WireApps

Intern / Junior DevOps Engineer

Technical Assessment

Name: Dinuk Kaumika Ramawickrama

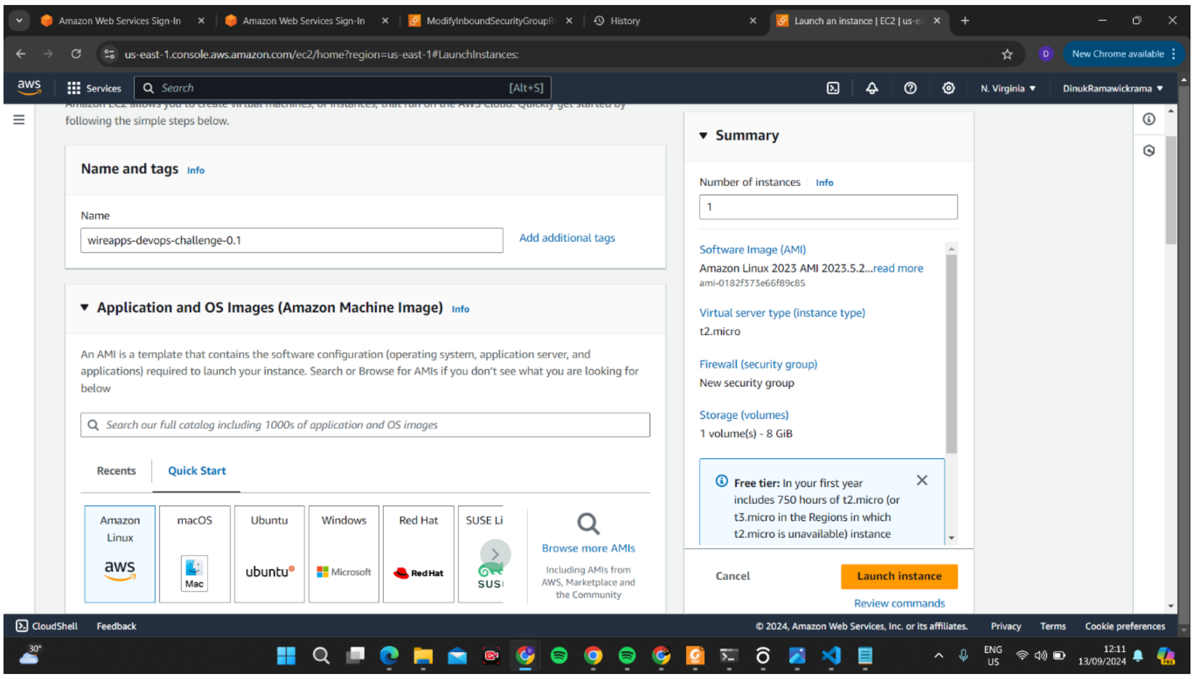
Date : 14/09/2024

Assignment Tasks

Step 1: Provision a Virtual Machine on cloud platform.

Provision EC2 Instance:

- I used AWS to create an EC2 instance with Ubuntu as the operating system.



us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:

following the simple steps below.

Name and tags Info

Name

wireapps-devops-challenge-0.1

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents | Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Li

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Summary

Number of instances Info

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.5.2...read more

ami-0182f373e66f89c85

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

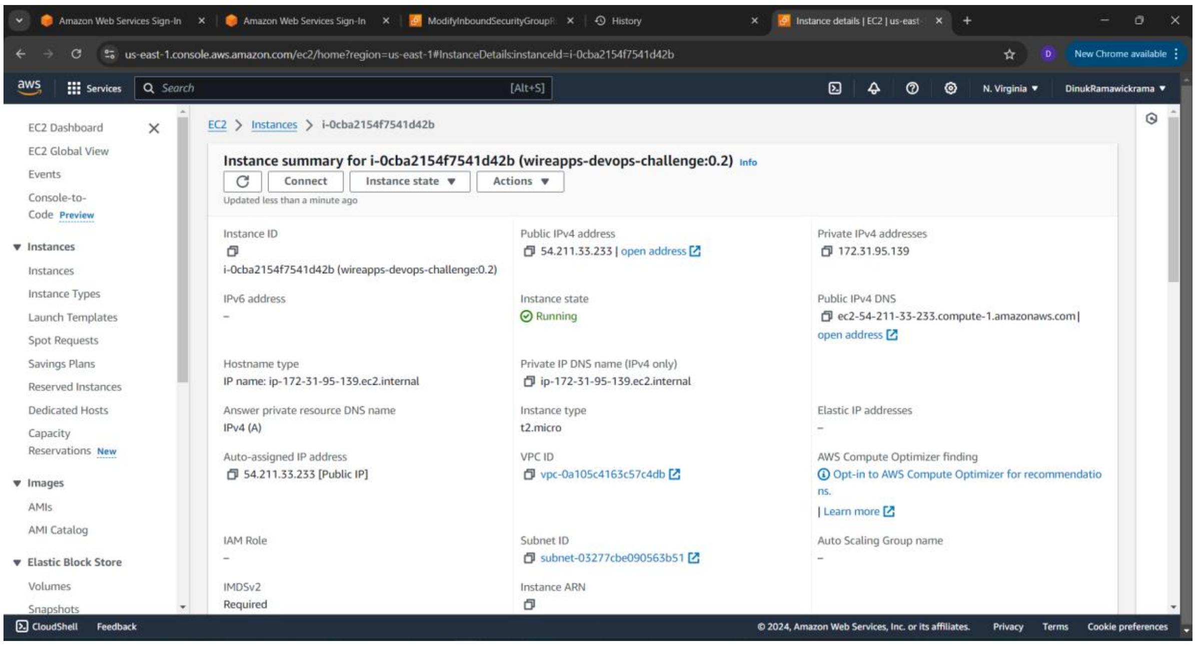
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance

Cancel | Launch instance

Review commands

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

- **Public IP**: Assigned a public IP to the instance for external access.


Amazon Web Services Sign-In | Amazon Web Services Sign-In | ModifyInboundSecurityGroupR | History | Instance details | EC2 | us-east-

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#InstanceDetails:instanceId=i-0cba2154f7541d42b

EC2 > Instances > i-0cba2154f7541d42b

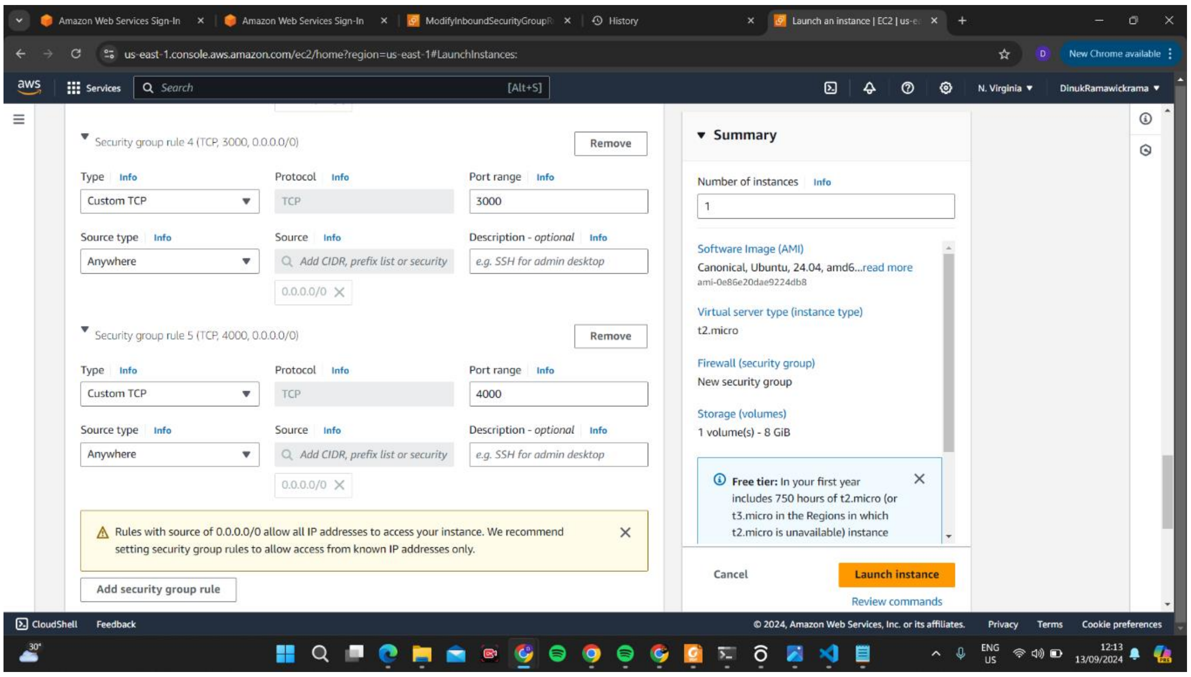
Instance summary for i-0cba2154f7541d42b (wireapps-devops-challenge:0.2) Info

Updated less than a minute ago

Instance ID i-0cba2154f7541d42b (wireapps-devops-challenge:0.2)	Public IPv4 address 54.211.33.233 | open address	Private IPv4 addresses 172.31.95.139
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-54-211-33-233.compute-1.amazonaws.com | open address
Hostname type IP name: ip-172-31-95-139.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-95-139.ec2.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address 54.211.33.233 [Public IP]	VPC ID vpc-0a105c4163c57c4db	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	Subnet ID subnet-03277cbe090563b51	Auto Scaling Group name –
IMDSv2 Required	Instance ARN	

© 2024, Amazon Web Services, Inc. or its affiliates.

- **Security Group Configuration**:
- Allowed inbound traffic on ports **80** (HTTP) and **443** (HTTPS).
- Enabled SSH access (port 22) to manage the server.


Amazon Web Services Sign-In | Amazon Web Services Sign-In | ModifyInboundSecurityGroupR | History | Launch an instance | EC2 | us-e

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:

Security group rule 4 (TCP, 3000, 0.0.0.0/0)

Type	Protocol	Port range
Custom TCP	TCP	3000

Source type	Source	Description - optional
Anywhere	0.0.0.0/0	e.g. SSH for admin desktop

Security group rule 5 (TCP, 4000, 0.0.0.0/0)

Type	Protocol	Port range
Custom TCP	TCP	4000

Source type	Source	Description - optional
Anywhere	0.0.0.0/0	e.g. SSH for admin desktop

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Add security group rule

Summary

Number of instances: 1

Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more ami-0e86e20dae9224db8

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

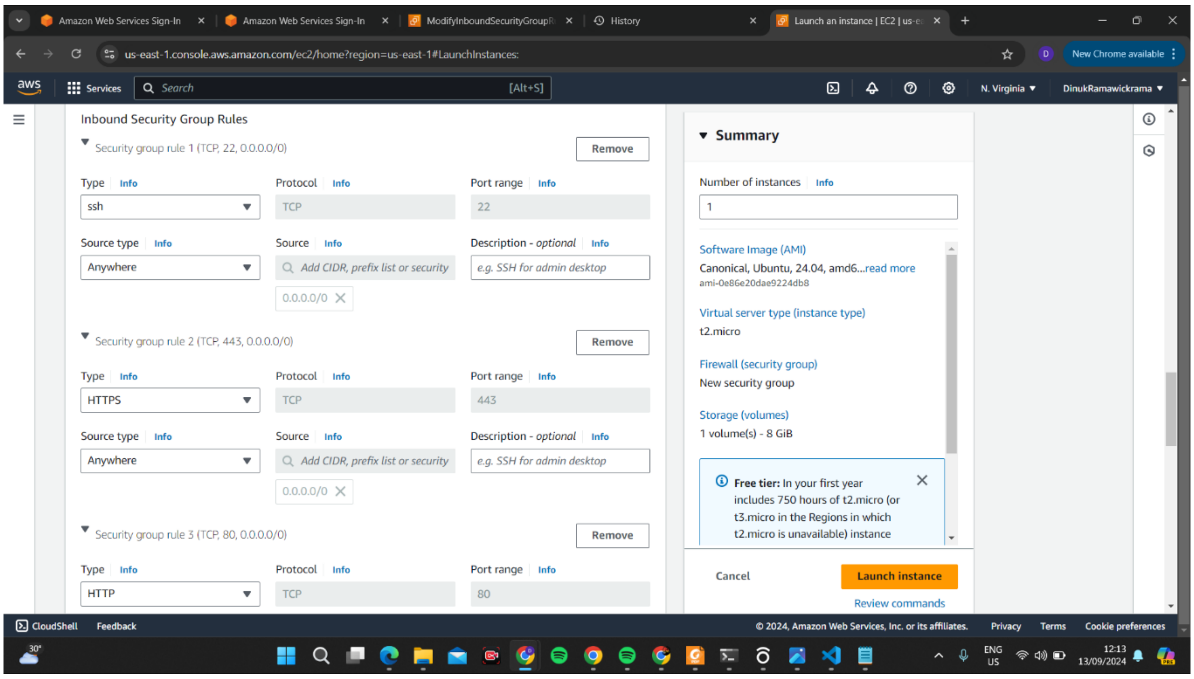
Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance

Cancel | Launch instance | Review commands

© 2024, Amazon Web Services, Inc. or its affiliates.

12:13 13/09/2024


Amazon Web Services Sign-In
Amazon Web Services Sign-In
ModifyInboundSecurityGroupR
History
Launch an instance | EC2 | us-e
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
New Chrome available
Services
Search
[Alt+S]
N. Virginia
DinukRamawickrama

Inbound Security Group Rules

Security group rule 1 (TCP, 22, 0.0.0.0/0) — Remove

Type	Protocol	Port range
ssh	TCP	22

Source type	Source	Description - *optional*
Anywhere	Add CIDR, prefix list or security; 0.0.0.0/0	e.g. SSH for admin desktop

Security group rule 2 (TCP, 443, 0.0.0.0/0) — Remove

Type	Protocol	Port range
HTTPS	TCP	443

Source type	Source	Description - *optional*
Anywhere	Add CIDR, prefix list or security; 0.0.0.0/0	e.g. SSH for admin desktop

Security group rule 3 (TCP, 80, 0.0.0.0/0) — Remove

Type	Protocol	Port range
HTTP	TCP	80

Summary

Number of instances: 1

Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0e86e20dae9224db8

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

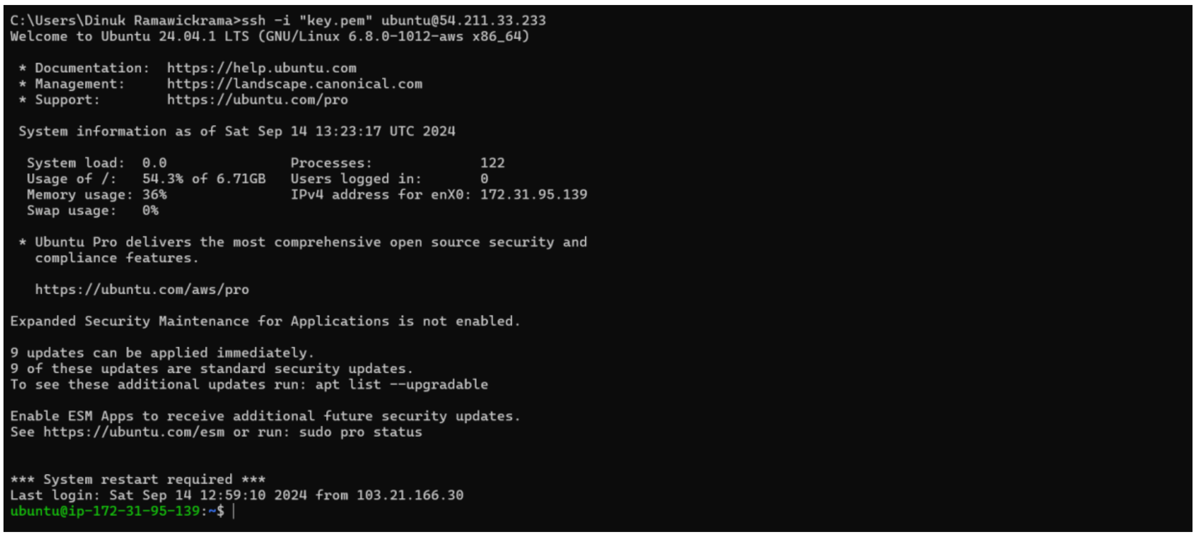
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance

Cancel — Launch instance — Review commands

Step 2: Basic Server Setup with a Bash Script

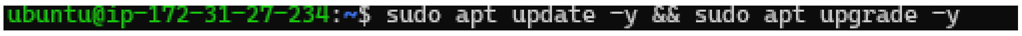
1. **Update System Packages:**

First log in to ec2 intence using command ssh -i “key.pem” ubuntu@54.209.87.23 -p 22



```
C:\Users\Dinuk Ramawickrama>ssh -i "key.pem" ubuntu@54.211.33.233
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Sat Sep 14 13:23:17 UTC 2024

  System load:  0.0               Processes:             122
  Usage of /:   54.3% of 6.71GB   Users logged in:       0
  Memory usage: 36%               IPv4 address for enX0: 172.31.95.139
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

9 updates can be applied immediately.
9 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


*** System restart required ***
Last login: Sat Sep 14 12:59:10 2024 from 103.21.166.30
ubuntu@ip-172-31-95-139:~$ |
```

- Ran the command sudo apt update -y && sudo apt upgrade -y to update the package manager's lists.



```
ubuntu@ip-172-31-27-234:~$ sudo apt update -y && sudo apt upgrade -y
```

2. **Install Nginx, Docker, and Docker Compose**:

   - Installed Nginx using
     ```
     sudo apt install nginx -y.
     ```
   - Installed Docker using
     ```
     sudo apt install docker.io -y
     ```
   - Installed Docker Compose using
     ```
     sudo apt install docker-compose -y.
     ```

3. **Start Services**:

   Started Nginx and Docker with

   ```
   sudo systemctl start nginx

   sudo systemctl start docker.
   ```

   Enabled them to start on boot with

   ```
   sudo systemctl enable nginx

   sudo systemctl enable docker.
   ```

Docker Setup:

Clone and Set Up the Repository

Clone the Busbud Repository:

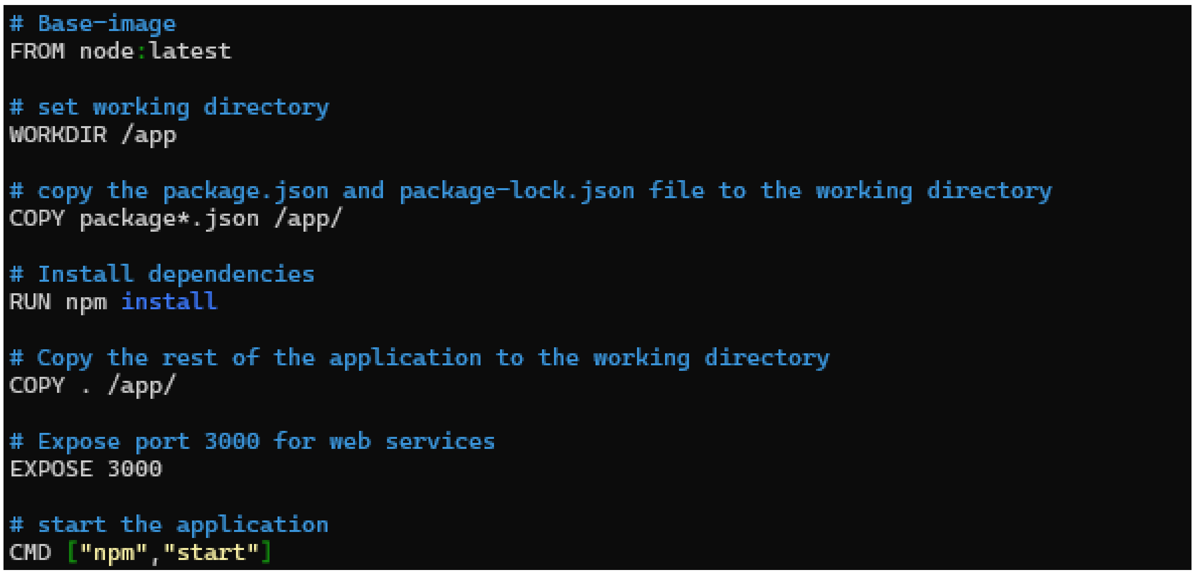
Clone the busdbud/devops-challenge-apps repository to the server.

```
git clone https://github.com/busbud/devops-challenge-apps.git.

cd devops-challenge-apps
```

1. **Create Dockerfile for Web**:

   Created a Dockerfile in the web folder to build the application container.

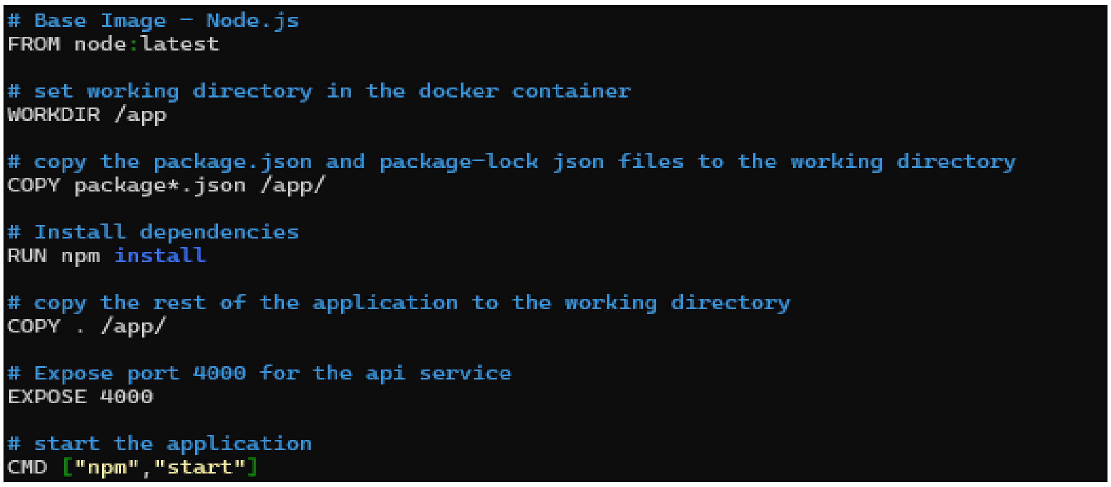
   Exposed port **3000** for web services.



```
# Base-image
FROM node:latest

# set working directory
WORKDIR /app

# copy the package.json and package-lock.json file to the working directory
COPY package*.json /app/

# Install dependencies
RUN npm install

# Copy the rest of the application to the working directory
COPY . /app/

# Expose port 3000 for web services
EXPOSE 3000

# start the application
CMD ["npm","start"]
```

2. **Create Dockerfile for API**:

   Created a Dockerfile in the api folder, which exposed port **4000** for API services.

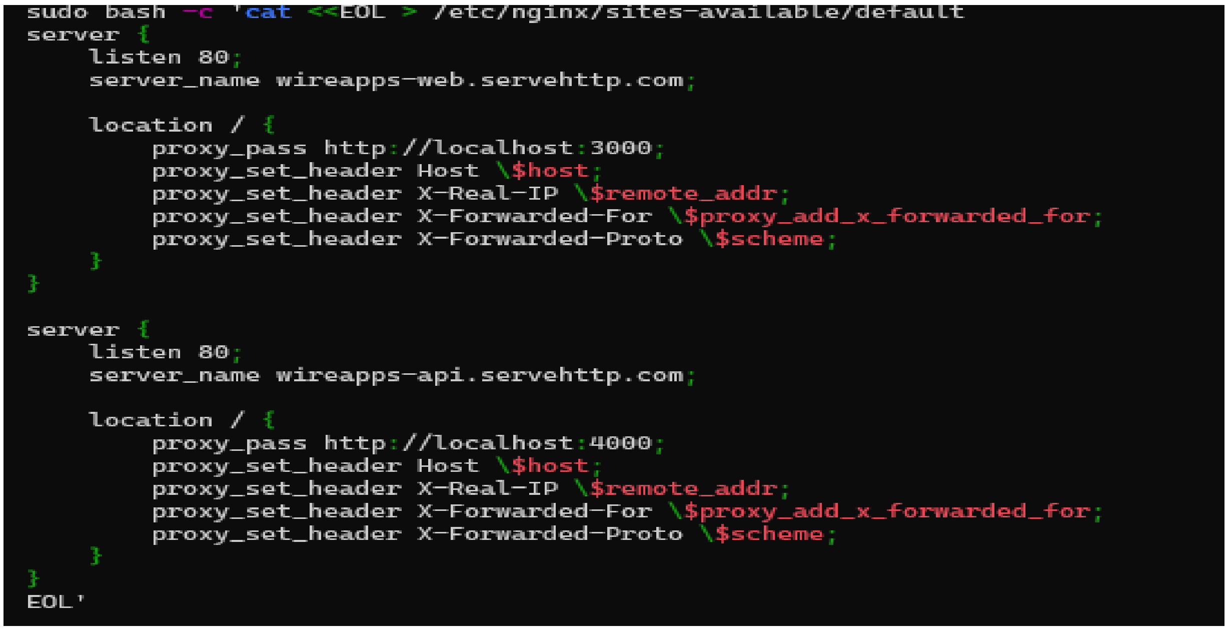


```
# Base Image - Node.js
FROM node:latest

# set working directory in the docker container
WORKDIR /app

# copy the package.json and package-lock json files to the working directory
COPY package*.json /app/

# Install dependencies
RUN npm install

# copy the rest of the application to the working directory
COPY . /app/

# Expose port 4000 for the api service
EXPOSE 4000

# start the application
CMD ["npm","start"]
```

Nginx Host-Based Routing Configuration:

Configure Nginx for Host-Based Routing

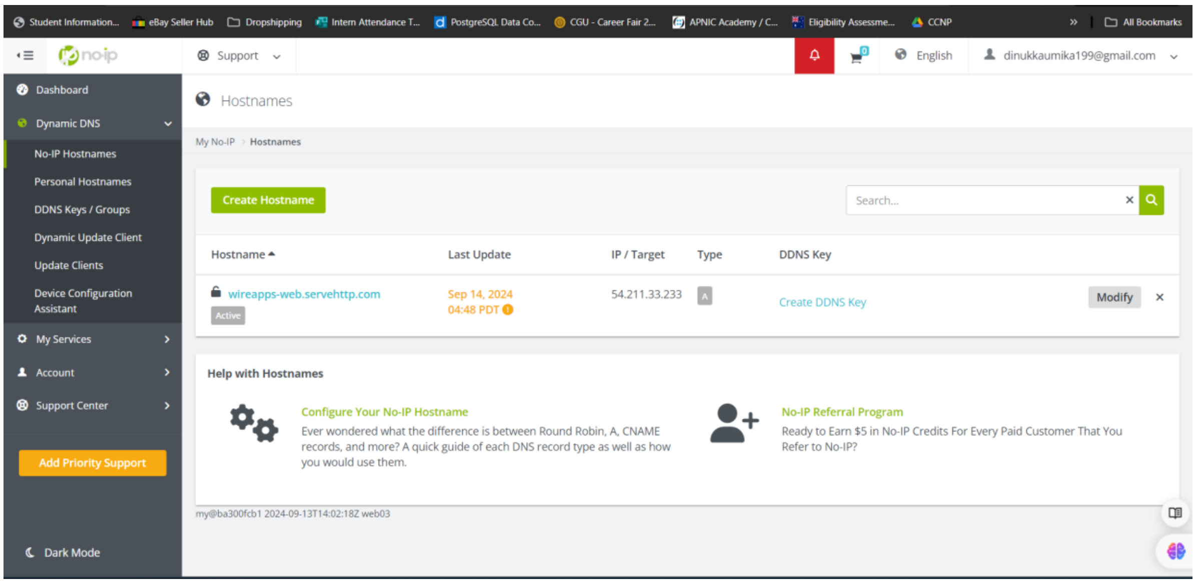
1. **Nginx Configuration**:
   - Configured Nginx to route traffic based on the domain names.
   - Set up routing so that:



```
sudo bash -c 'cat <<EOL > /etc/nginx/sites-available/default
server {
    listen 80;
    server_name wireapps-web.servehttp.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
        proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto \$scheme;
    }
}

server {
    listen 80;
    server_name wireapps-api.servehttp.com;

    location / {
        proxy_pass http://localhost:4000;
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
        proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto \$scheme;
    }
}
EOL'
```

- wireapps-web.servehttp.com directs to the web app running on port **3000**.
- wireapps-api.servehttp.com directs to the API running on port **4000**.

I used noip.com to get free domain names for testing



Hostname	Last Update	IP / Target	Type	DDNS Key	
wireapps-web.servehttp.com Active	Sep 14, 2024 04:48 PDT	54.211.33.233	A	Create DDNS Key	Modify ×

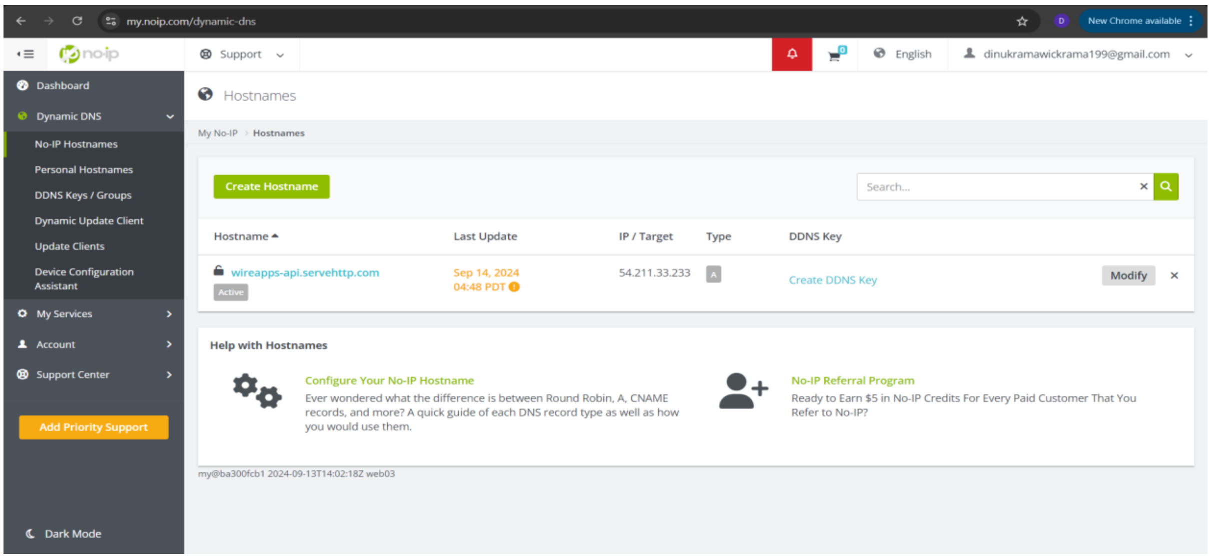
Help with Hostnames

Configure Your No-IP Hostname
Ever wondered what the difference is between Round Robin, A, CNAME records, and more? A quick guide of each DNS record type as well as how you would use them.

No-IP Referral Program
Ready to Earn $5 in No-IP Credits For Every Paid Customer That You Refer to No-IP?

my@ba300fcb1 2024-09-13T14:02:18Z web03

web app - wireapps-web.servehttp.com



Hostname	Last Update	IP / Target	Type	DDNS Key	
wireapps-api.servehttp.com Active	Sep 14, 2024 04:48 PDT	54.211.33.233	A	Create DDNS Key	Modify ×

Help with Hostnames

Configure Your No-IP Hostname
Ever wondered what the difference is between Round Robin, A, CNAME records, and more? A quick guide of each DNS record type as well as how you would use them.

No-IP Referral Program
Ready to Earn $5 in No-IP Credits For Every Paid Customer That You Refer to No-IP?

my@ba300fcb1 2024-09-13T14:02:18Z web03

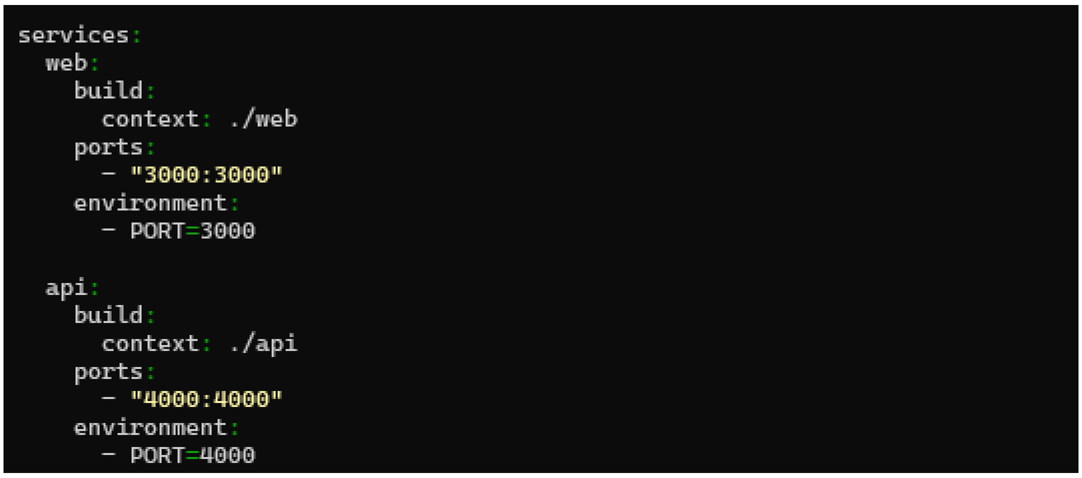
Api app - wireapps-api.servehttp.com

Docker Compose Setup

1. **Create docker-compose.yml**:

   Created a docker-compose.yml file to bring up both the web and API services in separate containers.

   Configured port mappings for both services (web: **3000**, API: **4000**).

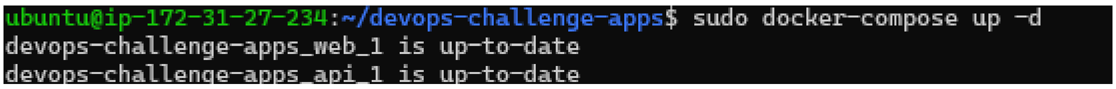


```
services:
  web:
    build:
      context: ./web
    ports:
      - "3000:3000"
    environment:
      - PORT=3000

  api:
    build:
      context: ./api
    ports:
      - "4000:4000"
    environment:
      - PORT=4000
```

2. **Pull and Build Docker Images**:

   Built the Docker images using

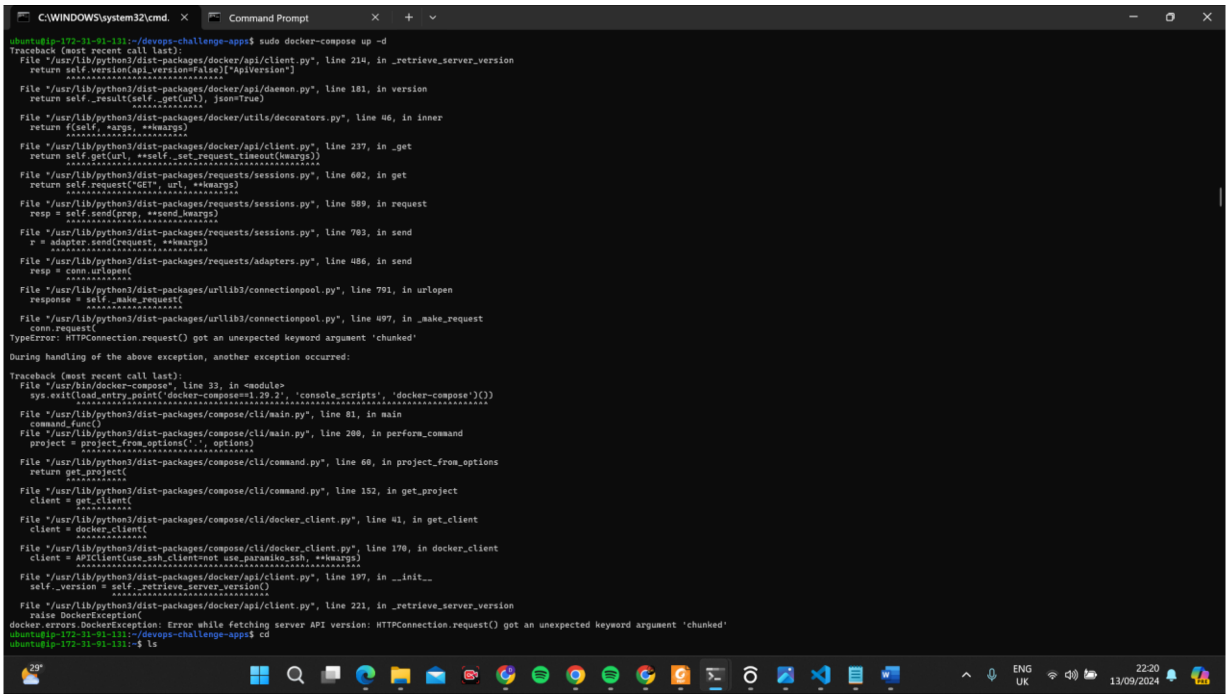
   docker-compose up -d.



```
ubuntu@ip-172-31-27-234:~/devops-challenge-apps$ sudo docker-compose up -d
devops-challenge-apps_web_1 is up-to-date
devops-challenge-apps_api_1 is up-to-date
```

Dependency Issues and Fixes

1. **Python urllib3 Issue**:



```
ubuntu@ip-172-31-91-131:~/devops-challenge-apps$ sudo docker-compose up -d
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/docker/api/client.py", line 214, in _retrieve_server_version
    return self.version(api_version=False)["ApiVersion"]
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/docker/api/daemon.py", line 181, in version
    return self._result(self._get(url), json=True)
                        ^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/docker/utils/decorators.py", line 46, in inner
    return f(self, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/docker/api/client.py", line 237, in _get
    return self.get(url, **self._set_request_timeout(kwargs))
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/requests/sessions.py", line 602, in get
    return self.request("GET", url, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/requests/sessions.py", line 589, in request
    resp = self.send(prep, **send_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/requests/sessions.py", line 703, in send
    r = adapter.send(request, **kwargs)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/requests/adapters.py", line 486, in send
    resp = conn.urlopen(
           ^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/urllib3/connectionpool.py", line 791, in urlopen
    response = self._make_request(
               ^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/urllib3/connectionpool.py", line 497, in _make_request
    conn.request(
TypeError: HTTPConnection.request() got an unexpected keyword argument 'chunked'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/bin/docker-compose", line 33, in <module>
    sys.exit(load_entry_point('docker-compose==1.29.2', 'console_scripts', 'docker-compose')())
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/compose/cli/main.py", line 81, in main
    command_func()
  File "/usr/lib/python3/dist-packages/compose/cli/main.py", line 200, in perform_command
    project = project_from_options('.', options)
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/compose/cli/command.py", line 60, in project_from_options
    return get_project(
           ^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/compose/cli/command.py", line 152, in get_project
    client = get_client(
             ^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/compose/cli/docker_client.py", line 41, in get_client
    client = docker_client(
             ^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/compose/cli/docker_client.py", line 170, in docker_client
    client = APIClient(use_ssh_client=not use_paramiko_ssh, **kwargs)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/docker/api/client.py", line 197, in __init__
    self._version = self._retrieve_server_version()
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/docker/api/client.py", line 221, in _retrieve_server_version
    raise DockerException(
docker.errors.DockerException: Error while fetching server API version: HTTPConnection.request() got an unexpected keyword argument 'chunked'
ubuntu@ip-172-31-91-131:~/devops-challenge-apps$ cd
ubuntu@ip-172-31-91-131:~$ ls
```

- Encountered an issue with python3-urllib3 during the installation.
- Fixed it by removing the conflicting package and installing a compatible version using pip:

Fixing Docker Compose Error

While working on setting up Docker Compose for the application deployment, I encountered an issue when trying to run the docker-compose command. The error looked like this:

This error occurred because Docker was unable to communicate with the Docker daemon, and it prevented Docker Compose from functioning properly.

Solution Steps:

Step 1: Install python3-pip

The first step was to ensure that pip was installed correctly on the system.

```
sudo apt install python3-pip -y
```

Step 2: Remove Conflicting python3-urllib3 Package

There was a conflict with the python3-urllib3 package, which caused issues with Docker Compose.

```
sudo apt-get remove python3-urllib3 -y
```

Step 3: Install a Compatible Version of urllib3

To fix the issue, I installed an older version of urllib3 using pip that was compatible with the system and Docker Compose.

```
sudo pip install 'urllib3<2' --break-system-packages
```

This step ensures that Docker Compose can function without conflicts from incompatible urllib3 versions.

Step 4: Reinstall Docker Compose

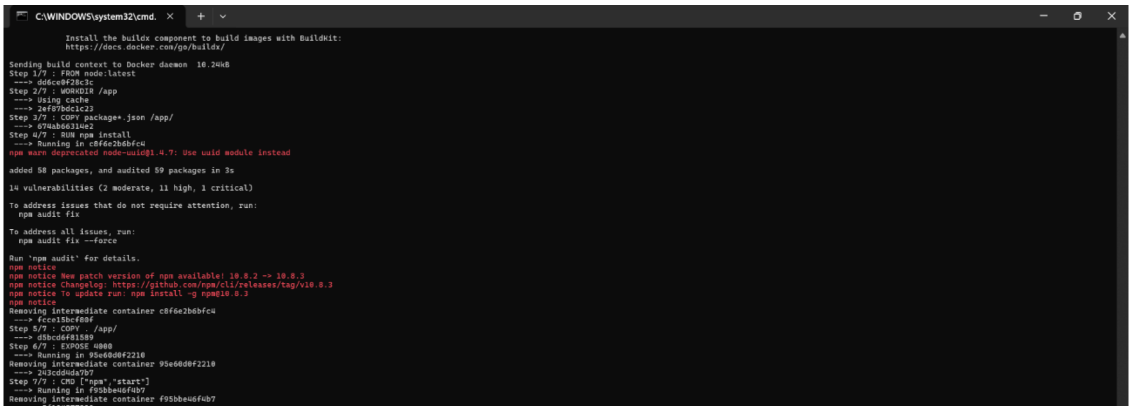
After resolving the urllib3 issue, I reinstalled Docker Compose to ensure the version was correct and compatible with the Docker daemon.

```
sudo apt install docker-compose -y
```

2. **NPM Dependency Issues**:

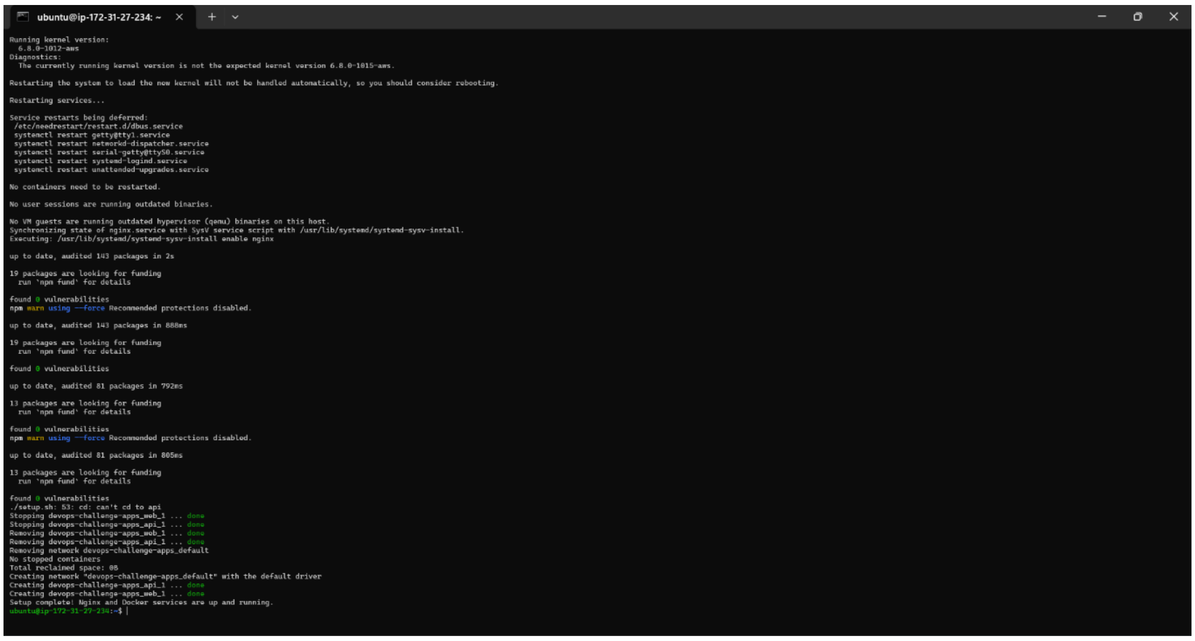
   Faced errors with NPM dependencies, which I resolved by running:

This is error ,



```
          Install the buildx component to build images with BuildKit:
          https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  10.24kB
Step 1/7 : FROM node:latest
 ---> dd6ce0f28c3c
Step 2/7 : WORKDIR /app
 ---> Using cache
 ---> 2ef87bdc1c23
Step 3/7 : COPY package*.json /app/
 ---> 674ab66314e2
Step 4/7 : RUN npm install
 ---> Running in c8f6e2b6bfc4
npm warn deprecated node-uuid@1.4.7: Use uuid module instead

added 58 packages, and audited 59 packages in 3s

14 vulnerabilities (2 moderate, 11 high, 1 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.
npm notice
npm notice New patch version of npm available! 10.8.2 -> 10.8.3
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.8.3
npm notice To update run: npm install -g npm@10.8.3
npm notice
Removing intermediate container c8f6e2b6bfc4
 ---> fcce15bcf80f
Step 5/7 : COPY . /app/
 ---> d5bcd6f81589
Step 6/7 : EXPOSE 4000
 ---> Running in 95e60d0f2210
Removing intermediate container 95e60d0f2210
 ---> 243cdd4da7b7
Step 7/7 : CMD ["npm","start"]
 ---> Running in f95bbe46f4b7
Removing intermediate container f95bbe46f4b7
```

After fixing the error, the fully automated script is now running successfully.


```
Running kernel version:
  6.8.0-1012-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.8.0-1015-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...

Service restarts being deferred:
 /etc/needrestart/restart.d/dbus.service
 systemctl restart getty@tty1.service
 systemctl restart networkd-dispatcher.service
 systemctl restart serial-getty@ttyS0.service
 systemctl restart systemd-logind.service
 systemctl restart unattended-upgrades.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
Synchronizing state of nginx.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nginx

up to date, audited 143 packages in 2s

19 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm warn using --force Recommended protections disabled.

up to date, audited 143 packages in 888ms

19 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

up to date, audited 81 packages in 792ms

13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm warn using --force Recommended protections disabled.

up to date, audited 81 packages in 805ms

13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
./setup.sh: 53: cd: can't cd to api
Stopping devops-challenge-apps_web_1 ... done
Stopping devops-challenge-apps_api_1 ... done
Removing devops-challenge-apps_web_1 ... done
Removing devops-challenge-apps_api_1 ... done
Removing network devops-challenge-apps_default
No stopped containers
Total reclaimed space: 0B
Creating network "devops-challenge-apps_default" with the default driver
Creating devops-challenge-apps_api_1 ... done
Creating devops-challenge-apps_web_1 ... done
Setup complete! Nginx and Docker services are up and running.
ubuntu@ip-172-31-27-234:~$ |
```

Update npm Dependencies :

Navigating to the web and API directories:

The script first navigates to the **web** directory (cd web).

After the web directory is done, it moves to the **api** directory (cd ../api).

Running npm install:

npm install --legacy-peer-deps:

This command installs the necessary dependencies listed in the package.json file for both the web and API projects.

The --legacy-peer-deps flag is used to bypass issues with peer dependencies (which may cause installation errors). It ensures the installation continues even if there are conflicts between dependency versions.

Running npm audit fix:

npm audit fix --force:

This command automatically tries to fix any security vulnerabilities found in the installed dependencies. The --force flag ensures that the fixes are applied even if some updates might potential ly break compatibility.

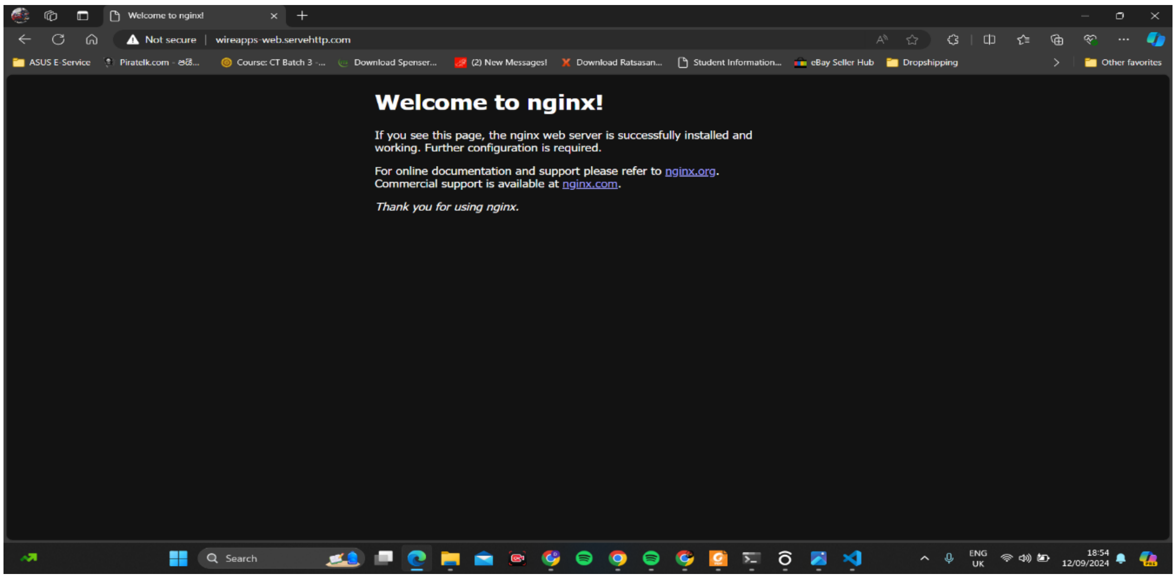
|| true:

This part ensures that even if the commands fail, the script doesn't stop and continues to the next steps.

Testing the Application

1. **Access the Web and API**:

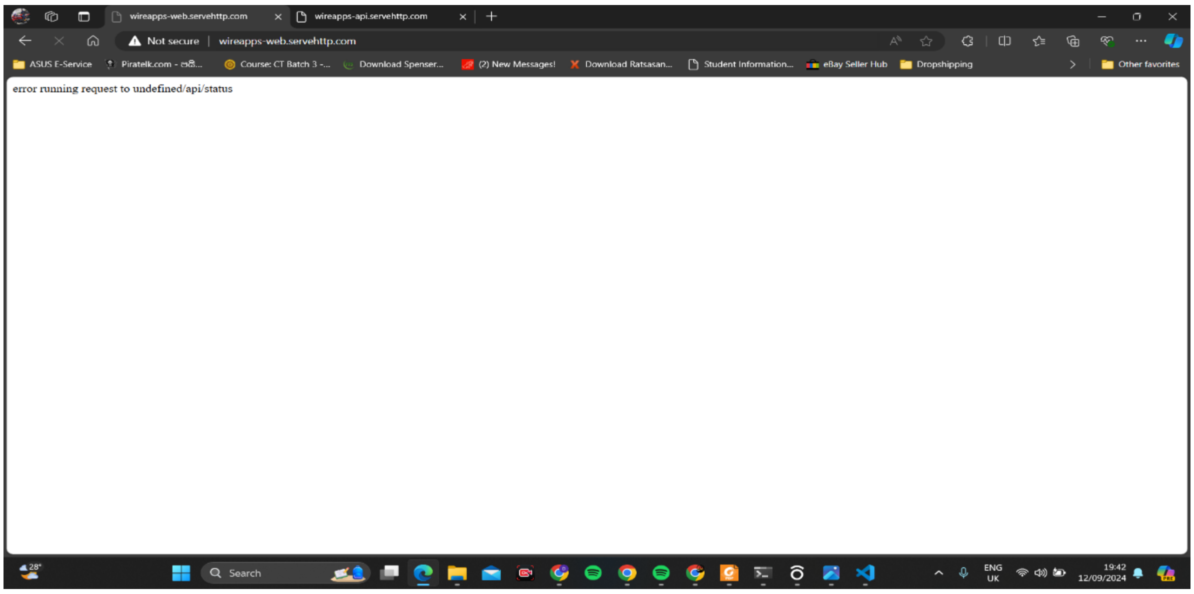
   Before fixing the error - wireapps-web.servehttp.com(screen shot)



```
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.
```

After completing the setup, I tested the applications by accessing the,

After Fixing the error – wireapps-web.servehttp.com

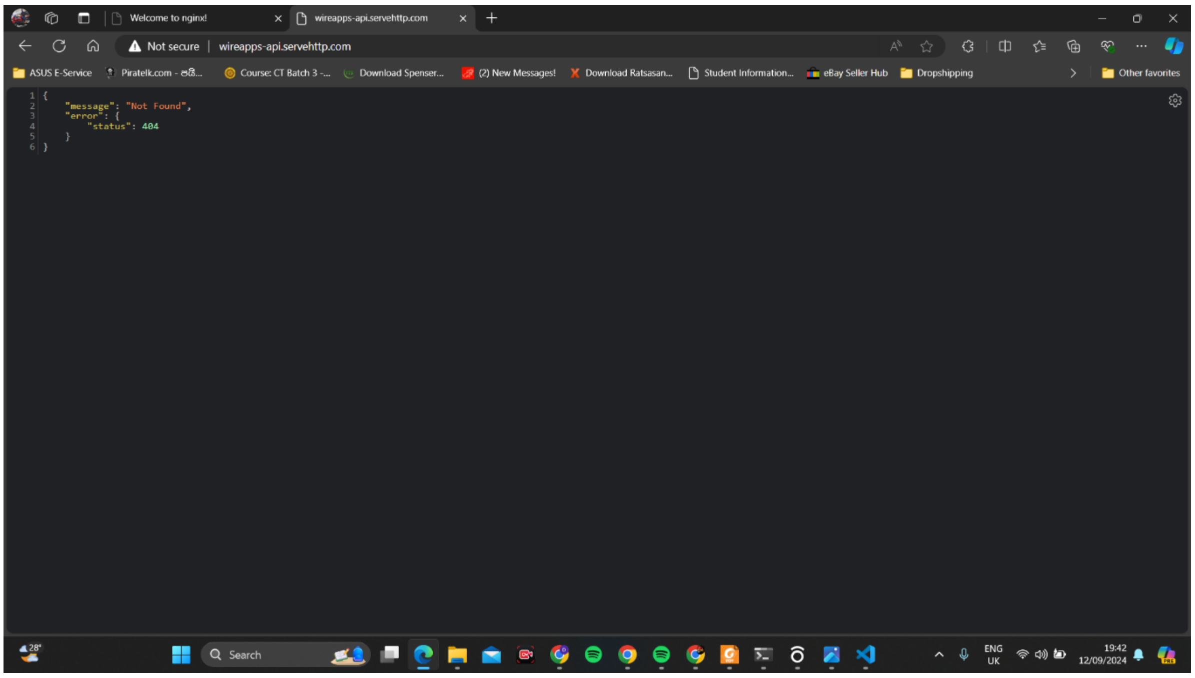


wireapps-web.servehttp.com
wireapps-api.servehttp.com
Not secure | wireapps-web.servehttp.com

```
error running request to undefined/api/status
```

*Note In this repository, there is a package build error that I attempted to fix, but the repository cannot be forked.

Api app – wireapps-api.servehttp.com



```
{
    "message": "Not Found",
    "error": {
        "status": 404
    }
}
```

*Note -In this repository, there is a package build error that I attempted to fix, but the repository cannot be forked.

Automating the Entire Process

Objective:

Create a Bash script to automate all the steps for setting up the environment and deploying the applications.

Bash Script:

I created a script that:

- Updates the package list
- Installs dependencies
- Configures Docker and Nginx
- Clones the application repositories
- Builds and deploys Docker containers

- Fixes any issues with Docker Compose

Here's the complete setup.sh script:

First using – sudo nano setup.sh

Then Create this script,

```
# Wireapps-DevOps-Challenge--[Dinuk-Kaumika-Ramawickrama]

# 1. Update and upgrade package list
sudo apt-get update -y
sudo apt-get upgrade -y

# 2. Install nginx
sudo apt install nginx -y

# 3. Install Docker
sudo apt install docker.io -y

# 4. Install Python3 pip
sudo apt install python3-pip -y

# 5. Remove problematic version of python3-urllib3
sudo apt-get remove python3-urllib3 -y

# 6. Install compatible version of urllib3 using pip
sudo pip install 'urllib3<2' --break-system-packages

# 7. Install Docker Compose
sudo apt install docker-compose -y
```

```
# 8. Start and enable Nginx
sudo systemctl start nginx
sudo systemctl enable nginx

# 9. Start and enable Docker
sudo systemctl start docker
sudo systemctl enable docker

# 10. Clone the busbud/devops-challenge-apps repository
if [ ! -d "devops-challenge-apps" ]; then
  git clone https://github.com/busbud/devops-challenge-apps.git
fi
cd devops-challenge-apps

# 11. Update npm dependencies
cd web
npm install --legacy-peer-deps || true
npm audit fix --force || true

cd ../api
npm install --legacy-peer-deps || true
npm audit fix --force || true

# 12. Create the Dockerfile for API and add it to the 'api' folder
cd api

cat << 'EOF' > ./Dockerfile
# Base Image - Node.js
FROM node:latest

# set working directory in the docker container
```

```
WORKDIR /app

# copy the package.json and package-lock json files to the working directoryCOPY
package*.json /app/

# Install dependencies
RUN npm install

# copy the rest of the application to the working directory
COPY . /app/

# Expose port 4000 for the api service
EXPOSE 4000

# start the application
CMD ["npm","start"]
EOF

# 13. Create the Dockerfile for Web and add it to the 'web' folder
cd ../web

cat << 'EOF' > ./Dockerfile
# Base-image
FROM node:latest

# set working directory
WORKDIR /app

# copy the package.json and package-lock.json file to the working directory
COPY package*.json /app/
```

```
# Install dependencies
RUN npm install

# Copy the rest of the application to the working directory
COPY . /app/

# Expose port 3000 for web services
EXPOSE 3000

# start the application
CMD ["npm","start"]
EOF

# 14. Set up Nginx configuration for host-based routing
sudo bash -c 'cat <<EOL > /etc/nginx/sites-available/default
server {
  listen 80;
  server_name wireapps-web.servehttp.com;

  location / {
    proxy_pass http://localhost:3000;
    proxy_set_header Host \$host;
    proxy_set_header X-Real-IP \$remote_addr;
    proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto \$scheme;
  }
}

server {
  listen 80;
  server_name wireapps-api.servehttp.com;
```

```
    location / {
        proxy_pass http://localhost:4000;
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
        proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto \$scheme;
    }
}
EOL'

# 15. Restart Nginx to apply the new configuration
sudo systemctl restart nginx

# 16. Build Docker images for API and Web applications using Docker Compose
cd /devops-challenge-apps

sudo docker-compose down

# Remove old containers and images
sudo docker system prune -f

# 17. Pull or build Docker images and bring up services
# Create docker-compose.yml file
cat <<EOL > /devops-challenge-apps/docker-compose.yml
version: '3.8'

services:
  web:
    build:
      context: ./web
    ports:
      - "3000:3000"
    environment:
      - PORT=3000

  api:
    build:
      context: ./api
    ports:
      - "4000:4000"
    environment:
      - PORT=4000
EOL

# Start Docker Compose
sudo docker-compose up -d

echo "Setup complete! Nginx and Docker services are up and running."
```

- Then after save this bash script using – Ctrl+X – press Y – press Enter
- After that it make as executable file using

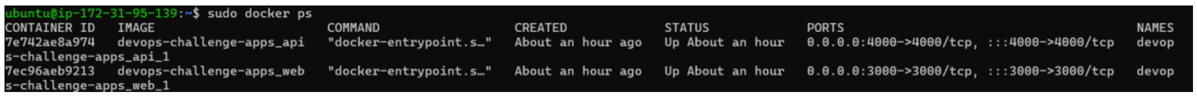
```
chmod +x setup.sh
```

- Running the bash script using –

```
sudo ./setup.sh
```

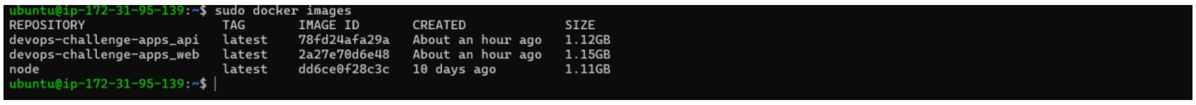
<u>Final Testing</u>

Checking docker containers running perfectly after run this scrip .



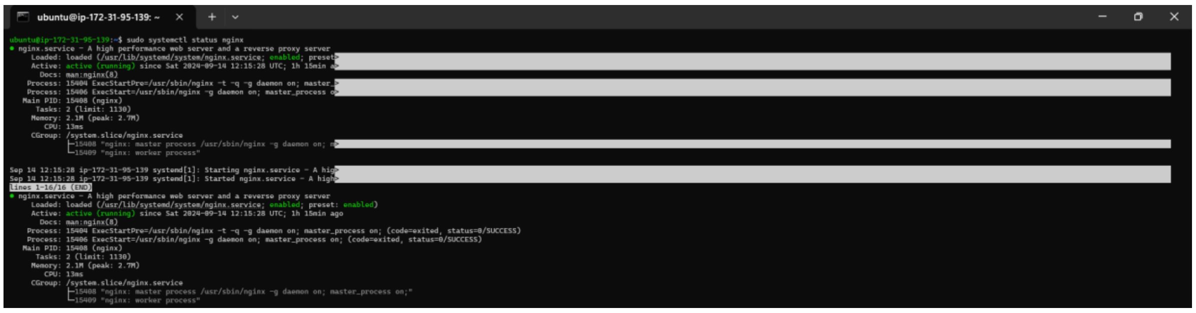
```
ubuntu@ip-172-31-95-139:~$ sudo docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED             STATUS              PORTS                                        NAMES
7e742ae8a974   devops-challenge-apps_api   "docker-entrypoint.s…"   About an hour ago   Up About an hour   0.0.0.0:4000->4000/tcp, :::4000->4000/tcp   devop
s-challenge-apps_api_1
7ec96aeb9213   devops-challenge-apps_web   "docker-entrypoint.s…"   About an hour ago   Up About an hour   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   devop
s-challenge-apps_web_1
```

Checking docker images



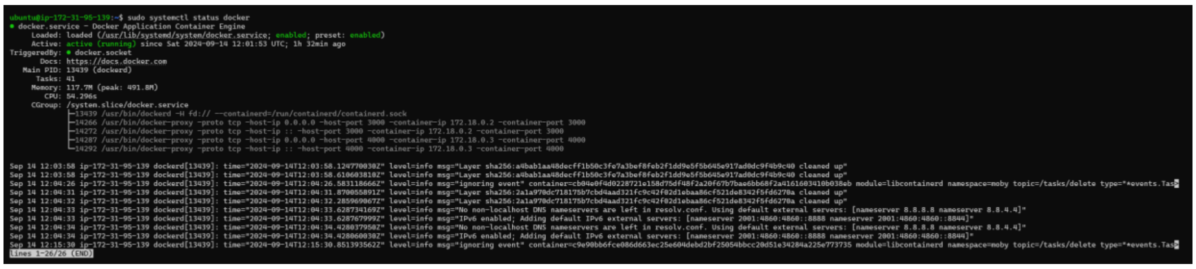
```
ubuntu@ip-172-31-95-139:~$ sudo docker images
REPOSITORY                 TAG       IMAGE ID       CREATED            SIZE
devops-challenge-apps_api  latest    78fd24afa29a   About an hour ago  1.12GB
devops-challenge-apps_web  latest    2a27e70d6e48   About an hour ago  1.15GB
node                       latest    dd6ce0f28c3c   10 days ago        1.11GB
ubuntu@ip-172-31-95-139:~$ |
```

Cheking Nginx is running



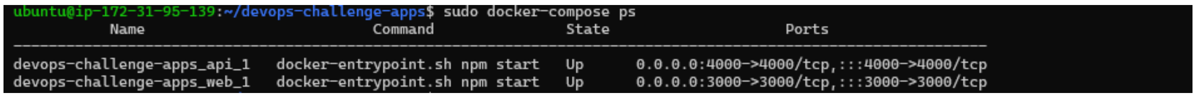
```
ubuntu@ip-172-31-95-139:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset>
     Active: active (running) since Sat 2024-09-14 12:15:28 UTC; 1h 15min a>
       Docs: man:nginx(8)
    Process: 15404 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_>
    Process: 15406 ExecStart=/usr/sbin/nginx -g daemon on; master_process o>
   Main PID: 15408 (nginx)
      Tasks: 2 (limit: 1130)
     Memory: 2.1M (peak: 2.7M)
        CPU: 13ms
     CGroup: /system.slice/nginx.service
             ├─15408 "nginx: master process /usr/sbin/nginx -g daemon on; m>
             └─15409 "nginx: worker process"

Sep 14 12:15:28 ip-172-31-95-139 systemd[1]: Starting nginx.service - A hig>
Sep 14 12:15:28 ip-172-31-95-139 systemd[1]: Started nginx.service - A high>
lines 1-16/16 (END)
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-09-14 12:15:28 UTC; 1h 15min ago
       Docs: man:nginx(8)
    Process: 15404 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
    Process: 15406 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 15408 (nginx)
      Tasks: 2 (limit: 1130)
     Memory: 2.1M (peak: 2.7M)
        CPU: 13ms
     CGroup: /system.slice/nginx.service
             ├─15408 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─15409 "nginx: worker process"
```

Verify Docker is running



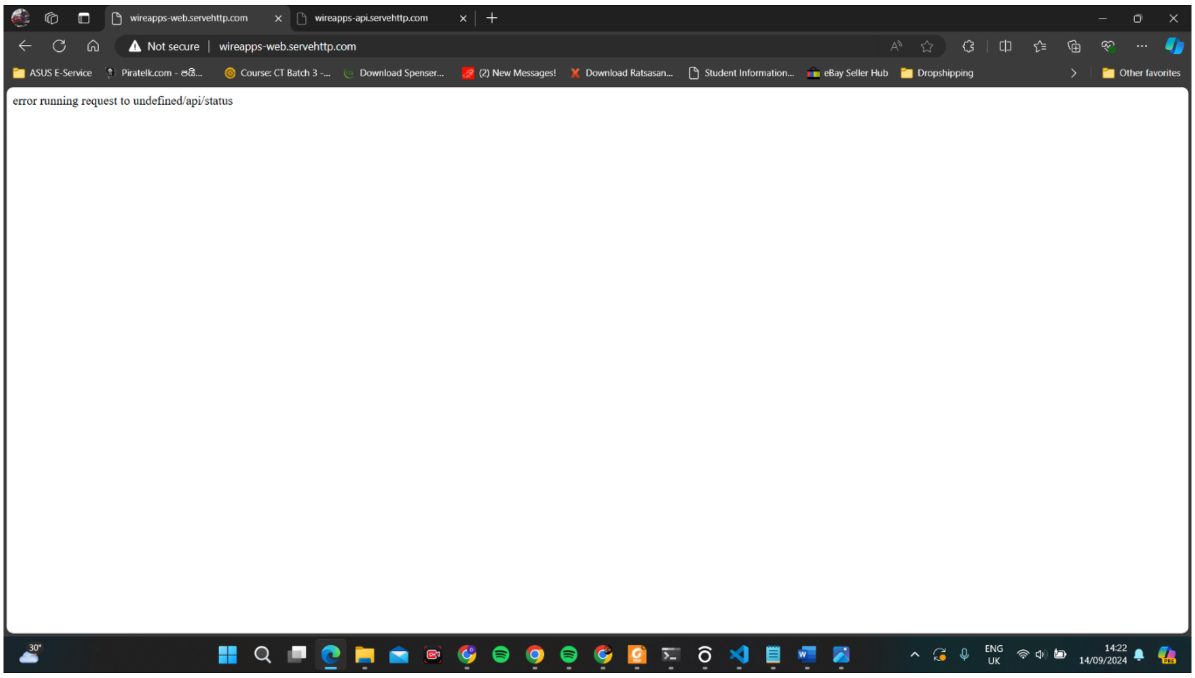
```
ubuntu@ip-172-31-95-139:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-09-14 12:01:53 UTC; 1h 32min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 13439 (dockerd)
      Tasks: 41
     Memory: 117.7M (peak: 491.8M)
        CPU: 54.296s
     CGroup: /system.slice/docker.service
             ├─13439 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
             ├─14266 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 3000 -container-ip 172.18.0.2 -container-port 3000
             ├─14272 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 3000 -container-ip 172.18.0.2 -container-port 3000
             ├─14287 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 4000 -container-ip 172.18.0.3 -container-port 4000
             └─14292 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 4000 -container-ip 172.18.0.3 -container-port 4000

Sep 14 12:03:58 ip-172-31-95-139 dockerd[13439]: time="2024-09-14T12:03:58.124770030Z" level=info msg="Layer sha256:a4bab1aa48decff1b50c3fe7a3bef8feb2f1dd9e5f5b645e917ad0dc9f4b9c40 cleaned up"
Sep 14 12:03:58 ip-172-31-95-139 dockerd[13439]: time="2024-09-14T12:03:58.610603810Z" level=info msg="Layer sha256:a4bab1aa48decff1b50c3fe7a3bef8feb2f1dd9e5f5b645e917ad0dc9f4b9c40 cleaned up"
Sep 14 12:04:26 ip-172-31-95-139 dockerd[13439]: time="2024-09-14T12:04:26.583118666Z" level=info msg="ignoring event" container=cb04e0f4d0228721e158d75df48f2a20f67b7bae6bb68f2a4161603410b038eb module=libcontainerd namespace=moby topic=/tasks/delete type="*events.Tas>
Sep 14 12:04:31 ip-172-31-95-139 dockerd[13439]: time="2024-09-14T12:04:31.870055891Z" level=info msg="Layer sha256:2a1a970dc718175b7cbd4aad321fc9c42f02d1ebaa86cf521de8342f5fd6270a cleaned up"
Sep 14 12:04:32 ip-172-31-95-139 dockerd[13439]: time="2024-09-14T12:04:32.285969067Z" level=info msg="Layer sha256:2a1a970dc718175b7cbd4aad321fc9c42f02d1ebaa86cf521de8342f5fd6270a cleaned up"
Sep 14 12:04:33 ip-172-31-95-139 dockerd[13439]: time="2024-09-14T12:04:33.628734169Z" level=info msg="No non-localhost DNS nameservers are left in resolv.conf. Using default external servers: [nameserver 8.8.8.8 nameserver 8.8.4.4]"
Sep 14 12:04:33 ip-172-31-95-139 dockerd[13439]: time="2024-09-14T12:04:33.628767999Z" level=info msg="IPv6 enabled; Adding default IPv6 external servers: [nameserver 2001:4860:4860::8888 nameserver 2001:4860:4860::8844]"
Sep 14 12:04:34 ip-172-31-95-139 dockerd[13439]: time="2024-09-14T12:04:34.428037950Z" level=info msg="No non-localhost DNS nameservers are left in resolv.conf. Using default external servers: [nameserver 8.8.8.8 nameserver 8.8.4.4]"
Sep 14 12:04:34 ip-172-31-95-139 dockerd[13439]: time="2024-09-14T12:04:34.428060030Z" level=info msg="IPv6 enabled; Adding default IPv6 external servers: [nameserver 2001:4860:4860::8888 nameserver 2001:4860:4860::8844]"
Sep 14 12:15:30 ip-172-31-95-139 dockerd[13439]: time="2024-09-14T12:15:30.851393562Z" level=info msg="ignoring event" container=c9e90bb6fce086d663ec25e604debd2bf25054bbcc20d51e34284a225e773735 module=libcontainerd namespace=moby topic=/tasks/delete type="*events.Tas>
lines 1-26/26 (END)
```

Check Docker Compose Health



```
ubuntu@ip-172-31-95-139:~/devops-challenge-apps$ sudo docker-compose ps
           Name                        Command             State                       Ports
---------------------------------------------------------------------------------------------------------------
devops-challenge-apps_api_1   docker-entrypoint.sh npm start   Up      0.0.0.0:4000->4000/tcp,:::4000->4000/tcp
devops-challenge-apps_web_1   docker-entrypoint.sh npm start   Up      0.0.0.0:3000->3000/tcp,:::3000->3000/tcp
```

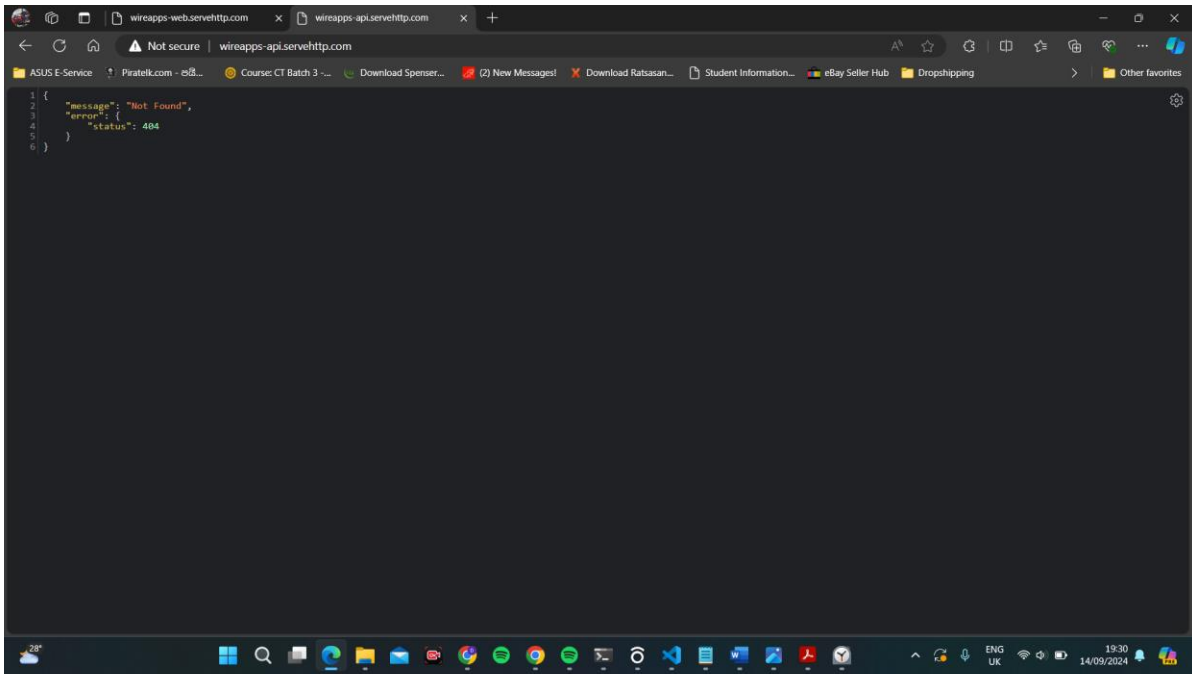
Web Test - wireapps-web.servehttp.com



wireapps-web.servehttp.com
wireapps-api.servehttp.com
Not secure
wireapps-web.servehttp.com

error running request to undefined/api/status

Api Test - wireapps-api.servehttp.com

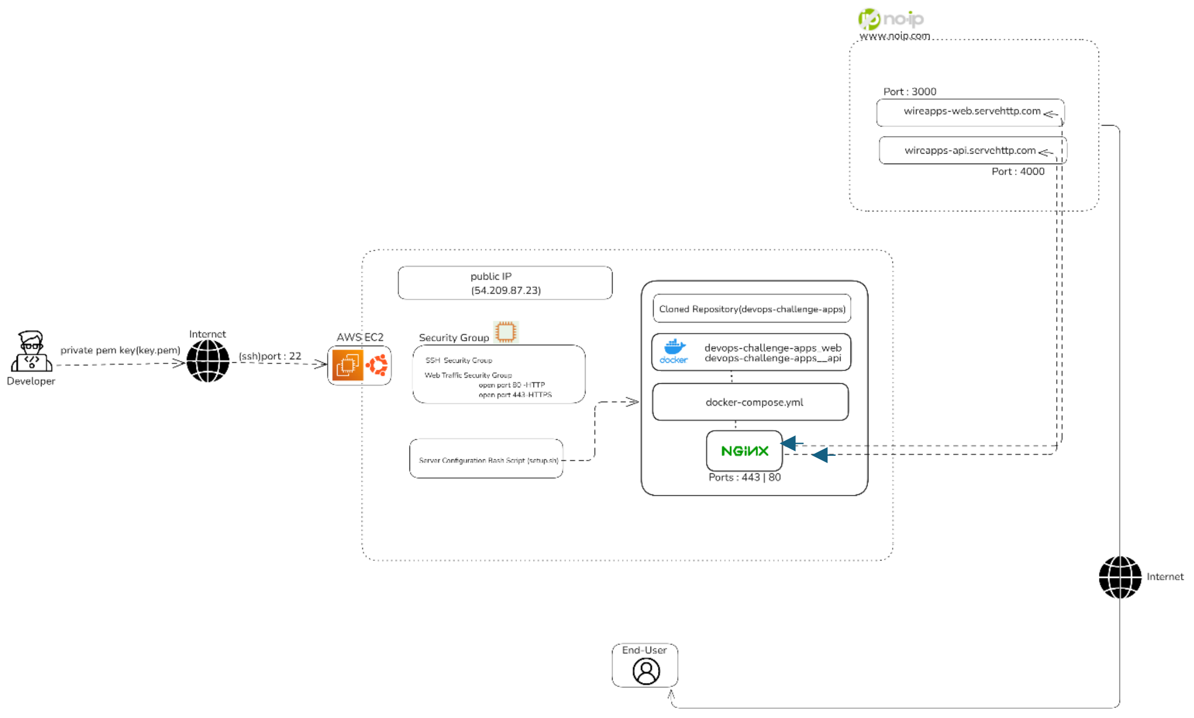


```
{
    "message": "Not Found",
    "error": {
        "status": 404
    }
}
```

*Note In this repository, there is a package build error that I attempted to fix, but the repository cannot be forked.

*Note In this colned repository has 2 docker-compose.yml files for API and Web that 2 docker compose files are both services were configured with the same port (5000) . Creating separate Dockerfiles for the Web and API services was essential to avoid port conflicts and to manage each service effectively. By assigning unique ports and configuring the Dockerfiles appropriately, the deployment process becomes more streamlined, and each service can be independently managed and scaled.

Step 3: Architecture Diagram and Documentation



Extra: Basic Automation with a Bash Script:

- **Write a bash script that updates content or configuration in one of the deployed apps.**

  Script Explanation:

  1. **Navigate to the project directory** containing the app.
  2. **Stop the web container** to apply the update.
  3. **Update the content/configuration** of the app (e.g., index.html or config.js).
  4. **Rebuild and restart** the Docker container to apply changes.

This is the bash script that updates the index.html file for the **web** app and restarts the service:

```
# Define variables

WEB_APP_DIR="/home/ubuntu/devops-challenge-apps/web"

DOCKER_COMPOSE_DIR="/home/ubuntu/devops-challenge-apps"


# Step 1: Navigate to the web app directory

cd $WEB_APP_DIR


# Step 2: Stop the web container

echo "Stopping the web container..."

sudo docker-compose -f $DOCKER_COMPOSE_DIR/docker-compose.yml stop web


# Step 3: Update the index.html file or any other content

echo "Updating index.html file..."

echo "<h1>Updated Web App</h1>" > $WEB_APP_DIR/public/index.html


# update configuration

# Example: Update config.js

# echo "export const API_URL = 'https://wireapps-api.servehttp.com/ ';" >
$WEB_APP_DIR/src/config.js


# Step 4: Rebuild and restart the web container

echo "Rebuilding and restarting the web container..."
```

```
sudo docker-compose -f $DOCKER_COMPOSE_DIR/docker-compose.yml up -d --
build web



# Step 5: Confirm update success

echo "Web app has been updated and restarted!"
```

Explaining what will happened in this script each and every step.

Step 1: Navigate to the Web App Directory

```
cd $WEB_APP_DIR
```

⍰ This command changes the current working directory to the folder where the web application files are located.

⍰ Before making changes to the web app, the script needs to be in the correct directory to access the necessary files, like index.html and config.js.

Step 2: Stop the Web Container

```
sudo docker-compose -f $DOCKER_COMPOSE_DIR/docker-compose.yml stop web
```

⍰ This command stops the web application Docker container.

⍰ Stopping the container ensures that we can safely update the files without interfering with the running app. Once the app is stopped, we can modify its content or configuration.

Step 3: Update the Web App Content

```
echo "<h1>Updated Web App</h1>" > $WEB_APP_DIR/public/index.html
```

⍰ This command updates the index.html file, which is the main page of the web app, by writing new content into it.

⍰ Changing the content in index.html allows us to modify what users will see when they visit the web app. This is useful for updating the look, text, or structure of the web page.

Update Configuration File (Optional)

```
# echo "export const API_URL = 'https://wireapps-api.servehttp.com/';" > $WEB_APP_DIR/src/config.js
```

Rebuild and Restart the Web Container

```
sudo docker-compose -f $DOCKER_COMPOSE_DIR/docker-compose.yml up -d --build web
```

- This command rebuilds the Docker container for the web app and restarts it in the background.
- After updating the app's content or configuration, we need to restart the container so the changes take effect. Rebuilding ensures that any modifications are included in the new version of the app.

Step 5: Confirm the Update

```
echo "Web app has been updated and restarted!"
```

This message is displayed to confirm that the web app has been successfully updated and is running again.

It provides feedback to let the user know that the process was completed without errors.

Explain how you would set up this script to run automatically every day using cron.

1. Open the Cron Scheduler

To schedule tasks, the cron program is used. First, open the crontab (cron's task list) by running:

```
crontab -e
```

2: Add the Cron Job

In the crontab file, add a new line to specify when and how often the script should run. For daily execution, use the following syntax:

```
0 0 * * * /bin/bash /script.sh
```

0 0 * * *: This means the script will run every day at midnight.

- 0 0: Hour and minute of execution (midnight).
- * * *: Every day, every month, every day of the week.

In this assessment project set as :

```
0 0 * * * /bin/bash /home/ubuntu/setup.sh
```

3: Save and Exit

Save the file and exit. In most text editors, you can save and close by pressing CTRL + X, followed by Y, and then Enter.

4: Verify the Cron Job

To confirm that the cron job was successfully added, you can list all cron jobs with:

```
crontab -l
```

It Works:

- The cron daemon will now run the script automatically every day at midnight.
- This script will update the web app content or configuration as defined and restart the Docker container.

Reference

Stack Overflow

The bug was discussed and a solution was provided in a Stack Overflow post titled "Docker errors.DockerException: Error while fetching server API version".

The solution from Stack Overflow suggested that this issue could be related to Python's `urllib3` version mismatch, which was resolved by reinstalling a compatible version.

Diagram design Tool: https://excalidraw.com