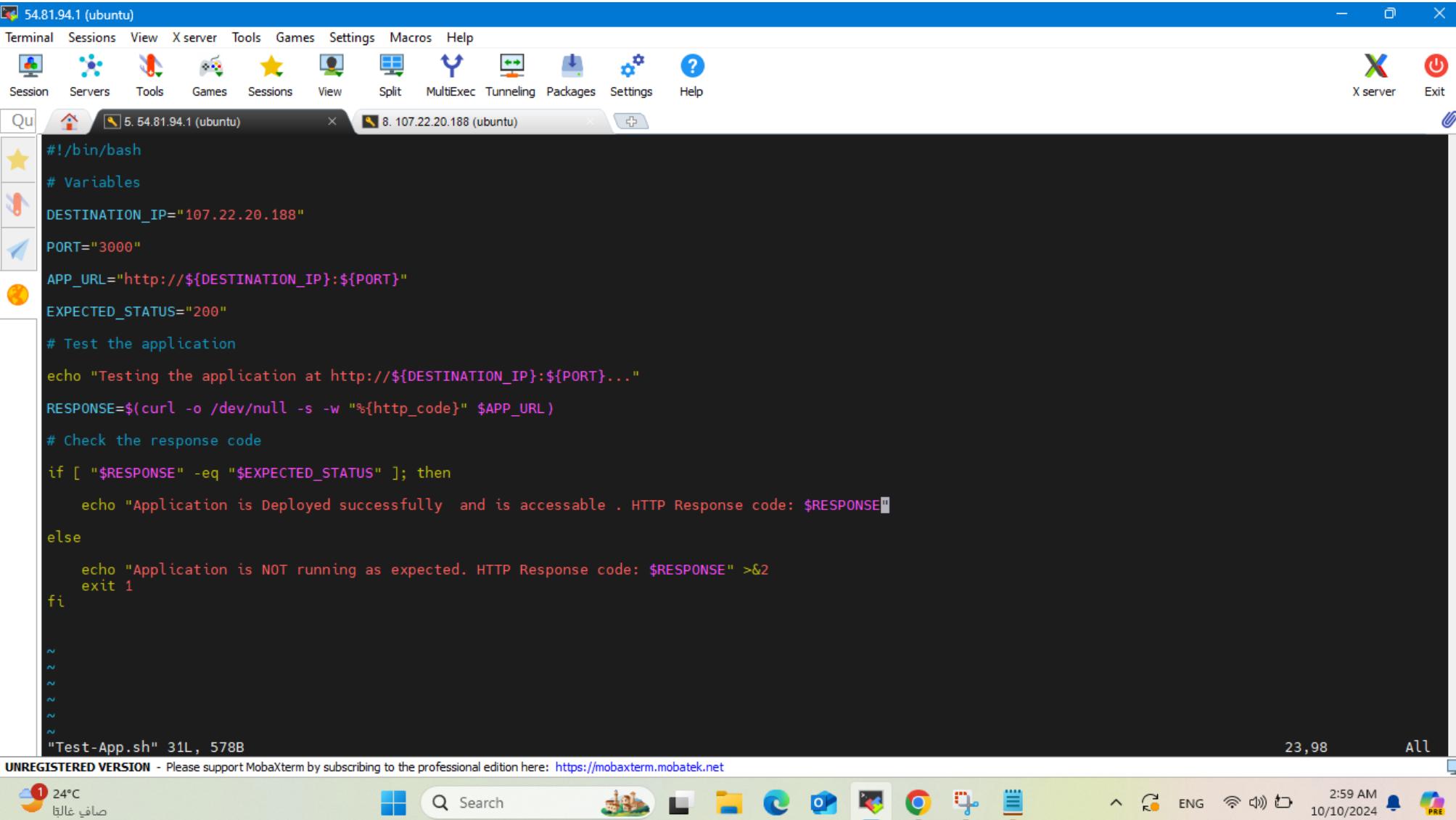
### 33- Application successfully stopped in destination server and the three containers are removed by executing stop bash script from main server



The screenshot shows a desktop interface with a terminal window open. The terminal window title is "5. 54.81.94.1 (ubuntu)". The terminal content displays the output of a command to stop and remove Docker containers:

```
Stopping node-api-mongo-app_web_1 ...
Stopping node-api-mongo-app_api_1 ...
Stopping node-api-mongo-app_mongo_1 ...
Stopping node-api-mongo-app_web_1 ... done
Stopping node-api-mongo-app_api_1 ... done
Stopping node-api-mongo-app_mongo_1 ... done
Removing node-api-mongo-app_web_1 ...
Removing node-api-mongo-app_api_1 ...
Removing node-api-mongo-app_mongo_1 ...
Removing node-api-mongo-app_web_1 ... done
Removing network node-api-mongo-app_network-backend
Application is stopped successfully .
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$
```

# 34- Writing bash script to test if the application has successfully deployed using (curl)



The screenshot shows a terminal window titled "54.81.94.1 (ubuntu)" in the MobaXterm interface. The window contains a bash script named "Test-App.sh". The script uses curl to check if an application is running at a specific IP and port, and then prints a message indicating whether it was successful or not.

```
#!/bin/bash
# Variables
DESTINATION_IP="107.22.20.188"
PORT="3000"
APP_URL="http://${DESTINATION_IP}:${PORT}"
EXPECTED_STATUS="200"

# Test the application
echo "Testing the application at http://${DESTINATION_IP}:${PORT}..."
RESPONSE=$(curl -o /dev/null -s -w "%{http_code}" $APP_URL)

# Check the response code
if [ "$RESPONSE" -eq "$EXPECTED_STATUS" ]; then
    echo "Application is Deployed successfully and is accessible . HTTP Response code: $RESPONSE"
else
    echo "Application is NOT running as expected. HTTP Response code: $RESPONSE" >&2
    exit 1
fi

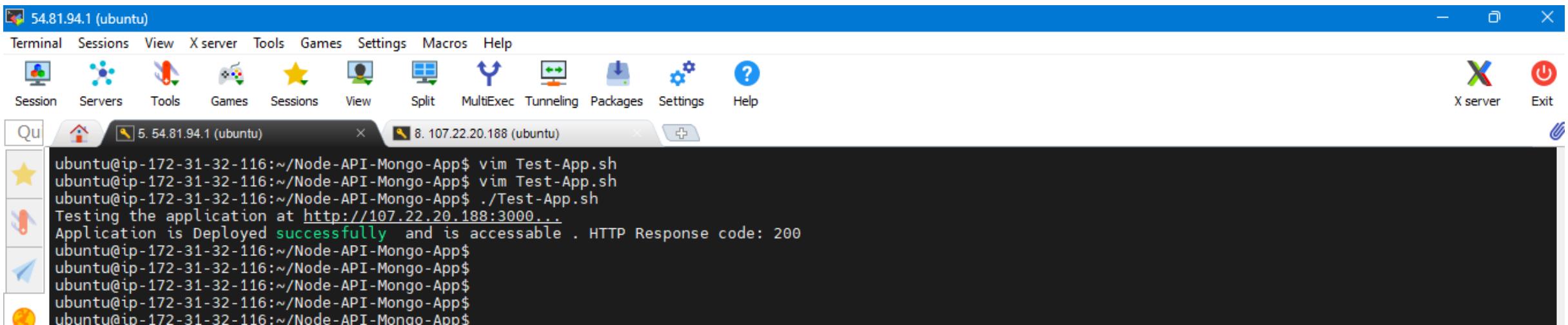
~
```

"Test-App.sh" 31L, 578B

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

24°C 2:59 AM 10/10/2024 ENG PRE

# 35- Successfully testing the application and get the expected response



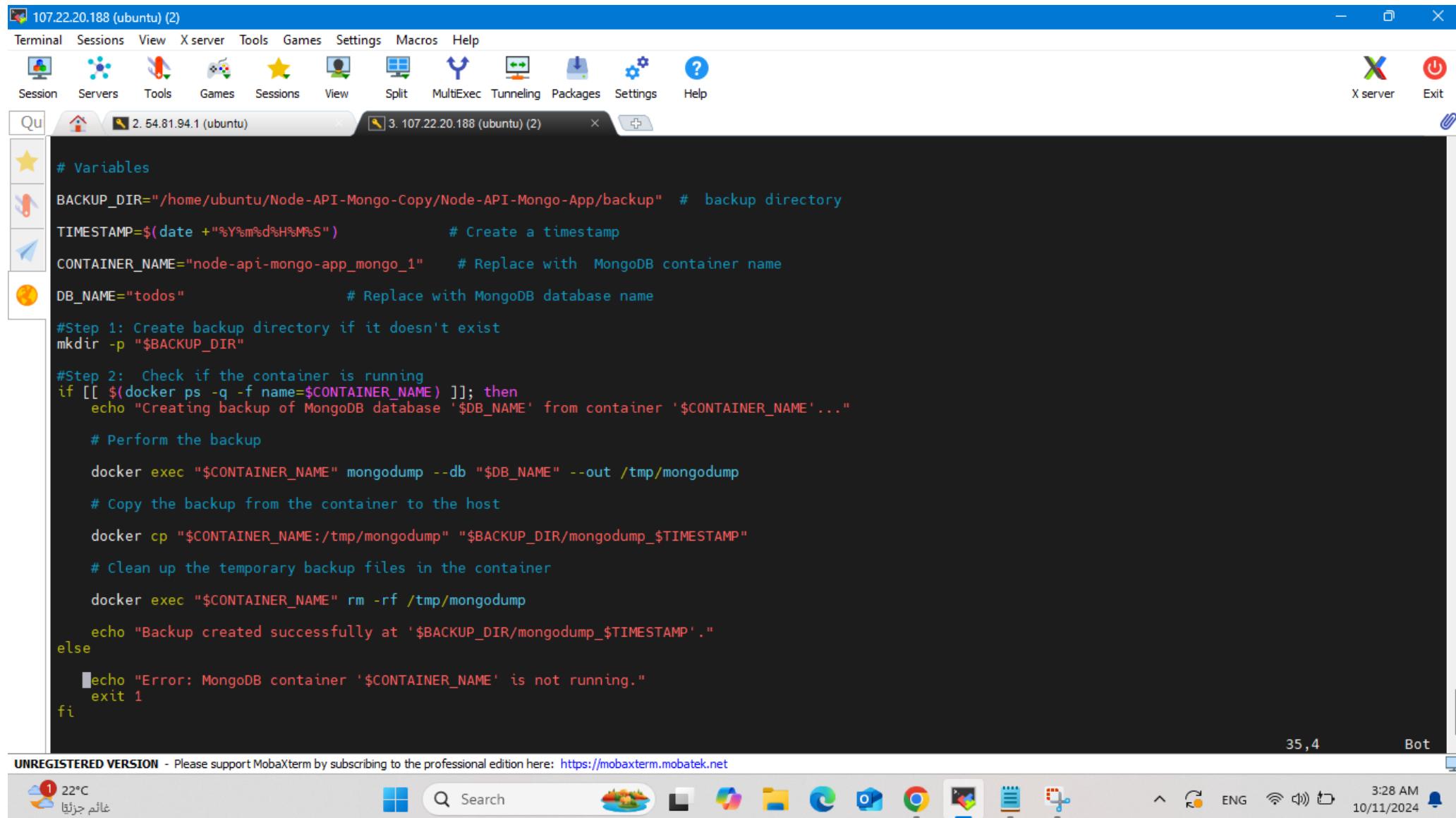
The screenshot shows a terminal window titled "54.81.94.1 (ubuntu)" with the following session details:

- Session: 5. 54.81.94.1 (ubuntu)
- Servers: 8. 107.22.20.188 (ubuntu)

The terminal output is as follows:

```
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ vim Test-App.sh
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ vim Test-App.sh
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$ ./Test-App.sh
Testing the application at http://107.22.20.188:3000...
Application is Deployed successfully and is accessible . HTTP Response code: 200
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$
ubuntu@ip-172-31-32-116:~/Node-API-Mongo-App$
```

# 36- Writing bash script to backup all application data



The screenshot shows a MobaXterm window titled "107.22.20.188 (ubuntu) (2)". The terminal window contains a bash script for backing up a MongoDB database from a Docker container. The script defines variables for the backup directory, timestamp, container name, and database name. It checks if the container is running, performs a mongodump, copies the backup to the host, and then removes temporary files from the container. If the container is not running, it exits with an error code.

```
# Variables
BACKUP_DIR="/home/ubuntu/Node-API-Mongo-Copy/Node-API-Mongo-App/backup" # backup directory
TIMESTAMP=$(date +"%Y%m%d%H%M%S") # Create a timestamp
CONTAINER_NAME="node-api-mongo-app_mongo_1" # Replace with MongoDB container name
DB_NAME="todos" # Replace with MongoDB database name

#Step 1: Create backup directory if it doesn't exist
mkdir -p "$BACKUP_DIR"

#Step 2: Check if the container is running
if [[ $(docker ps -q -f name=$CONTAINER_NAME) ]]; then
    echo "Creating backup of MongoDB database '$DB_NAME' from container '$CONTAINER_NAME'..."
    # Perform the backup
    docker exec "$CONTAINER_NAME" mongodump --db "$DB_NAME" --out /tmp/mongodump
    # Copy the backup from the container to the host
    docker cp "$CONTAINER_NAME:/tmp/mongodump" "$BACKUP_DIR/mongodump_$TIMESTAMP"
    # Clean up the temporary backup files in the container
    docker exec "$CONTAINER_NAME" rm -rf /tmp/mongodump
    echo "Backup created successfully at '$BACKUP_DIR/mongodump_$TIMESTAMP'."
else
    echo "Error: MongoDB container '$CONTAINER_NAME' is not running."
    exit 1
fi
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

22°C غامن جزئیا 3:28 AM 10/11/2024

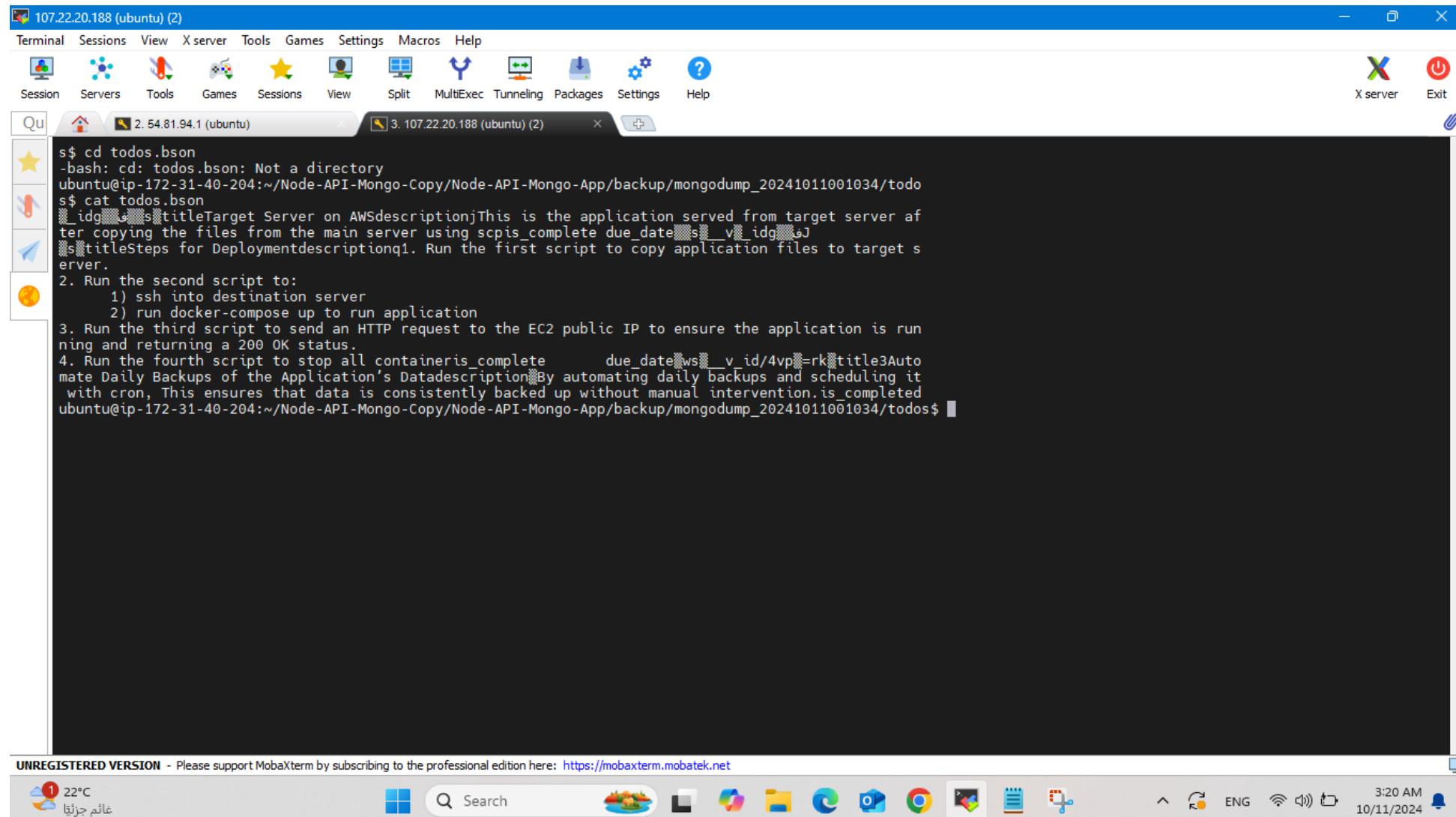
## 37- Testing and executing the backup script

The screenshot shows a MobaXterm window titled "107.22.20.188 (ubuntu) (2)". The terminal session on port 2 displays the following command output:

```
api_1
e7e951db2708    mongo          "docker-entrypoint.s..."  4 hours ago
Up 4 hours      0.0.0.0:27017->27017/tcp, :::27017->27017/tcp  node-api-mongo-app
_mongo_1
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App$ vim mongodb-b
ackup.vim
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App$ chmod +x mong
odb-backup.vim
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App$ ./mongodb-bac
kup.vim
Creating backup of MongoDB database 'todos' from container 'node-api-mongo-app_
mongo_1'...
2024-10-11T00:10:35.262+0000      writing todos.todos to /tmp/mongodump/todos/tod
os.bson
2024-10-11T00:10:35.266+0000      done dumping todos.todos (3 documents)
Successfully copied 4.61kB to /home/ubuntu/Node-API-Mongo-Copy/Node-API-Mongo-A
pp/backup/mongodump_20241011001034
Backup created successfully at '/home/ubuntu/Node-API-Mongo-Copy/Node-API-Mongo
-App/backup/mongodump_20241011001034'.
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App$ cd backup
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App/backup$ cd mon
godump_20241011001034'.
>
>
> ^C
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App/backup$ ls
mongodump_20241011001034
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App/backup$ cd mongodump_20241011001034
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App/backup/mongodump_20241011001034$ ls
todos
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App/backup/mongodump_20241011001034$ cd
todos
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App/backup/mongodump_20241011001034/todo
s$ ls
todos.bson  todos.metadata.json
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App/backup/mongodump_20241011001034/todo
s$ cd todos.bson
-bash: cd: todos.bson: Not a directory
```

The status bar at the bottom left shows "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>". The system tray icons include a cloud with "22°C", a battery icon, and a network signal icon.

# 38- Content of backup file indicating successfully execution of backup script

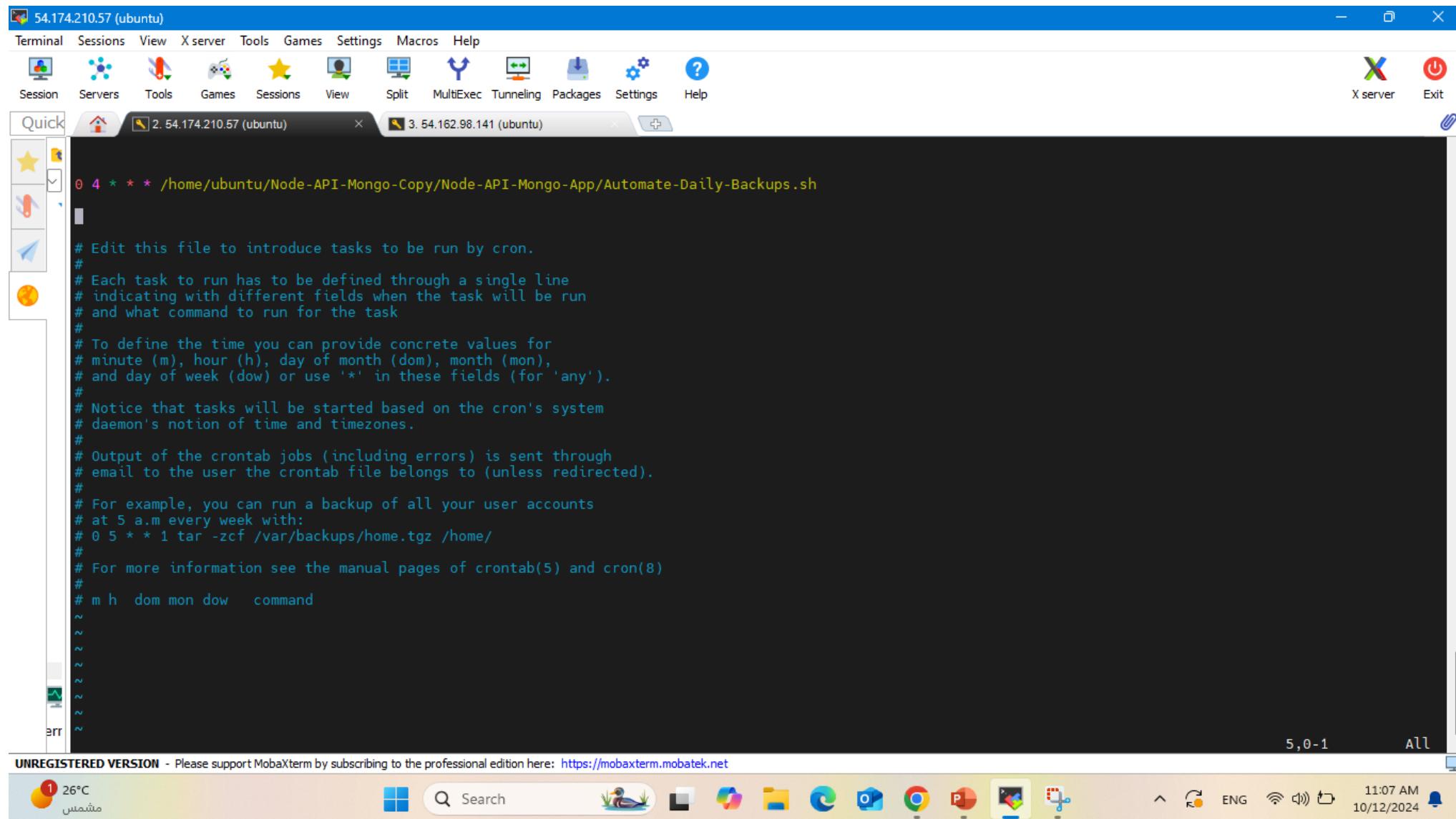


The screenshot shows a MobaXterm window titled "107.22.20.188 (ubuntu) (2)". The terminal session is running on port 107.22.20.188. The user has run the command "cat todos.bson" and the output is displayed:

```
s$ cd todos.bson
-bash: cd: todos.bson: Not a directory
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App/backup/mongodump_20241011001034/todo
s$ cat todos.bson
idg[titleTarget Server on AWSdescription]This is the application served from target server af
ter copying the files from the main server using scpis_complete due_date[ws] v_idg[titleSteps for Deploymentdescriptionq1. Run the first script to copy application files to target s
erver.
2. Run the second script to:
   1) ssh into destination server
   2) run docker-compose up to run application
3. Run the third script to send an HTTP request to the EC2 public IP to ensure the application is run
ning and returning a 200 OK status.
4. Run the fourth script to stop all containeris_complete      due_date[ws] v_id/4vp[title3Auto
mate Daily Backups of the Application's Data]description]By automating daily backups and scheduling it
with cron, This ensures that data is consistently backed up without manual intervention.is_completed
ubuntu@ip-172-31-40-204:~/Node-API-Mongo-Copy/Node-API-Mongo-App/backup/mongodump_20241011001034/todos$
```

At the bottom of the terminal window, there is a watermark: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

# 39- Create a cron job to automate daily backup at 4 am



The screenshot shows a MobaXterm window titled "54.174.210.57 (ubuntu)". The terminal tab displays a cron configuration script:

```
0 4 * * * /home/ubuntu/Node-API-Mongo-Copy/Node-API-Mongo-App/Automate-Daily-Backups.sh

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
~ ~ ~ ~ ~
```

The status bar at the bottom indicates an unregistered version and shows system information like temperature (26°C), battery level (1), search bar, and system icons.

## 40- Problems I faced and solved

- 1- Authentication with git I made ssh
- 2- API URL was local host and has to be changed to Public IP of aws ec2
- 3- Run ssh in interactive shell
- 4- Connection to mongo dB to make backup

## 41- Github Repository

<https://github.com/salmasalam024/Node-API-Mongo-App>