



Azure-Based Multi-Layered DDoS Prevention System

This guide walks you step-by-step through building a **3-layer DDoS defense system** using **Azure services, Python (Flask + ML)**, and a **honeypot + CAPTCHA trap**. Built to work within **Azure's free/student tier**.

📁 Project Structure

- Layer 1: Azure WAF (Application Gateway)
- Layer 2: ML Anomaly Detection (Flask API)
- Layer 3: Honeypot + reCAPTCHA

_LAYER 1: Web Application Firewall (WAF)

✓ Step 1: Create a Resource Group

1. Search: Resource Groups → + Create
2. Fill in details:
 - Name: DDoS-RG
 - Region: e.g., Norway East

Create a resource group ...

Basics Tags Review + create

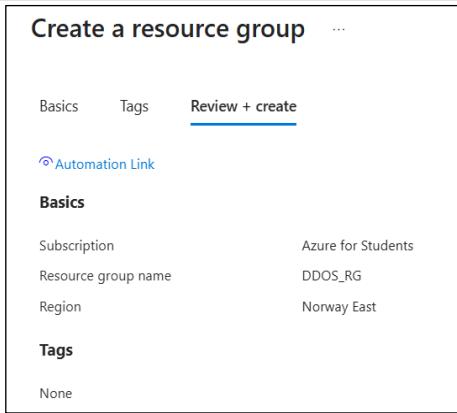
Automation Link

Basics

Subscription	Azure for Students
Resource group name	DDOS_RG
Region	Norway East

Tags

None



✓ Step 2: Create a Virtual Network (VNet)

1. Search: Virtual Networks → + Create
2. Fill:
 - Name: DDoS-VNet
 - Subnet 1: default
 - Address range: leave as default
3. Click Review + Create

Create virtual network ...

Basics Security IP addresses Tags Review + create

Basics

Subscription	Azure for Students
Resource Group	DDOS_RG
Name	DDOS_VN
Region	Norway East

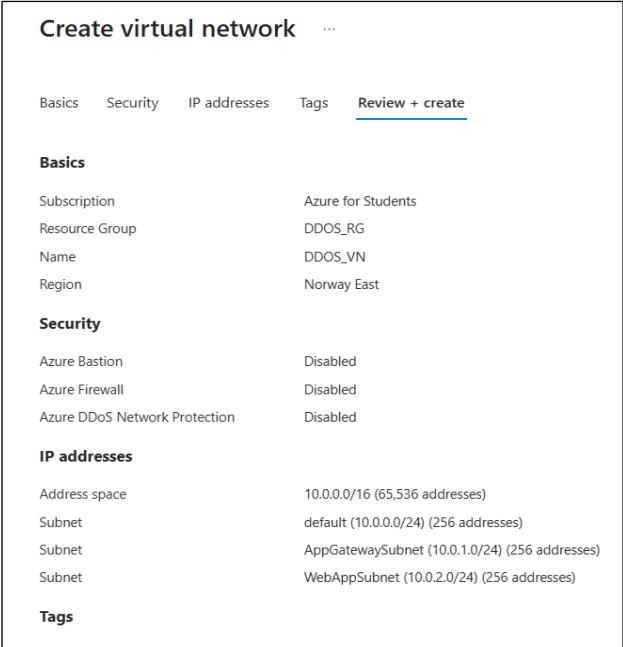
Security

Azure Bastion	Disabled
Azure Firewall	Disabled
Azure DDoS Network Protection	Disabled

IP addresses

Address space	10.0.0.0/16 (65,536 addresses)
Subnet	default (10.0.0.0/24) (256 addresses)
Subnet	AppGatewaySubnet (10.0.1.0/24) (256 addresses)
Subnet	WebAppSubnet (10.0.2.0/24) (256 addresses)

Tags



Step 3: Deploy Azure Virtual Machine

1. Search: Virtual Machines → + Create
2. Fill in:
 - VM Name: DDoS-VM
 - Region: same as above
 - Image: Ubuntu 22.04 LTS
 - Size: B1s (free)
 - Username: azureuser
 - Authentication: SSH or Password
3. Download the .pem file if SSH is chosen

Basics Subscription: Azure for Students Resource group: DDOS_RG Virtual machine name: WebVM Region: Norway East Availability options: Availability zone Zone options: Self-selected zone Availability zone: 1 Security type: Trusted launch virtual machines Enable secure boot: Yes Enable vTPM: Yes Integrity monitoring: No Image: Ubuntu Server 22.04 LTS - Gen2 VM architecture: x64 Size: Standard B1s (1 vcpu, 1 GiB memory) Enable Hibernation: No Authentication type: SSH public key Username: azureuser SSH Key format: RSA Key pair name: WebVM_key Public inbound ports: SSH, HTTP Azure Spot: No	Disks OS disk size: Image default OS disk type: Standard SSD LRS Use managed disks: Yes Delete OS disk with VM: Enabled Ephemeral OS disk: No	Networking Virtual network: DDOS_VN Subnet: WebAppSubnet (10.0.2.0/24) Public IP: (new) WebVM-ip Accelerated networking: Off Place this virtual machine behind an existing load balancing solution?: No Delete public IP and NIC when VM is deleted: Disabled	Monitoring Alerts: Off Boot diagnostics: On Enable OS guest diagnostics: Off Enable application health monitoring: Off	Advanced Extensions: None VM applications: None Cloud init: No User data: No Disk controller type: SCSI Proximity placement group: None Capacity reservation group: None
--	---	--	---	--

Step 4: Connect via SSH & Install Nginx

```
ssh -i path-to-key.pem azureuser@<VM_PUBLIC_IP>
sudo apt update && sudo apt install nginx -y
```

Verify by visiting:

http://<VM_PUBLIC_IP> → You should see the Nginx welcome page.

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

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

azureuser@WebVM:~$
```

✓ Step 5: Set Up Application Gateway + WAF

1. Search: Application Gateway → + Create
2. Choose:
 - Tier: WAF V2
 - Associate with previously created VNet and subnet
 - Create a public IP
3. Attach the backend VM IP to the App Gateway backend pool

Basics	
Subscription	Azure for Students
Resource group	DDoS_RG
Name	DDoSAppGateway
Region	Norway East
Tier	WAF_V2
Enable autoscaling	Enabled
Minimum instance count	2
Maximum instance count	10
WAF status	Enabled
WAF mode	Detection
Availability zone	Zones 1, 2, 3
HTTP2	Enabled
Virtual network	DDOS_VN
Subnet	AppGatewaySubnet (10.0.1.0/24) (new) DDoS_WAF
Web application firewall	
Frontends	
Public IPv4 address name	AppGatewayIP
SKU	Standard
Assignment	Static
Availability zone	ZoneRedundant

✓ Step 6: Configure WAF Policy

1. Search: WAF Policies
2. Select: DDOS_WAF
3. Enable OWASP Rule Set 3.2
4. Add Custom Rules:
 - **Block IP Rule:**
 - Rule Name: IPBlock
 - Match: RemoteAddr == <ip to block>
 - Action: Block
 - **Geo Block Rule:**
 - Match: GeoLocation == Russia
 - Action: Block

Click Save

The screenshot shows two side-by-side custom rule configurations in the Azure portal:

Custom rule name *: IPBlock

Enable rule:

Rule type: Match Rate limit

Priority *: 100

Conditions

If

Match type: IP address
Operation: Does contain
IP address or range: 123.123.123.123

Then: Deny traffic

Custom rule name *: GeoMatch

Enable rule:

Rule type: Match Rate limit

Priority *: 99

Conditions

If

Match type: Geo location
Match variables: RemoteAddr
Operation: Is
Country/Region: Russia

Then: Deny traffic

✓ Layer 1 Complete

Layer 2: Machine Learning Detection API

Step 1: Install Requirements (via SSH on VM)

```
sudo apt update  
sudo apt install python3-pip git -y  
pip3 install flask scikit-learn pandas numpy
```

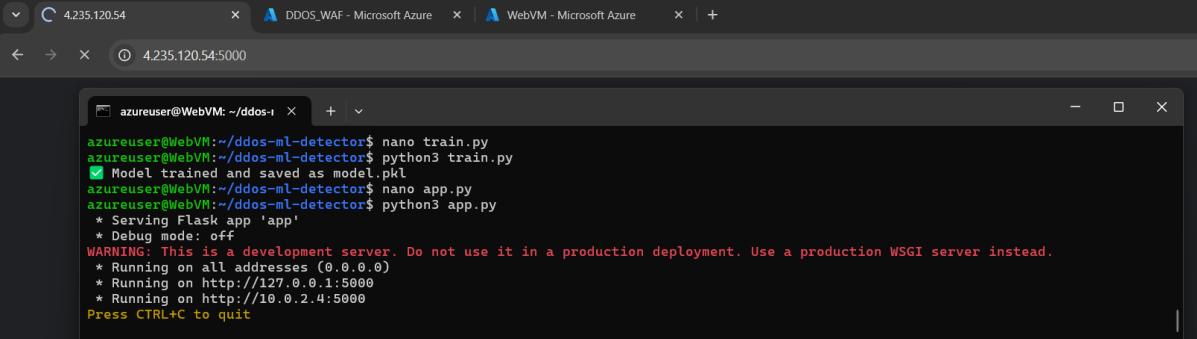
Step 2: Build the ML Detector

```
mkdir ddos-ml-detector  
cd ddos-ml-detector  
nano train.py
```

Paste code from [GitHub](#) (ML model with RandomForest)

Then run:

```
python3 train.py
```



```
azureuser@WebVM:~/ddos-ml-detector$ nano train.py  
azureuser@WebVM:~/ddos-ml-detector$ python3 train.py  
✓ Model trained and saved as model.pkl  
azureuser@WebVM:~/ddos-ml-detector$ nano app.py  
azureuser@WebVM:~/ddos-ml-detector$ python3 app.py  
* Serving Flask app 'app'  
* Debug mode: off  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on all addresses (0.0.0.0)  
* Running on http://127.0.0.1:5000  
* Running on http://10.0.2.4:5000  
Press CTRL+C to quit
```

Step 3: Create and Run Flask API

```
nano app.py
```

Paste code from [GitHub](#) (Flask endpoint using model.pkl)

Run it:

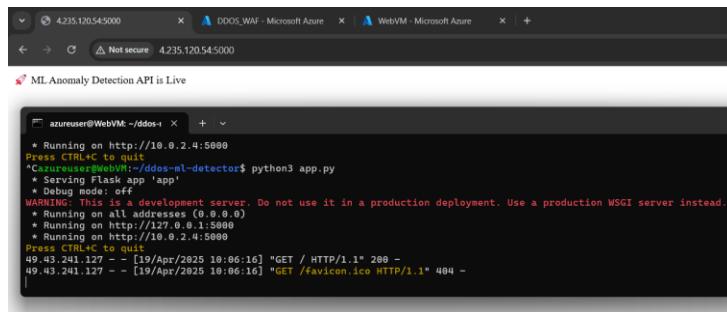
```
python3 app.py
```

✓ Step 4: Open Port 5000 in Azure

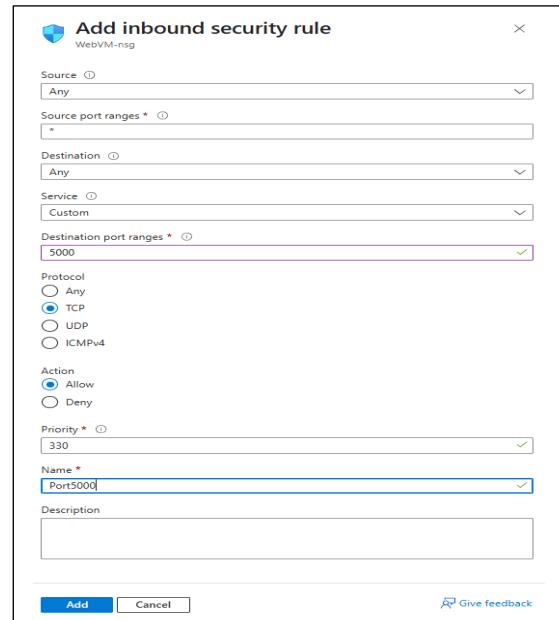
1. Go to VM > Networking > Inbound Rules
2. Add rule:
 - o Port: 5000
 - o Protocol: TCP
 - o Source: Any

Restart app:

```
python3 app.py
```



```
azuser@WebVM:~/ddos$ * Running on http://10.0.2.4:5000
Press CTRL+C to quit
azuser@WebVM:~/ddos$ ./dos-ml-detector$ python3 app.py
* Serving Flask app "app"
* Using Debug Mode (WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.0.2.4:5000
Press CTRL+C to quit
[9-Apr/2025 10:06:16] "GET / HTTP/1.1" 200 -
[9-Apr/2025 10:06:16] "GET /favicon.ico HTTP/1.1" 404 -
```



✓ Layer 2 Complete

⌚ Layer 3: Honeypot + reCAPTCHA

✓ Step 1: SSH into the VM (same)

```
cd ~
mkdir honeypot
cd honeypot
pip3 install flask
nano honeypot.py
```

Paste honeypot code from [GitHub](#).

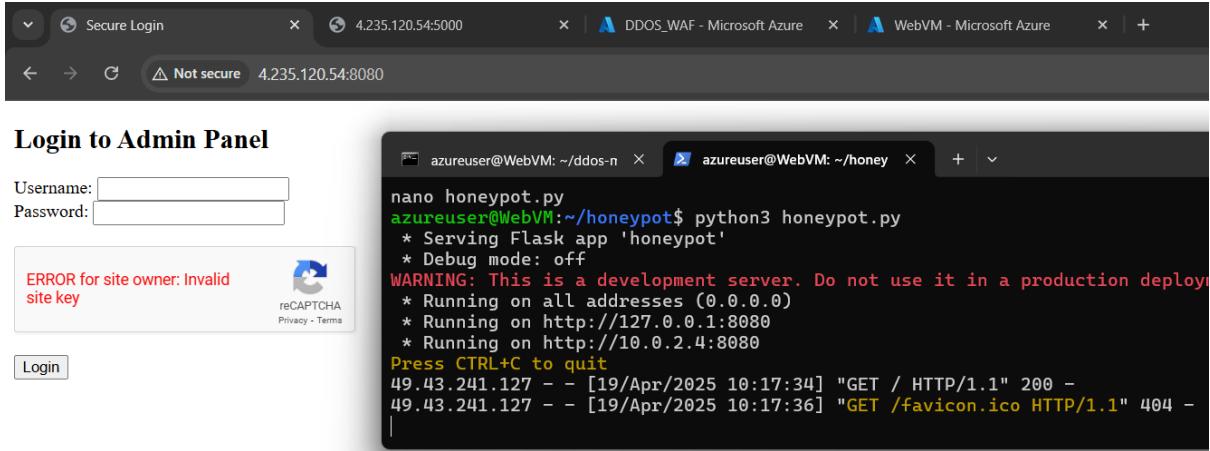
✓ Step 2: Run the Honeypot

```
python3 honeypot.py
```

Step 3: Open Port 8080 (if used)

1. Go to VM > Networking
2. Add Inbound Rule for Port 8080

Now open http://<VM_IP>:8080



The screenshot shows a browser window titled "Secure Login" at address "4.235.120.54:5000". It displays a login form with fields for "Username" and "Password". Below the form, a red error message says "ERROR for site owner: Invalid site key". To the right of the browser is a terminal window titled "azurouser@WebVM: ~/honey" showing the command "python3 honeypot.py" running. The terminal output includes a warning about the development server, log entries for two GET requests, and a prompt to press Ctrl+C to quit.

Step 4: Setup reCAPTCHA

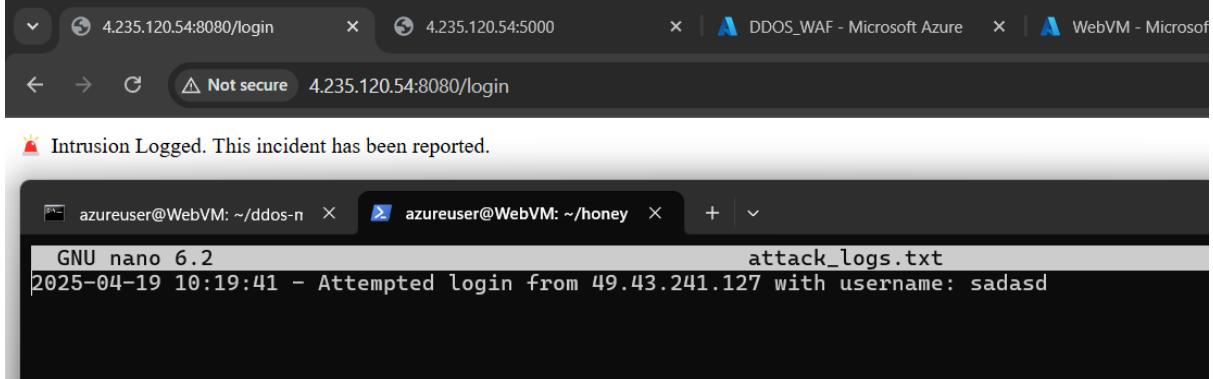
1. Go to: <https://www.google.com/recaptcha/admin>
2. Register your site:
 - o Type: reCAPTCHA v2 ("I'm not a robot")
 - o Add your VM IP as a domain
3. Add your **site key** and **secret key** in HTML + Flask code

Step 5: View Logs

`cat honeypot/attack_logs.txt`

You'll see:

203.0.113.45 tried at 2025-04-19 19:32:22



The screenshot shows a browser window titled "4.235.120.54:login" at address "4.235.120.54:8080/login". Below it is another browser window titled "4.235.120.54:5000". To the right is a terminal window titled "azurouser@WebVM: ~/honey" showing the file "attack_logs.txt" open in nano editor. The log file contains a single entry: "2025-04-19 10:19:41 - Attempted login from 49.43.241.127 with username: sadasd".

 Layer 3 Complete

Testing Checklist

Layer 1:

- **IP Block Test:** Block mobile IP → visit site → should return **403**
- **Geo Block Test:** Connect VPN to Russia → visit → get **403**

Layer 2:

- Predict via curl → should return:
{"is_anomalous": true}

Layer 3:

- Visit honeypot page → fake login → check attack_logs.txt
-