
$ ssh -p 22 unknown@beestreet
Connecting to beestreet...
Connection established.
Server: Cowrie SSH Honeypot
Fingerprint:
SHA256:22:B3:EE:ST:RE:E5:22:
H0:NE:YP:0T:22:00:22:22:22
Warning: You are being logged.
All actions are monitored.
unknown@beestreet:~$
CONNECTION LOGGED
IP ADDRESS 203.0.113.22
LOCATION UNKNOWN
USER unknown
ACTION login attempt
STATUS captured
PROJECT
22
BEE STREET
CLOUD SSH HONEYPOT
HONEY. DATA. CONTROL.
WE DON'T GET HACKED.
WE HUNT.
PORT
22
SSH
DECEPTION LAYER: ACTIVE
HONEYPOT STATUS: ONLINE
aws
BEE STREET
22
INTRUDERS CHECK IN.
WE LOG IT ALL.
AWS EC2 • COWRIE • PORT 22 DECEPTION

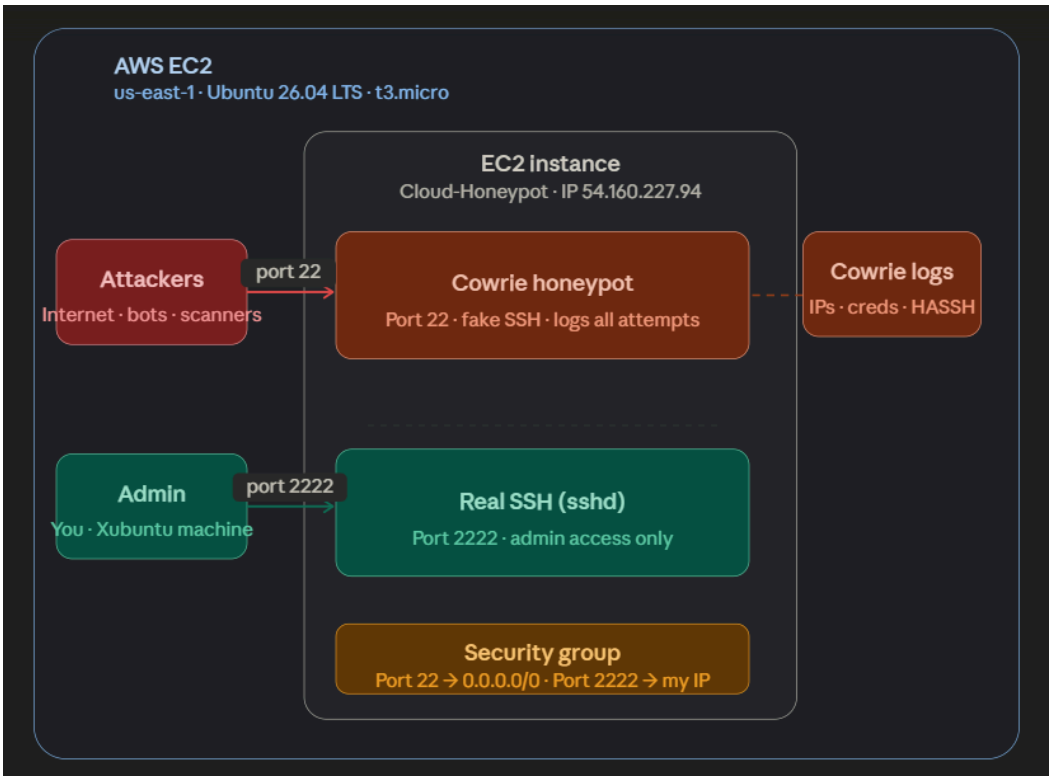
# Phase 1 - Cloud Honeypot Foundation

## Introduction

This phase created the foundation for a cloud honeypot deployed on an AWS EC2 instance. The central problem addressed in Phase 1 is that internet-facing cloud SSH services are constantly targeted by automated scanning, brute-force attacks, and credential-stuffing attempts.

One 2026 Linux security analysis showed that fresh cloud server instances can receive their first SSH probing attempt within 90 seconds of deployment. A 2025 security report also demonstrated that attacks against internet-facing login services and edge infrastructure were occurring from approximately 2.8 million IP addresses per day, indicating that these attacks are heavily automated and botnet-driven.

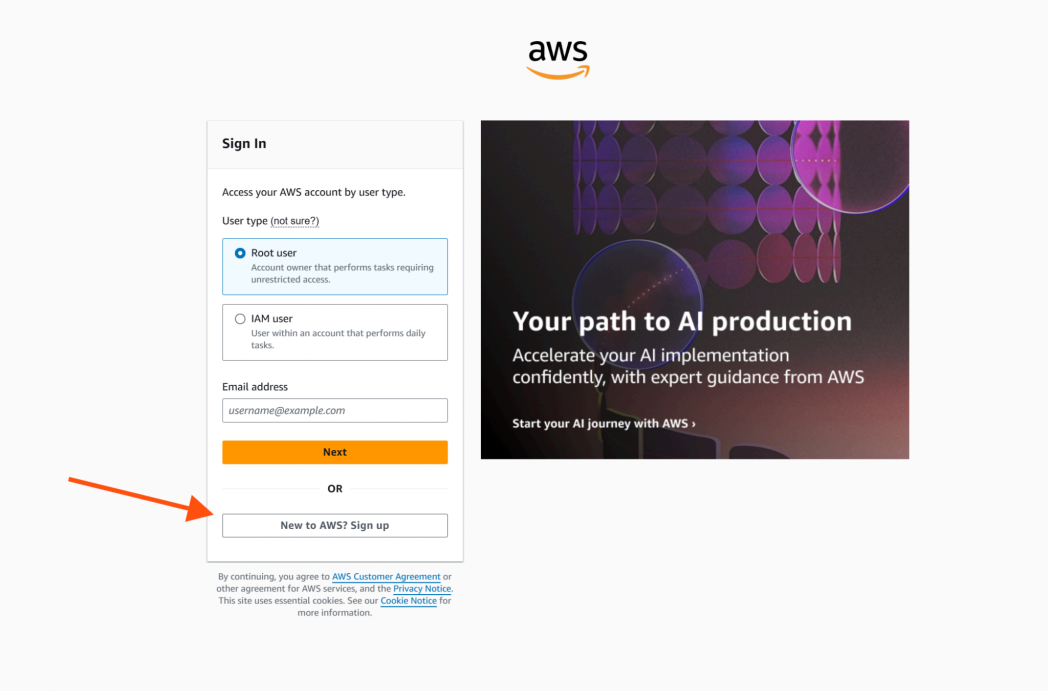
Phase 1 addresses this problem by hardening the SSH architecture. The real SSH service is moved from the default port, 22, to a nonstandard port, 2222. Cowrie is then deployed on port 22, SSH's default port, to act as a honeypot and capture attacker activity while keeping legitimate administrative access separate.

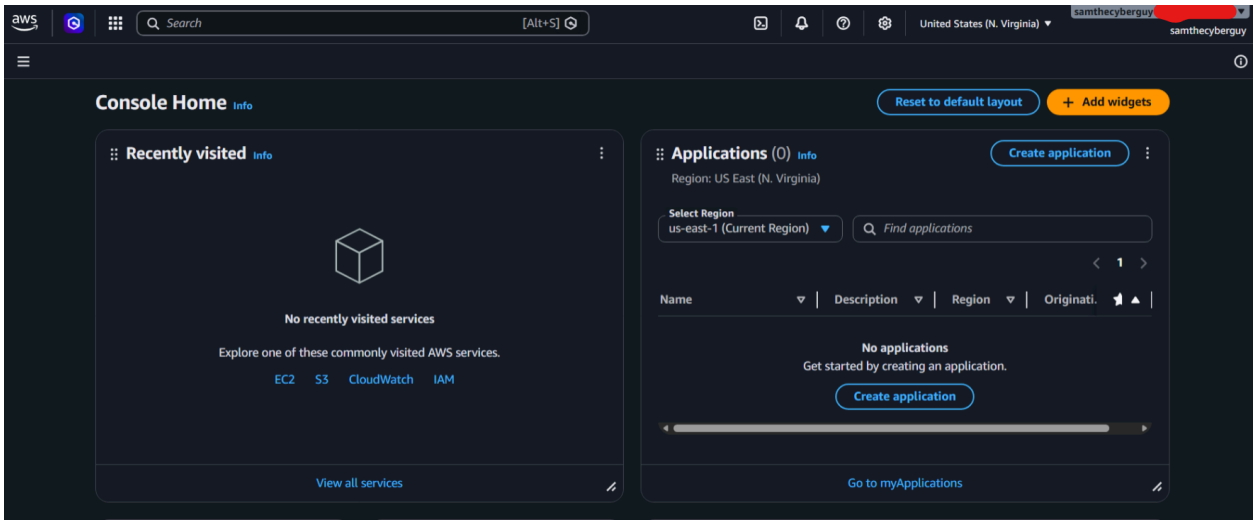



*Cloud Honeypot Foundation Diagram*

# Step 1 - Get an AWS Account

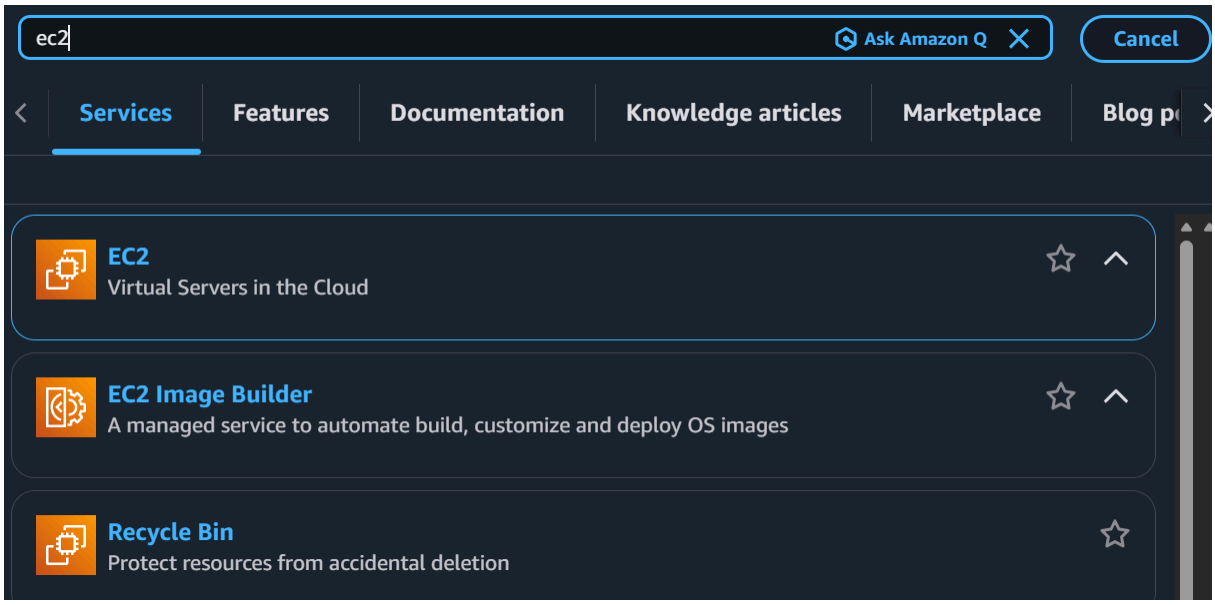
Set up a free AWS account on Windows 11. New users receive 100 dollars in credit, which can be used for 6 months.



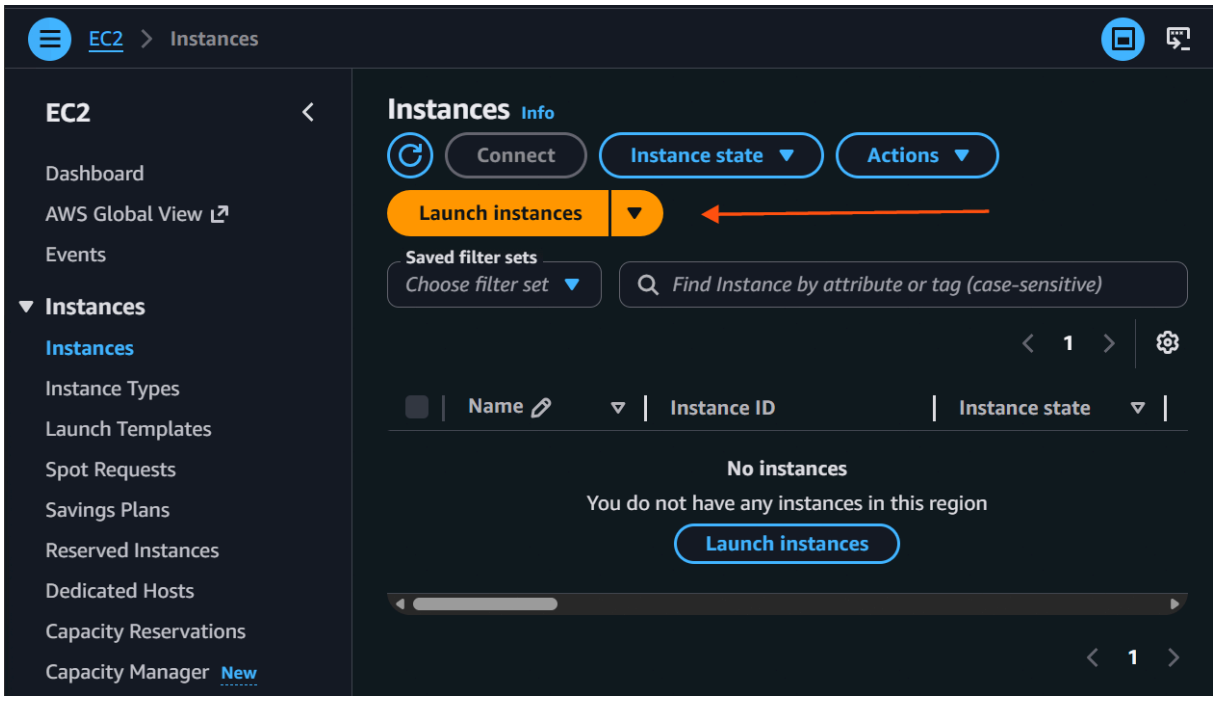



## Step 2 - Configure and Launch an EC2 Instance

Search for **EC2**. After selecting it, navigate to **Instances**.

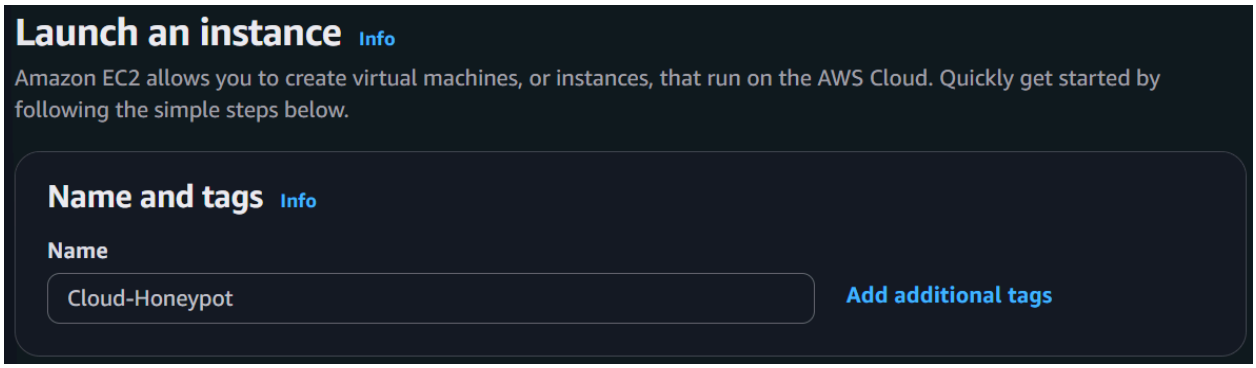


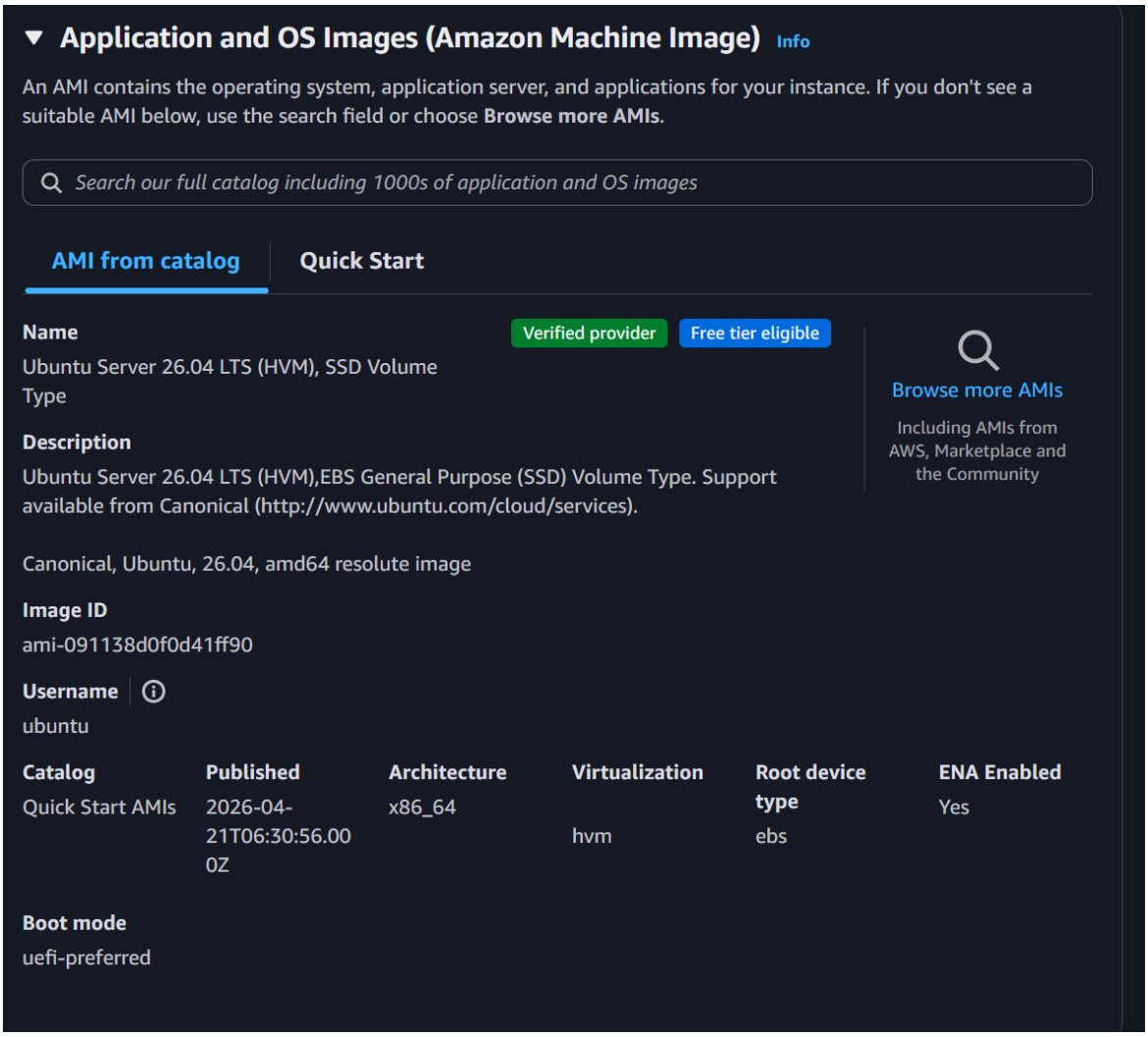

Open the **Launch instances** dropdown and select **Launch an instance**.

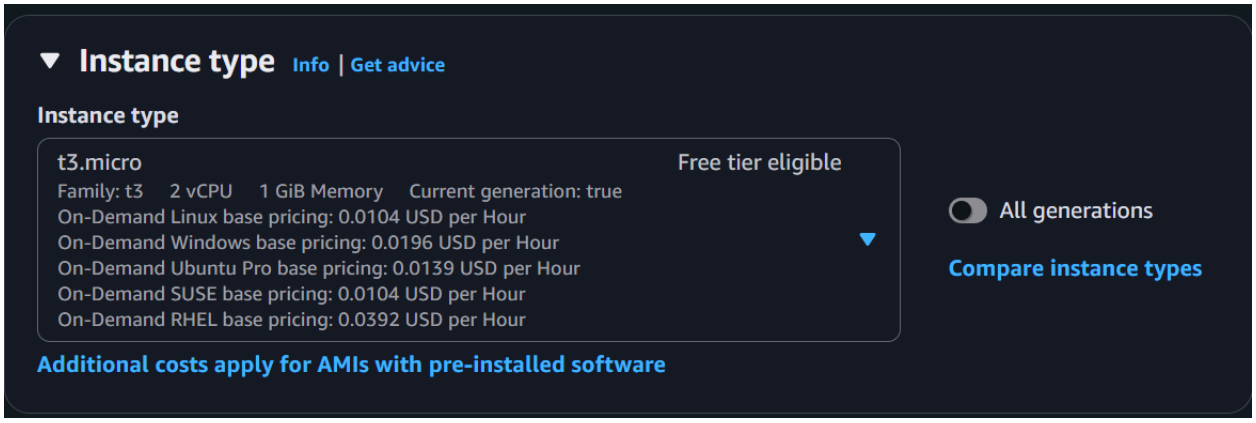


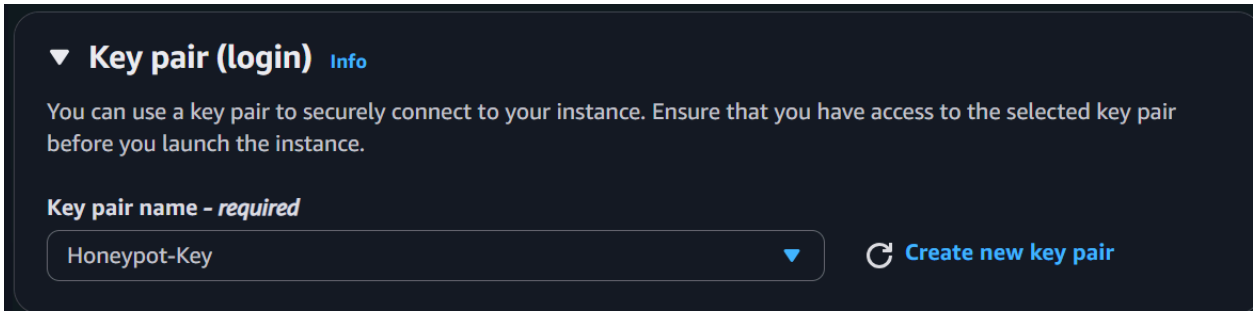

Configure the instance. Name the instance **Cloud-Honeypot**. Choose **Ubuntu Server 26.04 LTS x86** as the OS. Select **t3.micro** as the instance type. Create a key pair

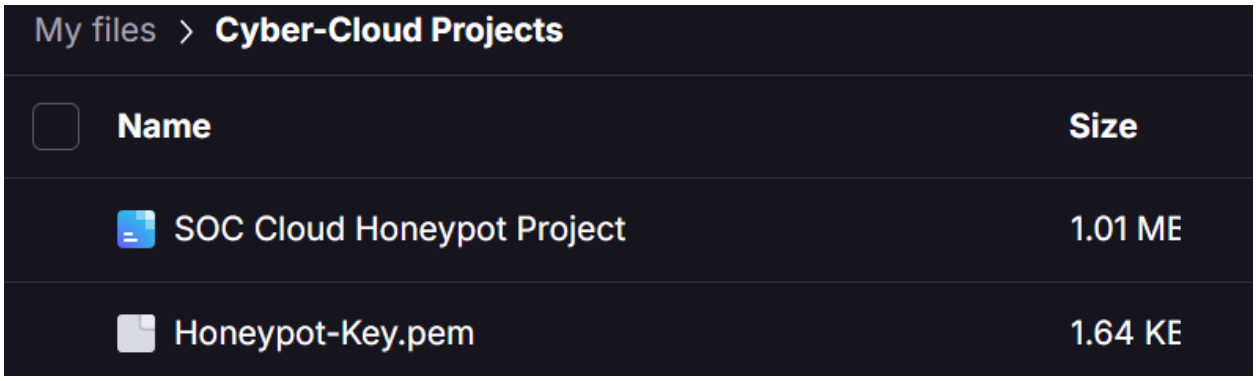
called **Honeypot-Key** and choose **RSA** and **.pem**. Save the key pair file to the Linux machine where the honeypot will be deployed. It will be used to login to SSH later.



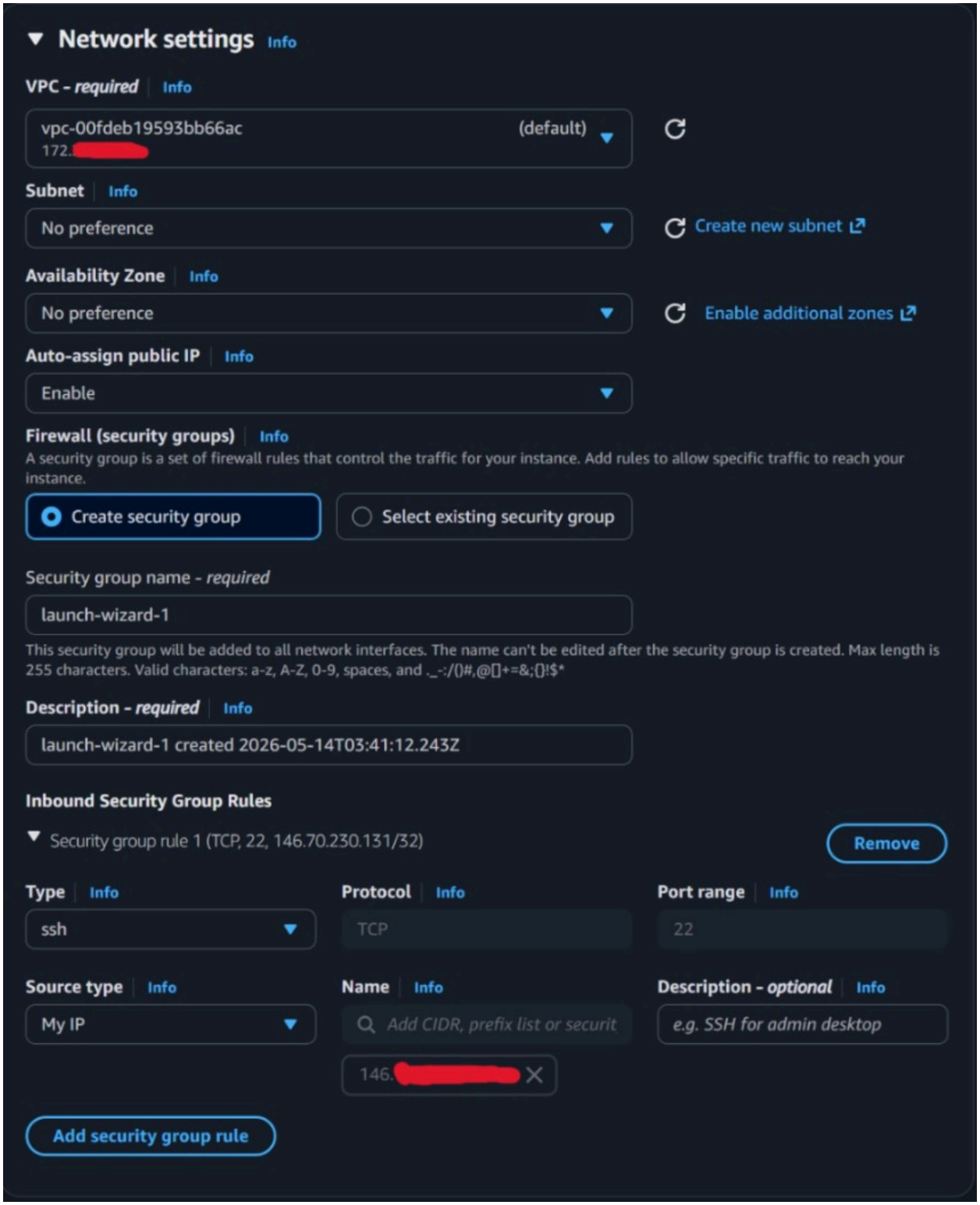





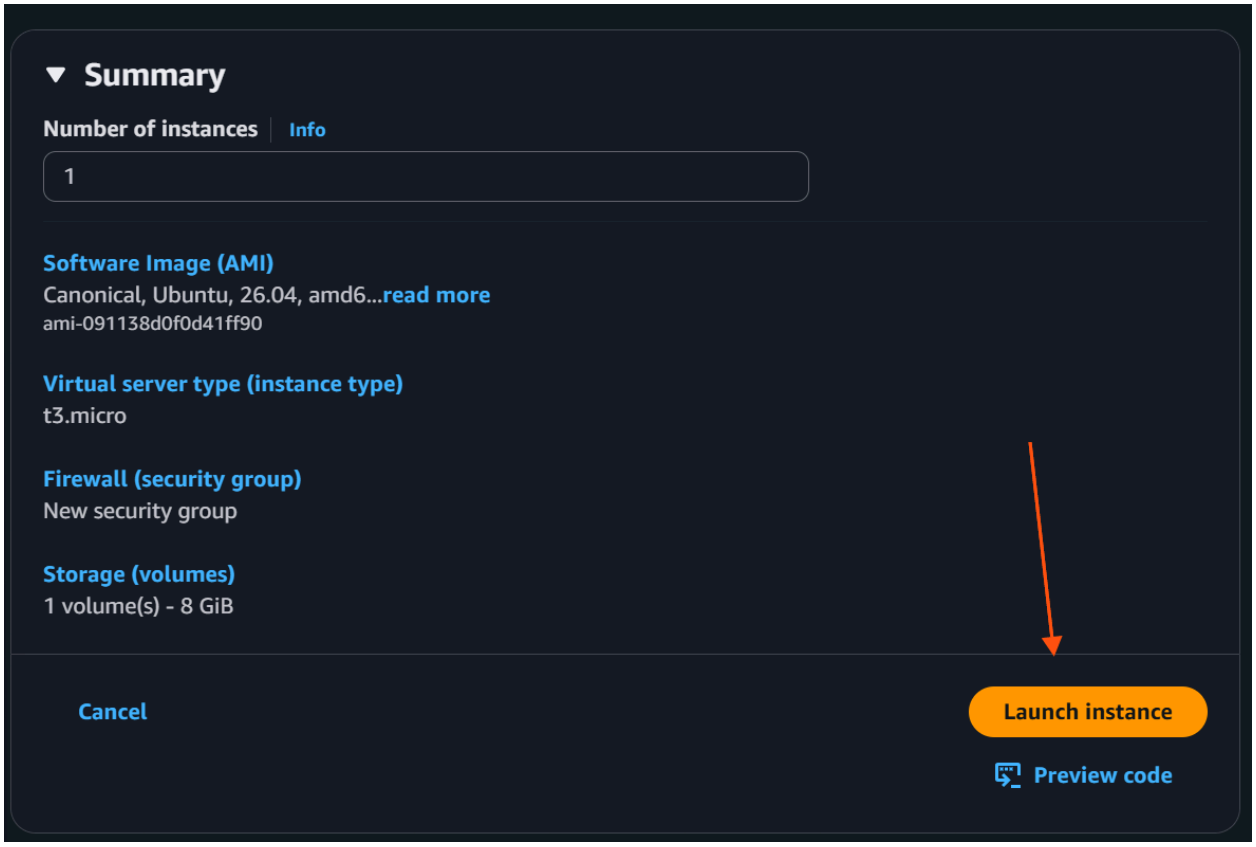



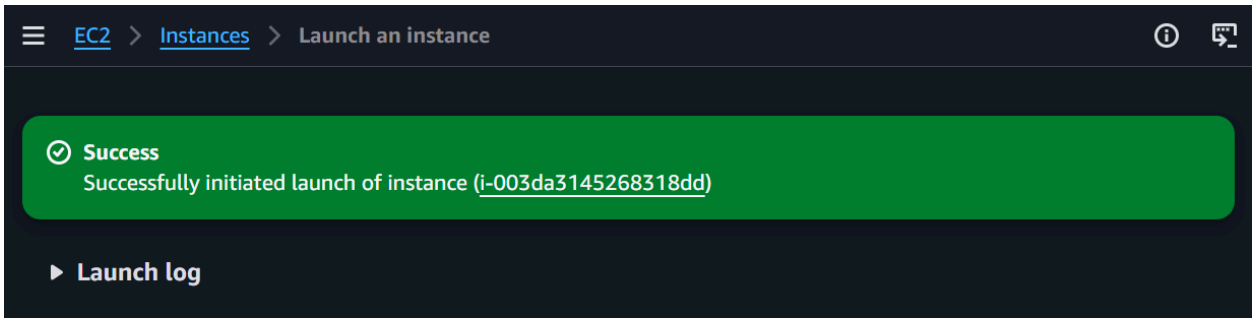




To configure the security group within network settings, start by selecting **Create security group**. Then, navigate under **Inbound Security Group Rules**. There, for **Type** select **ssh** on port **22** and for **Source type** select **My IP**.



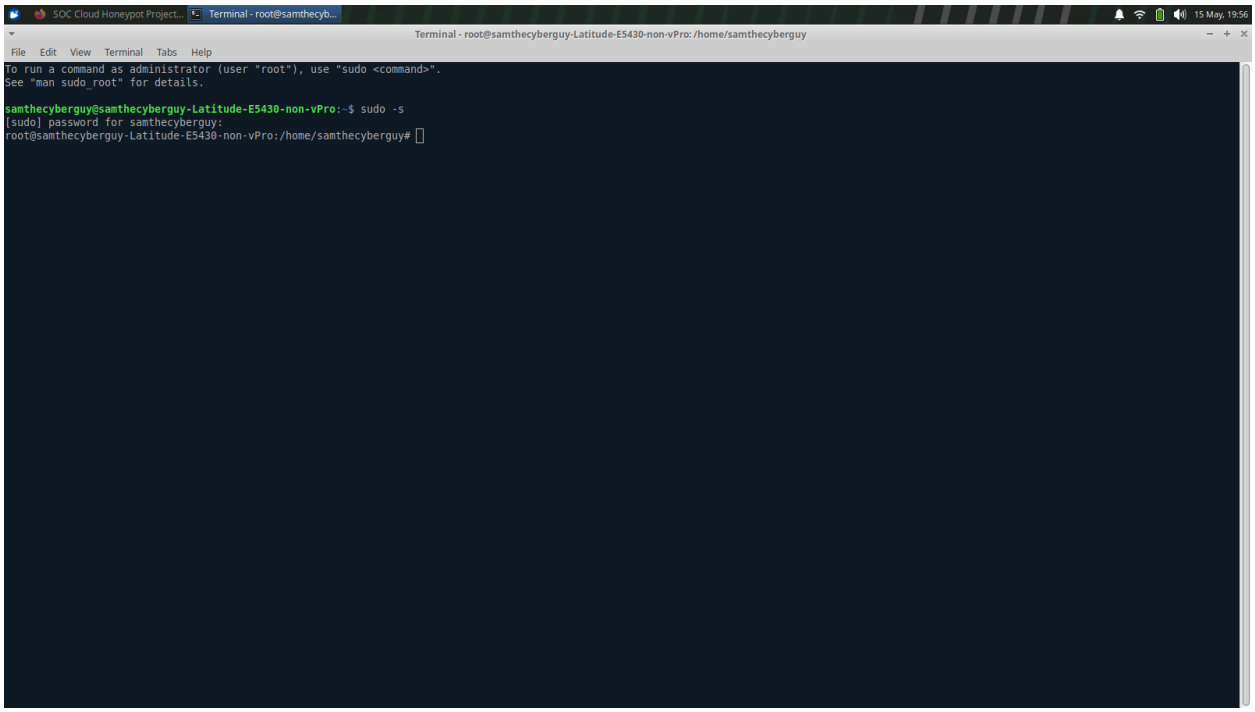

Select **Launch instance** and AWS will provision the virtual linux server, public IP, networking, and storage.


Summary
Number of instances
Info
1
Software Image (AMI)
Canonical, Ubuntu, 26.04, amd6...read more
ami-091138d0f0d41ff90
Virtual server type (instance type)
t3.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Cancel
Launch instance
Preview code

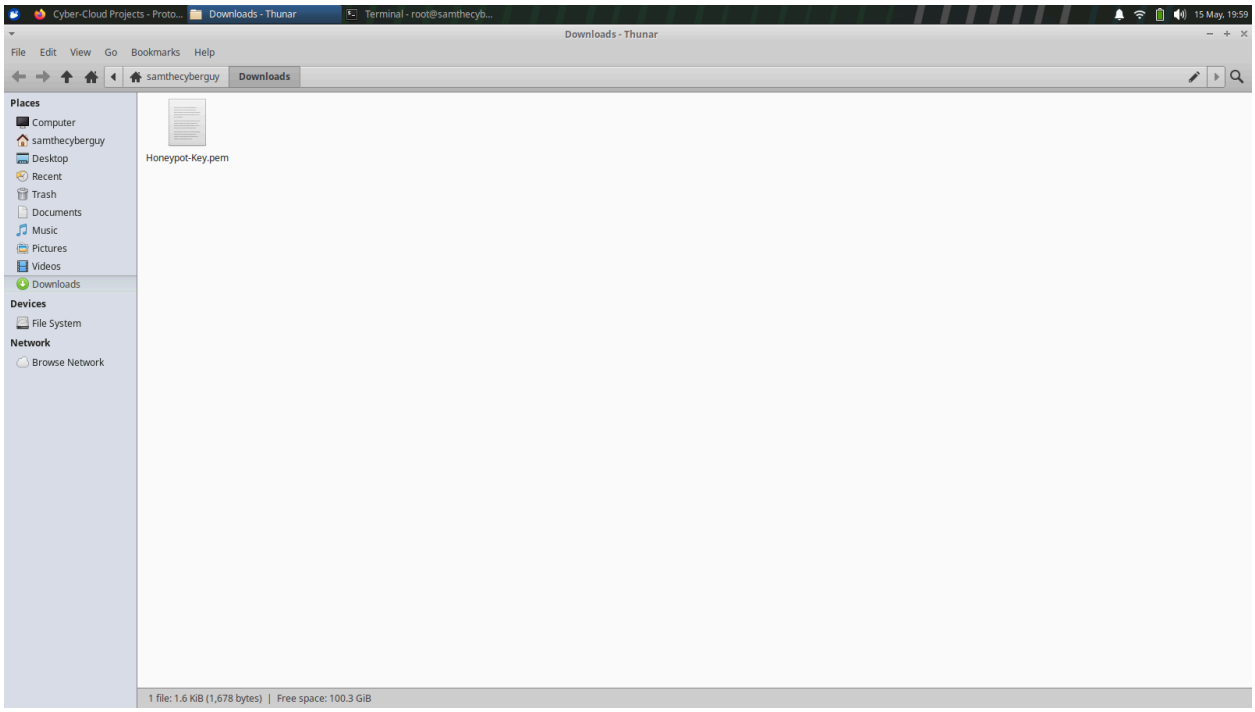

EC2
Instances
Launch an instance
Success
Successfully initiated launch of instance (i-003da3145268318dd)
Launch log

# Step 3 - Connect to EC2 Instance from Xubuntu

Open the terminal in Xubuntu.




Locate the PEM key downloaded from AWS.



Create an SSH directory and move the PEM key to it.

```
samthecyberguy@samthecyberguy-Latitude-E5430-non-vPro:~$ mkdir -p ~/.ssh
samthecyberguy@samthecyberguy-Latitude-E5430-non-vPro:~$ mv ~/home/samthecyberguy/Downloads/Honeypot-Key.pem ~/.ssh/
```

```
samthecyberguy@samthecyberguy-Latitude-E5430-non-vPro:~$ cd ~/.ssh
samthecyberguy@samthecyberguy-Latitude-E5430-non-vPro:~/.ssh$ ls
authorized_keys  Honeypot-Key.pem
```

Set the PEM key file to be readable by only the owner (you). To do this use **chmod**. SSH will reject keys with loose permissions as a security measure.

```
samthecyberguy@samthecyberguy-Latitude-E5430-non-vPro:~/.ssh$ chmod 400 Honeypot-Key.pem
samthecyberguy@samthecyberguy-Latitude-E5430-non-vPro:~/.ssh$
```

Locate the EC2 instance's public ip address.

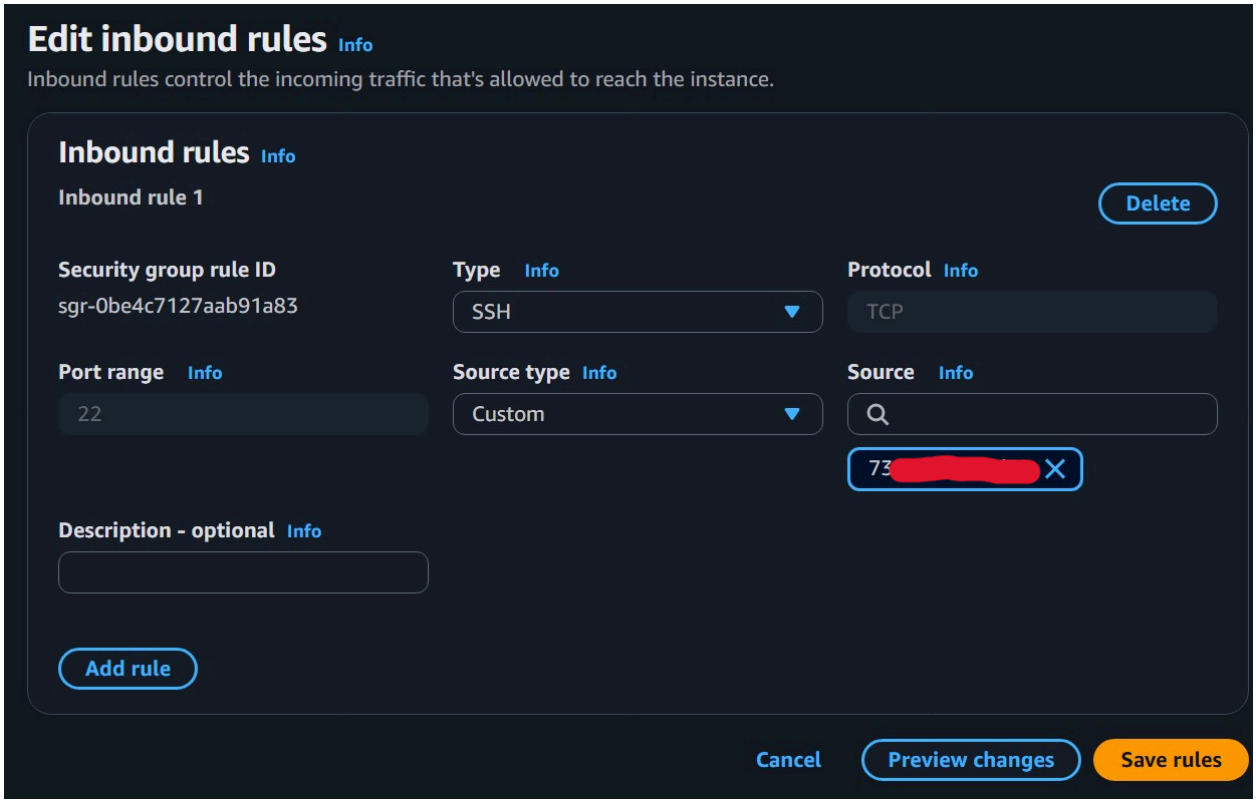

Attempt to remote into the EC2 instance via SSH. This may return a **connection timed out** error.

```
samthecyberguy@samthecyberguy-Latitude-E5430-non-vPro:~$ ssh -i ~/.ssh/Honeypot-Key.pem ubuntu@54.160.227.94

ssh: connect to host 54.160.227.94 port 22: Connection timed out
```

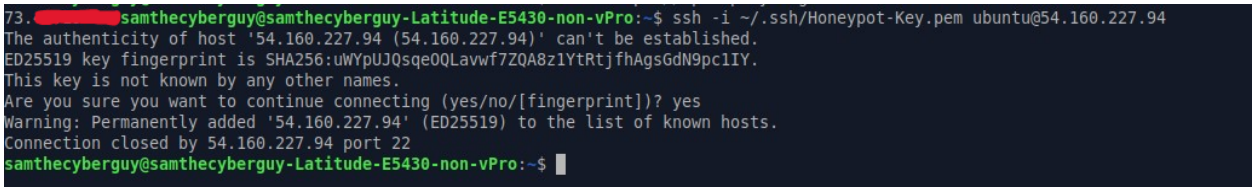
After receiving the error, use the **netcat** command to test the connection to the EC2 instance public IP.

```
samthecyberguy@samthecyberguy-Latitude-E5430-non-vPro:~$ nc -zv 54.160.227.94 22
nc: connect to 54.160.227.94 port 22 (tcp) failed: Connection timed out
```

If the connection times out, check the inbound rule on the EC2 security group. The source IP must match the machine being used to connect, not a private IP from a different machine. Update the inbound rule with the correct public IP and attempt the connection again.


Beyond network connection issues, it is possible to get an SSH handshake error.




If there is an SSH handshake error, first confirm that the PEM key on Xubuntu is the same as the PEM key assigned to the EC2 instance. Next, confirm that the AMI is Ubuntu and not another OS. Lastly, add a **-v** to the original command to give it verbose output.

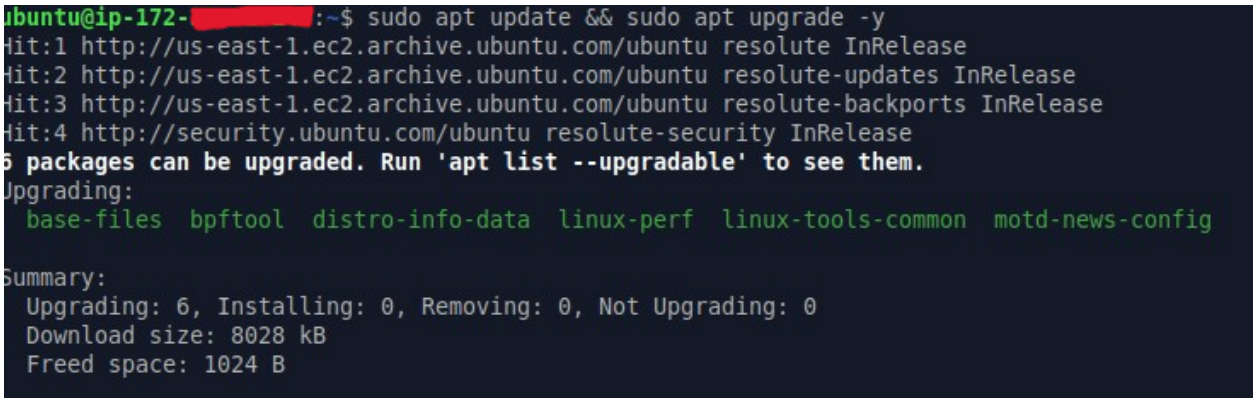
```
samthecyberguy@samthecyberguy-Latitude-E5430-non-vPro:~$ ssh -v -i ~/.ssh/Honeypot-Key.pem ubuntu@54.160.227.94
OpenSSH_9.6p1 Ubuntu-3ubuntu13.16, OpenSSL 3.0.13 30 Jan 2024
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: include /etc/ssh/ssh_config.d/*.conf matched no files
debug1: /etc/ssh/ssh_config line 21: Applying options for *
debug1: Connecting to 54.160.227.94 [54.160.227.94] port 22.
debug1: Connection established.
debug1: identity file /home/samthecyberguy/.ssh/Honeypot-Key.pem type -1
debug1: identity file /home/samthecyberguy/.ssh/Honeypot-Key.pem-cert type -1
debug1: Local version string SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.16
debug1: Remote protocol version 2.0, remote software version OpenSSH_10.2p1 Ubuntu-2ubuntu3.2
debug1: compat_banner: match: OpenSSH_10.2p1 Ubuntu-2ubuntu3.2 pat OpenSSH* compat 0x04000000
debug1: Authenticating to 54.160.227.94:22 as 'ubuntu'
debug1: load_hostkeys: fopen /home/samthecyberguy/.ssh/known_hosts2: No such file or directory
debug1: load_hostkeys: fopen /etc/ssh/ssh_known_hosts: No such file or directory
debug1: load_hostkeys: fopen /etc/ssh/ssh_known_hosts2: No such file or directory
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: algorithm: sntrup761x25519-sha512@openssh.com
debug1: kex: host key algorithm: ssh-ed25519
debug1: kex: server->client cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: kex: client->server cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: expecting SSH2_MSG_KEX_ECDH_REPLY
debug1: SSH2_MSG_KEX_ECDH_REPLY received
```

...

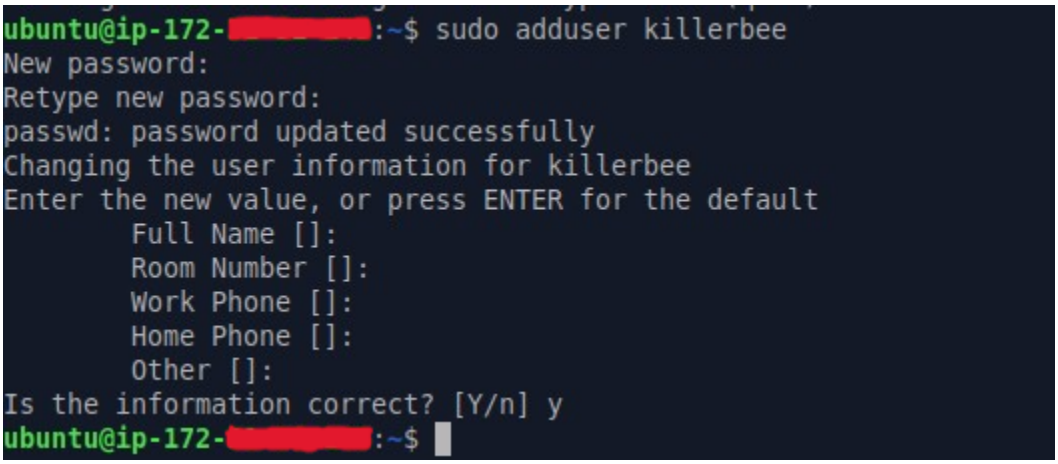
```
Welcome to Ubuntu 26.04 LTS (GNU/Linux 7.0.0-1004-aws x86_64)

 * Documentation:  https://docs.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sat May 16 19:36:28 UTC 2026

  System load:  0.0               Temperature:           -273.1 C
  Usage of /:   34.8% of 6.61GB   Processes:             114
  Memory usage: 29%               Users logged in:       0
  Swap usage:   0%                IPv4 address for ens5: 172.[redacted]

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

6 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

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



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.

ubuntu@ip-172-[redacted]:~$
```

## Step 4 - Initial Server Setup

Now that connection from the Xubuntu machine to the AWS EC2 instance has been established via SSH, update packages for the server.




Create an admin user.




Give sudo access to the admin user.

## Step 5 - Move Real SSH to Port 2222

As outlined in the introduction, real SSH access will now move from port 22 to port 2222, freeing port 22 for the Cowrie honeypot to intercept attacker traffic.

Start by opening the SSH configuration file. Then add an additional line with **port 2222** and uncomment each port line by removing the **#** character.

```
ubuntu@ip-172-[redacted]:~$ sudo nano /etc/ssh/sshd_config
ubuntu@ip-172-[redacted]:~$
```

```
# For changes to take effect, run:
#
#   systemctl daemon-reload
#   systemctl restart ssh.socket
#
#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key
```

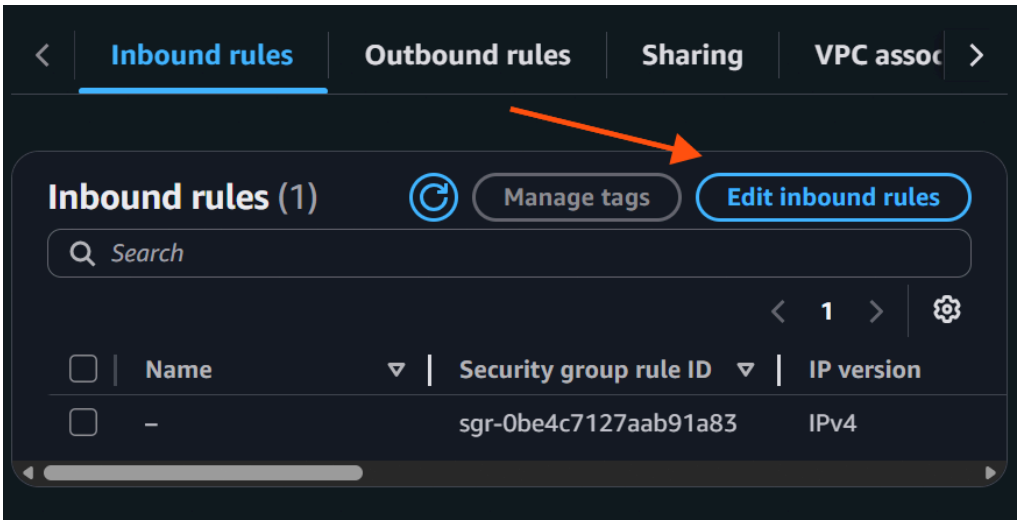
```
# For changes to take effect, run:
#
#   systemctl daemon-reload
#   systemctl restart ssh.socket
#
Port 22
Port 2222
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key
```

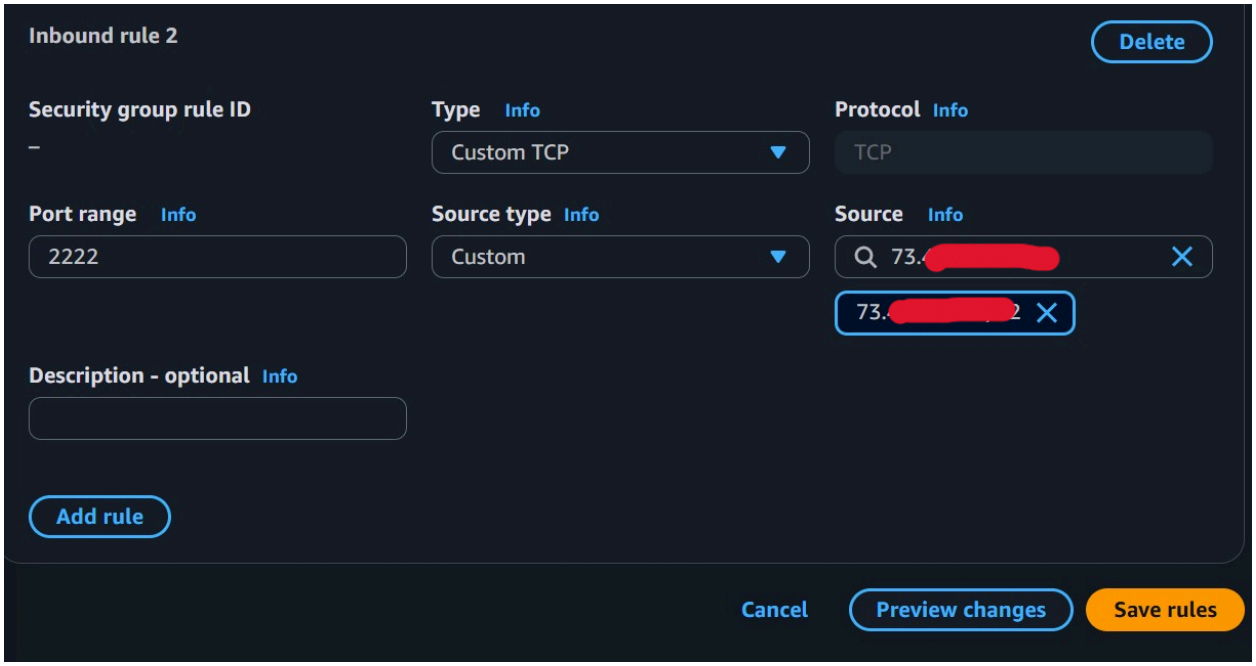
In the config file, remove **prohibit-password** and add **no** after **PermitRootLogin**. Then, uncomment that line. This makes it so root login is disabled over SSH. The root account has the highest privilege and attackers often target this account, so by disabling it, the risk of root compromise is reduced.

```
# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

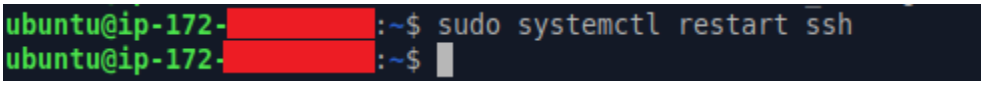
```
# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

Add a new inbound rule allowing SSH from the Xubuntu machine on port 2222.

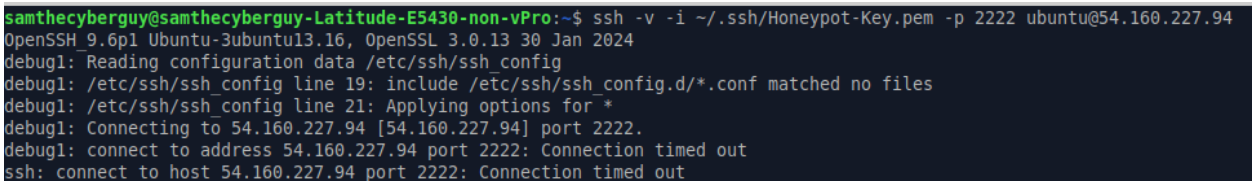




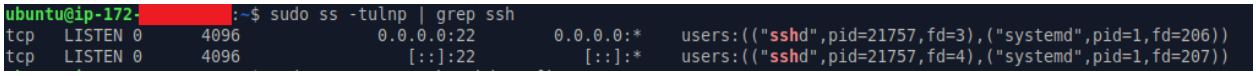

Restart SSH.




Open a new terminal window and attempt to SSH into the EC2 instance through port 2222 to test the new change. A connection timed out error might occur.




Return to the original terminal to check if port 2222 is now listening on SSH. It is possible only port 22 may still be present.




Confirm that port 2222 has been added to the SSH config file and uncommented as well.

```
# When systemd socket activation is used (the default), the socket
# configuration must be re-generated after changing Port, AddressFamily, or
# ListenAddress.
#
# For changes to take effect, run:
#
#   systemctl daemon-reload
#   systemctl restart ssh.socket
#
Port 22
Port 2222
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

If this is the case, disable **ssh.socket**. This feature overrides changes made to the SSH config file by default on Ubuntu.

```
ubuntu@ip-172-[redacted]:~$ sudo systemctl disable ssh.socket
Removed '/etc/systemd/system/ssh.service.requires/ssh.socket'.
Removed '/etc/systemd/system/sockets.target.wants/ssh.socket'.
ubuntu@ip-172-[redacted]:~$ sudo systemctl stop ssh.socket
```

Restart SSH again and verify that SSH is now listening on port 2222.

```
ubuntu@ip-172-[redacted]:~$ sudo systemctl restart ssh
ubuntu@ip-172-[redacted]:~$ sudo ss -tulnp | grep ssh
tcp   LISTEN 0      128              0.0.0.0:22        0.0.0.0:*    users:(("sshd",pid=22370,fd=8))
tcp   LISTEN 0      128              0.0.0.0:2222      0.0.0.0:*    users:(("sshd",pid=22370,fd=6))
tcp   LISTEN 0      128                 [::]:22           [::]:*    users:(("sshd",pid=22370,fd=9))
tcp   LISTEN 0      128                 [::]:2222         [::]:*    users:(("sshd",pid=22370,fd=7))
```

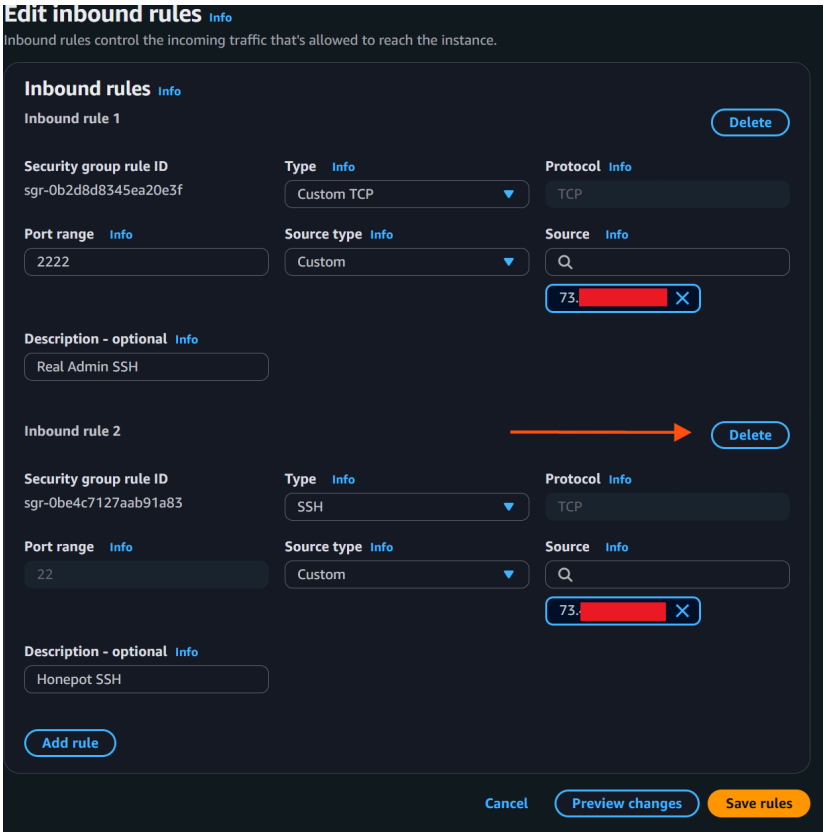
For now, leave SSH on port 22 and port 2222. Later, once Cowrie is configured to be on port 22, port 22 will be removed from the SSH config file.

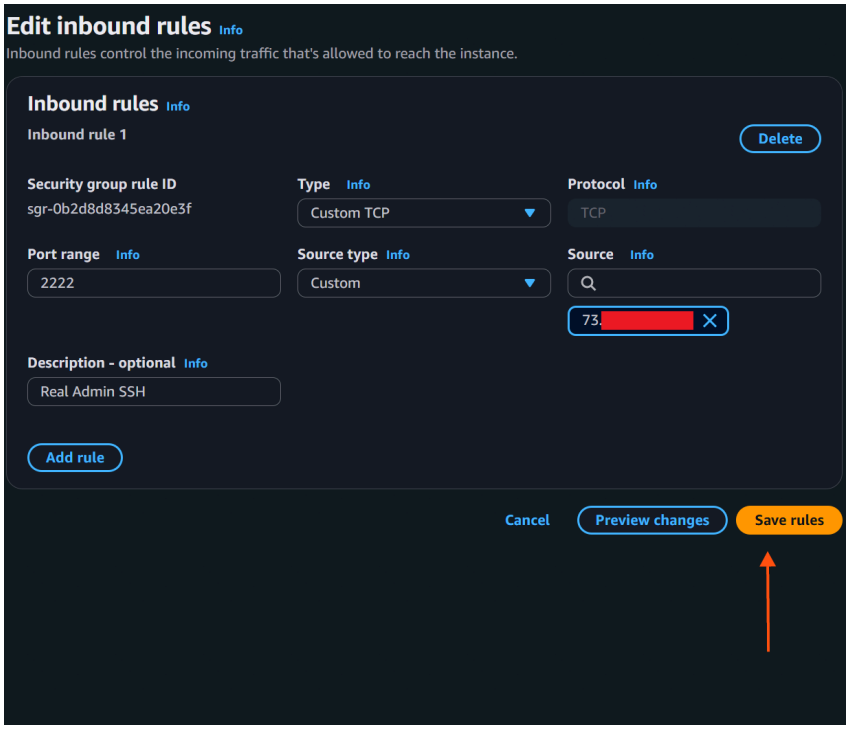
Navigate to the new terminal and attempt to SSH into the EC2 instance through port 2222.

```
samthecyberguy@samthecyberguy-Latitude-E5430-non-vPro:~$ ssh -v -i ~/.ssh/Honeypot-Key.pem -p 2222 ubuntu@54.160.227.94
OpenSSH_9.6p1 Ubuntu-3ubuntu13.16, OpenSSL 3.0.13 30 Jan 2024
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: include /etc/ssh/ssh_config.d/*.conf matched no files
debug1: /etc/ssh/ssh_config line 21: Applying options for *
debug1: Connecting to 54.160.227.94 [54.160.227.94] port 2222.
debug1: Connection established.
```

To finalize the transfer of SSH from port 22 to port 2222, remove port 22 admin access by going into the EC2 instance Security Group settings and editing the inbound rule. The intention is to stop using port 22 for legitimate administration, laying the

groundwork for Cowrie honeypot to use port 22, while port 2222 will be used for legitimate SSH administration.



# Step 6 - Install Cowrie Honeypot

The goal is to install and configure Cowrie Honeypot on the AWS Ubuntu server so it can simulate a fake SSH server to capture attacker activity.

To begin, connect to the EC2 instance in the Xubuntu terminal.

```
samthecyberguy@samthecyberguy-Latitude-E5430-non-vPro:~$ ssh -v -i ~/.ssh/Honeypot-Key.pem -p 2222 ubuntu@54.160.227.94
OpenSSH_9.6p1 Ubuntu-3ubuntu13.16, OpenSSL 3.0.13 30 Jan 2024
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: include /etc/ssh/ssh_config.d/*.conf matched no files
debug1: /etc/ssh/ssh_config line 21: Applying options for *
debug1: Connecting to 54.160.227.94 [54.160.227.94] port 2222.
debug1: Connection established.
```

Update packages and install dependencies to prepare for the Cowrie installation.

```
ubuntu@ip-172-[redacted]:~$ sudo apt update && sudo apt upgrade -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu resolute InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu resolute-updates InRelease [136 kB]
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu resolute-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu resolute-security InRelease [136 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu resolute-updates/main amd64 Components [2844 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu resolute-updates/universe amd64 Components [20.6 kB]
Get:7 http://security.ubuntu.com/ubuntu resolute-security/main amd64 Components [2844 B]
Get:8 http://security.ubuntu.com/ubuntu resolute-security/universe amd64 Components [20.6 kB]
Fetched 318 kB in 0s (1841 kB/s)
All packages are up to date.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
ubuntu@ip-172-[redacted]:~$
```

```
ubuntu@ip-172-[redacted]:~$ sudo apt install -y \
git \
python3-venv \
python3-dev \
libssl-dev \
libffi-dev \
build-essential \
libpython3-dev \
authbind \
virtualenv
git is already the newest version (1:2.53.0-1ubuntu1).
git set to manually installed.
Installing:
  authbind  build-essential  libffi-dev  libpython3-dev  libssl-dev  python3-dev  python3-venv  virtualenv

Installing dependencies:
  bzip2                  g++                    gcc-15-x86-64-linux-gnu    libc6-dev              libgomp1          libquadmath0       manpages-dev            python3.14-dev
  cpp                    g++-15                 gcc-x86-64-linux-gnu       libcc1-0               libhwasan0        libstdc++-15-dev   python3-distlib         python3.14-venv
  cpp-15                 g++-15-x86-64-linux-gnu libalgorithm-diff-perl     libdpkg-perl           libisl23          libtsan2           python3-filelock        rpcsvc-proto
  cpp-15-x86-64-linux-gnu g++-x86-64-linux-gnu  libalgorithm-diff-xs-perl  libexpat1-dev          libitm1           libubsan1          python3-pip-whl         zlib1g-dev
  cpp-x86-64-linux-gnu   gcc                    libalgorithm-merge-perl    libfakeroot            liblsan0          linux-libc-dev     python3-platformdirs
  dpkg-dev               gcc-15                 libasan8                   libfile-fcntllock-perl libmpc3           lto-disabled-list  python3-setuptools-whl
  fakeroot               gcc-15-base            libc-dev-bin               libgcc-15-dev          libpython3.14-dev make               python3-virtualenv

Suggested packages:
  bzip2-doc  gcc-15-locales  debian-keyring            g++-multilib     gcc-15-doc    autoconf  libtool  bison  gcc-doc          gdb-x86-64-linux-gnu  glibc-doc  libssl-doc        make-doc
  cpp-doc    cpp-15-doc      debian-tag2upload-keyring g++-15-multilib  gcc-multilib  automake  flex     gdb    gcc-15-multilib  libc-devtools         bzr        libstdc++-15-doc

Summary:
  Upgrading: 0, Installing: 61, Removing: 0, Not Upgrading: 0
  Download size: 93.6 MB
  Space needed: 329 MB / 4608 MB available
```

Since Cowrie will be exposed to attackers, create a Cowrie user without root privileges. Using a basic user account is a security measure that can prevent potential root access by an attacker and a compromise to the EC2 instance.

After creating the Cowrie account, login to it.

```
ubuntu@ip-172-[redacted]:~$ sudo adduser --disabled-password cowrie
Changing the user information for cowrie
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
ubuntu@ip-172-[redacted]:~$ sudo su - cowrie
cowrie@ip-172-[redacted]:~$
```

Clone the Cowrie repository from GitHub.

```
cowrie@ip-172-[redacted]:~$ git clone http://github.com/cowrie/cowrie
Cloning into 'cowrie'...
warning: redirecting to https://github.com/cowrie/cowrie/
remote: Enumerating objects: 22049, done.
remote: Counting objects: 100% (644/644), done.
remote: Compressing objects: 100% (381/381), done.
remote: Total 22049 (delta 433), reused 268 (delta 263), pack-reused 21405 (from 3)
Receiving objects: 100% (22049/22049), 11.93 MiB | 28.95 MiB/s, done.
Resolving deltas: 100% (15191/15191), done.
cowrie@ip-172-[redacted]:~$
```

Enter the Cowrie directory.

```
cowrie@ip-172-[redacted]:~$ cd cowrie
cowrie@ip-172-[redacted]:~/cowrie$
```

Create a python virtual environment and then enter the environment so Cowrie uses its own python packages and not system wide packages, as that could cause accidental conflicts.

```
cowrie@ip-172-[redacted]:~/cowrie$ python3 -m venv cowrie-env
cowrie@ip-172-[redacted]:~/cowrie$ source cowrie-env/bin/activate
(cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$ ^C
(cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$
```

Install Cowrie requirements, dependencies, and Cowrie as an editable python package with the launcher executable.

```
(cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$ pip install --upgrade pip
Requirement already satisfied: pip in ./cowrie-env/lib/python3.14/site-packages (25.1.1)
Collecting pip
  Downloading pip-26.1.1-py3-none-any.whl.metadata (4.6 kB)
Downloading pip-26.1.1-py3-none-any.whl (1.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 42.3 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 25.1.1
    Uninstalling pip-25.1.1:
      Successfully uninstalled pip-25.1.1
Successfully installed pip-26.1.1
```

```
(cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$ pip install --upgrade -r requirements.txt
WARNING: Cache entry deserialization failed, entry ignored
Collecting attrs==26.1.0 (from -r requirements.txt (line 5))
  Downloading attrs-26.1.0-py3-none-any.whl.metadata (8.8 kB)
Collecting bcrypt==5.0.0 (from -r requirements.txt (line 6))
  Downloading bcrypt-5.0.0-cp39-abi3-manylinux_2_34_x86_64.whl.metadata (10 kB)
Collecting cryptography==48.0.0 (from -r requirements.txt (line 7))
  Downloading cryptography-48.0.0-cp311-abi3-manylinux_2_34_x86_64.whl.metadata (4.3 kB)
Collecting hyperlink==21.0.0 (from -r requirements.txt (line 8))
  Downloading hyperlink-21.0.0-py2.py3-none-any.whl.metadata (1.5 kB)
```

...

```
Downloading treq-25.5.0-py3-none-any.whl (77 kB)
Downloading twisted-26.4.0-py3-none-any.whl (3.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.2/3.2 MB 143.2 MB/s  0:00:00
Downloading charset_normalizer-3.4.7-cp314-cp314-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl (215 kB)
Downloading pyasn1-0.6.3-py3-none-any.whl (83 kB)
Downloading appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
Downloading automat-25.4.16-py3-none-any.whl (42 kB)
Downloading certifi-2026.4.22-py3-none-any.whl (135 kB)
Downloading cffi-2.0.0-cp314-cp314-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (219 kB)
Downloading constantly-23.10.4-py3-none-any.whl (13 kB)
Downloading incremental-24.11.0-py3-none-any.whl (21 kB)
Downloading pyopenssl-26.2.0-py3-none-any.whl (55 kB)
Downloading typing_extensions-4.15.0-py3-none-any.whl (44 kB)
Downloading zope_interface-8.4-cp314-cp314-manylinux1_x86_64.manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_5_x86_64.whl (269 kB)
Downloading multipart-1.3.1-py3-none-any.whl (15 kB)
Downloading pycparser-3.0-py3-none-any.whl (48 kB)
Installing collected packages: appdirs, zope-interface, urllib3, typing-extensions, tftpy, pycparser, pyasn1, packaging, multipart, idna, constantly,
ttrs, requests, pyasn1_modules, incremental, hyperlink, cffi, twisted, cryptography, service_identity, pyopenssl, treq
Successfully installed appdirs-1.4.4 attrs-26.1.0 automat-25.4.16 bcrypt-5.0.0 certifi-2026.4.22 cffi-2.0.0 charset_normalizer-3.4.7 constantly-23.10
4 incremental-24.11.0 multipart-1.3.1 packaging-26.2 pyasn1-0.6.3 pyasn1_modules-0.4.2 pycparser-3.0 pyopenssl-26.2.0 requests-2.34.0 service_identit
typing-extensions-4.15.0 urllib3-2.7.0 zope-interface-8.4
(cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$
```

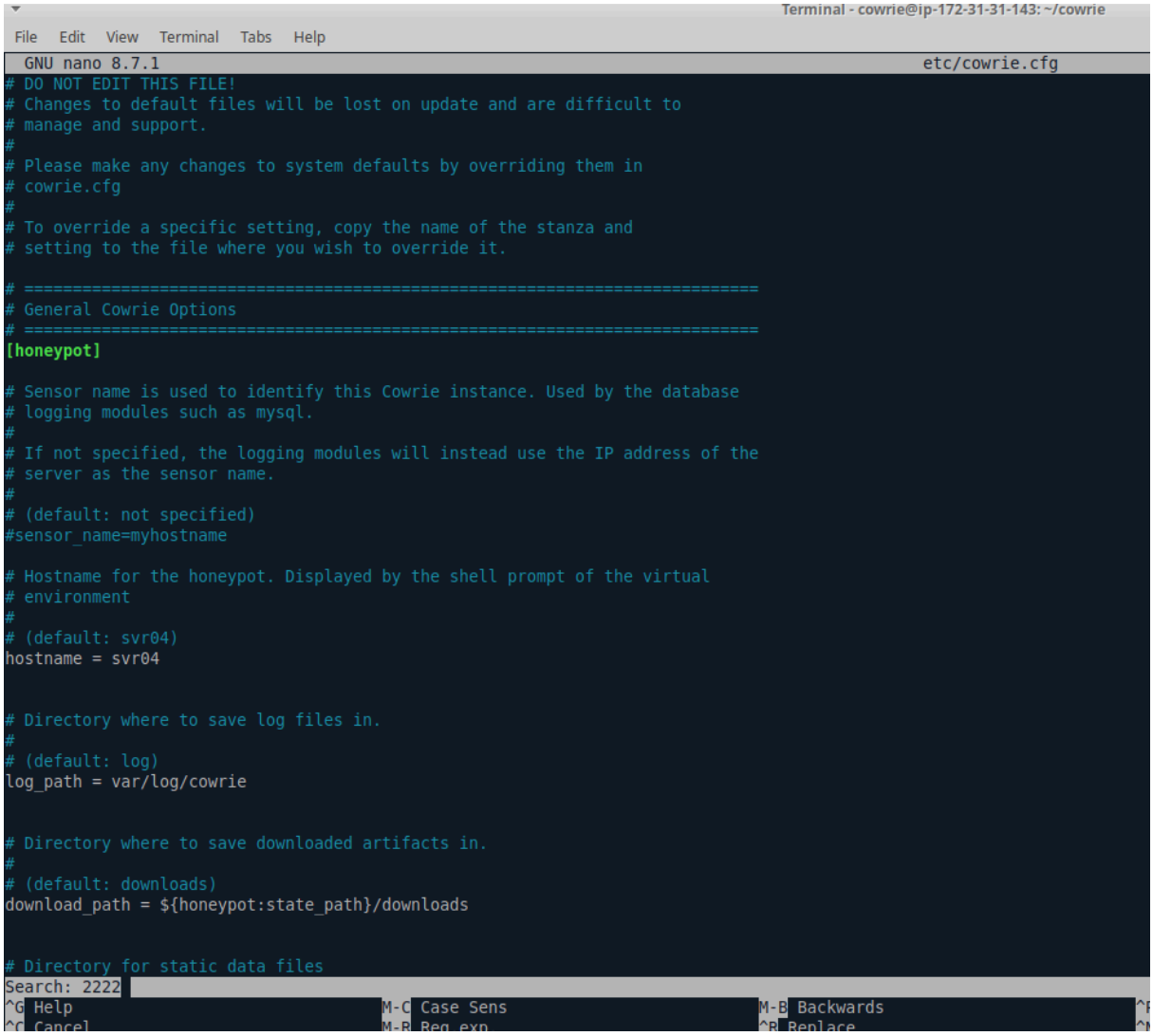
```
(cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$ pip install -e .
Obtaining file:///home/cowrie/cowrie
  Installing build dependencies ... done
  Checking if build backend supports build_editable ... done
  Getting requirements to build editable ... done
  Preparing editable metadata (pyproject.toml) ... done
Requirement already satisfied: attrs>=26.1.0 in ./cowrie-env/lib/python3.14/site-packages (from cowrie==2.9.19) (26.1.0)
Requirement already satisfied: bcrypt>=5.0.0 in ./cowrie-env/lib/python3.14/site-packages (from cowrie==2.9.19) (5.0.0)
Requirement already satisfied: cryptography>=48.0.0 in ./cowrie-env/lib/python3.14/site-packages (from cowrie==2.9.19) (48.0.0)
Requirement already satisfied: hyperlink>=21.0.0 in ./cowrie-env/lib/python3.14/site-packages (from cowrie==2.9.19) (21.0.0)
Requirement already satisfied: idna>=3.14 in ./cowrie-env/lib/python3.14/site-packages (from cowrie==2.9.19) (3.14)
Requirement already satisfied: packaging>=26.2 in ./cowrie-env/lib/python3.14/site-packages (from cowrie==2.9.19) (26.2)
Requirement already satisfied: pyasn1_modules>=0.4.2 in ./cowrie-env/lib/python3.14/site-packages (from cowrie==2.9.19) (0.4.2)
Requirement already satisfied: requests>=2.34.0 in ./cowrie-env/lib/python3.14/site-packages (from cowrie==2.9.19) (2.34.0)
Requirement already satisfied: service_identity>=24.2.0 in ./cowrie-env/lib/python3.14/site-packages (from cowrie==2.9.19) (24.2.0)
Requirement already satisfied: tftpy>=0.8.7 in ./cowrie-env/lib/python3.14/site-packages (from cowrie==2.9.19) (0.8.7)
Requirement already satisfied: treq>=25.5.0 in ./cowrie-env/lib/python3.14/site-packages (from cowrie==2.9.19) (25.5.0)
Requirement already satisfied: twisted>=26.4.0 in ./cowrie-env/lib/python3.14/site-packages (from twisted[conch]>=26.4.0->cowrie==2.9.19) (26.4.0)
Requirement already satisfied: urllib3>=2.7.0 in ./cowrie-env/lib/python3.14/site-packages (from cowrie==2.9.19) (2.7.0)
Requirement already satisfied: cffi>=2.0.0 in ./cowrie-env/lib/python3.14/site-packages (from cryptography>=48.0.0->cowrie==2.9.19) (2.0.0)
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in ./cowrie-env/lib/python3.14/site-packages (from pyasn1_modules>=0.4.2->cowrie==2.9.19) (0.6.3)
Requirement already satisfied: charset_normalizer<4,>=2 in ./cowrie-env/lib/python3.14/site-packages (from requests>=2.34.0->cowrie==2.9.19) (3.4.7)
Requirement already satisfied: certifi>=2023.5.7 in ./cowrie-env/lib/python3.14/site-packages (from requests>=2.34.0->cowrie==2.9.19) (2026.4.22)
Requirement already satisfied: incremental>=24.7.2 in ./cowrie-env/lib/python3.14/site-packages (from treq>=25.5.0->cowrie==2.9.19) (24.11.0)
Requirement already satisfied: multipart in ./cowrie-env/lib/python3.14/site-packages (from treq>=25.5.0->cowrie==2.9.19) (1.3.1)
Requirement already satisfied: typing-extensions>=3.10.0 in ./cowrie-env/lib/python3.14/site-packages (from treq>=25.5.0->cowrie==2.9.19) (4.15.0)
Requirement already satisfied: automat>=24.8.0 in ./cowrie-env/lib/python3.14/site-packages (from twisted==26.4.0->twisted[conch]==26.4.0->cowrie==2.9.19) (25.4.16)
Requirement already satisfied: constantly>=15.1 in ./cowrie-env/lib/python3.14/site-packages (from twisted==26.4.0->twisted[conch]==26.4.0->cowrie==2.9.19) (23.10.4)
Requirement already satisfied: zope-interface>=5 in ./cowrie-env/lib/python3.14/site-packages (from twisted==26.4.0->twisted[conch]==26.4.0->cowrie==2.9.19) (8.4)
Requirement already satisfied: appdirs>=1.4.0 in ./cowrie-env/lib/python3.14/site-packages (from twisted[conch]==26.4.0->cowrie==2.9.19) (1.4.4)
Requirement already satisfied: pycparser in ./cowrie-env/lib/python3.14/site-packages (from cffi>=2.0.0->cryptography>=48.0.0->cowrie==2.9.19) (3.0)
Requirement already satisfied: pyopenssl>=25.2.0 in ./cowrie-env/lib/python3.14/site-packages (from twisted[tls]>=22.10.0->treq>=25.5.0->cowrie==2.9.19) (26.2.0)
Building wheels for collected packages: cowrie
  Building editable for cowrie (pyproject.toml) ... done
  Created wheel for cowrie: filename=cowrie-2.9.19-0.editable-py3-none-any.whl size=5973 sha256=9e5c6d89ec16ed21900471c43f7990445e3a3bfa687d874b20a30a1ac847b5d4
  Stored in directory: /tmp/pip-ephem-wheel-cache-vhogn0w4/wheels/1f/6f/90/f26429e723aad6d0b0eadc48ac2d1b95ce70c68a199165b764
Successfully built cowrie
Installing collected packages: cowrie
Successfully installed cowrie-2.9.19
```

## Step 7 - Configure Cowrie

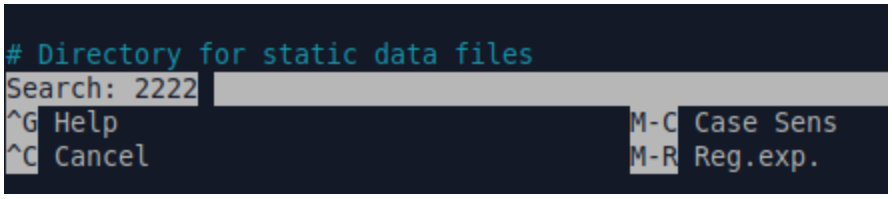
Now, it is time to configure Cowrie so it is assigned to port 22.

Copy this template for the Cowrie config file and then open the file to edit it.

```
cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$ cp etc/cowrie.cfg.dist etc/cowrie.cfg
cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$ nano etc/cowrie.cfg
```

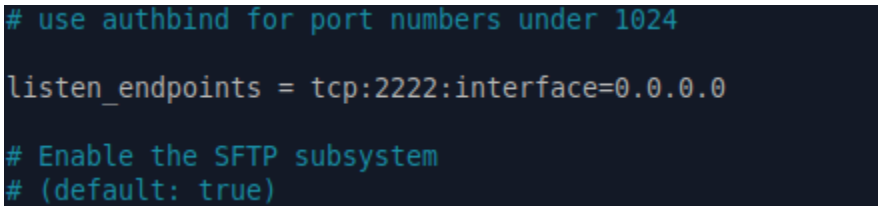


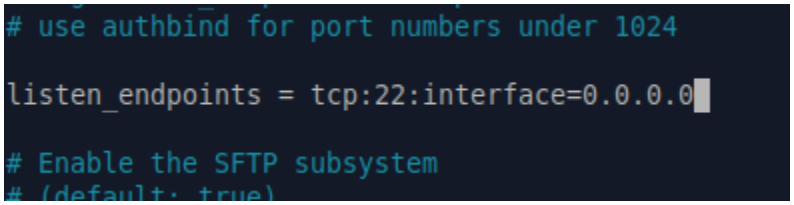

Press **Ctrl + W** and search for **2222**.




Locate the line *listen_endpoints = tcp:2222:interface=0.0.0.0*.

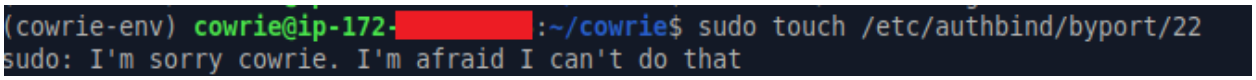
By default Cowrie has SSH on port 2222. In the file, replace **2222** with **22** (Cowrie's SSH port). Then save it and exit the file.

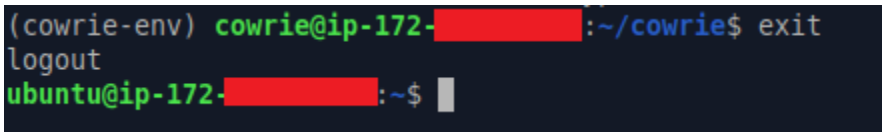


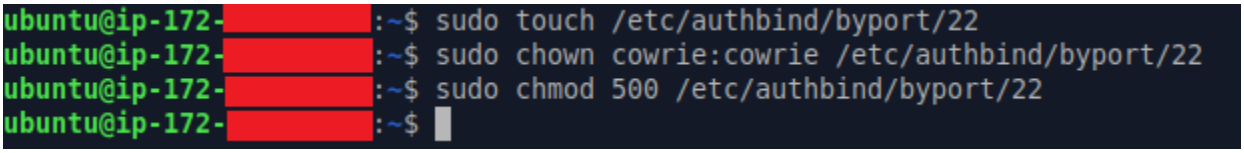




By default, Linux blocks non-root users from using ports below 1024, but earlier Cowrie was set up to run as a non-root user for security purposes.

In order to bypass this, configure **authbind**, which was installed earlier, to allow Cowrie to use port 22.




Within the Cowrie folder and the python virtual environment, start up Cowrie and check the status to make sure it is running.

```
(cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$ cowrie start

Join the Cowrie community at: https://www.cowrie.org/slack/

Starting cowrie: [twistd --umask=0022 --pidfile /home/cowrie/cowrie/var/run/cowrie.pid --logger cowrie.python.logfile.logger cowrie]...
(cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$ cowrie status
cowrie is running (PID: 32910).
(cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$
```

Attempt to see if Cowrie is listening on port 22. It is possible to receive an access error, which is actually expected because earlier Cowrie was set as a non-root user.

```
(cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$ sudo ss -tulnp | grep 2222
sudo: I'm sorry cowrie. I'm afraid I can't do that
(cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$ exit
logout
```

SSH is still on both port 22 and port 2222. Now that Cowrie has been set up on port 22, remove SSH from port 22 so that it is only on port 2222.

```
ubuntu@ip-172-[redacted]:~$ sudo ss -tulnp | grep 22
tcp   LISTEN 0      128          0.0.0.0:22        0.0.0.0:*    users:(("sshd",pid=22370,fd=8))
tcp   LISTEN 0      128          0.0.0.0:2222      0.0.0.0:*    users:(("sshd",pid=22370,fd=6))
tcp   LISTEN 0      128             [::]:22           [::]:*    users:(("sshd",pid=22370,fd=9))
tcp   LISTEN 0      128             [::]:2222         [::]:*    users:(("sshd",pid=22370,fd=7))
ubuntu@ip-172-[redacted]:~$
```

To do that, start by opening the SSH config file and delete **port 22** from it.

```
#
Port 22
Port 2222
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

```
#

Port 2222
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

Restart SSH.

```
ubuntu@ip-172-[redacted]:~$ sudo systemctl restart ssh
ubuntu@ip-172-[redacted]:~$
```

Confirm that SSH is no longer listening on port 22 and instead only on port 2222.

```
ubuntu@ip-172-[redacted]:~$ sudo ss -tulnp | grep 22
tcp   LISTEN 0      128              0.0.0.0:2222      0.0.0.0:*    users:(("sshd",pid=33058,fd=6))
tcp   LISTEN 0      128                 [::]:2222         [::]:*    users:(("sshd",pid=33058,fd=7))
ubuntu@ip-172-[redacted]:~$
```

Log into the Cowrie user and open the Cowrie config file, confirming that only port 22 is present.

```
# For both IPv4 and IPv6: listen_endpoints = tcp6:2222:1
# Listening on multiple endpoints is supported with a si
# e.g listen_endpoints = "tcp:2222:interface=0.0.0.0 tcp
# use authbind for port numbers under 1024

listen_endpoints = tcp:22:interface=0.0.0.0

# Enable the SFTP subsystem
# (default: true)
sftp_enabled = true
```

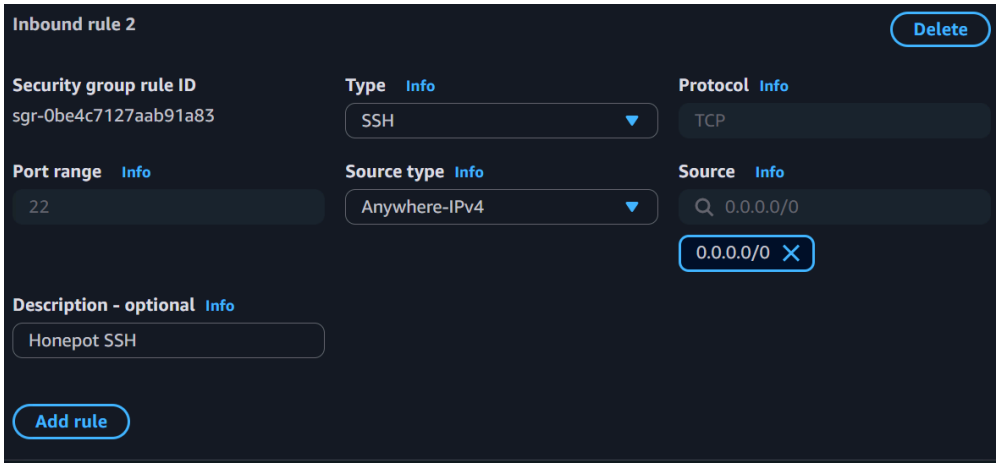
Enter Cowrie python virtual environment and stop Cowrie. Then start Cowrie to initiate the change.

```
cowrie@ip-172-[redacted]:~/cowrie$ source cowrie-env/bin/activate
(cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$ cowrie stop
Stopping cowrie...
(cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$ cowrie start
Starting cowrie: [twistd --umask=0022 --pidfile /home/cowrie/cowrie/var/run/cowrie.pid --logger cowrie.python.logfile.logger cowrie]...
```

Log out of the Cowrie user and confirm that port 22 is assigned to Cowrie and port 2222 to SSH. The user of port 22 should be **twistd**, which is the framework which Cowrie runs on.

```
(cowrie-env) cowrie@ip-172-[redacted]:~/cowrie$ exit
logout
ubuntu@ip-172-[redacted]:~$ sudo ss -tulnp | grep 22
tcp   LISTEN 0      50               0.0.0.0:22        0.0.0.0:*    users:(("twistd",pid=33243,fd=12))
tcp   LISTEN 0      128              0.0.0.0:2222      0.0.0.0:*    users:(("sshd",pid=33058,fd=6))
tcp   LISTEN 0      128                 [::]:2222         [::]:*    users:(("sshd",pid=33058,fd=7))
```

The current EC2 security group had the source IP for the inbound rule for port 22 set to the Xubuntu public IP. In order to allow attacks to begin, set the source IP for that inbound rule to **0.0.0.0/0** (anywhere). This would be unsafe in a normal SSH server, but since the real SSH admin server was transferred to port 2222 and the Cowrie honeypot was configured with non root access to port 22, it is safe. Port 22 has been properly isolated and secured.

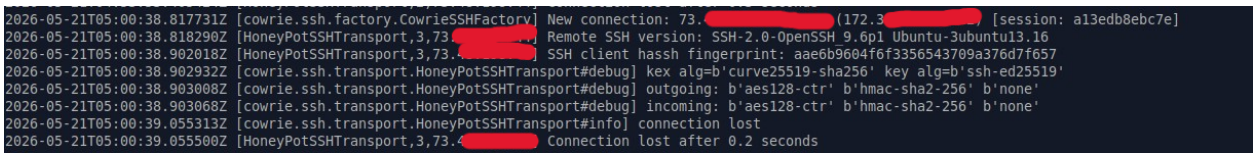




Open a fresh terminal window and attempt to SSH into the EC2 instance via port 22. If properly configured, it should return a **Host Key Verification Failed** error.

```
samthecyberguy@samthecyberguy-Latitude-E5430-non-vPro:~$ ssh -p 22 ubuntu@54.160.227.94
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:CwZjKmCi6jd+ZKg2UNPGv+zBlWomqa1feJqwRAj4gs4.
Please contact your system administrator.
Add correct host key in /home/samthecyberguy/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/samthecyberguy/.ssh/known_hosts:3
  remove with:
  ssh-keygen -f '/home/samthecyberguy/.ssh/known_hosts' -R '54.160.227.94'
Host key for 54.160.227.94 has changed and you have requested strict checking.
Host key verification failed.
samthecyberguy@samthecyberguy-Latitude-E5430-non-vPro:~$
```

Return back to the original terminal, logged in as the Cowrie user, and check the attack log.

The SSH connection attempt that was just made will be reflected in the log. The IP address of the attacker (you testing it in this case) will be visible.


Cowrie honeypot is now set up and running as intended.

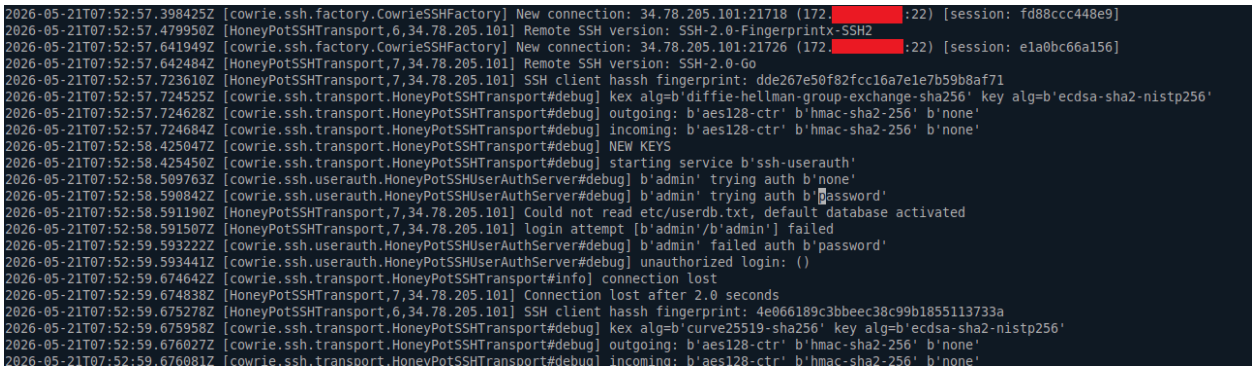
## Step 8 - Cowrie Log Analysis

Leave Cowrie and the EC2 instance running for a day.

Looking at the logs captured below, it appears that attacker interaction with the honeypot began 2 hours after turning on the honeypot for the first time. From that moment until the end of the 24 hour period, there were hundreds of interactions from various IPs using various commands.

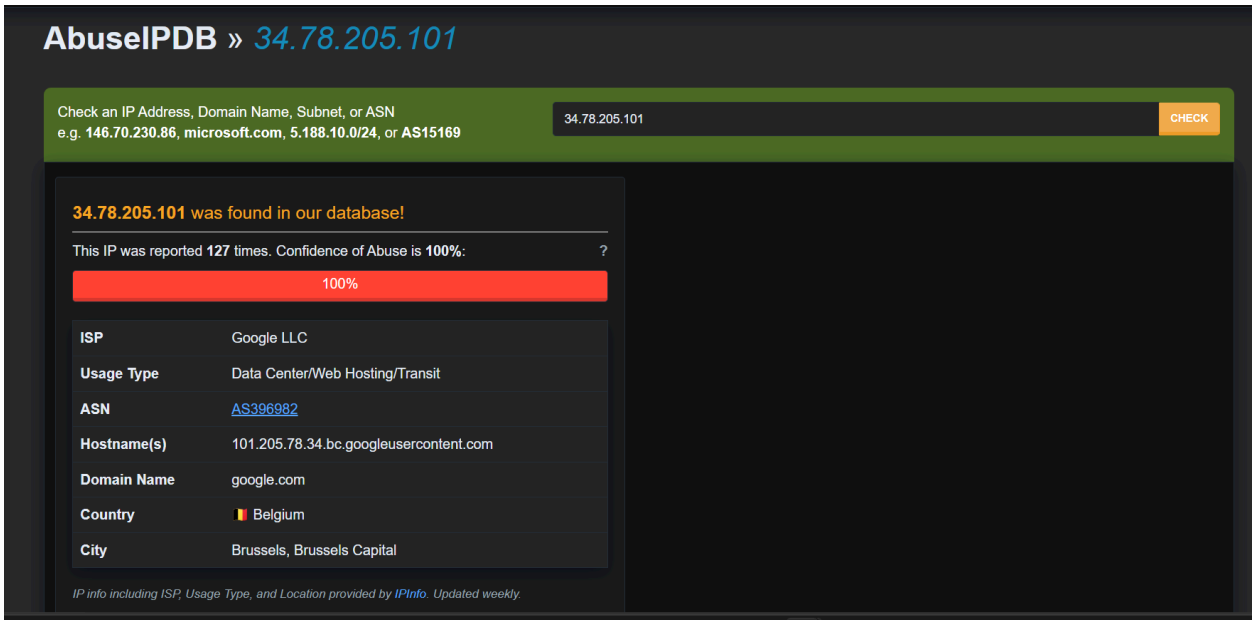
This is an example of one attack session from the logs.

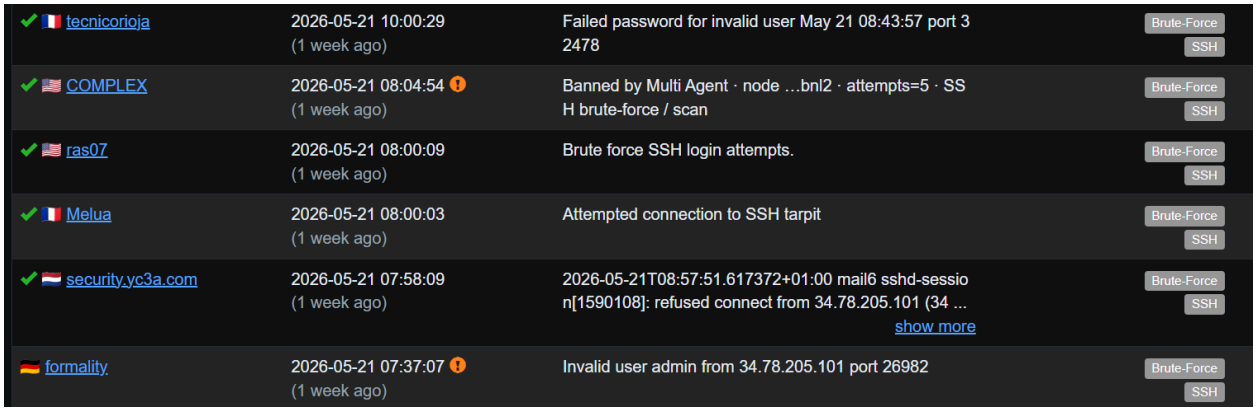
What we see below is an automated SSH brute force scanner or bot attempting to authenticate to the honeypot using default admin credentials.




The attacker's IP address is 34.78.205.101. After looking it up in AbuseIPDB, it has been reported 127 times and is confirmed to be a malicious IP that is known for brute forcing into SSH servers.

```
New connection: 34.78.205.101:21726
```




The SSH client revealed itself and shows the tools the attacker used to scan the honeypot. **SSH-2.0-Go** indicates that the tool was written in Go, which is common for automated scanners and brute force tools. **SSH-2.0-Fingerprintx-SSH2** indicates the attacker may be using the tool [Fingerprintx] which checks what kind of SSH service the target is running.

```
 Remote SSH version: SSH-2.0-Fingerprintx-SSH2
New connection: 34.78.205.101:21726 (172.[redacted]
 Remote SSH version: SSH-2.0-Go
```

Cowrie captured the HASSH, which is like the digital fingerprint for SSH clients. This can be used to correlate this session with similar SSH scanning or brute force activity across other honeypot logs.

```
SSH client hassh fingerprint: dde267e50f82fcc16a7e1e7b59b8af71
```

The attacker used default credentials to attempt to login, but Cowrie denied access.

```
] starting service b'ssh-userauth'
ebug] b'admin' trying auth b'none'
ebug] b'admin' trying auth b'password'
 read etc/userdb.txt, default database activated
empt [b'admin'/b'admin'] failed
ebug] b'admin' failed auth b'password'
ebug] unauthorized login: ()
```

The connection timed out in a matter of seconds, which indicates this is not a hands-on attack but rather an automatic scanner.

```
sport#info] connection lost
 Connection lost after 2.0 seconds
```

# Conclusion

Phase 1 successfully established a cloud-based Cowrie honeypot on an AWS EC2 instance. The project separated legitimate SSH administration from honeypot activity by moving the real SSH service to port 2222 and placing Cowrie on port 22, the default SSH port commonly targeted by attackers.

After the honeypot was exposed to the internet, it began receiving attacker activity within a short period of time. Cowrie captured useful security data, including source IP addresses, SSH banners, HASSH fingerprints, login attempts, and failed authentication activity. This confirmed that the honeypot was working as intended and demonstrated how quickly exposed cloud services can attract automated scanning and brute-force attempts.

The next phase of this project will focus on organizing, parsing, and analyzing Cowrie logs in more depth so the captured activity can be used for threat intelligence, attacker behavior analysis, and future SIEM integration.