[How To] Basic set-up of a Virtual Private Server (VPS) for Debian based Systems

This tutorial will show you how to do the initial set-up and configuration of a Virtual Private Server (VPS) based on Debian (the commands bellow are specific to Debian Stretch (9) but can easily be adapted to other Debian based systems).

By the end of this tutorial you will have a personalized and secure VPS with the basic systems configured and ready to deploy any advanced configuration.

Part I - Basic Configuration for a newly installed server (configure password, date, time, language, updates, etc..)

1. Generate an ssh key for the user and install it on the server (only needs to be done first time - skip if you already have an ssh key);

In your home PC, issue the command:

```
ssh-keygen
```

Copy the key to the server:

```
ssh-copy-id [your user]@[server ip address] -p [ssh port]
```

(provide your user/root password when asked and the key will be automatically added to the server)

The ssh key will be stored in the hidden folder .ssh in the user home – for example, /home/user/.ssh - backup the ssh key to a safe place so that you can restore later or for use it in another computer.

3. Change root password:

```
sudo su
```

Insert user password to log in as root

```
passwd
```

4. Force APT to use ipv4

```
echo 'Acquire::ForceIPv4 "true";' | tee /etc/apt/apt.conf.d/99force-ipv4
```

5. Make sure the sources list is correct and includes the backports repos (change from "stretch" to "buster" for latest Debian 10):

```
nano /etc/apt/sources.list
```

> “
> # Debian Stretch main, contrib and non-free repositores
>
> deb http://deb.debian.org/debian stretch main contrib non-free
> #deb-src http://deb.debian.org/debian stretch main contrib non-free
>
> deb http://deb.debian.org/debian-security/ stretch/updates main contrib non-free
> #deb-src http://deb.debian.org/debian-security/ stretch/updates main contrib non-free
>
> deb http://deb.debian.org/debian stretch-updates main contrib non-free
> #deb-src http://deb.debian.org/debian stretch-updates main contrib non-free
>
> # for the debian backports repository

> deb http://ftp.debian.org/debian stretch-backports main contrib non-free
> #deb-src http://ftp.debian.org/debian stretch-backports main contrib non-free

6. Update the repositories:

```
apt update
```

7. Install CA Certificates and apt-transport-https

```
apt install -y ca-certificates apt-transport-https
```

8. Install some basic initial packages (install only what you need):

```
apt -y install sudo vim htop git cron curl pwgen zip aptitude lsof python net-tools dnsutils dnsmasq
```

9. Enable easy copy/paste on vim in Debian 9:

```
vi /usr/share/vim/vim80/defaults.vim
```

change the "set mouse" field as per the bellow:

> In many terminal emulators the mouse works just fine. By enabling it you can position the cursor, Visually select and scroll with the mouse.
> if has('mouse')
> **set mouse=r**
> endif

10. Set the server hostname:

```
vi /etc/hostname
```

11. Set up the Hosts:

```
vi /etc/hosts
```

Make sure it has the right configuration and set the ip and fqdn

> “
> 127.0.0.1 localhost
> 127.0.1.1 name.domain.TLD name
>
> 139.134.64.184 subdomain.yourdomain.TLD
>
> # The following lines are desirable for IPv6 capable hosts
> ::1 localhost ip6-localhost ip6-loopback
> ff02::1 ip6-allnodes
> ff02::2 ip6-allrouters
>
> 3b09:5ef0:eeff:5d13::1000 subdomain.yourdomain.TLD

12. Find the default gateway and netmask:

```
ip route | grep default
```

```
ip a
```

```
ifconfig
```

13. Configure the Network Interfaces with static IP address:

```
vi /etc/network/interfaces
```

The example bellow is for KVM using ens3 - change as necessary for OVZ with venet0:

> “
> # This file describes the network interfaces available on your system
> # and how to activate them. For more information, see interfaces(5).
>
> source /etc/network/interfaces.d/*
>
> # The loopback network interface
> auto lo
> iface lo inet loopback
>
> # The primary network interface
> **auto ens3**
> **iface ens3 inet static**
> **address 139.134.64.184**
> **netmask 255.255.255.0**
> **gateway 139.134.64.1**
>
> **up ip addr add 139.134.64.184/24 dev ens3**
> **down ip addr del 139.134.64.184/24 dev ens3**
>
> **up ip addr add 139.134.64.185/24 dev ens3**
> **down ip addr del 139.134.64.185/24 dev ens3**
>
>
> **iface ens3 inet6 static**
> **address 3b09:5ef0:eeff:5d13::1000**
> **netmask 64**
> **#gateway 3b09:5ef0:eeff::1**
>
> **post-up /sbin/ip -r route add 3b09:5ef0:eeff::1 dev ens3**
> **post-up /sbin/ip -r route add default via 3b09:5ef0:eeff::1**
>
> **up ip addr add 3b09:5ef0:eeff:5d13::2000/64 dev ens3**
> **down ip addr del 23b09:5ef0:eeff:5d13::2000/64 dev ens3**

14. Configure system resolvers

`vi /etc/resolv.conf`

> nameserver 8.8.8.8
> nameserver 8.8.4.4
> nameserver 2001:4860:4860:0:0:0:0:8888
> nameserver 2001:4860:4860:0:0:0:0:8844
> nameserver 208.67.222.222
> nameserver 208.67.220.220
> nameserver 2620:0:ccc::2
> nameserver 2620:0:ccd::2

15. Update locale and date/time:

`dpkg-reconfigure locales`

`dpkg-reconfigure tzdata`

If locale generation is unsuccesful it can be done manually:

> locale-gen en_GB en_GB.UTF-8
> export LANGUAGE=en_GB.UTF-8
> export LANG=en_GB.UTF-8
> export LC_ALL=C
> locale-gen en_GB.UTF-8
> update-locale LANG=en_GB.UTF-8

> # another method is to edit or create the Language file:
>
> vi /etc/locale.gen
>
> # make sure it has the language, like en_GB.UTF-8 UTF-8

> /usr/sbin/locale-gen

16. Configure SSH access to the server

```
vi /etc/ssh/sshd_config
```

Change port, disable root login, disable password authentication and add at the end:

> Port **29781** # the ssh port to be used (default 22)
> AddressFamily **inet** # to listen only on IPv4 or
> AddressFamily **inet6** # to listen only on IPv6
> (...)
> PermitRootLogin **prohibit-password** # do not allow root login with password
> (...)
> PasswordAuthentication **no**
> (...)
> UseDNS **no**
> AllowUsers **[yourusername]** # Will only allow login from specified user - this will disable root login

Restart the ssh service:

```
systemctl restart sshd
```

confirm the changes are in effect:

```
ss -tnlp | grep ssh
```

17. Change the ssh port in the services file (only if you have changed the ssh port in the previous step):

```
vi /etc/services
```

18. Disable ipv6 if necessary by editing the sysctl conf file:

```
vi /etc/sysctl.conf
```

Add the following to the end of the file:

> # IPv6 disabled
> net.ipv6.conf.all.disable_ipv6 = 1

Save and activate the changes;

```
sysctl -p /etc/sysctl.conf
```

Check ifconfig to confirm they are disabled

```
ifconfig
```

19. Upgrade the system with the latest packages:

```
apt -y full-upgrade
```

20. Install additional software that you may require:

```
apt -y install software-properties-common
```

21. the system can be set up to perform automatic updates:

```
apt -y install unattended-upgrades
```

the upgrades can be configured by editing the file:

> vi /etc/apt/apt.conf.d/50unattended-upgrades

22. Install and Configure ntpd (to keep the server's time synchronized)

```
apt -y install ntp ntpdate
```

Configure the ntp servers if necessary:

```
vi /etc/ntp.conf
```

Use the following servers for europe:

> pool 0.europe.pool.ntp.org iburst
> pool 1.europe.pool.ntp.org iburst
> pool 2.europe.pool.ntp.org iburst
> pool 3.europe.pool.ntp.org iburst

Start the service

```
systemctl start ntp
```

Enable the service to start automatically on boot

```
systemctl enable ntp
```

23. Create a user account with sudo privileges:

```
adduser [yourusername]
```

```
adduser [yourusername] sudo
```

```
visudo
```

edit the new user to have root privileges:

> [yourusername] ALL=(ALL) ALL

Part II - Advanced Configuration (configure SSL certificates and email notifications)

24. Install a SSL Certificate with Let’s Encrypt (Certbot) client

```
apt install -y certbot
```

Obtain a SSL certificate for your domain name with this command (Must have the DNS records correctly configured before)

```
certbot certonly --standalone --email user@yourdomain.com --agree-tos -d yourdomain.com (-d www.yourdomain.com -d web.yourdomain.com ...)
```

Your SSL certificate will be saved under:

> /etc/letsencrypt/live/yourdomain.com/fullchain.pem   # for the certificate
> /etc/letsencrypt/live/yourdomain.com/privkey.pem   # for the key

create a deploy-hook script for renewals:

```
vi /usr/local/bin/certbot-deploy-hook
```

set in it the services that require restart after renewing certificates:

> #!/bin/sh

> systemctl restart postfix 2>/dev/null
> systemctl restart dovecot 2>/dev/null

make the script executable

```
chmod u+x /usr/local/bin/certbot-deploy-hook
```

copy the service script to systemd

```
cp /lib/systemd/system/certbot.service /etc/systemd/system/
```

edit the script to include the post-hook

```
vi /etc/systemd/system/certbot.service
```

edit the ExecStart line as per the bellow (assuming apache webserver will bee running in the server - adapt as necessary):

> ExecStart=/usr/bin/certbot renew --pre-hook "systemctl stop apache2" --post-hook "systemctl start apache2" --deploy-hook /usr/local/bin/certbot-deploy-hook >> /var/log/certbot-renew.log

reload systemd

```
systemctl daemon-reload
```

you can test the script with a dry run:

```
certbot --dry-run renew --pre-hook "systemctl stop apache2" --post-hook "systemctl start apache2" --deploy-hook /usr/local/bin/certbot-deploy-hook >> /var/log/certbot-renew.log
```

25. Install Postfix to receive email notifications from the server:

```
apt-get -y install mailutils postfix
```

When asked to configure postfix, choose internet site and the server dns record, e.g. server.yourdomain.com

Start by configuring the postfix main.conf:

```
vi /etc/postfix/main.cf
```

and change the folllowing default values:

> (...)
> # TLS parameters
> smtpd_tls_cert_file=**/etc/letsencrypt/live/yourdomain.com/fullchain.pem**
> smtpd_tls_key_file=**/etc/letsencrypt/live/yourdomain.com/privkey.pem**
> **smtpd_tls_security_level = may**
> **smtpd_tls_auth_only = yes**
> **smtp_tls_security_level = may**
> mydestination = **$myhostname, localhost.$mydomain, $mydomain**
> inet_interfaces = **loopback-only**

Save, close and restart postfix:

```
systemctl restart postfix
```

Test by sending an email:

```
echo "This is a test." | mail -s "Testing Postfix" youremail@yourdomain.com
```

26. Set up email forward for system notifications:

```
vi /etc/aliases
```

The default will forward system mail to the root account; to reroute system mail to your own mail address change the file to associate your email to root account (the email can NOT be in the same domain configured in hosts or it be delivered locally):

> # /etc/aliases
> mailer-daemon: postmaster
> postmaster: root
> nobody: root
> hostmaster: root

> usenet: root
> news: root
> webmaster: root
> www: root
> ftp: root
> abuse: root
> noc: root
> security: root
> root: youremail@yourdomain.com

Save and activate the new aliases with the command:

```
newaliases
```

you can check the configuration by sending an email to root and checking your email address:

```
echo "This is the body of the email" |  mail -s "This is the subject line" root
```

you can also configure additional virtual addresses if necessary:

```
vi /etc/postfix/virtual
```

specify the emails that you wish to create on the left-hand side, and username to deliver the mail to on the right-hand side, like this:

> user2@yourdomain.com user2

you can also set up certain addresses to forward to multiple accounts by using a comma-separated list:

> user2@yourdomain.com user2,root

Part III - Security (configure Firewall and Fail2Ban)

27. Configure iptable rules to firewall access to the server:

Start by checking current firewall rules, if any:

`iptables -L` # for ipv4 rules

`ip6tables -L` # for ipv6 rules

Create a new file to configure firewall rules (we'll use one file for both IPv4 and IPv6):

`vi /etc/iptables.firewall.rules`

Use the example bellow to configure iptable rules (change the parts in bold to your specific requirements):

> *filter
>
> # Base policy
> :INPUT DROP [0:0]
> :FORWARD DROP [0:0]
> :OUTPUT ACCEPT [0:0]
>
> # Don't attempt to firewall internal traffic on the loopback device.
> -A INPUT -i lo -j ACCEPT
>
> # Continue connections that are already established or related to an established
> # connection.
> -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

```
# Drop non-conforming packets, such as malformed headers, etc.
-A INPUT -m conntrack --ctstate INVALID -j DROP

# Block remote packets claiming to be from a loopback address.
-4 -A INPUT -s 127.0.0.0/8 ! -i lo -j DROP
-6 -A INPUT -s ::1/128 ! -i lo -j DROP

# Drop all packets that are going to broadcast, multicast or anycast address.
-4 -A INPUT -m addrtype --dst-type BROADCAST -j DROP
-4 -A INPUT -m addrtype --dst-type MULTICAST -j DROP
-4 -A INPUT -m addrtype --dst-type ANYCAST -j DROP
-4 -A INPUT -d 224.0.0.0/4 -j DROP

# Chain for preventing SSH brute-force attacks.
# Permits 10 new connections within 5 minutes from a single host then drops
# incomming connections from that host. Beyond a burst of 100 connections we
# log at up 1 attempt per second to prevent filling of logs.
-N SSHBRUTE
-A SSHBRUTE -m recent --name SSH --set
-A SSHBRUTE -m recent --name SSH --update --seconds 300 --hitcount 10 -m
limit --limit 1/second --limit-burst 100 -j LOG --log-prefix "iptables[SSH-brute]: "
-A SSHBRUTE -m recent --name SSH --update --seconds 300 --hitcount 10 -j
DROP
-A SSHBRUTE -j ACCEPT

# Chain for preventing ping flooding - up to 6 pings per second from a single
# source, again with log limiting. Also prevents us from ICMP REPLY flooding
# some victim when replying to ICMP ECHO from a spoofed source.
-N ICMPFLOOD
-A ICMPFLOOD -m recent --set --name ICMP --rsource
-A ICMPFLOOD -m recent --update --seconds 1 --hitcount 6 --name ICMP --
rsource --rttl -m limit --limit 1/sec --limit-burst 1 -j LOG --log-prefix
"iptables[ICMP-flood]: "
-A ICMPFLOOD -m recent --update --seconds 1 --hitcount 6 --name ICMP --
rsource --rttl -j DROP
-A ICMPFLOOD -j ACCEPT


##################################################
# 2. HOST SPECIFIC RULES #
# #
# This section is a good place to enable your host-specific services. #
##################################################

# Allow All Traffic from your Private Network
-4 -A INPUT -s 139.134.64.185/32 -j ACCEPT
```

```
-4 -A INPUT -s 182.74.31.112/32 -j ACCEPT
-6 -A INPUT -s 3b09:5ef0:eeff:5d12::1000 -j ACCEPT
-6 -A INPUT -s 1002:0bf0:0044:0007:0000:0000:5d9e:c34f -j ACCEPT

# Accept HTTP and HTTPS
-A INPUT -p tcp -m multiport --dports 80,443 --syn -m conntrack --
ctstate NEW -j ACCEPT

# Accept FTP only for IPv4
#-4 -A INPUT -p tcp --dport 21 --syn -m conntrack --ctstate NEW -j
ACCEPT

# Allows NTP access
#-A INPUT -p udp --dport 123 -j ACCEPT

# Allows SMTP access
#-A INPUT -p tcp --dport 25 -j ACCEPT
#-A INPUT -p tcp --dport 465 -j ACCEPT
#-A INPUT -p tcp --dport 587 -j ACCEPT

# Allows pop and pops connections
#-A INPUT -p tcp --dport 110 -j ACCEPT
#-A INPUT -p tcp --dport 995 -j ACCEPT

# Allows imap and imaps connections
#-A INPUT -p tcp --dport 143 -j ACCEPT
#-A INPUT -p tcp --dport 993 -j ACCEPT

##################################################
# 3. GENERAL RULES #
# #
# This section contains general rules that should be suitable for most hosts. #
##################################################

# Accept worldwide access to SSH and use SSHBRUTE chain for
preventing
# brute-force attacks.
-A INPUT -p tcp --dport 29781 --syn -m conntrack --ctstate NEW -j
SSHBRUTE

# Permit useful IMCP packet types for IPv4
# Note: RFC 792 states that all hosts MUST respond to ICMP ECHO requests.
# Blocking these can make diagnosing of even simple faults much more tricky.
# Real security lies in locking down and hardening all services, not by hiding.
-4 -A INPUT -p icmp --icmp-type 0 -m conntrack --ctstate NEW -j ACCEPT
```

```
-4 -A INPUT -p icmp --icmp-type 3 -m conntrack --ctstate NEW -j ACCEPT
-4 -A INPUT -p icmp --icmp-type 11 -m conntrack --ctstate NEW -j ACCEPT

# Permit needed ICMP packet types for IPv6 per RFC 4890.
-6 -A INPUT -p ipv6-icmp --icmpv6-type 1 -j ACCEPT
-6 -A INPUT -p ipv6-icmp --icmpv6-type 2 -j ACCEPT
-6 -A INPUT -p ipv6-icmp --icmpv6-type 3 -j ACCEPT
-6 -A INPUT -p ipv6-icmp --icmpv6-type 4 -j ACCEPT
-6 -A INPUT -p ipv6-icmp --icmpv6-type 133 -j ACCEPT
-6 -A INPUT -p ipv6-icmp --icmpv6-type 134 -j ACCEPT
-6 -A INPUT -p ipv6-icmp --icmpv6-type 135 -j ACCEPT
-6 -A INPUT -p ipv6-icmp --icmpv6-type 136 -j ACCEPT
-6 -A INPUT -p ipv6-icmp --icmpv6-type 137 -j ACCEPT
-6 -A INPUT -p ipv6-icmp --icmpv6-type 141 -j ACCEPT
-6 -A INPUT -p ipv6-icmp --icmpv6-type 142 -j ACCEPT
-6 -A INPUT -s fe80::/10 -p ipv6-icmp --icmpv6-type 130 -j ACCEPT
-6 -A INPUT -s fe80::/10 -p ipv6-icmp --icmpv6-type 131 -j ACCEPT
-6 -A INPUT -s fe80::/10 -p ipv6-icmp --icmpv6-type 132 -j ACCEPT
-6 -A INPUT -s fe80::/10 -p ipv6-icmp --icmpv6-type 143 -j ACCEPT
-6 -A INPUT -p ipv6-icmp --icmpv6-type 148 -j ACCEPT
-6 -A INPUT -p ipv6-icmp --icmpv6-type 149 -j ACCEPT
-6 -A INPUT -s fe80::/10 -p ipv6-icmp --icmpv6-type 151 -j ACCEPT
-6 -A INPUT -s fe80::/10 -p ipv6-icmp --icmpv6-type 152 -j ACCEPT
-6 -A INPUT -s fe80::/10 -p ipv6-icmp --icmpv6-type 153 -j ACCEPT

# Permit IMCP echo requests (ping) and use ICMPFLOOD chain for preventing
ping
# flooding.
-4 -A INPUT -p icmp --icmp-type 8 -m conntrack --ctstate NEW -j ICMPFLOOD
-6 -A INPUT -p ipv6-icmp --icmpv6-type 128 -j ICMPFLOOD

# Do not log packets that are going to ports used by SMB
# (Samba / Windows Sharing).
-A INPUT -p udp -m multiport --dports 135,445 -j DROP
-A INPUT -p udp --dport 137:139 -j DROP
-A INPUT -p udp --sport 137 --dport 1024:65535 -j DROP
-A INPUT -p tcp -m multiport --dports 135,139,445 -j DROP

# Do not log packets that are going to port used by UPnP protocol.
-A INPUT -p udp --dport 1900 -j DROP

# Do not log late replies from nameservers.
-A INPUT -p udp --sport 53 -j DROP

# Good practise is to explicately reject AUTH traffic so that it fails fast.
```

> -A INPUT -p tcp --dport 113 --syn -m conntrack --ctstate NEW -j REJECT --reject-with tcp-reset
>
> # Prevent DOS by filling log files.
> -A INPUT -m limit --limit 1/second --limit-burst 100 -j LOG --log-prefix "iptables[DOS]: "
>
> # Drop Everything else
> -P INPUT DROP
> -P FORWARD DROP
>
> COMMIT

Load the firewall rules:

```
iptables-restore < /etc/iptables.firewall.rules
```

```
ip6tables-restore < /etc/iptables.firewall.rules
```

Confirm that the firewall rules have loaded:

```
iptables -L
```

```
ip6tables -L
```

> “ # ONLY If running VPN service - you will need masquerade -
>
> for KVM with ens3:
>
> ```
> iptables -t nat -4 -A POSTROUTING -s 10.8.0.0/24 -o ens3 -j MASQUERADE
> ```
>
> ```
> ip6tables -t nat -6 -A POSTROUTING -s ::1/128 -o ens3 -j MASQUERADE
> ```
>
> for venet0 with SNAT
>
> ```
> iptables -t nat -A POSTROUTING -o venet0 -j SNAT --to-source 139.134.64.184
> ```
>
> # or

```
iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -j SNAT --to-source
139.134.64.184
```

Enable the iptables rules across reboots by installing iptables-persistent:

```
apt-get -y install iptables-persistent
```

> # To reconfigure iptables persistent after making changes to the rules:
>
> ```
> dpkg-reconfigure iptables-persistent
> ```

Confirm that all iptable rules are active:

```
iptables -vL
```

```
ip6tables -vL
```

27. Configure fail2ban to prevent brute-force attacks:

In Debian Stretch install fail2ban from stretch-backports for the latest version with ipv6 support:

```
apt-get install -t stretch-backports fail2ban -y
```

Copy the default local config files to edit:

```
cp /etc/fail2ban/fail2ban.conf /etc/fail2ban/fail2ban.local
```

```
cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

open the jail.local and edit as required:

```
vi /etc/fail2ban/jail.local
```

the most relevant fields to edit are:

> # MISCELLANEOUS OPTIONS
> #
>
> # "ignoreip" can be a list of IP addresses, CIDR masks or DNS hosts. Fail2ban
> # will not ban a host which matches an address in this list. Several addresses
> # can be defined using space (and/or comma) separator.
> ignoreip = 127.0.0.1/8 ::1 **139.134.64.185/32 182.74.31.112/32 3b09:5ef0:eeff:5d12::1000**
>
> # "bantime" is the number of seconds that a host is banned.
> bantime = **-1** # a negative number means indefintely
>
> # A host is banned if it has generated "maxretry" during the last "findtime"
> # seconds.
> findtime = **10000m** # the m means 'minues'
>
> # "maxretry" is the number of failures before a host get banned.
> maxretry = **5**
>
> (...)
>
> #
> # ACTIONS
> #
>
> # Some options used for actions
>
> # Destination email address used solely for the interpolations in
> # jail.{conf,local,d/*} configuration files.
> destemail = **youremail@yourdomain.com** # the email can NOT be in the same domain configured in hosts or it be delivered locally
>
> # Sender email address used solely for some actions
> sender = **fail2ban@yourdomain.com**
>
> (...)
>
> # Choose default action. To change, just override value of 'action' with the
> # interpolation to the chosen action shortcut (e.g. action_mw, action_mwl, etc) in jail.local
> # globally (section [DEFAULT]) or per specific section
> action = **%(action_mw)s**

> (...)

Enable the jails:

```
vi /etc/fail2ban/jail.d/defaults-debian.conf
```

Default Jails that can be enabled on start:

> [sshd]
> enabled = true
>
> [sshd-ddos]
> enabled = true
>
> [dropbear]
> enabled = true
>
> [pam-generic]
> enabled = true
>
> [xinetd-fail]
> enabled = true

Restart, check status and enable Fail2ban on reboots

```
systemctl restart fail2ban
```

```
systemctl status -l fail2ban
```

```
systemctl enable fail2ban
```

Create a script to easily check all fail2ban jails:

```
vi /opt/fail2ban-allstatus.sh
```

Paste the bellow:

```
#!/bin/bash
JAILS=`fail2ban-client status | grep "Jail list" | sed -E 's/^[^:]+:[ \t]+//' | sed 's/,//g'`
for JAIL in $JAILS
do
fail2ban-client status $JAIL
done
```

execute with the command:

```
bash /opt/fail2ban-allstatus.sh
```

To troubleshoot Fail2ban:

```
journalctl -ru fail2ban
```

or

```
tail -f /var/log/fail2ban.log
```

Reboot

Revision #12
Created Thu, Sep 12, 2019 9:35 AM by Editor
Updated Sat, Sep 14, 2019 3:18 PM by Editor