

## Project Precision – Implementing A Honeypot System in AWS

### Introduction

Welcome! A honeypot is a computer configured to be intentionally vulnerable, with additional logging measures in place to bolster cybersecurity defenses and gain valuable insights into potential threats by analyzing cyberattack data. This guide will demonstrate how to set up a honeypot system using an Amazon Web Services (AWS) virtual machine (otherwise known as an instance), and the open-source T-Pot software found on GitHub (<https://github.com/telekom-security/tpotce>). Before getting started, it's important to note a few key points:

- Using AWS requires inputting personal credit card verification and may incur costs, particularly if computing resources used exceed the limits of Free Tier. More information about AWS's Free Tier can be found at the following links:
  - [www.sourcefuse.com/resources/blog/aws-free-tier-limits/](http://www.sourcefuse.com/resources/blog/aws-free-tier-limits/)
  - <https://aws.amazon.com/free/>
- Technical experience, particularly with the Linux terminal, is essential for successful installation. If unfamiliar with this environment, visiting Linux Journey at the below link might be helpful in becoming acquainted with essential Linux commands and concepts:
  - <https://linuxjourney.com>

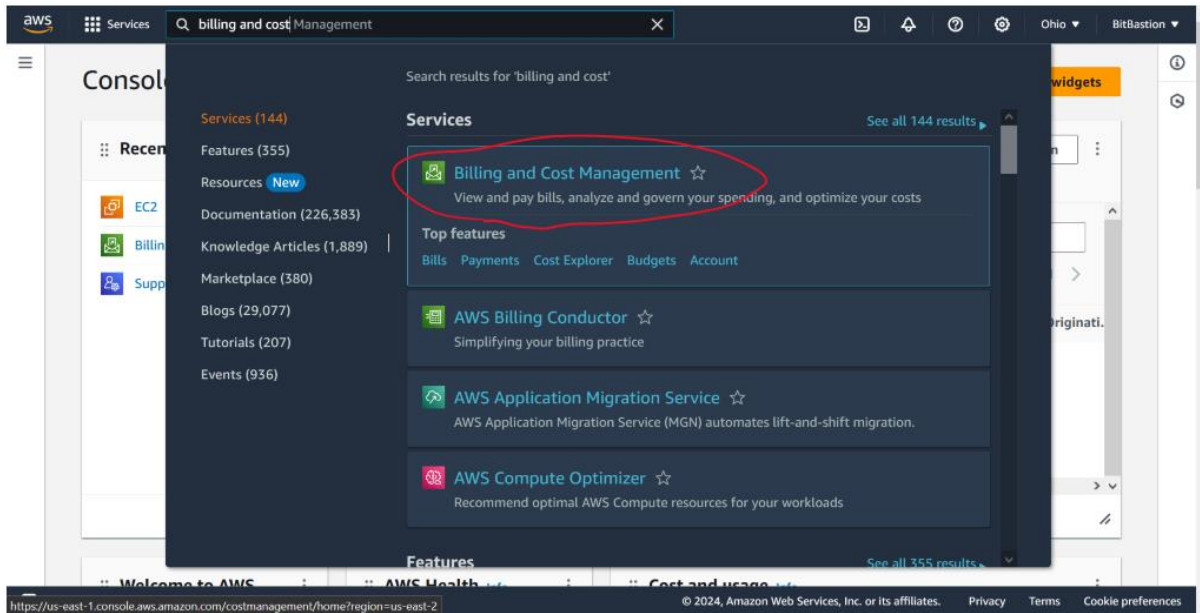
Let's get started!

### Configuring Amazon Web Services

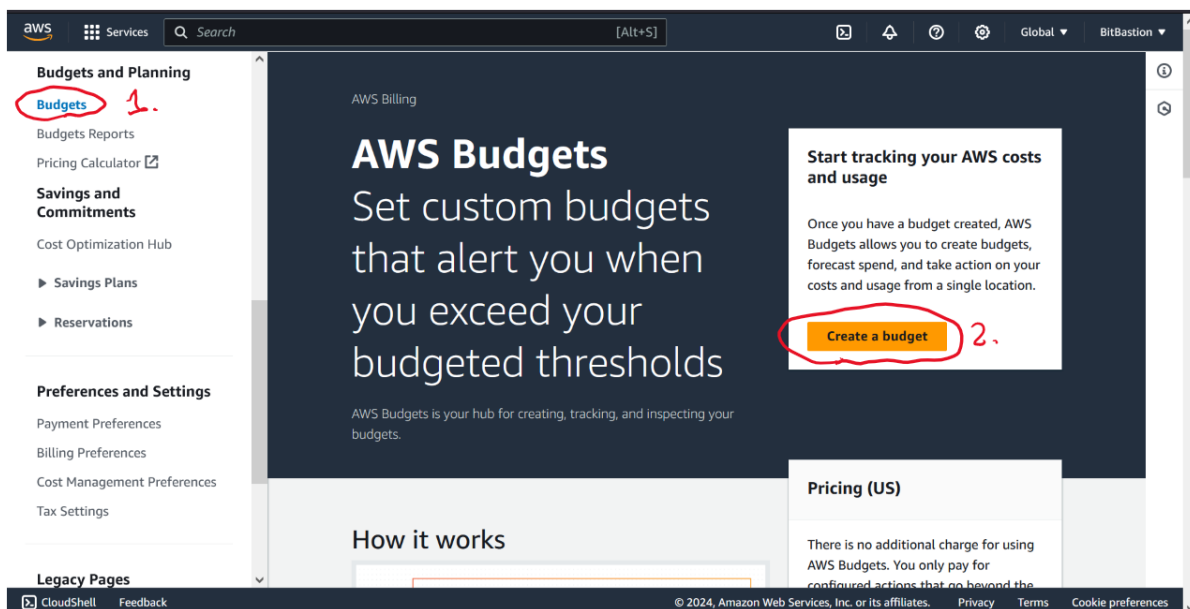
#### Setting Up an Account

To get started, navigate to <https://aws.amazon.com/> and click *Sign into The Console* → *Create A New AWS Account*. Create and sign into the account as a Root User. From there, follow the prompts to create and verify an AWS account.

After successfully creating an AWS account and logging into the Management Console, users will be redirected to the AWS Dashboard. It may be wise to set up cost alarms and controls to ensure that the cloud computing resources used are not more than anticipated. To do so, navigate to the search bar and type in *Billing and Cost*, clicking on the first search result hit:



Then, on the left-hand menu scroll down and click on *Budgets*, then select *Create a Budget*:



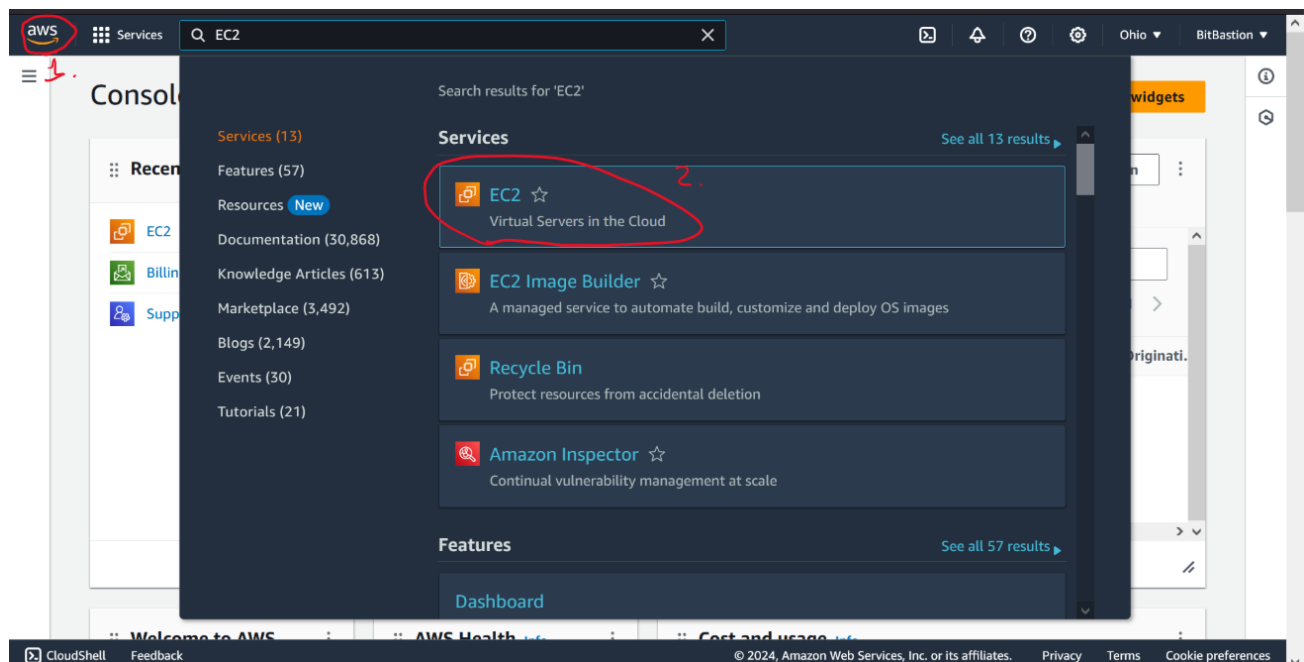
From there, select the following options:

| Menu                    | Recommended Selection | Notes |
|-------------------------|-----------------------|-------|
| <i>Budget Setup</i>     | Use A Template        |       |
| <i>Templates</i>        | Zero Spend Budget     |       |
| <i>Budget Name</i>      | <cool_name>           |       |
| <i>Email Recipients</i> | address@email.com     |       |

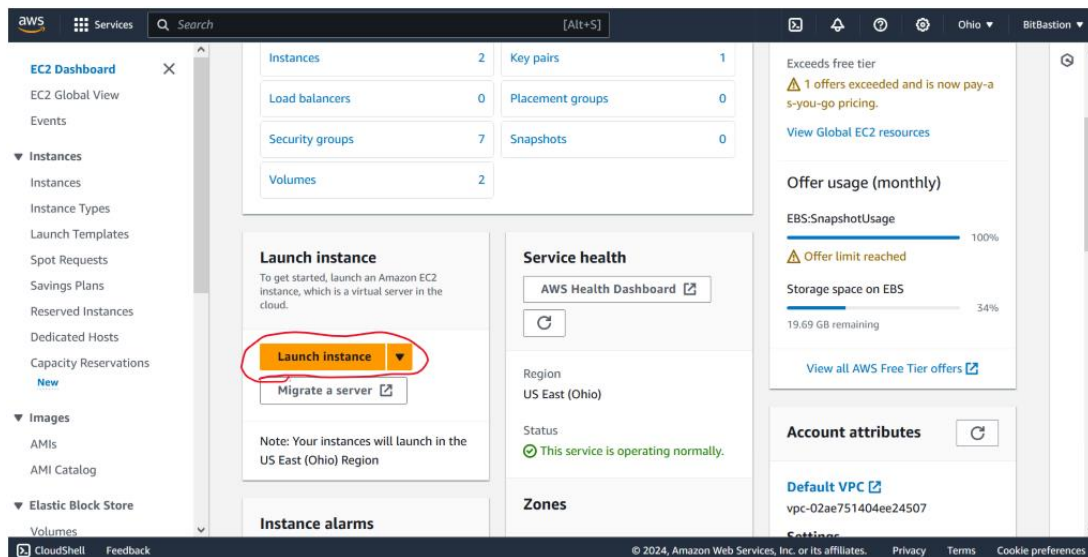
Select *Create Budget*. Email alerts will now be automatically sent whenever the usage of cloud computing resources exceeds the parameters of the Free Tier. Other budget options can be selected based on user preference.

## Configuring An Instance

Now deploy a virtual machine using AWS's EC2 Dashboard. EC2, which stands for Elastic Cloud Compute, serves as the primary interface for setting up and connecting to virtual machines. To navigate to the EC2 dashboard, click on the AWS logo in the upper left-hand corner of the window. Then go to the search bar and type in “EC2”, clicking on the first result:



From there, deploy an instance of a Debian virtual machine. Inside the main window scroll down slightly and click *Launch Instance*:



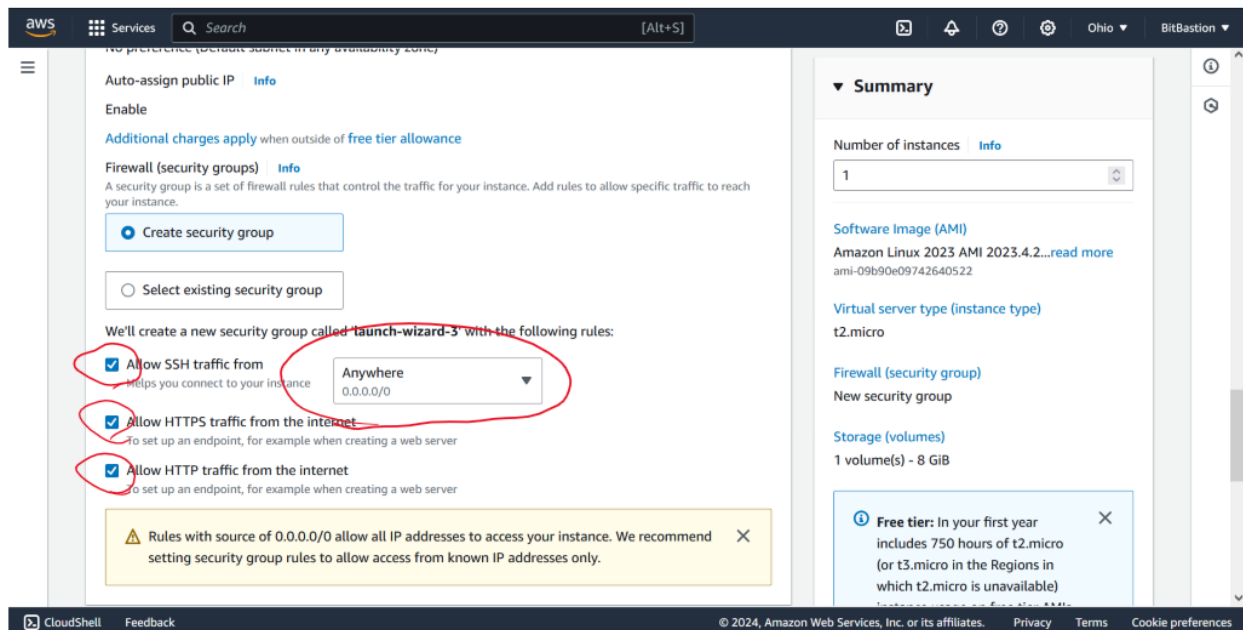
Select the following options:

| Menu                             | Recommended Selection | Notes  |
|----------------------------------|-----------------------|--|
| <i>Names And Tags</i>            | <cool_name>           |  |
| <i>Application And OS Images</i> | Debian                | - 64-bit x86, if applicable<br>- Using either the carousel menu or search function will result in the same selection |
| <i>Instance Type (2 Options)</i> | t2.micro              | Cost-effective, may suffer from performance issues   |
|                                  | t2.xlarge             | Good performance, <u>will likely incur costs</u>   |
| <i>Key Pair</i>                  | Create New Key Pair   | (see below)  |
| <i>Network Settings</i>          | Create Security Group | (see below)  |
| <i>Configure Storage</i>         | 256 GiB               | Select “gp3” root volume (see below)   |
| <i>Advanced Settings</i>         | N/A                   |  |

*Key Pair*: to securely connect to the instance, a Key Pair needs to be generated.

1. Click on *Create New Key Pair* and input a name.
2. For the “Key Pair Type,” select *RSA*.
3. Choose *.ppk* for the “Private Key File Format” radio button.
4. After clicking *Create Key Pair*, a *.ppk* file will be downloaded. Be sure to keep this file near at hand, as it will be used to authenticate the PuTTY connection to the instance.

*Network Settings*: these settings will define what connections AWS allows to the instance. For now, allow SSH, HTTP, and HTTPS traffic from anywhere on the Internet. After establishing the honeypot, these settings will be modified to allow all connections. However, for the time being, implement the following settings:



*Configure Storage*: This setting defines how much hard disk space is allocated to the instance. While the AWS Free Tier supports up to 30 GiB of data, the T-Pot software gathers an immense amount of data that needs to be stored on the hard disk. 256 GiB is recommended but will incur a cost. Input 256 in the “GiB” field and select “gp3” for the “Root Volume” drop-down if necessary.

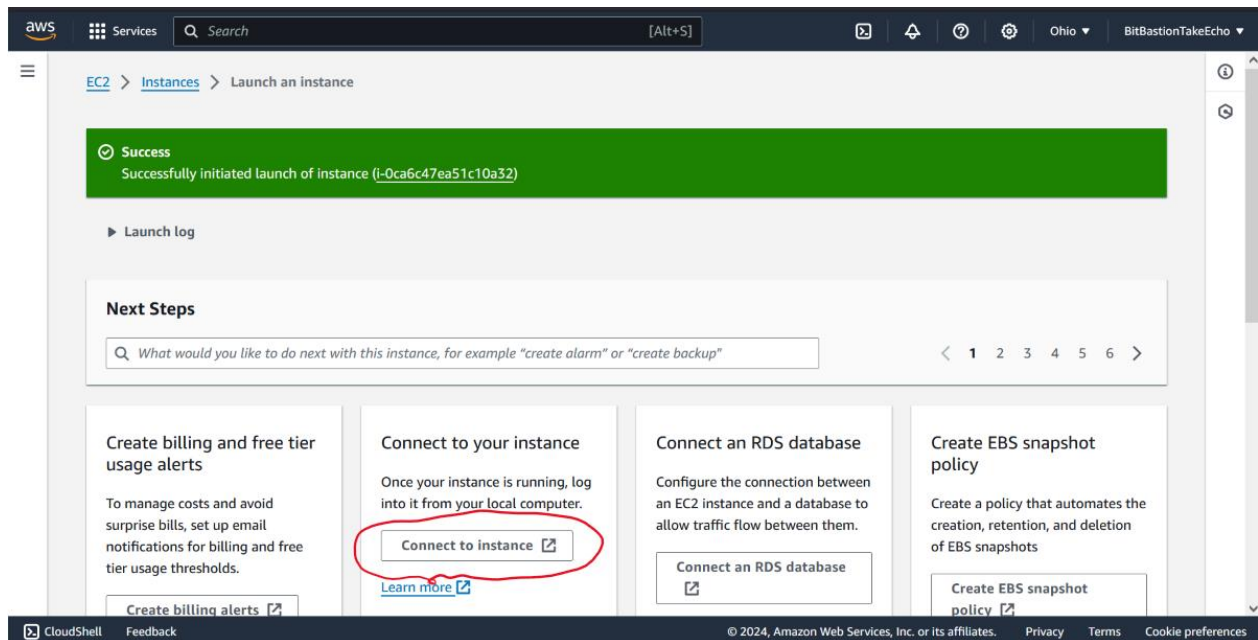
After configuring these settings, click *Launch Instance* in the lower right corner of the screen.

# Connect To Instance

## Initial Steps

Congrats! A virtual machine has now been created and is currently running on the AWS platform. Now to remotely connect to this computer and install T-Pot. This project will be using PuTTY to connect to the honeypot machine from a Windows computer. Download the PuTTY client at (<https://www.putty.org/>), clicking on *Download PuTTY* and then selecting the best software version for the hardware in use. After downloading the installer, execute it and navigate through the Installation Wizard, leaving all the default settings as-is. After installation open a PuTTY window and leave it open for a later step.

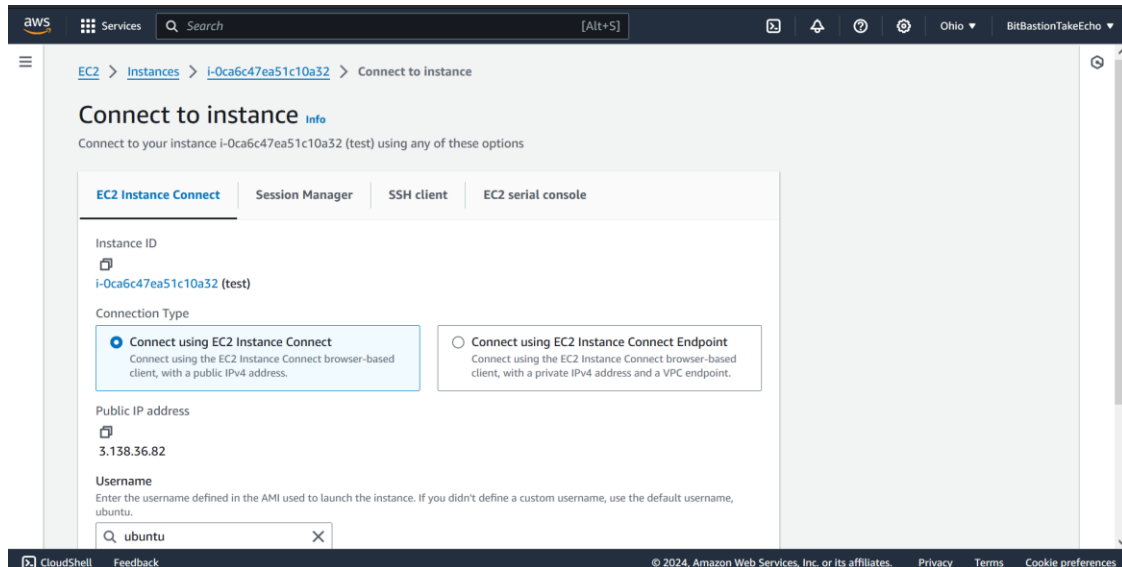
In the AWS Dashboard on the landing page immediately after clicking *Launch Instance*, click *Connect to Your Instance* on the bottom middle portion of the screen:



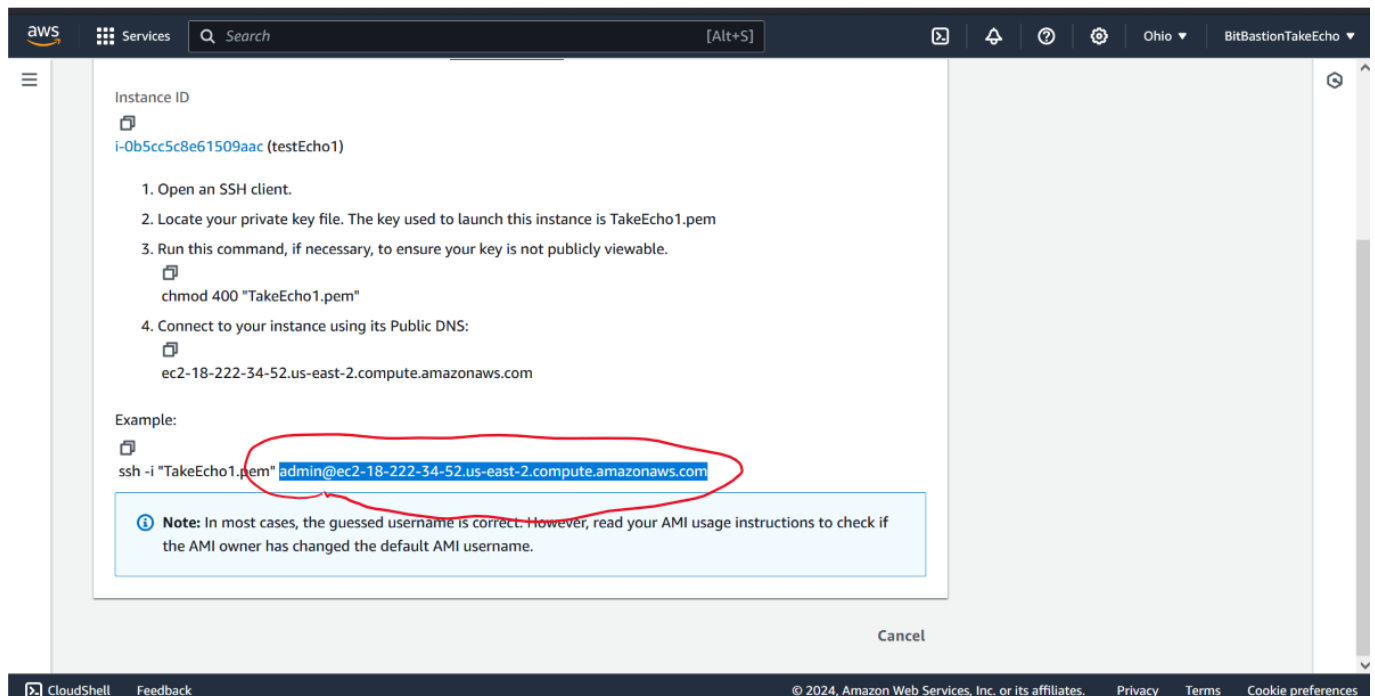
Alternatively, navigate to *Connect to Your Instance* from the AWS Dashboard and click:

*EC2 → Instances → <honeypot instance ID> → Connect*

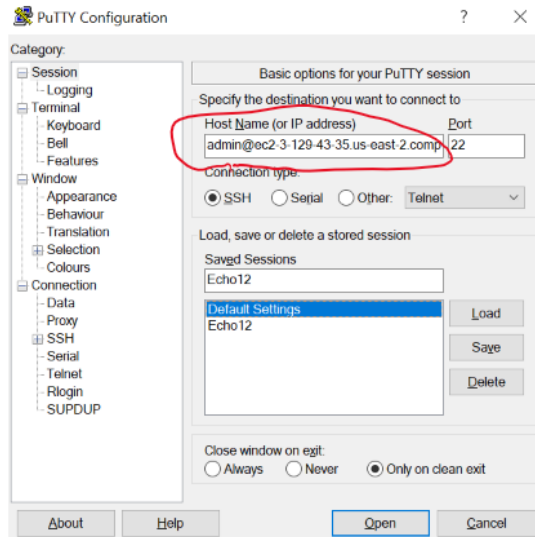
Either method should result in the following screen:



Click on *SSH Client* and scroll to the bottom of the screen. Copy the string listed under *Example*, but only from the portion beginning with “admin@...” to the end of the string, leaving out the SSH syntax. (i.e. admin@ec2-13-59-143-78.us-east-2.compute.amazonaws.com)



This copied string holds the account and routing information necessary for PuTTY to find and connect to the honeypot instance. Paste the string into the *Host Name (Or IP Address)* field of the open PuTTY window:



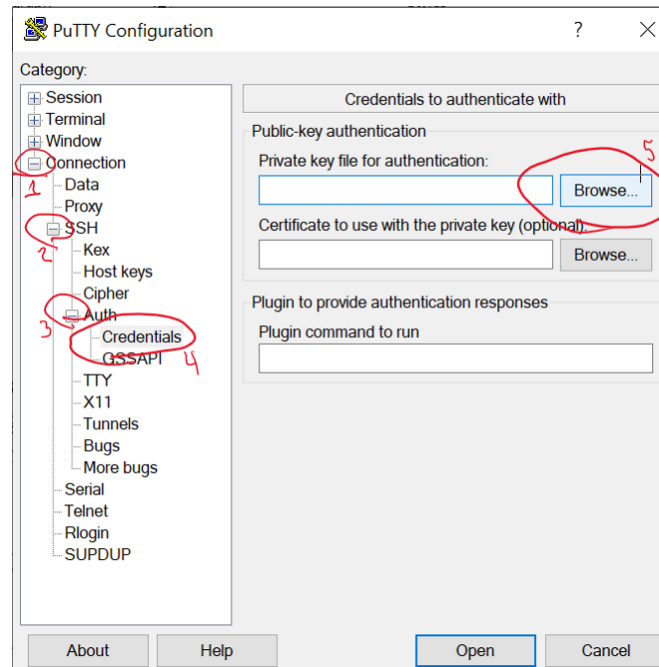
## Configuring PuTTY

Inside the PuTTY window, ensure the following settings are implemented:

| Field                            | Value             | Notes   |
|----------------------------------|-------------------|---|
| <i>Host Name (Or IP Address)</i> | <string from AWS> |   |
| <i>Port</i>                      | 22                | (after installing T-Pot this will become 64295) |
| <i>Connection Type</i>           | SSH               |   |

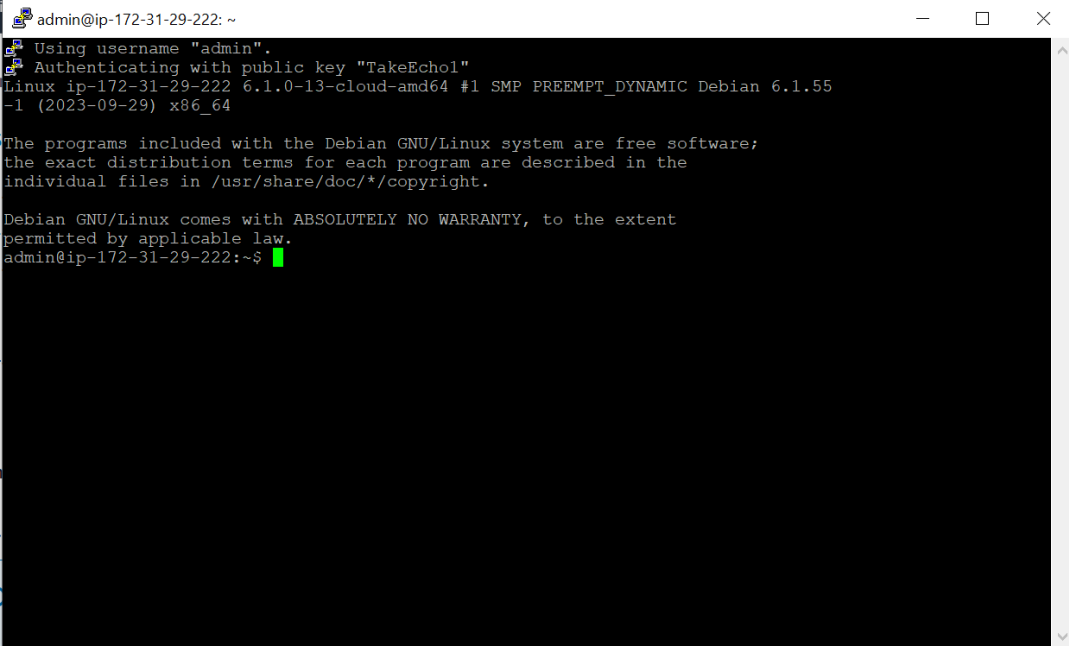
Finally, on the left-hand menu click *Connection* → *SSH* → *Auth* → *Credentials* to open the SSH authentication settings menu. Under “Private Key File for Authentication”, click *Browse*, and select the *.ppk* file that was generated while configuring the AWS instance:





On the left-hand menu, scroll up and click *Session* to return to the main menu. It may be helpful to save these connection settings for later. To do so, type a name into the *Saved Sessions* field, then click *Save*. To load those savings back after closing and reopening PuTTY, select the name given and click *Load*.

To connect to the honeypot instance, press *Open*. A “PuTTY Security Alert” window opens, displaying information about the connection. Click *Accept*. The following window should appear:

A terminal window titled 'admin@ip-172-31-29-222: ~' with standard window controls. The terminal output shows the login process for the 'admin' user using a public key 'TakeEcho1'. It identifies the system as 'Linux ip-172-31-29-222 6.1.0-13-cloud-amd64 #1 SMP PREEMPT\_DYNAMIC Debian 6.1.55-1 (2023-09-29) x86\_64'. It displays the Debian GNU/Linux system's free software notice and warranty disclaimer. The prompt 'admin@ip-172-31-29-222:~\$' is followed by a green cursor.

```
admin@ip-172-31-29-222: ~  
Using username "admin".  
Authenticating with public key "TakeEcho1"  
Linux ip-172-31-29-222 6.1.0-13-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-09-29) x86_64  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
admin@ip-172-31-29-222:~$
```

Congrats! A Debian virtual machine has been configured and booted on AWS's virtual platform, and a connection has been established to it.

## Installing T-Pot

Inside the PuTTY session, test the machine's Internet connection by inputting ping 8.8.8.8 and hitting Enter. Messages should appear similar to:

"64 bytes from 8.8.8.8: icmp\_seq=1 ttl=56 time=9.91 ms":

```
admin@ip-172-31-29-222: ~  
admin@ip-172-31-29-222:~$ ping 8.8.8.8  
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=56 time=9.99 ms  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=56 time=9.99 ms  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=56 time=9.93 ms  
^C  
--- 8.8.8.8 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2002ms  
rtt min/avg/max/mdev = 9.928/9.967/9.989/0.027 ms  
admin@ip-172-31-29-222:~$
```

Hit Ctrl + C to halt to command.

Before downloading T-Pot, it's best to uninstall or halt any software that may have a port conflict with T-Pot. In this case, the `exim4` and `systemd-resolved` software packages have conflicting ports with T-Pot (25 and 53, respectively). Removing the `exim4` message-passing software is a fair option, since this virtual machine's only purpose is to be a honeypot. However, the `systemd-resolved` process handles DNS resolution for Debian machines and should only be halted to ensure Internet connectivity. To do so, copy and paste the following commands into the PuTTY session. (After clicking into the PuTTY window, right-click to paste):

```
sudo apt-get remove exim4 exim4-base exim4-config exim4-daemon-light  
  
sudo systemctl stop systemd-resolved
```

To download and install the T-Pot software, paste and run the following commands. Hit Enter after each command and wait until each command completes to input the next one:

```
sudo apt-get update  
  
sudo apt-get install git      (Type "Y" when prompted to do so)  
  
git clone https://github.com/telekom-security/tpotce
```

```
cd tptotce
```

```
./install.sh
```

 (Type “Y” when prompted to do so)

Near the end of the installation process the following screen should be shown:

```
### Playbook was successful.

### Choose your T-Pot type:
### (H)ive - T-Pot Standard / HIVE installation.
###          Includes also everything you need for a distributed setup with sensors.
### (S)ensor - T-Pot Sensor installation.
###          Optimized for a distributed installation, without WebUI, Elasticsearch and Kibana.
### Install Type? (h/s) █
```

Press “h” when prompted and hit Enter to install for a standard hive-style honeypot installation. The installation process will then ask for a username and password that will be used to access the T-Pot web interface in the future. Note that when typing in a password, no characters will be displayed as a security feature to hide keystrokes.

Be sure to save these credentials in a secure location. After inputting and verifying a username and password, the installation process will continue to run. When the installation process is complete, the following screen will be shown, indicating a successful installation:

```
### Please review for possible honeypot port conflicts.
### While SSH is taken care of, other services such as
### SMTP, HTTP, etc. might prevent T-Pot from starting.

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       User        Inode        PID/Program name
tcp        0      0 127.0.0.54:53          0.0.0.0:*               LISTEN      996         2671         368/systemd-resolve
tcp        0      0 127.0.0.53:53          0.0.0.0:*               LISTEN      996         2669         368/systemd-resolve
tcp        0      0 0.0.0.0:5355           0.0.0.0:*               LISTEN      996         2662         368/systemd-resolve
tcp        0      0 127.0.0.1:25           0.0.0.0:*               LISTEN      0           23938        4711/exim4
tcp        0      0 0.0.0.0:64295          0.0.0.0:*               LISTEN      0           31479        7247/sshd: /usr/sbi
tcp6       0      0 :::5355                :::*                    LISTEN      996         2665         368/systemd-resolve
tcp6       0      0 :::64295                :::*                    LISTEN      0           31481        7247/sshd: /usr/sbi
tcp6       0      0 :::1:25                 :::*                    LISTEN      0           23939        4711/exim4
udp        0      0 0.0.0.0:5355           0.0.0.0:*               996         2661         368/systemd-resolve
udp        0      0 127.0.0.54:53          0.0.0.0:*               996         2670         368/systemd-resolve
udp        0      0 127.0.0.53:53          0.0.0.0:*               996         2668         368/systemd-resolve
udp        0      0 172.31.29.222:68       0.0.0.0:*               998         35043        417/systemd-network
udp6       0      0 :::5355                 :::*                    996         2664         368/systemd-resolve

### Done. Please reboot and re-connect via SSH on tcp/64295.

admin@ip-172-31-29-222:~/tptotce$ █
```

Type `sudo reboot` and hit Enter. A PuTTY message will display stating that the connection to the virtual machine was lost. Wait for a few minutes to let the machine restart. Close and reopen PuTTY, and load the settings saved previously. A few key settings will need to be changed before reconnecting to the freshly installed honeypot, both inside AWS and PuTTY.

## Reconnect To Instance

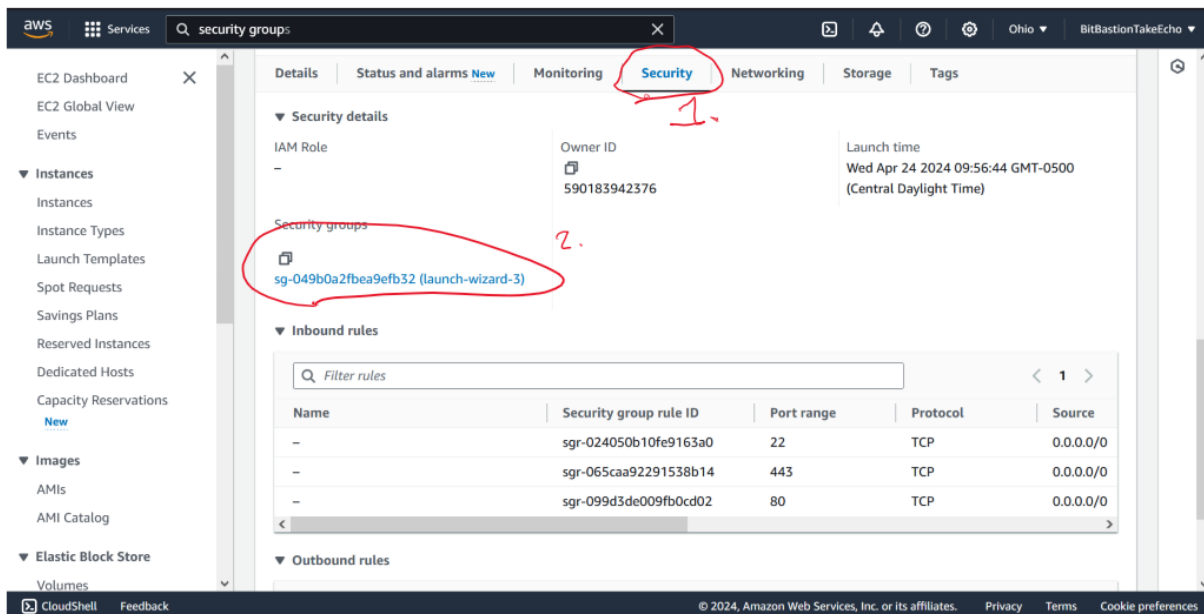
### Reconfigure PuTTY

Due to AWS's networking settings, the IP address used to connect to the honeypot virtual machine changes every time the machine boots up. Accordingly, return to the AWS Dashboard and navigate to the *SSH Client* tab of the *Connect* page for the virtual machine. Reload the page, and then re-copy the string at the bottom of the screen. Paste this string in the *Host Name (or IP Address)* field, replacing the previous value.

Additionally, change the value in the *Port* field from 22 to 64295. Click *Save*.

### Modify AWS Networking Settings

It's time to review the initial AWS networking settings and expose this virtual machine to the whole Internet in order to lure cyberattacks. To do so, return to the EC2 dashboard and click on the *Instance ID* of the honeypot machine in question. Scroll down and click on the *Security* tab, then click on the link listed under "Security Groups":



Scroll down and click *Edit Inbound Rules* in the lower right-hand portion of the screen. These rules determine what traffic AWS allows to connect to the virtual machine. By default, all connections will be denied, except those specified by the Inbound Rules. Currently three rules are implemented, delete each of them using the button on the right side of the menu.

Click *Add Rule* and implement each of the following. Ignore the security warnings – this instance should be exposed to the Internet to lure and track malicious attacks.

| Rule # | Type        | Port Range | Source    | Description       |
|--------|-------------|------------|-----------|-------------------|
| 1      | Custom TCP  | 64295      | My IP     | SSH Connection    |
| 2      | Custom TCP  | 64297      | My IP     | Web UI Connection |
| 3      | All Traffic |            | 0.0.0.0/0 | Anything Goes     |
| 4      | All Traffic |            | ::/0      | Anything Goes     |

*Note:* if utilizing a school or work connection where the connecting device’s IP address changes regularly, customization of the *Source* field may be necessary, depending on the circumstances.

Return to PuTTY and click *Open* to reconnect to the T-Pot virtual machine. A PuTTY Security Warning may appear, click *Accept* again as this is a new connection. To check if T-Pot is running, type `dps` and a screen similar to the one below should be shown:

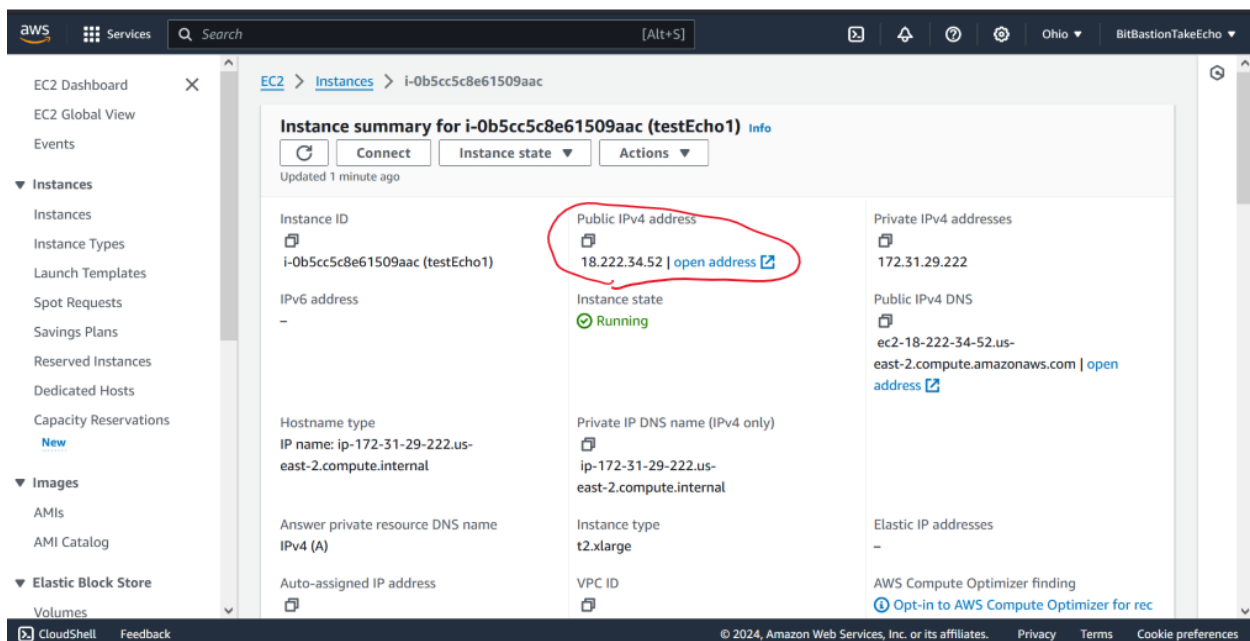
```
admin@ip-172-31-29-222:~$ dps
NAME      STATUS
adbhoney  Up 11 seconds (health: starting)
ciscoasa  Up 9 seconds
citrixhoneypot Up 9 seconds
conpot_guardian_ast Up 10 seconds (health: starting)
conpot_jec104 Up 9 seconds (health: starting)
conpot_ipmi Up 10 seconds (health: starting)
conpot_kamstrup_382 Up 9 seconds (health: starting)
cowrie    Up 10 seconds
dicomprot Up 9 seconds
dionaea   Up 9 seconds (health: starting)
, ::81->81/tcp, 0.0.0.0:1135->1135/tcp, :::135->135/tcp, 0.0.0.0:1445->1445/tcp, :::1445->1445/tcp, 0.0.0.0:1433->1433/tcp, :::1433->1433/tcp, 0.0.0.0:1723->1723/tcp, :::1723->1723/tcp, 0.0.0.0:1883->1883/tcp, :::1883->1883/tcp, 0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 0.0.0.0:27017->27017/tcp, :::27017->27017/tcp, 0.0.0.0:69->69/udp, :::69->69/udp
elasticpot Up 11 seconds
elasticsearch Up 11 seconds (health: starting)
ewaspotez Up 10 seconds
heralding  Up 8 seconds
:::465->465/tcp, 0.0.0.0:993->993/tcp, :::993->993/tcp, 0.0.0.0:995->995/tcp, :::995->995/tcp, 0.0.0.0:1080->1080/tcp, :::1080->1080/tcp, 0.0.0.0:5432->5432/tcp, :::5432->5432/tcp, 0.0.0.0:5900->5900/tcp, :::5900->5900/tcp
ipphoney   Up 12 seconds
map_redia Up 9 seconds
map_web    Up 9 seconds
medpot     Up 11 seconds
sentrypeer Up 9 seconds
spiderfoot Up 9 seconds (health: starting)
suricata   Up 14 seconds
tpotinit   Up 20 seconds (healthy)
wordpot    Up 12 seconds
admin@ip-172-31-29-222:~$
```

If a different screen is shown, there may be a port conflict causing the T-Pot software to crash. Please see the Troubleshooting section for further guidance.

## Logging Into the Dashboard

After T-Pot is up and running, it will automatically open multiple services (called sensors) that are open to attack. While these services and their behavior can be configured and optimized, doing so is entirely optional. T-Pot is entirely functional now - all that's left to do is wait for attacks.

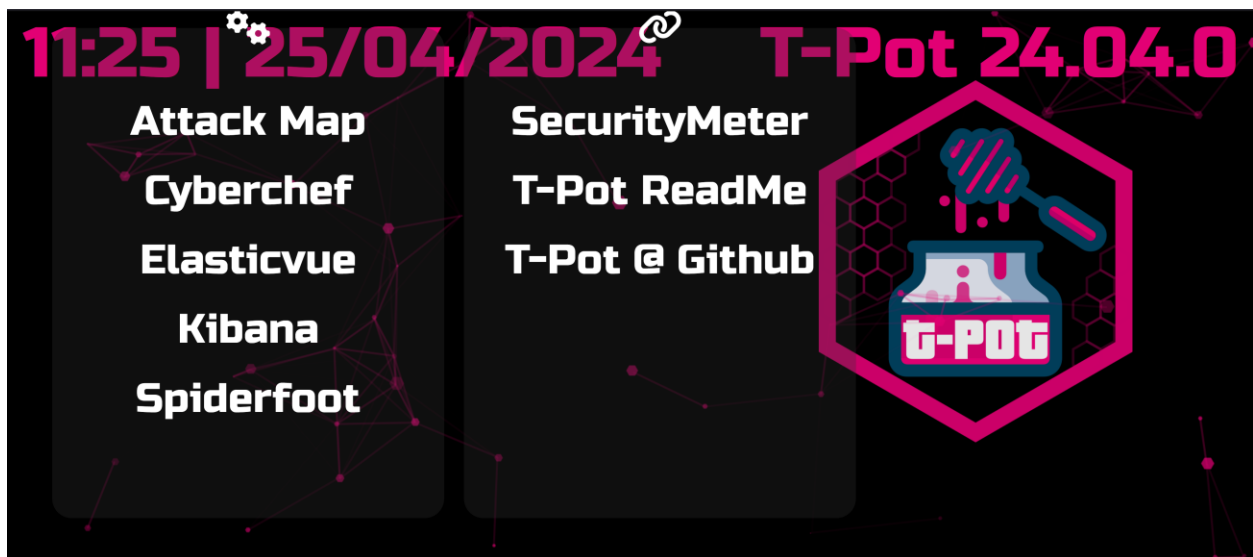
To monitor T-Pot, first determine the IP address of the virtual machine by navigating to the EC2 Dashboard inside AWS and clicking on the Instance ID of the machine. The public IP address used to reach the honeypot will be listed in the middle column:



Now, log in T-Pot's web-based Dashboard by opening up a browser and navigating to the following link (replacing the placeholder text with the virtual machine's public IP address):

`https://placeholder:64297`

A prompt will be shown asking for login credentials. Input the credentials saved during the installation of T-Pot. You may receive a warning from the browser about the security of the site, ignore it. After successful login, a screen similar to the one below should be shown:



From here, all the honeypot data can be accessed! A few notable highlights:

- Clicking on “Attack Map” will show a real-time display of the attempted connections and attacks to the honeypot.
- Clicking on “Kibana” will show the attack data for each sensor currently in use.
- Clicking on “Spiderfoot” will open a tool to launch an OSINT probe on the honeypot.

T-Pot has now been fully installed and implemented. Congratulations!

## Troubleshooting

### T-Pot Repeatedly Crashing: Port Conflict

If multiple services are using the same port, T-Pot will repeatedly crash and restart. If this is the case, running the `dps` command every 20 seconds or so for multiple minutes will show that many of T-Pot’s services start and crash within five minutes. To resolve this, first stop the T-Pot software by typing:

```
sudo systemctl stop tpot
```

Navigate to the `tpotce` folder and restart the T-Pot software by typing:

```
cd
```

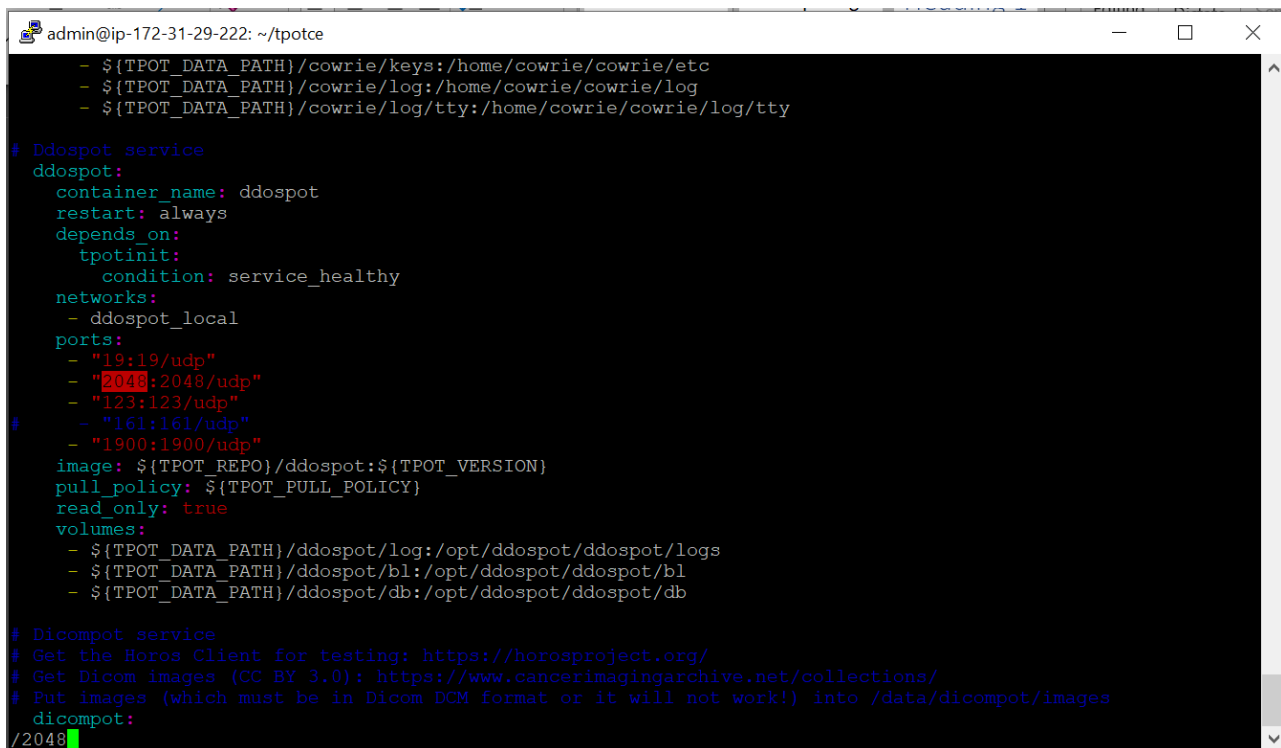
```
cd tpotce
```

```
docker compose up -d
```



If unsuccessful, the docker command will state which ports are in conflict before quitting. In testing, this project found port conflicts on TCP ports 25 and 53. Other conflicts can be resolved by either removing or halting the offending software, or modifying the ports that T-Pot uses by modifying the `docker_compose.yml` file inside the `tpotce` folder.

To modify the `docker_compose.yml` file, open it in an editor of your choice (the author uses `vim`), and search for the conflicting port (53 in this case). For each honeypot service used by T-Pot, the ports used are listed in the `ports` field. Change the value on both sides of colon to change the port used (be sure to not cause another conflict!)



```
admin@ip-172-31-29-222: ~/tpotce
- ${TPOT_DATA_PATH}/cowrie/keys:/home/cowrie/cowrie/etc
- ${TPOT_DATA_PATH}/cowrie/log:/home/cowrie/cowrie/log
- ${TPOT_DATA_PATH}/cowrie/log/tty:/home/cowrie/cowrie/log/tty

# Ddospot service
ddospot:
  container_name: ddospot
  restart: always
  depends_on:
    tpotinit:
      condition: service_healthy
  networks:
    - ddospot_local
  ports:
    - "19:19/udp"
    - "2048:2048/udp"
    - "123:123/udp"
    - "161:161/udp"
    - "1900:1900/udp"
  image: ${TPOT_REPO}/ddospot:${TPOT_VERSION}
  pull_policy: ${TPOT_PULL_POLICY}
  read_only: true
  volumes:
    - ${TPOT_DATA_PATH}/ddospot/log:/opt/ddospot/ddospot/logs
    - ${TPOT_DATA_PATH}/ddospot/bl:/opt/ddospot/ddospot/bl
    - ${TPOT_DATA_PATH}/ddospot/db:/opt/ddospot/ddospot/db

# Dicompot service
# Get the Horos Client for testing: https://horosproject.org/
# Get Dicom images (CC BY 3.0): https://www.cancerimagingarchive.net/collections/
# Put images (which must be in Dicom DCM format or it will not work!) into /data/dicompot/images
dicompot:
  /2048
```

After modifying the file appropriately, restart the T-Pot service with either `docker compose up -d` or `sudo systemctl start tpot`. After a few minutes, the `dps` command should show that T-Pot is running normally.

## Unknown Error: System Resources

Due to the vast number of services provided, T-Pot does use a significant amount of system resources. At the time of writing, 16GB of RAM and 256 GB of storage are recommended. In the case of erratic system behavior, it may be worthwhile to check the RAM

usage of the virtual machine using the `htop` command, or the current hard disk usage via the AWS platform or the `df` command to determine if the system is overwhelmed.

## Miscellaneous Tips

The default `admin` account used by AWS connections appears to have `sudo` privileges, though this may not be sufficient. To access the root account, type `sudo su -`.

After installing T-Pot, the default SSH port of 22 is changed to 64295. Further, the Web UI Dashboard can only be accessed on port 64297.

## Conclusion

T-Pot should now be up and running, gathering cyberattack data. Hopefully this guide was helpful – I gained a lot of technical knowledge throughout the process of putting this honeypot project together that I was eager to share. In particular, the commands for detecting port conflicts stumped me for a while, and hopefully sharing that knowledge will lead to less frustration for others looking to implement T-Pot. I'm excited to see what data the honeypot system brings in to see what services and attacks are most common on the open internet.

As a side note, T-Pot recently announced support for many other Linux distributions aside from Debian and can easily be installed through other means such as a locally hosted virtual machine. However, given the open-ended nature of honeypots, AWS seemed to be the logical choice to mitigate the risk of a knowledgeable attacker bypassing T-Pot and attacking critical data. At the time of writing, the T-Pot project on GitHub is still very much active and in development, with a thriving user base that is happy to assist with any issues or concerns that come up. I'd recommend checking that out if some novel issue crops up. Thanks for reading!

## References

### Honeypot Installation Guides:

- <https://medium.com/@ecojumper30/creating-a-honeypot-f2b4cc33385a>
- <https://medium.com/@mkmetty/deploying-t-pot-the-all-in-one-honeypot-platform-on-aws-ec2-33f019c645fb>

- [www.techtarget.com/whatis/feature/How-to-build-a-honeypot-to-increase-network-security](http://www.techtarget.com/whatis/feature/How-to-build-a-honeypot-to-increase-network-security)
- [intezer.com/blog/cloud-security/how-to-make-malware-honeypot](http://intezer.com/blog/cloud-security/how-to-make-malware-honeypot)
- <https://slashparity.com/?p=792>

### **T-Pot Documentation:**

- <https://www.cyberciti.biz/faq/linux-command-space-left-on-hard-disk-drive/>
- <https://github.com/telekom-security/tpotce/issues/470>
- <https://github.com/telekom-security/tpotce/issues/205>
- <https://github.com/telekom-security/tpotce/issues/407>
- <https://github.com/telekom-security/tpotce/issues/954>
- <https://github.com/telekom-security/tpotce/issues/406>
- <https://github.com/telekom-security/tpotce/discussions/899>
- <https://github.com/telekom-security/tpotce/discussions/895>
- <https://github.com/telekom-security/tpotce?tab=readme-ov-file#user-types>

### **Honeypot Data Analysis Guides:**

- <https://cdn.sisense.com/wp-content/uploads/AWS-Honeypot-Data-Visualizing-the-Threat-of-Cyberattacks.pdf>
- <https://blog.rothe.uk/analysing-honeypot-data/>
- <https://towardsdatascience.com/analysing-honeypot-data-using-kibana-and-elasticsearch-5e3d61eb2098>
- <https://hackertarget.com/cowrie-honeypot-analysis-24hrs/>
- <https://medium.com/@ashlyncmatthews/t-pot-cowrie-honeypot-analysis-5e3793bb9128>