

Beginner Firewall Project (UFW)

Step-by-Step with Visuals

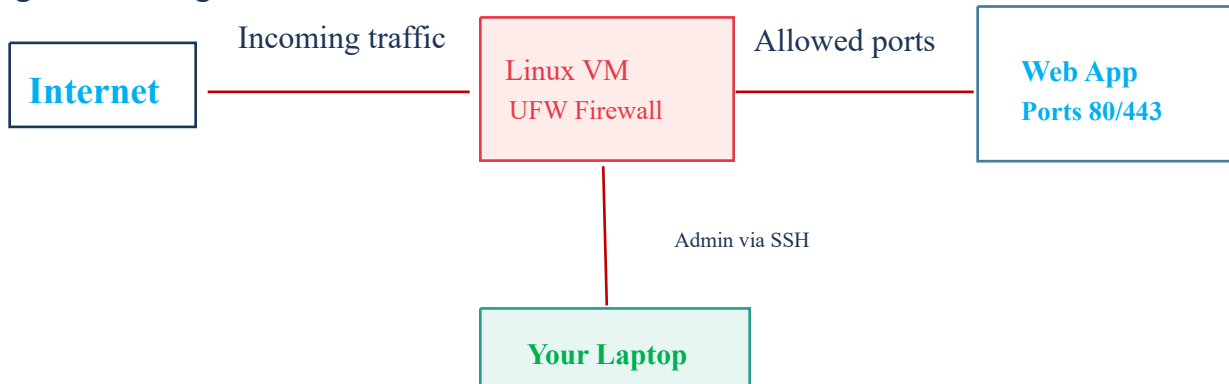
A simple, project with an optional AI auto-blocker

Sec	Section	Page
1.	What You'll Build & Architecture	2
2.	Prerequisites & Setup	3
3.	Install & Enable UFW	4
4.	Allow/Block Rules (SSH/HTTP/HTTPS)	6
5.	Default Policies & App Profiles	8
6.	Logging & Status Checks	10
7.	AI Auto-Blocker (Python)	12
8.	Testing & Validation	15
9.	Troubleshooting	17
10.	Cheatsheet	19

1) What You'll Build & Architecture

A secure-by-default linux server with UFW firewall. Only allow SSH (for admin) and web ports (80/443), deny everything else. Optional Python script watches auth logs and auto-blocks abusive IPs.

High-level Diagram



2) Prerequisites & Setup

- Ubuntu 22.04/24.04 VM (VirtualBox/VMware/WSL) or a small cloud VM (AWS Lightsail, GCP e2-micro).
- A user with sudo privileges.
- Internet connection.

Create Your Admin User

If using a fresh VM, create a new user and add to sudoers:

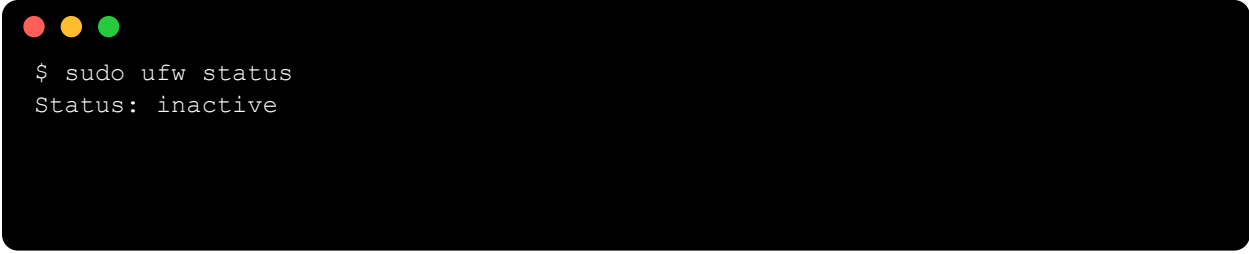
```
$ sudo adduser peter
$ sudo usermod -aG sudo peter
$ su - peter
```

Update the System

```
$ sudo apt update && sudo apt upgrade -y
```


3) Install & Enable UFW

UFW is usually preinstalled on Ubuntu. Check first:



```
$ sudo ufw status  
Status: inactive
```

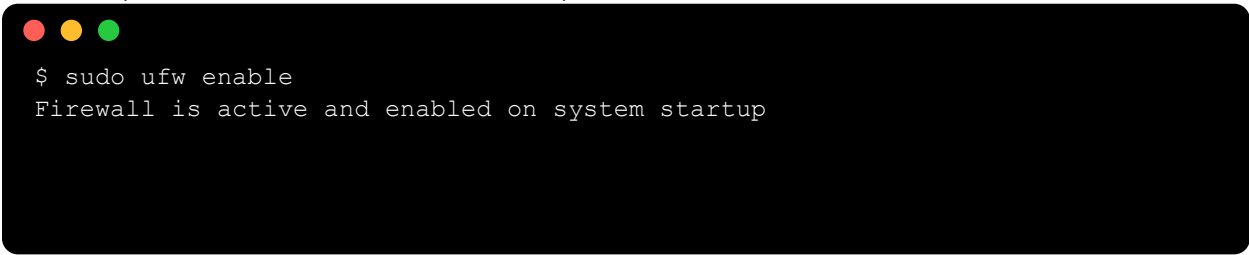
If not installed:



```
$ sudo apt install ufw -y
```

Enable UFW

Turn it on (it will remember rules across reboots):



```
$ sudo ufw enable  
Firewall is active and enabled on system startup
```

4) Allow/Block Rules (SSH/HTTP/HTTPS)

Always allow SSH first if you're using a remote/cloud VM, or you could lock yourself out.

```
● ● ●  
$ sudo ufw allow ssh  
Rule added
```

Allow common web ports if you plan to host a site:

```
● ● ●  
$ sudo ufw allow 80/tcp  
$ sudo ufw allow 443/tcp
```

You can allow by service name if available:

```
● ● ●  
$ sudo ufw app list  
Available applications:  
  OpenSSH  
  Apache  
  Nginx Full  
  Nginx HTTP  
  Nginx HTTPS
```

Examples:

```
● ● ●  
$ sudo ufw allow "Nginx Full" # opens 80 & 443  
$ sudo ufw delete allow 80/tcp # remove a rule
```

5) Default Policies & App Profiles

Set secure defaults: deny all incoming, allow outgoing:

```

$ sudo ufw default deny incoming
Default incoming policy changed to 'deny'
(Your rules will be reloaded)
$ sudo ufw default allow outgoing
Default outgoing policy changed to 'allow'
```

Verify your rules:

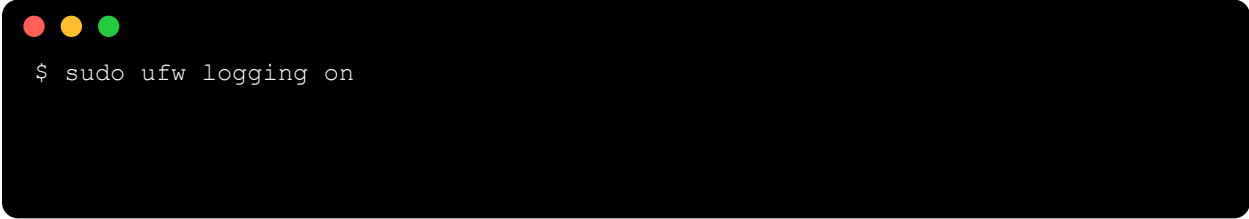
```

$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip
```

To	Action	From
--	-----	----
22/tcp (OpenSSH)	ALLOW	Anywhere
80/tcp	ALLOW	Anywhere
443/tcp	ALLOW	Anywhere

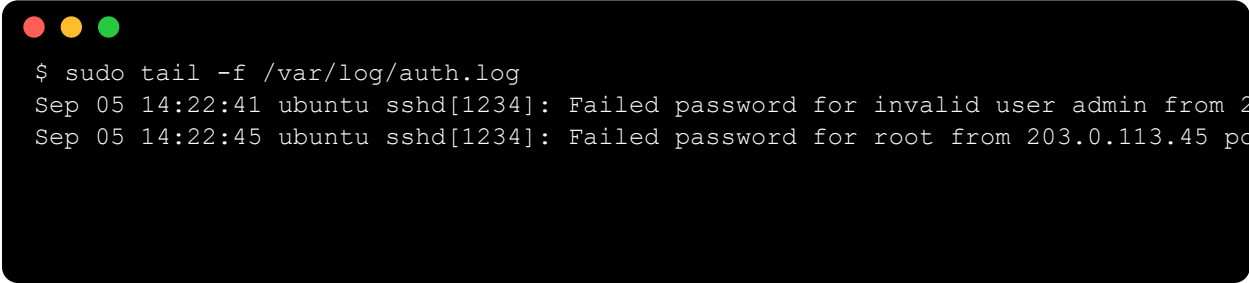
6) Logging & Status Checks

UFW integrates with system logs. Turn logging on (if not already):




```
$ sudo ufw logging on
```

Watch authentication attempts live:



```
$ sudo tail -f /var/log/auth.log
Sep 05 14:22:41 ubuntu sshd[1234]: Failed password for invalid user admin from 2
Sep 05 14:22:45 ubuntu sshd[1234]: Failed password for root from 203.0.113.45 po
```

List numbered rules (easy to delete by number):



```
$ sudo ufw status numbered
[ 1] 22/tcp ALLOW Anywhere
[ 2] 80/tcp ALLOW Anywhere
[ 3] 443/tcp ALLOW Anywhere
```

7) Optional: AI-style Auto-Blocker (Python)

This simple script monitors auth.log and auto-blocks IPs showing suspicious behavior. It's a rule-based approach (threshold) — later you can swap the logic with ML.

Create the script

```
$ sudo nano /usr/local/bin/auto_blocker.py
```

Make it executable and run

```
$ sudo chmod +x /usr/local/bin/auto_blocker.py
$ sudo /usr/local/bin/auto_blocker.py
[*] Auto Blocker watching /var/log/auth.log
```

To run at boot, create a systemd service:

```
$ sudo nano /etc/systemd/system/auto-blocker.service
```

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable --now auto-blocker.service
$ systemctl status auto-blocker.service
```

```
#!/usr/bin/env python3
import subprocess, re, time, os


def monitor():
    print("[*] Watching SSH logs on Kali...")
    while True:
        logs = subprocess.getoutput("journalctl -u ssh --since '1 minute ago'")
        for line in logs.split("\n"):
            if "Failed password" in line:
                ip_match = re.search(r"from (\d+\.\d+\.\d+\.\d+)", line)
                if ip_match:
                    ip = ip_match.group(1)
                    print(f"[!] Blocking IP: {ip}")
                    os.system(f"ufw deny from {ip}")
        time.sleep(60)

if __name__ == "__main__":
    monitor()
```

Here is an example code the main I used is different


8) Testing & Validation

From another machine (or using a container), attempt a few failed SSH logins, then confirm the IP gets blocked.



```
# On an attacker box (example IP 203.0.113.45)
$ for i in $(seq 1 6); do ssh root@YOUR_SERVER -p 22; done
# Back on the server:
$ sudo ufw status | grep 203.0.113.45
Anywhere                                DENY                                203.0.113.45
```

Troubleshoot by checking logs:



```
$ journalctl -u auto-blocker.service -e
$ sudo tail -f /var/log/auth.log
```

9) Troubleshooting

- Locked yourself out? Use the VM console to login and run **sudo ufw disable**.
- No SSH connectivity? Ensure port 22 is allowed and the VM network is bridged/NAT correctly.
- Service not starting? Check permissions and shebang (`#!/usr/bin/env python3`).

Reset UFW (last resort)



```
$ sudo ufw reset  
$ sudo ufw enable
```

10) Cheatsheet

Task	Command
Status	<code>sudo ufw status verbose</code>
Enable / Disable	<code>sudo ufw enable</code> <code>sudo ufw disable</code>
Allow SSH	<code>sudo ufw allow ssh</code>
Allow web	<code>sudo ufw allow 80/tcp</code> && <code>sudo ufw allow 443/tcp</code>
Deny an IP	<code>sudo ufw deny from 203.0.113.45</code>
Delete rule #3	<code>sudo ufw delete 3</code>
Turn on logging	<code>sudo ufw logging on</code>

