How to add swap on Ubuntu

One of the easiest way of increasing the responsiveness of your server (and guarding against out of memory errors in your applications) is to add some swap space. Swap is an area on a hard drive that has been designated as a place where the operating system can temporarily store data that it can no longer hold in RAM. This gives you the ability to increase the amount of information that your server can keep in its working “memory”. The space on the hard drive will be used mainly when space in RAM is no longer sufficient for data.The information written to disk will be slower than information kept in RAM. The OS will prefer to keep running application data in memory and use swap for the older data. Having swap space as a fall back for when your system’s RAM is depleted is a good safety net.

A more detailed (yet simple) explanation here

Follow these easy steps to add some swap space:

First, check the OS to see if we already have some swap space available. We can have multiple swap files or swap partitions, but generally one should be enough. We can see if the system has any configured swap by typing:

```
sudo swapon -s
Filename                Type        Size    Used    Priority
```

If you only get back the header of the table, you do not currently have any swap space enabled. Another way of checking for swap space is with the free utility, which shows us system memory usage. We can see our current memory and swap usage in Megabytes by typing:

```
free -m
             total       used       free     shared    buffers     cached
Mem:           490        479         10         64         11        124
-/+ buffers/cache:        343        146
Swap:            0          0          0
```

The typical way of allocating space for swap is to use a separate partition devoted to the task. However, altering the partitioning scheme is not always possible. We can just as easily create a swap file that resides on an existing partition. Before we do this, we should be aware of our current disk usage. We can get this information by typing:

```
df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda         20G  3.4G   16G  24% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            235M  4.0K  235M   1% /dev
tmpfs            50M  324K   49M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            246M     0  246M   0% /run/shm
none            100M     0  100M   0% /run/user
```

As you can see on the first line, our hard drive partition has 20 Gigabytes available, so we have some

disk space to work with. Although there are many opinions about the appropriate size of a swap space, it really depends on your personal preferences and your application requirements. Generally, an equal to or double the amount of RAM on your system is a good starting point. I will create a swap space of 1 Gigabytes to match my system's RAM.

We will create a file called swapfile in our root (/) directory. The file must allocate the amount of space we want for our swap file. There are two main ways of doing this:

The slower way

Create a file with preallocated space by using the dd command. This versatile disk utility writes from one location to another location. We can use this to write zeros to the file from a special device in Linux systems located at /dev/zero that just spits out as many zeros as requested. We specify the file size by using a combination of bs for block size and count for the number of blocks. What we assign to each parameter is almost entirely arbitrary. What matters is what the product of multiplying them turns out to be. For instance, we're looking to create a 1 Gigabyte file. We can do this by specifying a block size of 256 Megabytes and a count of 4:

```
dd if=/dev/zero of=/swapfile bs=256M count=4
4+0 records in
4+0 records out
1073741824 bytes (1.1 GB) copied, 8.4487 s, 129.7 MB/s
```

Double check your command before pressing ENTER because this has the potential to destroy data if you point the of to the wrong location. You can check that the file has been created and allocated:

```
ls -lh /swapfile
-rw------- 1 root root 1.0G Dec  8 05:08 /swapfile
```

The faster way

The quicker way of getting the same file is by using the fallocate program. This command creates a file of a preallocated size instantly, without actually having to write dummy contents. We can create a 1 Gigabyte file by typing:

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

At this time, swap file is created, but our system does not know that this is supposed to be used for swap. We need to tell our system to format this file as swap and then enable it. Before we do that though, we need to adjust the permissions on our file so that it isn't readable by anyone besides root. Allowing other users to read or write to this file would be a huge security risk. We can lock down the permissions by typing:

```
sudo chmod 600 /swapfile
ls -lh /swapfile
-rw------- 1 root root 1.0G Apr 28 17:19 /swapfile
```

As you can see, only the columns for the root user have the read and write flags enabled. Now that our file is more secure, we can tell our system to set up the swap space by typing:

```
sudo mkswap /swapfile
Setting up swapspace version 1, size = 1024000 KiB
no label, UUID=e2f1e9cf-c0a9-4ed4-b8ab-714b8a7d6944
```

Our file is now ready to be used as a swap space. We can enable this by typing:

```
sudo swapon /swapfile
```

We can verify that the procedure was successful by checking whether our system reports swap space now:

```
swapon -s
Filename                                Type            Size    Used
Priority
/swapfile                               file            1048572 162184  -1

free -m
             total       used       free     shared    buffers     cached
Mem:           490        354        135         35          8         88
-/+ buffers/cache:        257        232
Swap:         1023        158        865
```

Our swap has been set up successfully and our operating system will begin to use it as necessary (mine did it almost immediately).

We have our swap file enabled, but when we reboot, the server will not automatically enable the file. We can change that though by modifying the fstab file. Edit the file with root privileges, and at the bottom of the file, add a line that will tell the operating system to automatically use the file you created:

```
/swapfile none swap sw 0 0
```

Save and close the file when you are finished.

There are a few options that you can configure that will have an impact on your system's performance when dealing with swap. The swappiness parameter configures how often your system swaps data out of RAM to the swap space. This is a percentage (a value between 0 and 100). With values close to zero, the kernel will not swap data to the disk unless absolutely necessary. Values that are closer to 100 will try to put more data into swap in an effort to keep more RAM space free. Depending on your applications' memory profile or what you are using your server for, this might be better in some cases. Remember, interactions with the swap file are "expensive" in that they take a lot longer than interactions with RAM and they can cause a significant reduction in performance. Telling the system not to rely on the swap much will make your system faster. We can see the current swappiness value by typing:

```
cat /proc/sys/vm/swappiness
60
```

For a Desktop, a swappiness setting of 60 is not a bad value. For a VPS system, we'd probably want to move it closer to 0. 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
vm.swappiness = 10
```

Again, this setting will persist until the next reboot. We can set this value automatically at restart by adding the line to the bottom of the /etc/sysctl.conf file:

```
vm.swappiness=10
```

Another related value that you might want to modify is the vfs_cache_pressure. This setting configures how much the system will choose to cache inode and dentry information over other data. Basically, this is access data about the filesystem. This is generally very costly to look up and very frequently requested, so it's an excellent thing for your system to cache. You can see the current value by querying the proc filesystem again:

```
cat /proc/sys/vm/vfs_cache_pressure
100
```

As it is currently configured, our system removes inode information from the cache too quickly. We can set this to a more conservative setting like 50 by typing:

```
sudo sysctl vm.vfs_cache_pressure=50
vm.vfs_cache_pressure = 50
```

Yet again, this is only valid for our current session. We can change that by adding it to our configuration file like we did with our swappiness setting:

```
vm.vfs_cache_pressure = 50
```

Swap space is incredibly useful in avoiding some common problems. and following these steps will give you some breathing room in terms of your RAM usage. If you are running into out of memory errors, or if you find that your system is unable to use the applications you need, the best solution is to optimize your application configurations or upgrade your server. Configuring swap space, however, can give you more flexibility and can help buy you time on a less powerful server.

The fastest way

In short

```
fallocate -l 4G /swapfile
chmod 600 /swapfile
mkswap /swapfile
swapon /swapfile
swapon --show
sysctl vm.swappiness=10
```

```
sysctl vm.vfs_cache_pressure=50
echo '/swapfile none swap sw 0 0' | tee -a /etc/fstab
echo 'vm.swappiness=10' | tee -a /etc/sysctl.conf
echo 'vm.vfs_cache_pressure=50' | tee -a /etc/sysctl.conf
```