Cloud-Based Security Operation Center (SOC) Automation via Security Orchestration, Automation, Response (SOAR), With Threat Intelligence Platform (TIP) Capabilities

By
Victor Coil

Table of Contents

Project Objectives

The objective of this project was to gain hands-on experience with the SOAR tool Shuffle for automated active response, as well as to become familiar with how a threat intelligence platform can be used to enrich observables tied to an alert. Overall, the project focuses on detecting SSH brute-force attempts using Wazuh XDR via custom detection rules, then beginning the Shuffle workflow that performs preliminary enrichment of observables with the Virustotal API, case creation on TheHive, further enriching and storing observables in MISP, and executing approval-based responses.

Reflection and Future Improvements

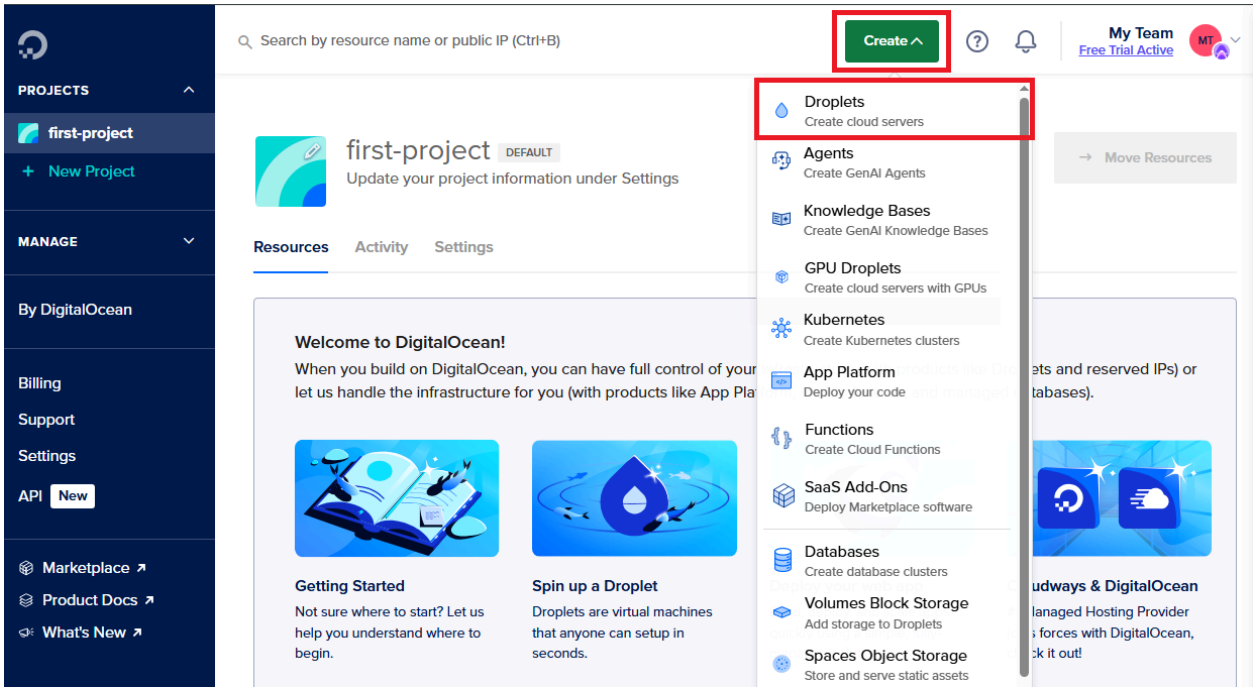
In future iterations, I plan to expand detection beyond bruteforce activity to include broader MITRE ATT&CK techniques, potentially interlink multiple cloud-based SOC environments under a unified MISP instance for threat correlation, and automate rule creation based on the enriched observables.

Section 1 - Wazuh XDR solution initial setup

The first thing on the to-do list for this project is to set up the Wazuh main Server. In this project, I will be using DigitalOcean. They generously offer $200 credits to first-time users. You can use them or any other cloud provider.

If you are using the same provider and have created your account, you should see the dashboard as shown below. Let us set up a droplet, which is the name for a Virtual Machine.

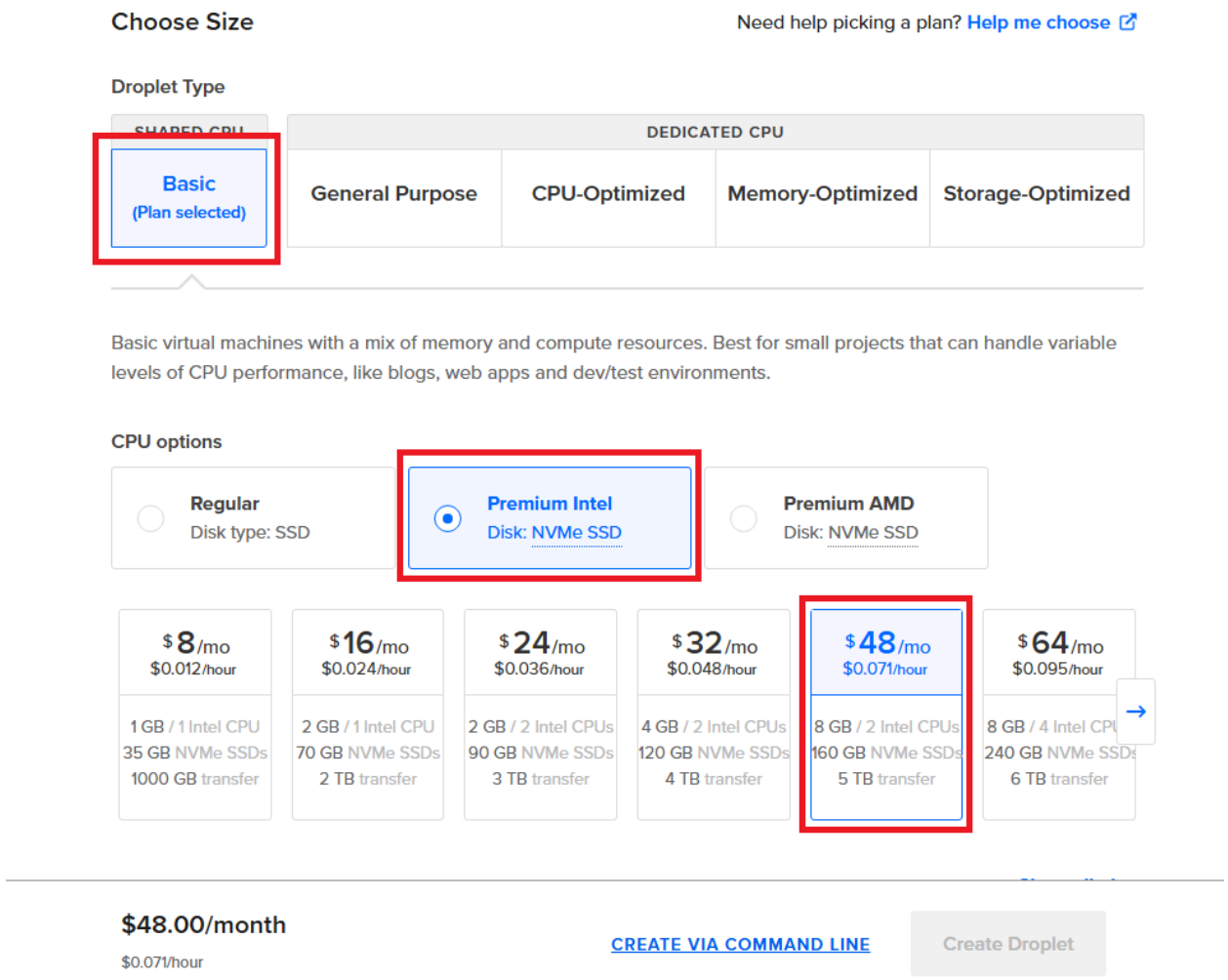
Click on the "**Create**" button, then select "Droplets" from the dropdown menu.



On the "**Create Droplets**" page, select the Region closest to you.

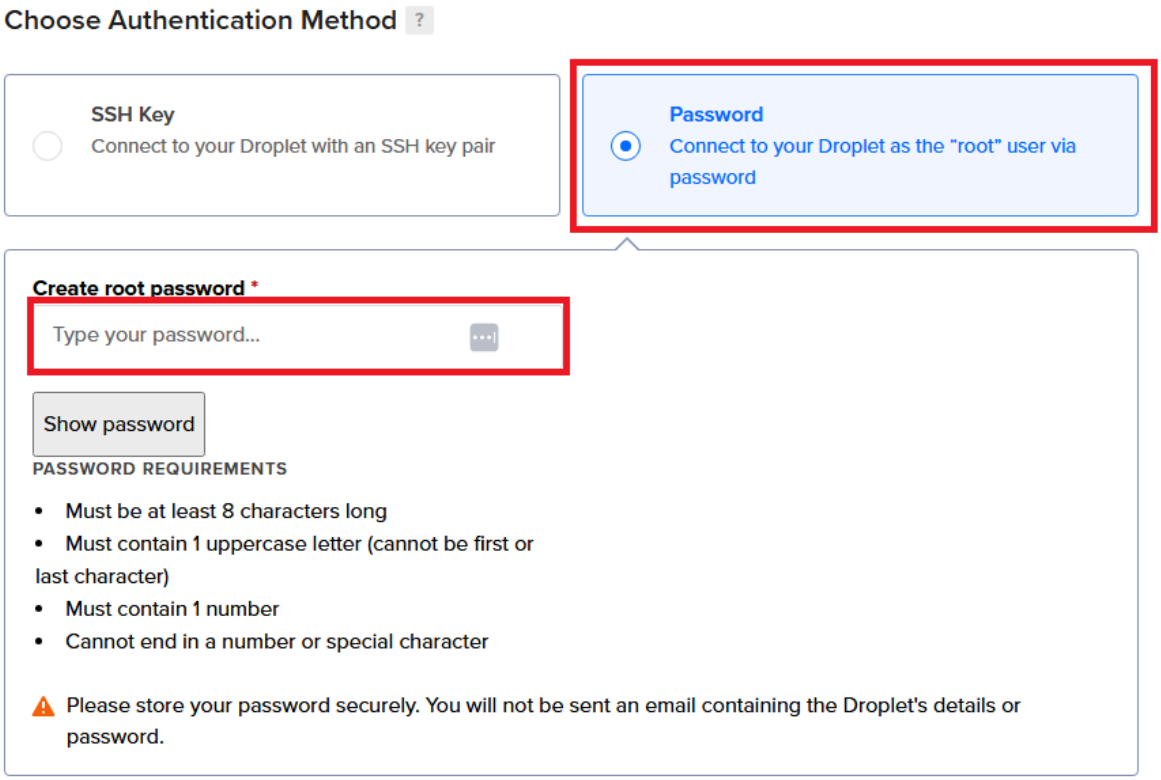
I selected the "**New York**" Region
As for the Image, I chose **Ubuntu**, version "**22.04 (LTS) x64**"as the OS

For the Droplet Type, select “**basic**”. For the CPU options, select “**Premium Intel**”. Then the $48 a month option as shown below.

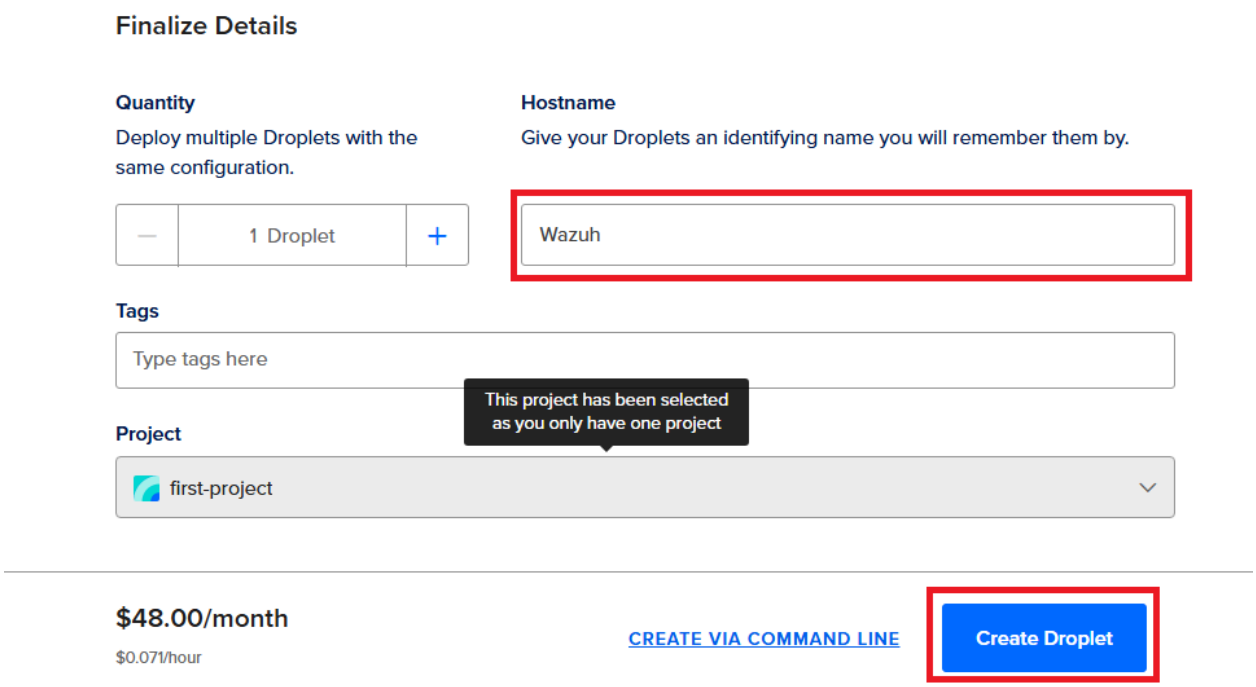


Scroll down a bit to find the “**Choose Authentication Method**” and select the “**Password**” option and input a strong password for the root account.

############################
 # Do not forget this password, as this project may take a few days.
############################

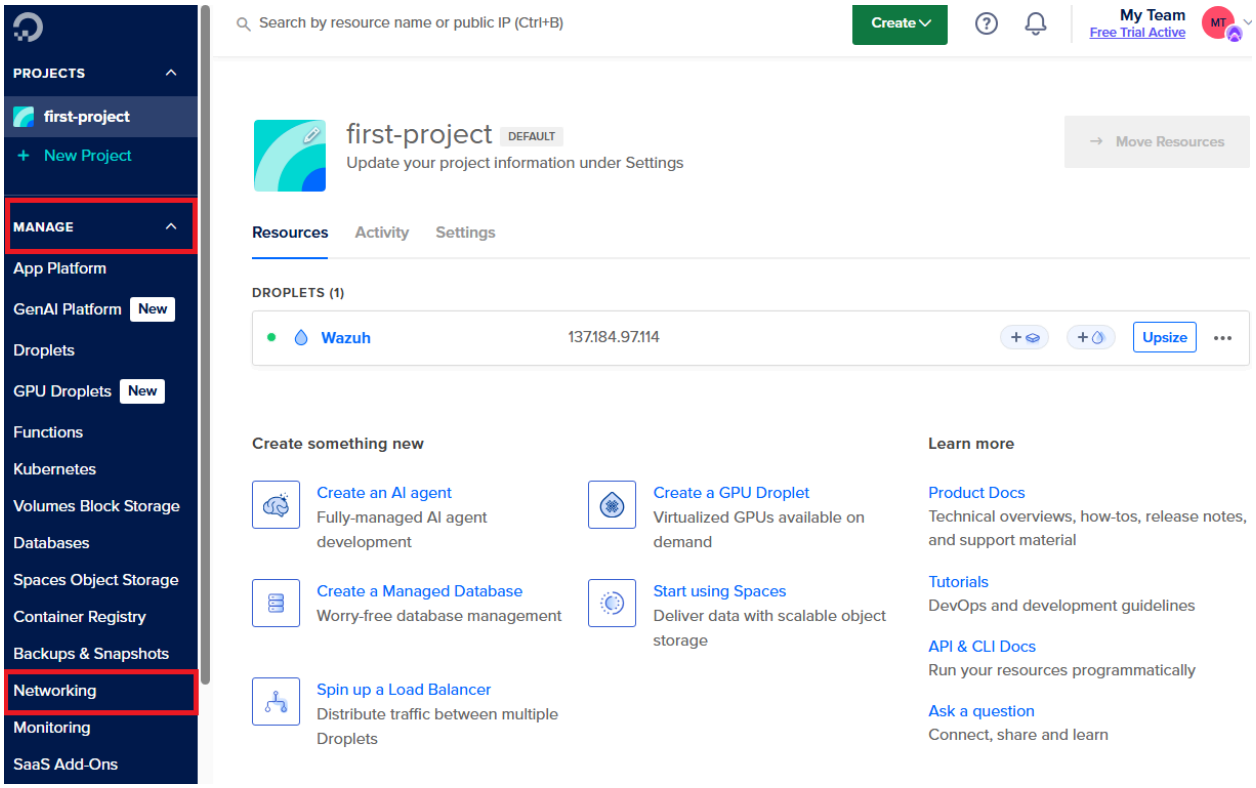


After inputting a strong password, move on down and change the "**Hostname**" to "**Wazuh**". Then click on the "**Create Droplet**" button.

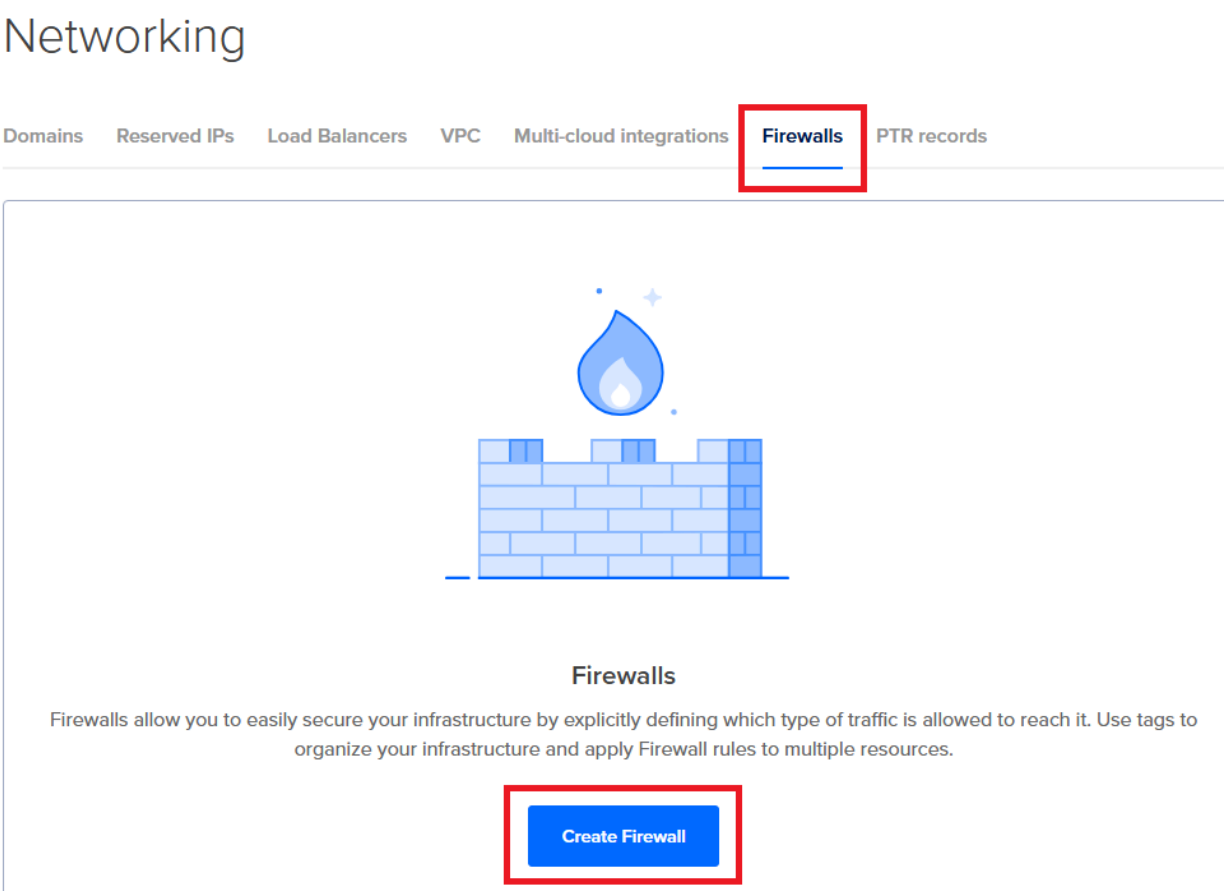


Subsection - Wazuh Firewall

After creating your droplet, it is time to create a firewall to stop any traffic from trying to get into the machine. On the left-hand side, expand "**Manage**", then click on "**Networking**".



Select the “**Firewall**” Tab, and then click on the “**Create Firewall**” button.

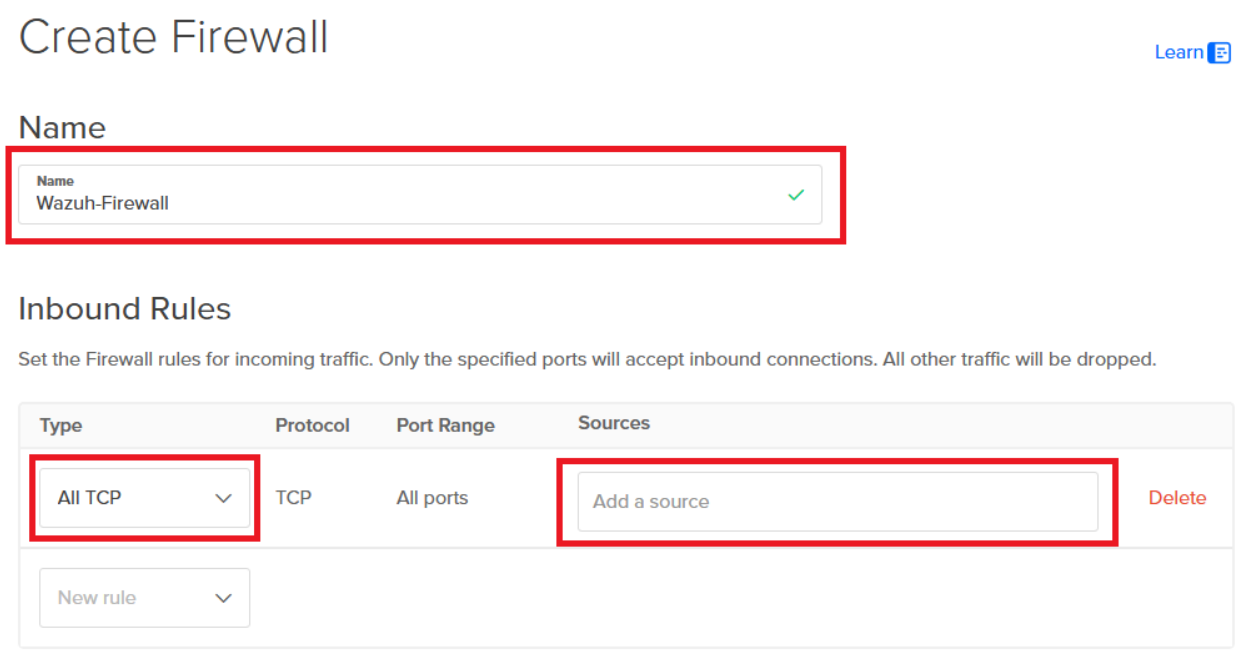


On the “**Create Firewall**” page, input the firewall's name. I selected “**Wazuh-Firewall.**”

Change the type from “**SSH**” to “**All TCP**,” clear the sources, and enter your IP Address.

You can get it from this site:

https://whatismyipaddress.com/

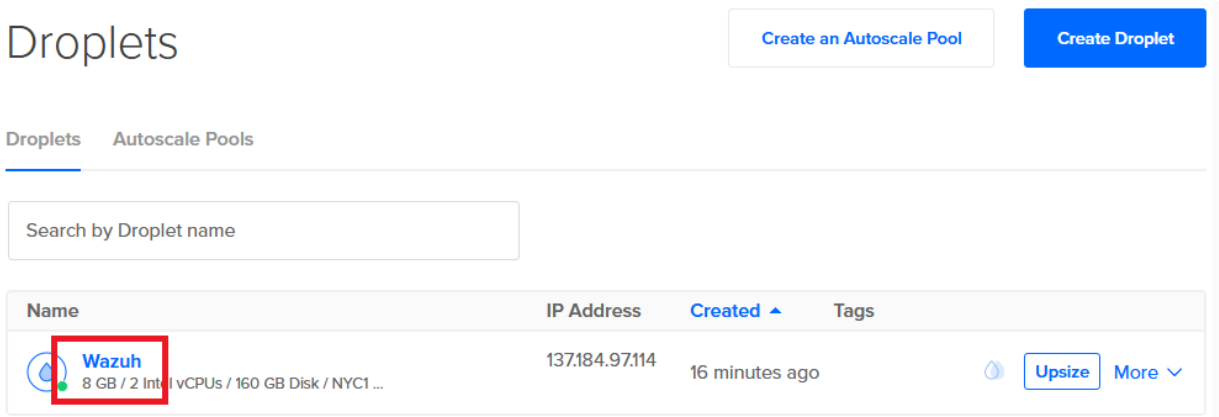


Add a new rule and do the same for “**All UDP**”, with sources being your IP address.

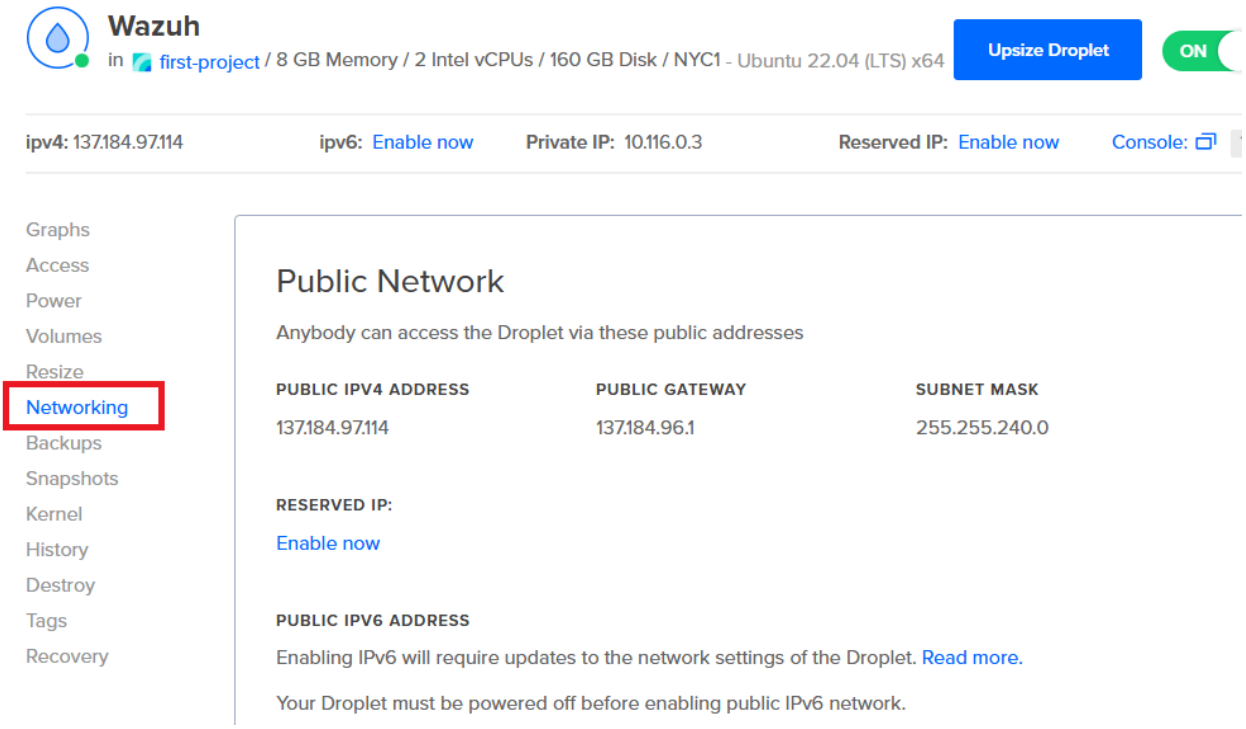
No changes need to be made for the “**Outbound Rules.**”

Scroll down and click on the “**Create Firewall**” button.

Apply the firewall to the Droplet.
Click on “**Droplets**” on the left-hand side. Then click on your droplet's Name.

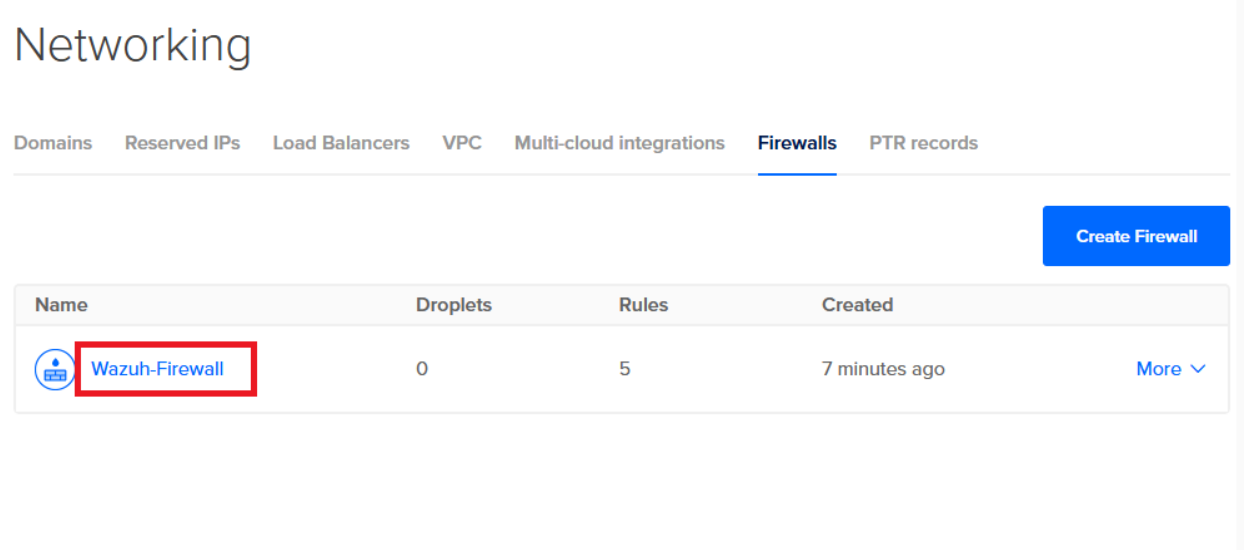


On the left-hand side, click on “**networking**”.

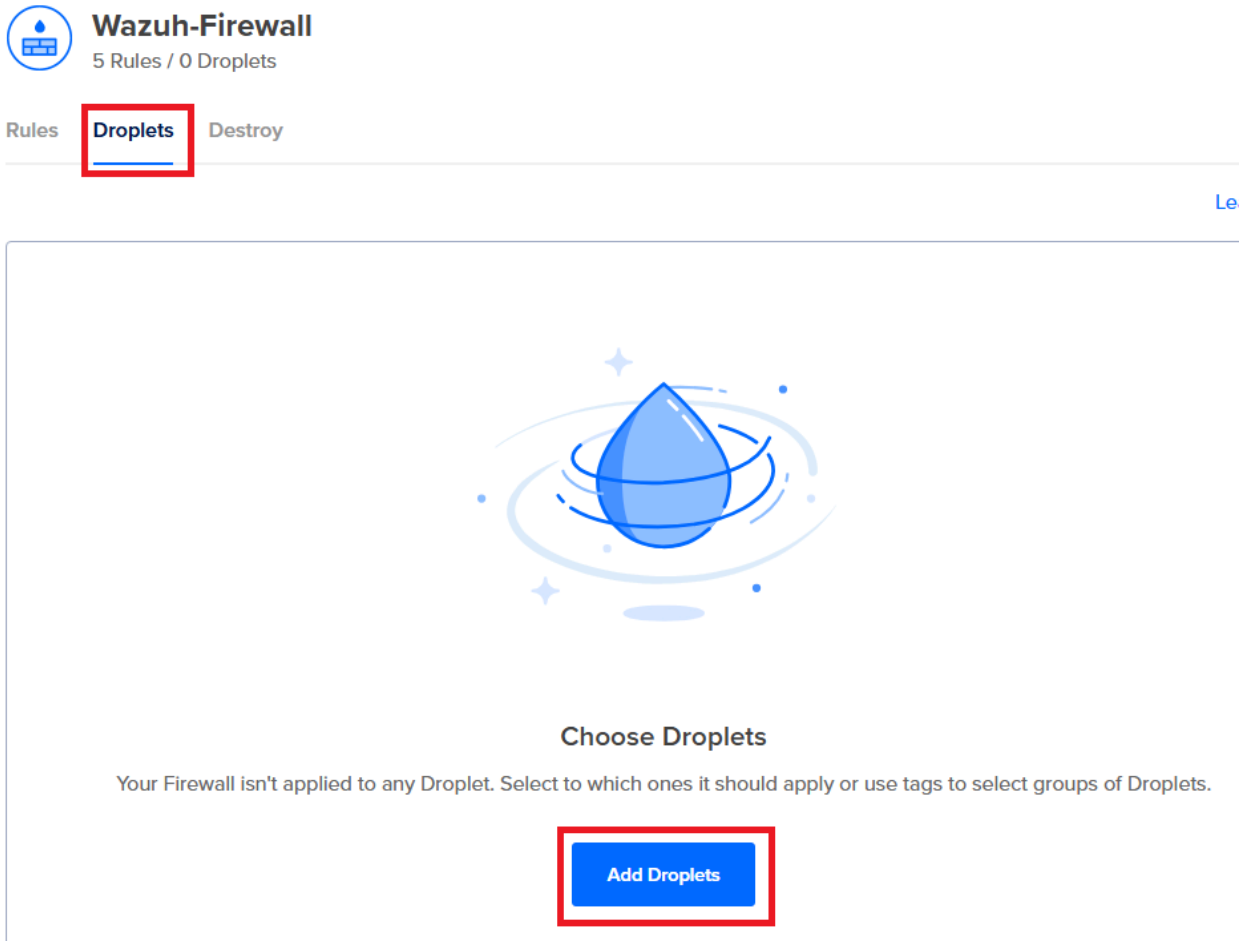


Then scroll down to the bottom to find the “**Firewalls**” section. Click on “**Edit**.”

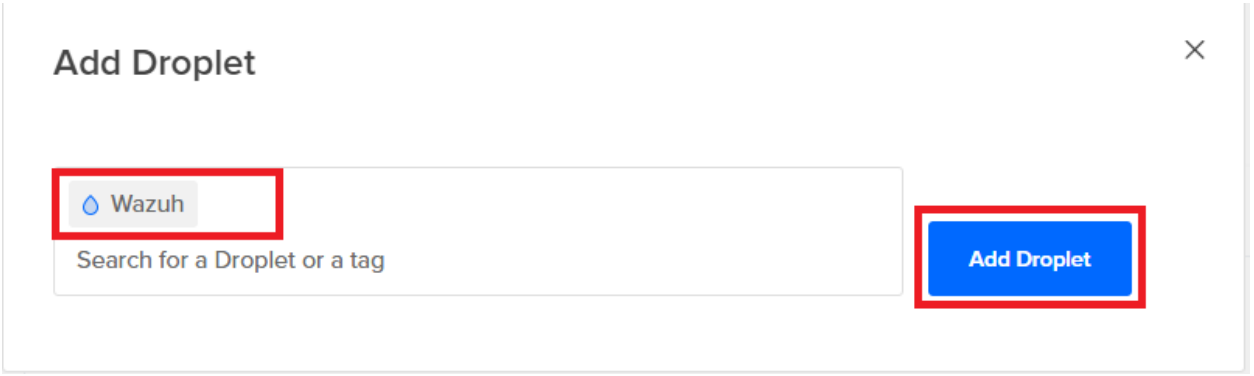
Click on your Wazuh Firewall.



Select the “**Droplets**” tab, then click on the “**Add droplet**” button.



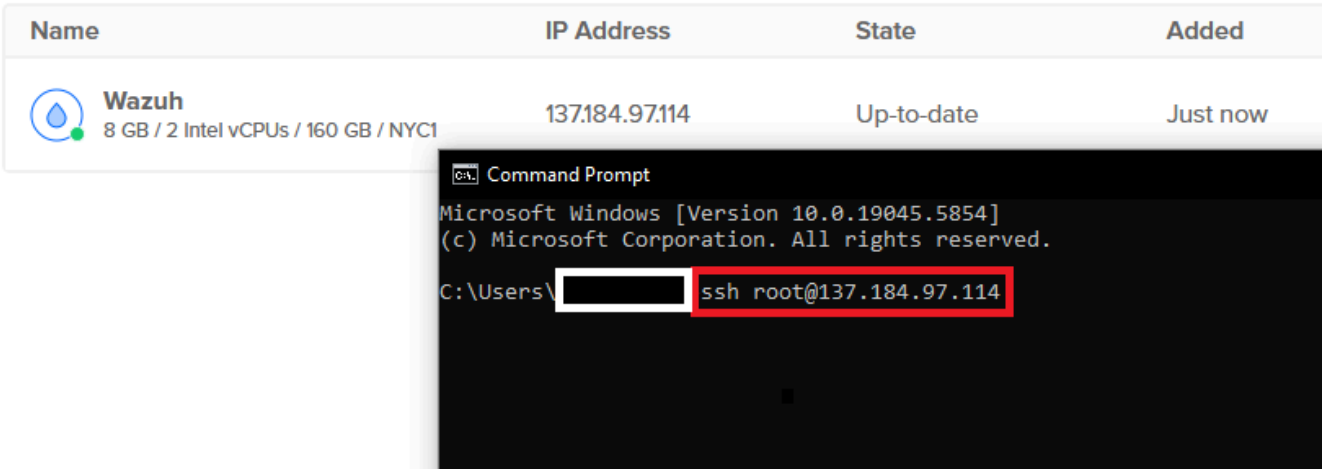
Type in “**Wazuh**”, select it, then click on the “**Add Droplet**” button.



Subsection - Initial commands in Wazuh

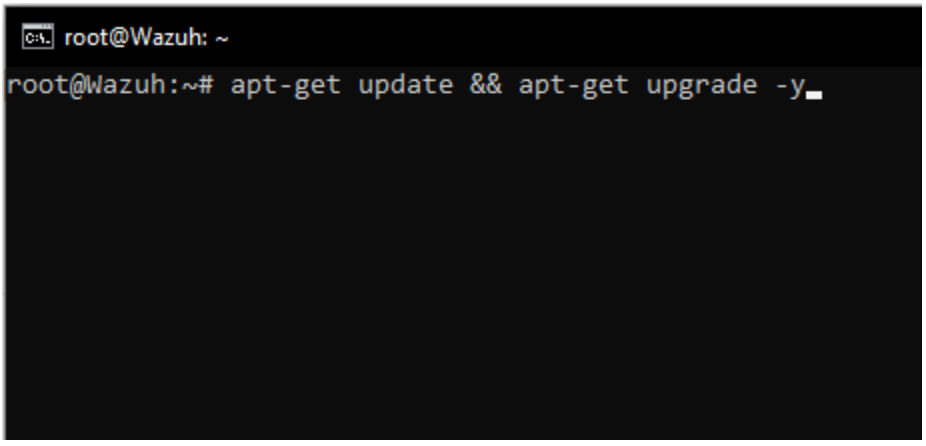
Log in to the Wazuh server using SSH on your Windows terminal/command prompt. You can use Putty as well.

 # The command format is "ssh <user>@<IP>
 # Capitalization matters
 # ssh root@<IP>

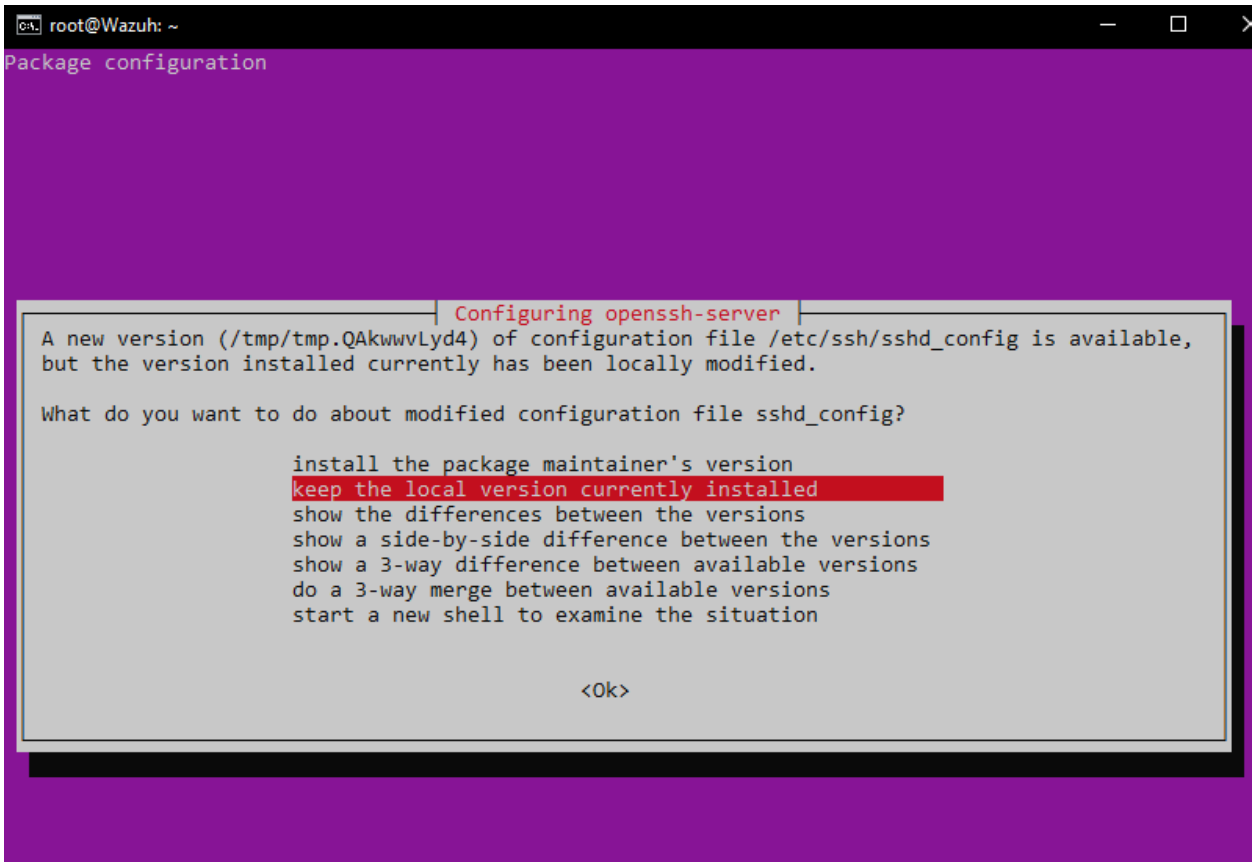


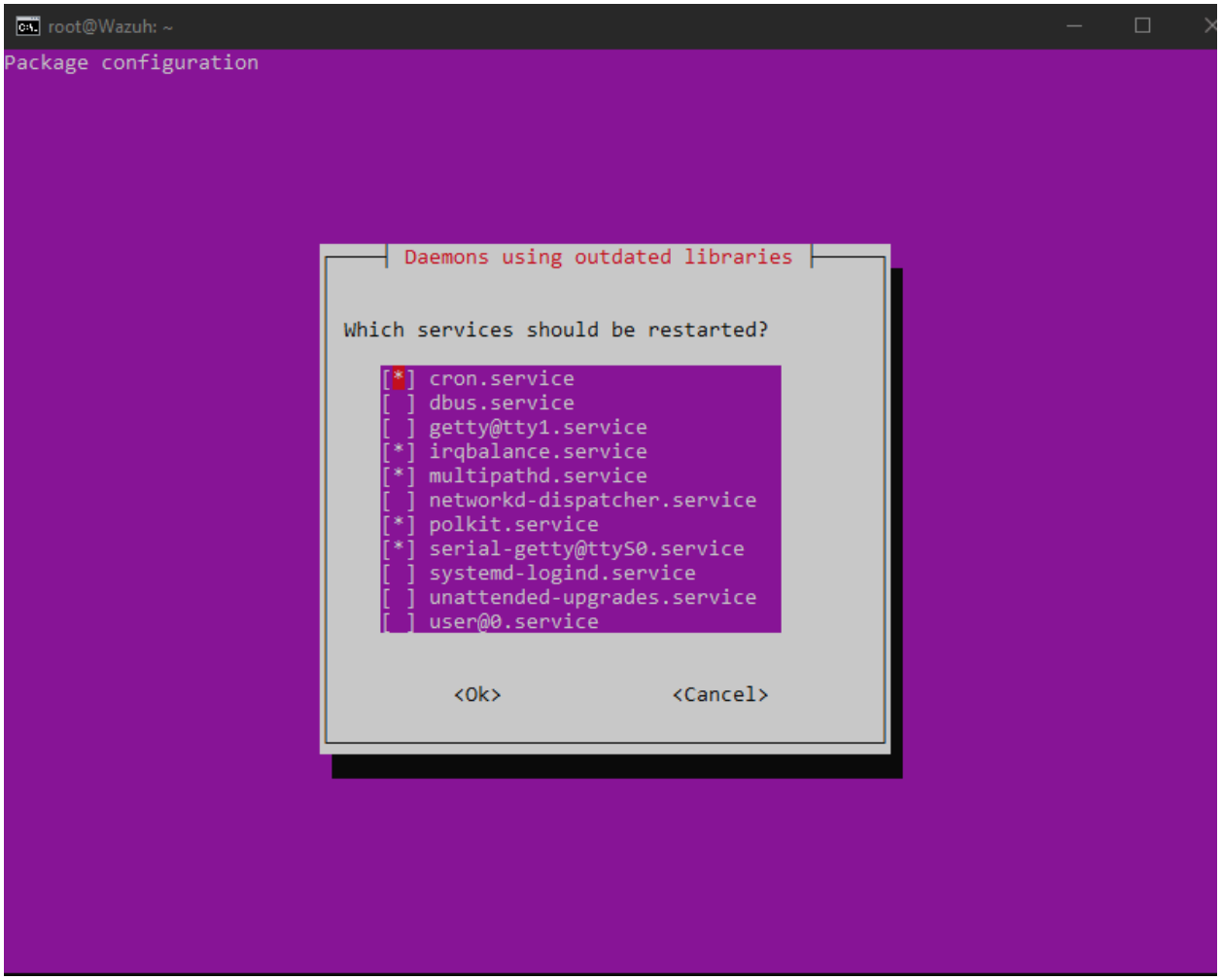
Input "**yes**" to continue with the connection. Input your password.

Run an APT update and upgrade as shown below.



When you get these next two screens, just hit “ENTER” on your keyboard.

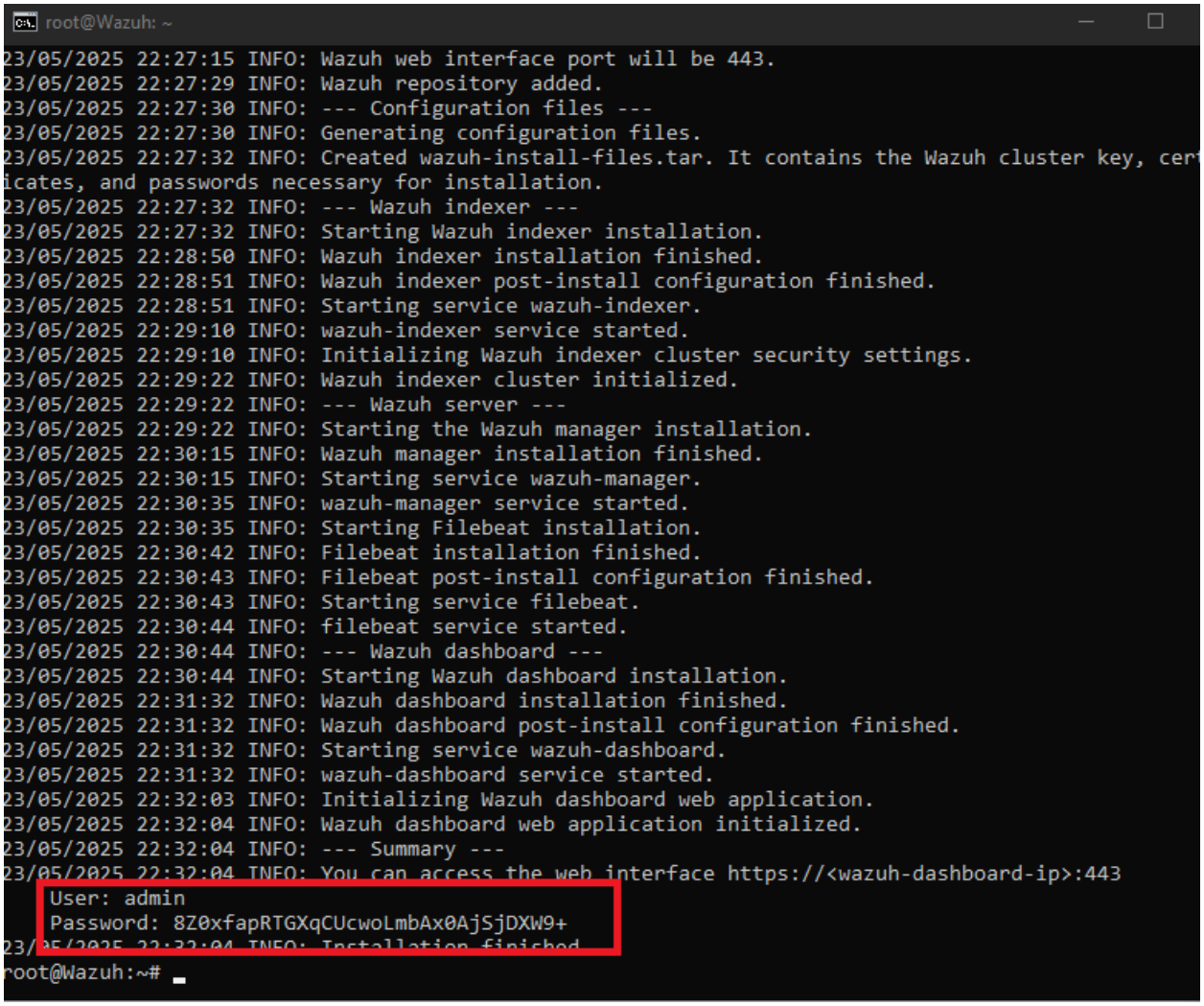



root@Wazuh: ~
Package configuration
Daemons using outdated libraries
Which services should be restarted?
[*] cron.service
[ ] dbus.service
[ ] getty@tty1.service
[*] irqbalance.service
[*] multipathd.service
[ ] networkd-dispatcher.service
[*] polkit.service
[*] serial-getty@ttyS0.service
[ ] systemd-logind.service
[ ] unattended-upgrades.service
[ ] user@0.service
<Ok>
<Cancel>

 # Once done updating and upgrading, we can install Wazuh with this curl command.

curl -sO https://packages.wazuh.com/4.7/wazuh-install.sh && sudo bash ./wazuh-install.sh -a

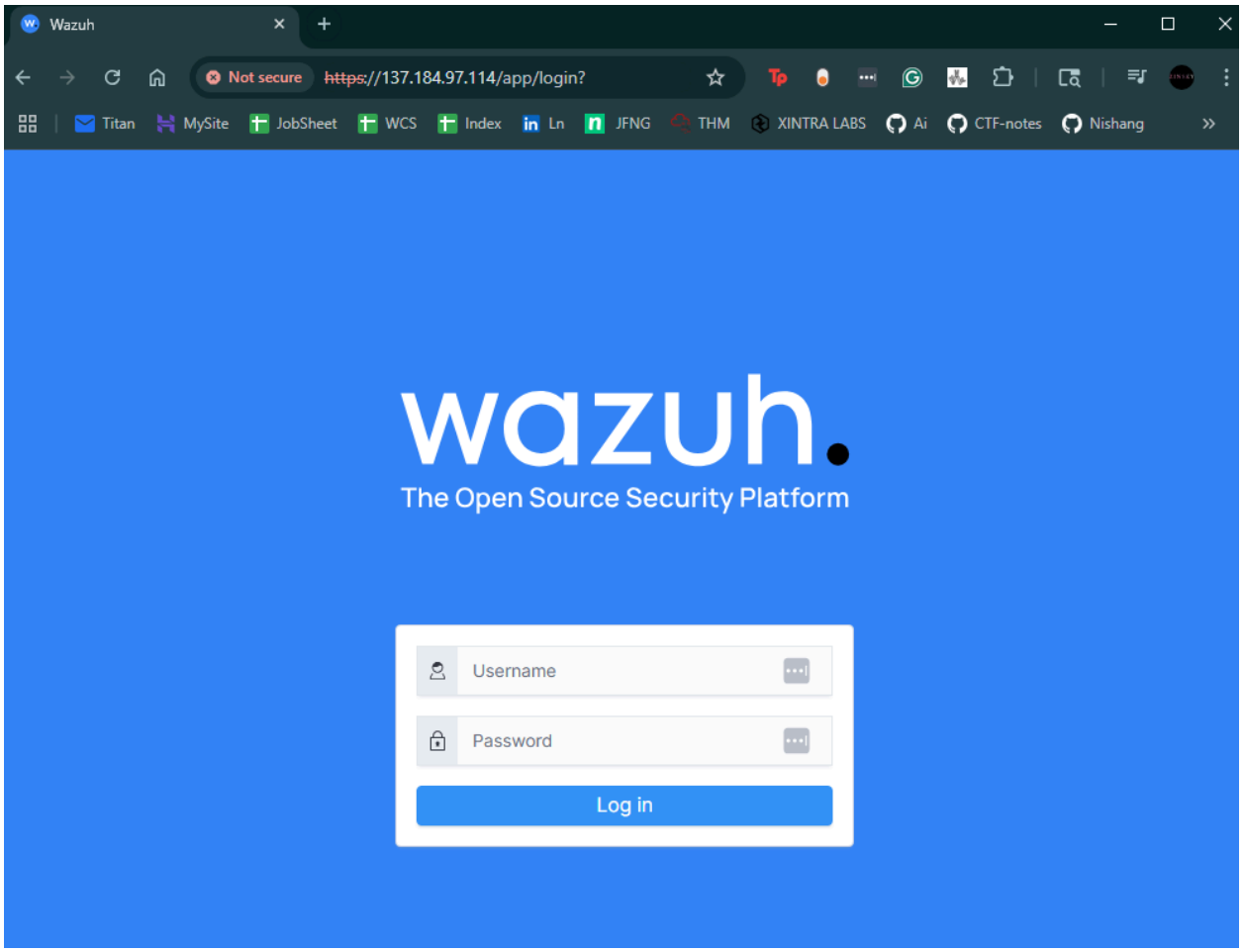
 # After installing Wazuh, take note of the username and password



 # Navigate to the IP address of your droplet

Use either **https://<IP>** or the **<IP>:443** to confirm you can reach the Wazuh dashboard

 # with "<IP>" being the public IP of your Wazuh server.



Section 2 - TheHive Case Management Solution Initial Setup

To start setting up TheHive, let us create a new droplet. Screenshots are omitted as these steps mirror the Wazuh Server setup process.

Back in the dashboard, click on "**Create**", then select "**Droplets**."

Select the region closest to you. I selected "**New York**".

For the OS: Ubuntu, Version: 22.04 (LTS) x64

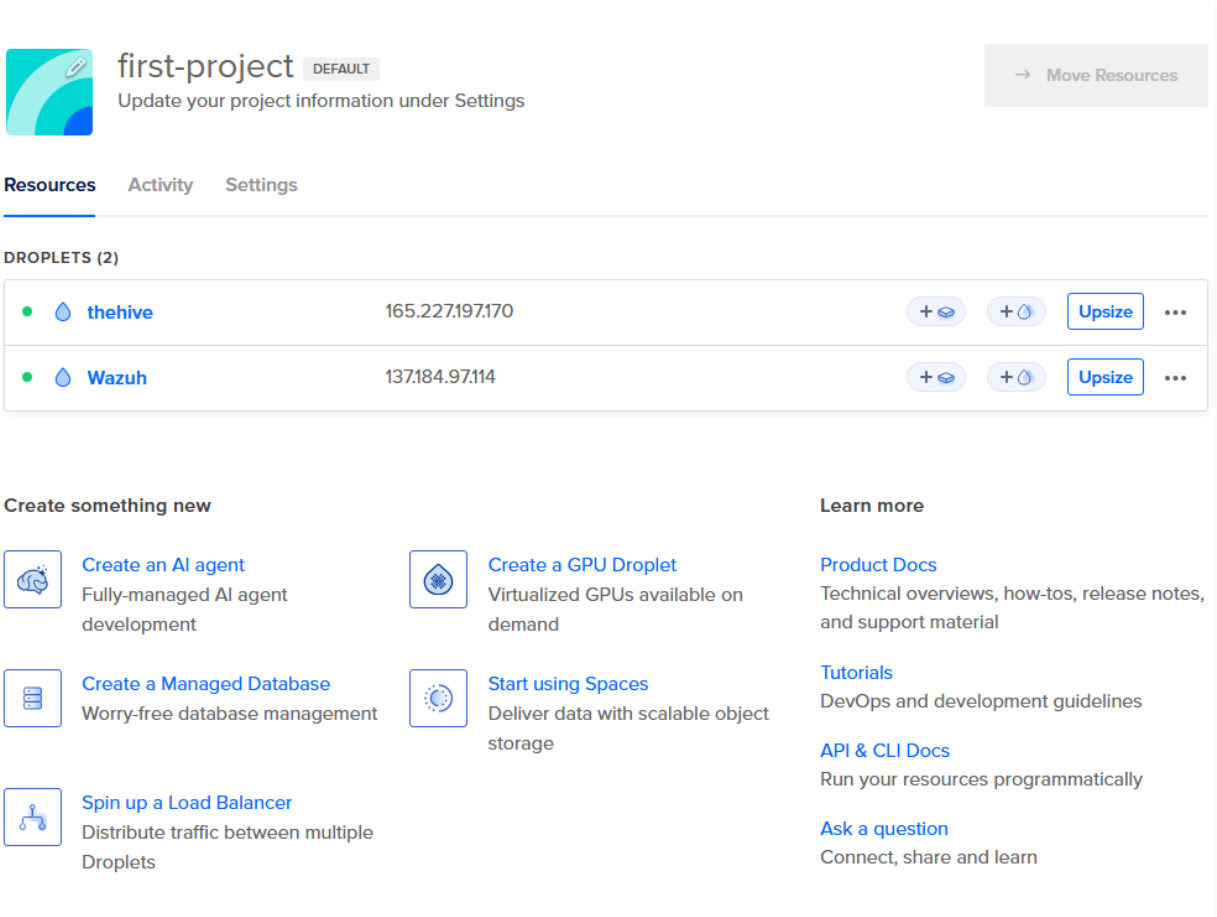
Droplet Type: Basic
CPU options: Premium Intel
$48 option for 8GB of memory

Authentication Method: Password (Input a strong password)

Hostname thehive

Subsection - TheHive Firewall

After creating the droplet, you should be redirected to your project listing both droplets. Let us add “thehive” to a firewall group.



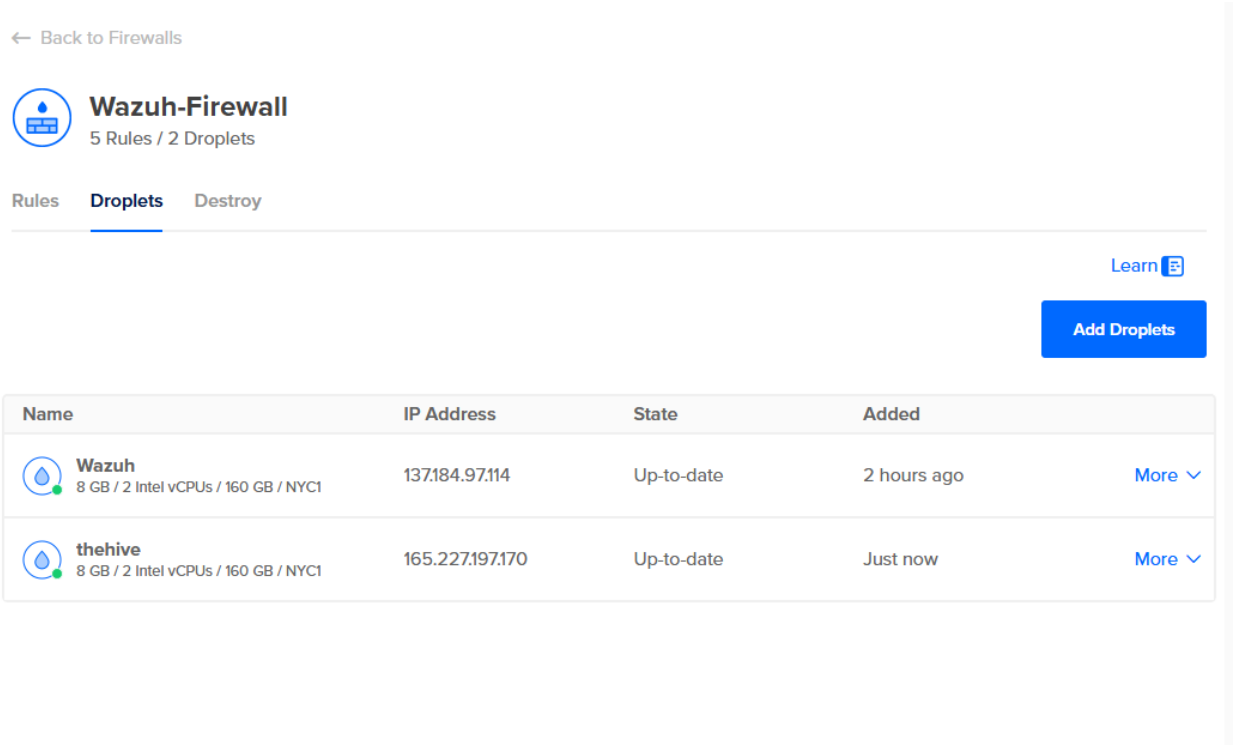
We can add TheHive to the Wazuh-Firewall by clicking on TheHive's droplet.

Select “**networking**” on the left-hand side.

Scroll down to find the “**Firewall**” section, then click on “**Edit**”.

Click on the firewall, select the “**Droplets**” tab. Then click on the “**Add Droplet**” button.

Type in “**thehive**”, click on it, then click on the “**Add Droplet**” button.



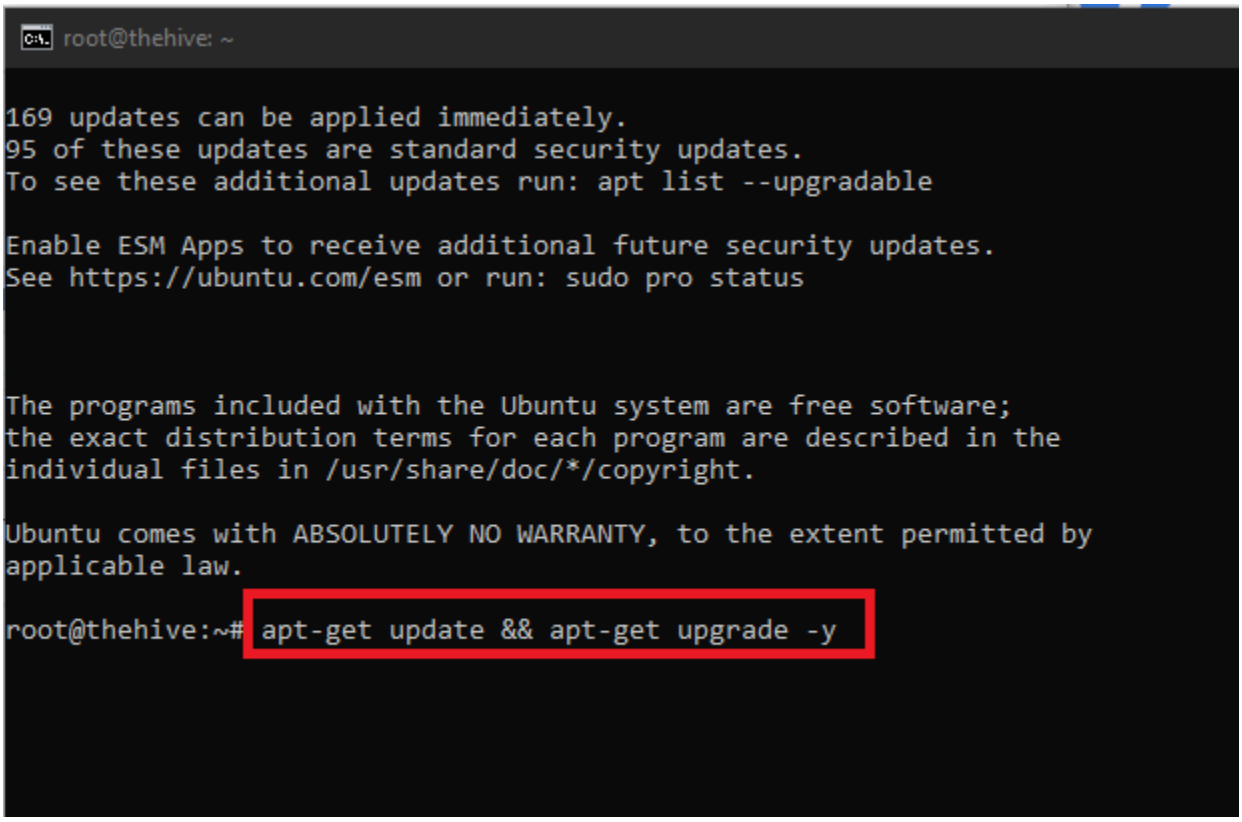
Subsection - Initial Commands in TheHive

Log in to TheHive server using SSH on your Windows terminal. You can use Putty as well.

 # The command format is "ssh <user>@<IP>
 # Capitalization matters
 # ssh root@<IP>

Accept key fingerprints

Once connected via SSH, run an apt update and upgrade on the system.



Click "**ENTER**" on any red screens.

Once done updating, we need to get 4 components for this server.
(Java, Cassandra, ElasticSearch, and TheHive)
Run these commands one by one for these components.

 # For Java, 6 commands, click ENTER on any red screens:

```
wget -qO- https://apt.corretto.aws/corretto.key | sudo gpg --dearmor  -o
/usr/share/keyrings/corretto.gpg

echo "deb [signed-by=/usr/share/keyrings/corretto.gpg] https://apt.corretto.aws stable main" |
sudo tee -a /etc/apt/sources.list.d/corretto.sources.list

sudo apt update

sudo apt install java-common java-11-amazon-corretto-jdk

echo JAVA_HOME="/usr/lib/jvm/java-11-amazon-corretto" | sudo tee -a /etc/environment

export JAVA_HOME="/usr/lib/jvm/java-11-amazon-corretto"
```

 # For Cassandra, 4 commands, click ENTER on any red screens:

```
wget -qO -  https://downloads.apache.org/cassandra/KEYS | sudo gpg --dearmor  -o
/usr/share/keyrings/cassandra-archive.gpg

echo "deb [signed-by=/usr/share/keyrings/cassandra-archive.gpg]
https://debian.cassandra.apache.org 40x main" |  sudo tee -a
/etc/apt/sources.list.d/cassandra.sources.list

sudo apt update

sudo apt install cassandra
```

 # For ElasticSearch, 5 commands, click ENTER on any red screens:
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg

sudo apt-get install apt-transport-https

echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-7.x.list

sudo apt update

sudo apt install elasticsearch

 # For TheHive, 4 commands, click ENTER on any red screens:
wget -O- https://archives.strangebee.com/keys/strangebee.gpg | sudo gpg --dearmor -o /usr/share/keyrings/strangebee-archive-keyring.gpg

echo 'deb [signed-by=/usr/share/keyrings/strangebee-archive-keyring.gpg] https://deb.strangebee.com thehive-5.2 main' | sudo tee -a /etc/apt/sources.list.d/strangebee.list

sudo apt-get update

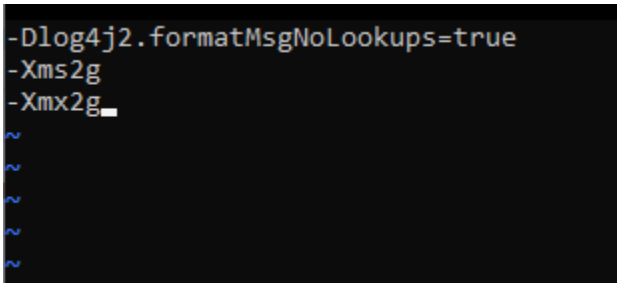
sudo apt-get install -y thehive

 # After running those commands, make a new file that will help with Elasticsearch.

Vim /etc/elasticsearch/jvm.options.d/jvm.options

 # And here is what to put in that file

-Dlog4j2.formatMsgNoLookups=true
-Xms2g
-Xmx2g

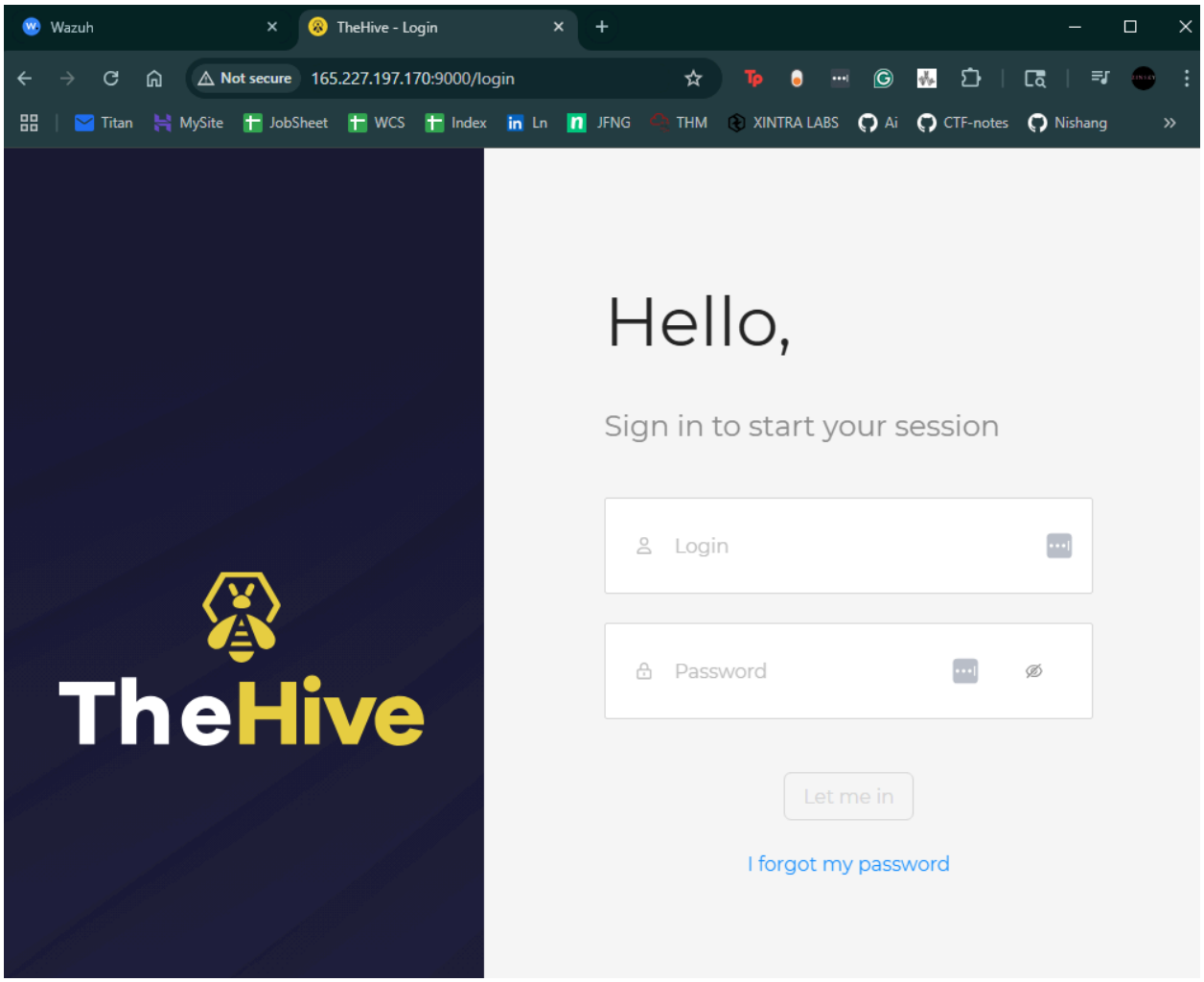


 # Press "Escape" on your keyboard, and type in ":wq" to save and exit Vim.

 # After that, do these commands individually

Systemctl stop elasticsearch
Rm -rf /var/lib/elasticsearch/*
Systemctl start elasticsearch
Systemctl restart thehive

After that, we should be able to navigate to the public IP of our Hive server on port 9000.



Section 3 - TheHive Additional Configurations

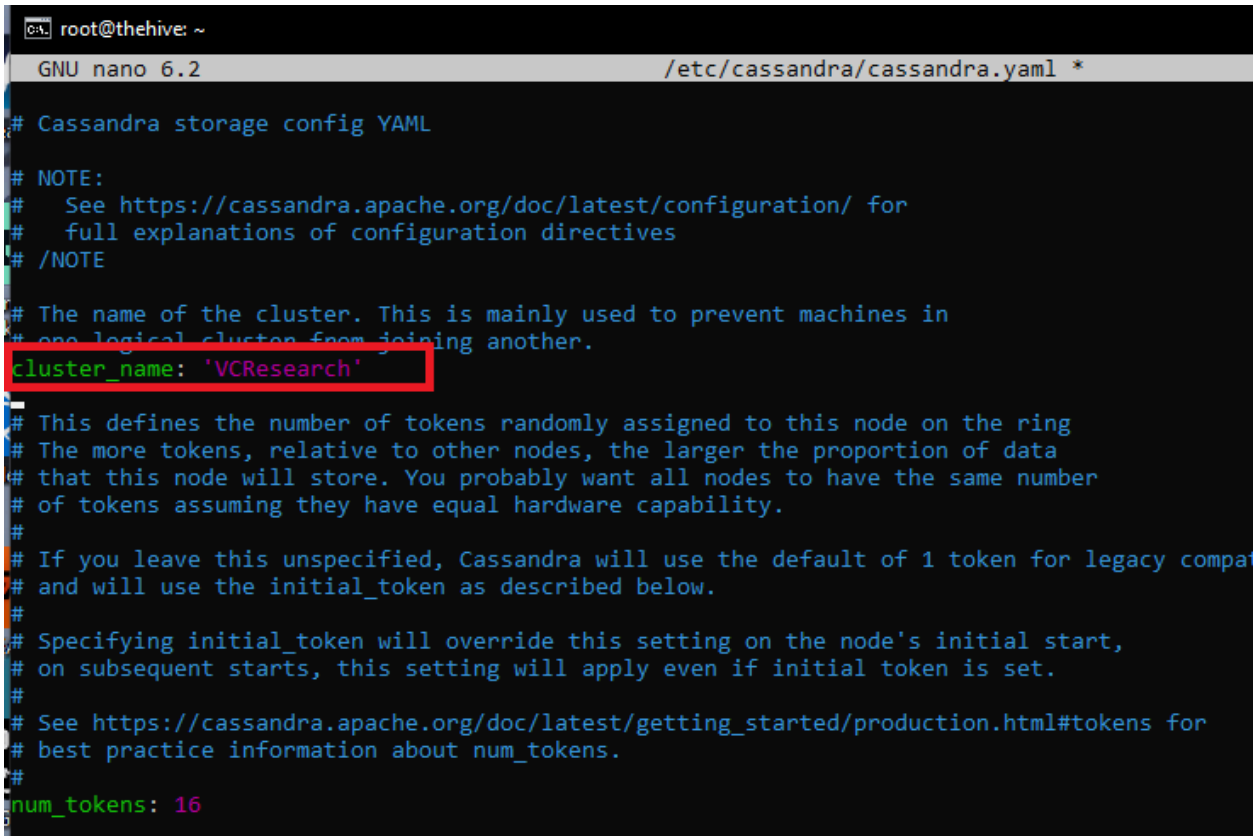
Going back to the SSH terminal for TheHive, we need to change some configurations for the components of the server.

Subsection - Cassandra Configurations

The first changes will be tied to Cassandra.

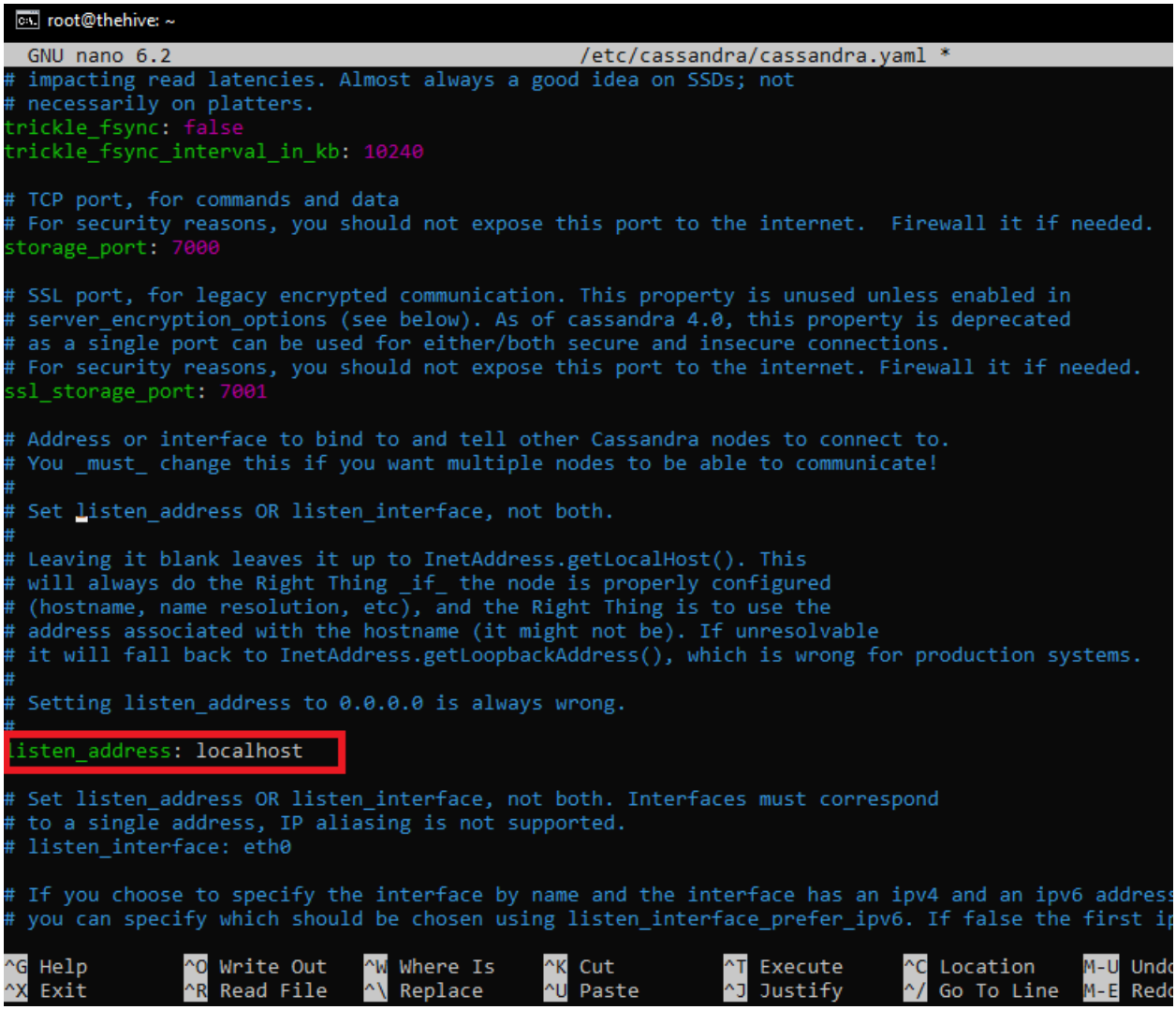
 # Nano into Cassandra's YAML file
Nano /etc/cassandra/cassandra.yaml

The first thing that I changed was the cluster name from "**Test Cluster**" to "**VCResearch**". You can leave it or change it to something you like.

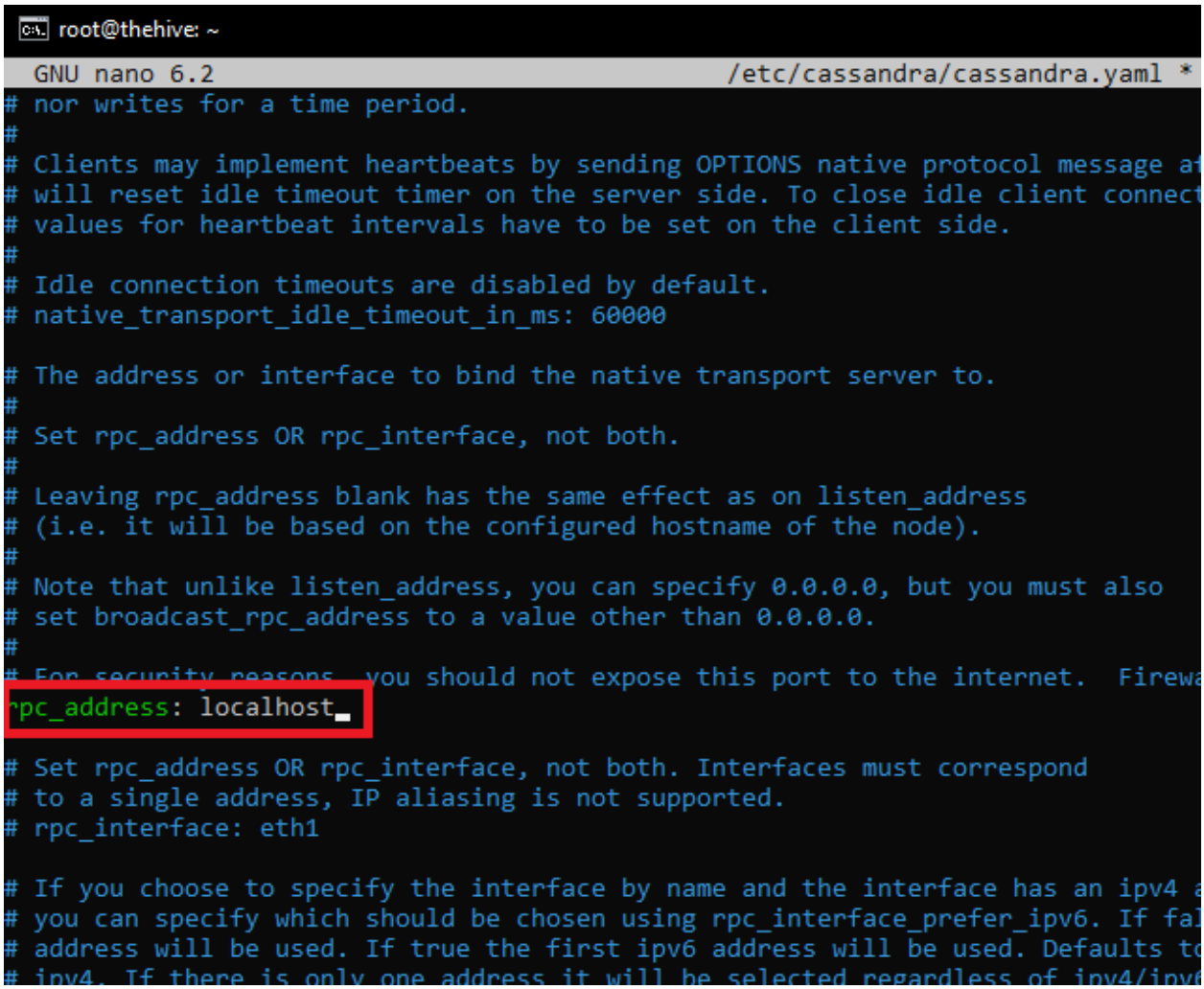


After changing the cluster name, click on the "**CTRL+W**" keys to activate the search functionality of Nano. Type in "**listen_address**" and click "**ENTER**." This will take you to the spot as shown below.

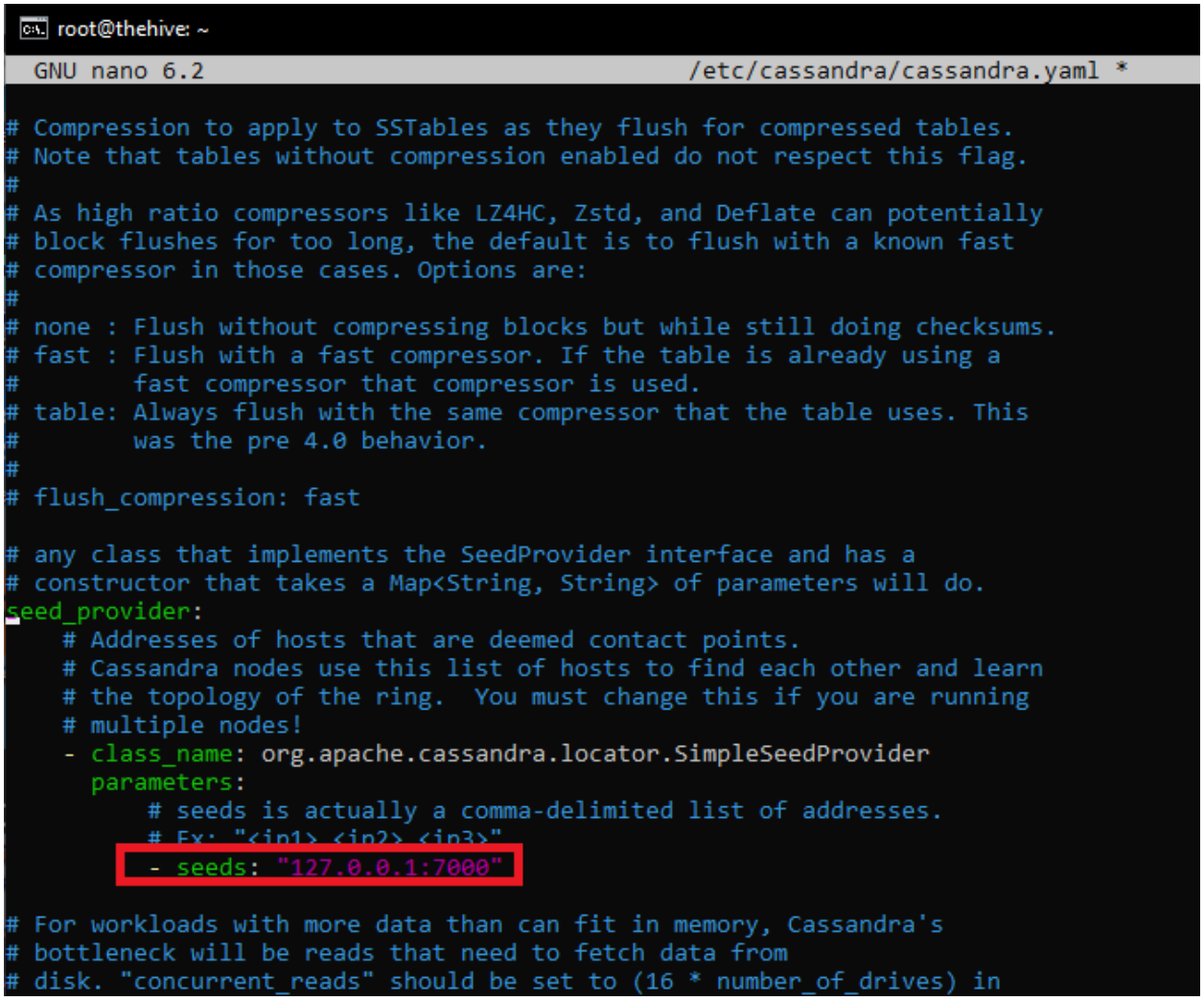
Replace the localhost with the public IP address of TheHive.



After inputting the listen address, scroll down or do the search feature again for "**rpc_address**", and replace localhost with the public IP of TheHive.



After inputting the rpc_address, use the search functionality to search for "**seed_provider**". Replace the 127.0.0.1 with the public IP of TheHive. Keep the port after the IP.



After those changes, on your keyboard, click on "ctrl+x" and Y to save, and then click "**ENTER**" to exit

 # Run these commands after exiting the YAML file.

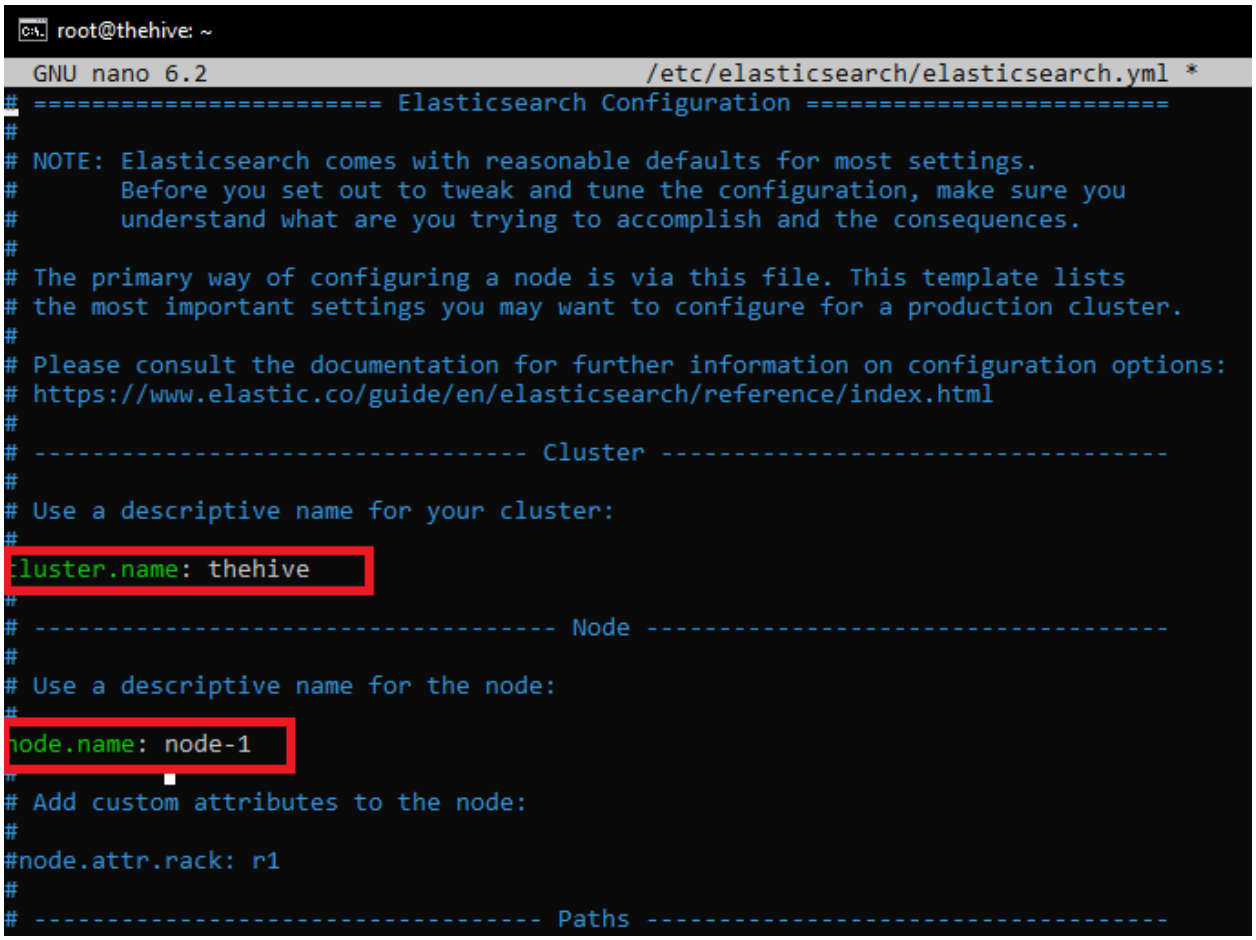
Systemctl stop cassandra.service
Rm -rf /var/lib/cassandra/*
Systemctl start cassandra.service
Systemctl status cassandra.service

Subsection - ElasticSearch Configurations.

 # Nano into the Elasticsearch yml file
Nano /etc/elasticsearch/elasticsearch.yml

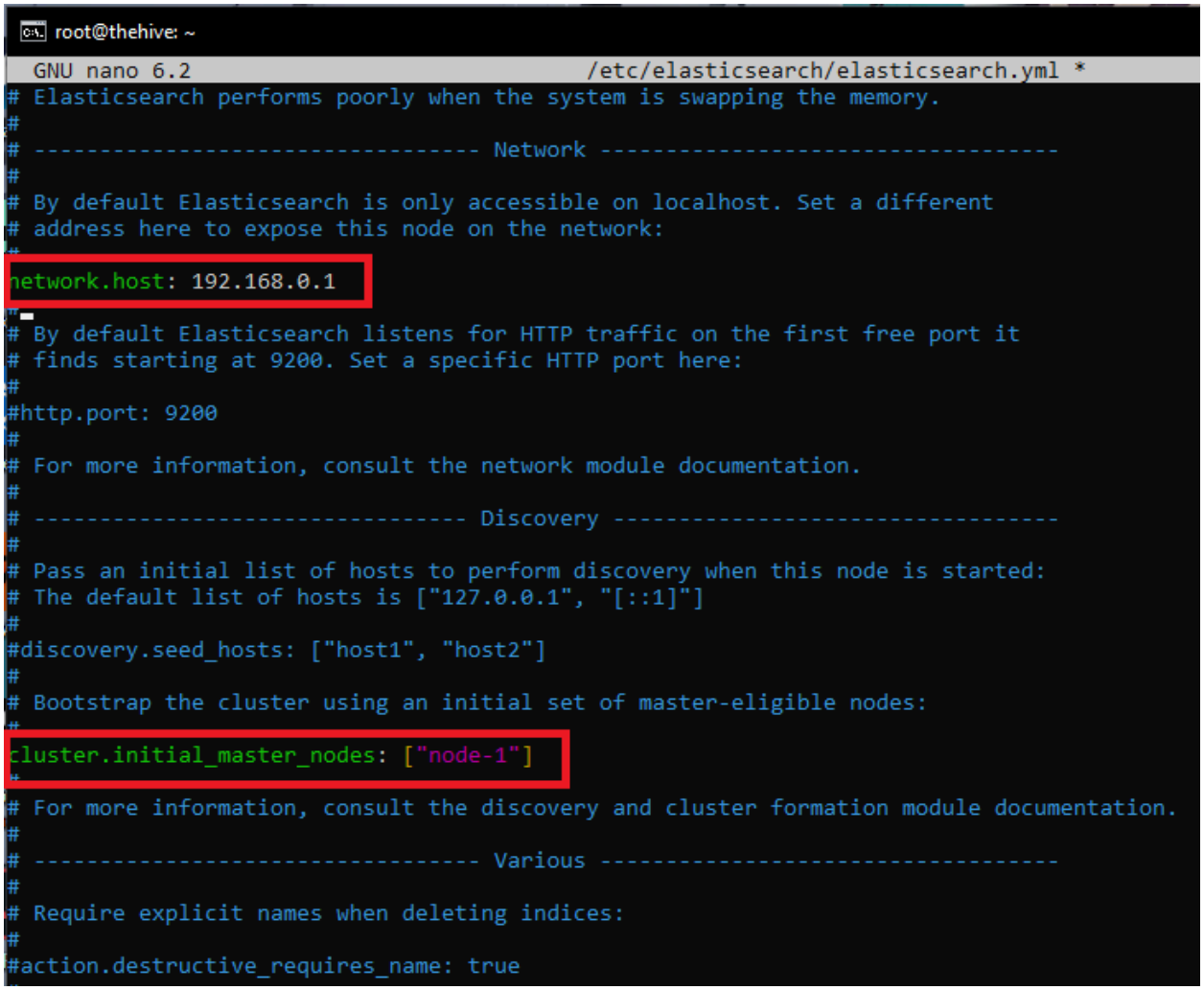
After opening the file, uncomment "**cluster.name**" and replace "**my-application**" with "**thehive**".

After that, uncomment "**node.name**".



Scroll down a bit past the "**paths and memory**" sections to the "**Network**" section and uncomment "**network.host**" and replace the placeholder IP with the public IP of TheHive.

Then, uncomment "**cluster.initial_master_nodes**" and delete "**node_2**" from the array.



 # After the changes, click "ctrl+x" and "Y" to save and click "ENTER" to exit

Systemctl start elasticsearch
Systemctl enable elasticsearch
Systemctl status elasticsearch

 # Check the status of Cassandra and verify that it is running

 # Check permissions on /opt/thp. If root is the owner of "thehive" change it
ls -la /opt/thp
Chown -R thehive:thehive /opt/thp

```
root@thehive:~# ls -la /opt/thp
total 12
drwxr-xr-x 3 root root 4096 May 24 00:53 .
drwxr-xr-x 5 root root 4096 May 24 00:53 ..
drwxr-xr-x 5 root root 4096 May 24 00:53 thehive
root@thehive:~# chown -R thehive:thehive /opt/thp
root@thehive:~# ls -la /opt/thp
total 12
drwxr-xr-x 3 thehive thehive 4096 May 24 00:53 .
drwxr-xr-x 5 root    root    4096 May 24 00:53 ..
drwxr-xr-x 5 thehive thehive 4096 May 24 00:53 thehive
root@thehive:~#
```

 # Confirm change
ls -la /opt/thp

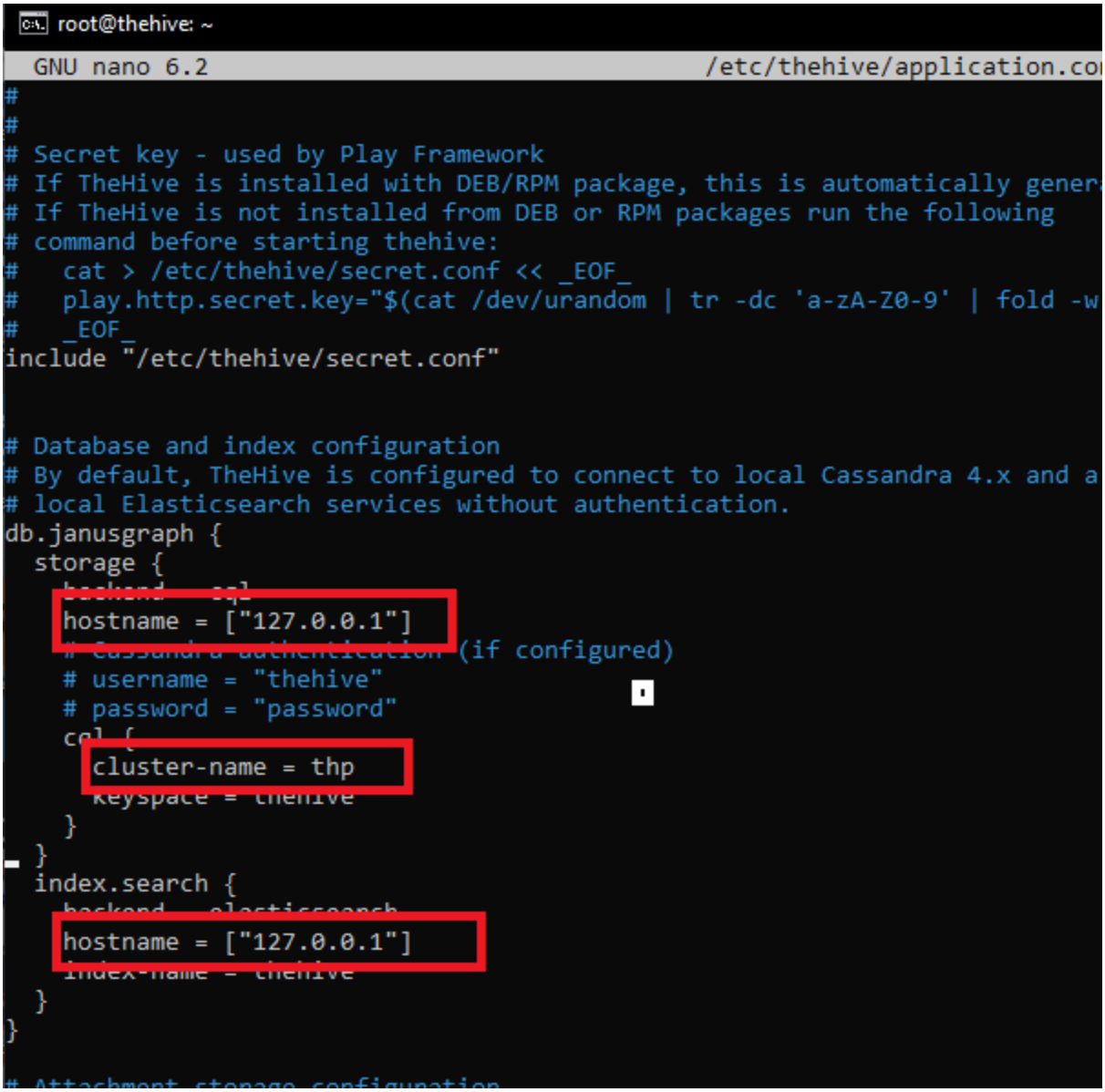
 # Nano into application.conf
Nano /etc/thehive/application.conf

Once inside of the application configuration file, scroll down to find the database and index configuration section.

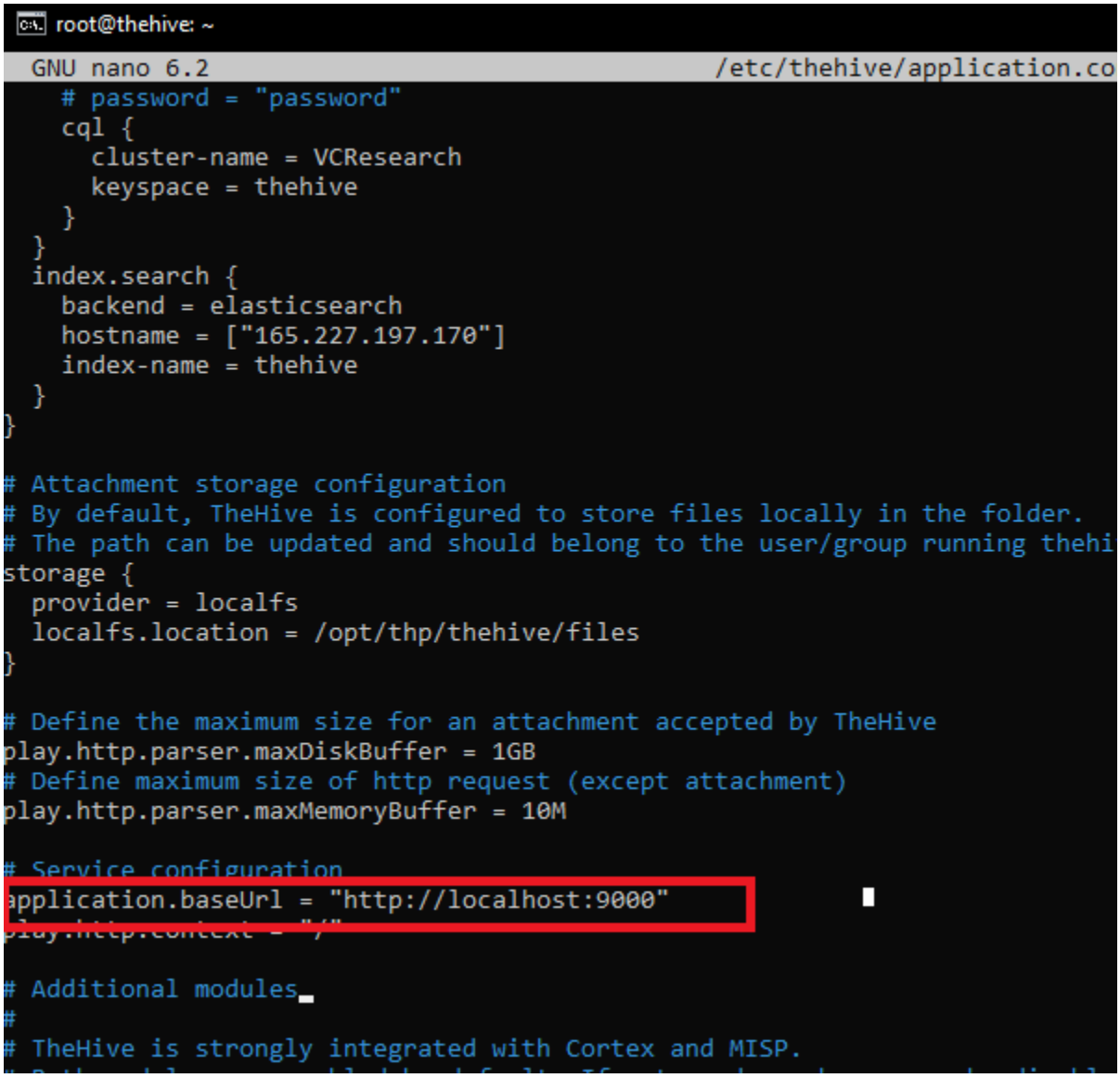
Under "**db.janusgraph**", it will have a "**hostname**" with the IP of "**127.0.0.1**", change it to the public IP of TheHive.

A bit lower, it will have a "**cluster-name**" section. I changed it to "**VCResearch**" to match the cluster name I put in the other config files.

Under "**index.search**", it will have a "**hostname**" with the IP of "**127.0.0.1**", change it to the public IP of TheHive


root@thehive: ~
GNU nano 6.2 /etc/thehive/application.co
Secret key - used by Play Framework
If TheHive is installed with DEB/RPM package, this is automatically gener
If TheHive is not installed from DEB or RPM packages run the following
command before starting thehive:
cat > /etc/thehive/secret.conf << _EOF_
play.http.secret.key="$(cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w
EOF
include "/etc/thehive/secret.conf"
# Database and index configuration
# By default, TheHive is configured to connect to local Cassandra 4.x and a
# local Elasticsearch services without authentication.
db.janusgraph {
storage {
hostname = ["127.0.0.1"]
(if configured)
username = "thehive"
password = "password"
cluster-name = thp
}
}
index.search {
hostname = ["127.0.0.1"]
}
}

After those changes, scroll down and look for "**application.baseurl**", it will have a URL pointing to localhost.
Change it to…
application.baseURL= "http://<TheHivePublicIP>:9000"



 # ctrl+x and Y to save and ENTER to exit

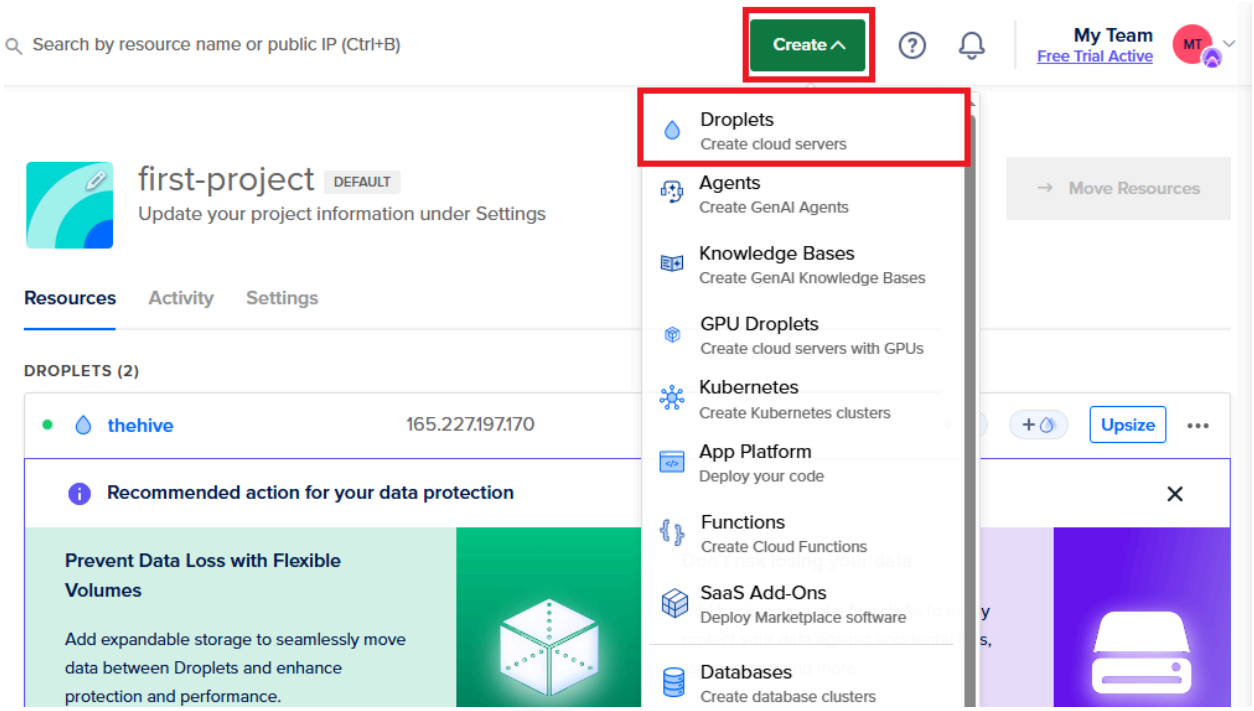
Systemctl start thehive
Systemctl enable thehive
Systemctl status thehive

 # Check the status of Cassandra, Elasticsearch, and TheHive. Confirm all are running

Section 4 - Linux Victim Honeypot Initial Setup and Agent Deployment

Let us deploy our Victim machine on DigitalOcean.

Click on the "**Create**" Button and select "**Droplets**" from the drop-down menu.



Select the region closest to you. I selected "**New York**".

For the OS: Ubuntu, Version: 22.04 (LTS) x64

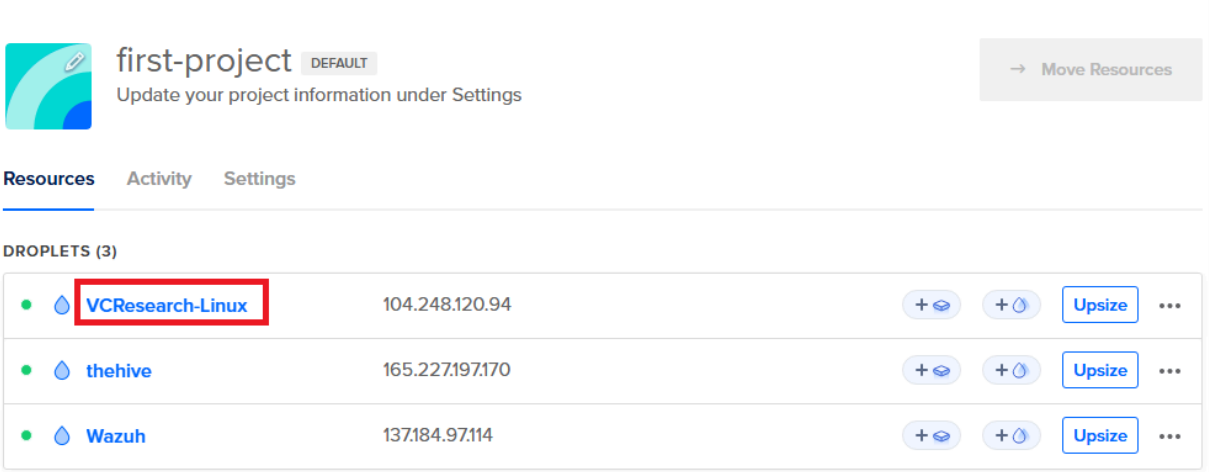
Droplet Type: Basic
CPU options: Regular
$18 option

Authentication Method: Password (Input a strong password)

Hostname VCResearch-Linux

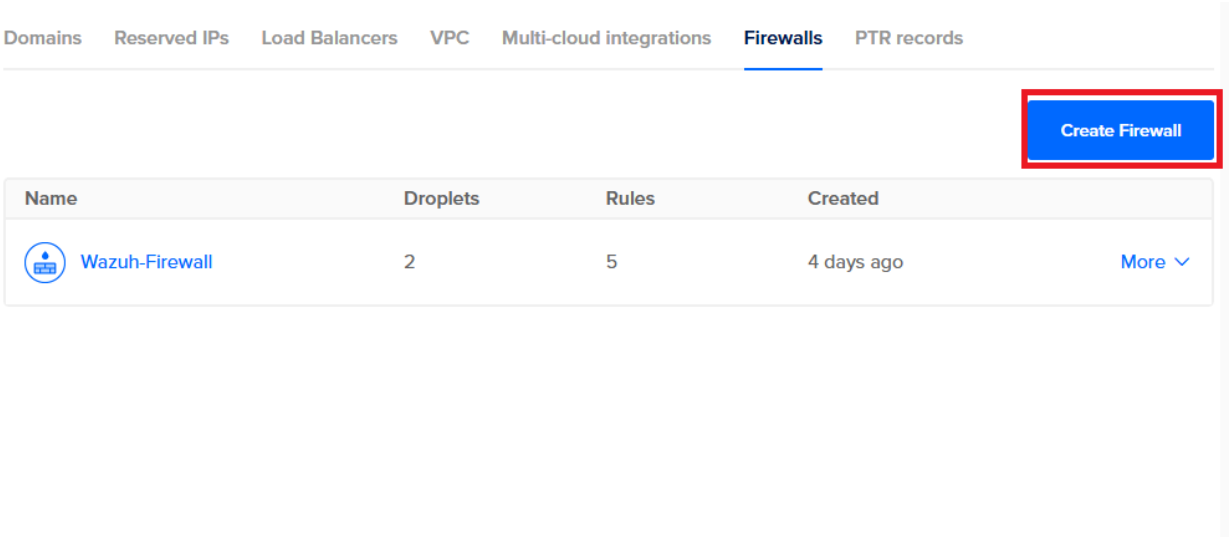
Subsection - Honeypot Firewall

Once you have created your droplet, let us make a firewall rule for it. Click on your new droplet.



On the left-hand side, click on "**Networking**". Scroll down to find the "**Firewalls**" section and click on the "**Edit**" button.

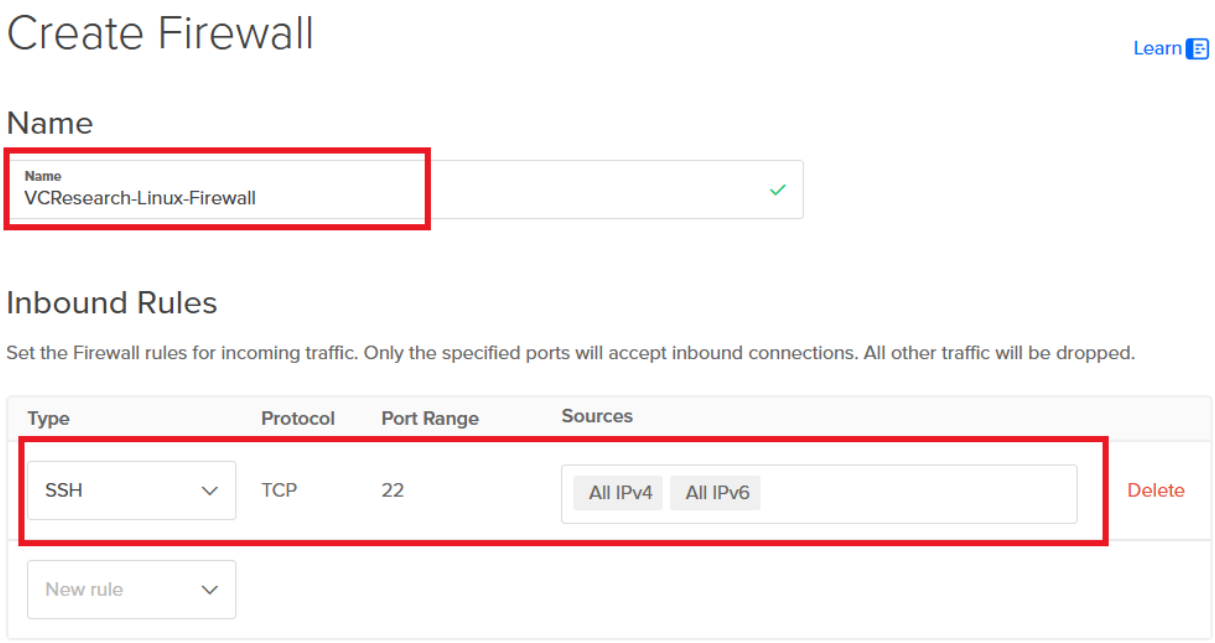
Then click on the "**Create Firewall**" button.



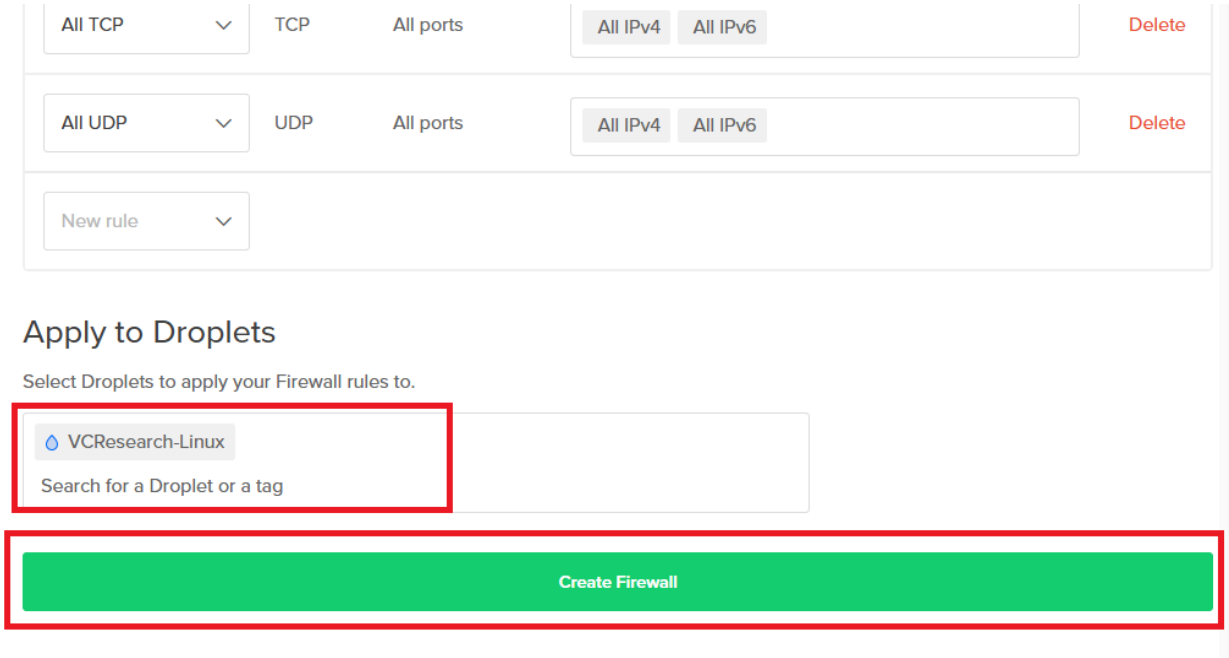
I named this firewall "**VCResearch-Linux-Firewall**". It is separate from the other servers because I will be exposing it to the internet once everything is up and running.

Just like the first firewall we created, we want to set two rules to allow all inbound TCP and UDP traffic to be allowed from our actual IP address.

 # In addition to our real IP, add the IP address for the Wazuh Server.



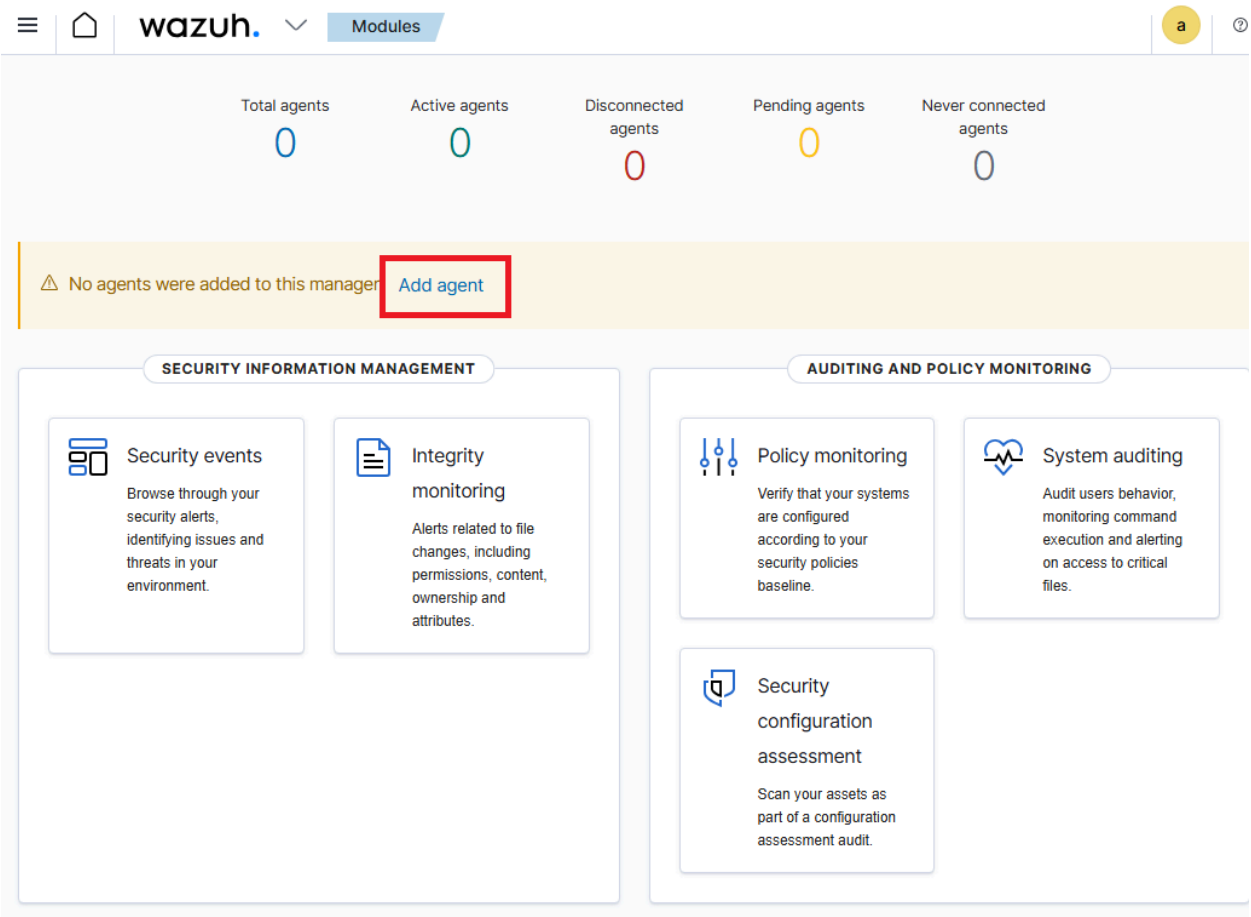
Once you have created the two rules, you can scroll down and apply the firewall to the Droplet we created. Once done with that, click on "**Create Firewall**".



Subsection - Agent Deployment

After creating and applying the new firewall, we need to log into the Wazuh Dashboard to get started on setting up the agent on the new Linux host. You will have to use the credentials that were given to you when you finished installing Wazuh.

Once logged in and on the dashboard, click on "**Add agent**".

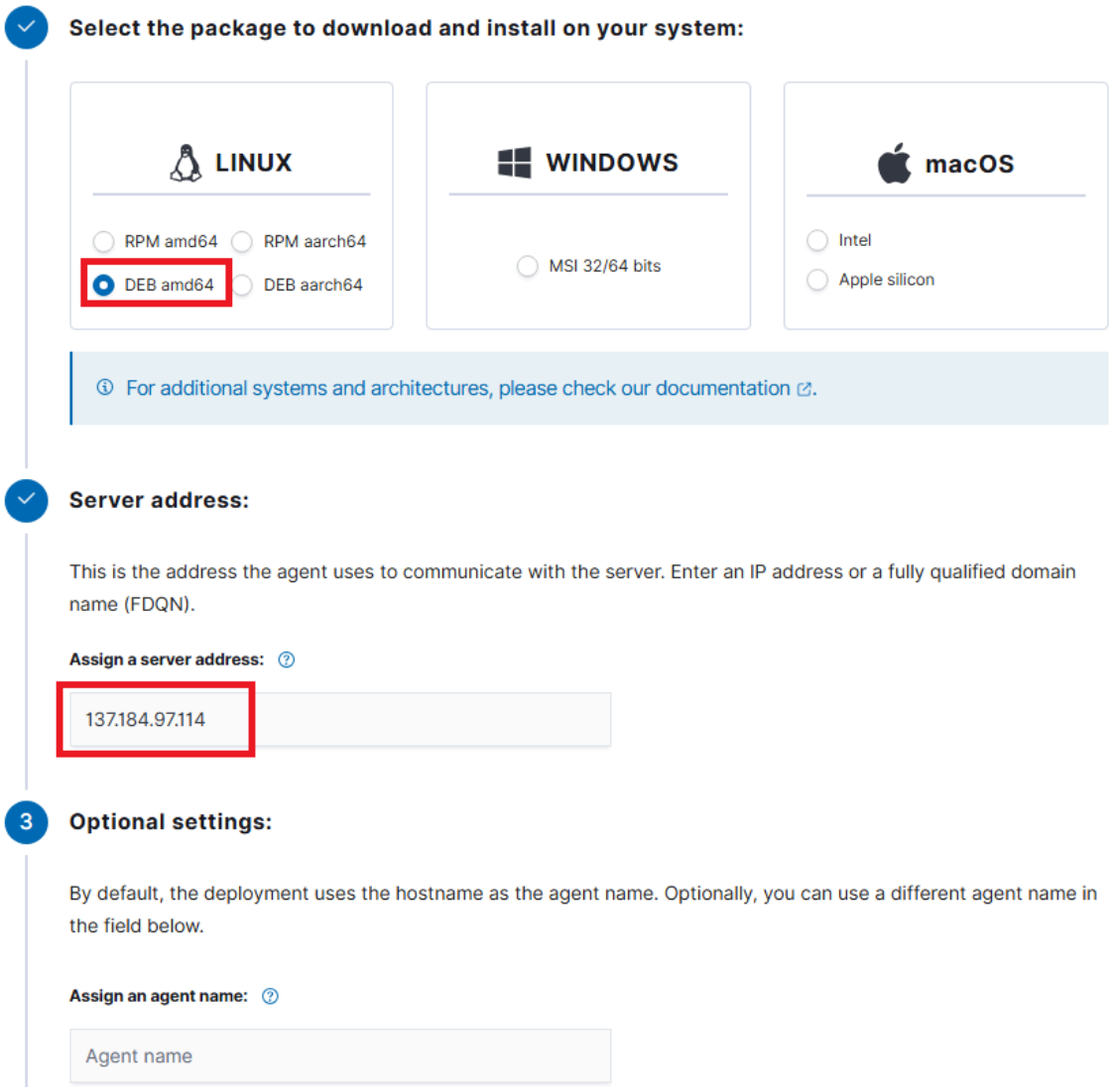


Under Linux, select “DEB amd64”.

For the Server address, input the Wazuh Public IP address.

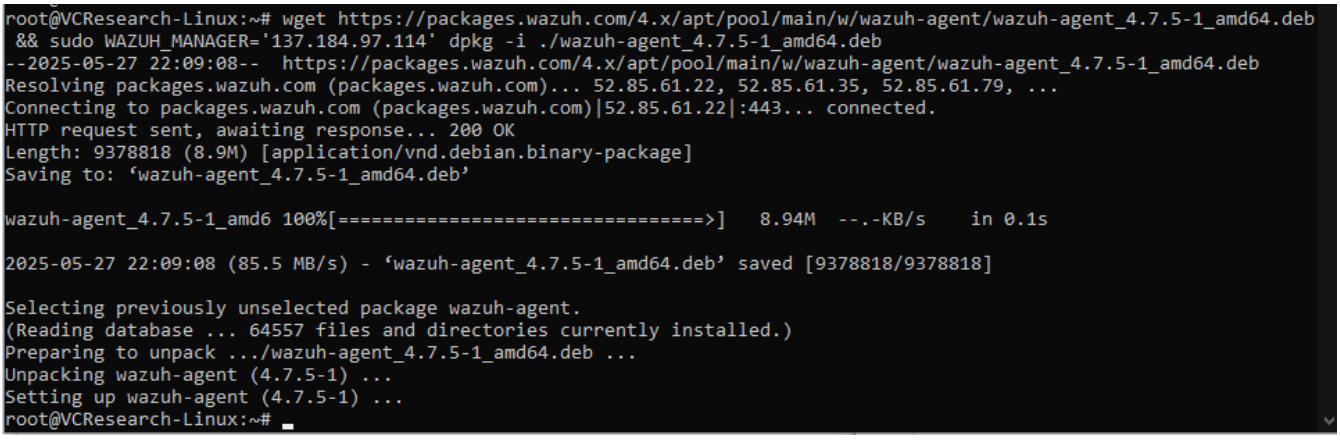
I did not add an Agent name as the hostname is fine.

Step 4 will contain the commands you need to run to set up the agent. So, let us connect to the Linux machine via SSH and do that.



On the Linux machine, once logged in for the first time. Run an **“apt-get update && apt-get upgrade”**. Click “**ENTER**” on any red screens.

The output after running the agent deployment commands should look like the output below.



On the “**deploy new agent**” page on Wazuh, it will allow you to copy the commands, but I recommend running them one by one manually. Running them all at once can lead to issues.

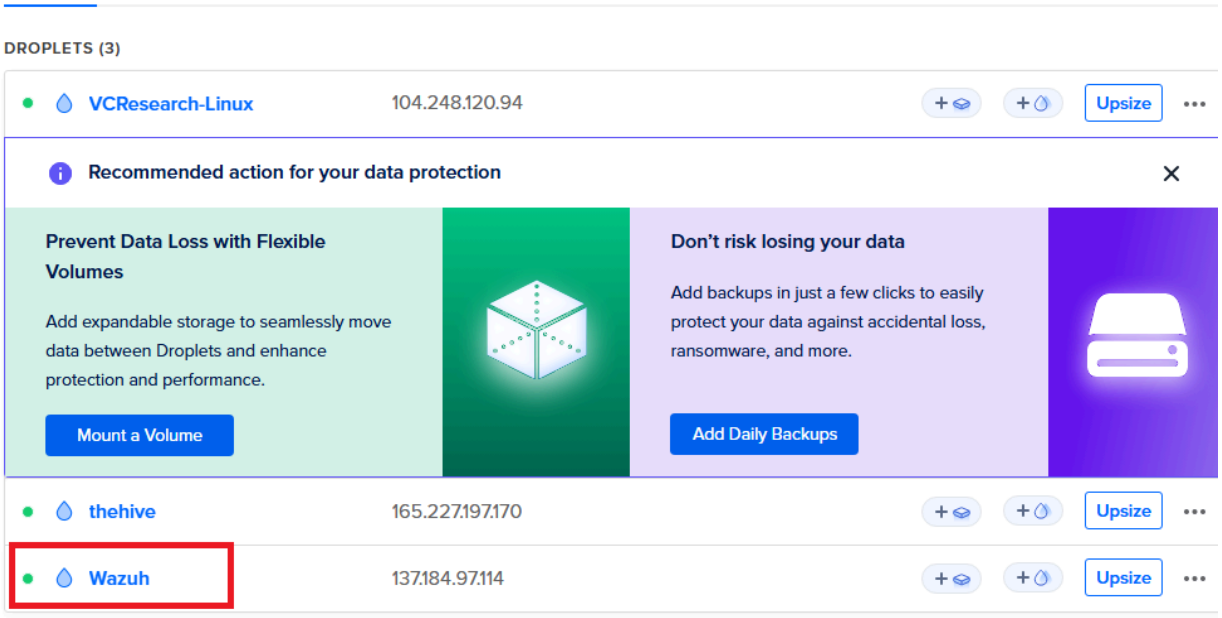
 # Commands are:

Systemctl daemon-reload
Systemctl enable wazuh-agent
Systemctl start wazuh-agent

 # Confirm agent is running with:
Systemctl status wazuh-agent

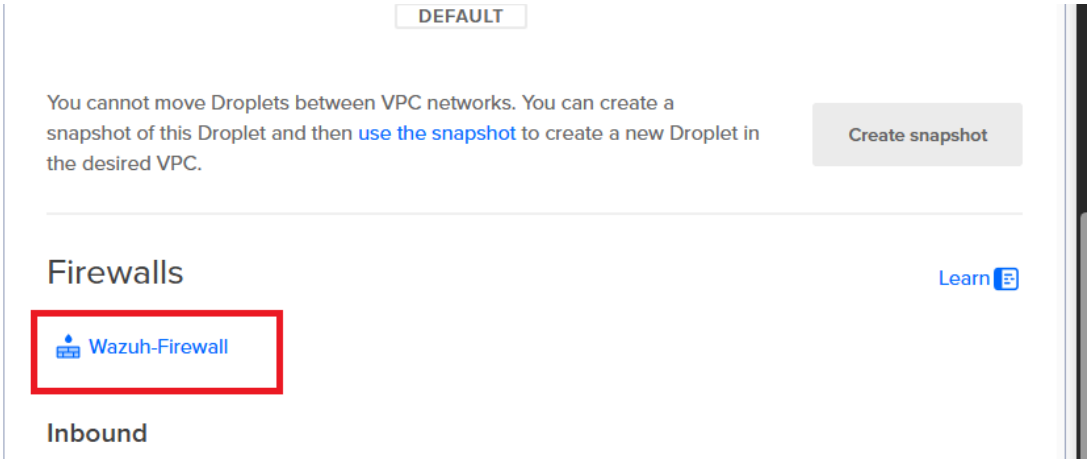
We will need to allow Wazuh to be able to communicate with the agent. We need to change the firewall rules that are applied to Wazuh.

On DigitalOcean, click on your Wazuh Droplet.



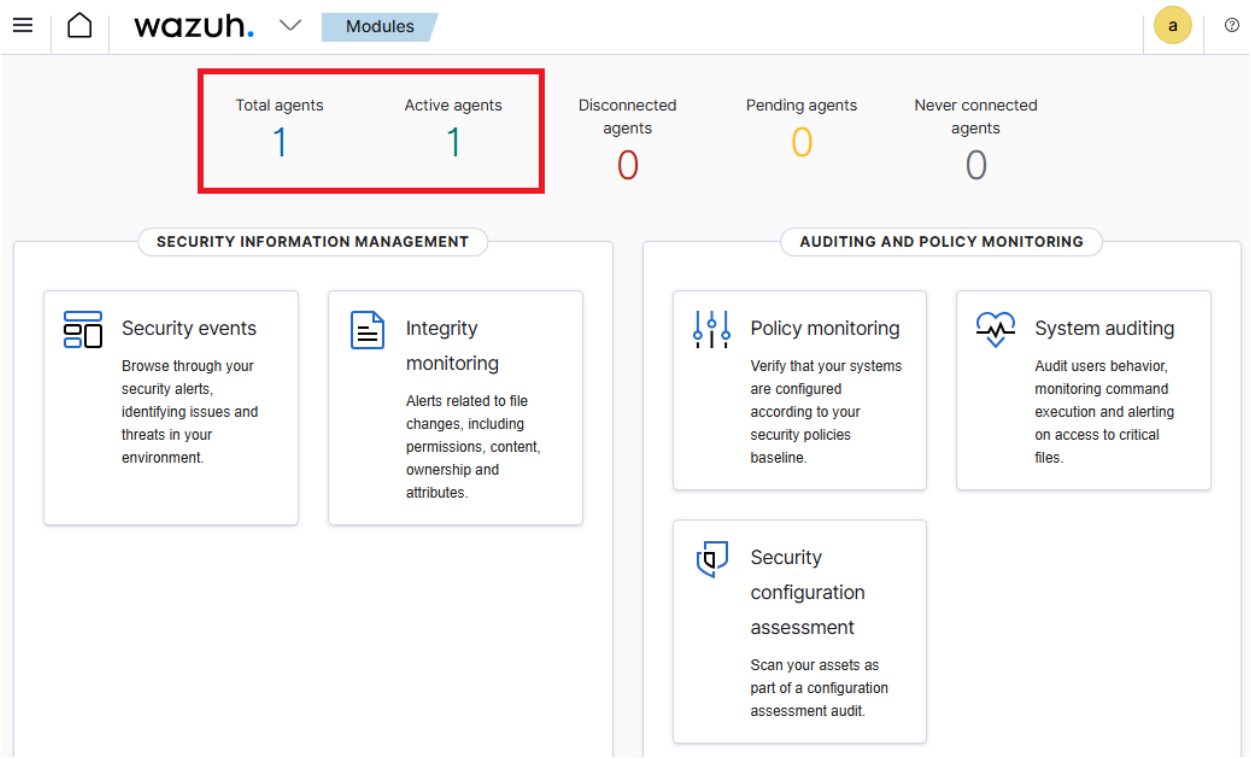
On the left-hand side, click on "**networking**". Then scroll down to the firewall section.

Click on "**Wazuh-Firewall**".



Once you are in the "**Rules**" page for the firewall, on the far-right side of the inbound rules, there will be a "**More**" dropdown. Click on it and select "**Edit**". Type in the Agent IP address and click "**ENTER**". Do this for the TCP and UDP rules.

Once you save those changes and check back in your Wazuh dashboard, you should be able to see an active agent.



Section 5 - Wazuh XDR Server Configurations

After deploying our agent, we need to do some additional configurations on the Wazuh Server.

 # On the Wazuh Server, create a backup for ossec.conf in case of any mistakes
cp /var/ossec/etc/ossec.conf ~/ossec-backup.conf

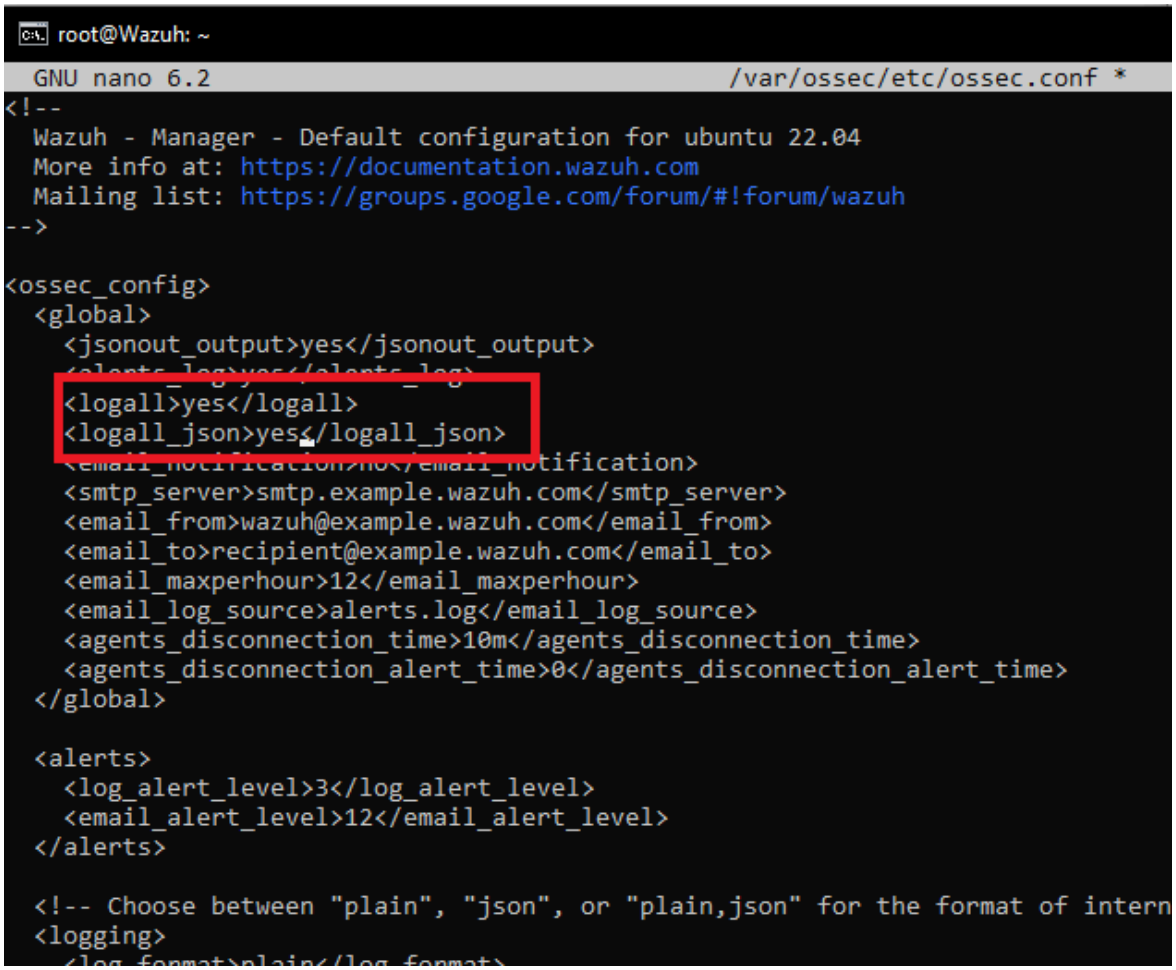
 # nano into the ossec file on the Wazuh server
nano /var/ossec/etc/ossec.conf

 # In the file, change these

Logall from “no” to “yes”
Logall_json “no” to “yes”

 # ctrl+x and Y to save, and ENTER to exit

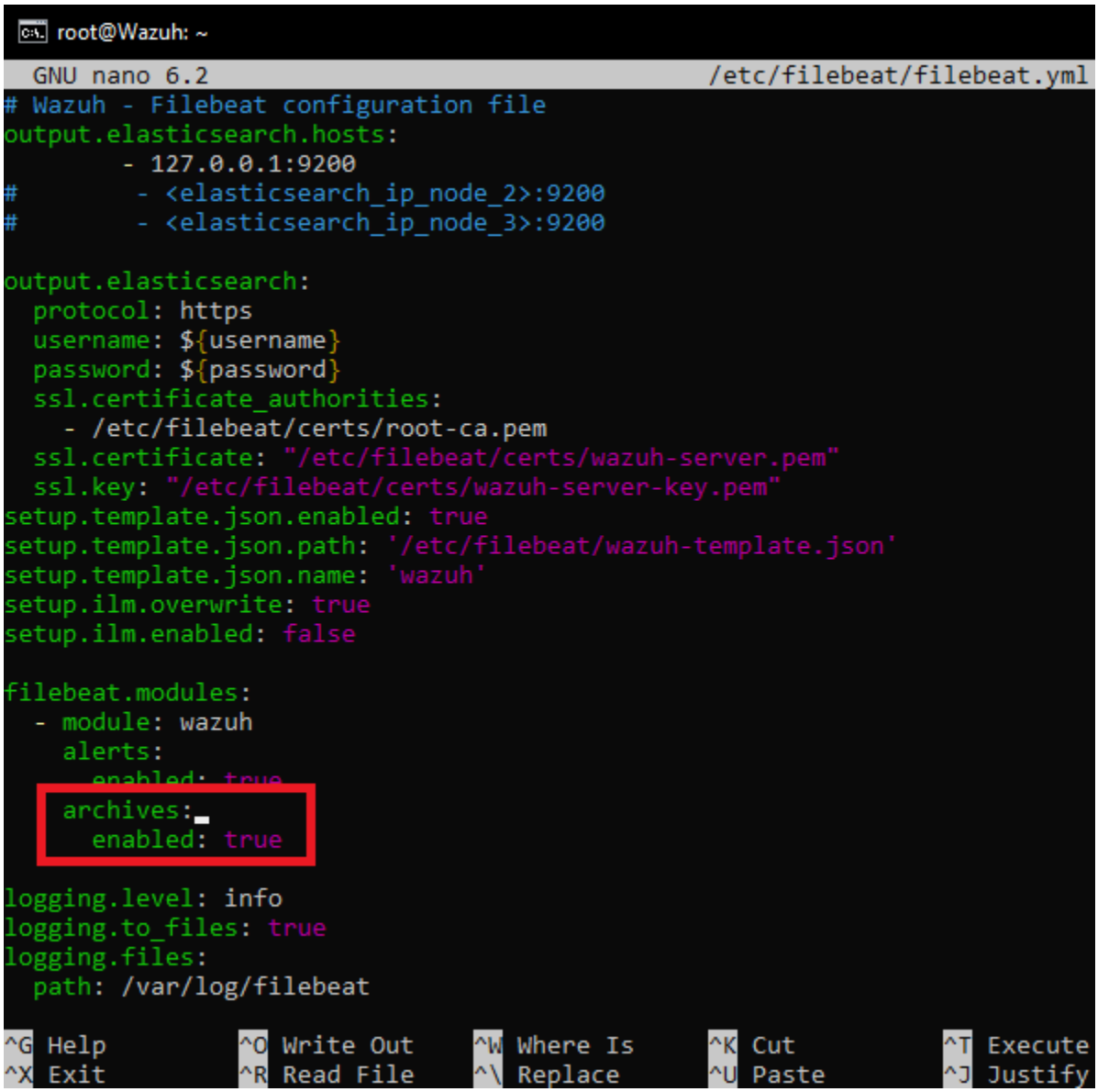
Systemctl restart wazuh-manager.service


root@Wazuh: ~
GNU nano 6.2 /var/ossec/etc/ossec.conf *
<!--
Wazuh - Manager - Default configuration for ubuntu 22.04
More info at: https://documentation.wazuh.com
Mailing list: https://groups.google.com/forum/#!forum/wazuh
-->
<ossec_config>
<global>
<jsonout_output>yes</jsonout_output>
<logall>yes</logall>
<logall_json>yes</logall_json>
<smtp_server>smtp.example.wazuh.com</smtp_server>
<email_from>wazuh@example.wazuh.com</email_from>
<email_to>recipient@example.wazuh.com</email_to>
<email_maxperhour>12</email_maxperhour>
<email_log_source>alerts.log</email_log_source>
<agents_disconnection_time>10m</agents_disconnection_time>
<agents_disconnection_alert_time>0</agents_disconnection_alert_time>
</global>
<alerts>
<log_alert_level>3</log_alert_level>
<email_alert_level>12</email_alert_level>
</alerts>
<!-- Choose between "plain", "json", or "plain,json" for the format of intern
<logging>

 # Just a note that Wazuh logs can be found at "/var/ossec/logs/archives"

 # To start ingesting these logs, we need to modify the filebeat file.
nano /etc/filebeat/filebeat.yml

 # Scroll down until you find filebeat modules, archives will be on "false", swap to "true"

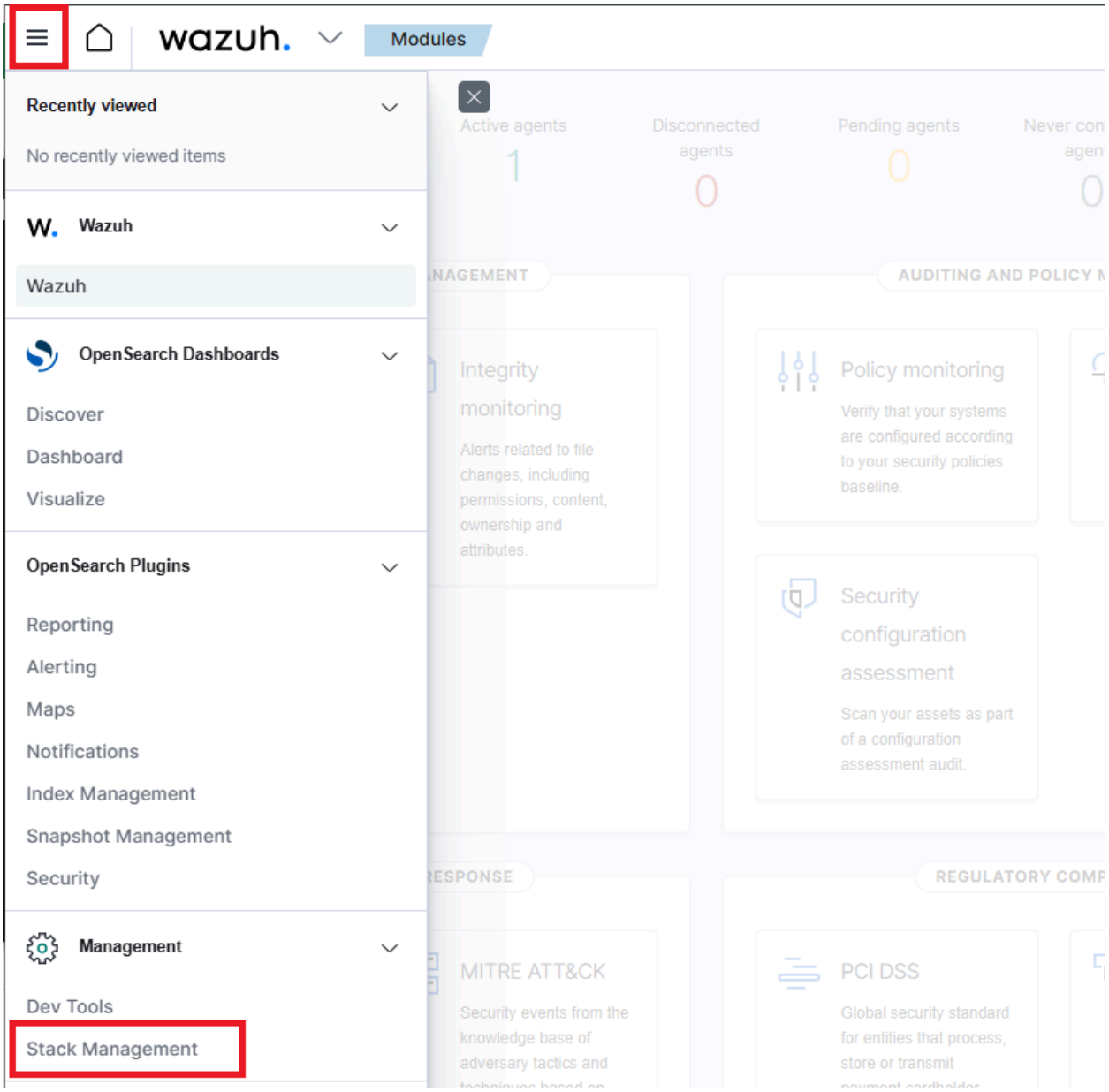


 # ctrl+x and Y to save, and ENTER to exit

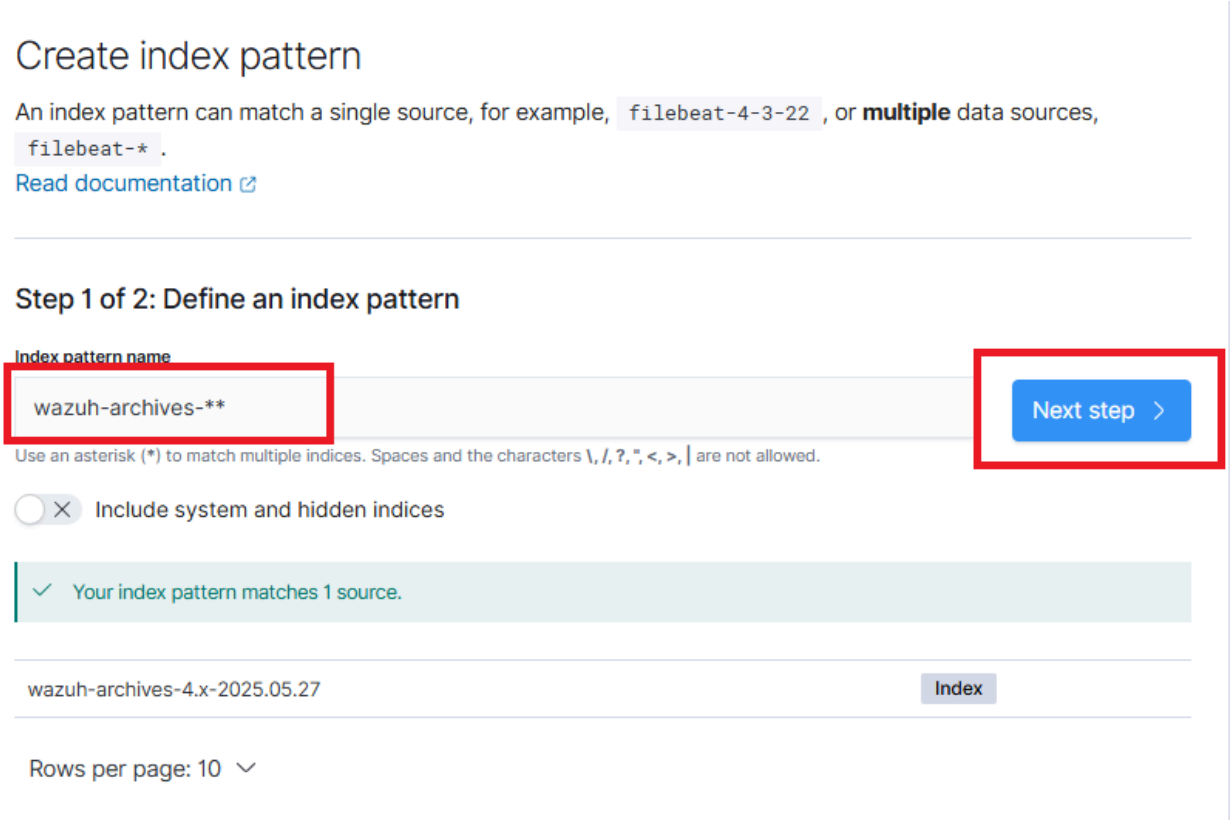
Systemctl restart filebeat

 # After that, go to the Wazuh dashboard, and click on the hamburger in the top left corner.

 # Select “Stack Management” near the bottom.



Then click on “**Index pattern**”, after that click on the “**create index**” button.
Name it “**wazuh-archives-***”, and then click on “Next Step”.



For “**Time field**” select “**timestamp**”. Then click on the “**Create Index Pattern**” button.

Click on the top left hamburger, then click on Discover.

You will see “**wazuh-alerts**” with a drop-down arrow. Click on it and select the index we just made.

This is where we will be keeping an eye out for our events

Section 6 - Wazuh Detection Rule Creation

We are going to create a custom rule to detect SSH brute-force attacks.

Let us navigate back to the Wazuh dashboard.

Click on the home icon in the top left ⟶ Click on the down Arrow to the right of "**Wazuh**" ⟶ Click on "**Management**" ⟶ "**Rules**" ⟶ "**Manage rule file**" ⟶ "**Custom Rules**" button on the right ⟶ Click on the pencil for Local_rules.

You should see the file shown below.

Subsection - SSH Bruteforce Rule

The rule there will be the one that inspires our own rule. This is the one I will be using.

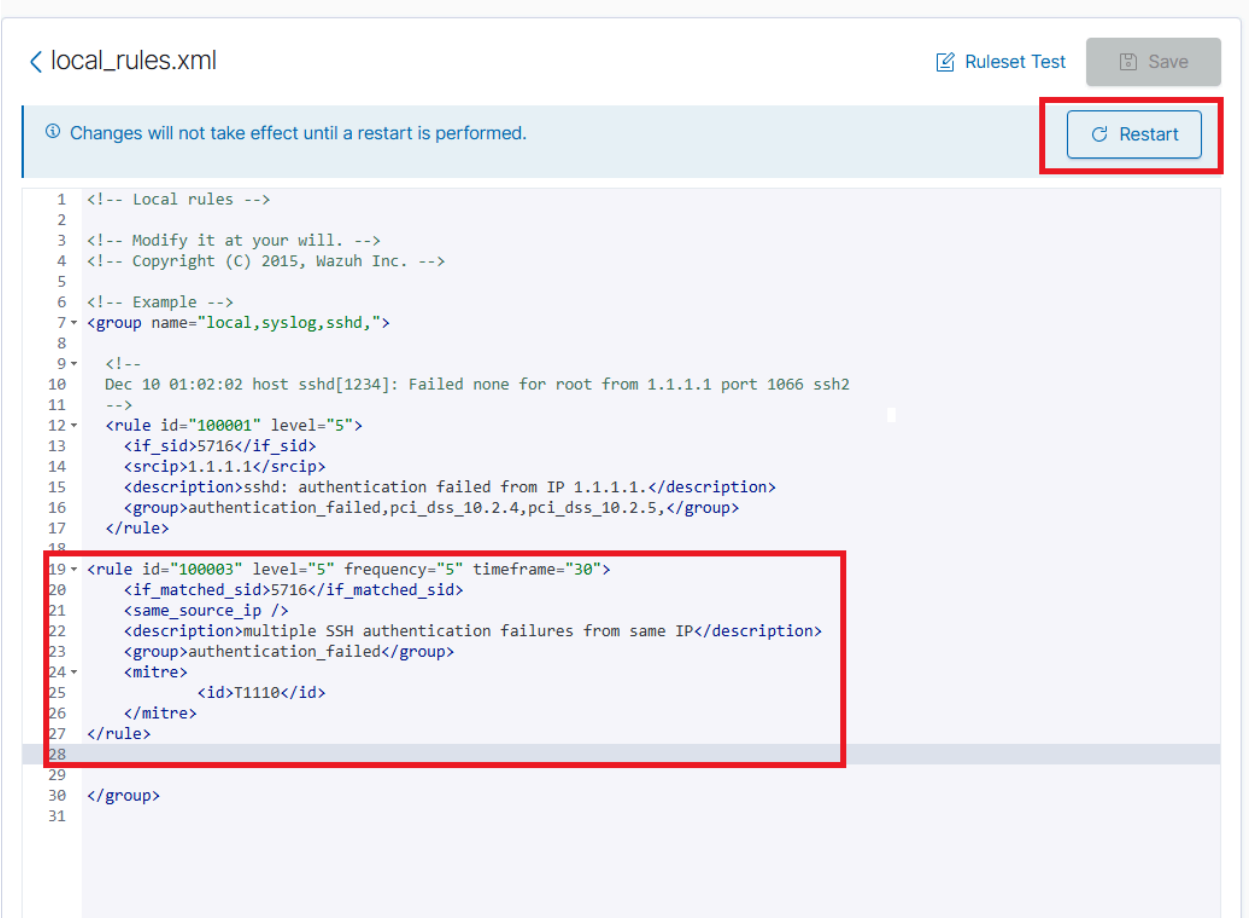
Template for SSH Bruteforce and secondary critical rule for successful logins
 # Be advised: If you copy and paste, the wrong variant of quotation marks may be used.
 # Swap them out in the XML file.

```
<rule id="100003" level="13" frequency="3" timeframe="30">
        <if_matched_sid>5710</if_matched_sid>
        <same_source_ip />
        <description>multiple SSH authentication failures from same IP</description>
        <group>authentication_failed</group>
        <mitre>
  <id>T1110</id>
        </mitre>
</rule>


<rule id="100005" level="15">
  <if_sid>5715</if_sid>
  <match>^Accepted|authenticated.$</match>
  <description>sshd: authentication success.</description>
  <group>authentication_success</group>
</rule>
```

####

Once done inputting the rule, click “**Save**” on the upper right.

After saving, it will want you to restart the manager.



Section 7 - MISP Threat Intelligence Platform Solution initial setup

It is time to launch MISP. Just like the other droplets…

In the main dashboard on DigitalOcean, click on "**Create**", then "**Droplets**."

Select the region closest to you. I selected "**New York**".

For the OS: Ubuntu, Version: 22.04 (LTS) x64

Droplet Type: Basic
CPU options: Premium Intel
$48 option for 8GB of memory

Authentication Method: Password (Input a strong password)

Hostname: misp

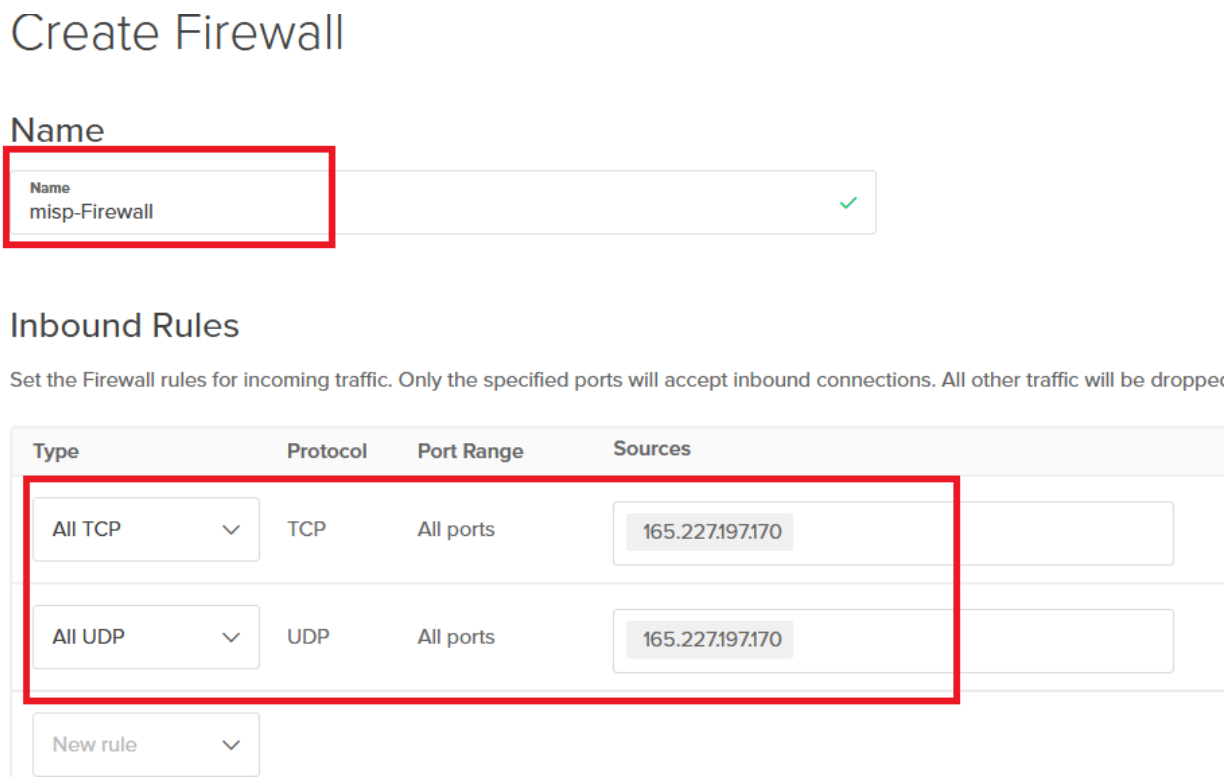
 # The link below is the official MISP GitHub in case you run into any issues I do not cover.
Helpful Link: Source: https://misp.github.io/MISP/xINSTALL.ubuntu2204

Once you have created the VM, click on "**misp**". On the left-hand side, click on "**networking**". Scroll down to the firewall section and click "**Edit**".

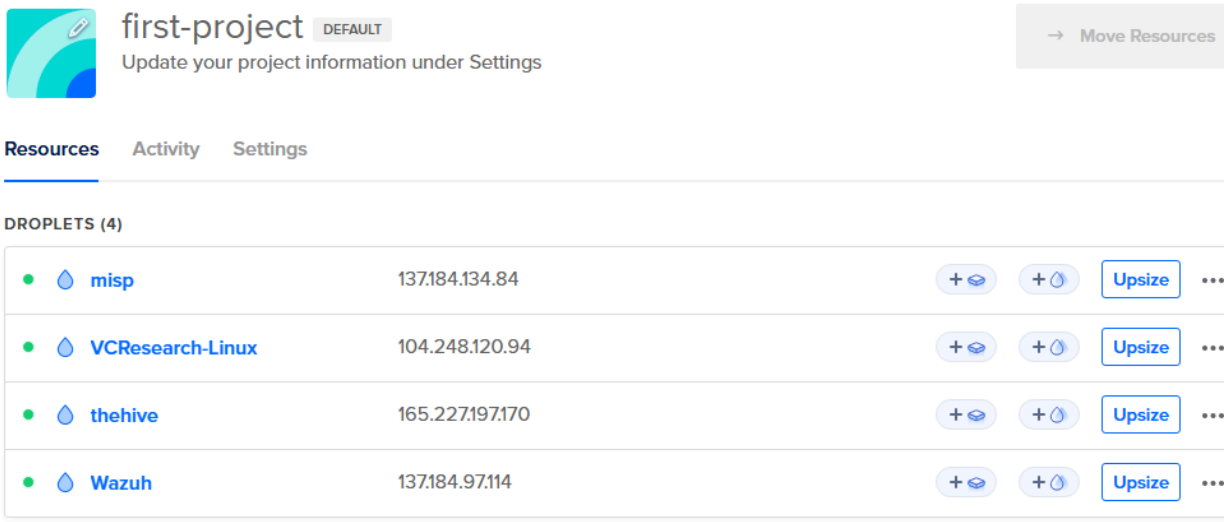
Create a New Firewall for MISP.

I named mine "**misp-Firewall**". For the inbound rules, allow all TCP and UDP from your IP address and TheHive's IP address.

Once done creating the rules, scroll down and apply the firewall to the "**misp**" droplet.



Once the MISP VM is running, connect to it via SSH as root.



 # Once connected, run the commands below.
 # These commands are to add the misp user, put them into the sudo group, then switch to that user.

adduser misp
Usermod -aG sudo misp
Su - misp

 # Once you are the misp user, do a …
Sudo apt-get update

 # The command below will install everything needed for misp, no sudo needed

wget --no-cache -O /tmp/INSTALL.sh
https://raw.githubusercontent.com/MISP/MISP/2.4/INSTALL/INSTALL.sh

 # Run without sudo (Takes a while)
bash /tmp/INSTALL.sh -A

 # Click "ENTER" on any red screens

 # If asked to input the baseURL, put the public IP of the MISP cloud
 # instance. Ex. "https://<InstanceIP>" then hit enter

 # In the case that it is needed,
 # Manual URL setup can be done here:
 # nano /var/www/MISP/app/Config/config.php
 # Change Base URL

 # After the “install.sh” script is done. Run the following commands
 # Activate an interactive sudo shell.

Sudo -i

 # The below command will activate the virtual environment
Source /var/www/MISP/venv/bin/activate

 # The command below will install all MISP modules
pip install misp-modules[all]

 # # The below is not needed but can be done to confirm that it is working.
 # Then you should be able to run “misp-modules” as a command

Subsection - Creating a persistent service

 # Run the below commands to get a service running
sudo vim /etc/systemd/system/misp-modules.service

##Add the below into the file
########################
 # Be advised: The single quotes around the source command may be the wrong variant.

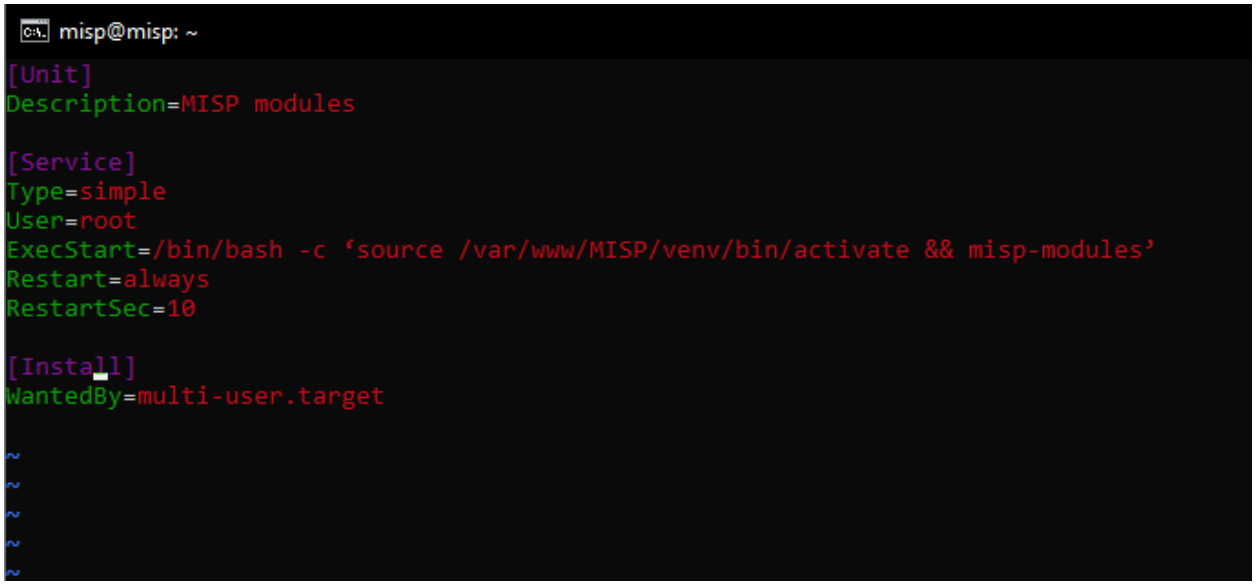
[Unit]
Description=MISP modules

[Service]
Type=simple
User=root
ExecStart=/bin/bash -c 'source /var/www/MISP/venv/bin/activate && misp-modules'
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target

########################

 # Once done, do a ":wq!" to save and exit Vim



 # Enable and start the service

sudo systemctl daemon-reexec
sudo systemctl enable misp-modules
sudo systemctl start misp-modules

 # Confirm

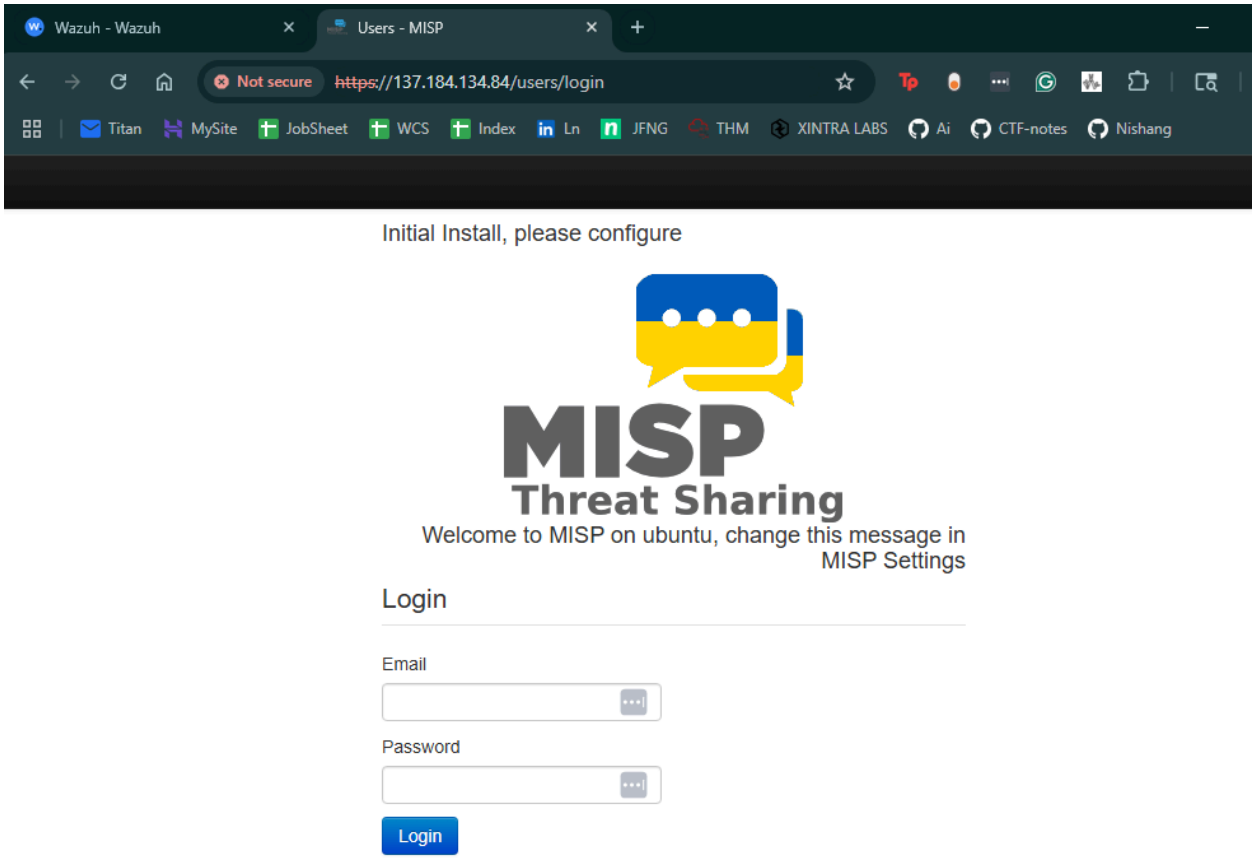
sudo systemctl status misp-modules

 # If the service shows that it is not running, try to exit out to the root user again and run the "reboot"
 # command. Log in again as the user misp and rerun the status command.

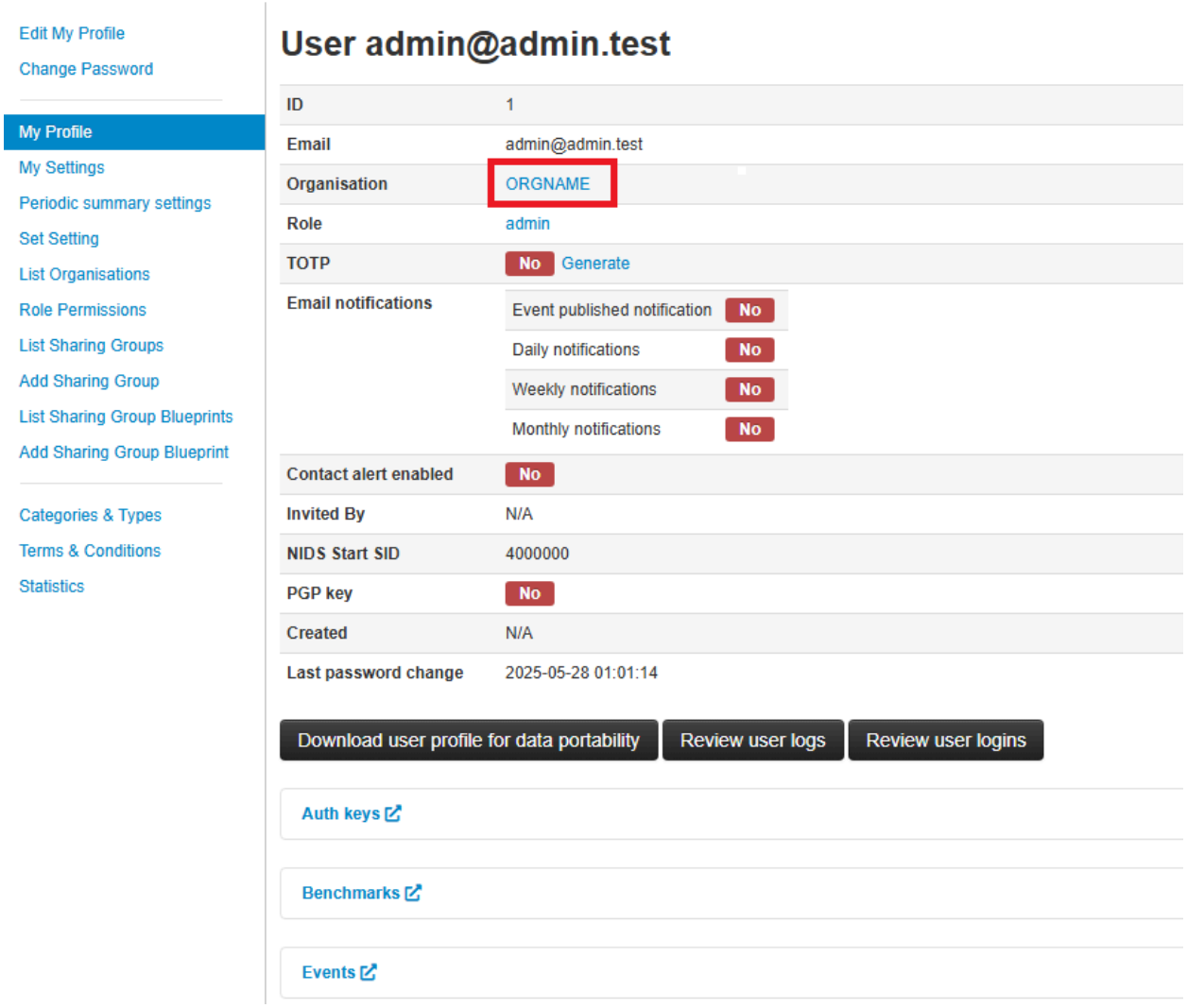
Subsection - MISP login, Organization and User Creation

 # Log in with the default Email/Pass and change it

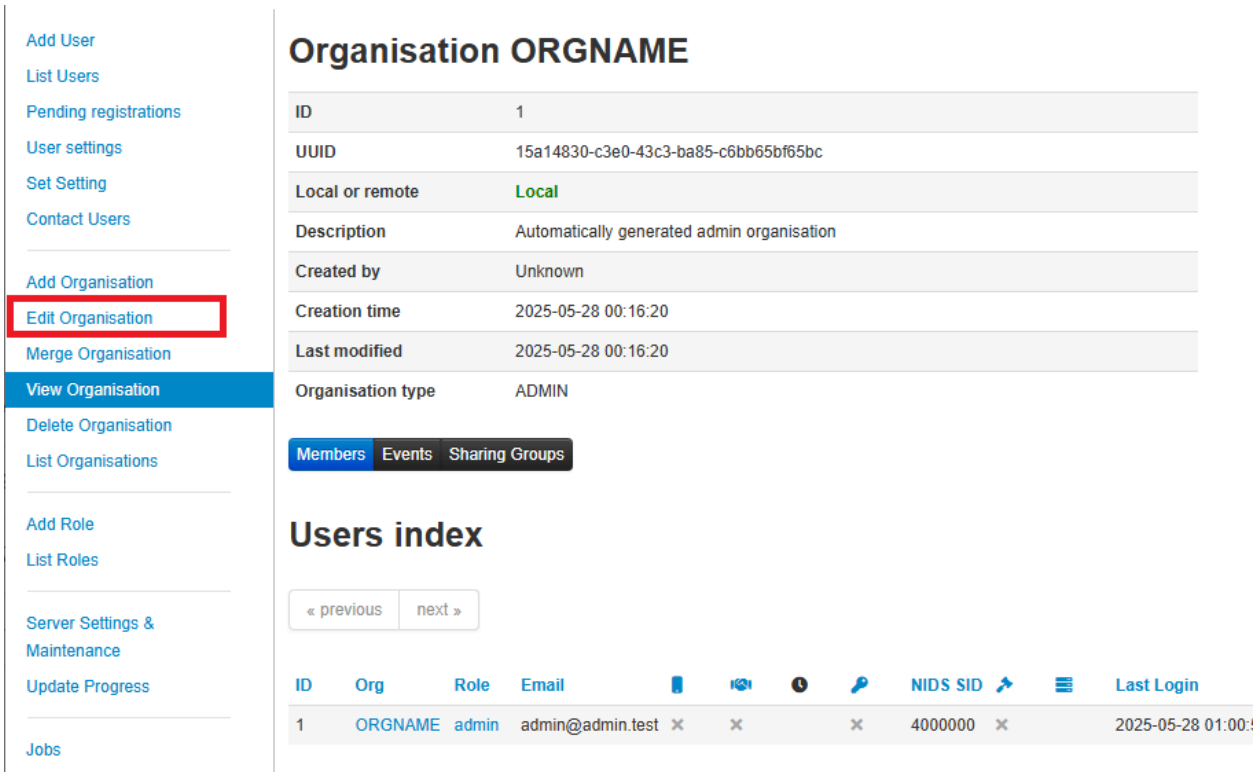
admin@admin.test
admin



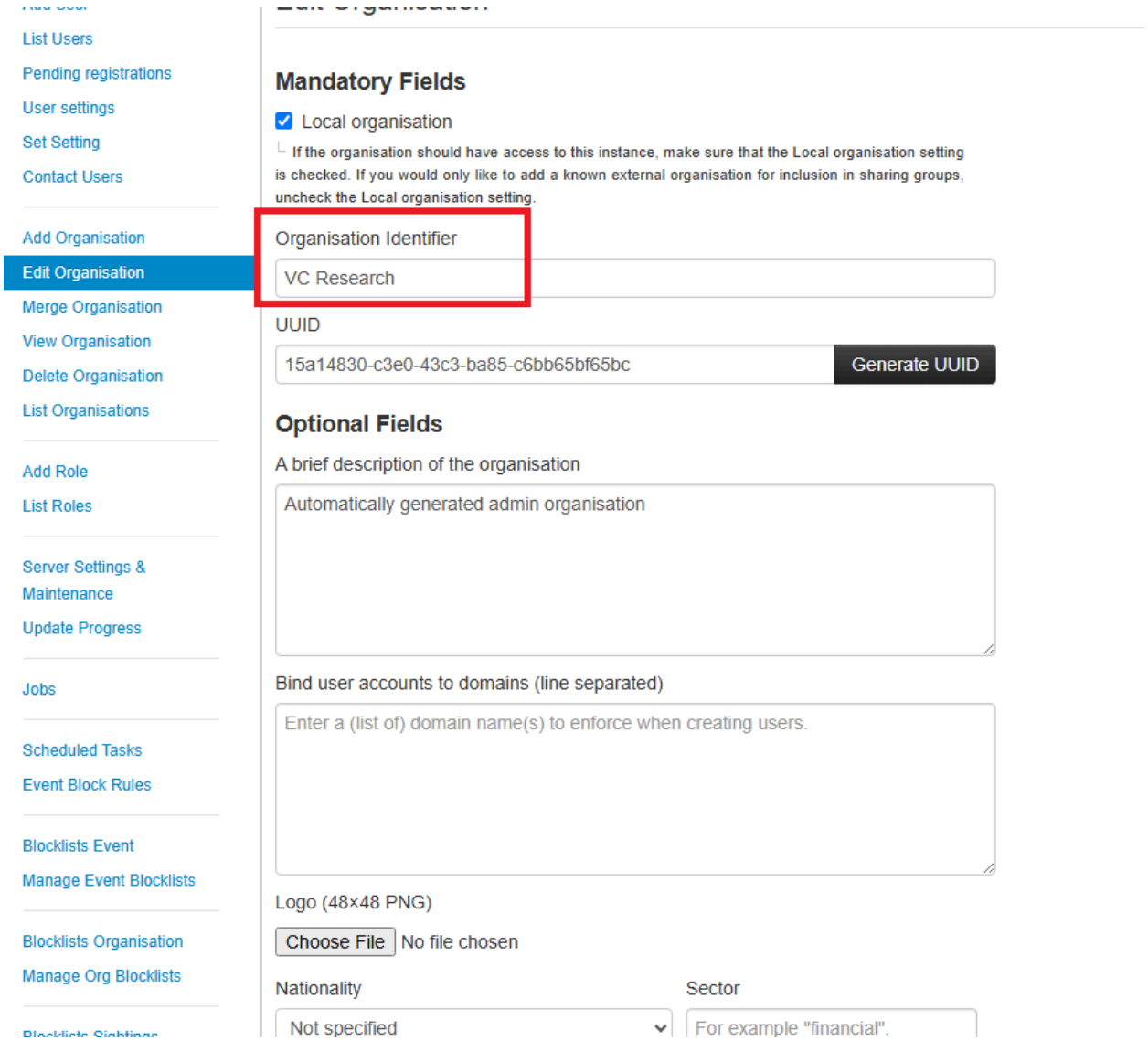
Let us edit the default organization name. Click on “**ORGNAME**”



Click on “**Edit Organization**”

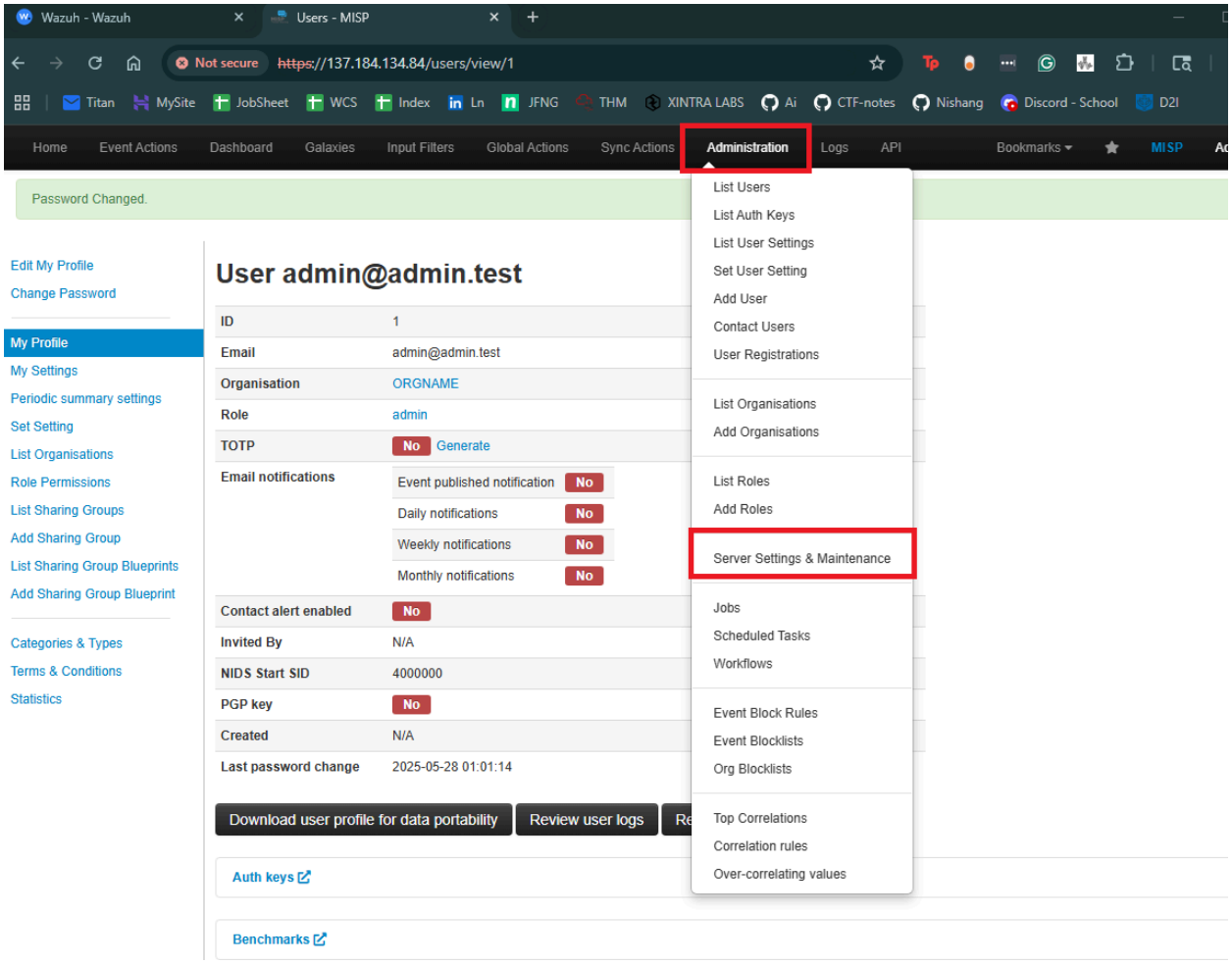


Change the name, I went with VC Research. Once done, scroll down and click on “**Submit**”.

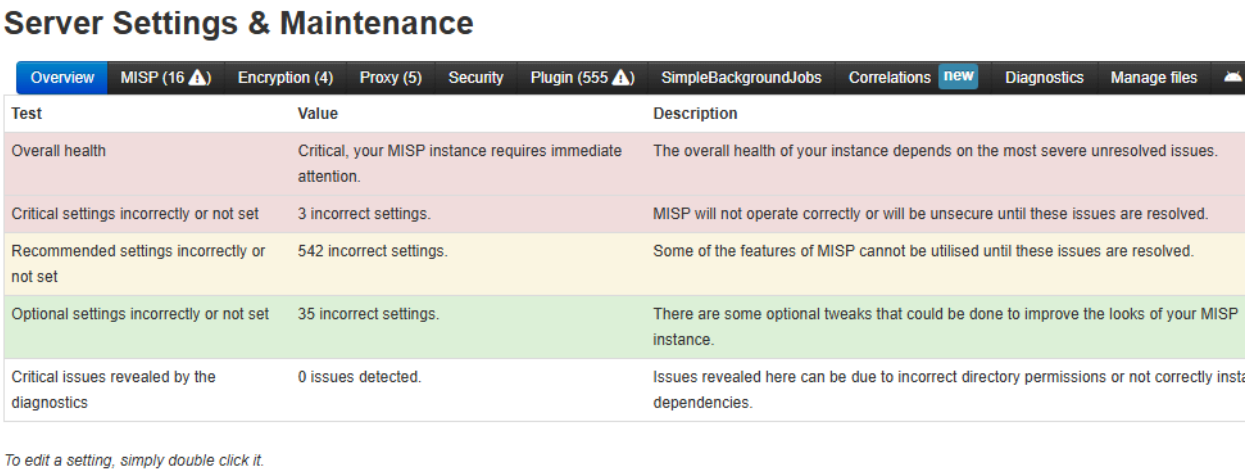


Subsection - MISP Server Settings & Maintenance

Navigate to **“Server Settings & Maintenance”**



This is where you can see the overall health of the instance, plugins, and issues that may show up. In a production environment, I highly recommend going through all of it. But as I am just using this to test certain things out, I will not be worrying much about it.



I recommend you go through the plugins to see all the enrichment modules available to use!

Since this project is based on taking in SSH brute-force IPs, in this project, I used the APIs mentioned below.

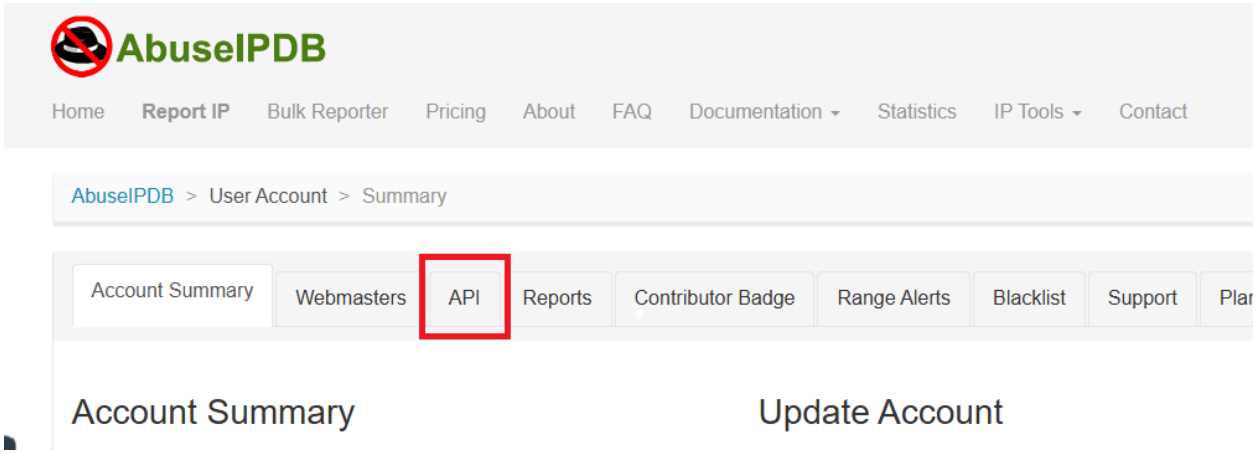
Some enrichment modules to consider. I used these 2 alongside the VirusTotal API.

 # Ip2location.io

Create an account with them to get a 7-day trial. Once you confirm your email and go past the quickstart, you'' be able to see your API key

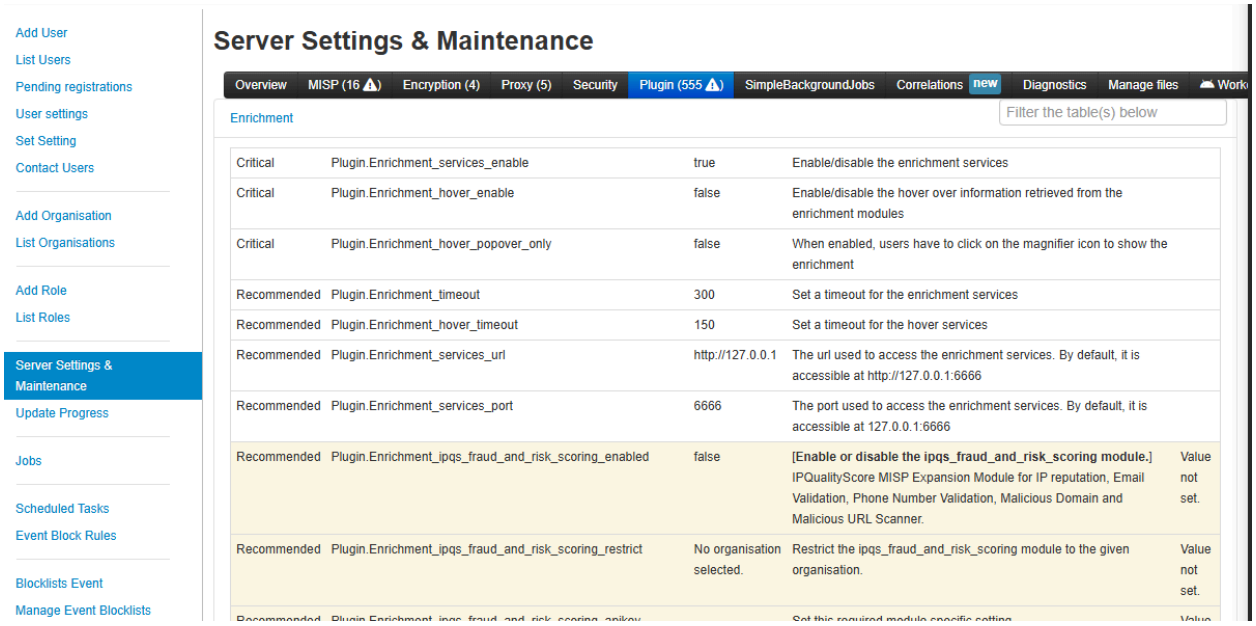
 # AbuseIPDB

Create an account and confirm your email. Then in the API tab, you will have an option to create an API key.



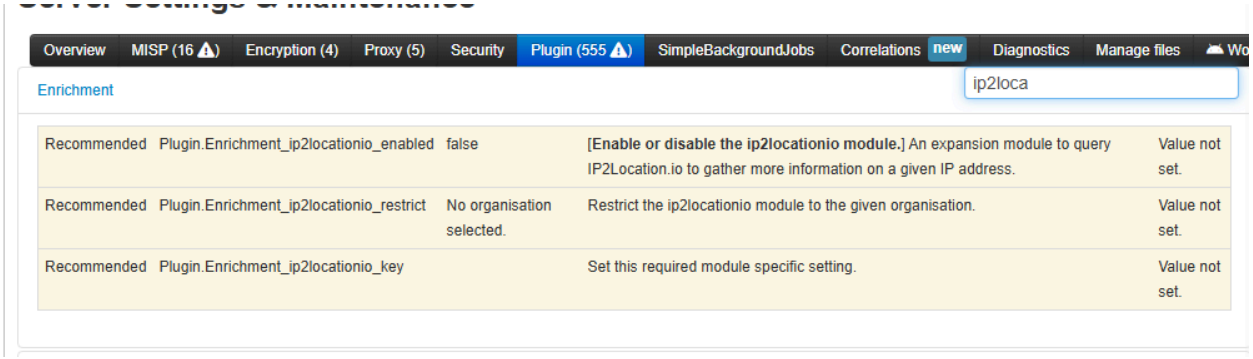
To enable the enrichment modules that we want, we need to go into the plugins tab.

In the top right, we can search for the plugins.



For Ip2location, we will need to enable the plug-in by switching it from false to true. We can leave the second line as it is. Then, we will enter our API key on the third line.

You can change the false by double-clicking on it and then clicking the checkmark.

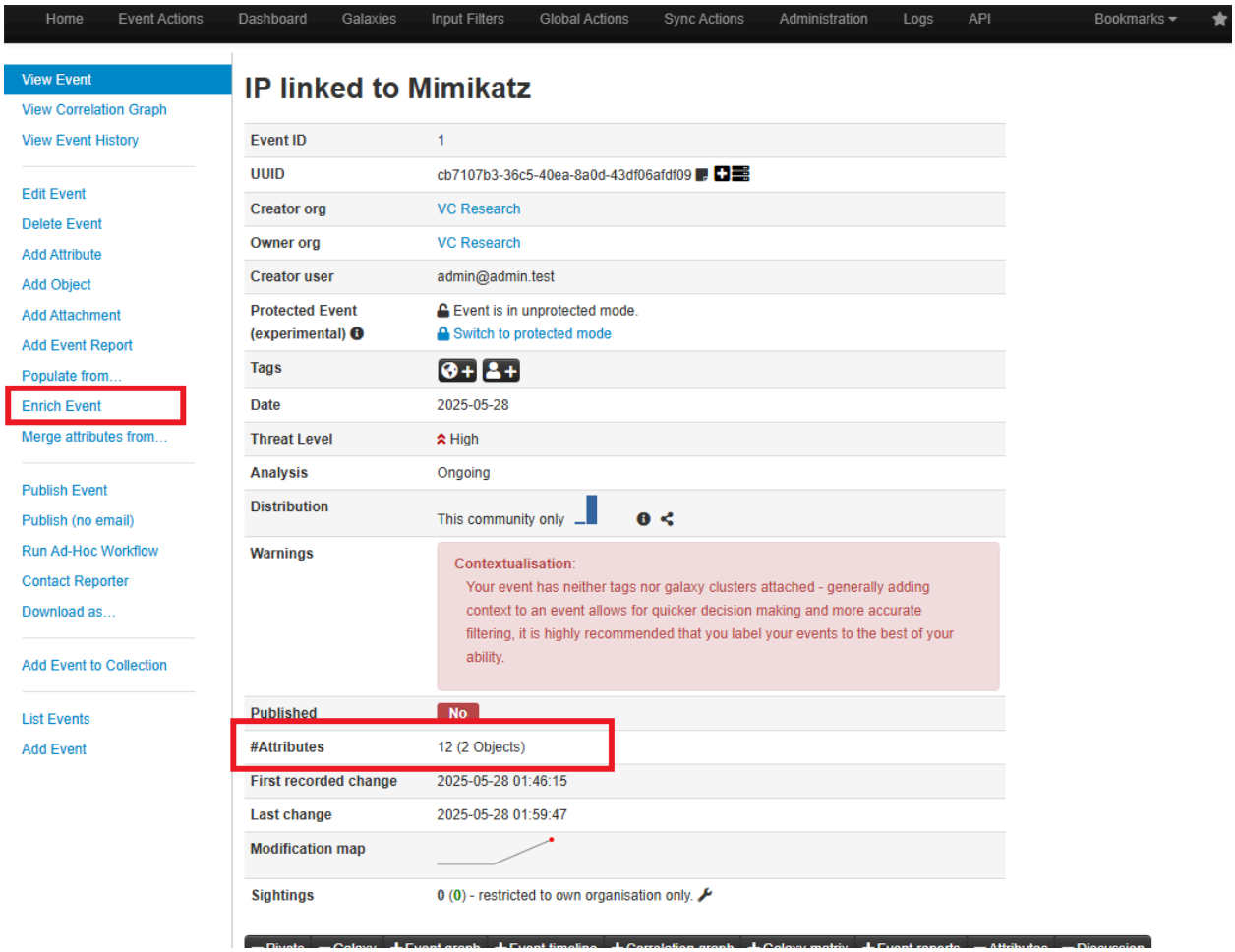


For AbuseIPDB, you will have to
1 Activate it
2. Input API key
3. Put a max age in days, I put 90
4. Abuse Threshold 60

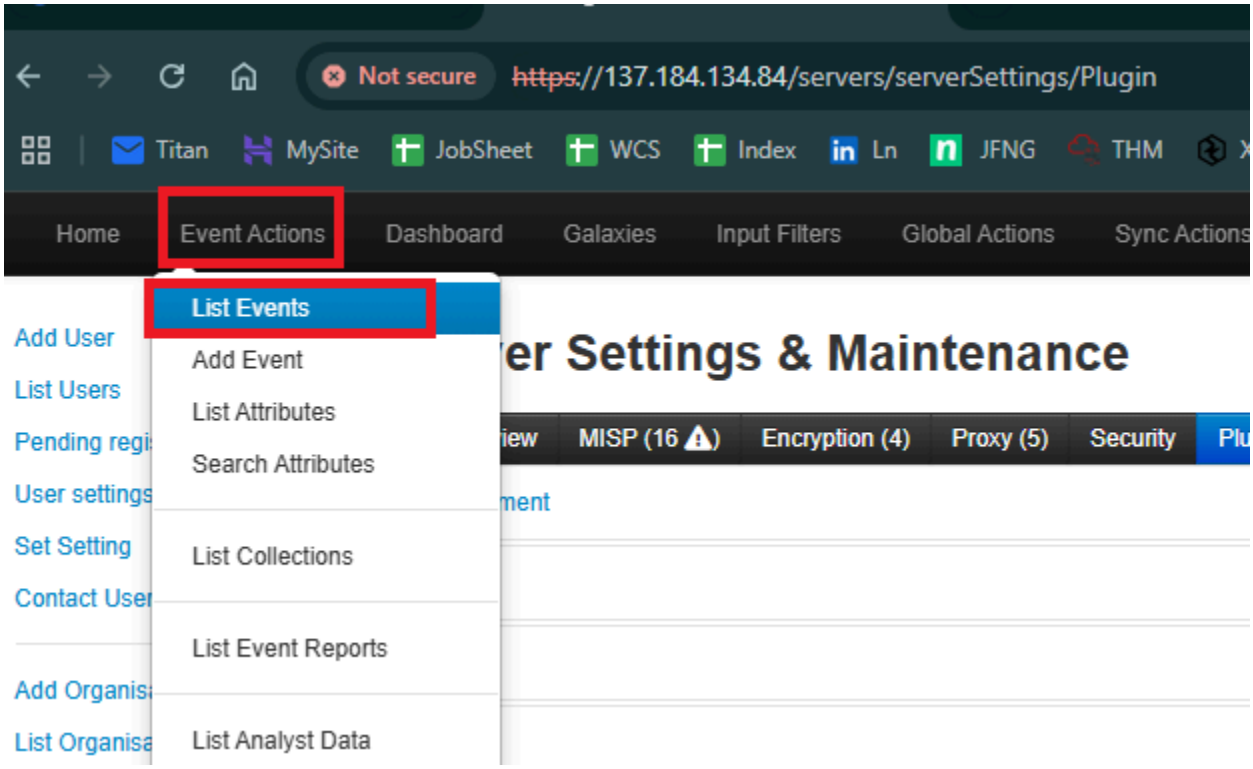
Subsection - Manual Enrichment

After putting in the two APIs, I went out and grabbed an IP that was linked to Mimikatz.

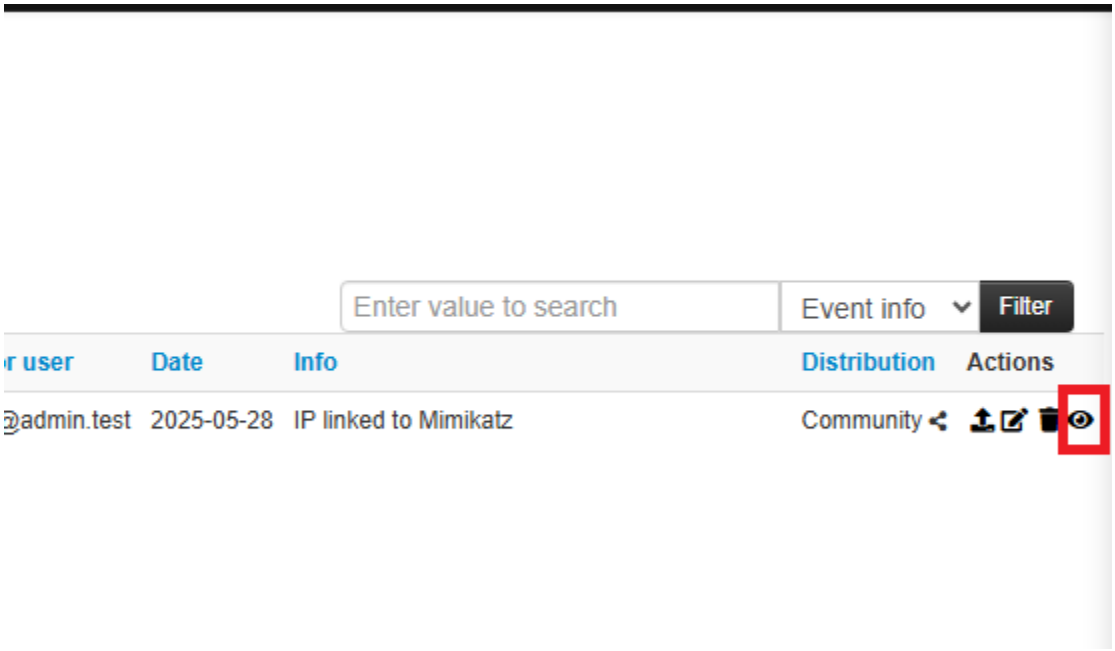
I created an event and added that IP as an attribute to it. Then clicked “Enrich Event”. AbuseIPDB and IP2Location then managed to get 11 other attributes connected to that IP.



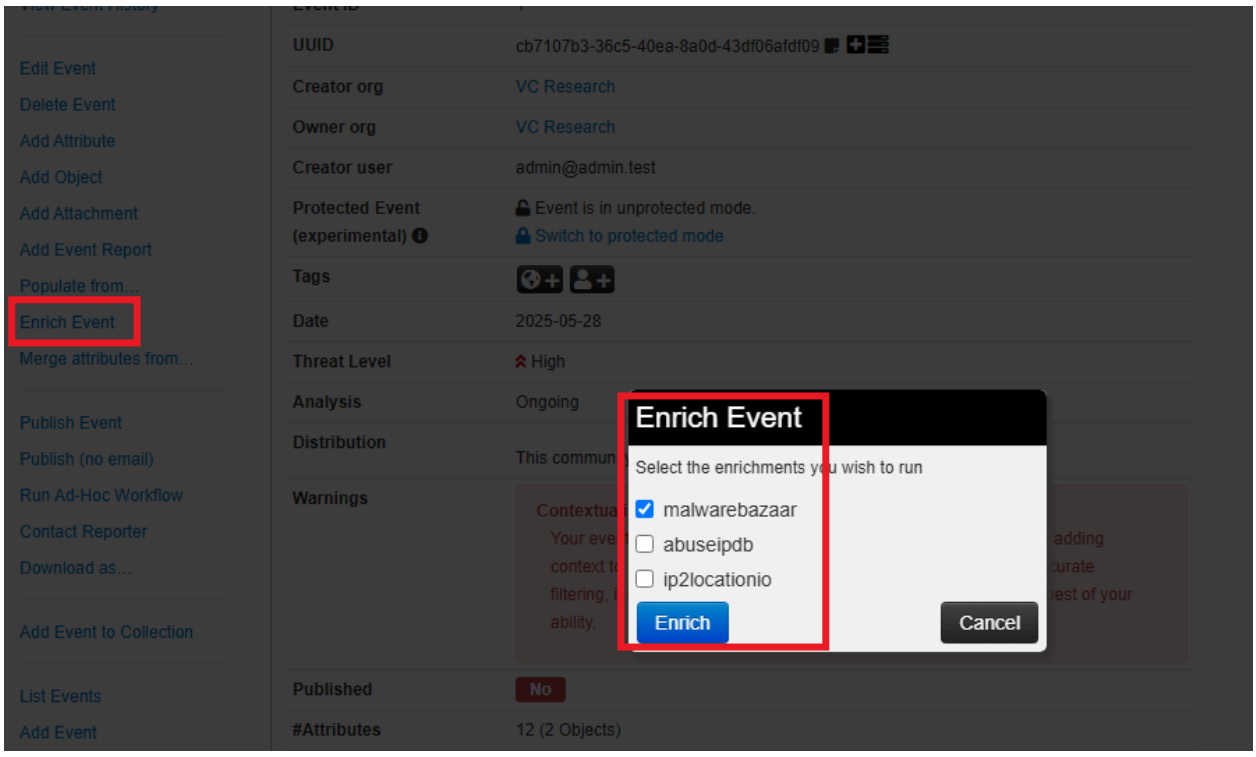
After this, I activated the MalwareBazzar plugin as it does not require an API.



On the far right of the event, you will see an eye that allows you to see the event.

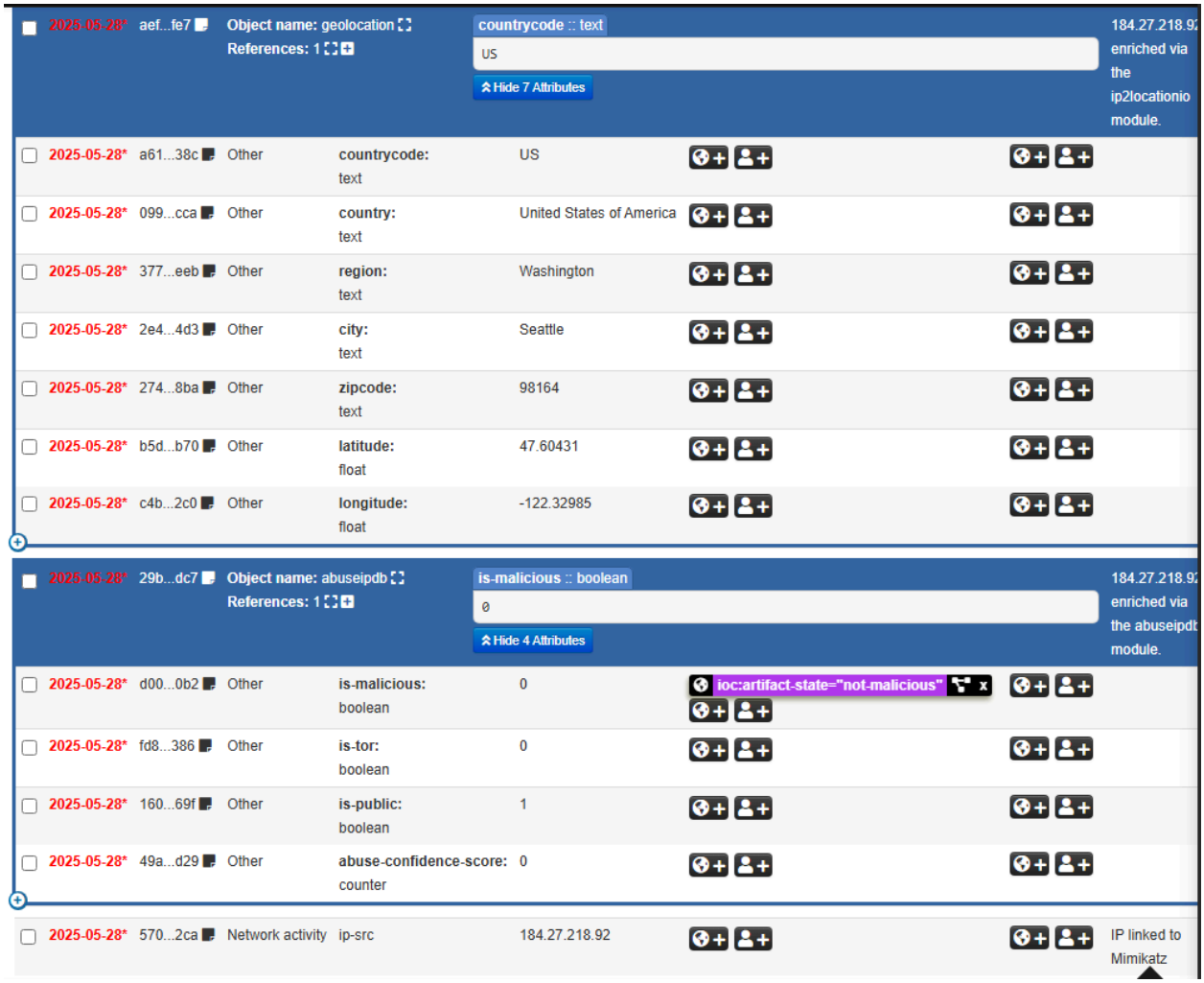


Click on Enrich Event and select the plugins you want to use.



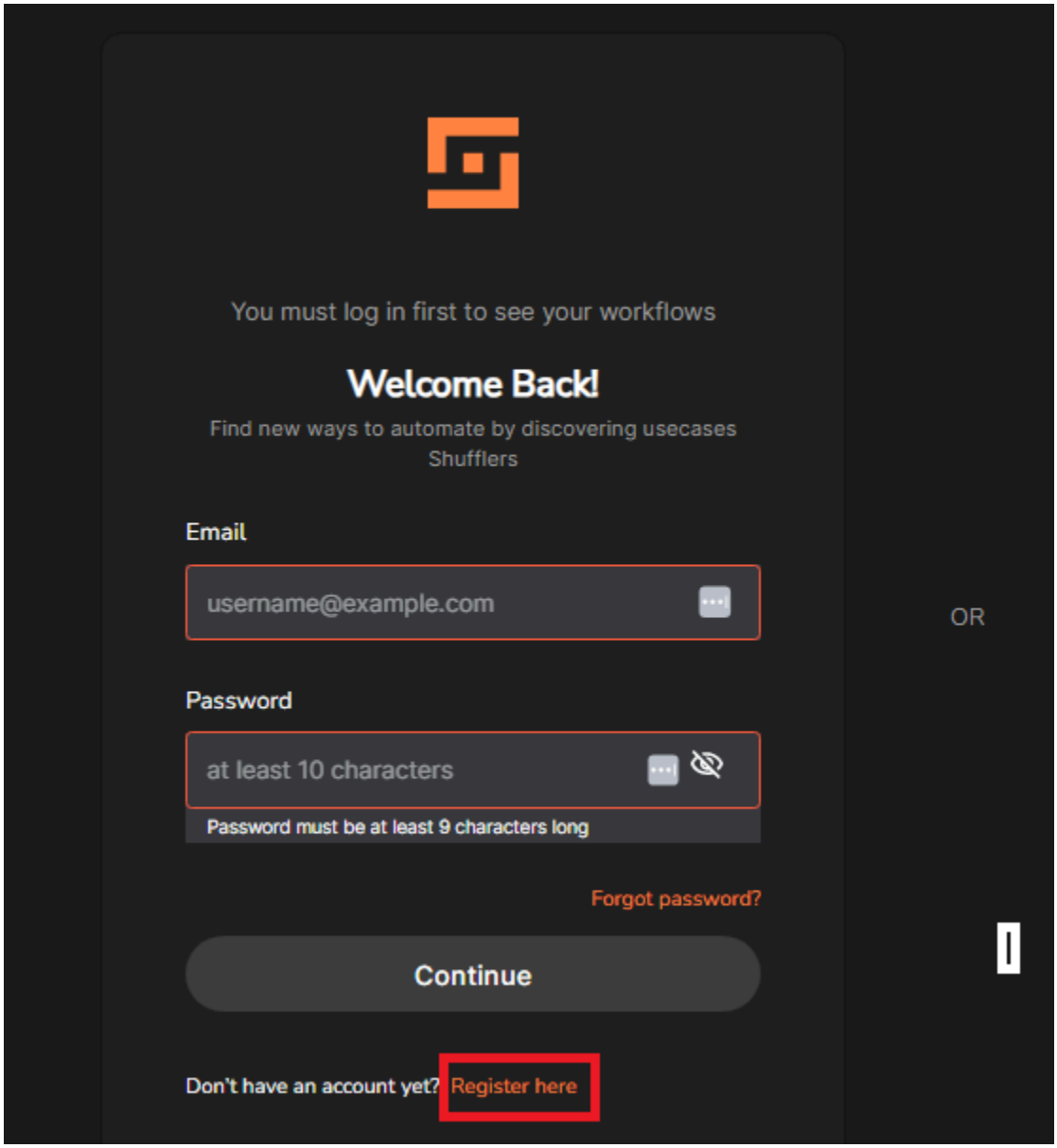
No new events were added, as this is usually used on malware hashes. But I just wanted to go through and show the process of enriching.

I suspect that if the VirusTotal API is used in MISP, it will help form those connections. But this should show the importance and crazy potential that MISP offers.

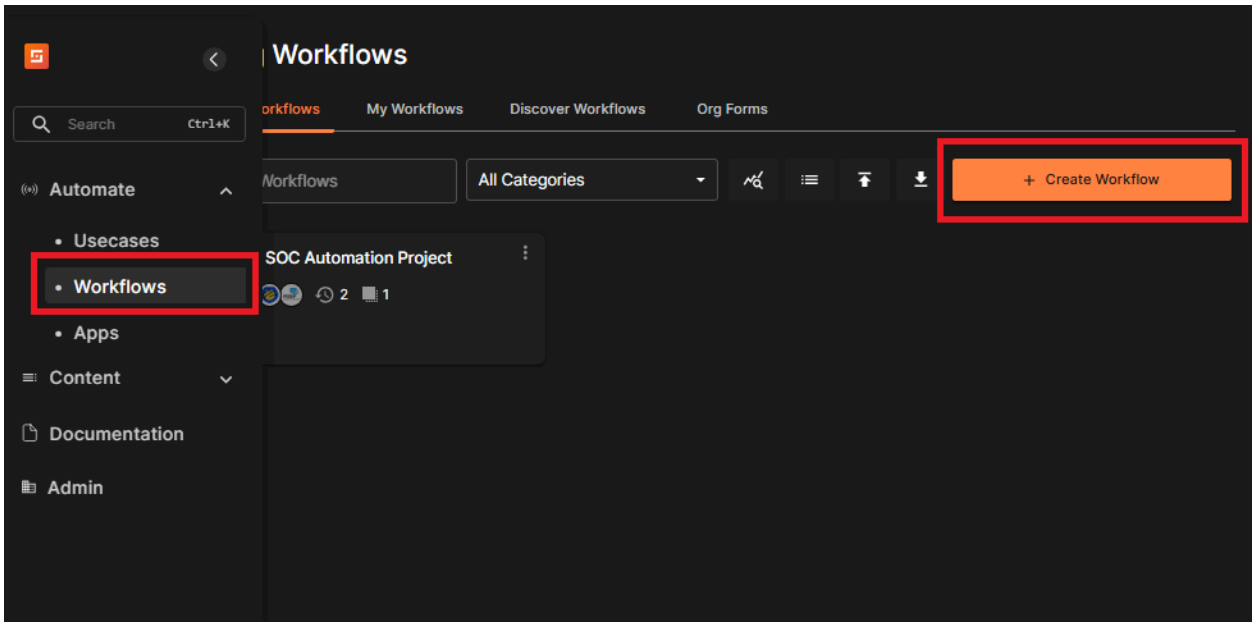


Section 8 - Shuffle SOAR solution deployment

For our SOAR solution, we will be using shuffle. Go on over to their website and make an account.



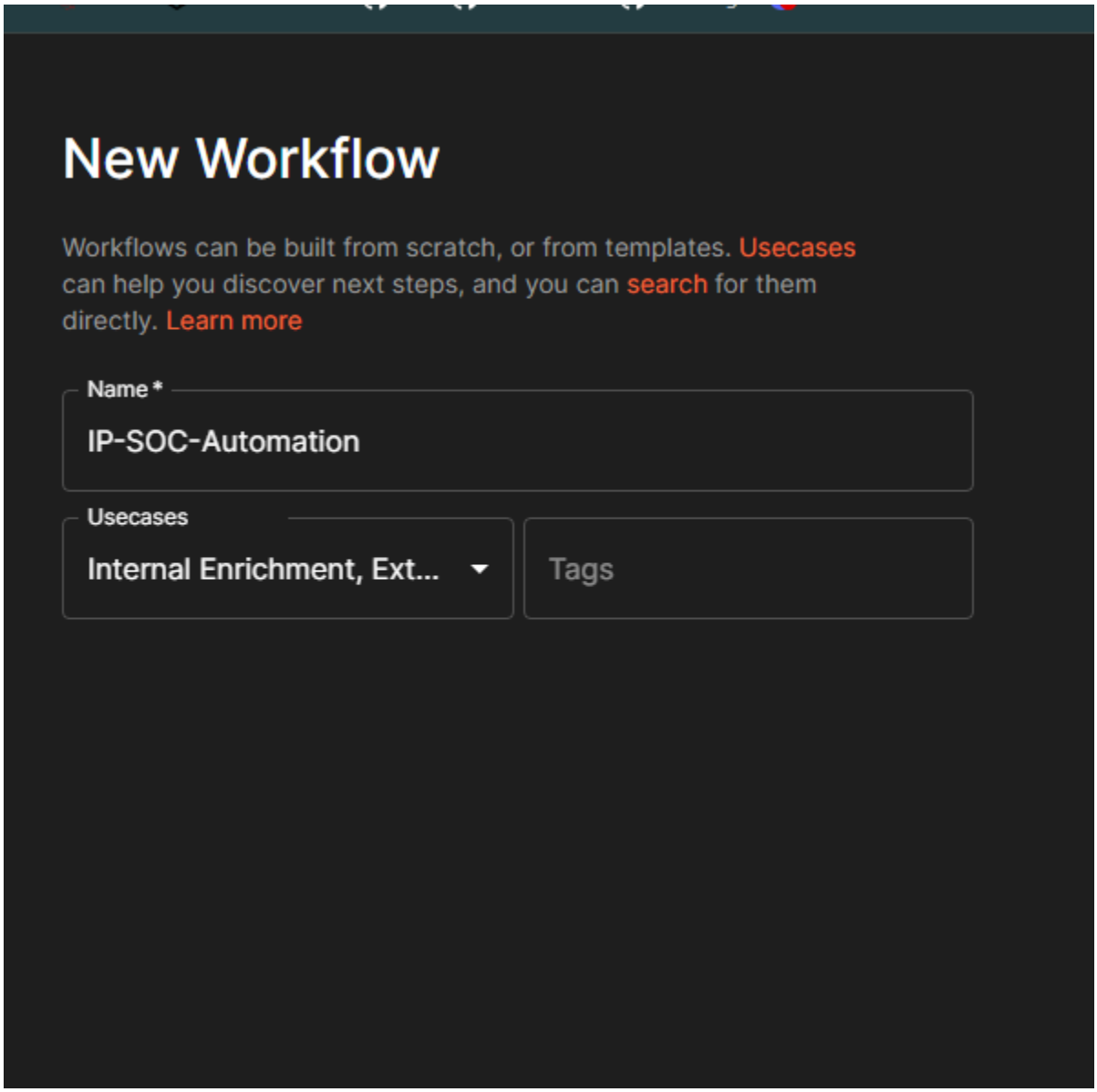
Once your account is all set, navigate your way to your workflows and click on the "**Create Workflow**" button.



You can name it however you want; I chose “**IP-SOC-Automation**”.

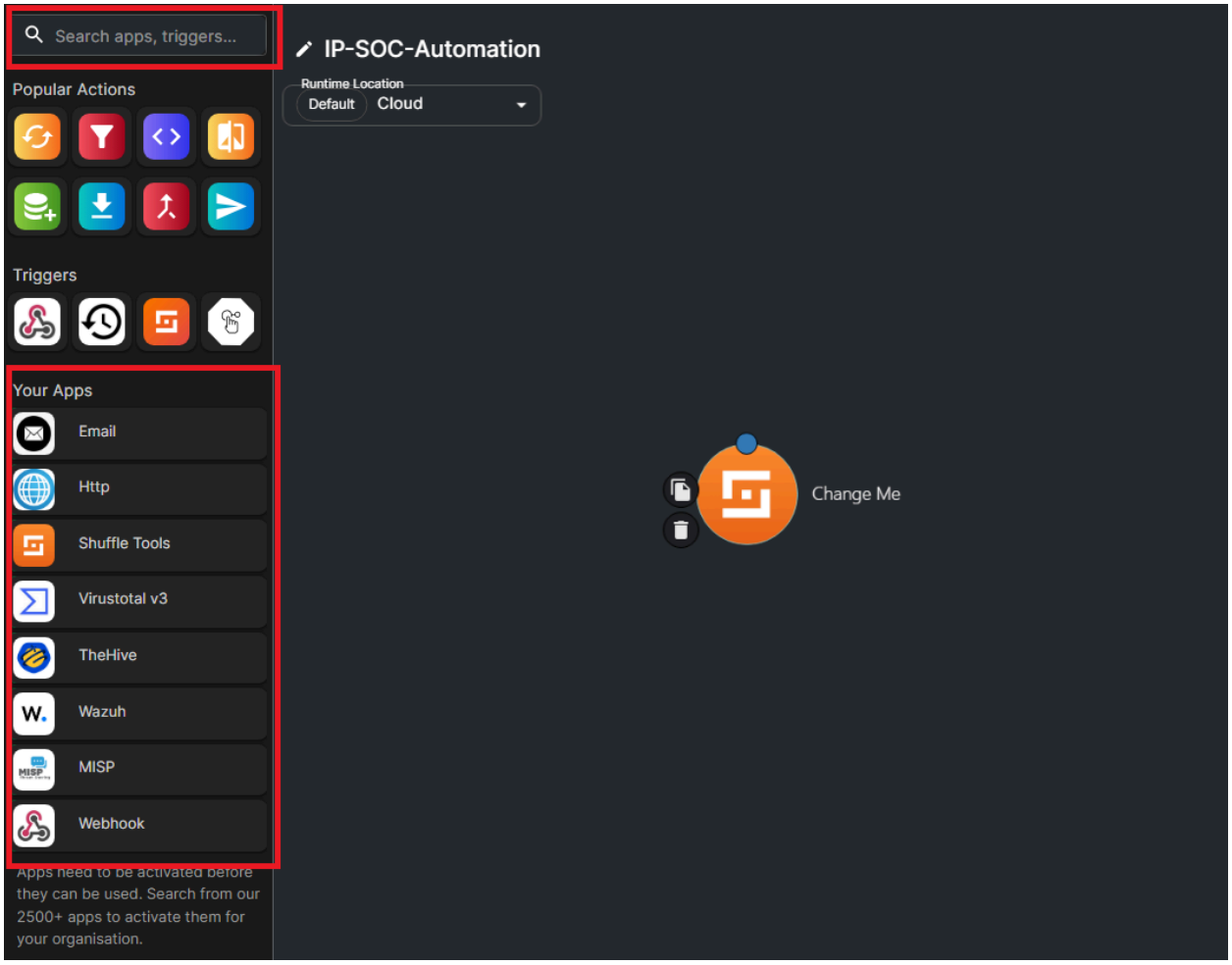
As for the use cases, I selected the following down below. After that, we can select “**Save changes**.”

- EDR to ticket
- Internal Enrichment
- External historical Enrichment
- Honeypot access
- Block an IP



After saving changes, we will be introduced to our workspace as shown below.

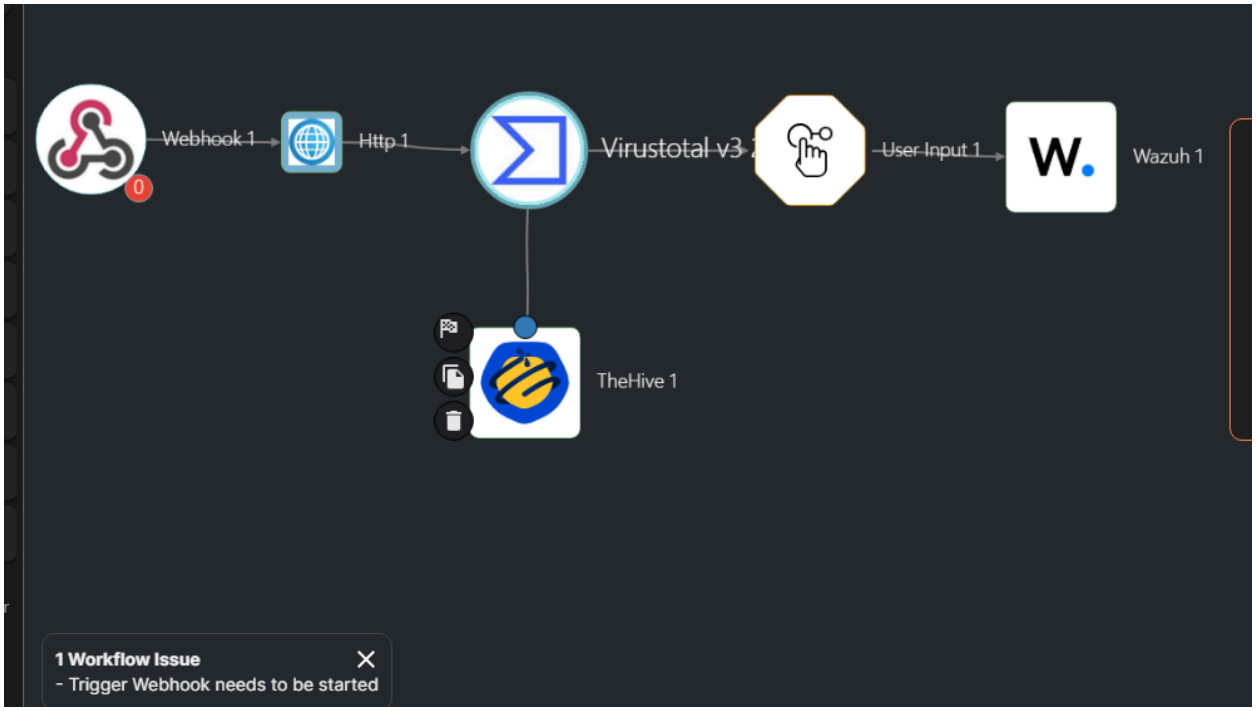
The left-hand side will have the widgets that we will be using. I have run through this project before, so you may not have some of the apps there, but you can search for them in the top left search bar.



Below is what our workflow will be.

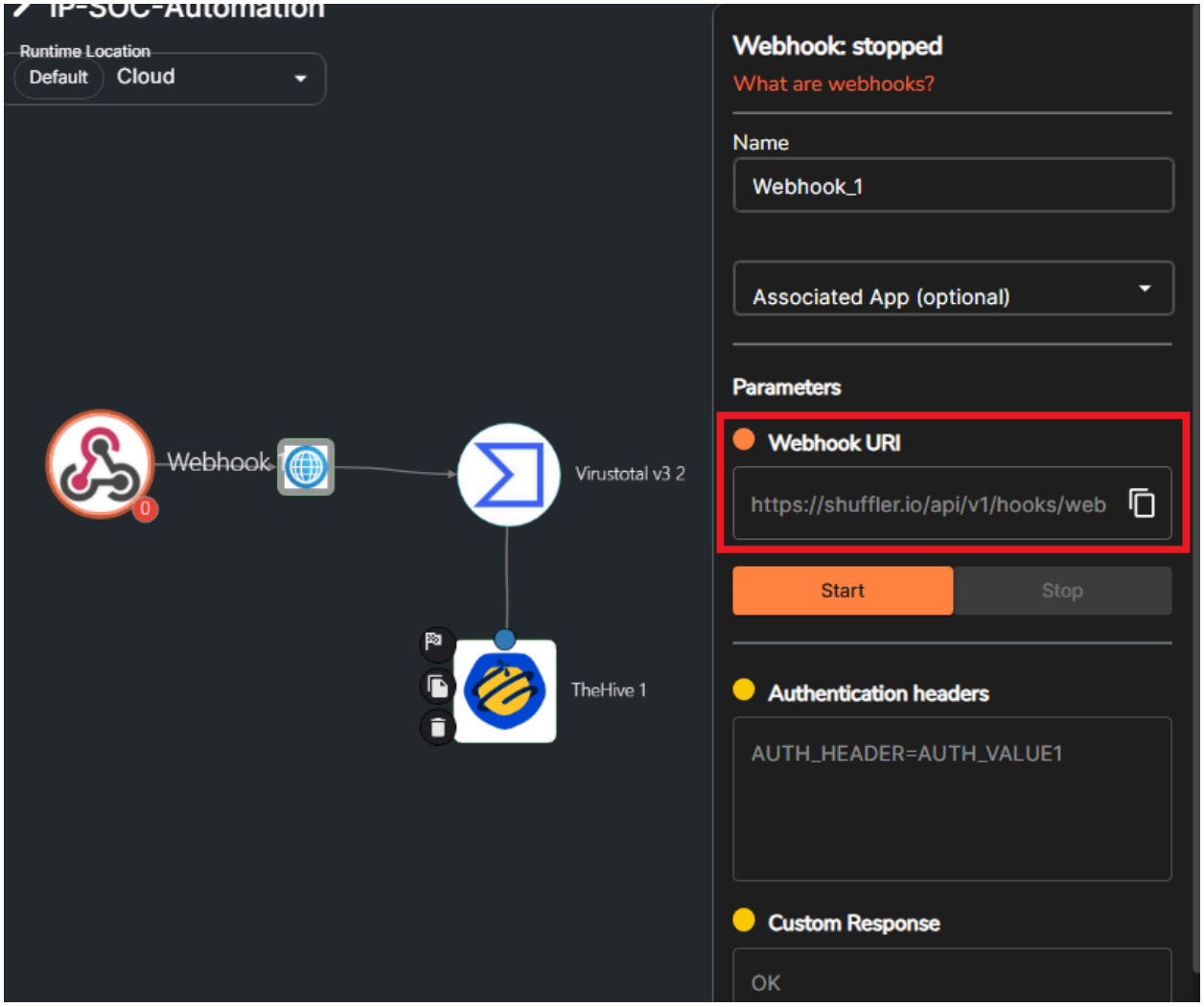
Webhook (Activates when Rule is Triggered) ⟶ HTTP (Gets Wazuh API) ⟶ VirusTotal (Enriches gathered IP) ⟶ Makes a case in TheHive AND sends an email to the Analyst for response approval ⟶ Response on Wazuh

The following sections will explain how to set up each widget.



Subsection - Webhook

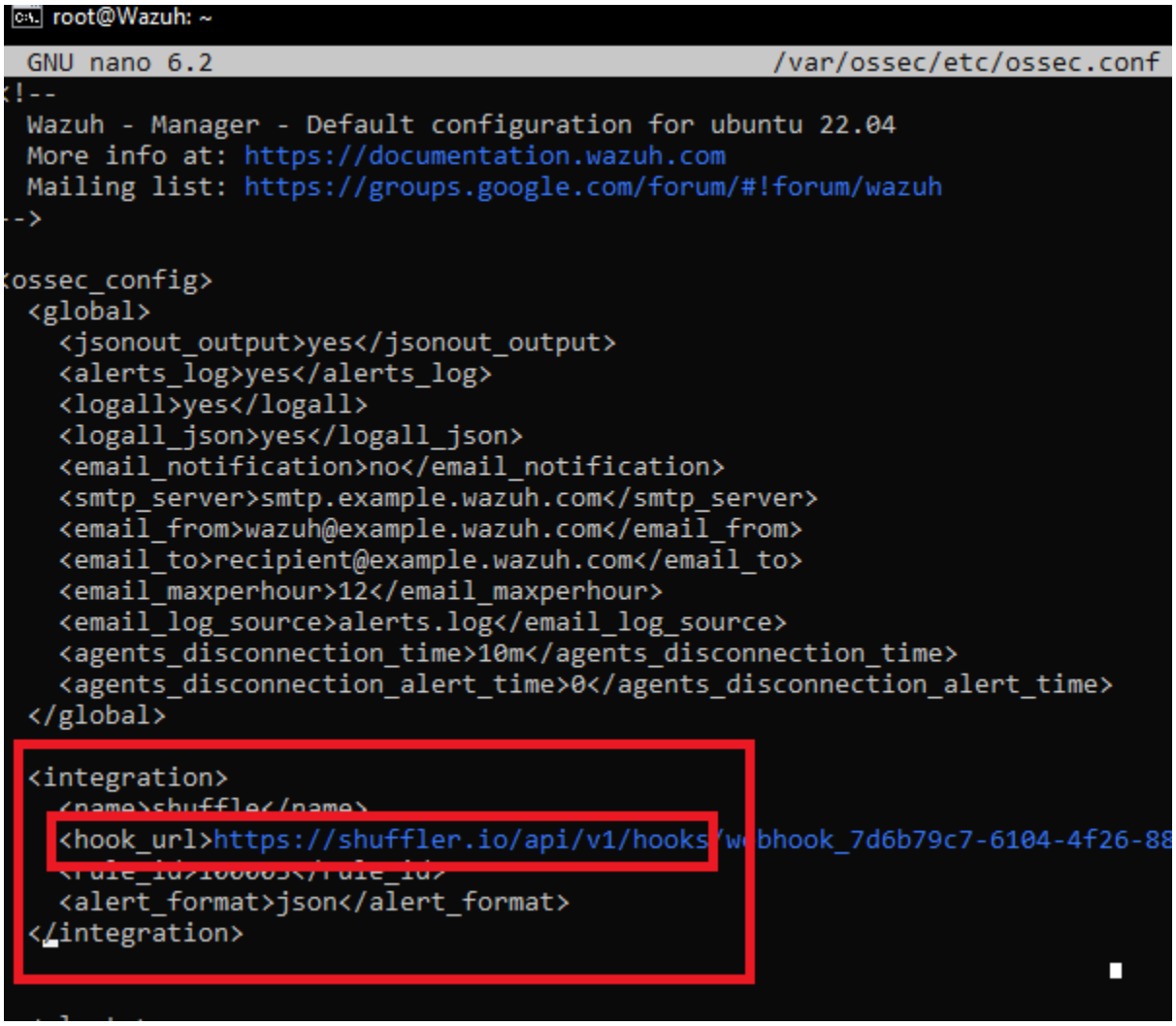
Clicking on the Webhook Widget will display a parameter that provides the Webhook URI, which we need to place in the Wazuh server. Click on it to copy it to your clipboard.



 # On the Wazuh server, nano into the ossec.conf file
nano /var/ossec/etc/ossec.conf

 # Scroll down a bit, and right below the first "Global" block, add in the code below. Make sure the
 # block aligns with the other blocks' spacing. Input your Webhook URI in the hook_url section.

```
  <integration>
   <name>shuffle</name>
    <hook_url></hook_url>
    <rule_id>100003</rule_id>
    <alert_format>json</alert_format>
  </integration>
```



 # ctrl+x and Y to save, and ENTER to exit

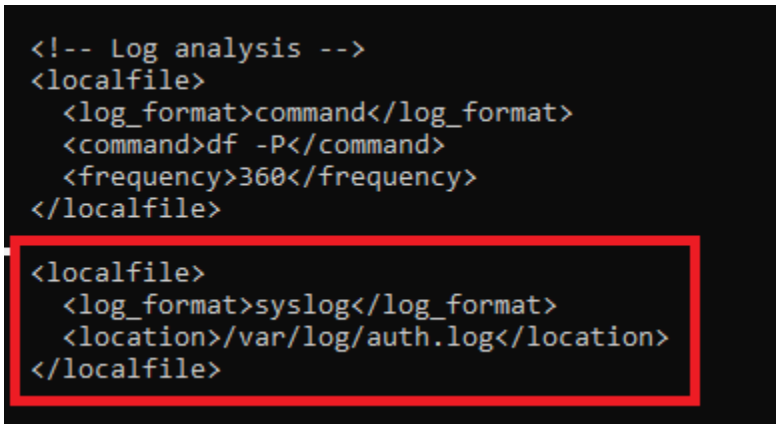
Systemctl restart wazuh-manager.service
Systemctl status wazuh-manager.service

On the Wazuh Agent / Linux Honeypot, we need to add a log analysis code block

 # On the Agent, nano into the ossec.conf file
nano /var/ossec/etc/ossec.conf

 # CTRL+W and search for "Log analysis", add the code. Make sure the spacing matches the other code.

<localfile>
 <log_format>syslog</log_format>
 <location>/var/log/auth.log</location>
</localfile>



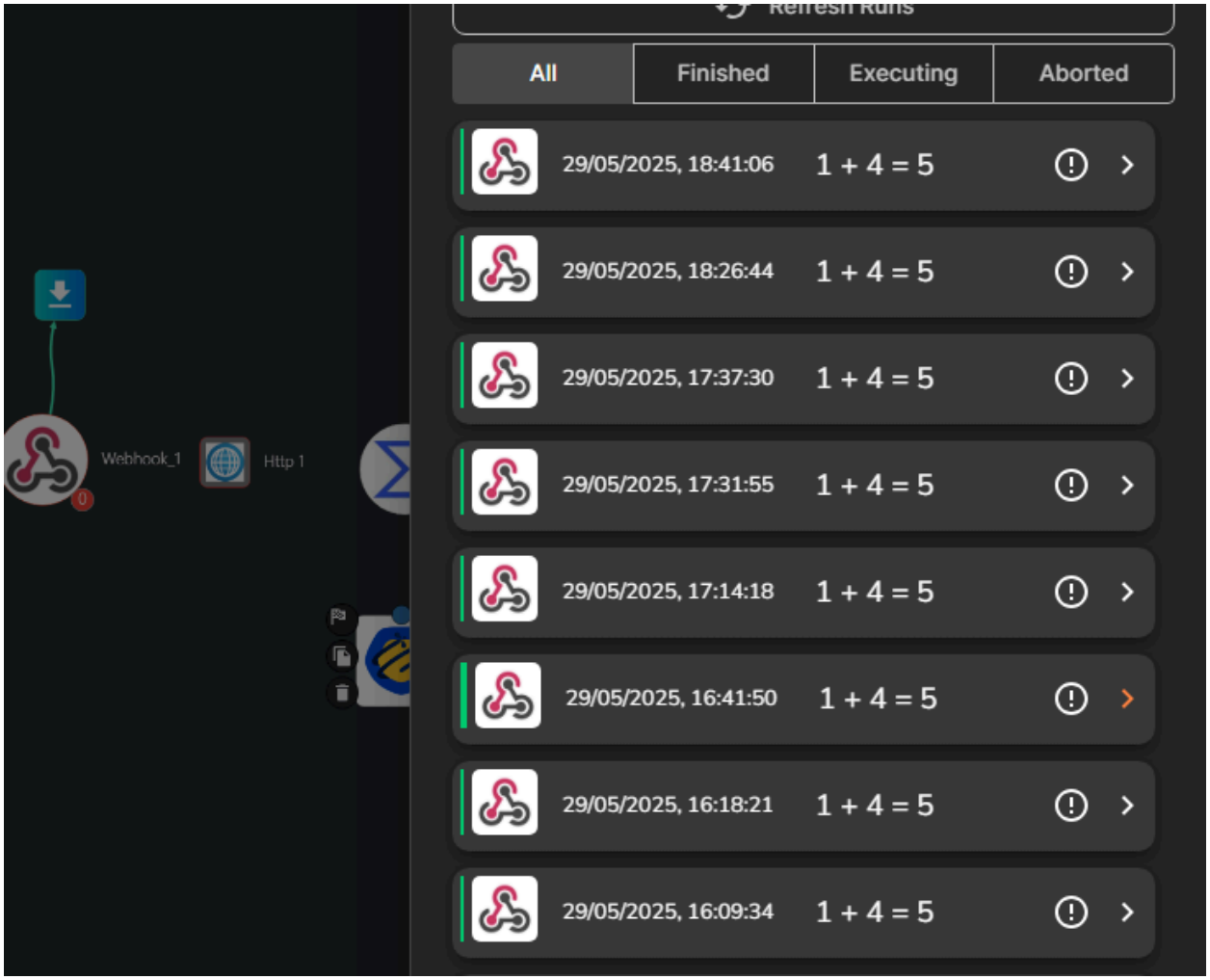
 # ctrl+x and Y to save, and ENTER to exit

Systemctl restart wazuh-agent
Systemctl status wazuh-agent

We can test this by triggering an SSH alert (we can lower the threshold or remove it)

Before you try and fail a SSH login, make sure you click on the Webhook widget, and click on “start”. After that, try and trigger the rule.

You know it was successful if it shows up in the “Explore Runs” section.



Subsection - HTTP widget

 # On the Wazuh Firewall, make a rule to allow all inbound traffic on port 55000

 # On DigitalOcean, click on your Wazuh Droplet, on the left-hand side, click “networking”,
 # Scroll down to the Firewall section and click on the Firewall name.

 # For the inbound rule, allow all TCP on port 55000 from any IPv4. Remember to save.

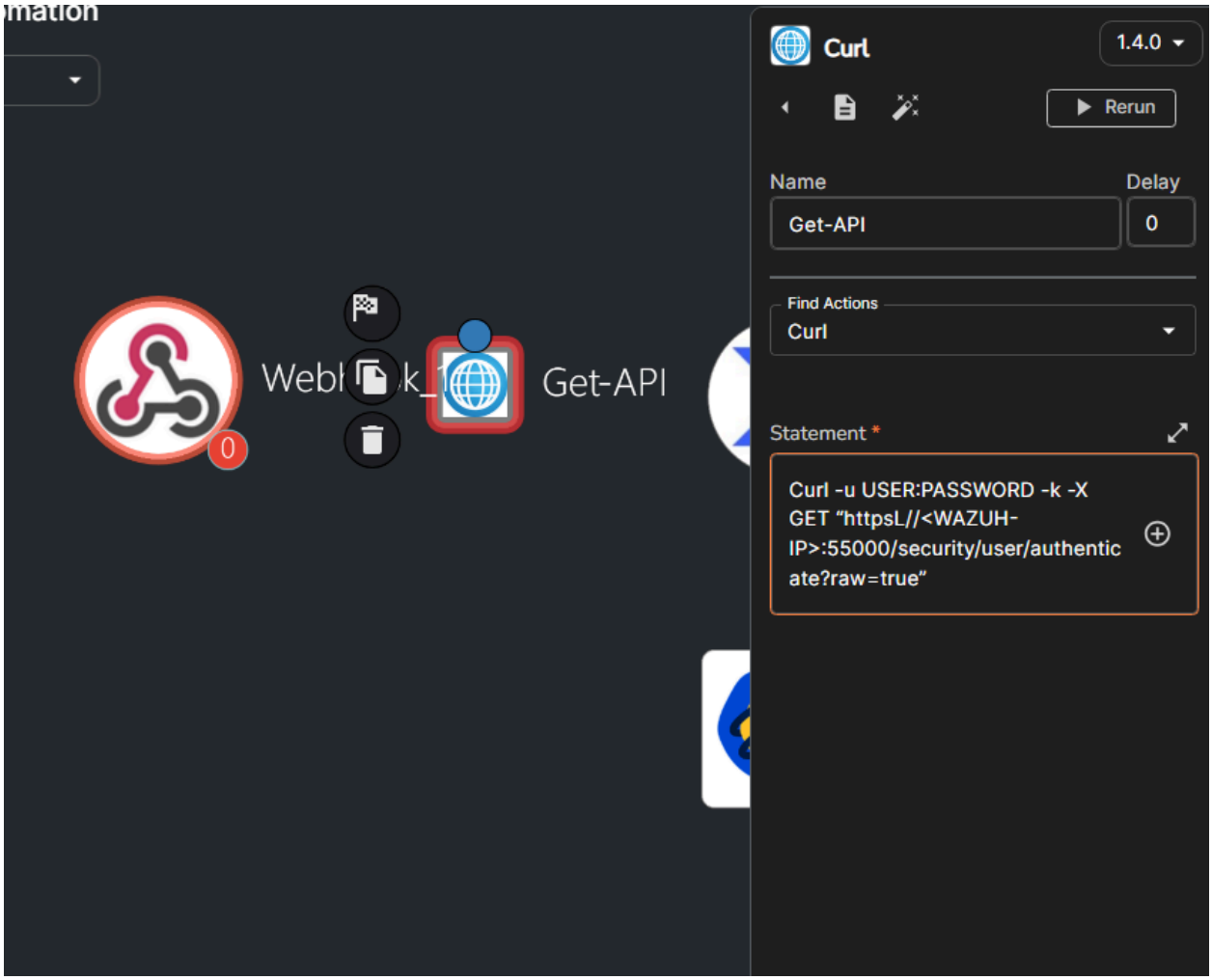
 # Back on Shuffle, change the fields below for the HTTP widget

Rename to “Get-API”

Find action: Curl

 # Replace the User and Password with the “Wazuh API” user credentials. As well as swapping the
 # localhost with the Wazuh public IP
 # The next page will explain on how to get the API credentials.

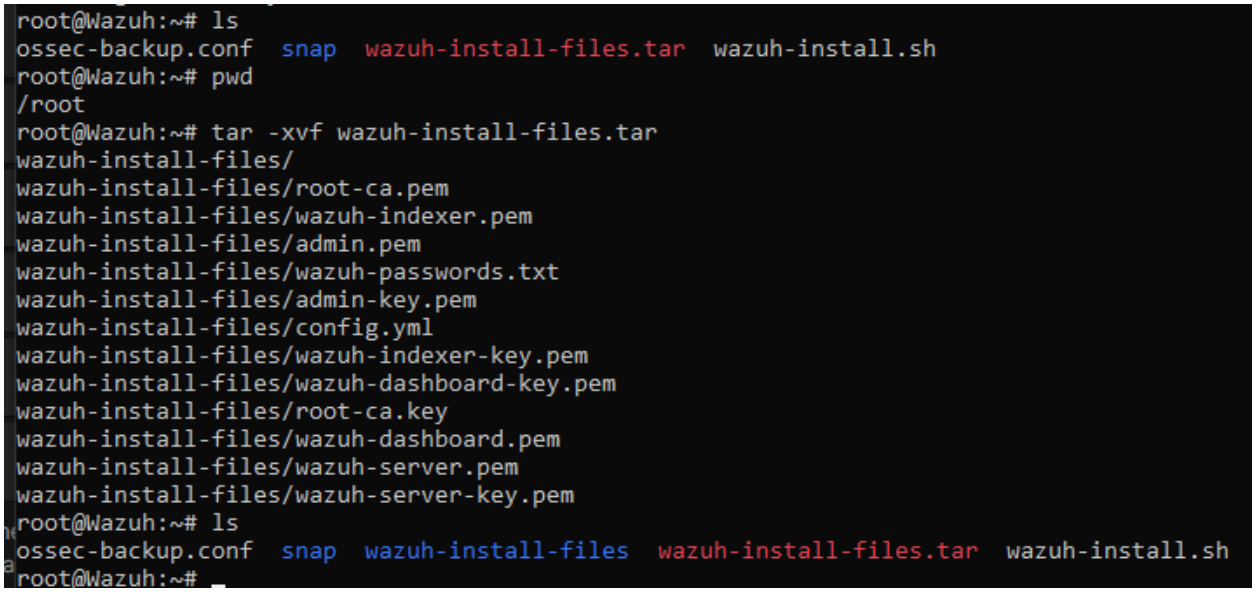
Statement:
curl -u <user>:<password> -k -X GET "https://localhost:55000/security/user/authenticate?raw=true"



 # To extract the API user credentials, go to the directory where you downloaded Wazuh. And run…

tar -xvf wazuh-install-files.tar

 # The output should be similar to the picture below.



Go into the "**wazuh-install-files**" directory and cat out the "**wazuh-passwords.txt**".

The credentials we need are the "wazuh API user". Use those to fill in the curl command on shuffle.

Friendly reminder to save your shuffle workflow at this point.

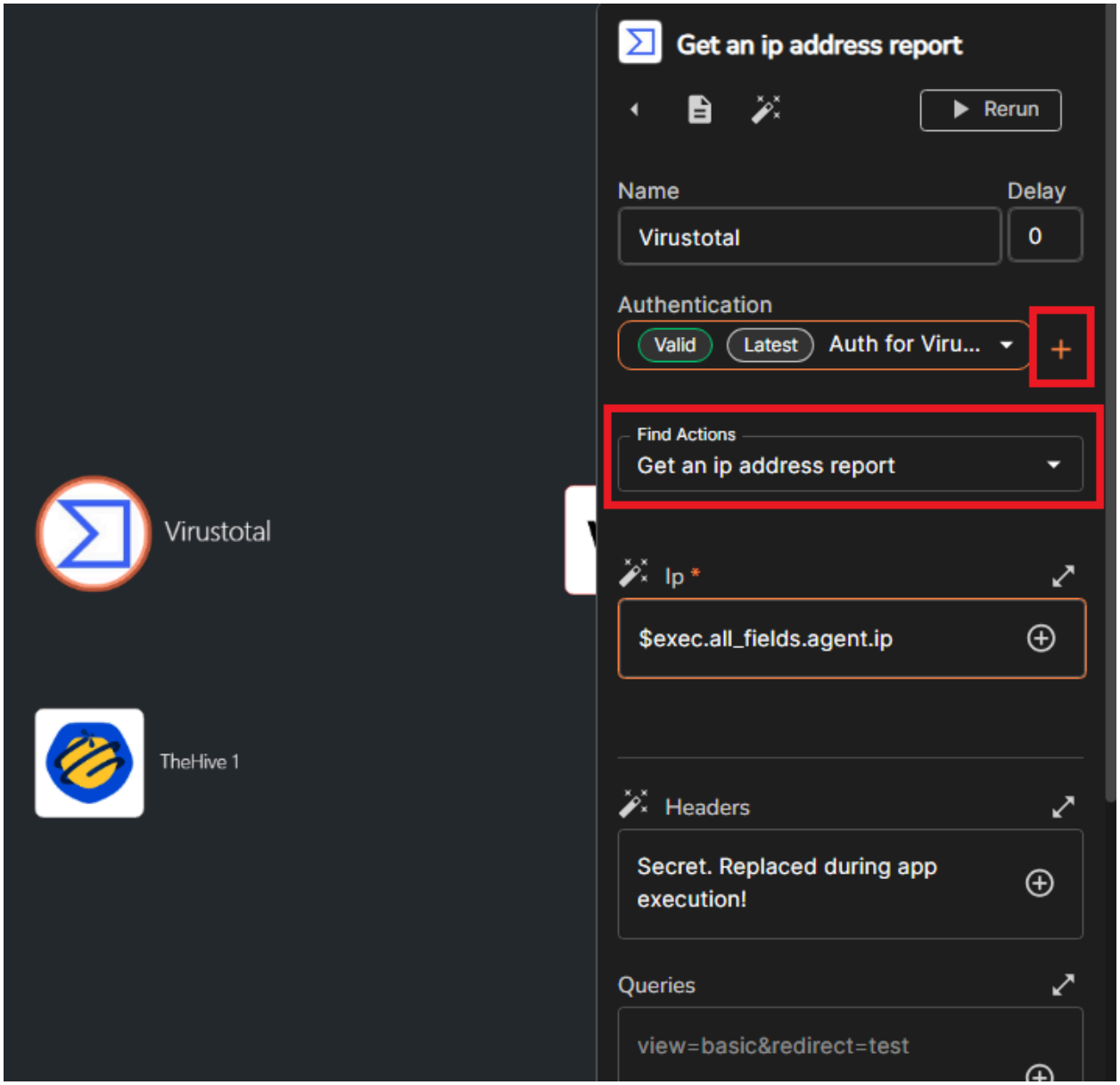
Subsection - VirusTotal Widget

Get the API after making an account with VirusTotal. Once logged in, click on your name and then "**API key**" from the dropdown. It will be at the top of the page.

Back in shuffle, click on the VirusTotal Widget (make sure it is the square widget)

Rename to: VirusTotal
Find Actions: Get an IP address report
Click on the "+" sign to authenticate.
Input API key
Leave URL as is

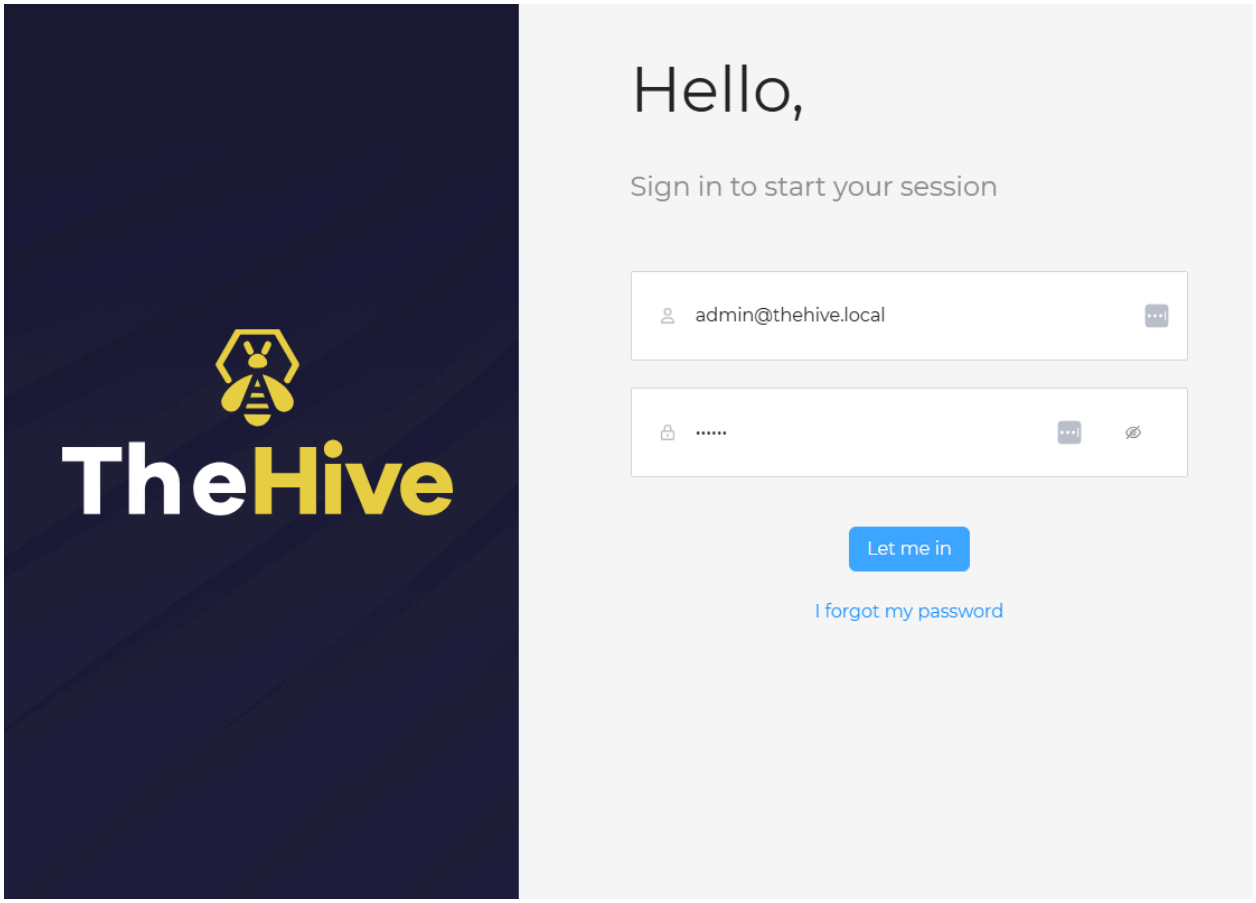
The IP field should be "**$exec.all_fields.data.srcip**"



Subsection - TheHive Widget

Let us log into TheHive Server using the default creds

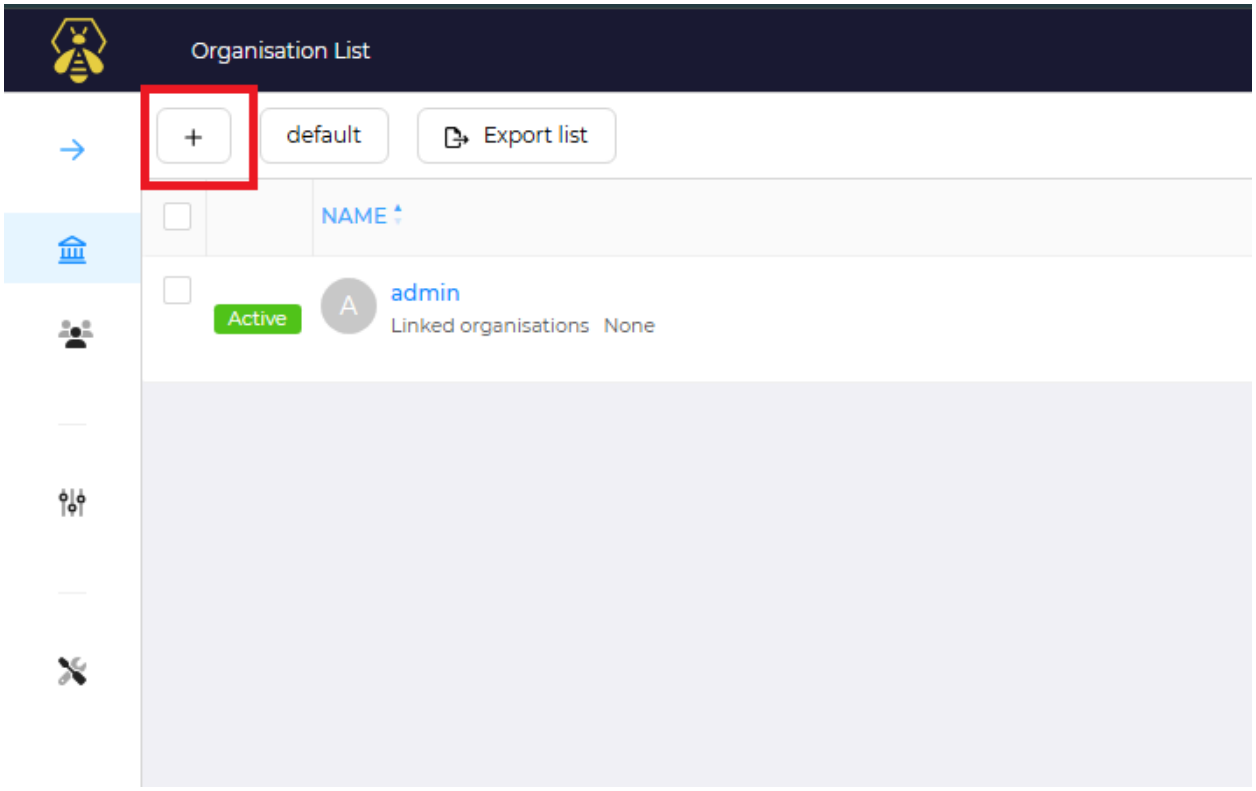
admin@thehive.local
secret



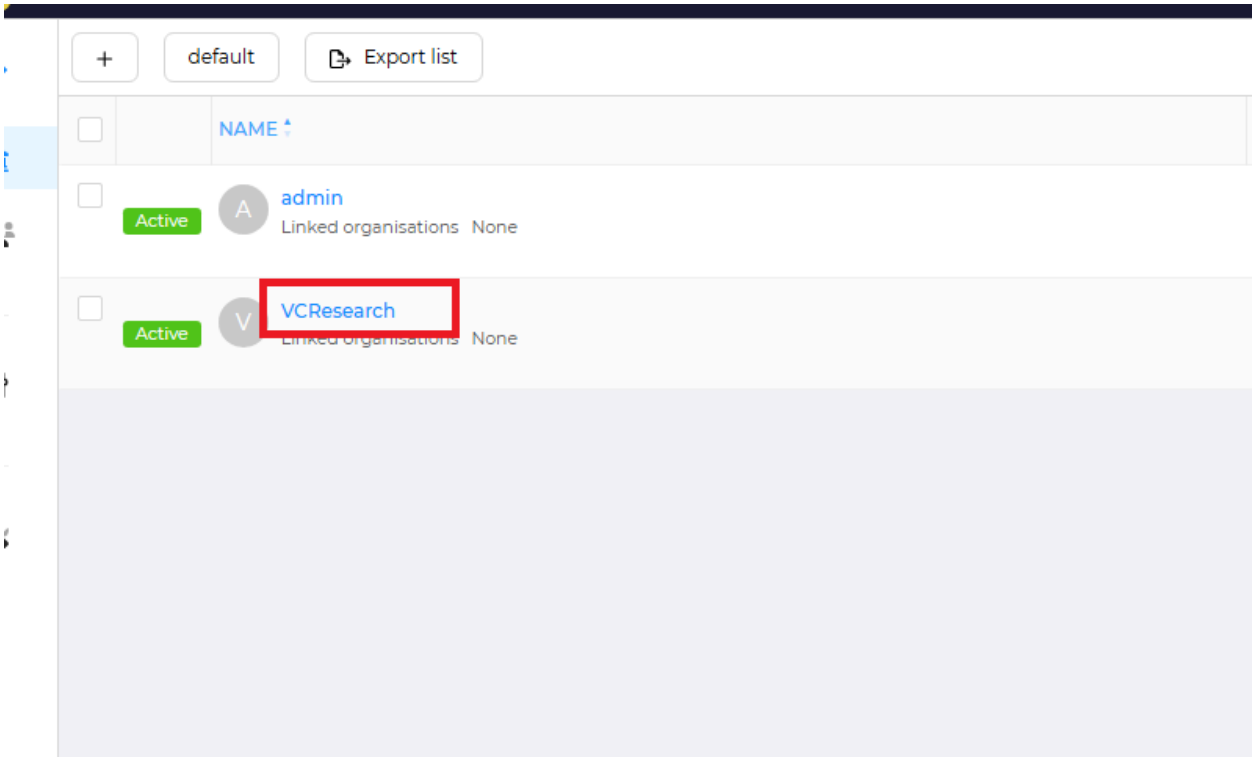
Create a new organization, and input this (You can name it something else)

- **Name:** VCResearch
- **Description:** SOC Automation project
- **Task sharing rule:** Manual
- **Observation sharing rule:** Manual

Click confirm at the bottom.



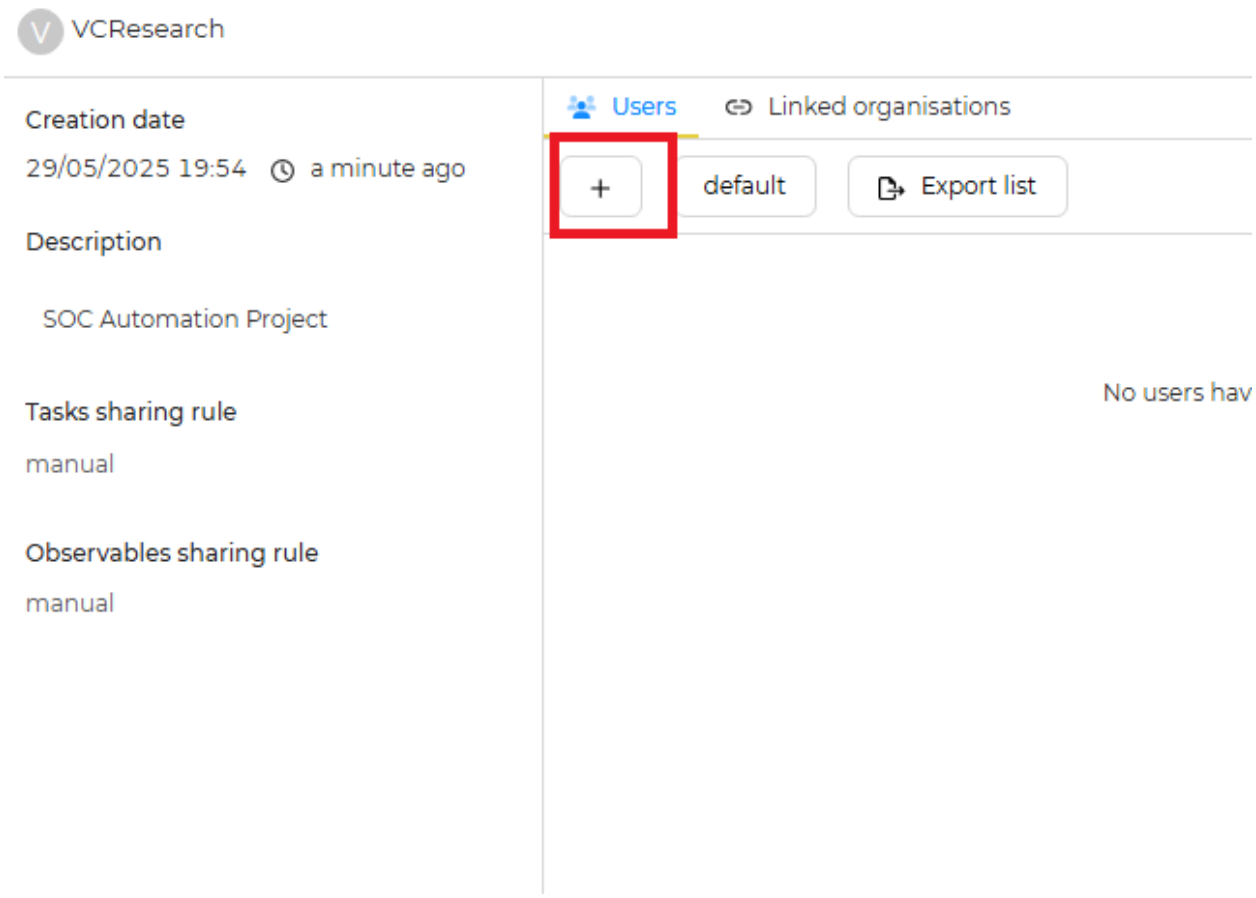
Click into the organization



And add a new user

- **Type:** Normal
- **Login:** VCresearcher@test.com
- **Name:** VCResearcher
- **Profile:** Analyst

Click on **“Save and add another.”**



For the second user

- **Type:** Service
- **Login:** shuffle@test.com
- **Name:** SOAR
- **Profile:** Analyst

Click on "**Confirm**" at the bottom.

On the left-hand side, click on "**Users**", and select "**Preview**" next to the VCResearcher account (Or equivalent)

Scroll down and input a password for the account. Click on "**Confirm**" when done.

For the SOAR User, click on "**Preview**," but this time create an API key (**Take note of this API**)

 # Change the Admin Password too.

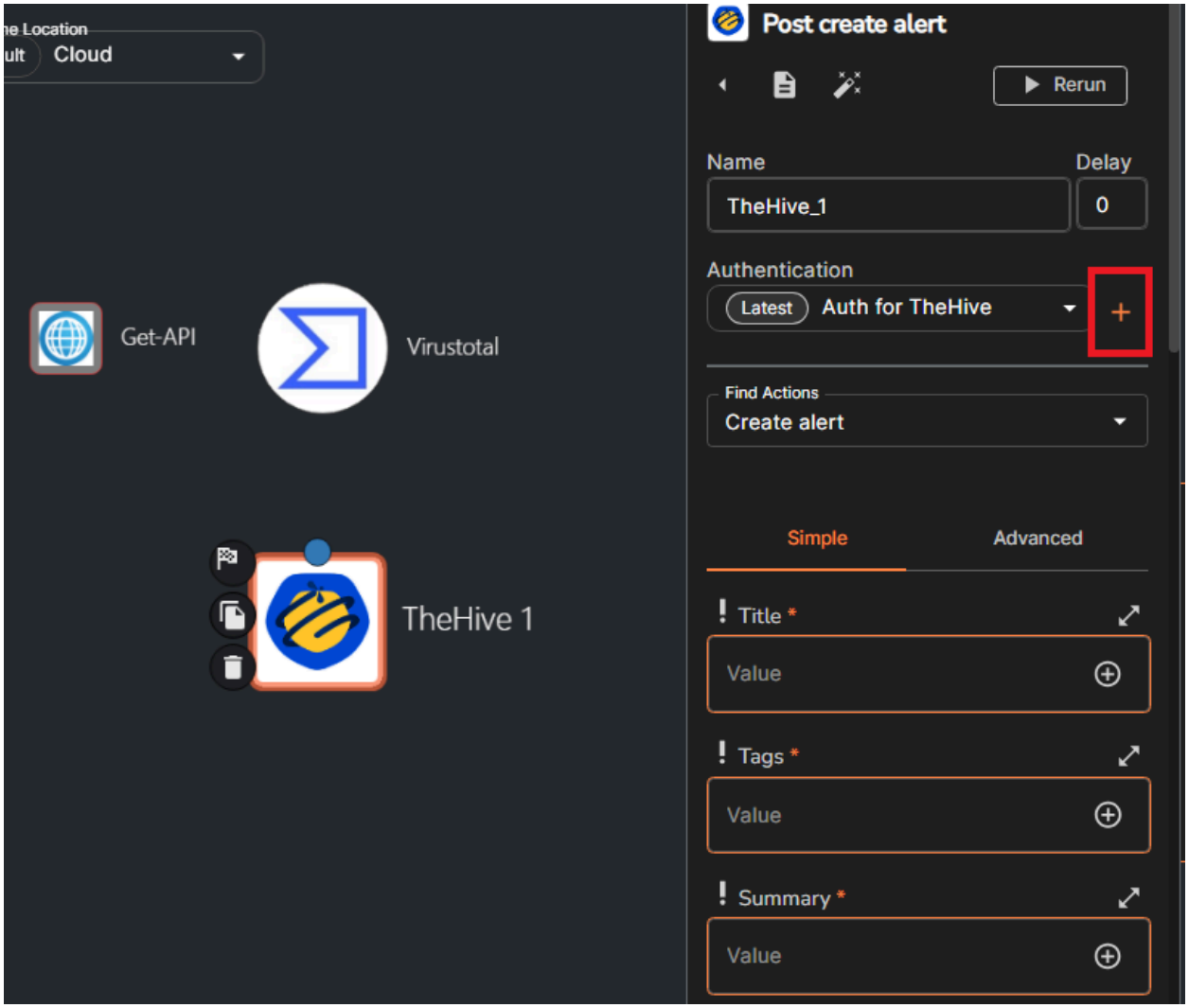
After copying the API key and changing the passwords, log out of the admin account and log in to the VCResearcher account.

 # Note: Use the Email for the account, not the name.

Back on shuffle, authenticate to TheHive just like with VirusTotal.

Click on the "+" sign to authenticate

- API key
- URL is the public IP and port of TheHive



Just like with Wazuh, we need to modify the DigitalOcean Firewalls.

You can make a separate firewall for TheHive and separate it or just allow the current firewall group to accept TCP Traffic on Port 9000 from all IPv4 addresses.

Back in shuffle, inside of TheHive Widget, click on the "**advanced**" tab and use the body template below, and fill in the title, tag, summary, and severity. Modify it as you see fit.

 # IMPORTANT NOTE: This is after I completed and brought down the infrastructure.
 # The payload below will have a "sourceRef" field.
 # I recommend you add the timestamp from your webhook to that number.

 # The reason is that TheHive will not make any more alerts if an alert with this sourceRef already
 # exists. The timestamp will make it unique.

Template for shuffle TheHive payload

IP Trigger message

 # Body

```
{
 "description": "{{ '''SSH Brute Force Detected''' | replace: '\n', '\\r\\n' }}",
 "externallink": "",
 "flag": "",
 "pap": "",
 "severity": "2",
 "source": "Wazuh",
 "sourceRef": "\"Rule:100003\"",
 "status": "",
 "summary": "SSH Brute Force Detected",
 "tags": ["T1110"],
 "title": "Shuffle response",
 "tlp": "2",
 "type": "Internal",
 "observables": [
  {
   "dataType": "ip",
   "data": "$exec.all_fields.data.srcip",
   "ioc": true
  }
 ]
}
```

> Title
Shuffle response

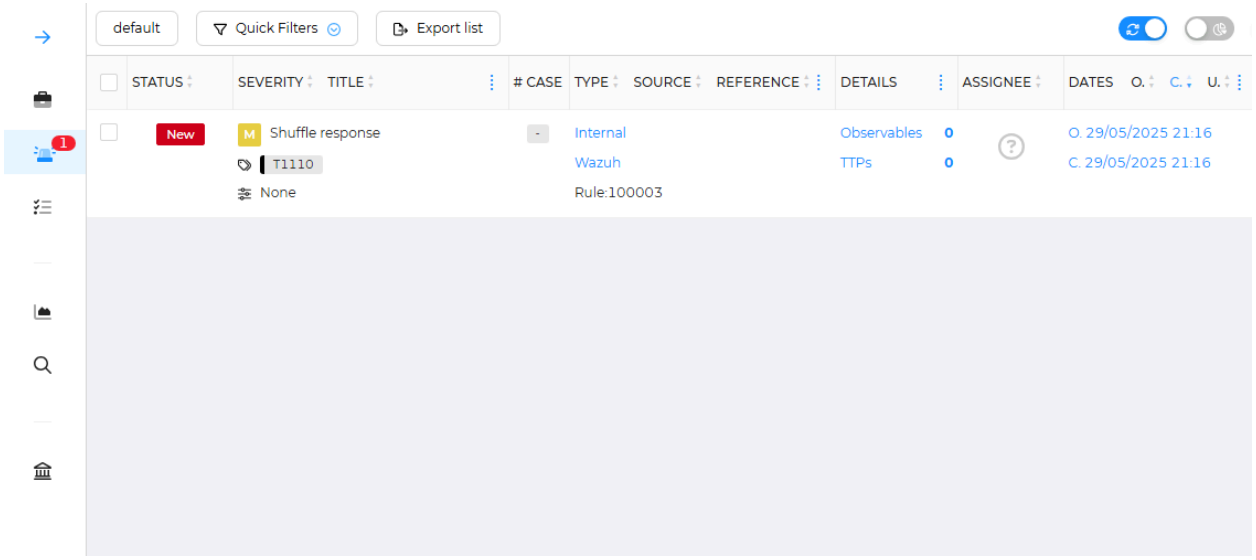
> - Tags
["T1110"]

> - Summary
RDP Brute Force Detected

> - Severity
2

 # NOTE: Severity is 0-3, Low to severe

We can rerun previous attempts at SSH login failures, and it should make a case in TheHive.



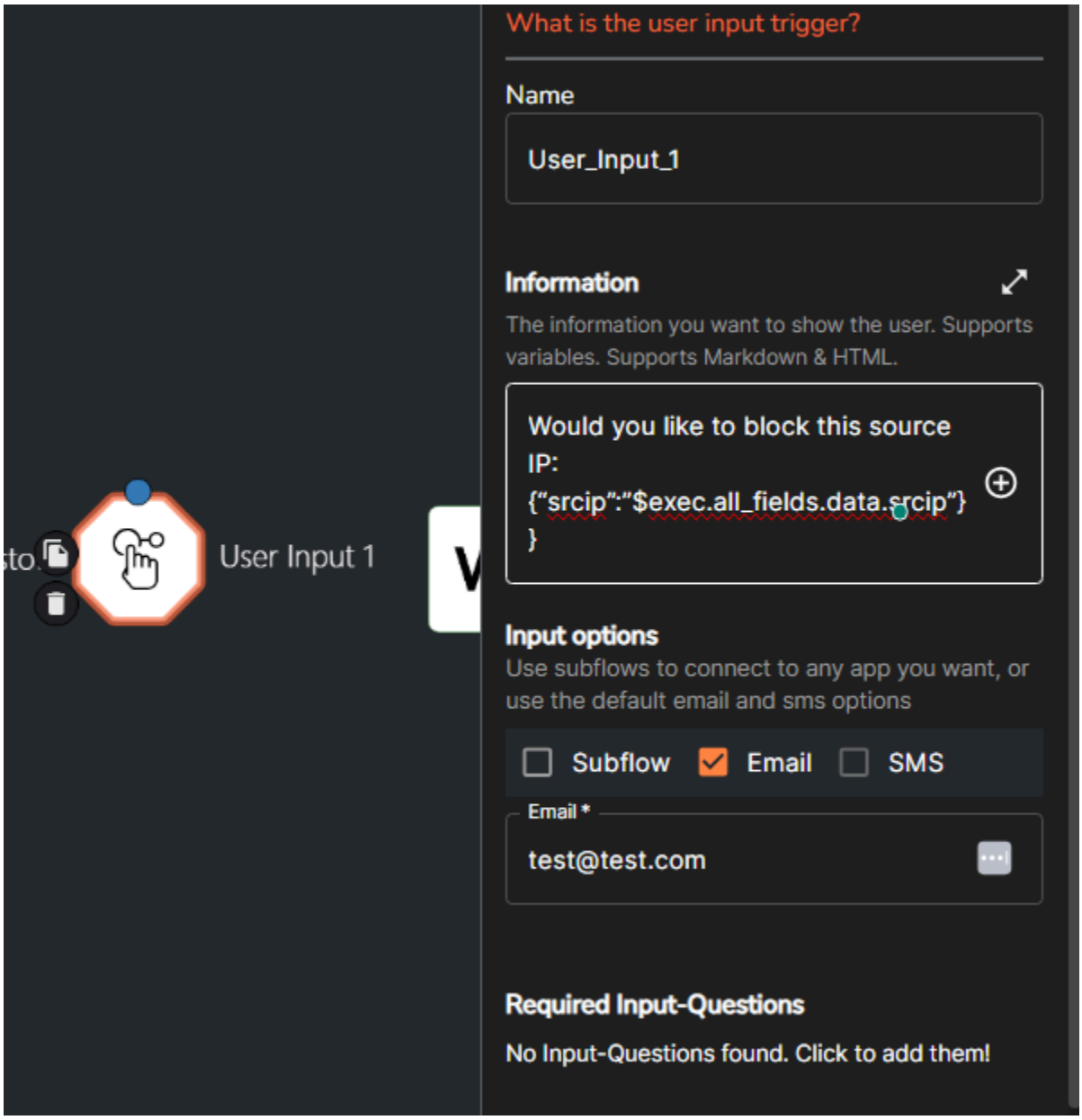
Subsection - User Input Widget

To set up the Email widget that will be used for approving the active response. On the User Input Widget, fill out the fields with the information below.

Information: Would you like to block this source IP: {“srcip”:”$exec.all_fields.data.srcip”}}

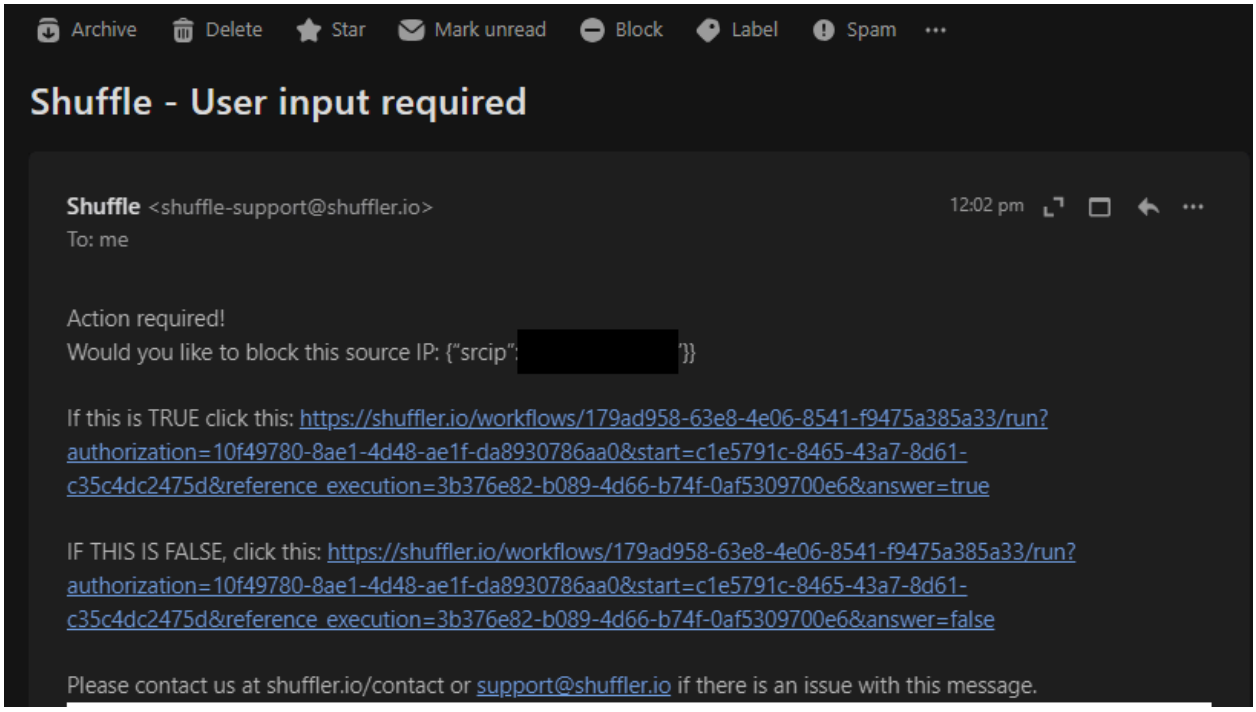
Contact Option: Email

Email: Your email



The email to be received should be like the one below.

The top link will be to continue the workflow.
The bottom will abort it.



Subsection - Wazuh Active Response Widget

Parameters to change

Find Actions: Run command

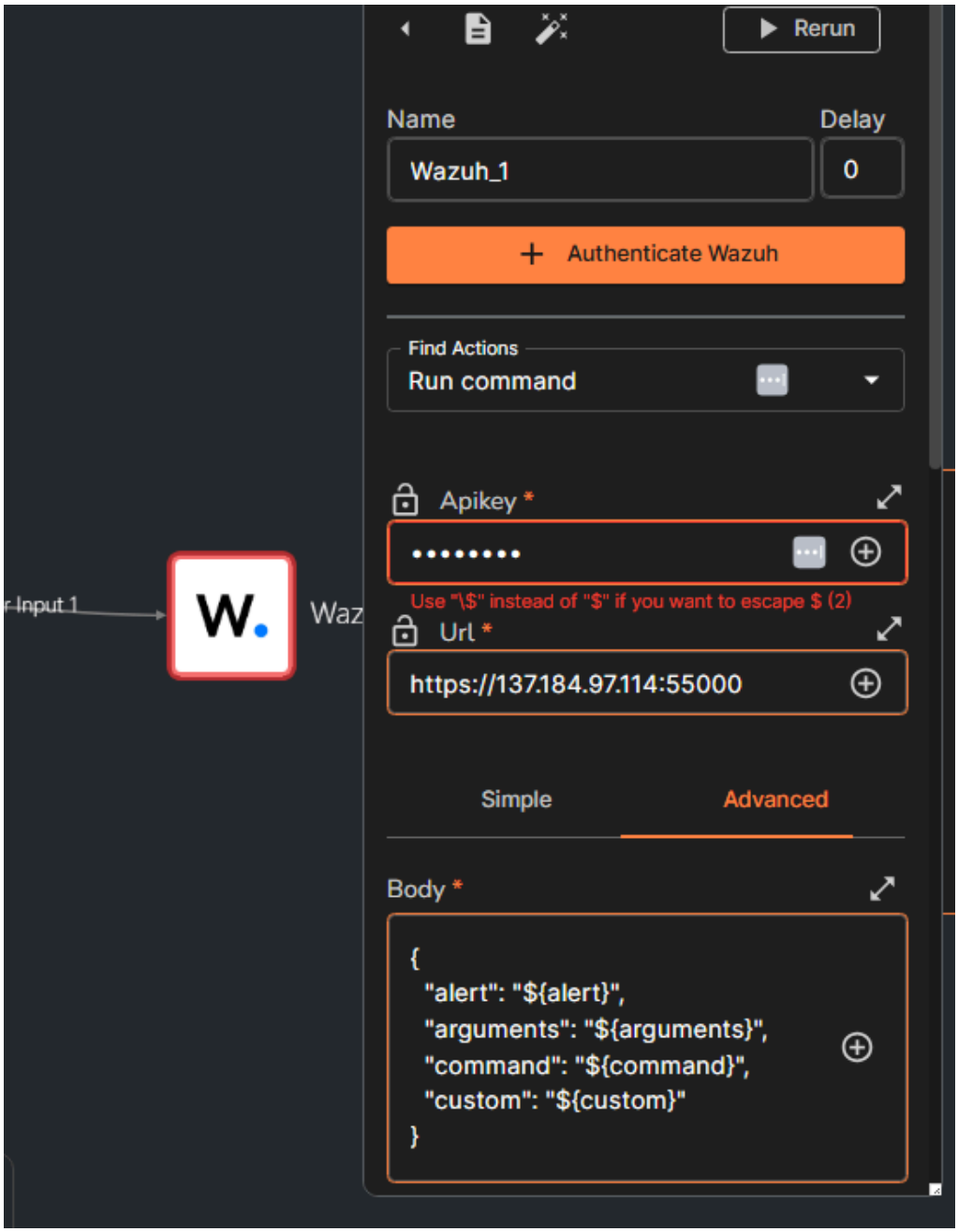
ApiKey: # Should be the output of “Get-API”

URL: swap localhost to Wazuh public IP

Command: firewall-drop0

Agent list: $exec.all_fields.agents.id

Wait for complete: true



 # In the Wazuh server
#Nano into the ossec.conf file
Nano /var/ossec/etc/ossec.conf

 # Click on CTRL+W and search for "active response"

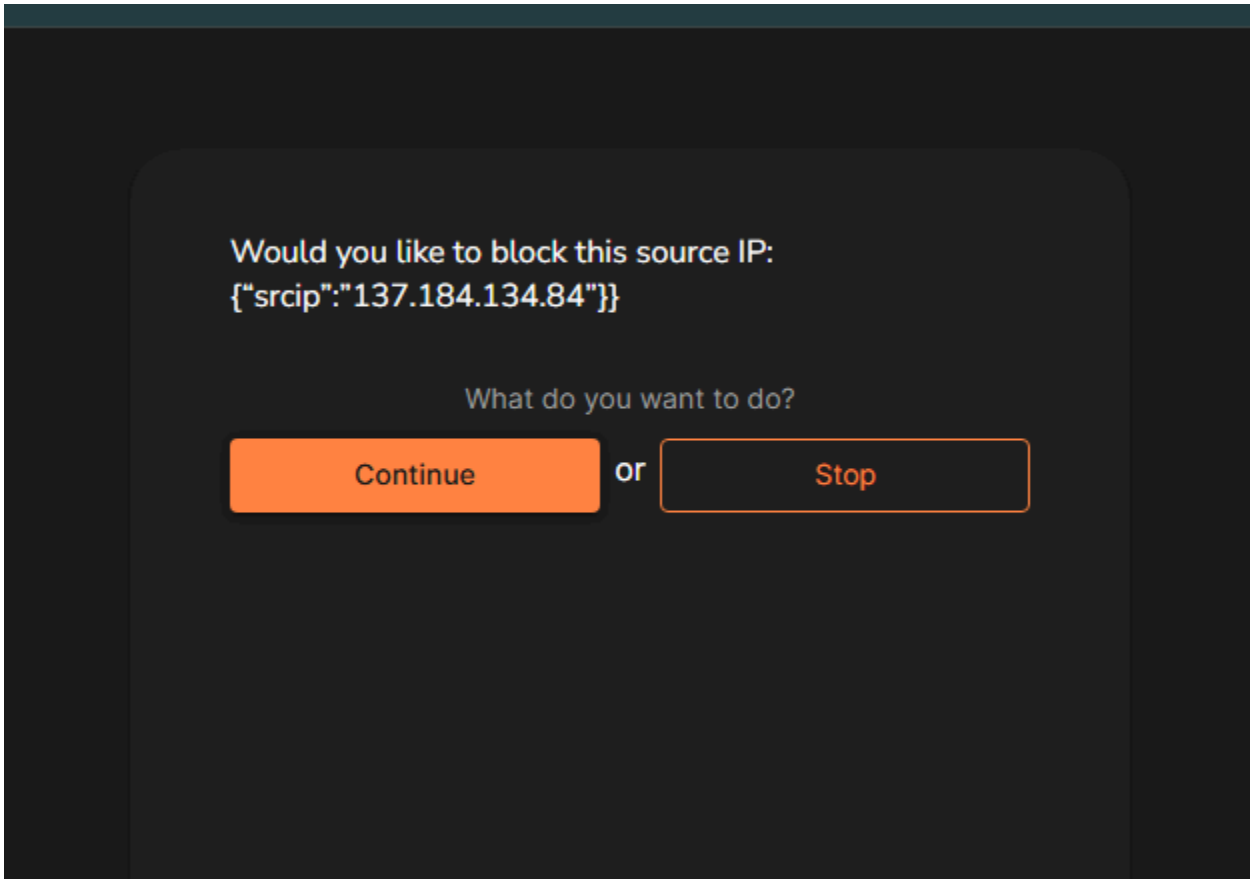
 # Scroll down to the bottom of active response, uncomment and add in the block below to where it
 # says "active response commands here"

<active-response>
<command>firewall-drop</command>
<location>local</location>
<level>13</level>
<timeout>no</timeout>
</active-response>

 # ctrl+x and Y to save, and ENTER to exit
Systemctl restart wazuh-manager

I tested this active response feature by allowing MISP to SSH to the Honeypot.

When the workflow was triggered and I clicked on the link in the Email, I was directed to this page.



After clicking on "**continue**", the response begins.

We can verify through the shuffle logs or run an iptables command on the honeypot to see the blocked IP.

```
root@VCResearch-Linux:/var/ossec/bin#
root@VCResearch-Linux:/var/ossec/bin# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

DROP       all  --  137.184.134.84       anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
```

 # Since this is the MISP IP, we would want to remove it.

 # If you are unfamiliar with iptables, run the command below to list blocked IPs and their line number.
Iptables -L –line-number

 # For the MISP IP, it was number 1, so I had to run these commands to unblock it…
Iptables -D INPUT 1
Iptables -D FORWARD 1

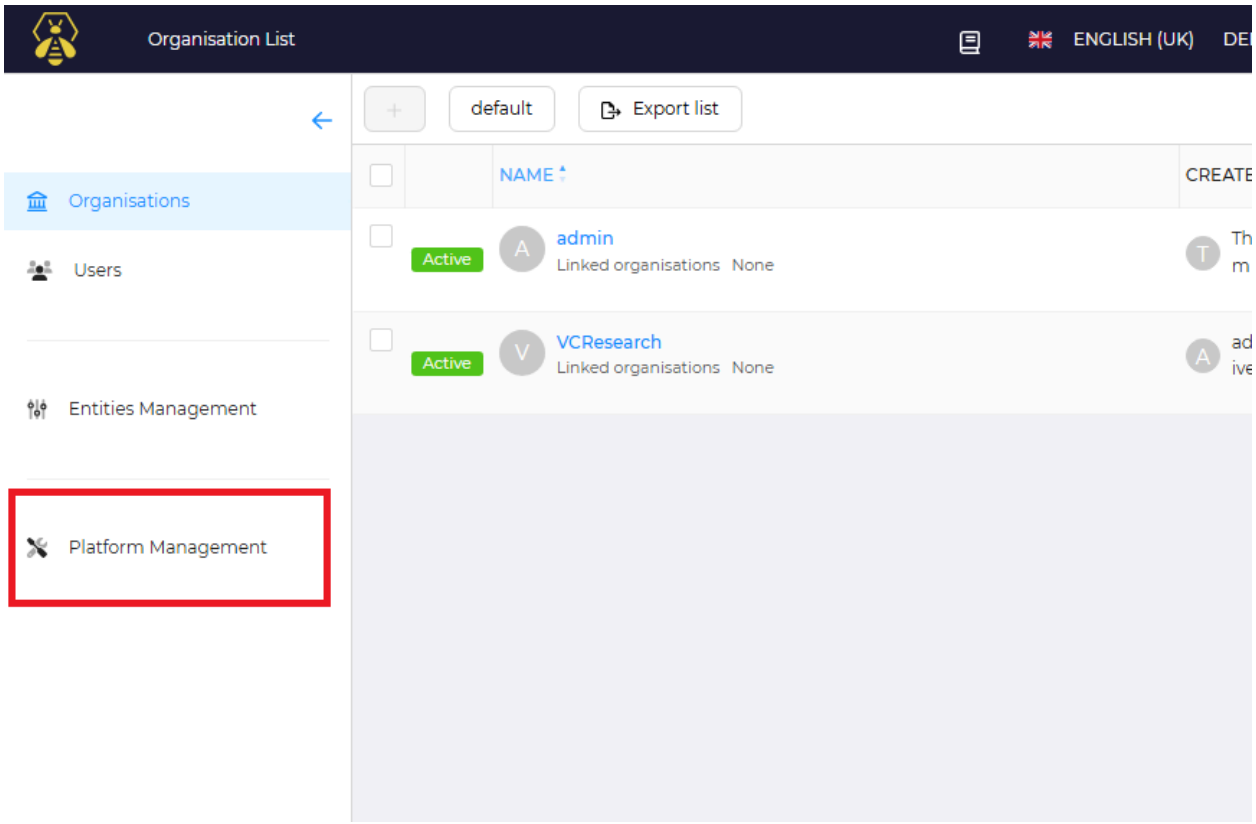
 # The picture shows an empty iptables again.

```
root@VCResearch-Linux:/var/ossec/bin# iptables -D INPUT 1
root@VCResearch-Linux:/var/ossec/bin# iptables -D FORWARD 1
root@VCResearch-Linux:/var/ossec/bin# iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
root@VCResearch-Linux:/var/ossec/bin#
```

Section 9 - Integrating TheHive into MISP

Will need to make a firewall rule to allow communications between TheHive and MISP
The firewall rule can just be all IPv4 traffic between the two.

After changing those rules,
Log in to the admin account in TheHive, then click on "**Platform management.**"

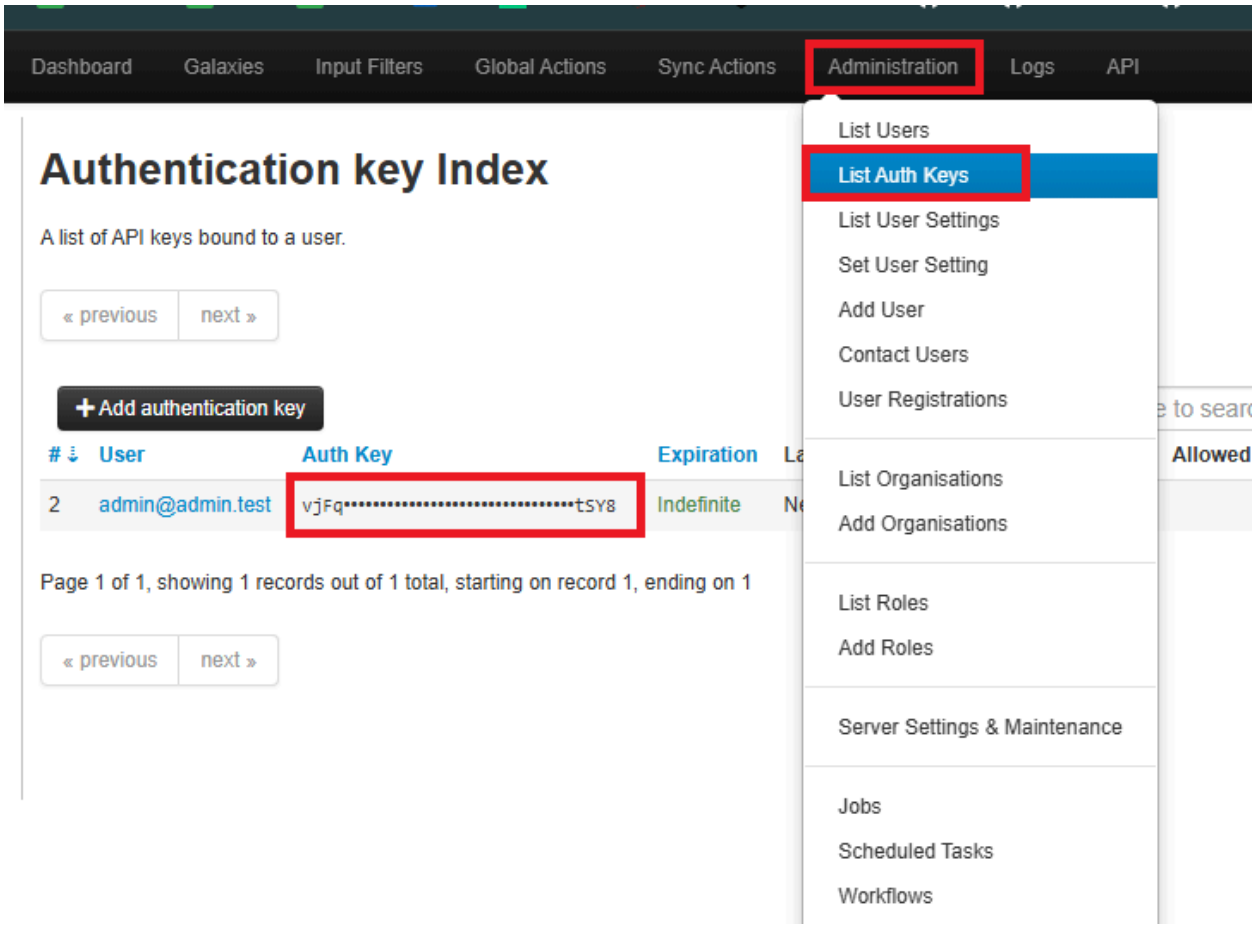


Fill out the information.

Interval: 1 minute

Server Name: MISP
Server URL: MISP URL
API: <Grab from MISP>
Purpose: Import and Export
Proxy disabled
Check that the **cert authority** is **disabled**, just because this is a project.
Everything else was left as default

To get the MISP API key, log in to MISP, select the administration tab, and list the Auth keys. You will have to create a new one.



Auth keys are used for API access. A user can have more than one authkey, so if you would like to use separate keys per tool that queries MISP, add additional keys. Use the comment field to make identifying your keys easier.

User

admin@admin.test

Comment

MISP to Hive Key

Allowed IPs

165.227.197.170/

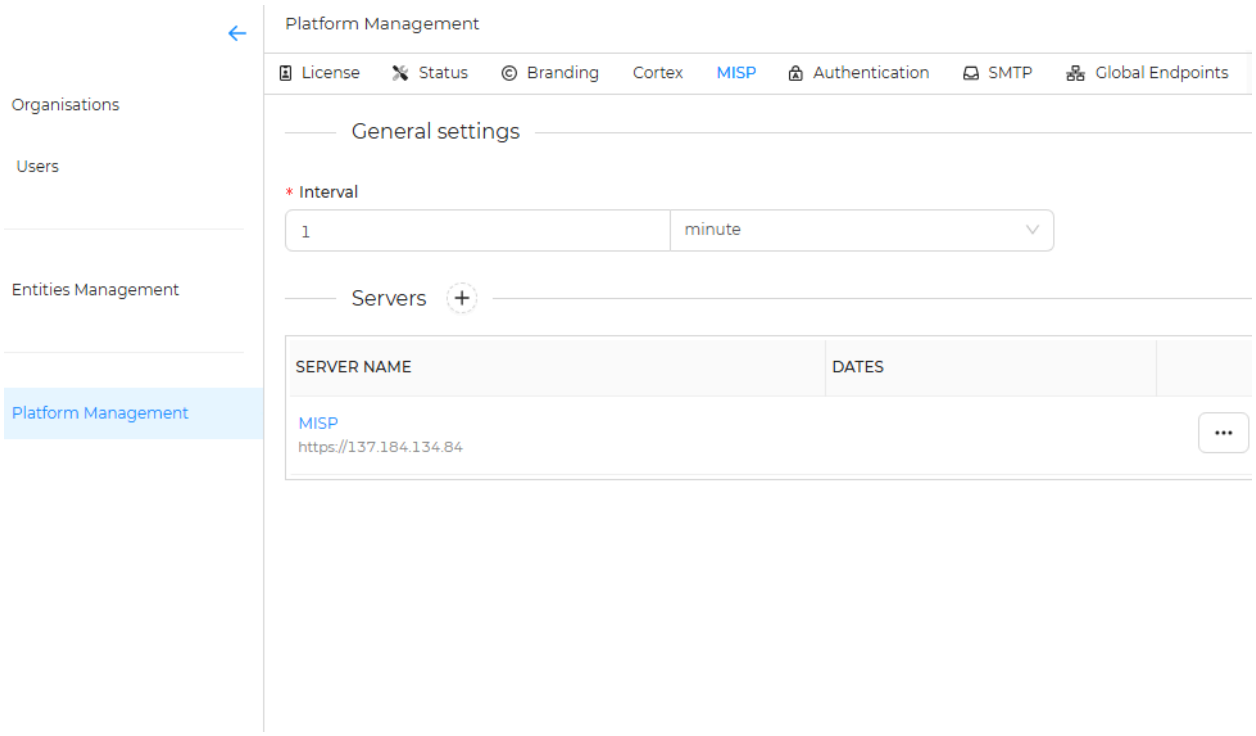
Expiration (keep empty for indefinite)

YYYY-MM-DD

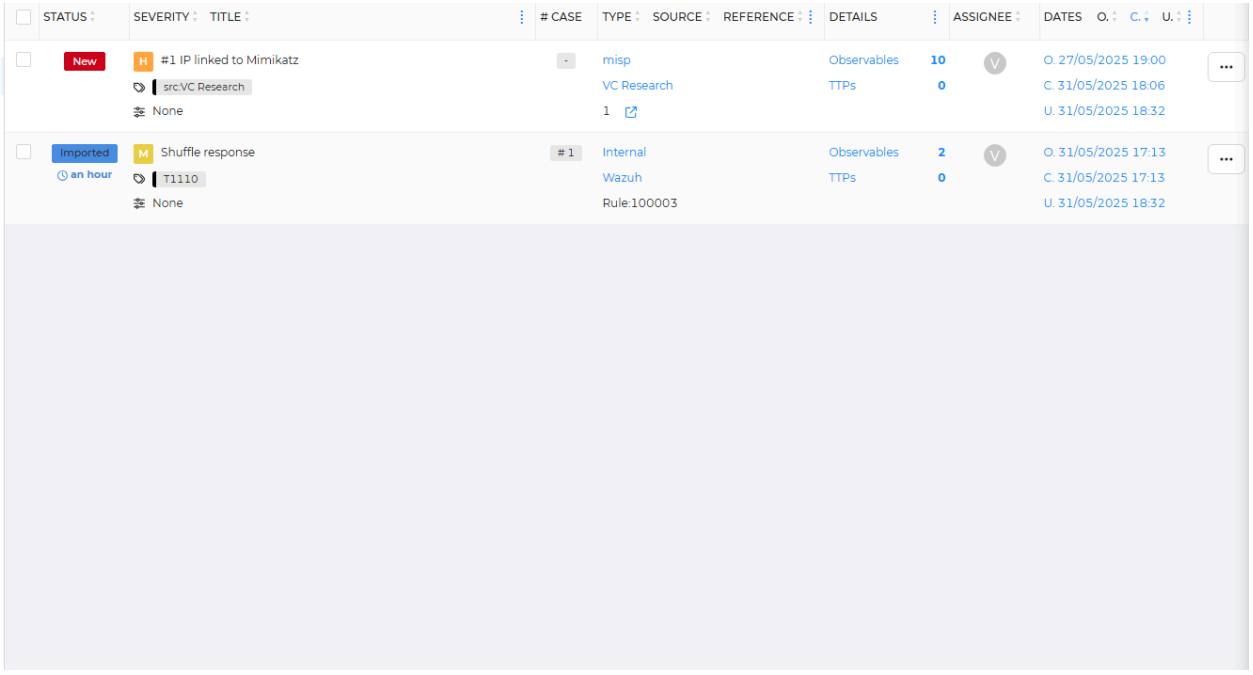
Read only (it will unset all permissions. This should not be used for sync users)

Submit Cancel

At the bottom of the configurations, there will be a button to test the connection. Click on it, if it succeeds, then confirm it.



Can test it out by making a case with the Mimikatz IP and see if MISP enriches TheHive



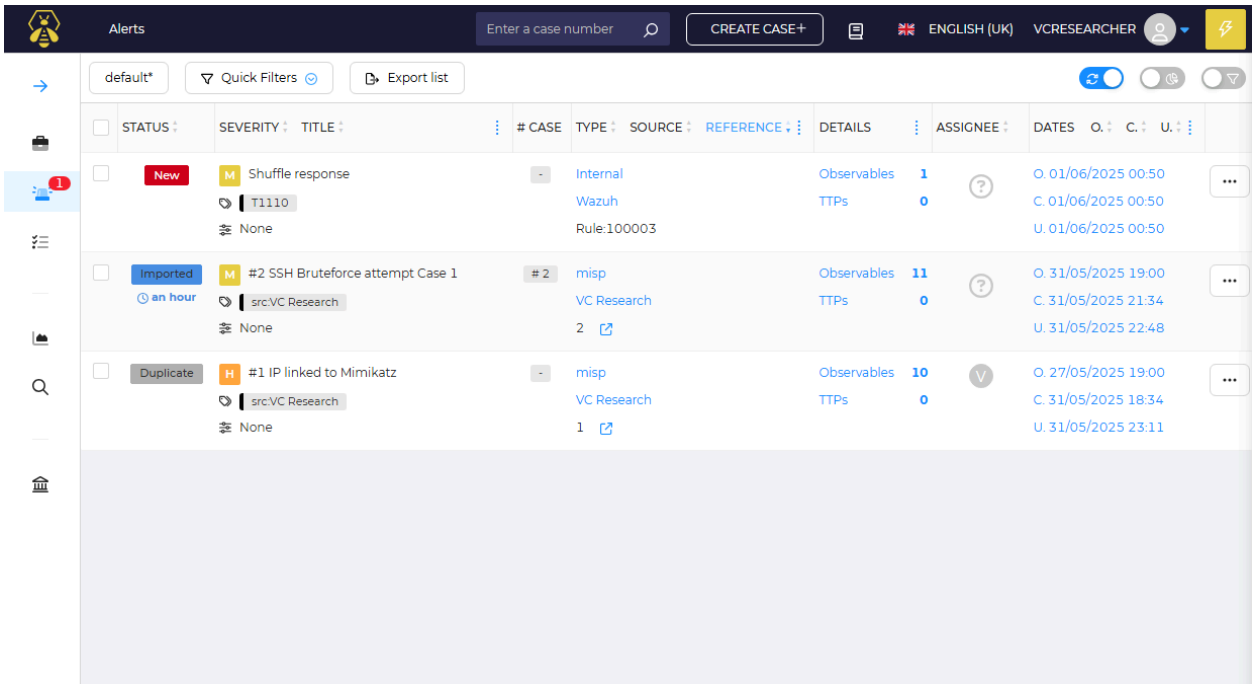
Section 10 - Atomic Threat Intelligence Gathered

Because of other matters I had to take care of, I had to bring the infrastructure down fast. Which led to me not being able to troubleshoot the alert generation issue. I do plan to append the results when I run through this lab a third time.

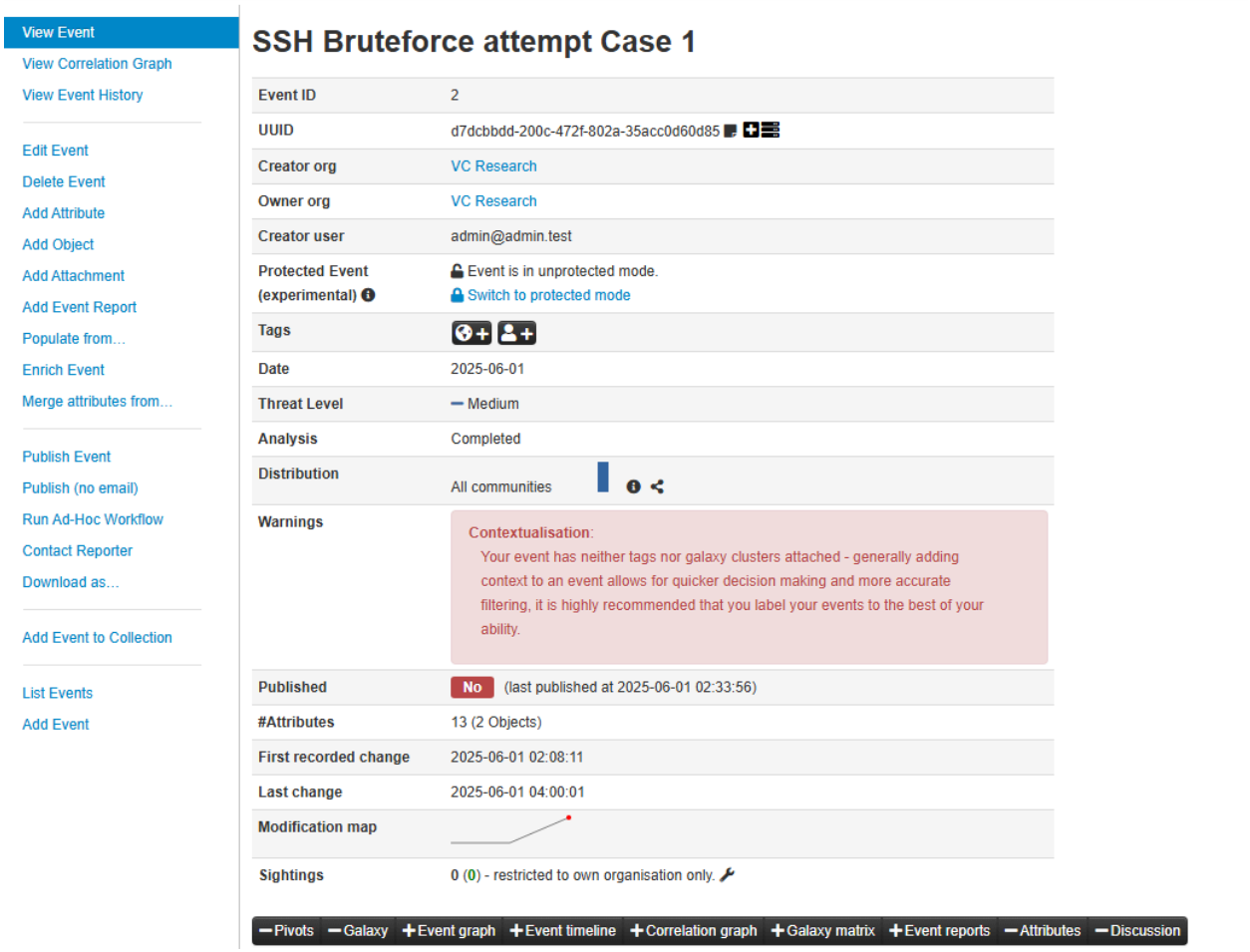
As brought up before, the sourceRef parameter in shuffle was not unique, so I ran into the problem that new events were not being generated in the hive. But the emails and active response were still working.

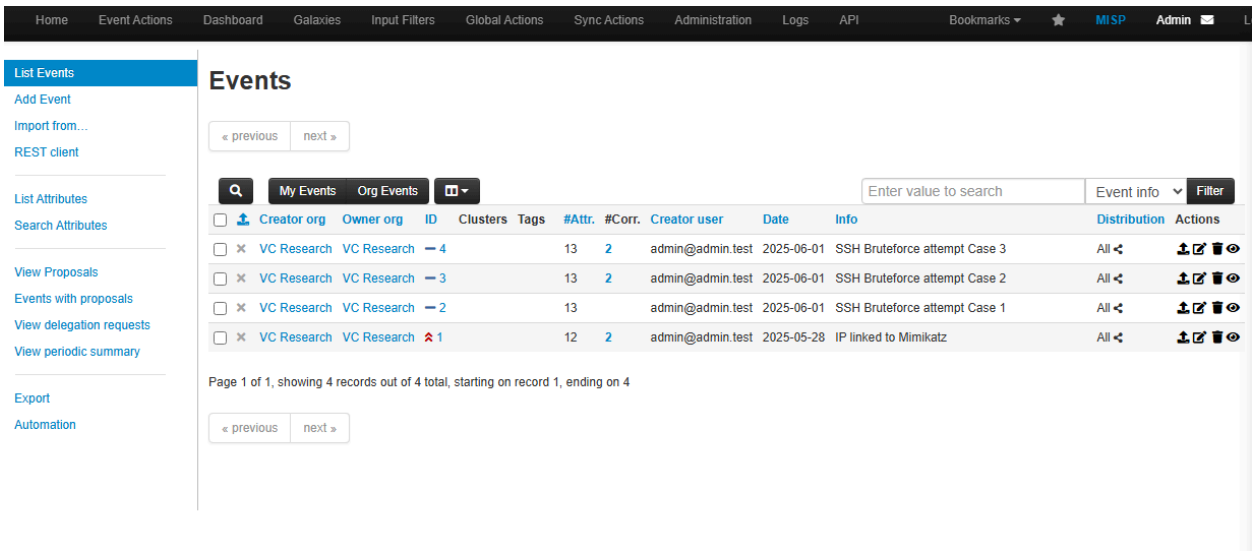
The image below is just the first brute force attempt alert that was created, and a secondary alert is being created with the enriched observables.

The third was the enriched alert that was created when I tested the Mimikatz-linked IP.



In the alerts created, they have a link pointing back to MISP, which allows analysts to see the full list of observables and how they were correlated. The next two images show those MISP pages.





– End of project –