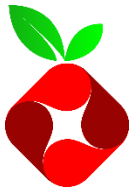


Docker DNS Server

Miraz Hossain | ID:181-15-10829

Project Submitted by:

MD. Miraz Hossain
ID: 181-15-10829
Section: A (O-1)
Department of Computer Science & Engineering

Project Submitted to:

Ms. Subhenur Latif
Assistant Professor
Faculty of Science & Information Technology
Daffodil International University

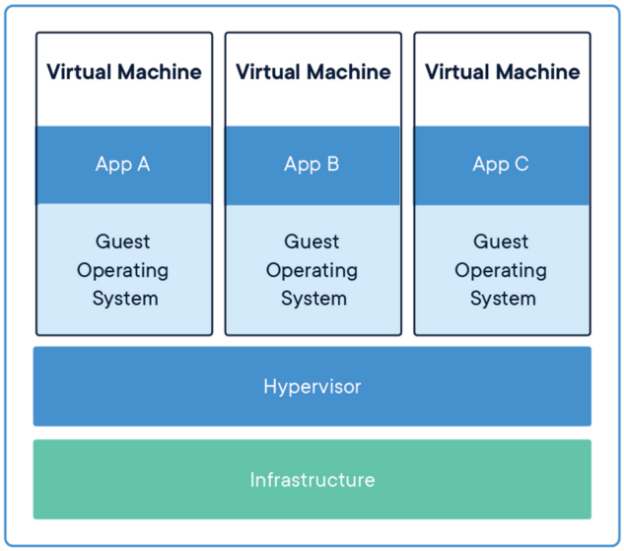
What I used to build Docker DNS Server:

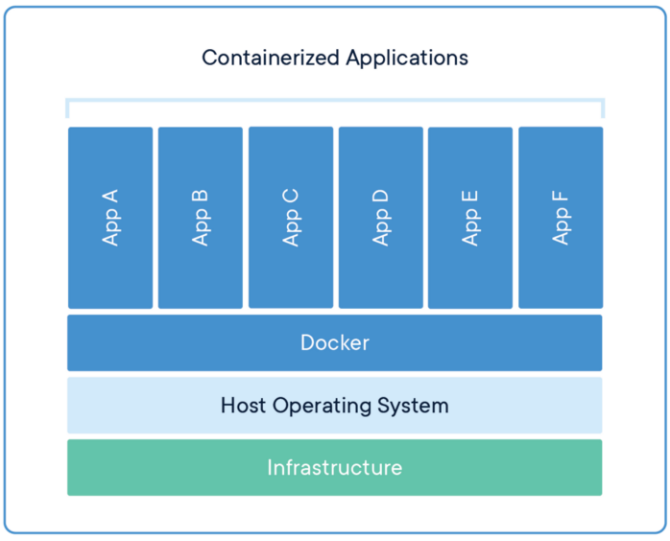
- Ubuntu Server 20.04
- Docker Engine
- Pi-Hole Docker Image
- Flask with Python

Why had I used Docker?

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all the parts it needs, such as libraries and other dependencies, and deploy it as one package.

Docker is a bit like a virtual machine. But unlike a virtual machine, rather than





creating a whole virtual operating system, Docker allows applications to use.

the same Linux kernel as the system uses its core kernel and only requires applications be shipped with things not already running on the host computer. This gives a huge performance boost and reduces the size of the application.

And docker help a host server to run multiple application simultaneously. I have OpenVPN Access Server & DNS Server on a single host. In order to run both of them docker helps a lot. Both servers are isolated from each other by docker container.

Why I build a Docker DNS Server?

Pi-hole is a DNS server with powerful ads blocking service. Instead of browser plugins or other software on each computer, Pi-hole in one place and entire network is protected.

Network-level blocking allows to block ads in non-traditional places such as mobile apps and smart TVs, regardless of hardware or OS.

Voice Assistant Integration: Voice assistant like Alexa, Google Assistant, Cortana allows block & unblock certain websites like social networks, Entertainment Sites, CDN networks (block server from DNS level to prevent media, database as well as API) etc.

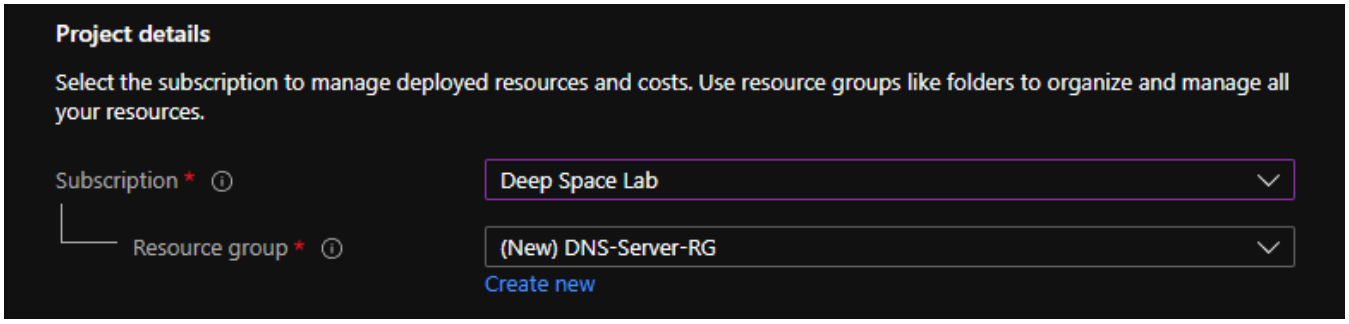
Server Configuration & Networking:

A DNS server cannot run on locally hosted VM to work networkwide such as router, smartphone, game console, TV etc. DNS server needs a public/external IP & always running so that any device or anyone can access it. Therefore, I used **Microsoft Azure** as cloud computing service.

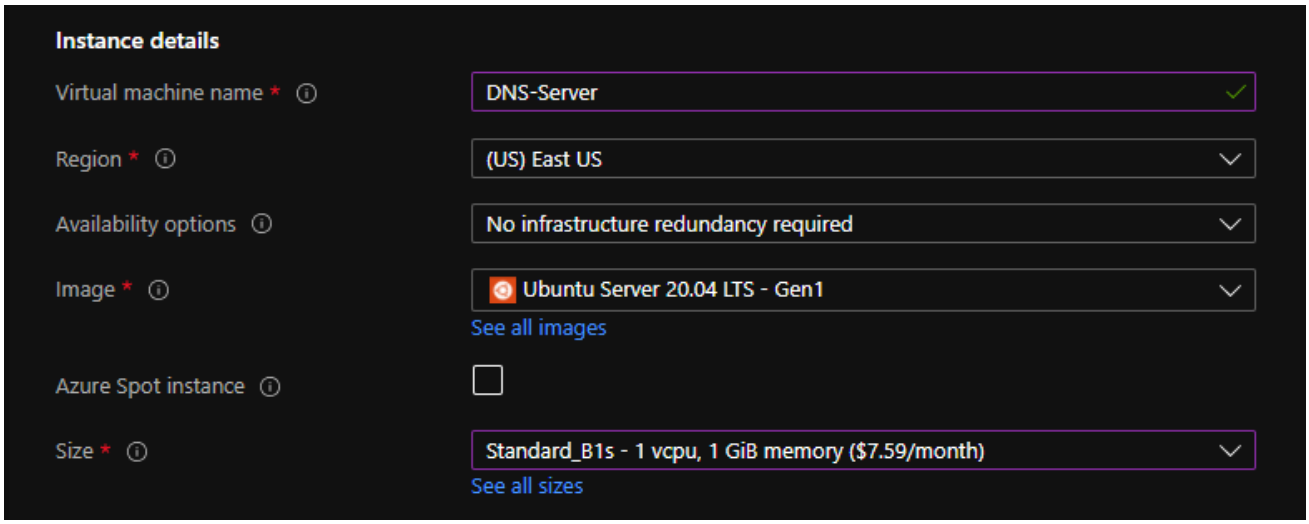
Microsoft Azure

Creating a virtual machine on azure:

Giving a resource group to my project:

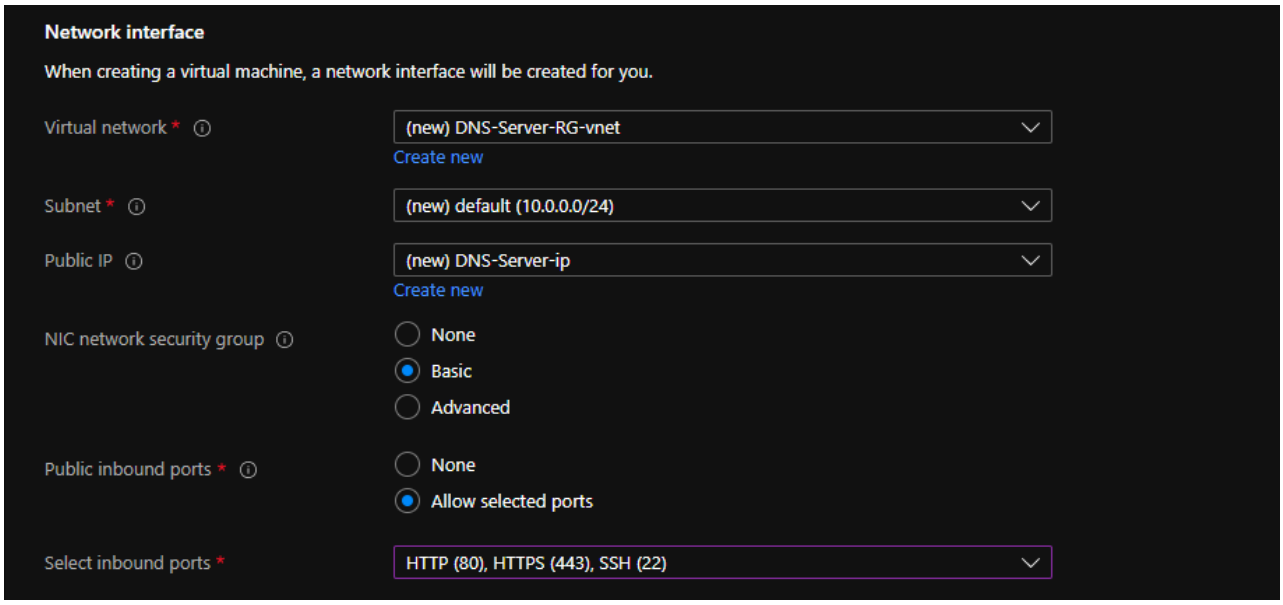


Creating an Ubuntu Server 20.04 instance:

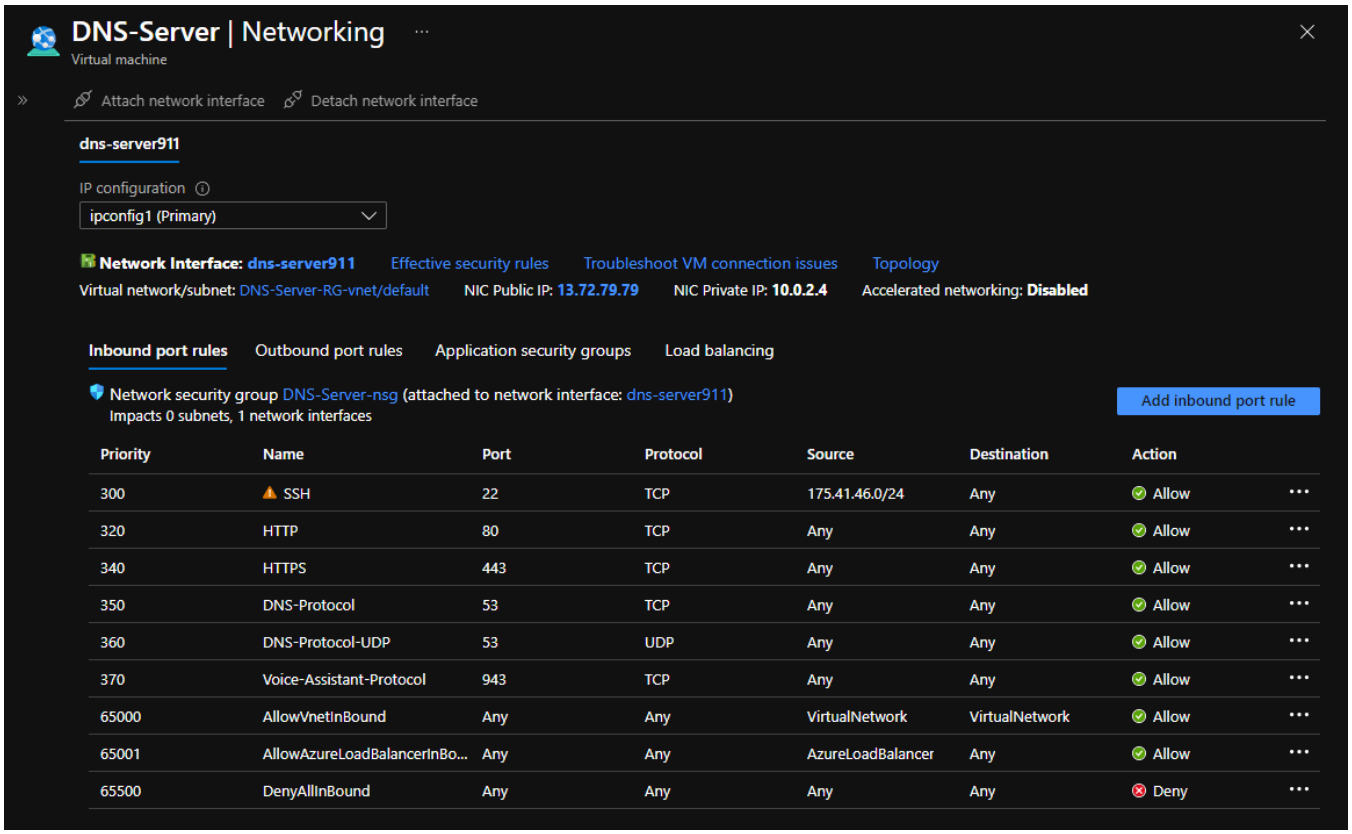


Configuring Network Interface:

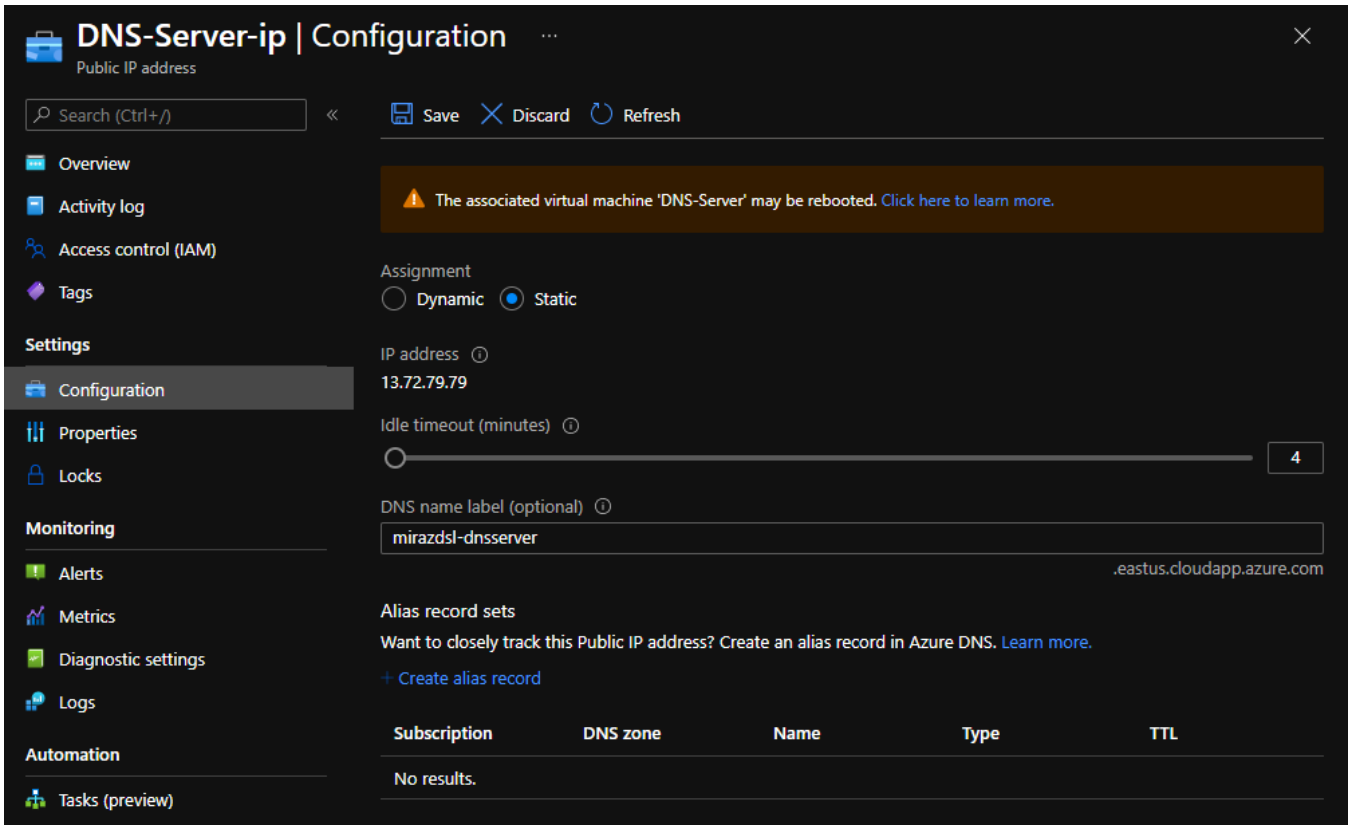
Giving a name to virtual network & creating a range of subnets for server:



Opening port **53** for DNS, **80/443** for web interface, **943** for call web request:



Giving a public/external static IP for DNS & DNS name level for web interface:



The server is now up & running and ready to install Docker Engine.
Now, logging in to server through PuTTY software using IP **13.72.79.79**

Prerequisite:

Ubuntu has a build-in local DNS resolver which can create conflict with pi-hole docker DNS resolver. To run pihole docker DNS without conflict local DNS resolver should be disabled. After that adding an external DNS such as 1.1.1.1 or 8.8.8.8 for local resolver through **"nano"** at **/etc/resolv.conf**

```
sudo systemctl stop systemd-resolved.service
```

```
sudo systemctl disable systemd-resolved.service
```

Installing Docker Engine on ubuntu server:

First, updating my existing list of packages:

```
sudo apt update
```

Next, installing a few prerequisite packages which let apt use packages over HTTPS:

```
sudo apt install apt-transport-https ca-certificates curl
software-properties-common
```

Then adding the GPG key for the official Docker repository to my system:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo apt-key add -
```

Adding the Docker repository to APT sources:

```
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu focal stable"
```

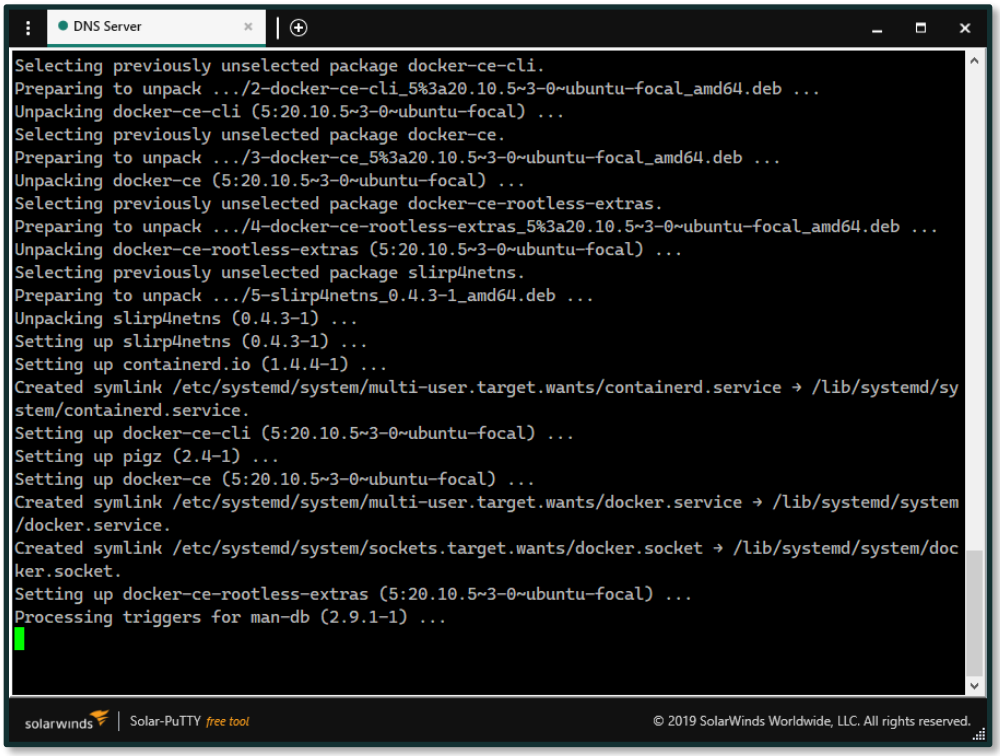
Next, updating the package database with the Docker packages from the newly added repository by **apt update.**

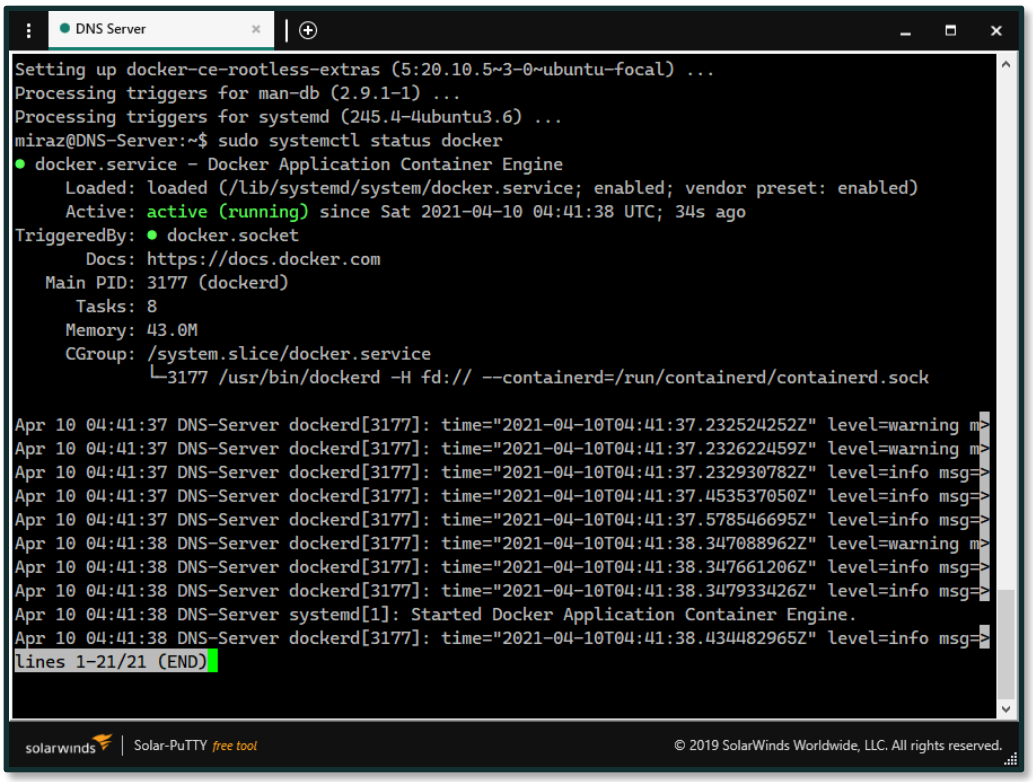
Finally, installing Docker Engine:

```
sudo apt install docker-ce
```

**Source & additional information of Docker is here

A snippet of docker engine from my ubuntu server:





Note: every snippet can be view as high-resolution by clicking on them.

Pulling Pi-Hole image from Docker Hub:

Creating a shell script file using nano to pull pi-hole image on my docker server:

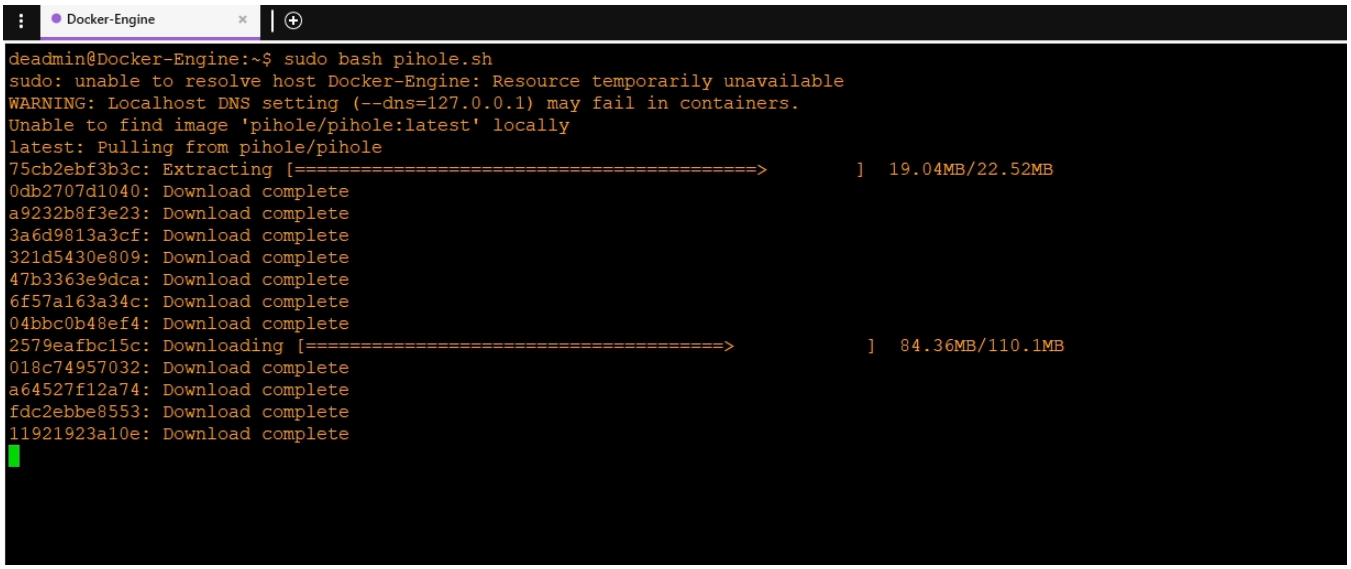
```
docker run -d \
    --name pihole \
    -p 53:53/tcp -p 53:53/udp \
    -p 80:80 \
    -p 443:443 \
    -p 943:943 \
    -e TZ="America/New_York" \
    -v "$(pwd)/etc-pihole/:/etc/pihole/" \
    -v "$(pwd)/etc-dnsmasq.d/:/etc/dnsmasq.d/" \
    --dns=127.0.0.1 --dns=1.1.1.1 \
    --restart=unless-stopped \
    --hostname pi.hole \
    -e VIRTUAL_HOST="pi.hole" \
    -e PROXY_LOCATION="pi.hole" \
    -e ServerIP="13.92.153.59" \
    pihole/pihole:latest

printf 'Starting up pihole container '
for i in $(seq 1 20); do
    if [ "$(docker inspect -f "{{.State.Health.Status}}" pihole)" == "healthy" ] ; then
        printf ' OK'
        echo -e "\n$(docker logs pihole 2> /dev/null | grep 'password:') for your pi-hole:
https://${IP}/admin/"
        exit 0
    else
        sleep 3
        printf '.'
    fi

    if [ $i -eq 20 ] ; then
        echo -e "\nTimed out waiting for Pi-hole start, consult container logs for more
info (\`docker logs pihole\`)"
        exit 1
    fi
done;
```

Raw code also can be found here

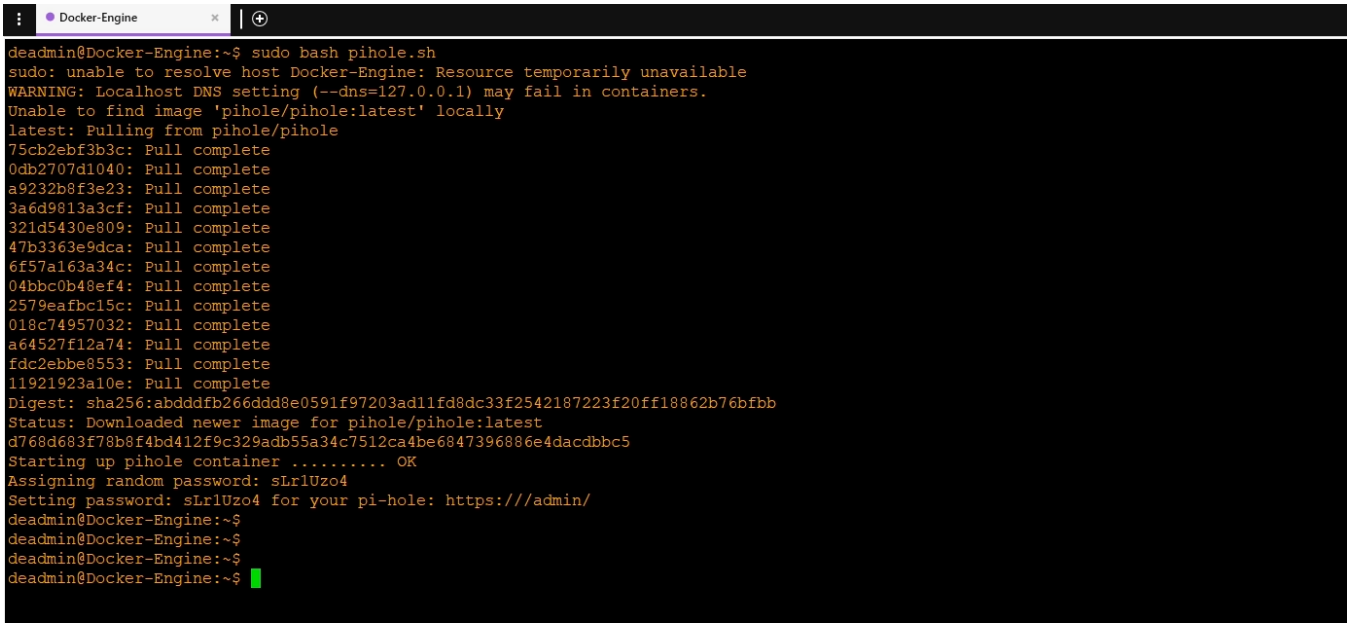
Next, giving bash command to shell script:

```
sudo bash pihole.sh
```

Now, the docker pi-hole image is pulling from docker hub:



The server is now ready.



Logging-in to pi-hole container:

Firstly, to log-in to pihole container I must use docker commands:

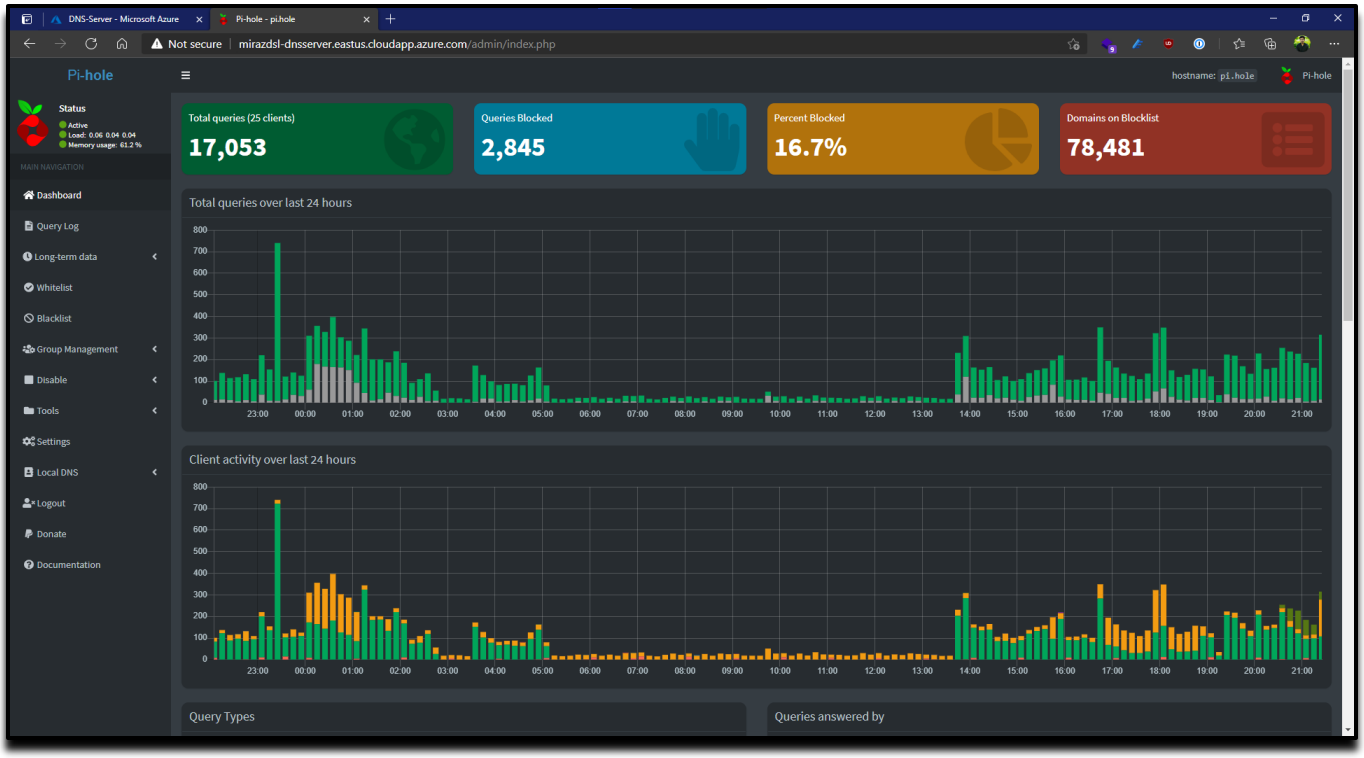
```
sudo docker exec -it pihole bash
```

Now changing pi-hole container's random created password:

```
pihole -a -p
```

After that, accessing our DNS server from public internet:

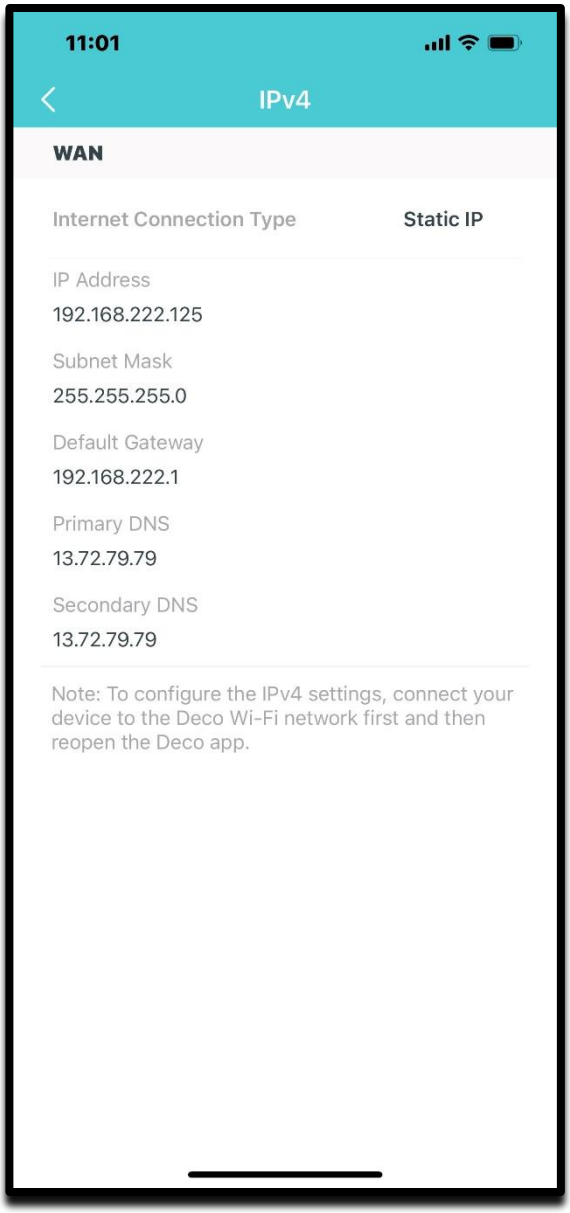
http://mirazdsl-dnsserver.eastus.cloudapp.azure.com/admin

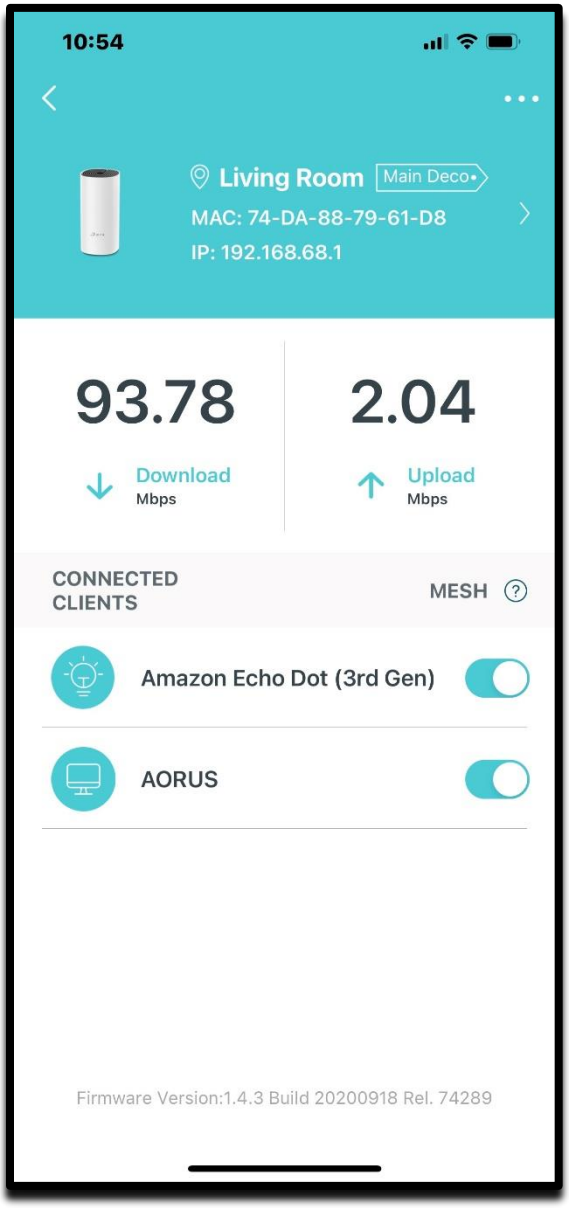


**Source & additional information of Pi-hole is here

Configuring DNS on TP-Link router:

The address of my DNS server is: **13.72.79.79**





Now it is working fine & blocking annoying ads on websites.

Allowing DNS server to Interact with virtual assistant:

Using Flask Framework (python), I opened a port (943) on my public IP specially to get call-request:

```
import time
import os
from flask import Flask, render_template, request
app = Flask(__name__)
@app.route("/<deviceName>/")
def action(deviceName):
    if deviceName != 'monstermash':
        if deviceName == 'block':
            os.system('./blockregex.sh')
        if deviceName == 'unblock':
            os.system('./unblockregex.sh')

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=943)
```

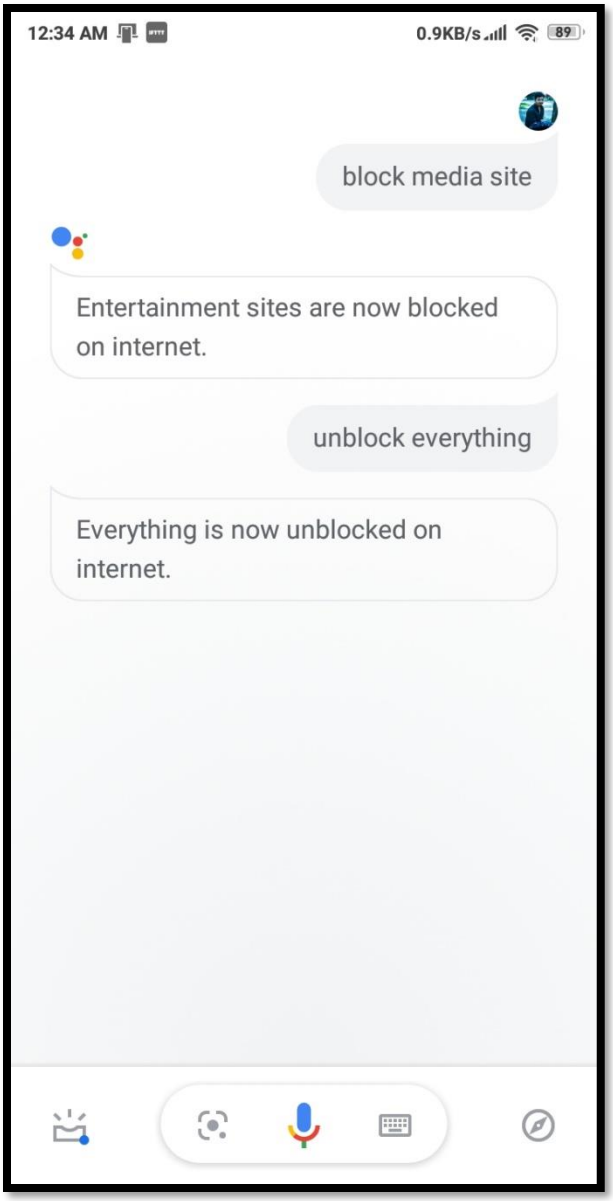
Raw code also can be found here: show me

When Flask get a call-request on **"13.72.79.79:943/block or /unblock"** it will execute a bash command on a specific shell file. And those files are coded with pi-hole regex filter.

- blockregex.sh - Raw code can be found on blob-storage here
- unblockregex.sh - Raw code can be found on blob-storage here

[Left] Configuring IFTTT to post a call-request from Google Assistant.

[Right] giving commands to google assistant to block certain media sites which is coded on blockregex.sh & unblockregex.sh



GitHub

https://github.com/Miraz4300/docker-dns-server

LinkedIn

https://www.linkedin.com/in/miraz4300/

Google Drive

server configuration & adding ad list to DNS server video from scratch:

https://drive.google.com/drive/folders/1belifgjosouKesDP9rmG6lTgIN7hN0Vr?usp=sharing

**This project is made possible with Microsoft Azure, Ubuntu Server, Docker, Pi-hole, Python, Flask & IFTTT.

THANKS

author: miraz hossain