



# Launching, Configuring, and Accessing a Free Cloud VM (AWS Example)

**Abhiram T**

# CONTENTS

<b>Overview.....</b>	<b>2</b>
<b>EC2 Instance Launch Process .....</b>	<b>2</b>
Accessing the EC2 Console.....	3
Starting the EC2 Instance Launch .....	4
Selecting the Amazon Machine Image (AMI) .....	4
Choosing an Instance Type .....	4
Configuring Instance Details .....	5
Setting Up Security Group Rules.....	5
Final Review and Launch.....	5
<b>Connectivity Testing and Instance Access .....</b>	<b>5</b>
Verifying Public IP Connectivity via Ping.....	5
Accessing the EC2 Instance via SSH .....	6
<b>Testing Public IP Connectivity from the Cloud VM .....</b>	<b>7</b>
Ping Test from the EC2 Instance .....	7
<b>Terminating the VM .....</b>	<b>7</b>
<b>Summary of Results.....</b>	<b>8</b>

## 1. Overview

The purpose of this task is to research and start a free Virtual Machine (VM) in a cloud environment (preferably Azure but AWS or GCP can also be used). The process includes configuring the VM, obtaining its IP address, verifying connectivity through ping, and accessing the VM via SSH. Additionally, tasks include performing network tests, exploring the VM, and properly shutting it down after use.

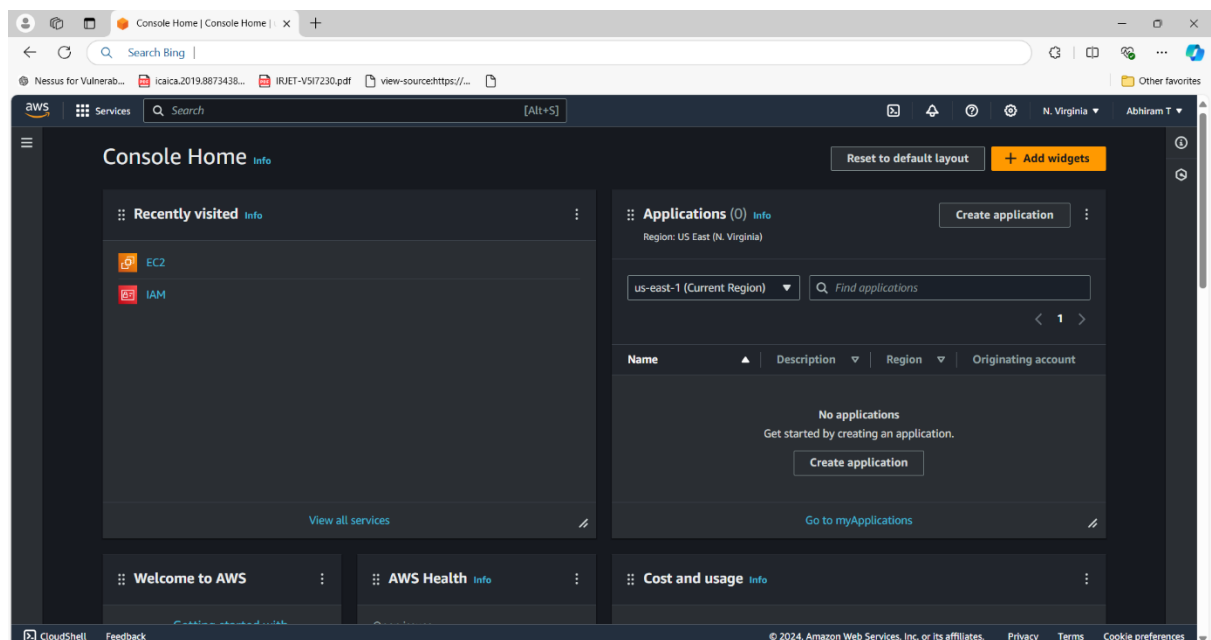
For this report, I have used AWS as the cloud provider.

## 2. EC2 Instance Launch Process

### 2.1 Accessing the EC2 Console

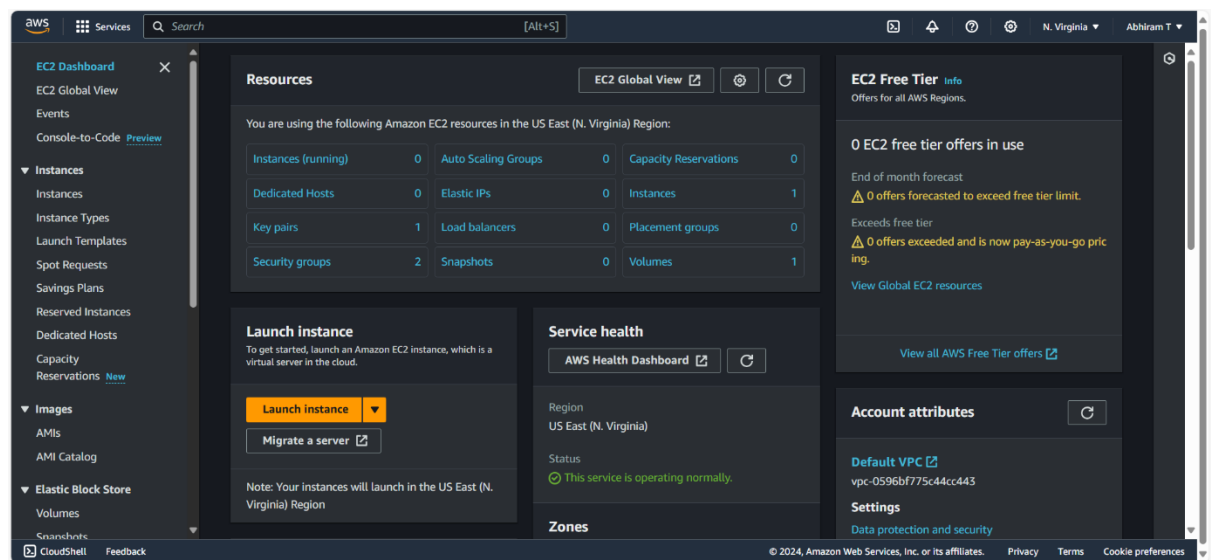
To begin the setup process, I accessed the **EC2 Dashboard** by performing the following actions:

1. I logged into the AWS Management Console and used the search bar to locate **EC2**.



Screenshot 1

2. Selecting **EC2** from the results redirected me to the EC2 dashboard.

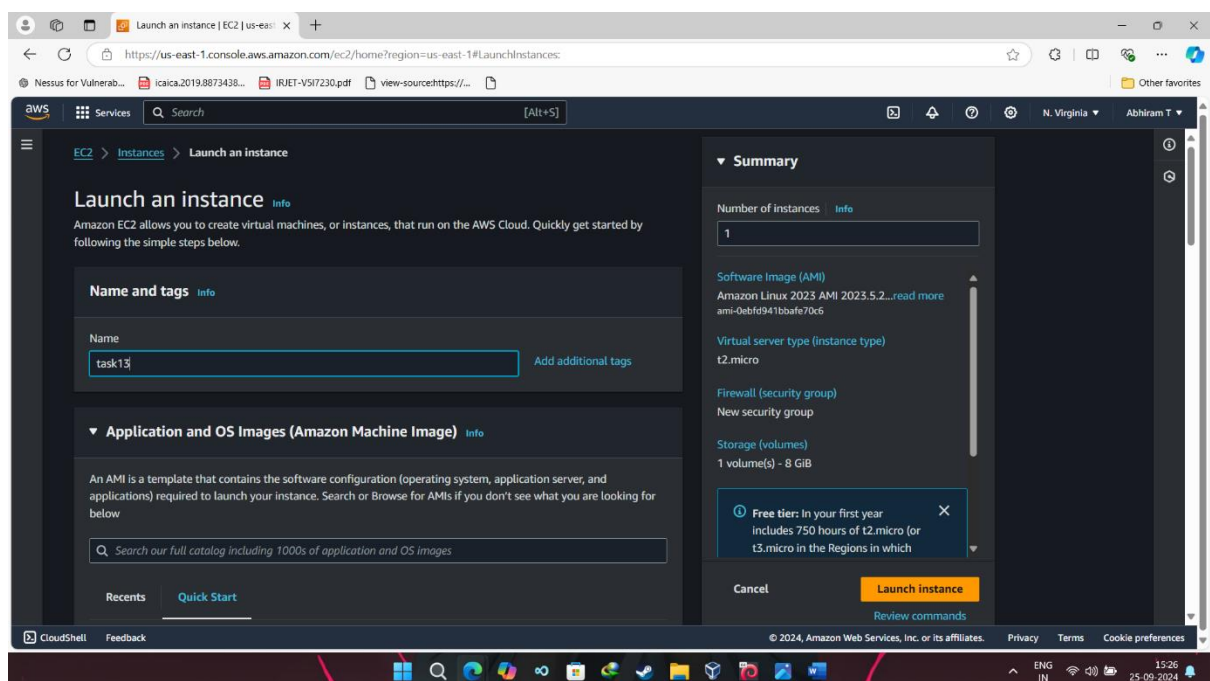


Screenshot 2

## 2.2 Starting the EC2 Instance Launch

To initiate the creation of an EC2 instance:

1. I clicked the **Launch Instance** button on the EC2 dashboard.
2. This brought me to the **Launch an Instance** page, where I could configure the settings for the virtual machine (VM).

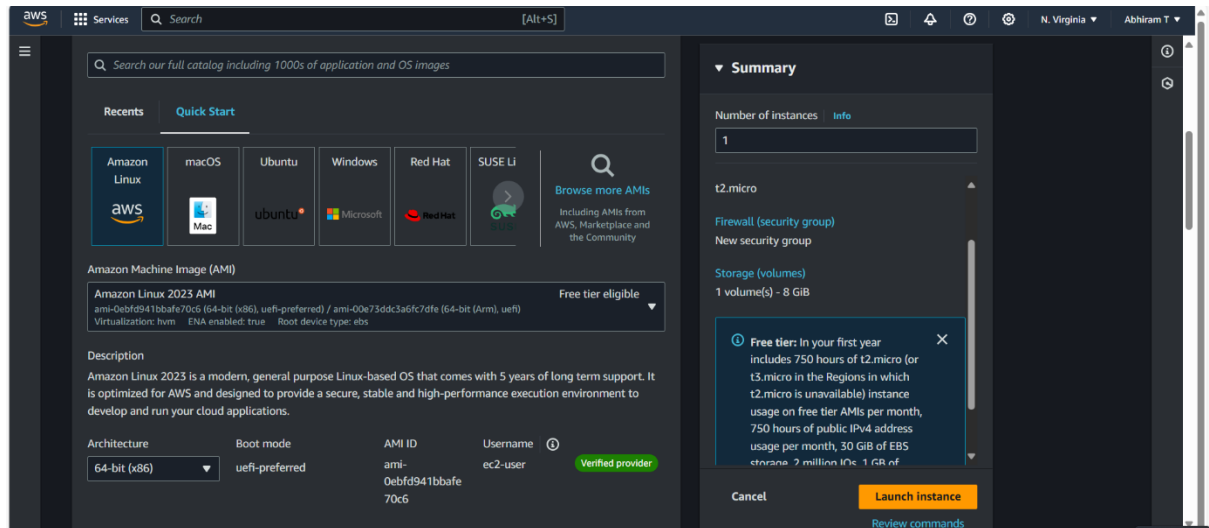


Screenshot 3

## 2.3 Selecting the Amazon Machine Image (AMI)

The Amazon Machine Image (AMI) determines the operating system for the EC2 instance:

1. I browsed through the available AMIs and chose **Amazon Linux**, which is **free-tier eligible**.
2. Other options like **macOS**, **Ubuntu** or **Windows Server** are also available, but I selected Amazon Linux for this setup.

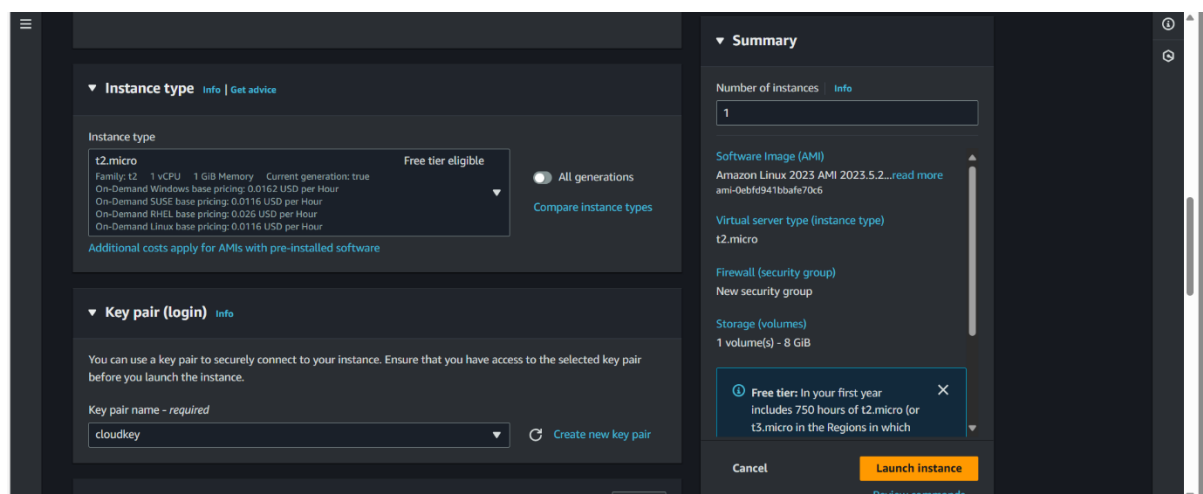


Screenshot 4

## 2.4 Choosing an Instance Type

Instance types define the resource configuration (CPU, memory, etc.):

1. I selected the **t2.micro** instance type, which is suitable for light workloads and falls under the **AWS free-tier** offering.
2. Selected **key pair** renamed to **cloudkey** [a key pair consists of a public key and a private key used for secure access to EC2 instances.]



Screenshot 5

## 2.5 Configuring Instance Details

At this stage, I configured additional settings for the instance:

1. I used the **default VPC (Virtual Private Cloud)** for networking, leaving most of the default options unchanged.
2. This configuration was adequate for my needs, as I didn't require any advanced networking configurations.

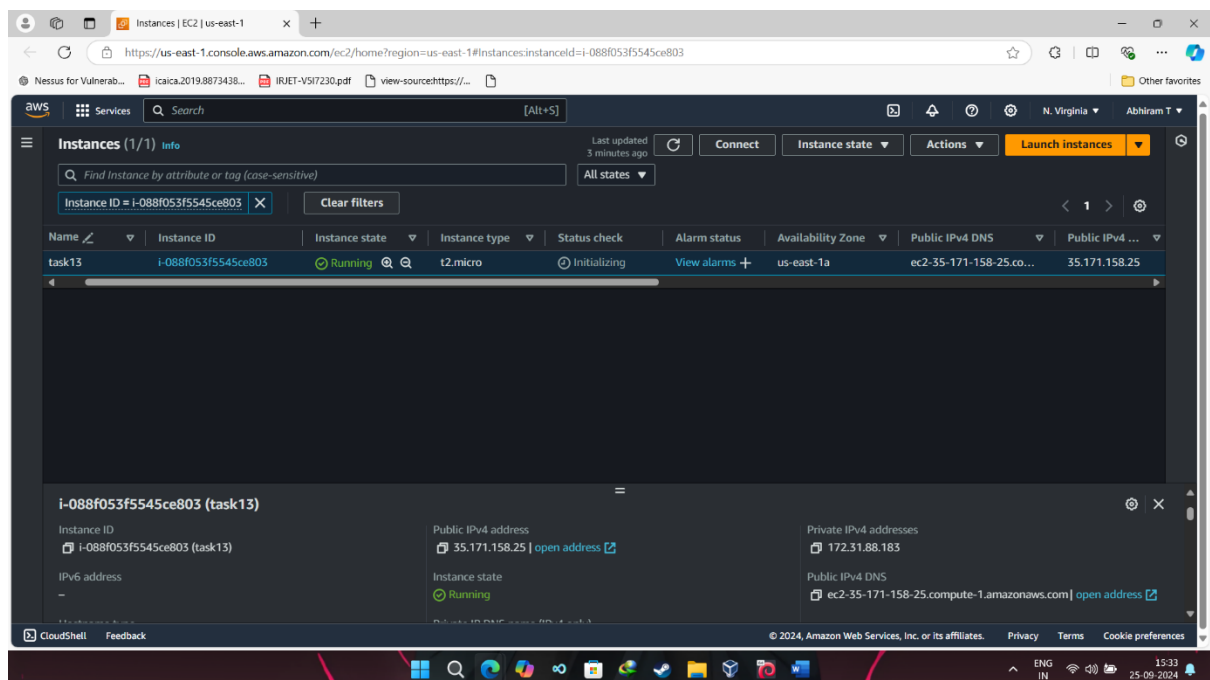
## 2.6 Setting Up Security Group Rules

The **Security Group** is a virtual firewall that controls the traffic to and from the instance:

1. For this instance, **SSH (port 22)** was pre-configured to allow remote access for Linux.
2. I later made modifications to the **inbound rules** to enable specific traffic, such as allowing ICMP for ping tests. (*refer screenshot 8*)

## 2.7 Final Review and Launch

1. I reviewed the configuration settings and confirmed they were correct.
2. After verifying the setup, I clicked **Launch Instance**, which successfully created and deployed the EC2 instance.



Screenshot 6

## 3. Connectivity Testing and Instance Access

### 3.1 Verifying Public IP Connectivity via Ping

Once the instance was up and running, I tested the **public IP connectivity**:

1. The public IP assigned to my EC2 instance was:

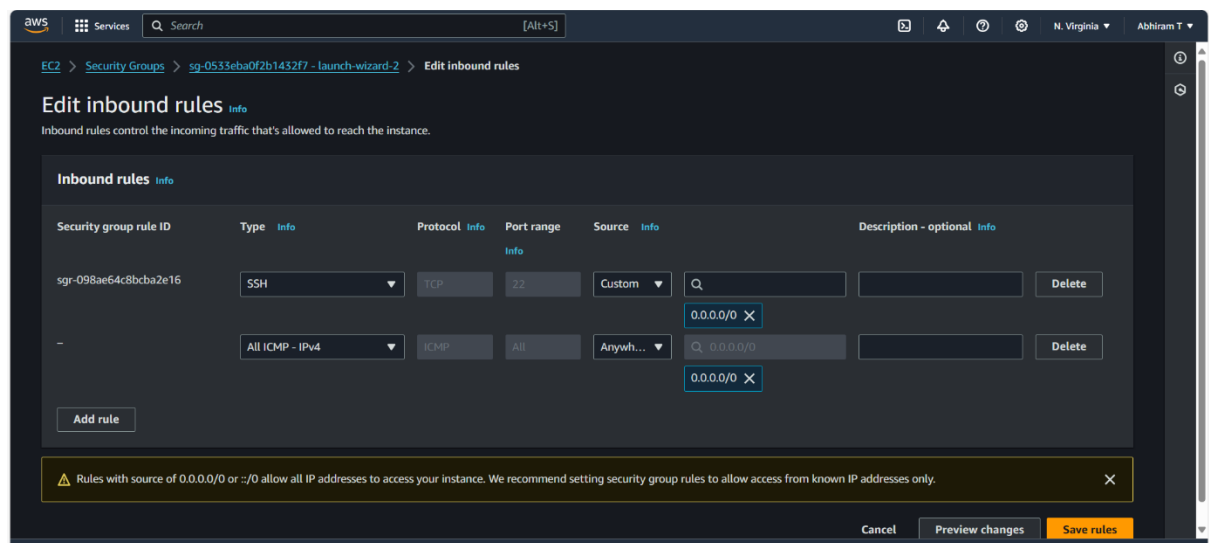
Public IP: 35.171.158.25

- Initially, I was unable to ping the instance from outside.

```
dybala@abht:~$ ping 35.171.158.25
PING 35.171.158.25 (35.171.158.25) 56(84) bytes of data.
^C
--- 35.171.158.25 ping statistics ---
42 packets transmitted, 0 received, 100% packet loss, time 42023ms
```

Screenshot 7

- However, after modifying the **inbound rules** in the security group to allow **ICMP traffic**.



Screenshot 8

I was able to successfully ping the public IP

```
dybala@abht:~$ ping 35.171.158.25
PING 35.171.158.25 (35.171.158.25) 56(84) bytes of data.
64 bytes from 35.171.158.25: icmp_seq=1 ttl=113 time=281 ms
64 bytes from 35.171.158.25: icmp_seq=2 ttl=113 time=277 ms
64 bytes from 35.171.158.25: icmp_seq=3 ttl=113 time=279 ms
64 bytes from 35.171.158.25: icmp_seq=4 ttl=113 time=276 ms
64 bytes from 35.171.158.25: icmp_seq=5 ttl=113 time=274 ms
^C
--- 35.171.158.25 ping statistics ---
6 packets transmitted, 5 received, 16.667% packet loss, time 5026ms
rtt min/avg/max/mdev = 273.851/276.651/281.356/2.535 ms
dybala@abht:~$
```

Screenshot 9

### 3.2 Accessing the EC2 Instance via SSH

To access the EC2 instance remotely, I used **SSH** with the following steps:

- I used the security key **[cloudkey]** (generated during the instance setup) to securely log in to the instance from my local machine.
- On my **Kali Linux** system, I used the following command to establish the SSH connection:

```
sudo ssh -i Documents/cloudkey.pem ec2-user@52.91.114.191
```

3. I successfully logged into the instance and gained shell access to the EC2 virtual machine.

[illegible]

Screenshot 10 [Due to some issues, I had to stop and restart the instance, which resulted in the assignment of a new public IP.]

#### 4. Testing Public IP Connectivity from the Cloud VM

### 4.1 Ping Test from the EC2 Instance

I attempted to verify the public IP connectivity by pinging my **public IP** from within the EC2 instance itself:

1. However, I was unable to ping the public IP due to restrictions imposed by the **ISP** blocking **ICMP (ping) packets**.
2. This is a common limitation where Internet Service Providers prevent ICMP traffic for security or network performance reasons.

[illegible]

Screenshot 11

## 5. Terminating the VM

Finally, I returned to the AWS EC2 dashboard and **terminated the VM** to ensure it no longer incurs any charges or resources.



## **SUMMARY OF RESULTS**

1. I successfully started a free-tier VM on AWS and obtained the public IP.
2. After adjusting the security settings, I was able to ping the VM from my Kali Linux machine.
3. I logged into the VM via SSH, ran the ip a command, and captured the screenshot.
4. I attempted to ping the public IP from within the cloud VM, but the request was blocked by ISP restrictions.
5. After completing the tasks, I shut down and terminated the VM as required.