Optimizing Linux Server Performance: Configuring Swap Memory on AWS EC2

Key takeaways:

1. **Understanding Swap Memory**
2. **Configuring Swap Memory on Linux**
3. **Automating Swap Configuration**:
4. **Stress Testing for Performance Evaluation**
5. **Monitoring System Performance**:

Introduction:

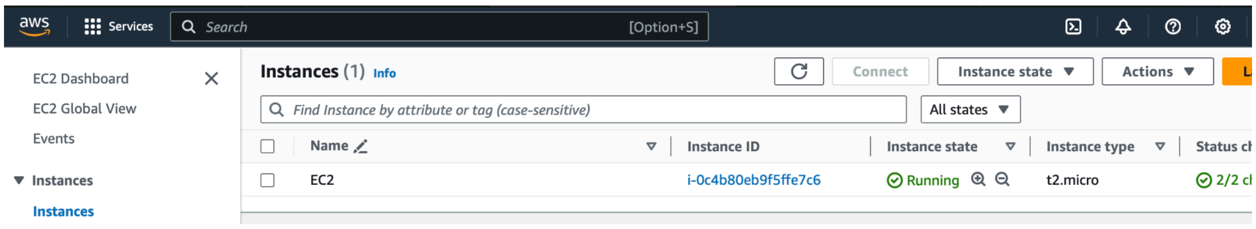
In this guide, we'll walk through the implementation of swap memory on an Ubuntu virtual machine hosted on AWS EC2((Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2024-03-01).Swap memory, also known as swap space, plays a crucial role in managing system memory effectively, especially when the RAM is fully utilized. Additionally, we'll explore stress testing to evaluate system performance under heavy CPU load.

What is Swap Memory?

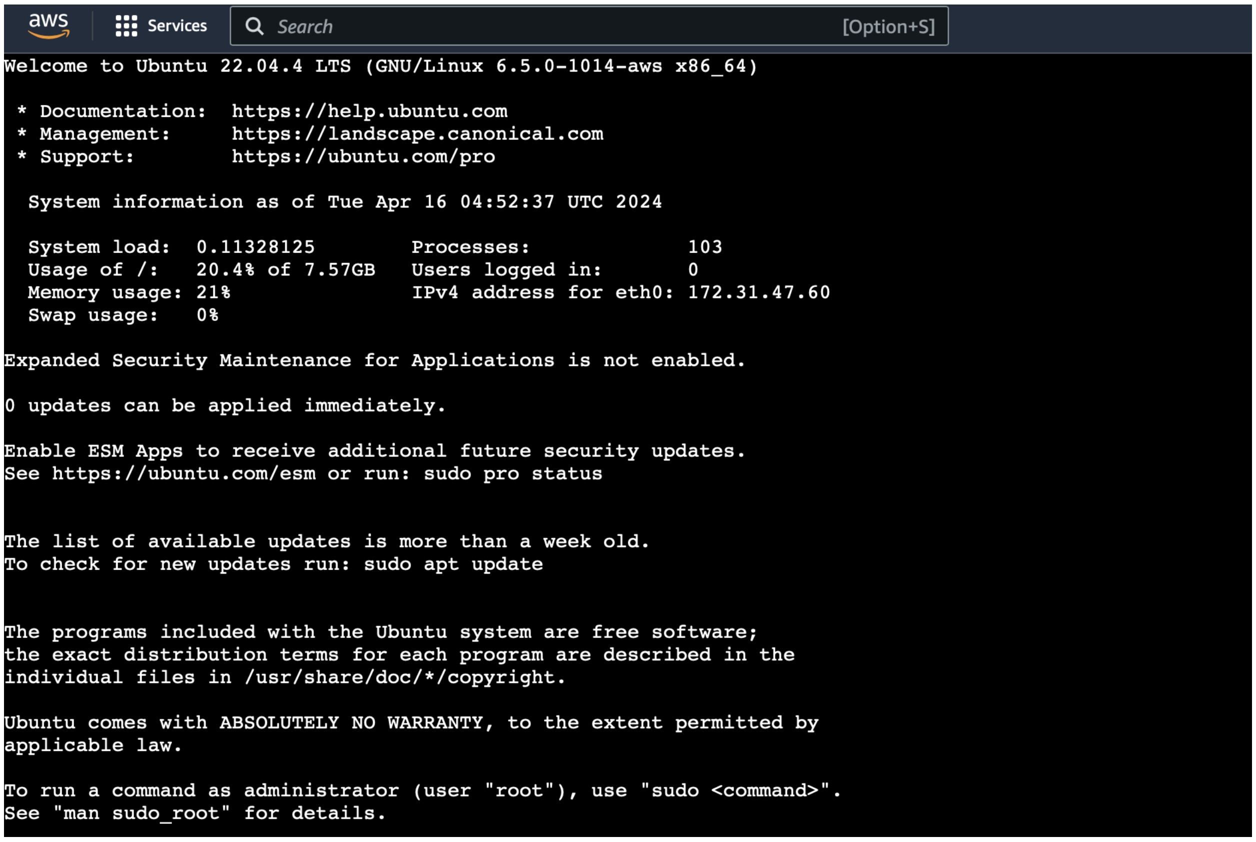
Swap memory, often known as swap space, Swap memory serves as an extension of physical RAM, residing on disk space. When RAM is fully utilized, the operating system transfers inactive data from RAM to swap space to make room for active processes. While accessing data from swap space is slower compared to RAM, it prevents the system from running out of memory.

Steps Involved:

1.Creating an AWS EC2 Instance: Launch an Ubuntu EC2 instance to configure swap memory.



Name	Instance ID	Instance state	Instance type	Status c
EC2	i-0c4b80eb9f5ffe7c6	Running	t2.micro	2/2 c

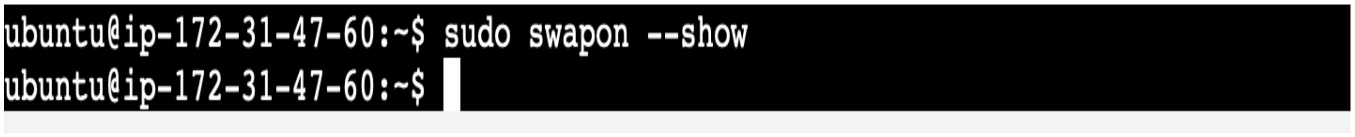


```
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1014-aws x86_64)

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

  System information as of Tue Apr 16 04:52:37 UTC 2024

  System load:  0.11328125        Processes:             103
  Usage of /:   20.4% of 7.57GB   Users logged in:       0
  Memory usage: 21%               IPv4 address for eth0: 172.31.47.60
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

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


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


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.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

2.Checking Current Swap Status:

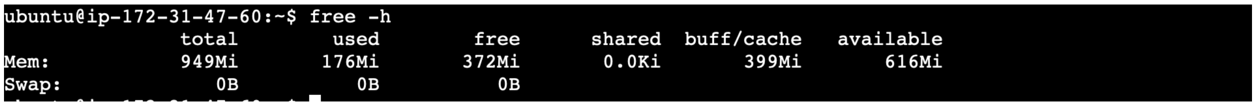
Before proceeding, let's check the current swap status. Although it's possible to have multiple swap files or partitions, usually one suffices. To verify, execute the following command:

```
$Sudo swapon --show
```



```
ubuntu@ip-172-31-47-60:~$ sudo swapon --show
ubuntu@ip-172-31-47-60:~$
```

If you receive no output, it indicates that your system currently lacks swap space. You can verify that there is no active swap using the free utility:



```
ubuntu@ip-172-31-47-60:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           949Mi       176Mi       372Mi       0.0Ki       399Mi       616Mi
Swap:             0B          0B          0B
```

In the output, you can observe that there is no active swap on the system, as indicated in the Swap row.

3.Creating a Swap File: Determine the appropriate swap size based on RAM and create a swap file using "fallocate" command.

When deciding how much swap space to use based on your RAM size and whether you want hibernation, keep in mind that hibernation requires more space because it saves your system's state when you turn it off and restores it when you turn it back on

This table lists the swap sizes recommended depending on your RAM

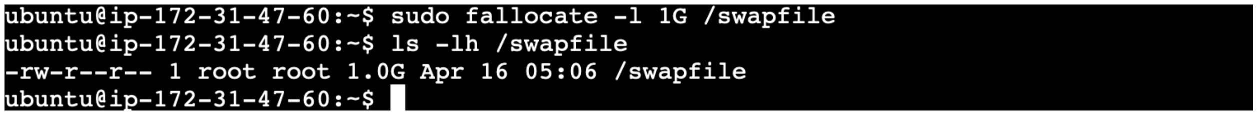
RAM Size	Swap size (without hibernation)	Swap size (with hibernation)
256MB	256MB	512MB
512MB	512MB	1GB
1GB	1GB	2GB
2GB	1GB	3GB
3GB	2GB	5GB
4GB	2GB	6GB
6GB	2GB	8GB
8GB	3GB	11GB

So in my case my system has 1GB of main memory. So we will create a swap file of 1 GB in size.
The best way of creating a swap file is with the `fallocate` program. This command instantly creates a file of the specified size.In my case my system has 1GB of main memory and am going to create a swap file of 1 GB in size.

```
sudo fallocate -l 1G /swapfile
```

We can verify that the correct amount of space was reserved by typing:

```
ls -lh /swapfile
```



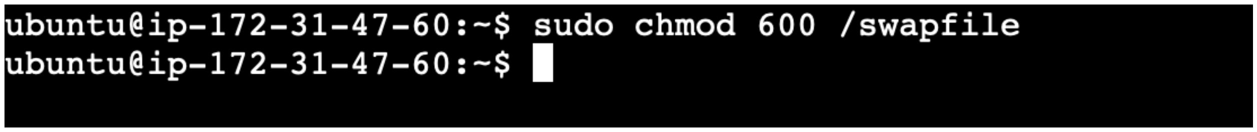
```
ubuntu@ip-172-31-47-60:~$ sudo fallocate -l 1G /swapfile
ubuntu@ip-172-31-47-60:~$ ls -lh /swapfile
-rw-r--r-- 1 root root 1.0G Apr 16 05:06 /swapfile
ubuntu@ip-172-31-47-60:~$
```

4.Enabling Swap File: Set appropriate permissions using "chmod" and mark the file as swap space using "mkswap". Enable swap file usage with "swapon".

To make the file accessible only to root, type the following command:

Sudo chmod 600 /path/to/your/swapfile

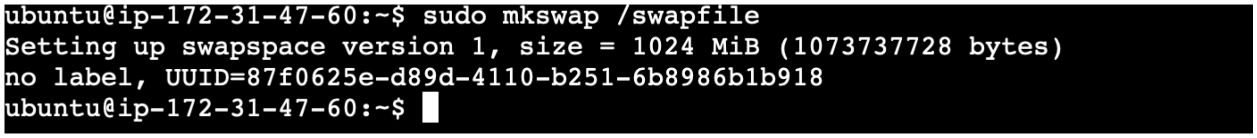
This command sets the permissions of the swap file to allow read and write access only for the owner (root), and no access for other users.



```
ubuntu@ip-172-31-47-60:~$ sudo chmod 600 /swapfile
ubuntu@ip-172-31-47-60:~$
```

We can now mark the file as swap space by typing:

sudo mkswap /swapfile



```
ubuntu@ip-172-31-47-60:~$ sudo mkswap /swapfile
Setting up swapspace version 1, size = 1024 MiB (1073737728 bytes)
no label, UUID=87f0625e-d89d-4110-b251-6b8986b1b918
ubuntu@ip-172-31-47-60:~$
```

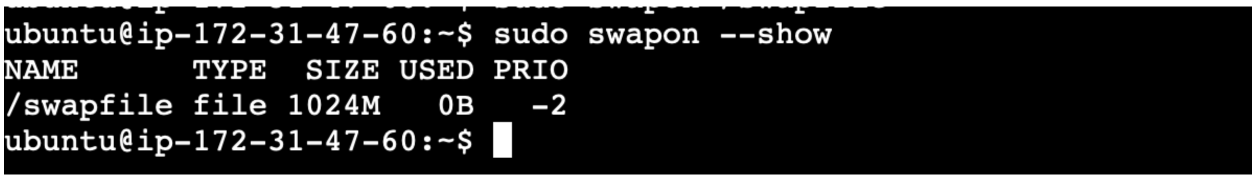
After marking the file, we can enable the swap file, allowing our system to start using it:

sudo swapon /swapfile

5.Verifying Swap Memory:

Verify that the swap is available by typing:

sudo swapon --show

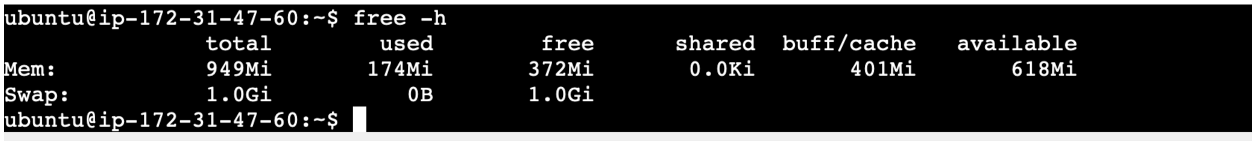


```
ubuntu@ip-172-31-47-60:~$ sudo swapon --show
NAME      TYPE  SIZE USED PRIO
/swapfile file 1024M   0B   -2
ubuntu@ip-172-31-47-60:~$ 
```

6.Making the Swap File Permanent

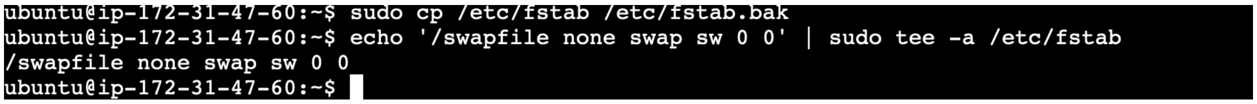
We have successfully added swap memory to our system. Execute one of the below commands to view current active swap memory on our system:

```
free -h
```



```
ubuntu@ip-172-31-47-60:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           949Mi       174Mi       372Mi       0.0Ki       401Mi       618Mi
Swap:          1.0Gi          0B       1.0Gi
ubuntu@ip-172-31-47-60:~$ 
```

Our recent changes have enabled the swap file for the current session. However, if we reboot, the server will not retain the swap settings automatically. We can Add the swap file to "/etc/fstab" for automatic activation on system boot.Back up the `/etc/fstab` file in case anything goes wrong:



```
ubuntu@ip-172-31-47-60:~$ sudo cp /etc/fstab /etc/fstab.bak
ubuntu@ip-172-31-47-60:~$ echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
/swapfile none swap sw 0 0
ubuntu@ip-172-31-47-60:~$ 
```

7. Updating Swappiness Parameter

Adjust the swappiness parameter to control the frequency of swapping between RAM and swap space.

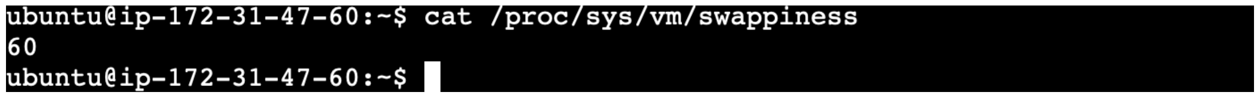
The swappiness parameter determines how frequently your system moves data from RAM to the swap space. It's a value ranging from 0 to 100, representing a percentage.

With values near zero, the system avoids swapping data to disk unless absolutely necessary. Remember, accessing the swap file takes much longer than accessing RAM and can significantly reduce performance. Thus, reducing reliance on swap generally improves system speed.

Values closer to 100 prioritize moving more data into swap to keep more RAM space available. Depending on your application's memory usage or server purpose, this approach might be preferable in certain scenarios.

We can see the current swappiness value by typing:

cat /proc/sys/vm/swappiness



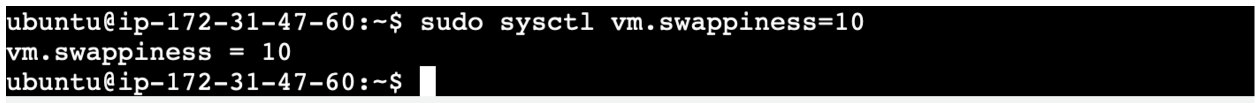
```
ubuntu@ip-172-31-47-60:~$ cat /proc/sys/vm/swappiness
60
ubuntu@ip-172-31-47-60:~$ 
```

Now change the swappiness kernel parameter as per our requirement. It tells the system how often system utilize this swap area.

We can set the swappiness to a different value by using the `sysctl` command.

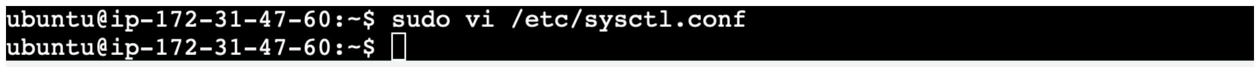
For instance, to set the swappiness to 10, we could type:

sudo sysctl vm.swappiness=10



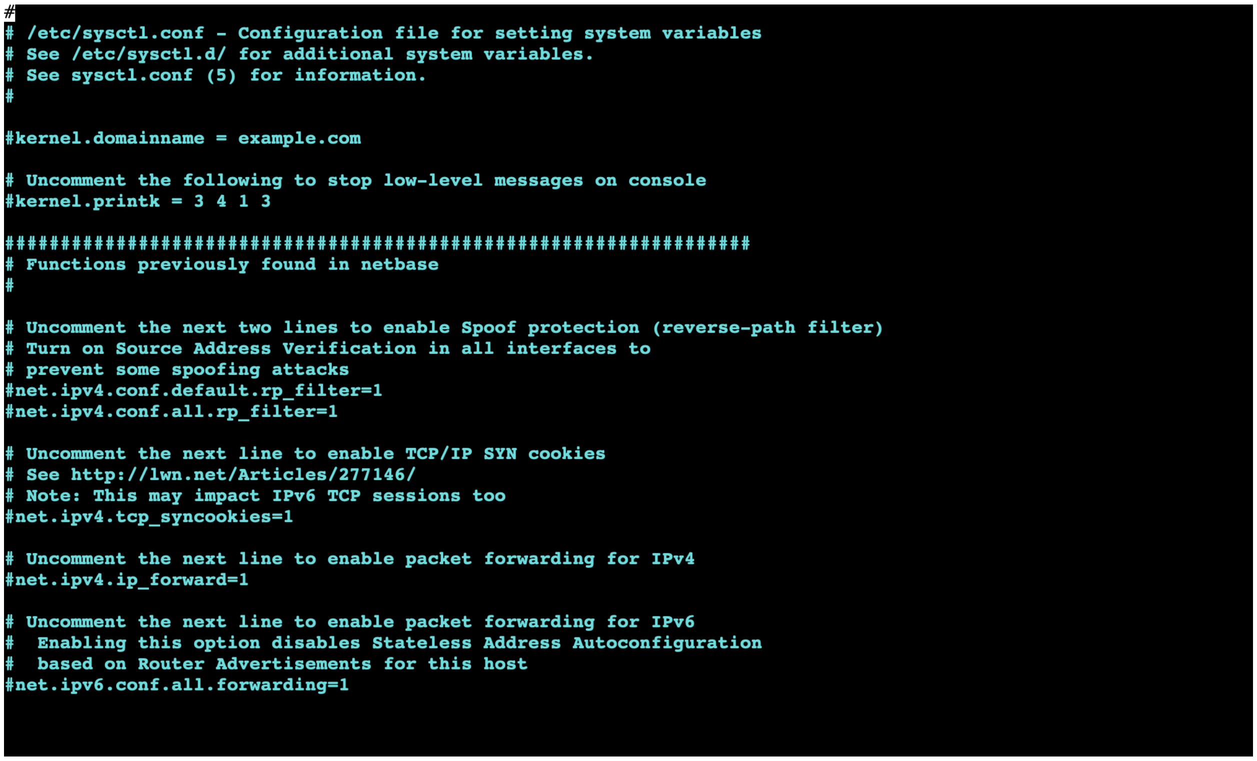
```
ubuntu@ip-172-31-47-60:~$ sudo sysctl vm.swappiness=10
vm.swappiness = 10
ubuntu@ip-172-31-47-60:~$ 
```

This setting will persist until the next reboot. We can set this value automatically at restart by adding the line to our `/etc/sysctl.conf` file:



```
ubuntu@ip-172-31-47-60:~$ sudo vi /etc/sysctl.conf
ubuntu@ip-172-31-47-60:~$ 
```

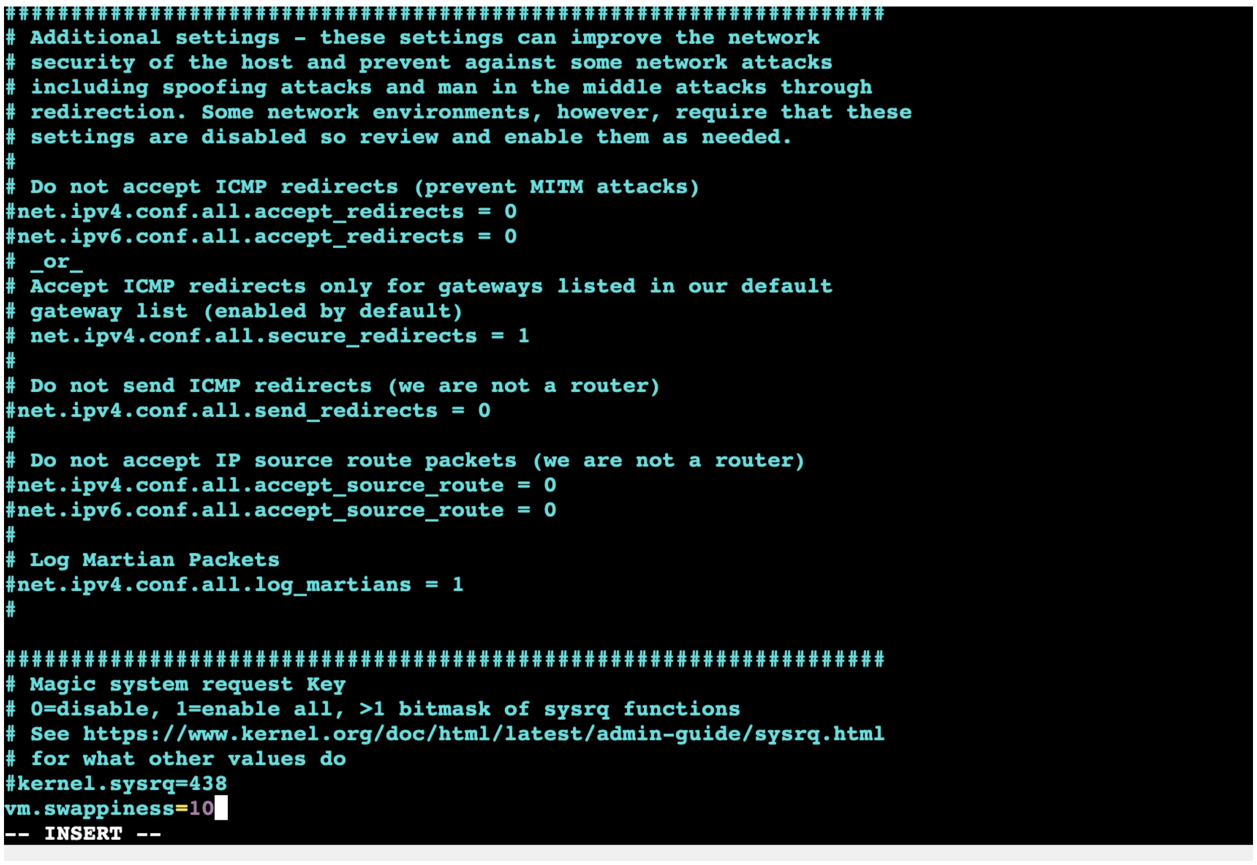
A text editor is open as below



```
#
# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables.
# See sysctl.conf (5) for information.
#

#kernel.domainname = example.com

# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3

##############################################################
# Functions previously found in netbase
#

# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
#net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
#  Enabling this option disables Stateless Address Autoconfiguration
#  based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1
```

Insert the following command at the end of the file, then save it.

```
vm.swappiness=10
```



```
###################################################################
# Additional settings - these settings can improve the network
# security of the host and prevent against some network attacks
# including spoofing attacks and man in the middle attacks through
# redirection. Some network environments, however, require that these
# settings are disabled so review and enable them as needed.
#
# Do not accept ICMP redirects (prevent MITM attacks)
#net.ipv4.conf.all.accept_redirects = 0
#net.ipv6.conf.all.accept_redirects = 0
# _or_
# Accept ICMP redirects only for gateways listed in our default
# gateway list (enabled by default)
# net.ipv4.conf.all.secure_redirects = 1
#
# Do not send ICMP redirects (we are not a router)
#net.ipv4.conf.all.send_redirects = 0
#
# Do not accept IP source route packets (we are not a router)
#net.ipv4.conf.all.accept_source_route = 0
#net.ipv6.conf.all.accept_source_route = 0
#
# Log Martian Packets
#net.ipv4.conf.all.log_martians = 1
#

###################################################################
# Magic system request Key
# 0=disable, 1=enable all, >1 bitmask of sysrq functions
# See https://www.kernel.org/doc/html/latest/admin-guide/sysrq.html
# for what other values do
#kernel.sysrq=438
vm.swappiness=10
-- INSERT --
```

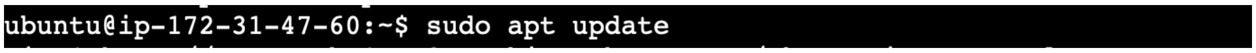
Save and close the file

8.Installing the "stress" Tool: Install the stress tool to simulate heavy CPU load for stress testing.

I am using the stress tool here to increase CPU utilization

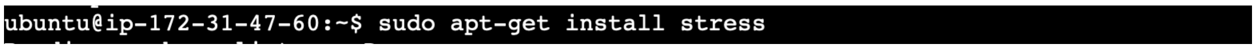
(The stress tool is a command-line utility used for testing the robustness of computer systems, particularly their stability under heavy loads. It's designed to simulate high CPU, memory, I/O, and disk usage to stress-test the system and uncover potential weaknesses or issues)
Steps to install stress tool

sudo apt update



```
ubuntu@ip-172-31-47-60:~$ sudo apt update
```

sudo apt-get install stress



```
ubuntu@ip-172-31-47-60:~$ sudo apt-get install stress
```

9.Conducting Stress Testing: Use the stress tool to increase CPU utilization and observe system behavior.

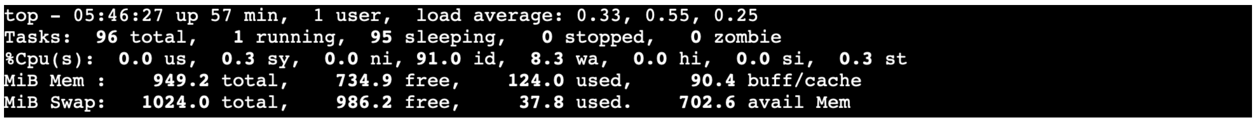
command:

stress --cpu 8 --io 4 --vm 4 --vm-bytes 1024M --timeout 01m

This command aims to increase CPU utilization to more than 75%.

Monitor the system to observe the transfer of load from RAM to swap memory using top command.

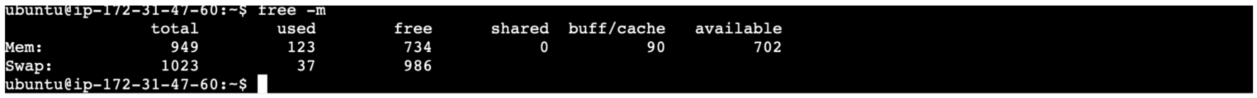
the resulting readings.



```
top - 05:46:27 up 57 min,  1 user,  load average: 0.33, 0.55, 0.25
Tasks:  96 total,   1 running,  95 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.3 sy,  0.0 ni, 91.0 id,  8.3 wa,  0.0 hi,  0.0 si,  0.3 st
MiB Mem :    949.2 total,    734.9 free,    124.0 used,     90.4 buff/cache
MiB Swap:   1024.0 total,    986.2 free,     37.8 used.    702.6 avail Mem
```

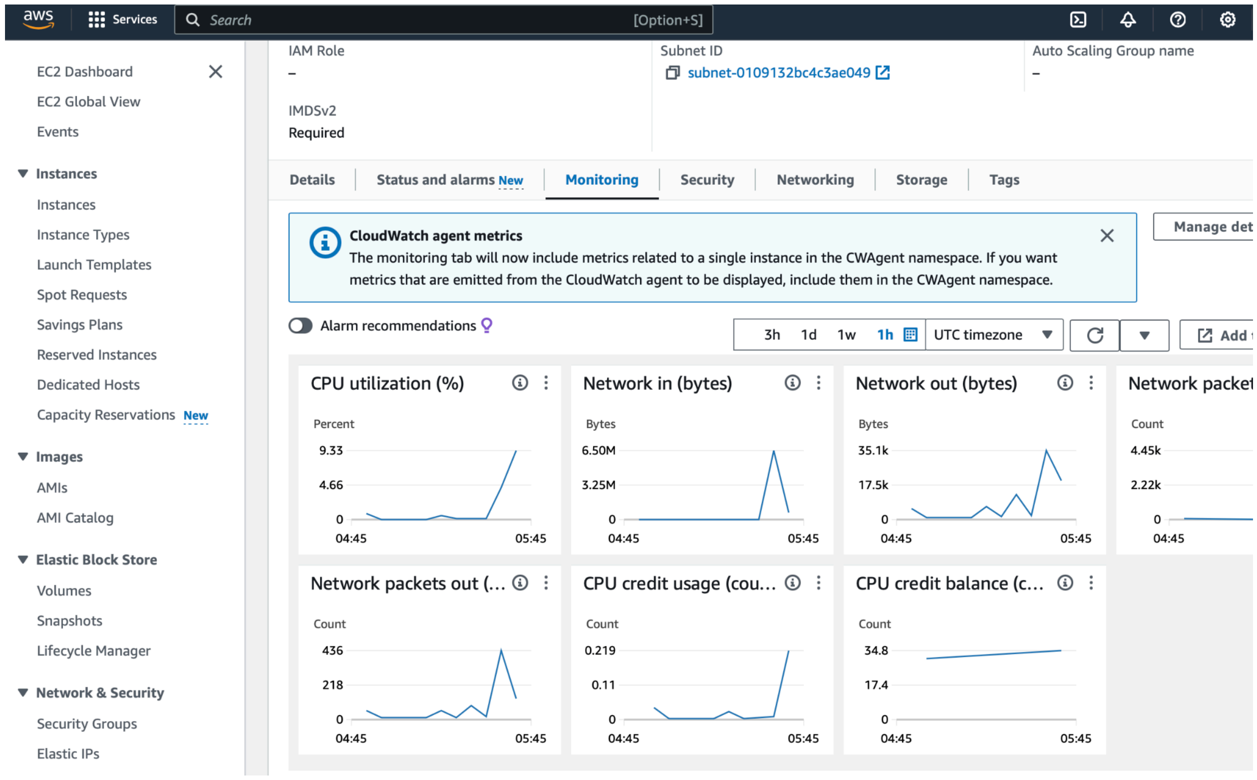
10.Monitoring System Performance: Utilize AWS EC2 Dashboard under the Monitoring tab to monitor CPU utilization.

As we can see the swap memory has taken load off ram memory and the readings are stable.



```
ubuntu@ip-172-31-47-60:~$ free -m
               total        used        free      shared  buff/cache   available
Mem:             949         123         734           0          90         702
Swap:           1023          37         986
ubuntu@ip-172-31-47-60:~$ 
```

If you look at the AWS EC2 Dashboard under the Monitoring tab, you can see that the CPU utilization has increased



IAM Role
–

Subnet ID
subnet-0109132bc4c3ae049

Auto Scaling Group name
–

IMDSv2
Required

CloudWatch agent metrics
The monitoring tab will now include metrics related to a single instance in the CWAgent namespace. If you want metrics that are emitted from the CloudWatch agent to be displayed, include them in the CWAgent namespace.

Conclusion:

Configuring swap memory on Linux servers, especially in cloud environments like AWS EC2, is essential for optimizing system performance and ensuring stability under varying workloads. By following the steps outlined in this guide, you can effectively manage system memory and enhance overall server performance. Additionally, stress testing provides valuable insights into system behavior under heavy loads, enabling proactive performance optimization.