You got it! Here's a complete set of files, including the deploy.sh script, app.py, config.json, index.html, and main.js, which are all needed for your application detection system to work.

**1. deploy.sh (Deployment Script)**

#!/bin/bash

# --- User Input ---

# Ask for domain name (optional)

read -p "Enter your domain name (leave blank if not using a domain): " domain\_name

# Ask for the GitHub repository URL

read -p "Enter your GitHub repository URL: " repo\_url

# --- Install Packages ---

sudo apt update

sudo apt install apache2 python3-pip npm git -y

# Install Vue.js CLI

sudo npm install -g @vue/cli

# --- Create Project Directories ---

mkdir -p /var/www/html/app-detection/frontend

# --- Install Flask and Dependencies ---

cd /var/www/html/app-detection

pip3 install flask sqlite3

# --- Install Vue.js Project ---

cd frontend

vue create .

# --- Copy Code from GitHub ---

# Flask application code

cd /var/www/html/app-detection

git clone $repo\_url

mv $repo\_url/\* .

rm -rf $repo\_url

# Vue.js template

cd frontend

wget -O public/index.html https://raw.githubusercontent.com/your-github-username/your-repo/main/index.html

wget -O src/main.js https://raw.githubusercontent.com/your-github-username/your-repo/main/main.js

# --- Configure Apache ---

sudo a2enmod rewrite headers

# Create a virtual host (if a domain name was provided)

if [[ -n "$domain\_name" ]]; then

sudo tee /etc/apache2/sites-available/$domain\_name.conf <<EOF

<VirtualHost \*:80>

ServerName $domain\_name

ServerAlias www.$domain\_name

DocumentRoot /var/www/html/app-detection

<Directory /var/www/html/app-detection>

Options Indexes FollowSymLinks

AllowOverride All

Require all granted

</Directory>

</VirtualHost>

EOF

sudo a2ensite $domain\_name.conf

fi

# Restart Apache

sudo systemctl restart apache2

# --- Build Vue.js Application ---

cd frontend

npm run build

# --- Move Vue.js Build to Apache Directory ---

mv dist/\* /var/www/html/app-detection

# --- Set up nftables Table ---

sudo nft add table inet filter {

prerouting {

type filter hook prerouting priority filter; policy accept;

}

}

# --- Start the Flask Application ---

cd /var/www/html/app-detection

nohup python3 app.py &

# --- Success Message ---

echo "Application detection system deployed successfully!"

**2. app.py (Flask Application)**

from flask import Flask, render\_template, request, jsonify

import subprocess

import json

import sqlite3

app = Flask(\_\_name\_\_)

# Database setup

db = sqlite3.connect("app\_config.db")

cursor = db.cursor()

# Create table if it doesn't exist

cursor.execute("""

CREATE TABLE IF NOT EXISTS application\_rules (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

interface TEXT NOT NULL

)

""")

db.commit()

@app.route("/")

def index():

return render\_template("index.html")

@app.route("/get\_applications")

def get\_applications():

cursor.execute("SELECT name, interface FROM application\_rules")

applications = cursor.fetchall()

return jsonify(applications)

@app.route("/add\_rule", methods=["POST"])

def add\_rule():

name = request.form["name"]

interface = request.form["interface"]

cursor.execute(

"INSERT INTO application\_rules (name, interface) VALUES (?, ?)",

(name, interface),

)

db.commit()

return "Rule added successfully"

@app.route("/delete\_rule", methods=["POST"])

def delete\_rule():

id = request.form["id"]

cursor.execute("DELETE FROM application\_rules WHERE id = ?", (id,))

db.commit()

return "Rule deleted successfully"

@app.route("/update\_rules")

def update\_rules():

applications = request.get\_json()

for app in applications:

cursor.execute(

"UPDATE application\_rules SET interface = ? WHERE name = ?",

(app["interface"], app["name"]),

)

db.commit()

return "Rules updated successfully"

@app.route("/get\_ports")

def get\_ports():

# Load port mapping from config.json

with open("config.json", "r") as f:

config = json.load(f)

ports = config.get("ports", {})

return jsonify(ports)

@app.route("/get\_interfaces")

def get\_interfaces():

# Get available interfaces (replace with appropriate system commands)

interfaces = subprocess.check\_output(

["ip", "-o", "link", "show"], text=True

).splitlines()

interfaces = [

interface.split()[1].strip(":")

for interface in interfaces

if "state UP" in interface

]

return jsonify(interfaces)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**3. config.json (Configuration File)**

{

"ports": {

"http": 80,

"https": 443,

"facebook": 443,

"youtube": 443,

"google": 443,

"outlook": 443,

"twitter": 443,

"instagram": 443,

"whatsapp": 443,

"netflix": 443,

"spotify": 443,

"amazon": 443,

"ebay": 443,

"linkedin": 443,

"zoom": 443,

"skype": 443,

"discord": 443,

"twitch": 443,

"reddit": 443,

"tiktok": 443,

"snapchat": 443,

"dropbox": 443,

"google\_drive": 443,

"onedrive": 443,

"vimeo": 443,

"pinterest": 443,

"slack": 443,

"telegram": 443,

"signal": 443,

"github": 443,

"gitlab": 443,

"bitbucket": 443,

"stackoverflow": 443,

"wordpress": 80,

"tumblr": 80,

"medium": 80,

"quora": 80,

"reddit": 80

}

}

**4. index.html (Vue.js Template)**

<!DOCTYPE html>

<html>

<head>

<title>Application Detection</title>

<script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

</head>

<body>

<div id="app">

<h1>Application Detection</h1>

<div v-if="loading">Loading...</div>

<div v-else>

<div class="add-rule">

<h2>Add New Rule</h2>

<label for="app-name">Application Name:</label>

<input type="text" id="app-name" v-model="newRule.name">

<label for="app-interface">Interface:</label>

<select id="app-interface" v-model="newRule.interface">

<option v-for="interface in interfaces" :value="interface">{{ interface }}</option>

</select>

<button @click="addRule">Add Rule</button>

</div>

<div class="rule-list">

<h2>Current Rules</h2>

<div v-for="(rule, index) in rules" :key="index">

<span>{{ rule.name }} is routed to {{ rule.interface }}</span>

<button @click="deleteRule(index)">Delete</button>

</div>

</div>

</div>

</div>

<script>

const app = new Vue({

el: '#app',

data: {

rules: [],

newRule: {

name: '',

interface: ''

},

interfaces: [],

loading: true

},

mounted() {

this.getApplications();

this.getInterfaces();

},

methods: {

getApplications() {

fetch('/get\_applications')

.then(response => response.json())

.then(data => {

this.rules = data;

this.loading = false;

});

},

getInterfaces() {

fetch('/get\_interfaces')

.then(response => response.json())

.then(data => {

this.interfaces = data;

});

},

addRule() {

fetch('/add\_rule', {

method: 'POST',

headers: {

'Content-Type': 'application/x-www-form-urlencoded'

},

body: `name=${this.newRule.name}&interface=${this.newRule.interface}`

})

.then(response => {

if (response.ok) {

this.getApplications();

this.newRule.name = '';

this.newRule.interface = '';

} else {

console.error('Error adding rule');

}

});

},

deleteRule(index) {

const ruleId = this.rules[index].id; // Assuming you have an ID for each rule

fetch('/delete\_rule', {

method: 'POST',

headers: {

'Content-Type': 'application/x-www-form-urlencoded'

},

body: `id=${ruleId}`

})

.then(response => {

if (response.ok) {

this.getApplications();

} else {

console.error('Error deleting rule');

}

});

}

}

});

</script>

</body>

</html>

**5. main.js (Vue.js Entry Point)**

new Vue({

el: '#app',

data: {

rules: [],

newRule: {

name: '',

interface: ''

},

interfaces: [],

loading: true

},

mounted() {

this.getApplications();

this.getInterfaces();

},

methods: {

getApplications() {

fetch('/get\_applications')

.then(response => response.json())

.then(data => {

this.rules = data;

this.loading = false;

});

},

getInterfaces() {

fetch('/get\_interfaces')

.then(response => response.json())

.then(data => {

this.interfaces = data;

});

},

addRule() {

fetch('/add\_rule', {

method: 'POST',

headers: {

'Content-Type': 'application/x-www-form-urlencoded'

},

body: `name=${this.newRule.name}&interface=${this.newRule.interface}`

})

.then(response => {

if (response.ok) {

this.getApplications();

this.newRule.name = '';

this.newRule.interface = '';

} else {

console.error('Error adding rule');

}

});

},

deleteRule(index) {

const ruleId = this.rules[index].id;

fetch('/delete\_rule', {

method: 'POST',

headers: {

'Content-Type': 'application/x-www-form-urlencoded'

},

body: `id=${ruleId}`

})

.then(response => {

if (response.ok) {

this.getApplications();

} else {

console.error('Error deleting rule');

}

});

}

}

})

**Deployment:**

1. **Create a GitHub Repository:** Create a new GitHub repository and push these files to it.
2. **Run deploy.sh:** On your Debian Buster server, make sure you have internet access, and run sudo ./deploy.sh.
3. **Input Information:** The script will prompt you for your domain name (optional) and GitHub repository URL.
4. **Test:** After deployment, access your web interface through http://your-server-ip or http://your-domain.com to configure rules.

Remember:

* **Replace Placeholders:** In the script and files, update "your-github-username" and "your-repo" with your actual GitHub account and repository names.
* **Security:** Implement security measures and consider a database like PostgreSQL instead of SQLite for a more robust system.

# STEPS

**Step 1: Prepare Your Server**

1. **Log in to your server:** Use SSH to log in as root or a user with sudo privileges.
2. **Ensure Internet Access:** Make sure your server has a working internet connection. You will need to download packages and files.
3. **Create a directory (optional):** You can create a directory to store your project files. For example:

mkdir /var/www/html/app-detection

**Step 2: Prepare the GitHub Repository**

1. **Create a GitHub Repository:** Create a new public GitHub repository for your project.
2. **Add Files:** Add the following files to your repository:
   * deploy.sh
   * app.py
   * config.json
   * index.html
   * main.js
3. **Commit and Push:** Commit your changes and push them to your GitHub repository.

**Step 3: Deploy the Application (Using deploy.sh)**

1. **Download the Script:** Download the deploy.sh script from your GitHub repository.

wget https://raw.githubusercontent.com/your-github-username/your-repo/main/deploy.sh

1. **Make it Executable:** Give the script executable permissions:

chmod +x deploy.sh

1. **Run the Script:** Execute the deployment script:

sudo ./deploy.sh

1. **Provide Input:** The script will ask you for:
   * **Domain Name:** Enter your domain name if you have one. If not, press enter to skip.
   * **GitHub Repository URL:** Paste the URL of your GitHub repository.
2. **Wait for Deployment:** The script will automatically install packages, download files, configure Apache, build Vue.js, and start the Flask application.

**Step 4: Access and Configure**

1. **Access the Web Interface:** Once the deployment is complete, open your web browser and navigate to:
   * http://your-server-ip if you didn't use a domain name.
   * http://your-domain.com if you provided a domain name during deployment.
2. **Configure Rules:** Use the graphical interface to add, delete, and update rules for the applications you want to monitor and route.
3. **Monitor Rules:** The dashboard will display the current rules being applied.

**Important Notes:**

* **GitHub Repository:** Ensure your GitHub repository is publicly accessible.
* **Firewall:** If you have a firewall configured, make sure it allows traffic to the web server on port 80 (HTTP).
* **Testing:** Thoroughly test your deployment to verify that everything is working correctly.
* **Security:** Always prioritize security. Implement authentication and authorization for your web interface to prevent unauthorized access.