# How to Limit Resources in Docker?

written by sysadmin | 14 May 2025

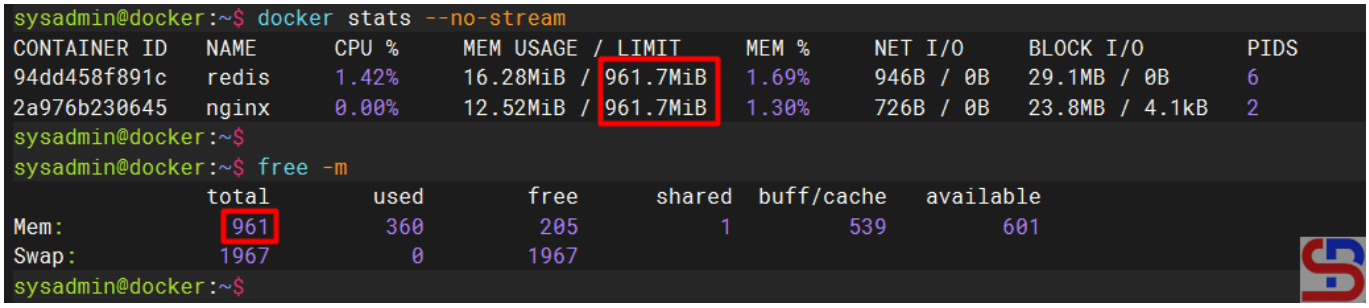
The previous article explained how to monitor an entire container in Docker. In addition to monitoring, you should also limit the resource usage of each container so that server performance remains in good condition.

## Problem

How to limit resources in Docker?

## Solution

If you run the docker stats command, the command will display the resource container as shown in the image below:



Display stats of the Docker

As you can see in the image above, there are 2 containers running, and both containers have a memory limit of 941 MB, where the memory size is the size of the memory of the server. This is dangerous because the application in the container can use all the memory or CPU on the server, causing the server to run abnormally. Therefore, you should limit the use of resources in the container.

### A. RAM limitation

There are 2 types of memory limitations in Docker:

- The hard limit is a maximum value that cannot be

exceeded. When a container exceeds a hard memory limit, Docker takes aggressive actions such as terminating the container so that there will be an OOM or Out Of Memory error in the container. The options used in Docker are **--memory** or **-m**.

- The soft limit is a limit that can be temporarily exceeded. When a soft limit is reached, Docker warns the user but does not take immediate action. The option used is **--memory-reservation**.

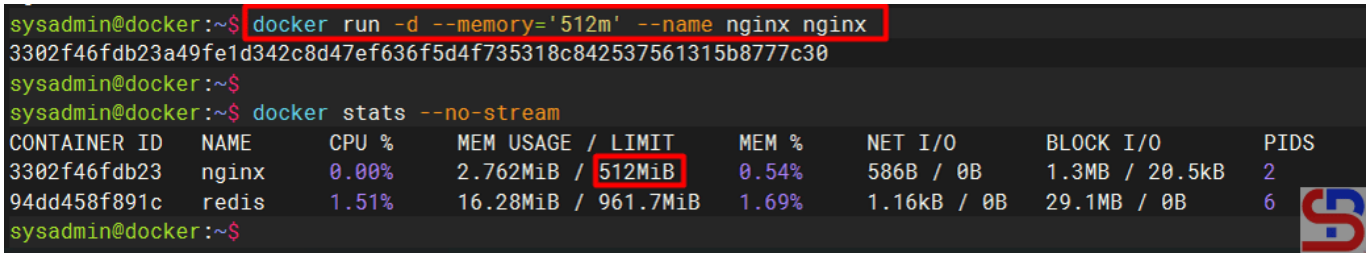
If you use swap on the server, you can use the **--memory-swap** option to allocate available swap memory to the container. This swap memory must be larger than the hard limit, usually twice larger than the hard limit. If you want to use a percentage for memory swap, use the **--memory-swappiness** option.

To limit the memory on the new container, use the format below:

```
docker run -d --memory='hard_limit_value' --name docker_name docker-image
```

For example, if you want to limit the memory on a container of 512 MB with a soft limit of 256 MB, then I run the command below:

```
docker run -d --memory='512m' --name nginx nginx
```



Run Docker with limited memory

You can immediately change the container memory when the container is running using the format below:

```
docker update docker_name --memory='hard_limit_value'
```

For example, I want to change the Redis container memory from 971 MB to 256 MB using the command below:

```
sysadmin@docker:~$ docker stats --no-stream
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O       BLOCK I/O        PIDS
3302f46fdb23   nginx     0.00%     2.762MiB / 512MiB     0.54%     936B / 0B     1.3MB / 20.5kB   2
94dd458f891c   redis     1.27%     16.28MiB / 961.7MiB   1.69%     1.23kB / 0B   29.1MB / 0B      6
sysadmin@docker:~$
sysadmin@docker:~$ docker update redis --memory='256m'
Error response from daemon: Cannot update container 94dd458f891c08d8e0a15eb00878fc83e5c66660d6af8beb0815a3a9040c57f7: Memo
ry limit should be smaller than already set memoryswap limit, update the memoryswap at the same time
sysadmin@docker:~$
```

Error when updating memory

But when running the command, there is an error as below:

Error Response from Daemon: Cannot Update Container 94DD458F891C08D8E0A15EB00878FC83E5C6660D6AF8Beb0815A3A9040C57F7: Memory Limit Should Be Smaller Than Already Set Time

To overcome the error, update the memory container swap, whose value must be greater than the memory value. Therefore, use the command below to increase the memory of a container:

```
docker update redis --memory='256m' --memory-swap='512mb'
```

And the container memory value should have changed as shown below:

```
sysadmin@docker:~$ docker stats --no-stream
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O       BLOCK I/O        PIDS
3302f46fdb23   nginx     0.00%     2.762MiB / 512MiB     0.54%     1.01kB / 0B   1.3MB / 20.5kB   2
94dd458f891c   redis     1.35%     16.28MiB / 961.7MiB   1.69%     1.3kB / 0B    29.1MB / 0B      6
sysadmin@docker:~$
sysadmin@docker:~$ docker inspect redis | grep Memory
            "Memory": 0,
            "MemoryReservation": 0,
            "MemorySwap": 0,
            "MemorySwappiness": null,
sysadmin@docker:~$
sysadmin@docker:~$ docker update redis --memory='256m'
Error response from daemon: Cannot update container 94dd458f891c08d8e0a15eb00878fc83e5c66660d6af8beb0815a3a9040c57f7: Memo
ry limit should be smaller than already set memoryswap limit, update the memoryswap at the same time
sysadmin@docker:~$
sysadmin@docker:~$ docker update redis --memory='256m' --memory-swap='512mb'
redis
sysadmin@docker:~$
sysadmin@docker:~$ docker inspect redis | grep Memory
            "Memory": 268435456,
            "MemoryReservation": 0,
            "MemorySwap": 536870912,
            "MemorySwappiness": null,
sysadmin@docker:~$
sysadmin@docker:~$ docker stats --no-stream
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O       BLOCK I/O        PIDS
3302f46fdb23   nginx     0.00%     2.762MiB / 512MiB     0.54%     1.01kB / 0B   1.3MB / 20.5kB   2
94dd458f891c   redis     1.56%     16.28MiB / 256MiB     6.36%     1.3kB / 0B    29.1MB / 0B      6
sysadmin@docker:~$
```

```
Update the memory in a container
```

**B. CPU limitation**

Before you limit the CPU in the container, you need to know how many CPU cores there are on your server by using the command below:

```
nproc
```

To limit the CPU used in the container, use the **--cpus** option to determine how many CPU cores can be used in a container. If your server has two CPUs and you set --cpus=’1.5′, the container is guaranteed at most one and a half of the CPUs and this is the equivalent of setting --cpu-period=’100000′ and --cpu-quota=’150000′ (cpu-period is used with cpu-quota to configure the CPU scheduler with defaults to 100000 microseconds and cpu-quota is used with cpu-period to configure the CPU scheduler). Use the format below to limit the CPU in the container:

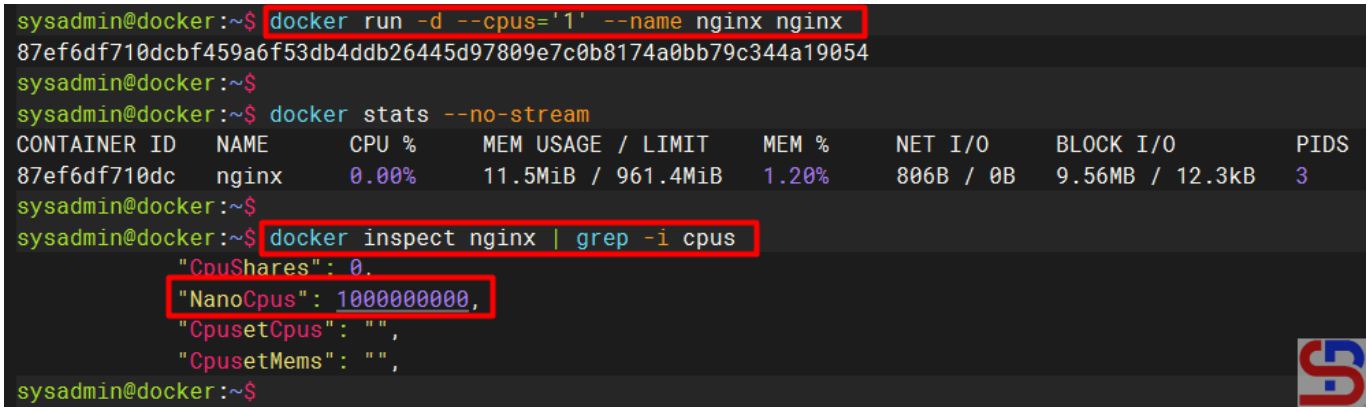
```
docker run -d --cpus='cpu_value' --name docker_name docker_image:tag
```

You can see the details of the CPU used by a container by using the format below:

```
docker inspect nginx | grep -e Cpu -e cpu
```

For example, I want to limit the CPU to 1, so I run the command below:

```
docker run -d --cpus='1' --name nginx nginx
```



Run a container with a limited CPU

Warning

You have to be careful in determining the CPU limitations of a container because if you limit the CPU too much (for example, giving 0.5 CPU to the container even though the container can run using 1 CPU) or give a CPU limitation that is greater than the CPU the server uses (for example The server CPU only has 1 CPU but you give the container 2 CPUs) then the container will immediately turn off automatically.

You can use the **--cpu-shares** option for a container to control the share of CPU cycles available, whose default value is 1024. This option is like a soft limit option on memory, so if you run the command below:

```
docker run -d --cpu-shares='2048' --name webapp1 nginx
```

If there is more than 1 container on a host and CPU cycles are constrained, then the webapp1 container will receive 2x more CPU than the other containers. You can update the CPU in a running container using the format:

```
docker update docker_name --cpus='cpu_value'
```

For example, I have 2 CPUs on the server, and initially, I use all the CPUs in the webapp1 container. Then, I wanted to update the CPU on the container to 1.5, so I ran the command below:

```
docker update webapp1 --cpus='1.5' nginx
```

```
sysadmin@docker:~$ docker run --name nginx -d nginx
8e506e9c6f058fbca872888e4f1db06c30836dc17b67a7a6e69f873eec476167
sysadmin@docker:~$
sysadmin@docker:~$ docker inspect nginx | grep -i cpus
            "CpuShares": 0,
            "NanoCpus": 0,
            "CpusetCpus": "",
            "CpusetMems": "",
sysadmin@docker:~$
sysadmin@docker:~$ docker update nginx --cpus="1.5"
nginx
sysadmin@docker:~$
sysadmin@docker:~$ docker inspect nginx | grep -i cpus
            "CpuShares": 0,
            "NanoCpus": 1500000000,
            "CpusetCpus": "",
            "CpusetMems": "",
sysadmin@docker:~$
```

Update the CPU in a container

### C. HDD limitation

As of this writing (April 2025), the HDD limitation can only be used for **btrfs**, **overlay2**, **windowsfilter**, and **zfs** storage drivers. For the overlay2 storage driver, the size option is only available if the backing filesystem is **xfs** and mounted with the **pquota** mount option. Type the command below to see the Docker settings on the server:

```
docker info | grep -e 'Filesystem' -e 'Support' -e 'Storage'
```

Run the command above, and here is the result:

```
sysadmin@docker:~$ docker info | grep -e 'Filesystem' -e 'Support' -e 'Storage'
WARNING: bridge-nf-call-iptables is disabled
WARNING: bridge-nf-call-ip6tables is disabled
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
sysadmin@docker:~$
```

Check the filesystem

To limit the hard disk in a container, follow the format below:

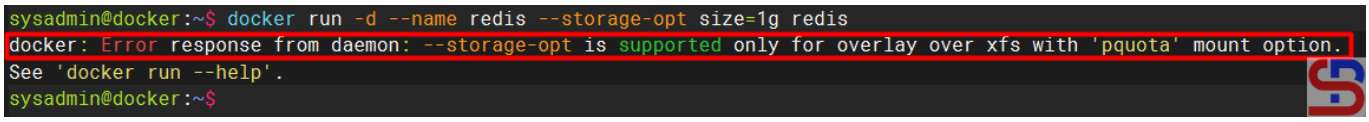
```
docker run -d --name nginx --storage-opt size=1g nginx
```

If I want to limit my hard disk in a container, I run the command below:

```
docker run -d --name nginx --storage-opt size=1g nginx
```

But, I have an error like the image below:

```
docker: Error response from daemon: –storage-opt is supported only for overlay pver xfs with ‘pquota’ mount option
```



Error when limiting HDD for a container

From the image above,  I can’t limit the HDD to the container because my Backing Filesystem in my server still uses extfs, not xfs. So, if I want to limit my HDD in my containers, I have to change my Backing Filesystem in my server. So I made an experiment by turning off Docker and deleting the Docker folder using the command below:

```
sudo systemctl stop docker
sudo rm -rf /var/lib/docker && sudo mkdir /var/lib/docker
```

Then I inserted an additional hard disk into the existing server, and then I converted the file system to xfs using the commands below:

```
sudo mkfs.xfs /dev/sdb1
sudo mount /dev/sdb1 /var/lib/docker
```

```
sysadmin@docker:~$ sudo mount /dev/sdb1 /var/lib/docker/
sysadmin@docker:~$
sysadmin@docker:~$ df -T
Filesystem                         Type  1K-blocks    Used Available Use% Mounted on
tmpfs                              tmpfs     98448    1112     97336   2% /run
/dev/mapper/ubuntu--vg-ubuntu--lv ext4    10218772 6110044   3568056  64% /
tmpfs                              tmpfs    492220       0    492220   0% /dev/shm
tmpfs                              tmpfs      5120       0      5120   0% /run/lock
/dev/sda2                          ext4     1768056   96560   1563364   6% /boot
tmpfs                              tmpfs     98444      12     98432   1% /run/user/1000
/dev/sdb1                          xfs     2030592   71980   1958612   4% /var/lib/docker
sysadmin@docker:~$
```

Create xfs filesystem

And I added to the **/etc/fstab** file the script below:

```
echo '/dev/sdb1 /var/lib/docker xfs defaults,quota,prjquota,pquota,gquota 0
0' | sudo tee -a /etc/fstab
```

I rebooted the server to test whether the fstab file settings were correct. After that, I tried to create a container by limiting the container's hard disk to 1GB using the command below:

```
docker run -d --name nginx --storage-opt size=1g nginx
```

```
sysadmin@docker:~$ docker info | grep -e 'Filesystem' -e 'Support' -e 'Storage'
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
WARNING: bridge-nf-call-iptables is disabled
WARNING: bridge-nf-call-ip6tables is disabled
sysadmin@docker:~$
sysadmin@docker:~$ docker run -d --name nginx --storage-opt size=1g nginx
8d609d92bcc7d2b6be5d0c3f888ad6f7d6135f05466ded4761cb6a6941c59a17
sysadmin@docker:~$
sysadmin@docker:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS     NAMES
8d609d92bcc7   nginx     "/docker-entrypoint.…"   10 seconds ago   Up 9 seconds    80/tcp    nginx
sysadmin@docker:~$
```

Limiting HDD in a container

## Note

You can combine more than one restriction above by using one command. For example, if you want to limit memory, CPU, and HDD in a container, then you can run the command below:

```
docker run -d --name webapp1 -m 512m --cpus=1.5 --storage-opt size=1g nginx
```

As far as I know, Sysadmin only limits the use of RAM and CPU in Docker, but rarely limits HDD in Docker. If you want to limit resources in Docker, you must discuss it with the developers so that there are no problems with the application in the future.

## References


phoenixnap.com
baeldung.com
docs.docker.com
nodramadevops.com
stackoverflow.com
hands-on.cloud
blogs.perficient.com
blog.devops.dev
youtube.com
reddit.com


---

# How to Monitor Containers in Docker?

written by sysadmin | 14 May 2025

After you run containers in Docker on your server or your Docker Host, you should monitor all existing containers to find out the performance of each container.

## Problem

How to monitor containers in Docker?

## Solution

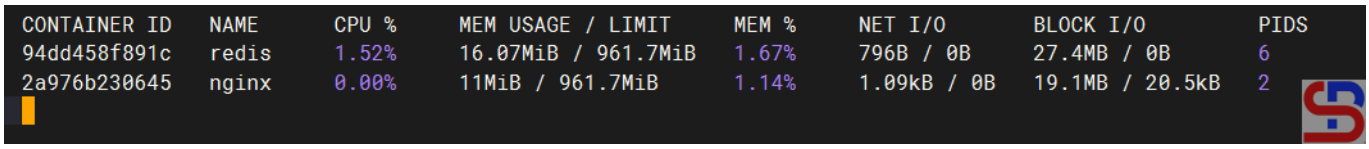
There are 2 methods for container monitors in Docker:

### A. Via CLI

In CLI, to monitor all containers in Docker, you can use the command:

```
docker stats
```

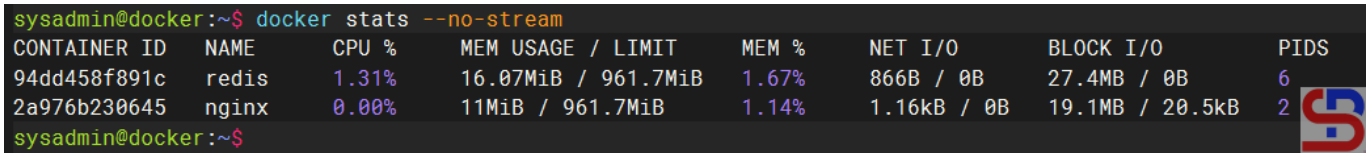
You will see the display as below:




Using the docker stats command

From the image above, you can see that the command displays the results in streaming, and to exit from the command above, press **Ctrl+Z** or **Ctrl-C**. If you don't want to display the results in streaming mode, then use the command below:

```
docker stats --no-stream
```




Display monitor containers in Docker without stream

### B. Via Website

If you want to monitor Docker via a website, you can use the Portainer tool. Portainer is a tool for managing containers through a browser that can support Docker host, Docker Swarm, Nomad, and Kubernetes. It has 2 components, namely Portainer Server, which is used to manage containers, networks, and environments, and Portainer Agent is the component installed on another Docker system to enable communication with the server. Portainer has 2 editions, namely Portainer Business Edition or PBE and Portainer Community Edition or PCE, where both editions at the time of this writing (April 2025) have version 2.27.3. This article will discuss how to install Portainer Community Edition.

Here are the steps:

**1. Create Docker Volume**

Type the command below to create a new volume in Docker:

```
docker volume create portainer_data
```

**2. Install Portainer**

Type the command below to install the latest version of Portainer:

```
docker run -d \
-p 8000:8000 \
-p 9443:9443 \
--name portainer \
--restart=always \
-v /var/run/docker.sock:/var/run/docker.sock \
-v portainer_data:/data portainer/portainer-ce
```

```
sysadmin@docker:~$ docker run -d \
-p 8000:8000 \
-p 9443:9443 \
--name portainer \
--restart=always \
-v /var/run/docker.sock:/var/run/docker.sock \
-v portainer_data:/data portainer/portainer-ce
Unable to find image 'portainer/portainer-ce:latest' locally
latest: Pulling from portainer/portainer-ce
e2e06b27b87e: Pull complete
1fed1531b45b: Pull complete
04de093ad5ed: Pull complete
86a7cce72d42: Pull complete
e09df2601140: Pull complete
eae3ebf29ea8: Pull complete
c12aa3fbd31a: Pull complete
f111bda3f9a6: Pull complete
81021110ed01: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:7f10a26bfdda3fc58295ea09b860117ecd86a642d66fb94ce1f27a4c221d4649
Status: Downloaded newer image for portainer/portainer-ce:latest
12496e61ee8addcff1a3a18ff95ade6802c951622fe6e4a6e2b23a030d6bb082
sysadmin@docker:~$
```

Install portainer

### 3. Check the Portainer

The following command can be used to determine whether Portainer is operating or not:

```
docker ps
```

```
sysadmin@docker:~$ docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          PORTS
                                                                                 NAMES
12496e61ee8a   portainer/portainer-ce   "/portainer"             4 minutes ago    Up 4 minutes    0.0.0.0:8000->8000/tcp,
:::8000->8000/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp, 9000/tcp   portainer
94dd458f891c   redis                    "docker-entrypoint.s…"   29 minutes ago   Up 29 minutes   6379/tcp
                                                                                 redis
2a976b230645   nginx                    "/docker-entrypoint.…"   31 minutes ago   Up 31 minutes   80/tcp
                                                                                 nginx
sysadmin@docker:~$
```

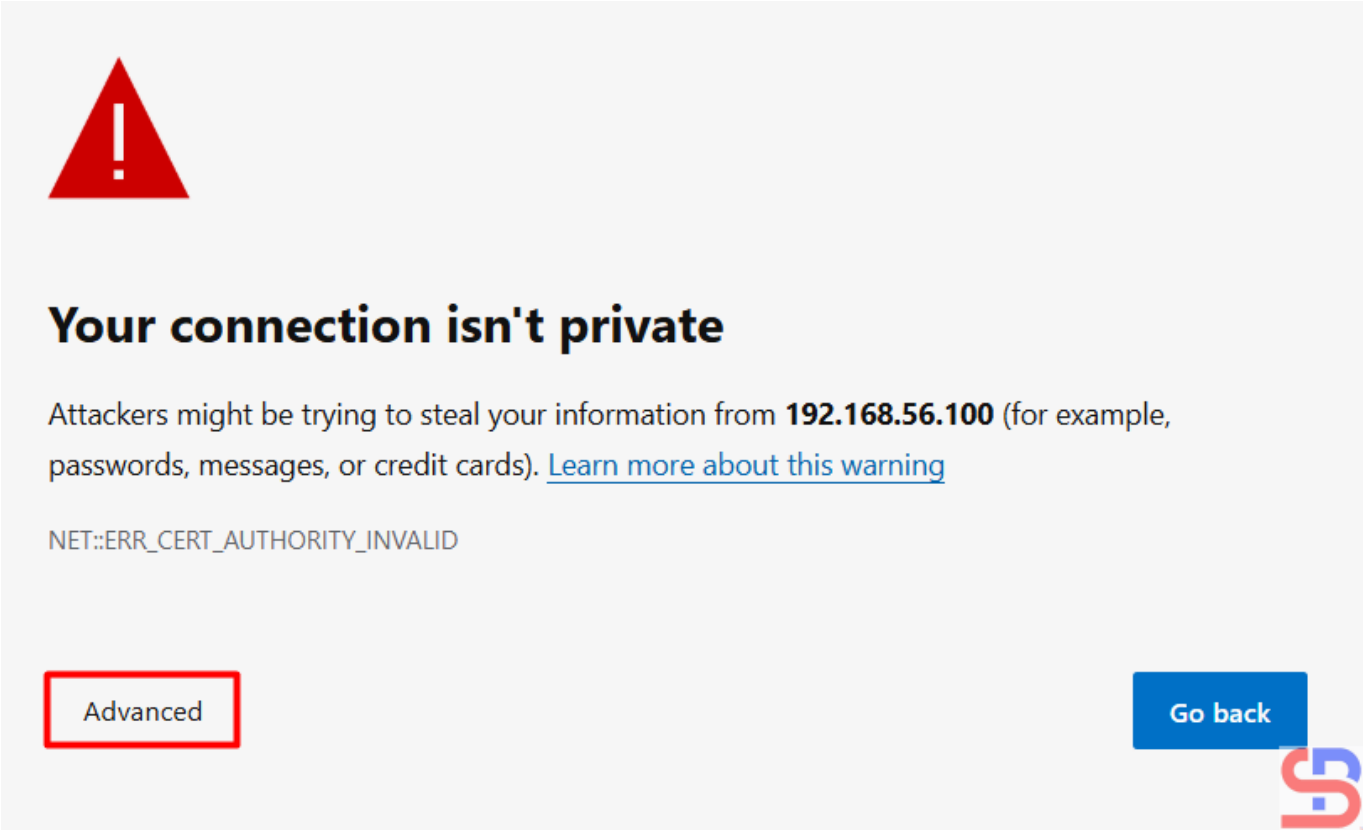
Check the container

### 4. Access the Portainer

After that, open your browser and type:

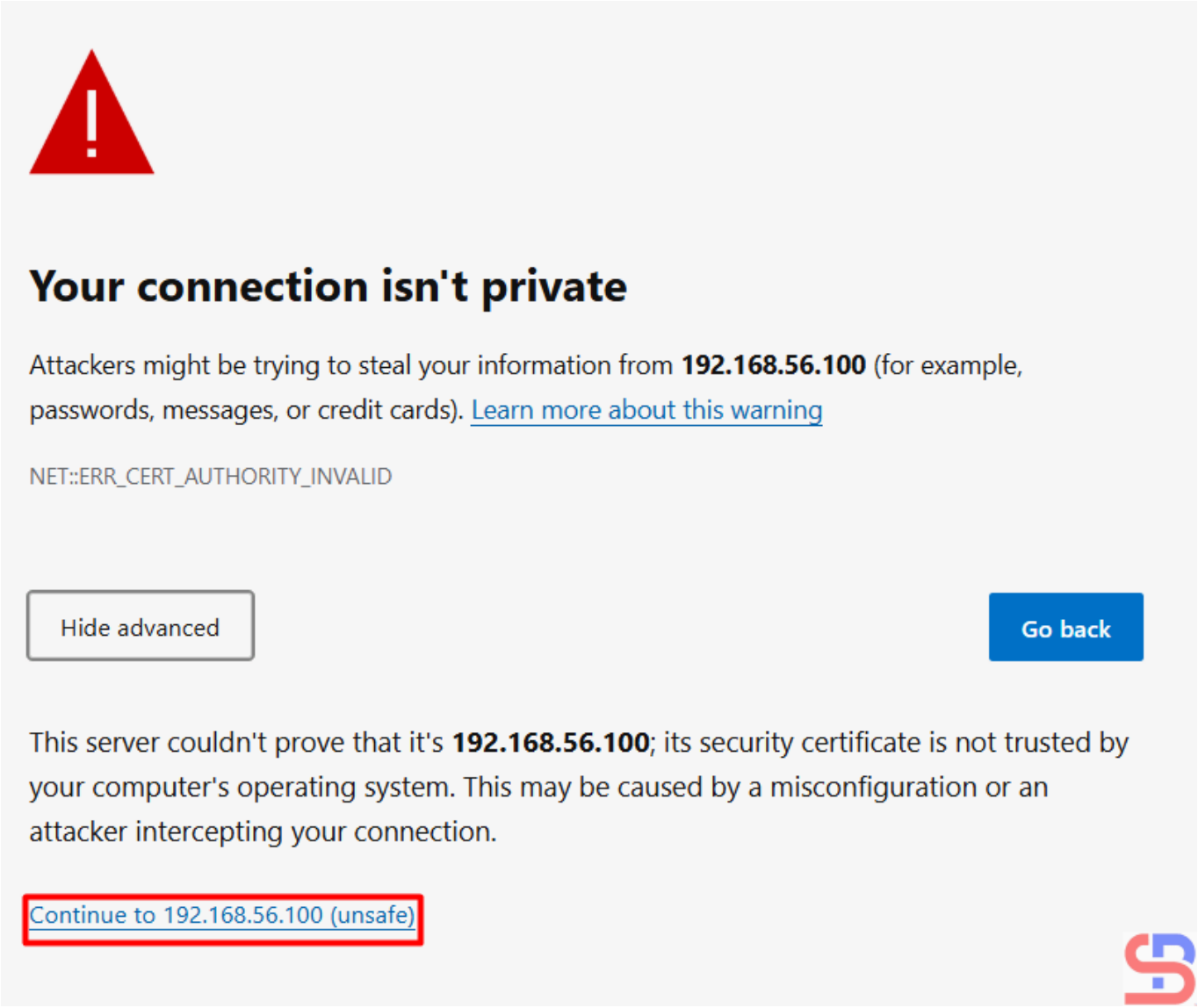
```
https://your_IP_server:9443
```

There will be an image like below:



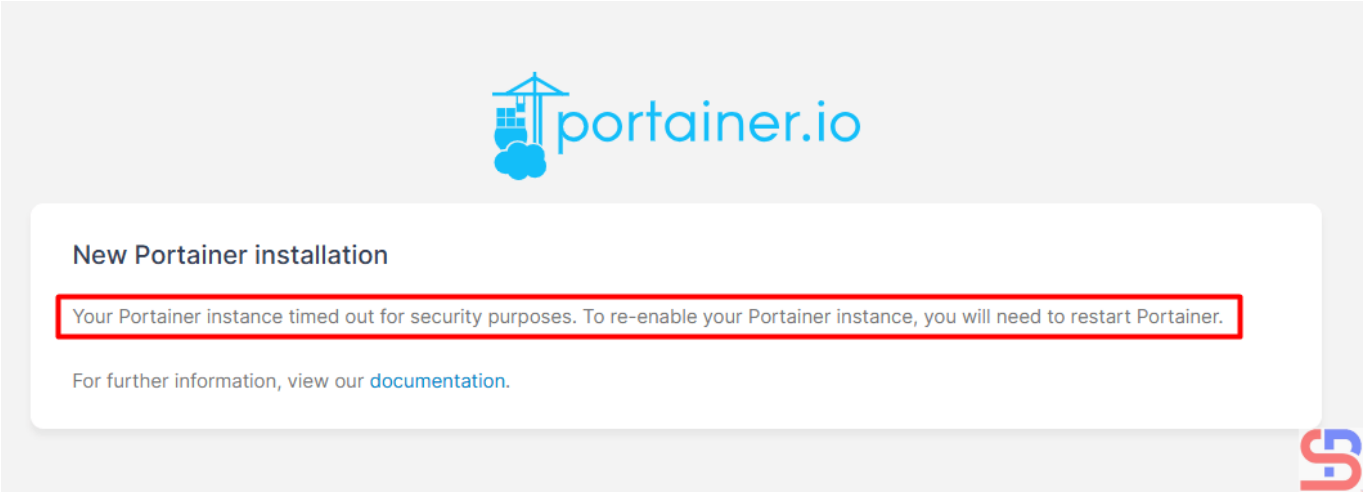

Click Advanced

A picture similar to the one below will appear when you click the **Advanced** button:




Click the unsafe link

Click the **unsafe** link in your browser, and then there will be an image like below:

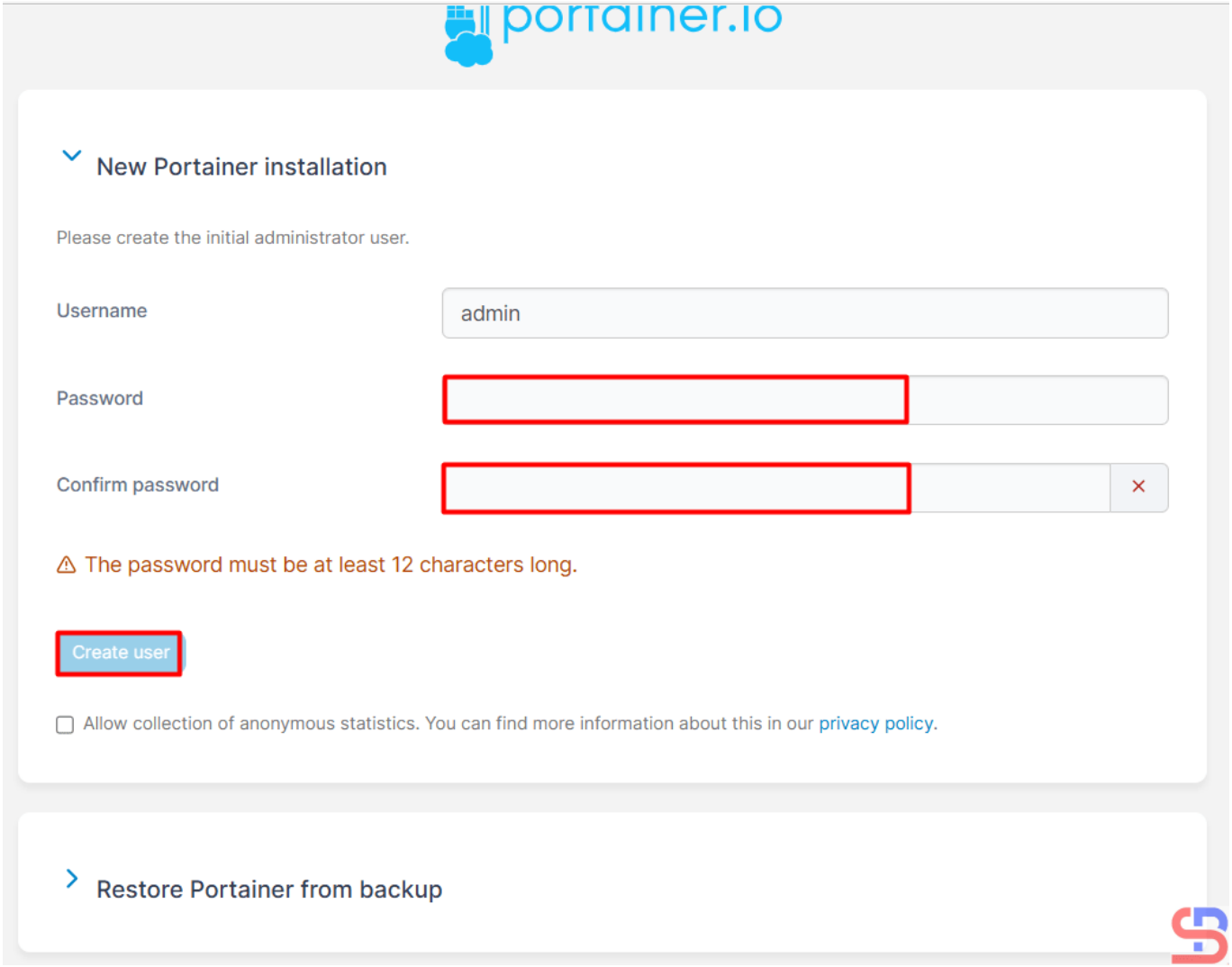



Restart Portainer

If you have an error like the picture above, then restart Portainer by running the command below:

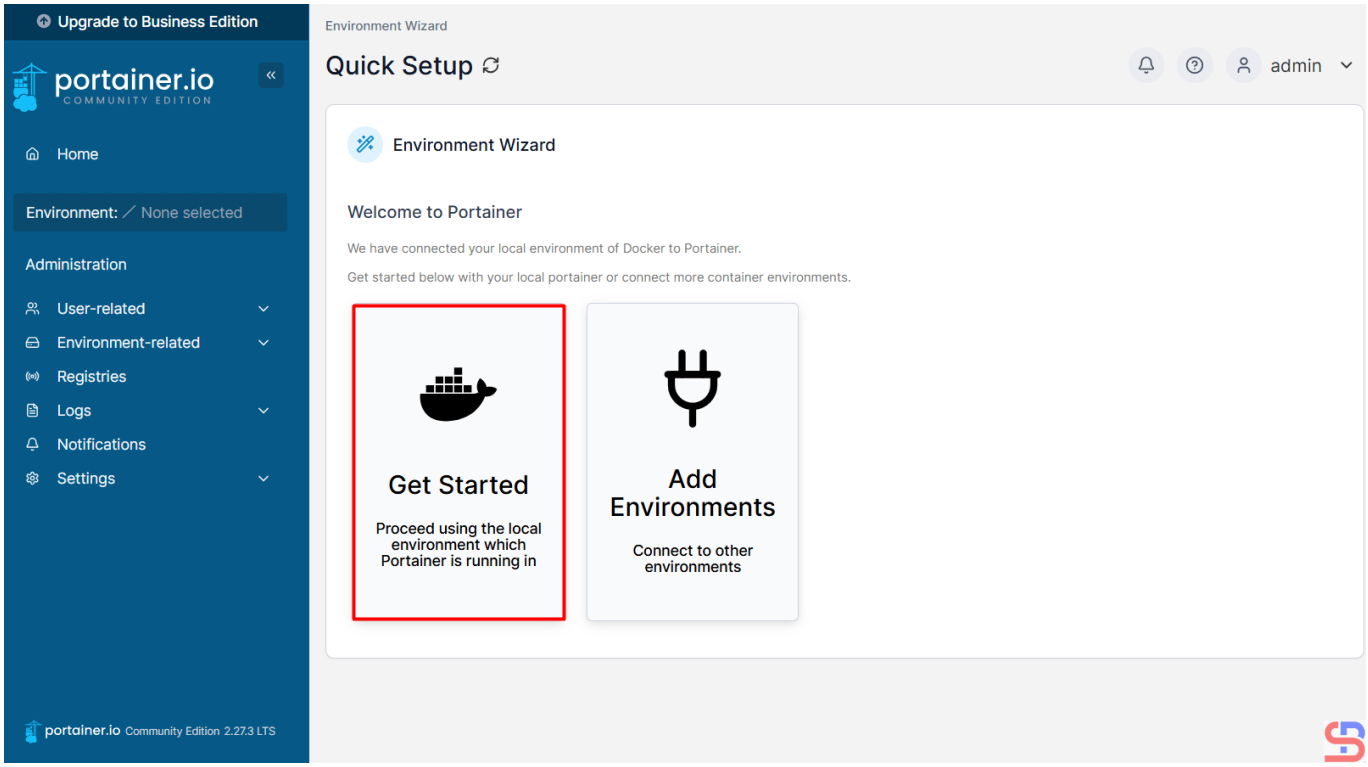
```
docker restart portainer
```

Enter the desired name and password, then click the **Create user** button, and you  will see an image below:



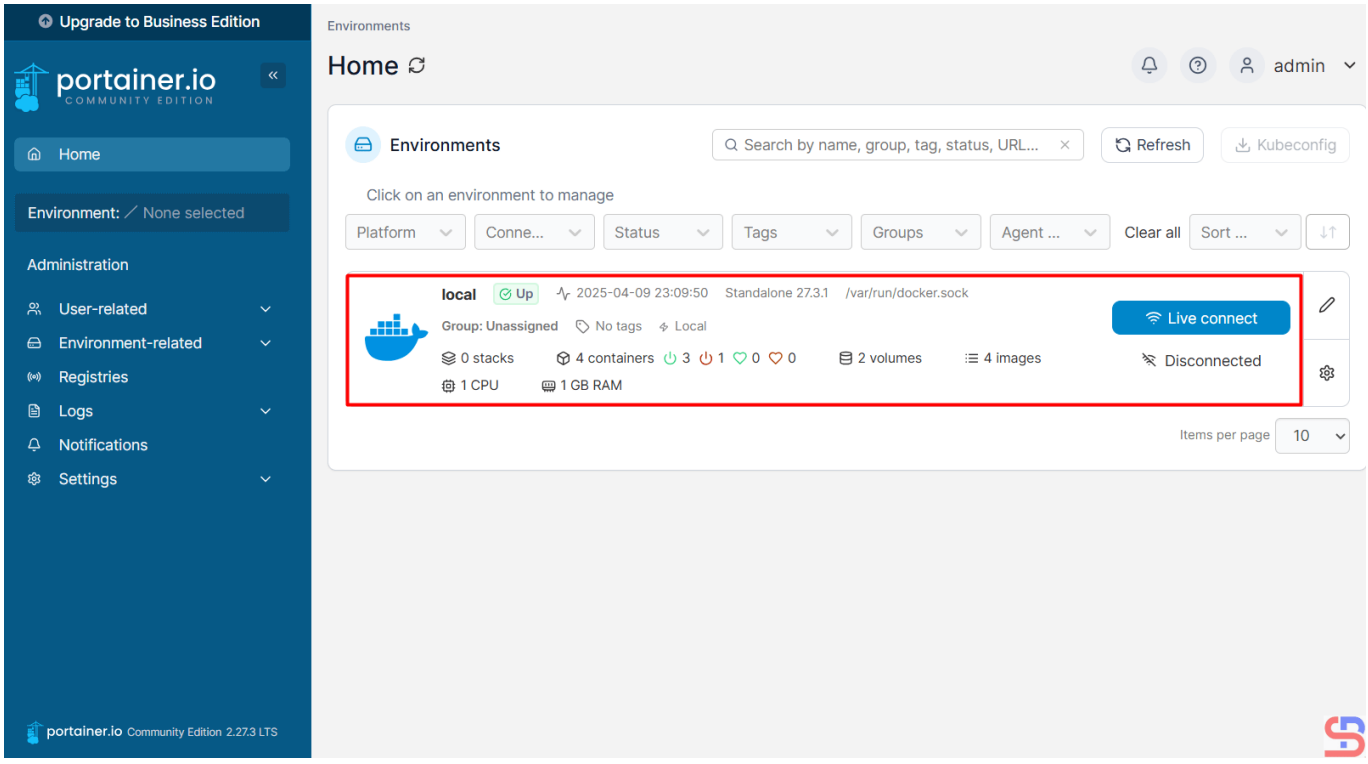

Write the username and password

The Portainer dashboard will appear. Click the **Get Started** box like in the above image, and there will be an image below:



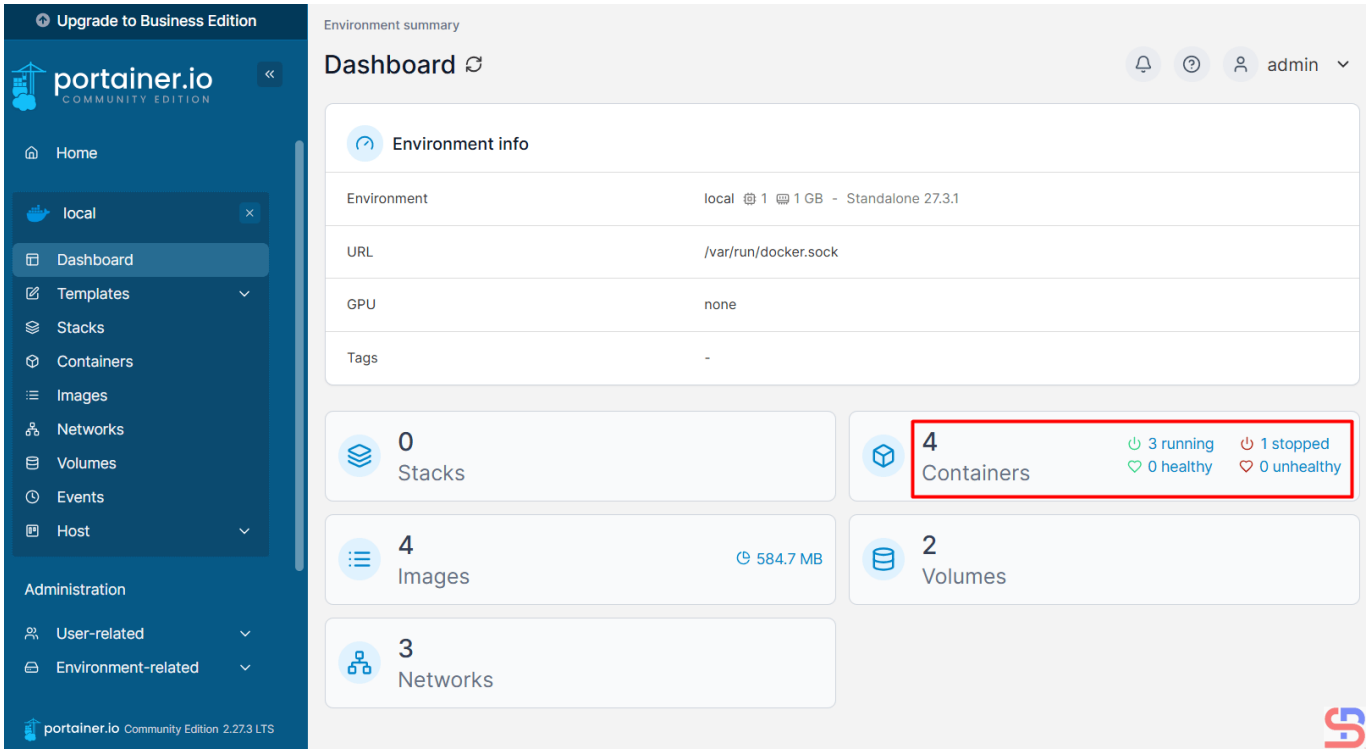

Click the Get Started box

There will be an image below:



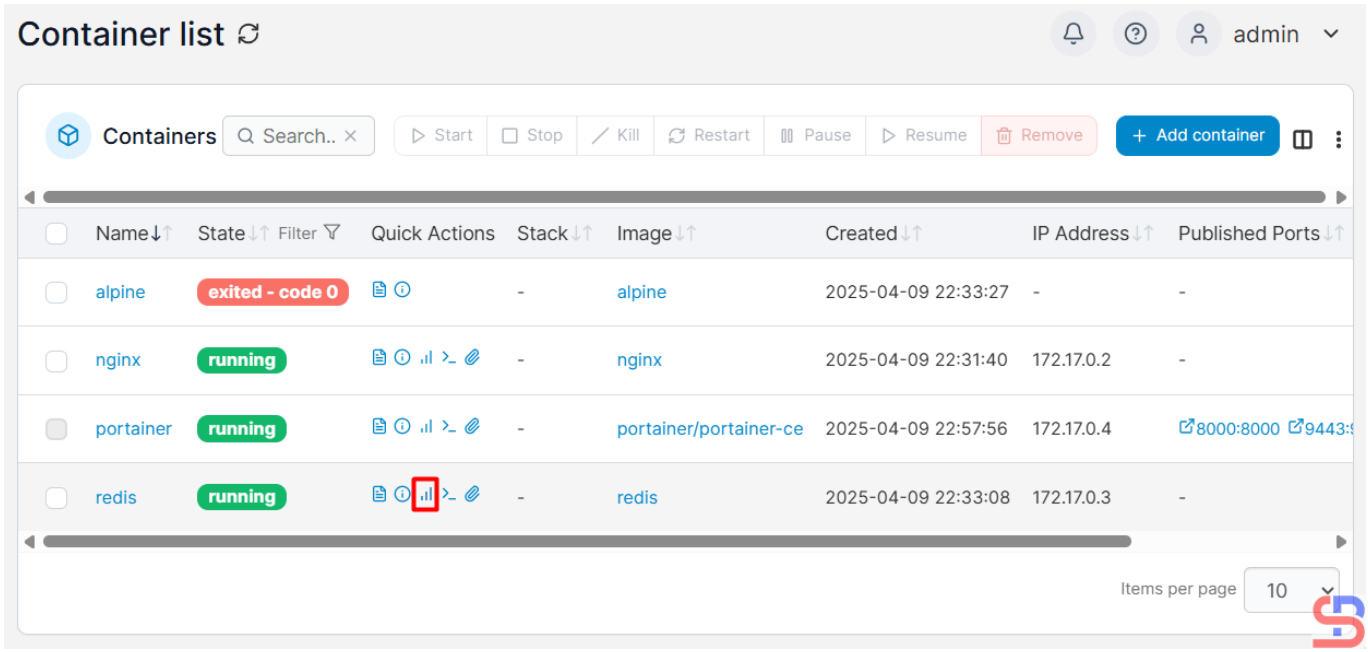

The Portainer dashboard

Click on the red box, and there will be an image below:




The information about the container(s) in Docker

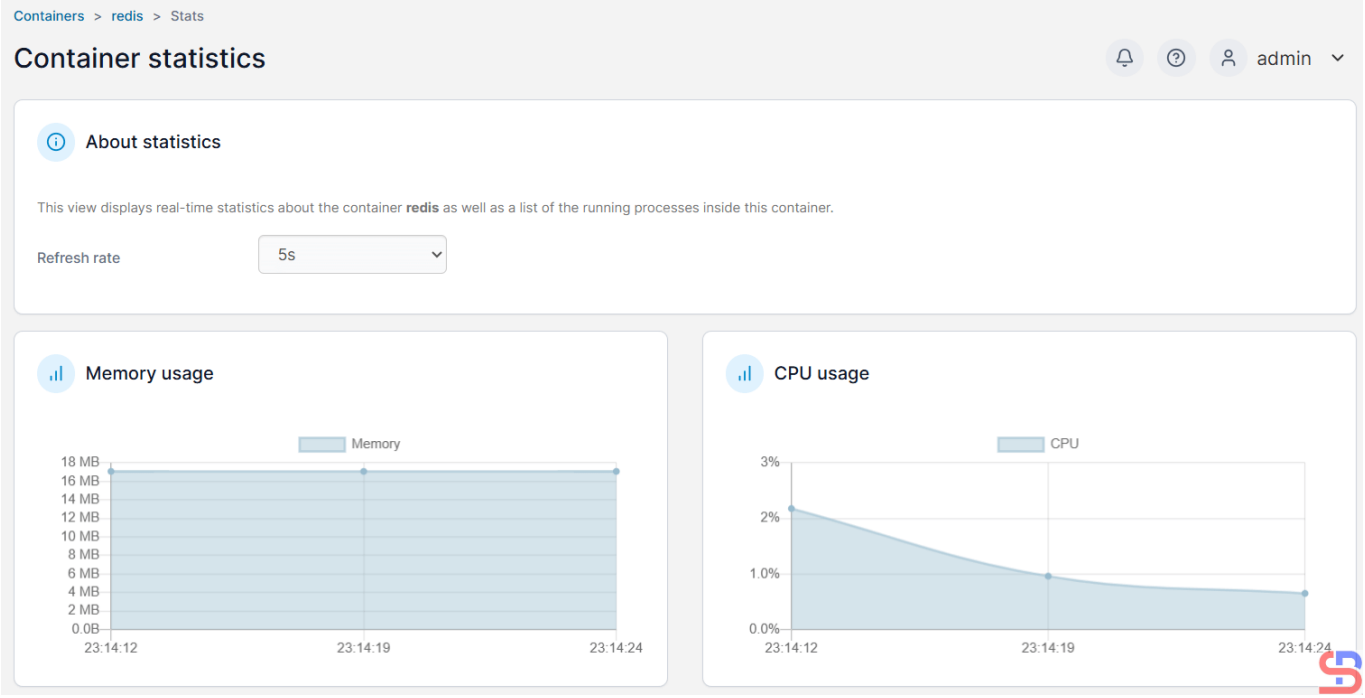
I have 4 containers in my server, but I want to detail each container, so I click in the red box, and there will be an image below:




The detailed information of each container

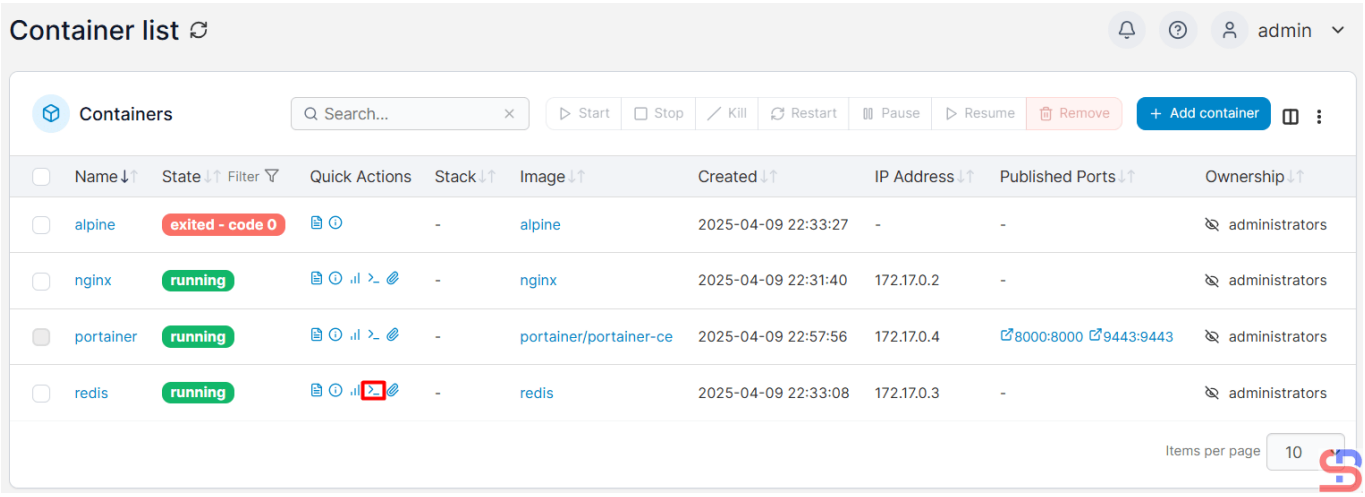
If I want to display the resource of the Redis instance, click the icon in the red box, and there will be a display

below:




The resource is displayed in a container

If I want to access a container, I click the icon like in the red box:

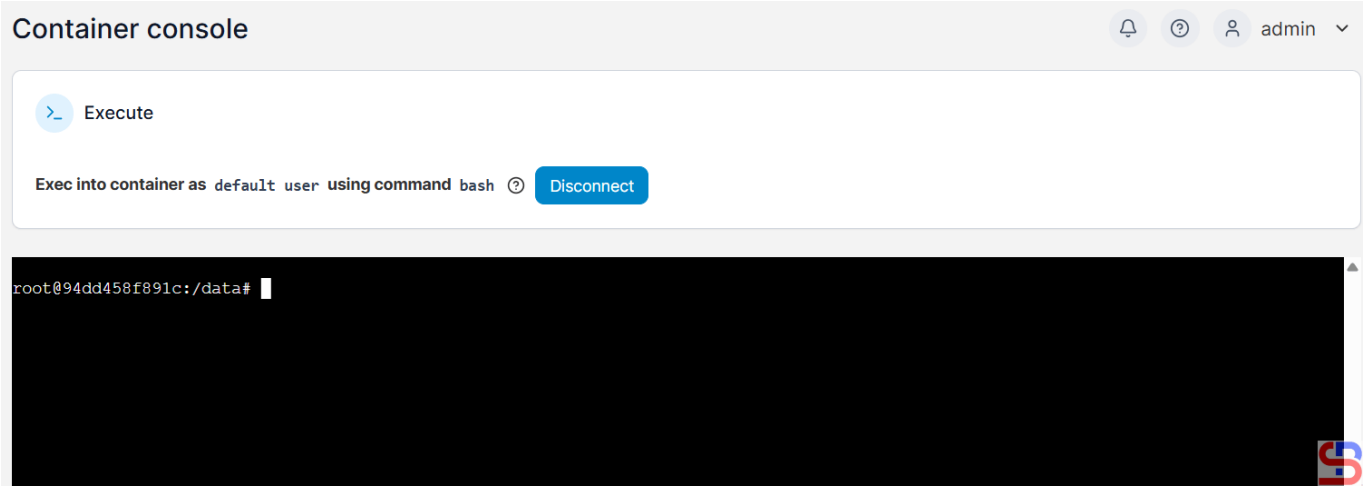



Click the icon to access the container

There will be a display like in the image below:

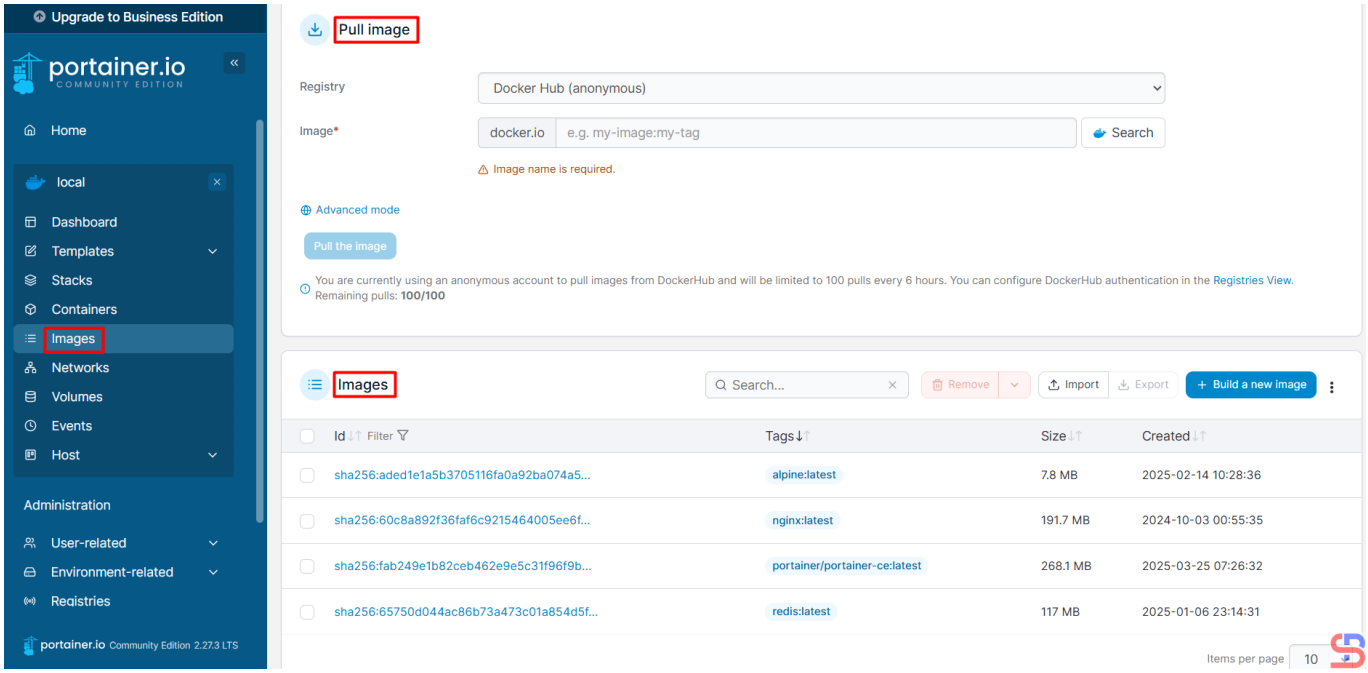
Click the Connect button after you choose the options

Select the command used in the container and select the desired user. After that, click the **Connect** button, and there will image like in the image below:



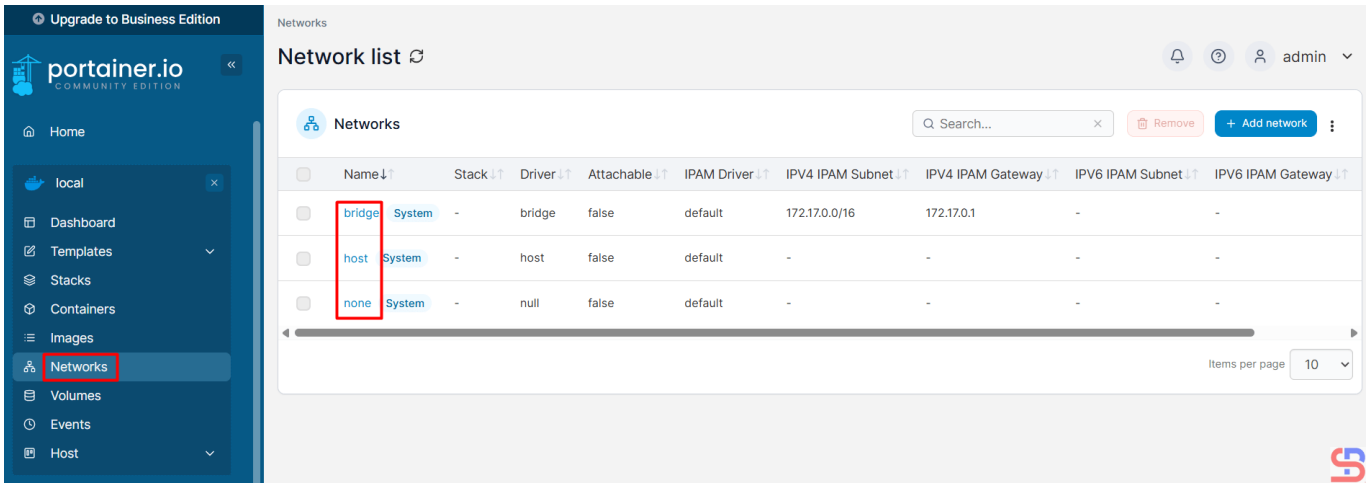
Access to the container

You can access the inside of the container and give the Linux command from your browser to the container. From this tool, you can see the images in your Docker when you click Images, like in the image below:




Display the Images

You can display the [Volume](#) in Docker after you click the Volumes, like in the image below:




Display the Volumes

## Note

If you want to monitor Docker on another server using Portainer, you have to install the agent using the command below:

```
curl -L https://downloads.portainer.io/agent-stack.yml -o agent-stack.yml && docker stack deploy --compose-file=agent-stack.yml portainer-agent
```

## References

youtube.dimas-maryanto.com
docs.portainer.io
phoenixnap.com
musaamin.web.id
letscloud.io

---

# How to Back up and Restore Docker Image(s)?

written by sysadmin | 14 May 2025

By default, if you want to create a Docker container on your server, you can download the required image directly using the **docker pull** command. But sometimes, there are some cases where you cannot download the image directly from the internet, and you have to back up the existing image and then restore the image to a server.

## Problem

How to back up and restore Docker image(s)?

## Solution

I have a server that, due to security issues, cannot be connected to the internet, while on the server, many applications run using Docker. Because it cannot directly download the Docker image, I have to download the Docker image on a server that is connected to the internet, and then the image will be installed on this server.

### A. Backup Docker image

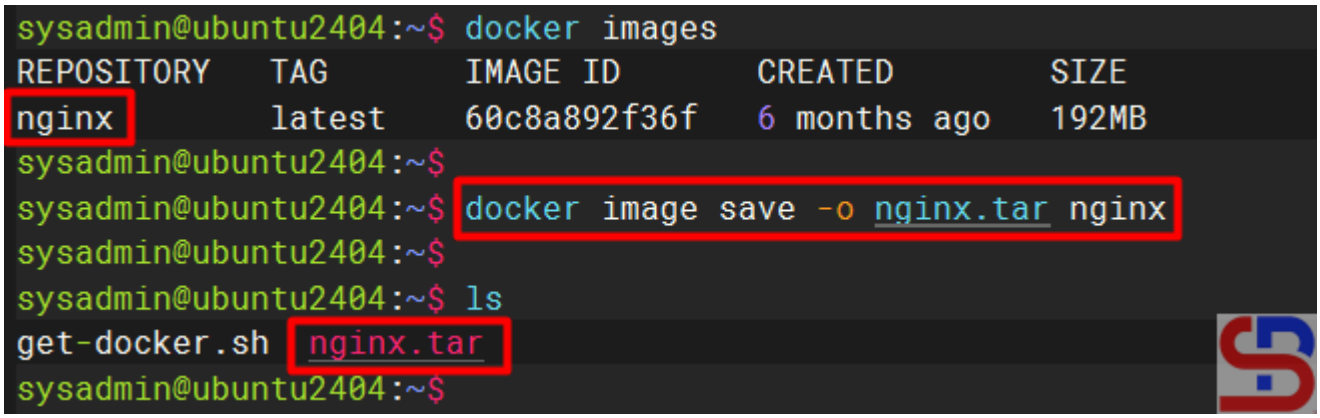
To back up a Docker image, use the format below:

```
docker image save -o image_name.tar image_name:tag
```

For example, if you want to back up the nginx image, then use the command below:

```
docker image save -o nginx.tar nginx
```

Wait until the process is complete, and if it is finished, the backup image file will be formed as shown below:



Back up one Docker image

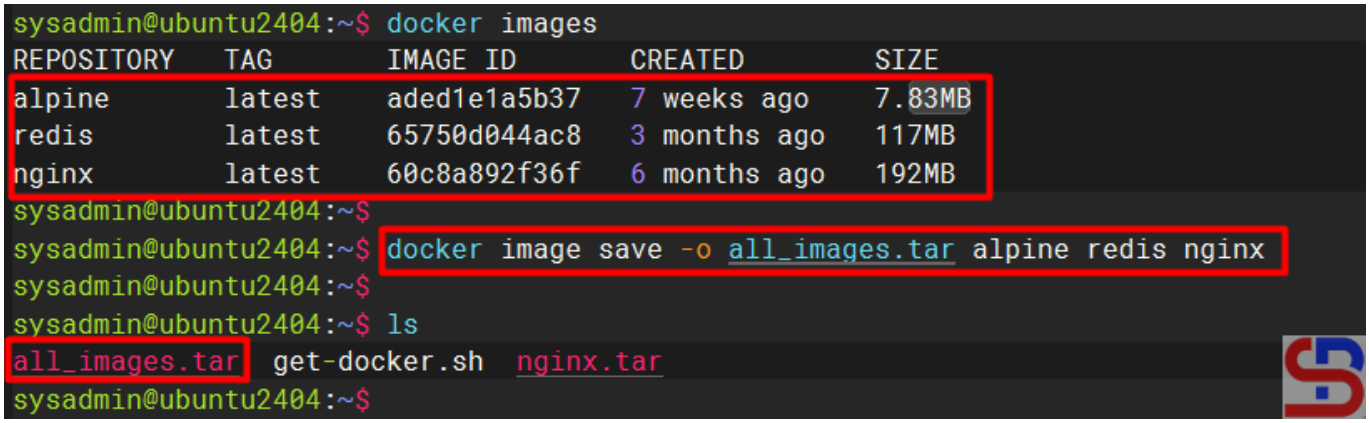
But use the format below if you want to back up more than one image:

```
docker image save -o image1_name.tar image2_name:tag ...
```

I have more than one Docker image on my server, so I want to back up all the images so I can run the images on my other server, which is not connected to the internet. Then use the command below to back up the docker image of more than one Docker image:

```
docker image save -o all_images.tar alpine redis nginx
```

And the backup image file should be formed according to the image below:



Backup more than one docker image

**B. Restore Image Docker**

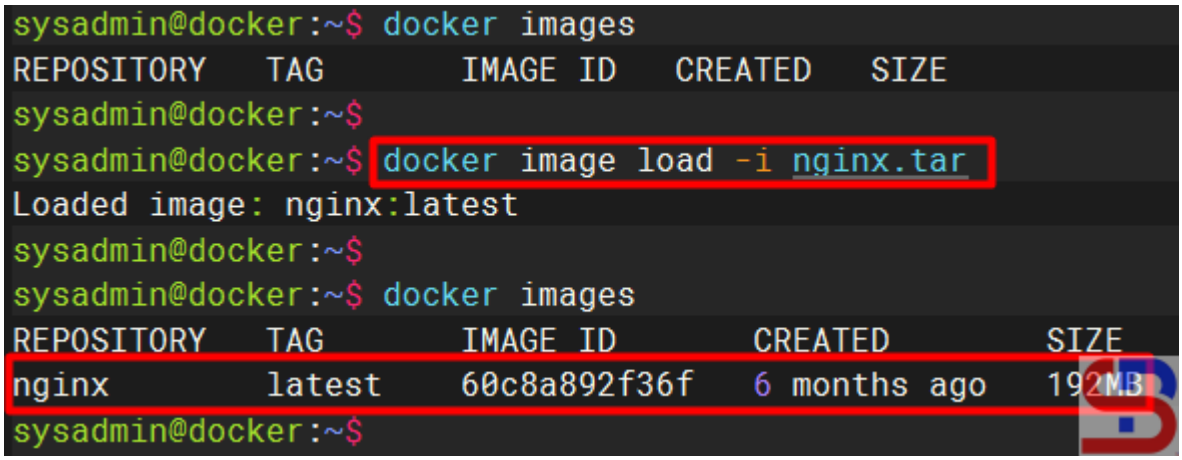
After you back up the Docker image, move your Docker image to the desired server. To restore the Docker image, use the format below:

```
docker image load -i image_name.tar
```

So I restored the Docker image backup file using the command below:

```
docker image load -i nginx.tar
```

Then the Nginx image will be restored on the server as shown below:

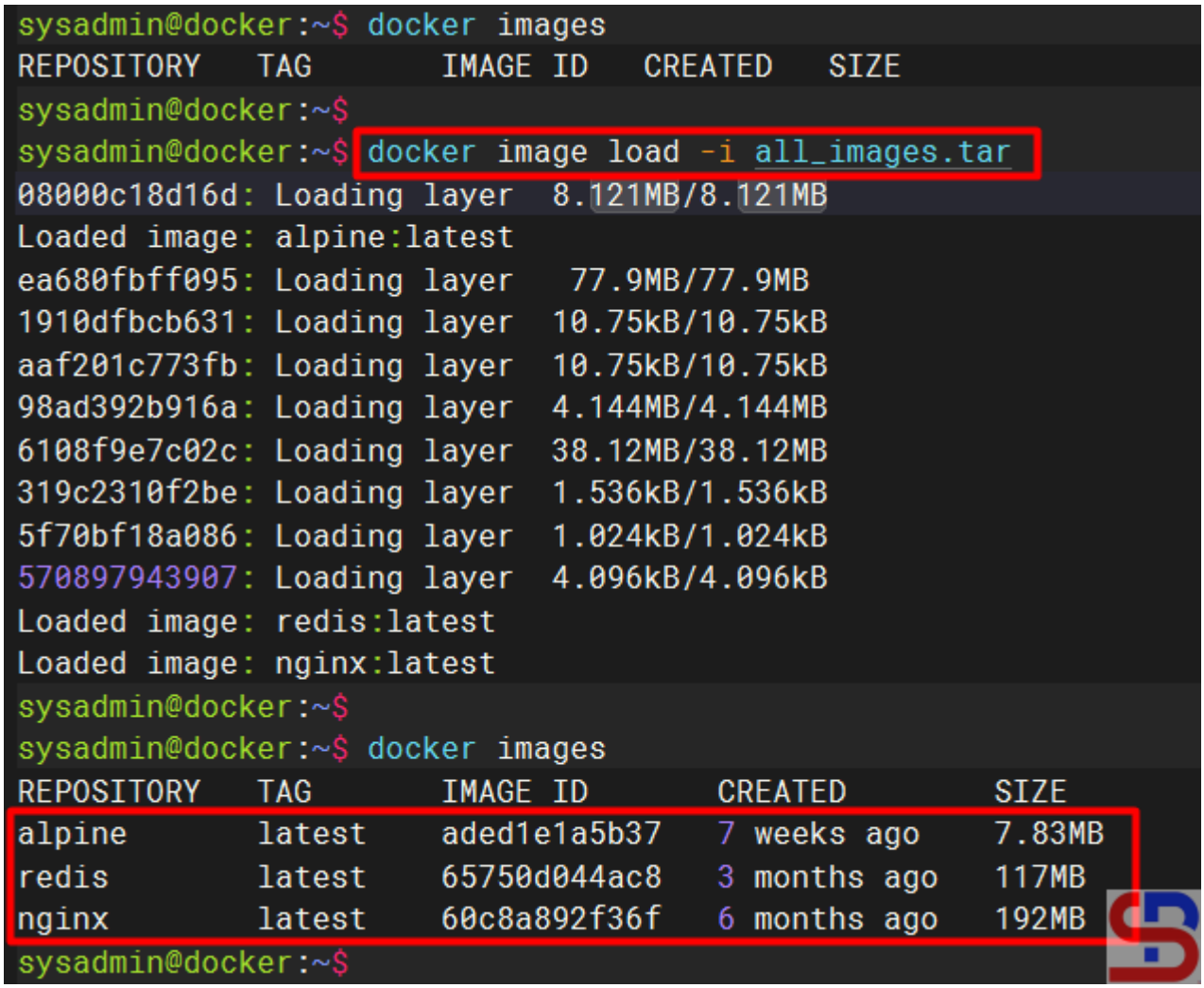


Restore one Docker image

With the same command, you can also restore more than one image using the command below:

```
docker image load -i all_images.tar
```

Then all Docker images will be restored on that server, like in the image below:



Restore more than one Docker image

## Note

You can use the command below to back up the Docker image using the format below:

```
docker save image_name | gzip -c > image_name.tgz
```

So if you want to back up the Nginx image, use the command

below:

```
docker save nginx | gzip -c > nginx.tgz
```

The advantage of using this command is that the size of the backup file is much smaller than using the previous command, as shown in the image below:

```
sysadmin@docker:~$ ls -lh
total 496M
-rw------- 1 sysadmin sysadmin 310M Apr  9 14:11 all_images.tar
-rw-rw-r-- 1 sysadmin sysadmin  22K Nov 24 10:03 get-docker.sh
-rw------- 1 sysadmin sysadmin 187M Apr  9 09:53 nginx.tar
sysadmin@docker:~$
sysadmin@docker:~$ docker save nginx | gzip -c > nginx.tgz
sysadmin@docker:~$ docker save nginx redis alpine | gzip -c > all_images.tgz
sysadmin@docker:~$
sysadmin@docker:~$ ls -lh
total 677M
-rw------- 1 sysadmin sysadmin 310M Apr  9 14:11 all_images.tar
-rw-rw-r-- 1 sysadmin sysadmin 114M Apr  9 14:44 all_images.tgz
-rw-rw-r-- 1 sysadmin sysadmin  22K Nov 24 10:03 get-docker.sh
-rw------- 1 sysadmin sysadmin 187M Apr  9 09:53 nginx.tar
-rw-rw-r-- 1 sysadmin sysadmin  68M Apr  9 14:34 nginx.tgz
sysadmin@docker:~$
```

Back up the Docker image using another command

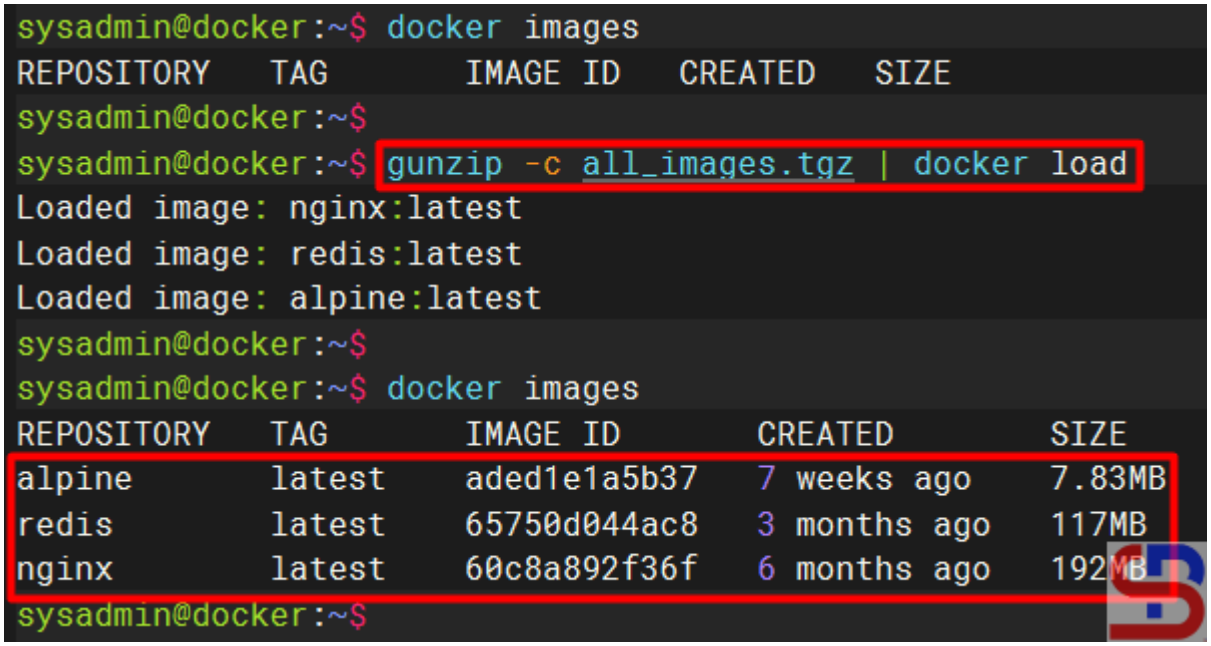
To restore, use the format below:

```
gunzip -c filename.tgz | docker load
```

So if you want to restore more than one image, use the command below:

```
gunzip -c all_images.tgz | docker load
```

And the Docker image will be restored on the server.



Restore the backup Docker image using another command

## References

youtube.dimas-maryanto.com
youtube.com
docs.docker.com
stackoverflow.com

---

# How to Limit the Use of CPU in Linux?

written by sysadmin | 14 May 2025

A large application that consumes a lot of CPU on a Linux server will cause unusual server conditions. Therefore, you have to limit the use of the CPU for the application.

## Problem

How to limit the use of CPU in Linux?

## Solution

You can use the cpulimit tool to limit CPU use. Below is the command to install the tool in Linux:

**Debian/Ubuntu**

```
sudo apt update
sudo apt-get install cpulimit -y
```

**RockyLinux/AlmaLinux/CentOS**

```
yum install epel-release -y
yum install cpulimit -y
```

**OpenSUSE15**

```
zypper install -y cpulimit
```

Here are some methods using the cpulimit command:

**A. Using --pid option**

To use the **--pid** or **-p** option, you need to know the PID of an application that you want to limit its CPU usage. To run this cpulimit tool, use the format below:

```
cpulimit --pid xxx --limit xxx --background
```

Where PID is the ID number of the ongoing application, you can see by using the format below:

```
ps aux | grep application_name
```

The limit is a percentage figure from the CPU that you want to limit for the use of the application. We will use the bash script to simulate this CPU server to be high. Create a **high_cpu.sh** file and copy the script below

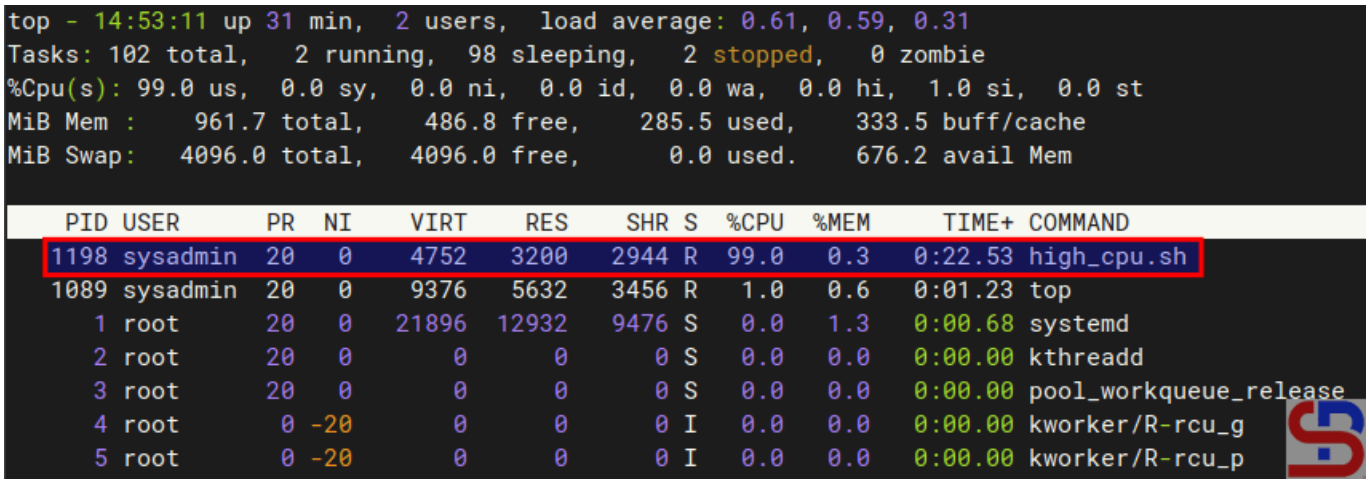
```
#!/bin/bash
```

```
# Simple infinite loop that uses 100% CPU
while true
do
    :  # No-op (no operation) to keep CPU busy
done
```

Run the command below to give permission and run this bash script:

```
chmod +x high_cpu.sh
./high_cpu.sh &
```

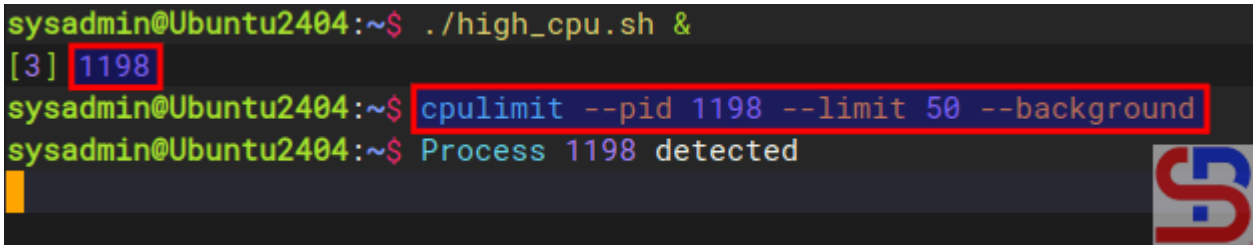
Run the **top** command on another terminal, and as you can see in the image below, the script uses 99 percent of the CPU on the server:



Before using the cpulimit command

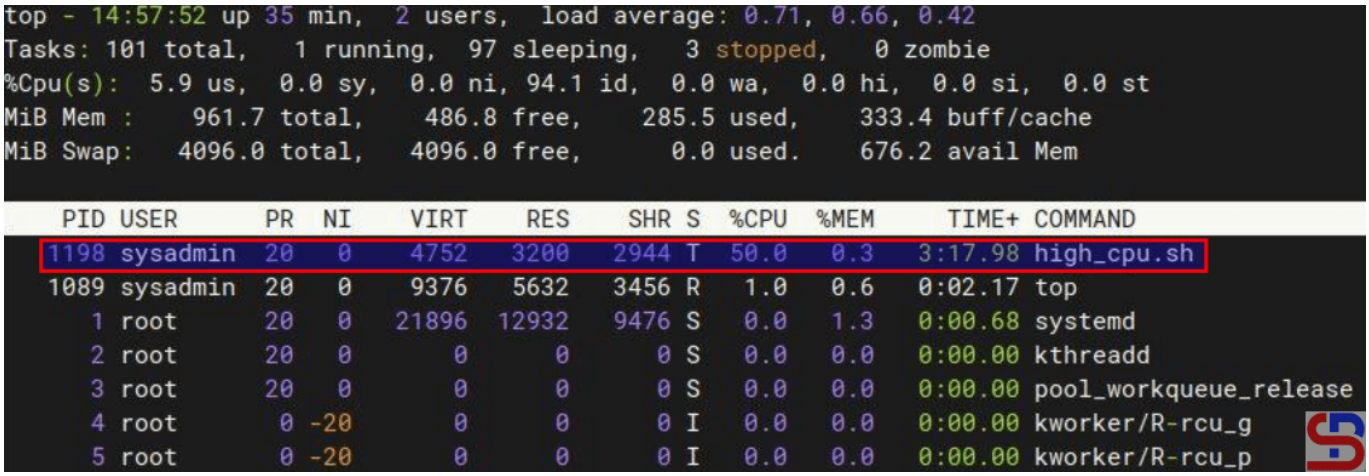
Now run the cpulimit command by limiting this script to run using only 50 percent of the CPU on this server:

```
cpulimit --pid 1198 --limit 50 --background
```



Run the cpulimit command

If you look at the top command, this script is no longer utilizing 99 percent of the CPU on this server.



After using the cpulimit command

WARNING

I don’t think there is a problem when an application consumes the CPU more than the limit you set after running the cpulimit command. For example, you limit the CPU to 50 percent using the cpulimit command, but in reality, the application runs with more than 50 percent in the use of CPU usage on your server. But at least the application doesn’t take up a lot of CPU.

The disadvantage of using the **--pid** option is that if the application is restarted, the PID of the application will change, and you have to change the cpulimit command. So, there is another option where you just write the name of the application in the cpulimit command using the --exe option.

**B. Using --exe option**

To use the **--exe** or **-e** option, you need to know the application name that you want to limit its CPU usage. To run this cpulimit tool, use the format below:

```
cpulimit --exe application_name --limit xxx --background
```

We will use the bash script to simulate a CPU server to be high with many PIDs. Create a **pids_high_cpu.sh** file in the

folder **/etc** and copy the script below:

```
#!/bin/bash

# Number of processes to spawn
num_processes=5

# Function to keep the CPU busy in each child process
cpu_intensive_task() {
    while true
    do
        :  # No-op command to keep the CPU busy
    done
}

# Spawn the specified number of processes, redirecting output to /dev/null
for ((i=0; i<num_processes; i++))
do
    cpu_intensive_task > /dev/null 2>&1 &  # Run the task in the background
and suppress output
done

# Optionally, wait for all background processes to complete (won't happen
unless manually killed)
wait
```

Run the command below to give permission and run this bash script:

```
chmod +x /etc/pids_high_cpu.sh
/etc/pids_high_cpu.sh &
```

Run the **top** command on another terminal. As you can see in the image below, the script has many PIDs and uses a lot of CPU resources on the server:

```
top - 16:21:55 up  1:59,  3 users,  load average: 3.95, 4.86, 15.72
Tasks: 111 total,   6 running, 105 sleeping,   0 stopped,   0 zombie
%Cpu(s): 78.9 us, 15.8 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  5.3 si,  0.0 st
MiB Mem :    961.7 total,    284.4 free,    304.1 used,    546.9 buff/cache
MiB Swap:   4096.0 total,   4081.7 free,     14.3 used.    657.6 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  29201 root      20   0    4752   1688   1536 R  21.6   0.2   0:05.11 pids_high_cpu.s
  29200 root      20   0    4752   1688   1536 R  20.6   0.2   0:05.10 pids_high_cpu.s
  29202 root      20   0    4752   1688   1536 R  20.6   0.2   0:05.10 pids_high_cpu.s
  29203 root      20   0    4752   1688   1536 R  19.6   0.2   0:05.10 pids_high_cpu.s
  29199 root      20   0    4752   1688   1536 R  18.6   0.2   0:05.09 pids_high_cpu.s
    887 sysadmin  20   0   14960   3884   3072 S   1.0   0.4   0:01.82 sshd
  29065 sysadmin  20   0    9376   5632   3456 R   1.0   0.6   0:04.15 top
      1 root      20   0   21872  13184   9600 S   0.0   1.3   0:03.51 systemd
```

Display the app with many PIDs

Run the cpulimit command  to restrict this script to utilizing just 50% of the server's CPU:

```
sudo cpulimit --exe pids_high_cpu.sh --limit 50 --background
```

```
root@Ubuntu2404:~# /etc/pids_high_cpu.sh &
[1] 29198
root@Ubuntu2404:~#
root@Ubuntu2404:~# ps aux | grep pids_high_cpu
root       29198  0.0  0.3   4752  3328 pts/3    S    16:21   0:00 /bin/bash /etc/pids_high_cpu.sh
root       29199 19.9  0.1   4752  1688 pts/3    R    16:21   0:19 /bin/bash /etc/pids_high_cpu.sh
root       29200 19.9  0.1   4752  1688 pts/3    R    16:21   0:19 /bin/bash /etc/pids_high_cpu.sh
root       29201 19.9  0.1   4752  1688 pts/3    R    16:21   0:19 /bin/bash /etc/pids_high_cpu.sh
root       29202 19.9  0.1   4752  1688 pts/3    R    16:21   0:19 /bin/bash /etc/pids_high_cpu.sh
root       29203 19.9  0.1   4752  1688 pts/3    R    16:21   0:19 /bin/bash /etc/pids_high_cpu.sh
root       29205  0.0  0.2   4088  2048 pts/3    S+   16:23   0:00 grep --color=auto pids_high_cpu
root@Ubuntu2404:~#
root@Ubuntu2404:~# sudo cpulimit --exe pids_high_cpu.sh --limit 50 --background
root@Ubuntu2404:~#
```

Run the cpulimit command with the --exe option

### C. Using --path option

Besides using the --exe option, you can also use the **--path** option or **-P** option. To use this option, you have to know the path of the application that you want to limit its CPU usage. Use the format below to run the **--path** option in the cpulimit command:

```
cpulimit --path /folder/path/of/the/application --limit xxx --background
```

For the same case, type the command below to use the cpulimit command with the **--path** option to limit CPU usage to **65** percent for **pids_high_cpu.sh** application in the /etc folder:

```
cpulimit --path /etc/pids_high_cpu.sh --limit 65 --background
```

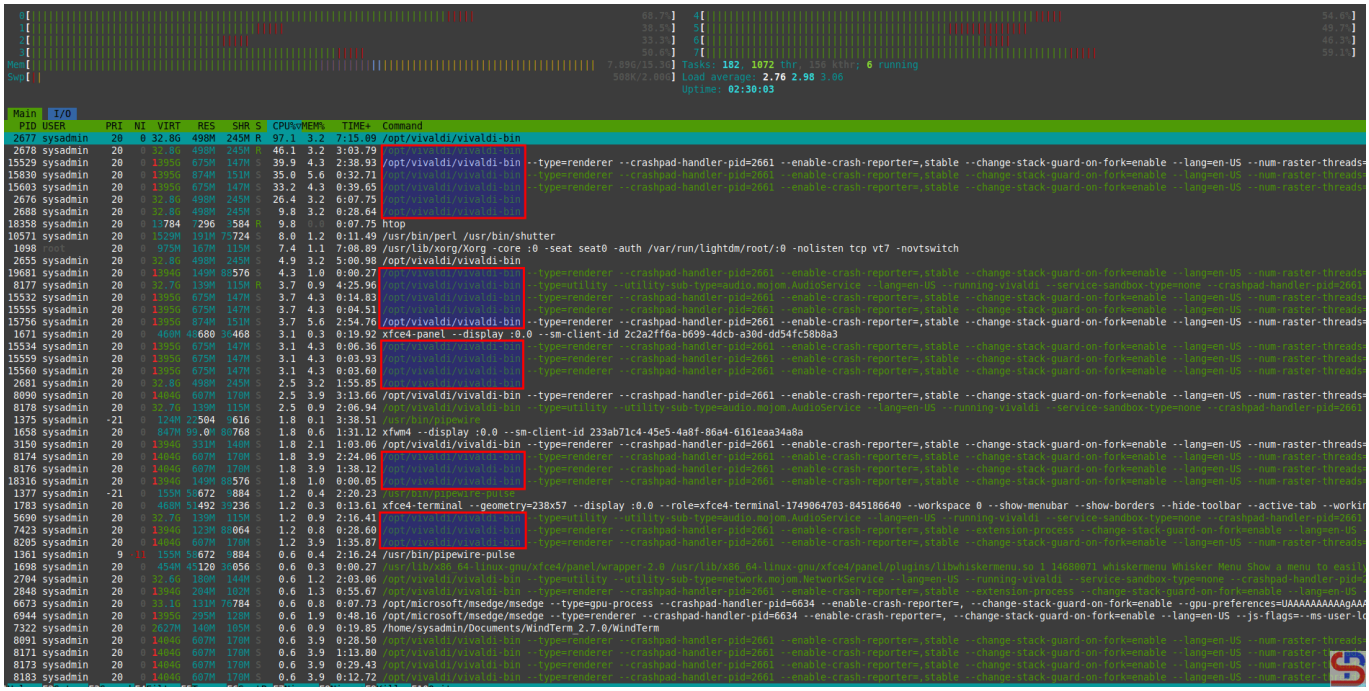
WARNING

You can enter the cpulimit command into the crontab so the command will run automatically after the device restarts. But, the cpulimit command that you enter into the crontab can only use the **--exe** or **--path** option because, by default, the application name or application path will never change after the server restarts. Here is an example of the crontab:

```
@reboot         cpulimit --path /etc/pids_high_cpu.sh --limit 65 --background
```

## Note

If you see an application running that generates a lot of PIDs, as shown in the image below:



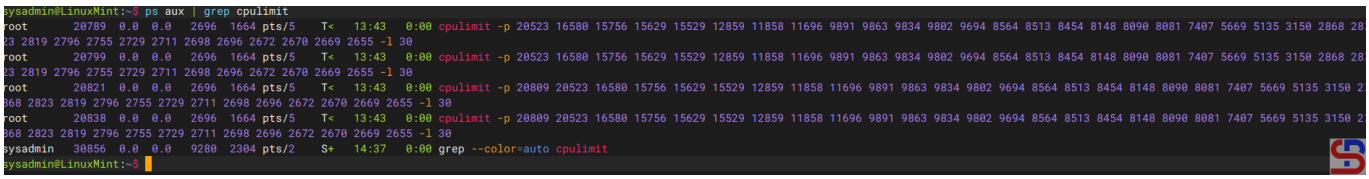
The application with many PIDs

From the picture above, you can see that the application

Vivaldi has many PIDs. So, you have to write a script to restrict apps with numerous PIDs. Here is a copy of the bash script:

```
#!/bin/bash

LIMIT=50
PROCESS_NAME="vivaldi-bin"

# Kill existing cpulimit instances related to vivaldi-bin
pkill -f "cpulimit -p" 2>/dev/null

# Get all vivaldi PIDs
PIDS=$(pidof $PROCESS_NAME)

if [[ -z "$PIDS" ]]; then
  echo "[$(date)] $PROCESS_NAME not running."
  exit 0
fi

echo "[$(date)] Limiting CPU for PIDs: $PIDS"

# Start cpulimit for each PID in background
for pid in $PIDS; do
  cpulimit -p "$pid" -l "$LIMIT" > /dev/null 2>&1 &
done
```

Change the **LIMIT** and **PROCESS_NAME** sections according to your needs and permit the script to be executed. If the script runs, it will limit the CPU to the PIDs.



Check the cpulimit command that runs in the background

Enter the script into the crontab using the script below:

```
#Limit CPU vivaldi
*/30 * * * *     /root/limit_vivaldi.sh
@reboot         /root/limit_vivaldi.sh
```

The script will restart every 30 minutes, and if the device

restarts, the script will start automatically.

### References


tecmint.com
linuxsec.org
id.ubunlog.com
linuxsec.org
youtube.com


---

## How to Increase HDD Capacity on a VM in GCP?

written by sysadmin | 14 May 2025

If you have a virtual machine at GCP, by default, the Linux system will only make one partition / only. If the partition is smaller, then you have to increase the hard disk server size

### Problem

How to increase HDD capacity on a VM in GCP?

### Solution

Currently, I have a VM Ubuntu Server 24.04 in GCP with an HDD capacity of 10 GB as in the image below:

```
root@vm-cloud:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       8.7G  8.1G  588M  94% /
tmpfs           3.9G     0  3.9G   0% /dev/shm
tmpfs           1.6G  952K  1.6G   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
efivarfs         56K   24K   27K  48% /sys/firmware/efi/efivars
/dev/sda16      881M   61M  759M   8% /boot
/dev/sda15      105M  6.1M   99M   6% /boot/efi
tmpfs           794M   12K  794M   1% /run/user/1001
root@vm-cloud:~#
```

The hard disk condition of my server

You can see from the image above that my partition / is very small, and here is the block device in my VM:

```
root@vm-cloud:~# lsblk
NAME    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0     7:0    0  63.7M  1 loop /snap/core20/2496
loop1     7:1    0 409.1M  1 loop /snap/google-cloud-cli/315
loop2     7:2    0  44.4M  1 loop /snap/snapd/23771
sda       8:0    0    10G  0 disk
├─sda1    8:1    0     9G  0 part /
├─sda14   8:14   0     4M  0 part
├─sda15   8:15   0   106M  0 part /boot/efi
└─sda16 259:0    0   913M  0 part /boot
root@vm-cloud:~#
```

The block devices in my VM

I want to increase the HDD capacity to 20 GB without rebooting the server. These are the steps below (recommended as a root user to do the steps below):

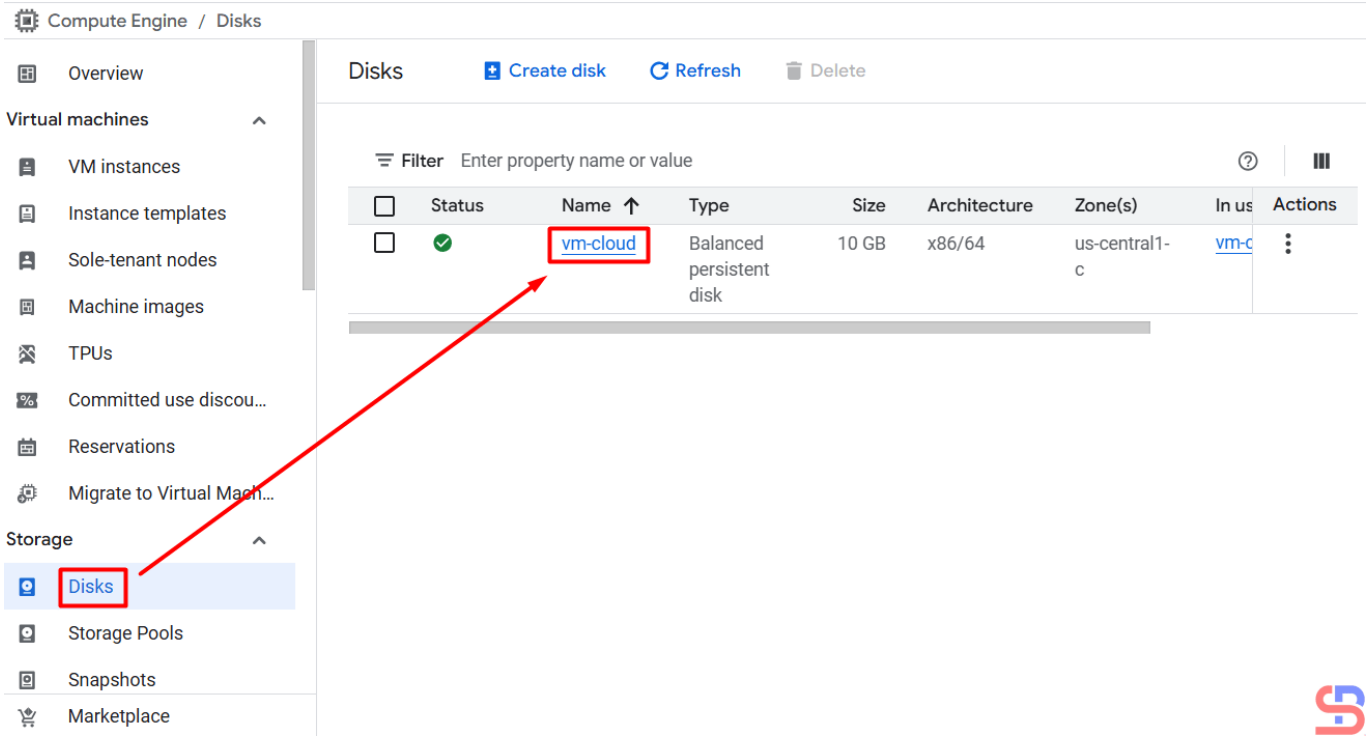
**1. Edit in the Disks section**

You can use the command below to increase the VM's hard disk to 20 GB in the cloud shell or from your laptop if you have already installed gcloud (change the VM name, size, and zone from the command below):

```
gcloud compute disks resize vm-cloud --size 20 --zone=us-central1-c
```

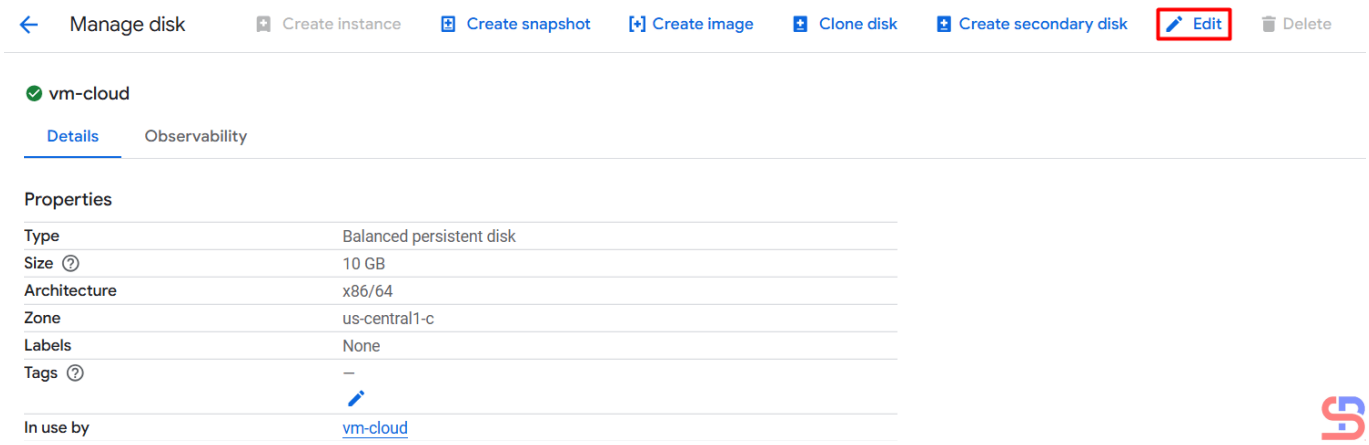
You can also increase the hard disk in GCP by entering GCP,

selecting **Compute Engine – Disks**, and then selecting VM.




Go to Disks

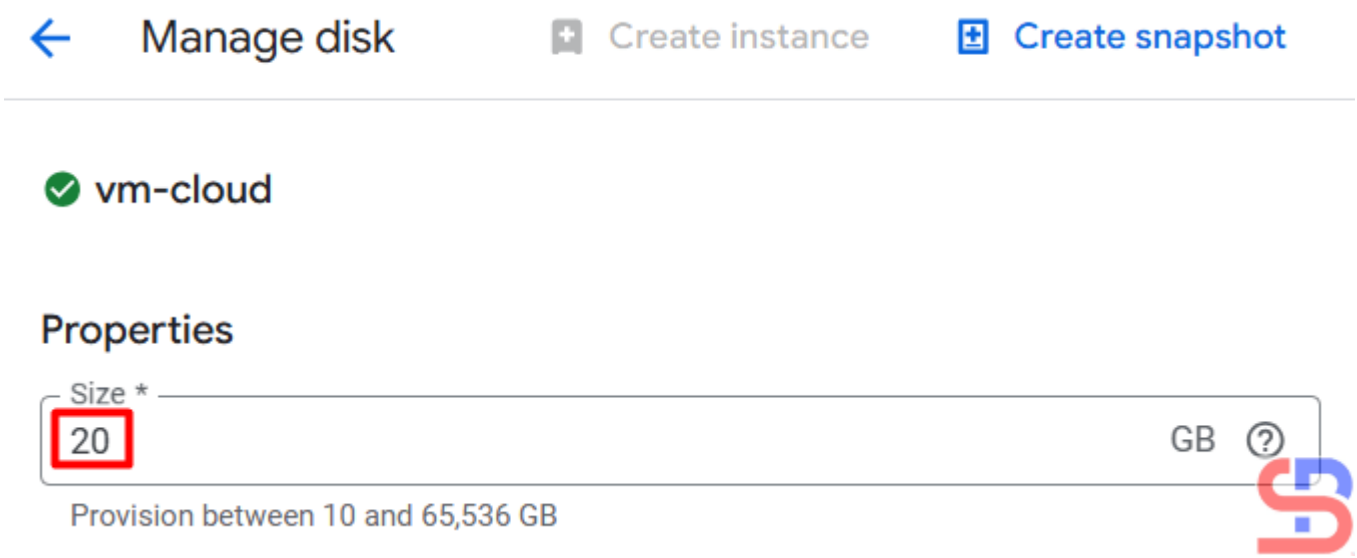
After that, click **Edit** as below:




Click the Edit

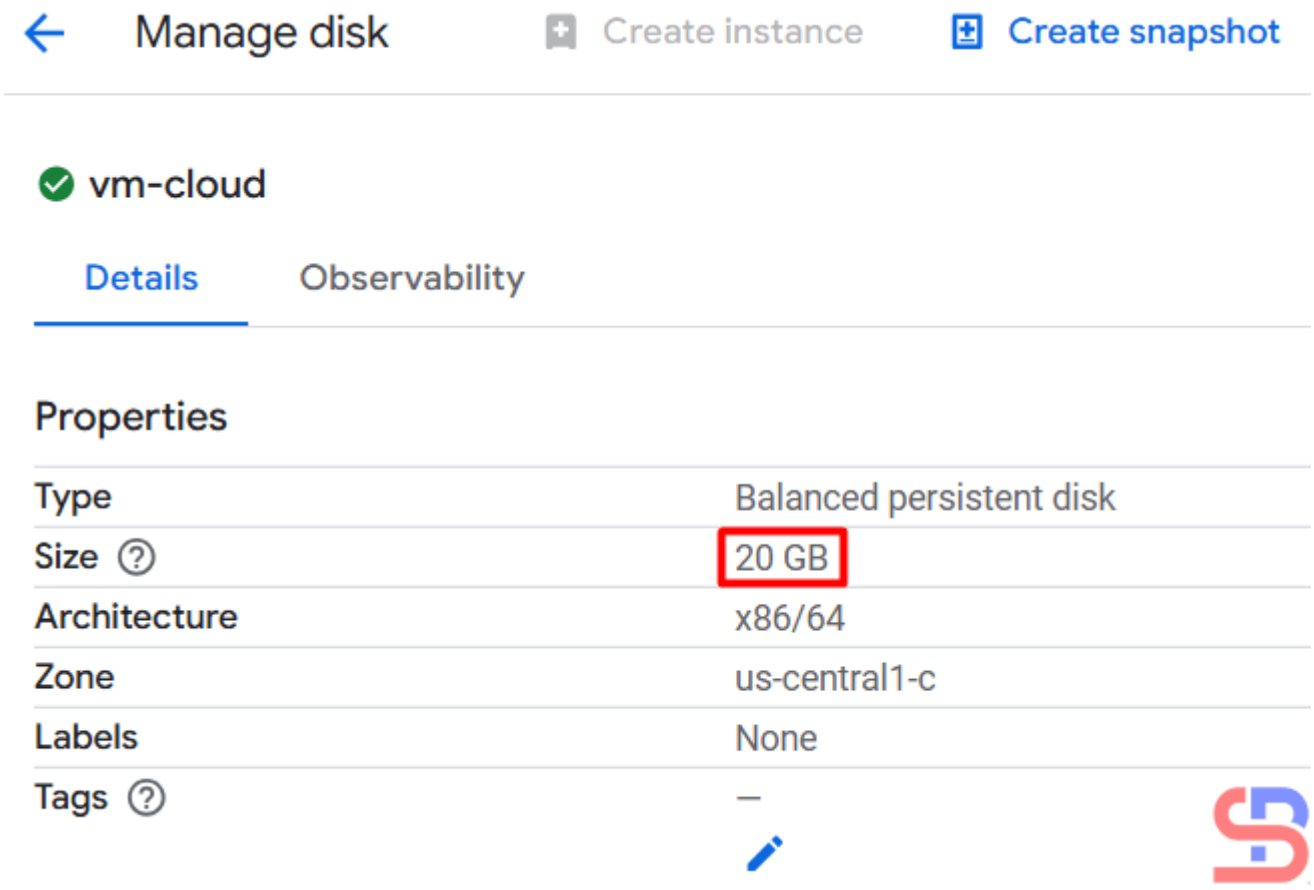
## 2. Increase HDD capacity

After that, increase the HDD capacity in the section as shown in the image below:




Change the HDD to 20 GB

Change it to 20 GB, and after that, press the **Save** button so there will be a display like below:




Server's HDD changed to 20 GB

WARNING

You cannot reduce the HDD capacity on a VM in GCP, for example, from 10 GB to

5 GB, but you can only increase the HDD capacity.

### 3. Check the block devices

Enter the VM, then we check the block devices using the command:

```
lsblk
```

```
root@vm-cloud:~# lsblk
NAME    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0     7:0    0  63.7M  1 loop /snap/core20/2496
loop1     7:1    0 409.1M  1 loop /snap/google-cloud-cli/315
loop2     7:2    0  44.4M  1 loop /snap/snapd/23771
sda       8:0    0    20G  0 disk
├─sda1    8:1    0     9G  0 part /
├─sda14   8:14   0     4M  0 part
├─sda15   8:15   0   106M  0 part /boot/efi
└─sda16 259:0    0   913M  0 part /boot
root@vm-cloud:~#
```

The block device after increasing the hard disk

You can see in the picture above that the HDD capacity is 20 GB.

### 4. Check the partition tables

Then check the partition tables using the command:

```
parted -l
```

```
root@vm-cloud:~# parted -l
Warning: Not all of the space available to /dev/sda appears to be used, you can
fix the GPT to use all of the space (an extra 20971520 blocks) or continue with
the current setting?
Fix/Ignore? F
Model: Google PersistentDisk (scsi)
Disk /dev/sda: 21.5GB
Sector size (logical/physical): 512B/4096B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
14      1049kB  5243kB  4194kB                     bios_grub
15      5243kB  116MB   111MB   fat32              boot, esp
16      116MB   1074MB  957MB   ext4               bls_boot
1       1075MB  10.7GB  9663MB  ext4


root@vm-cloud:~#
```

Check the partition table

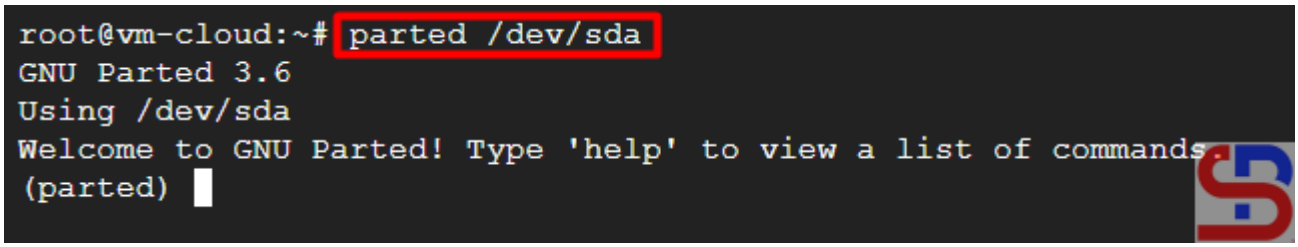
If you have a warning like in the image above, you can choose Fix or Ignore, but I chose Fix. From the picture above, you can see that the HDD in this VM has number **1** in the **/dev/sda** partition using the **ext4** extension.

WARNING
You have to be careful with the Number and Filesystem in this section because each Linux has a different Number and Filesystem.

### 5. Resize the partition

Use the command below to resize the partition:

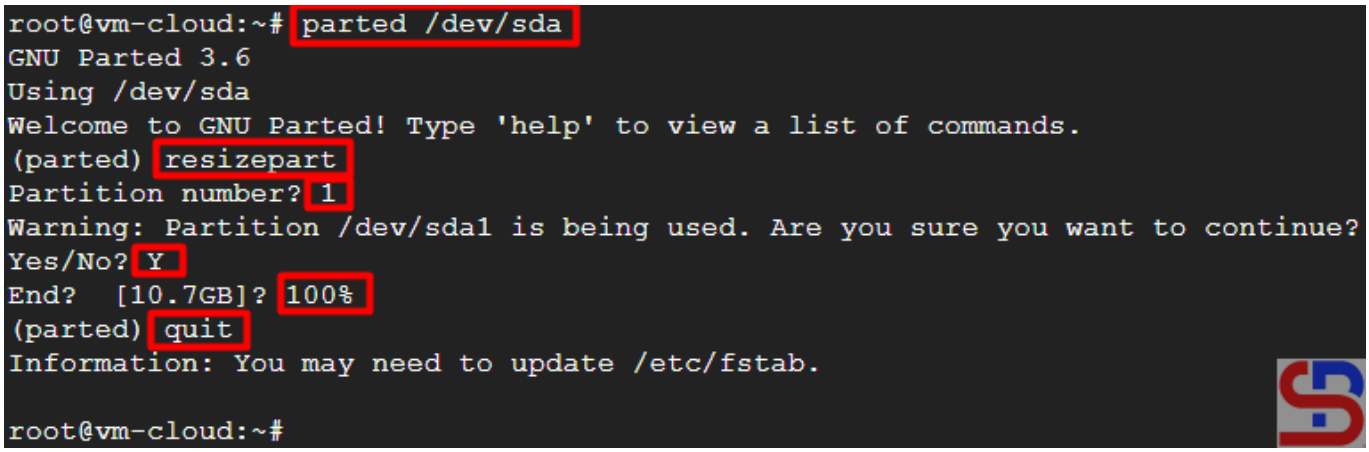
```
parted /dev/sda
```



Resize the partition /

Then type the command:

```
resizepart
```



Process to resize the partition /

Enter number **1** (check your partition number in step 5), type

**Y**, and type **100%**. After that, type **quit** to exit the prompt.

**6. Read the new partition table**

Use the command below to read the new partition table:

```
partprobe /dev/sda
```

**7. Extend the file system**

Use the command below to see the types of filesystems used in your VM:

```
df -T
```

To extend the file system, use the command below if you are using ext4 (and I am using this filesystem):

```
sudo resize2fs /dev/sda1
```

If you use the xfs filesystem, use the command:

```
sudo xfs_growfs -d /
```

But if you use btrfs, then use the command:

```
sudo btrfs filesystem resize max /
```

**8. Check HDD capacity**

Use **the df -h** command to check the hard disk capacity, and it should match the additional HDD in the GCP (in my case, the HDD capacity is 20 GB):

```
root@vm-cloud:~# partprobe /dev/sda
root@vm-cloud:~#
root@vm-cloud:~# df -Th
Filesystem     Type      Size  Used Avail Use% Mounted on
/dev/root      ext4      8.7G  8.1G  588M  94% /
tmpfs          tmpfs     3.9G     0  3.9G   0% /dev/shm
tmpfs          tmpfs     1.6G  952K  1.6G   1% /run
tmpfs          tmpfs     5.0M     0  5.0M   0% /run/lock
efivarfs       efivarfs   56K   24K   27K  48% /sys/firmware/efi/efivars
/dev/sda16     ext4      881M   61M  759M   8% /boot
/dev/sda15     vfat      105M  6.1M   99M   6% /boot/efi
tmpfs          tmpfs     794M   12K  794M   1% /run/user/1001
root@vm-cloud:~#
root@vm-cloud:~# resize2fs /dev/sda1
resize2fs 1.47.0 (5-Feb-2023)
Filesystem at /dev/sda1 is mounted on /; on-line resizing required
old_desc_blocks = 2, new_desc_blocks = 3
The filesystem on /dev/sda1 is now 4980475 (4k) blocks long.

root@vm-cloud:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        19G  8.1G   11G  44% /
tmpfs           3.9G     0  3.9G   0% /dev/shm
tmpfs           1.6G  952K  1.6G   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
efivarfs         56K   24K   27K  48% /sys/firmware/efi/efivars
/dev/sda16      881M   61M  759M   8% /boot
/dev/sda15      105M  6.1M   99M   6% /boot/efi
tmpfs           794M   12K  794M   1% /run/user/1001
root@vm-cloud:~#
```

Check the hard disk size

## Note

You should back up the important data on the VM first before following the steps above. However, you can increase the HDD capacity in a VM without doing the steps above by rebooting the VM after changing the HDD capacity in the GCP console (step 2).

## References


[cloud.google.com](cloud.google.com)
[man7.org](man7.org)
[medium.com](medium.com)
[gist.github.com](gist.github.com)
[youtube.com](youtube.com)

# How to Create a Virtual Machine Using CLI in GCP?

written by sysadmin | 14 May 2025

The previous article explained how to create a virtual machine in GCP using a GUI template. This article will explain how to create a virtual machine on GCP using the CLI.

## Problem

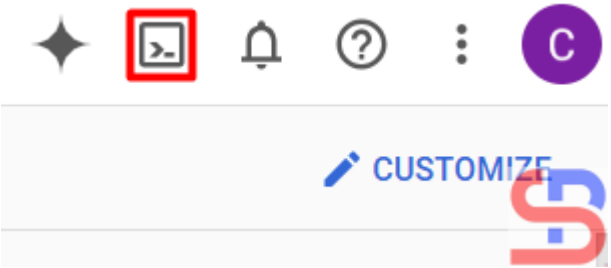
How to create a virtual machine using CLI in GCP?

## Solution

These are the steps to create a virtual machine using CLI:

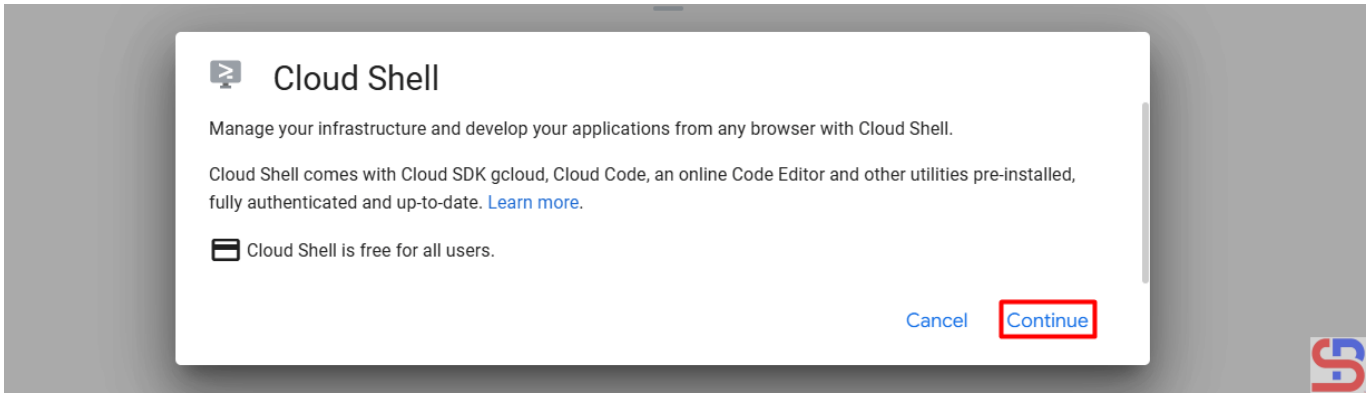
### A. Access to the gcloud

If you have already installed gcloud on your laptop, you can go to the next step. But, if you want to use gcloud in your GCP, go to the GCP dashboard, then click the small box as shown in the image below to activate the cloud shell (or you push the **G** then **S** button):



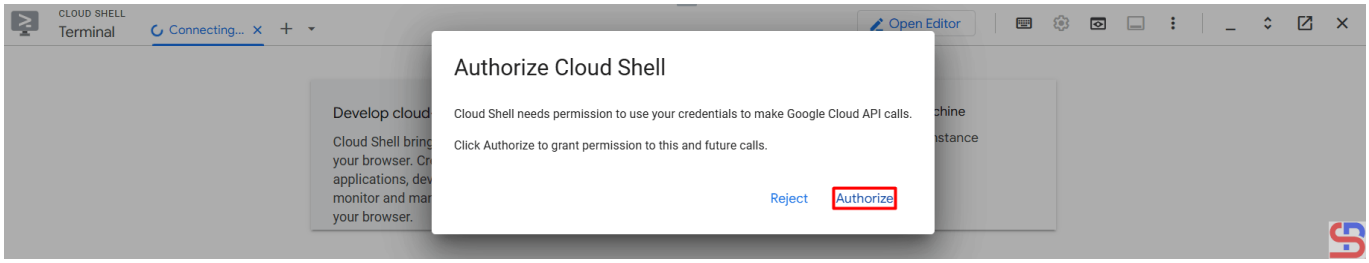

Click the Activate Cloud Shell button

There will be a display below at the bottom:



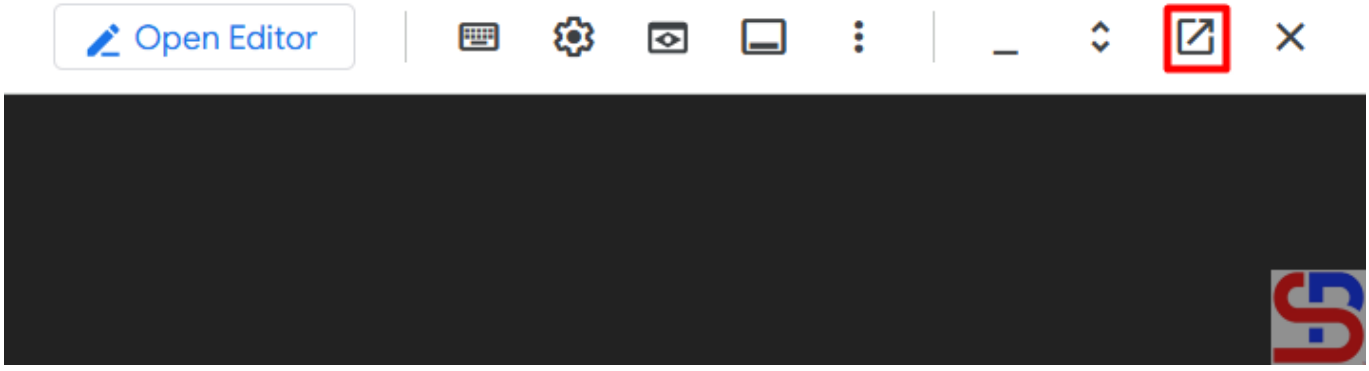
The Cloud Shell

After you click **Continue** and wait a minute, the screen shown in the picture below will appear:



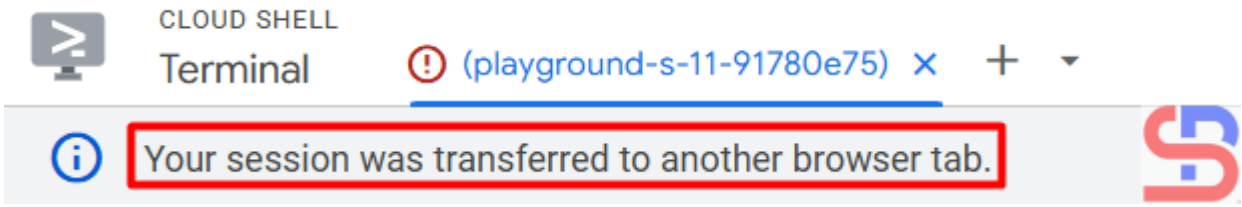
Click Authorize

Click **Authorize**, and the cloud shell is ready to use. If you want the cloud shell to have a larger screen, you can click the button below:



Click the icon

And the existing cloud shell will be inactive, as shown in the image below, so that the cloud shell will move to a new tab:



The inactive cloud shell

**B. Run the command**

By default, you can use the command below to display the options to create a virtual machine:

```
gcloud compute instances create --help
```

From the image above, you can see that you have many options. But actually, you can only use 3 options to make a VM: the zone, machine types, and image options. You have to know that by default, a VM will automatically get a hard drive size of 10 GB, so you don’t need to determine the size of a hard drive on a VM. To see the available zone options, use the command below:

```
gcloud compute zones list
```

Use the following command to view the machine type you wish to use:

```
gcloud compute machine-types list
```

To see the available images, use the command below:

```
gcloud compute images list
```

So, if you want to create a virtual machine in zone us-central1-c, use machine-type e2-standard-2, and use OS Ubuntu 24.04, use the command below:

```
gcloud compute instances create vm-cloud \
--zone=us-central1-c \
```

```
--machine-type "e2-standard-2" \
--image-project "ubuntu-os-cloud" \
--image-family "ubuntu-2404-lts-amd64" \
--subnet "default"
```

After that, check the existing VM in GCP using the command below:

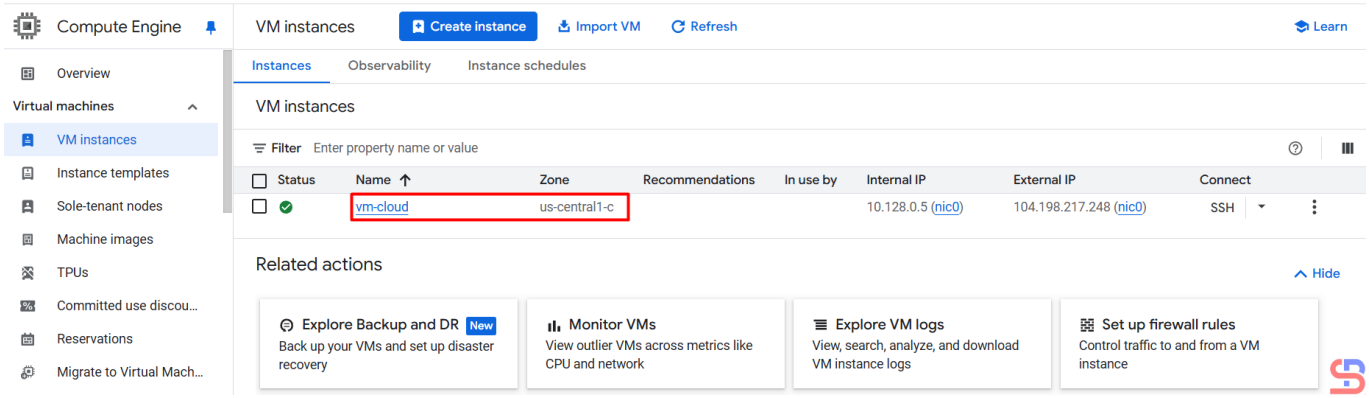
```
gcloud compute instances list
```

The VM should be made as shown below:

```
cloud_user_p_099e306e@cloudshell:~ (playground-s-11-91780e75)$ gcloud compute instances list
Listed 0 items.
cloud_user_p_099e306e@cloudshell:~ (playground-s-11-91780e75)$
cloud_user_p_099e306e@cloudshell:~ (playground-s-11-91780e75)$ gcloud compute instances create vm-cloud \
--zone=us-central1-c \
--machine-type "e2-standard-2" \
--image-project "ubuntu-os-cloud" \
--image-family "ubuntu-2404-lts-amd64" \
--subnet "default"
Created [https://www.googleapis.com/compute/v1/projects/playground-s-11-91780e75/zones/us-central1-c/instances/vm-cloud].
NAME: vm-cloud
ZONE: us-central1-c
MACHINE_TYPE: e2-standard-2
PREEMPTIBLE:
INTERNAL_IP: 10.128.0.2
EXTERNAL_IP: 35.224.230.13
STATUS: RUNNING
cloud_user_p_099e306e@cloudshell:~ (playground-s-11-91780e75)$
cloud_user_p_099e306e@cloudshell:~ (playground-s-11-91780e75)$ gcloud compute instances list
NAME: vm-cloud
ZONE: us-central1-c
MACHINE_TYPE: e2-standard-2
PREEMPTIBLE:
INTERNAL_IP: 10.128.0.2
EXTERNAL_IP: 35.224.230.13
STATUS: RUNNING
cloud_user_p_099e306e@cloudshell:~ (playground-s-11-91780e75)$
```

Create the VM using CLI

Or you can see the list of the VMs in the **VM instances** page in the image below:




The new VM appears in the GCP

## Note

If you want to create  a VM with a 50GB hard drive, use the command below:

```
gcloud compute instances create vm-cloud \
--zone=us-central1-c \
--machine-type "e2-standard-2" \
--image-project "ubuntu-os-cloud" \
--image-family "ubuntu-2404-lts-amd64" \
--boot-disk-size=50GB \
--boot-disk-type=pd-standard \
--subnet "default"
```

## References


cloud.google.com
medium.com
diana-moraa.medium.com
youtube.com


---

# How to Upgrade Ubuntu to the Latest Version?

written by sysadmin | 14 May 2025

I have a Linux Ubuntu server version 22.04, and I want to upgrade to the latest version of Ubuntu.

## Problem

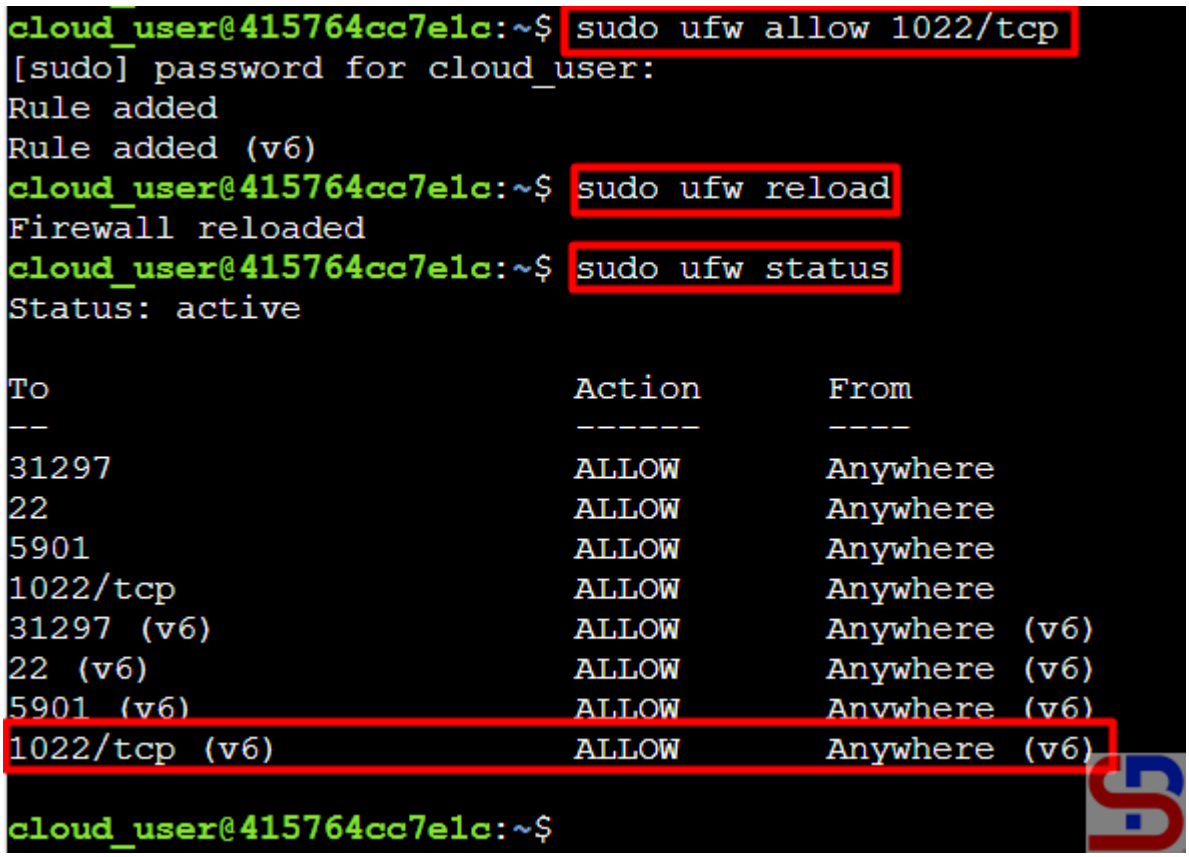
How to upgrade Ubuntu to the latest version?

## Solution

Before you upgrade your Ubuntu version, I think you have to back up your important data to other devices, and have

internet to download the packages needed to upgrade. After that, **open port 1022** on your laptop or server if you use the firewall using the below commands:

```
sudo ufw allow 1022/tcp
sudo ufw reload
sudo ufw status
```



Open the port

You should know that the Ubuntu version **upgrade process can only be done to one major LTS version**. So if you have Ubuntu version 20.04 and want to upgrade to the latest version (version 24.04 in November 2024), you have to do a 2x upgrade process, upgrading to version 22.04 first and then to version 24.04. I have Ubuntu version 22.04, like in the image below:

```
cloud_user@415764cc7e1c:~$ cat /etc/*release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=22.04
DISTRIB_CODENAME=jammy
DISTRIB_DESCRIPTION="Ubuntu 22.04.5 LTS"
PRETTY_NAME="Ubuntu 22.04.5 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.5 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
cloud_user@415764cc7e1c:~$
```

Check the version of Ubuntu

So, I type the command below:

```
sudo apt update
sudo apt upgrade -y
```

After that, reboot the server using the command below:

```
sudo reboot
```

After reboot, run the command below:

```
sudo do-release-upgrade
```

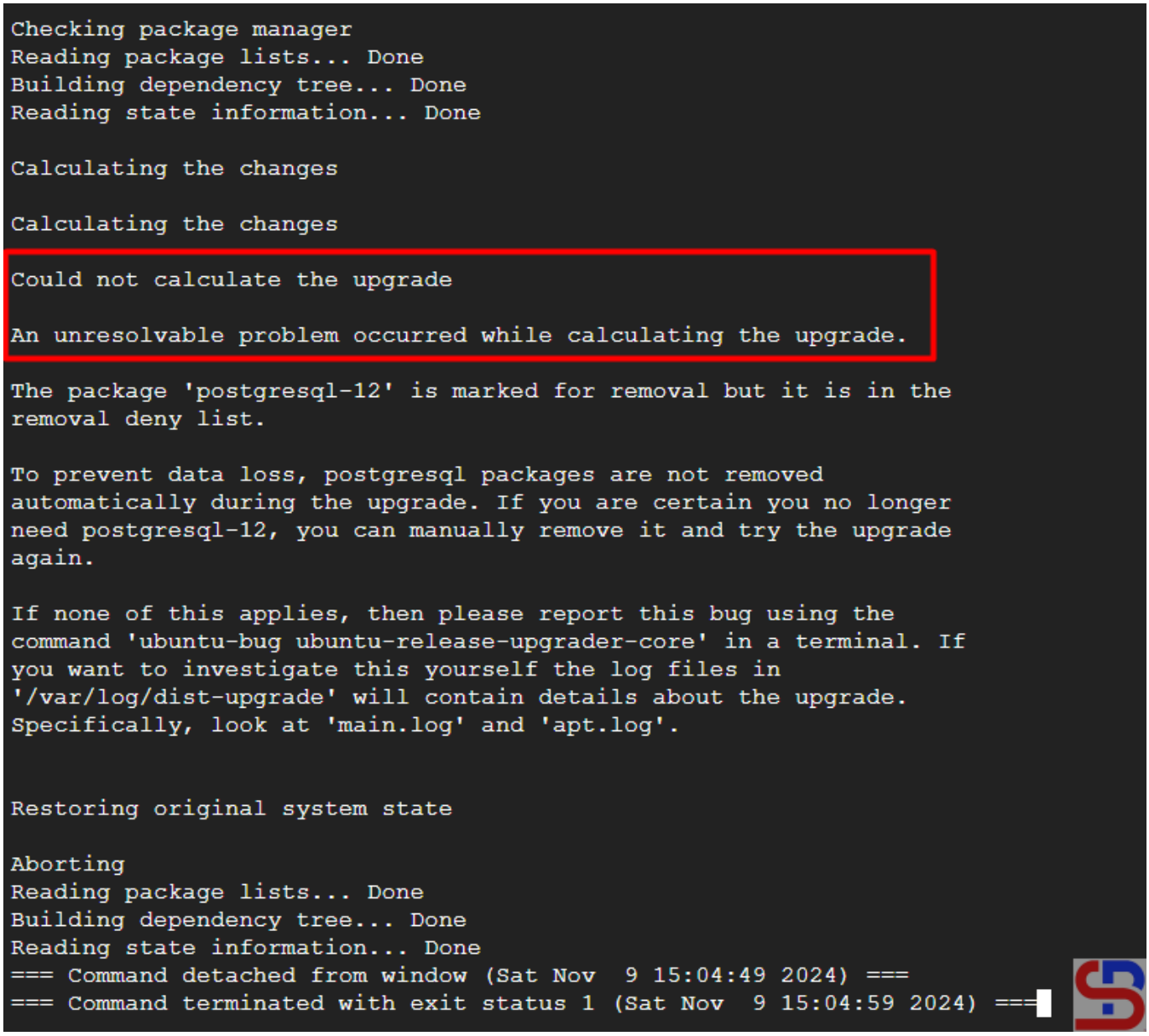
The server will start upgrading to Ubuntu version 24.04. Wait until finished, and sometimes you have to answer the questions asked by the Linux system when upgrading. After the upgrade finishes, check the version of Ubuntu, like in the image below:


Ubuntu was successfully upgraded

If during the upgrade process, there is a notification like the picture below:

Could not calculate the upgrade

An unresolvable problem occurred while calculating the upgrade.

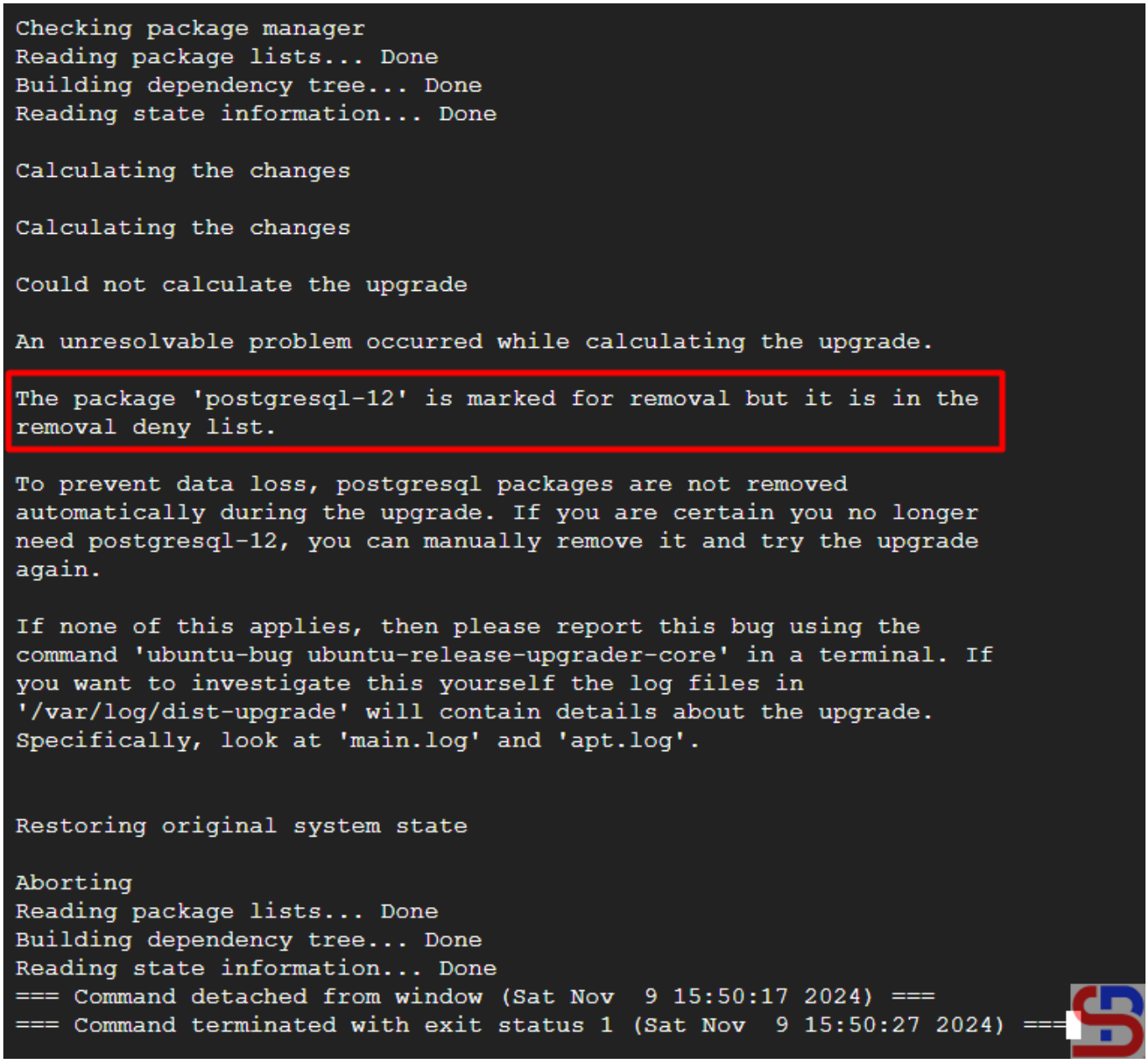


Error when upgrading Ubuntu

Type the command below to see the errors that occurred during the upgrade process:

```
cat /var/log/dist-upgrade/main.log | grep ERROR
```

In the log, you have to search for the cause of the error, but actually, you can find the root cause in the notification, like in the image below:




Find the root cause of the error

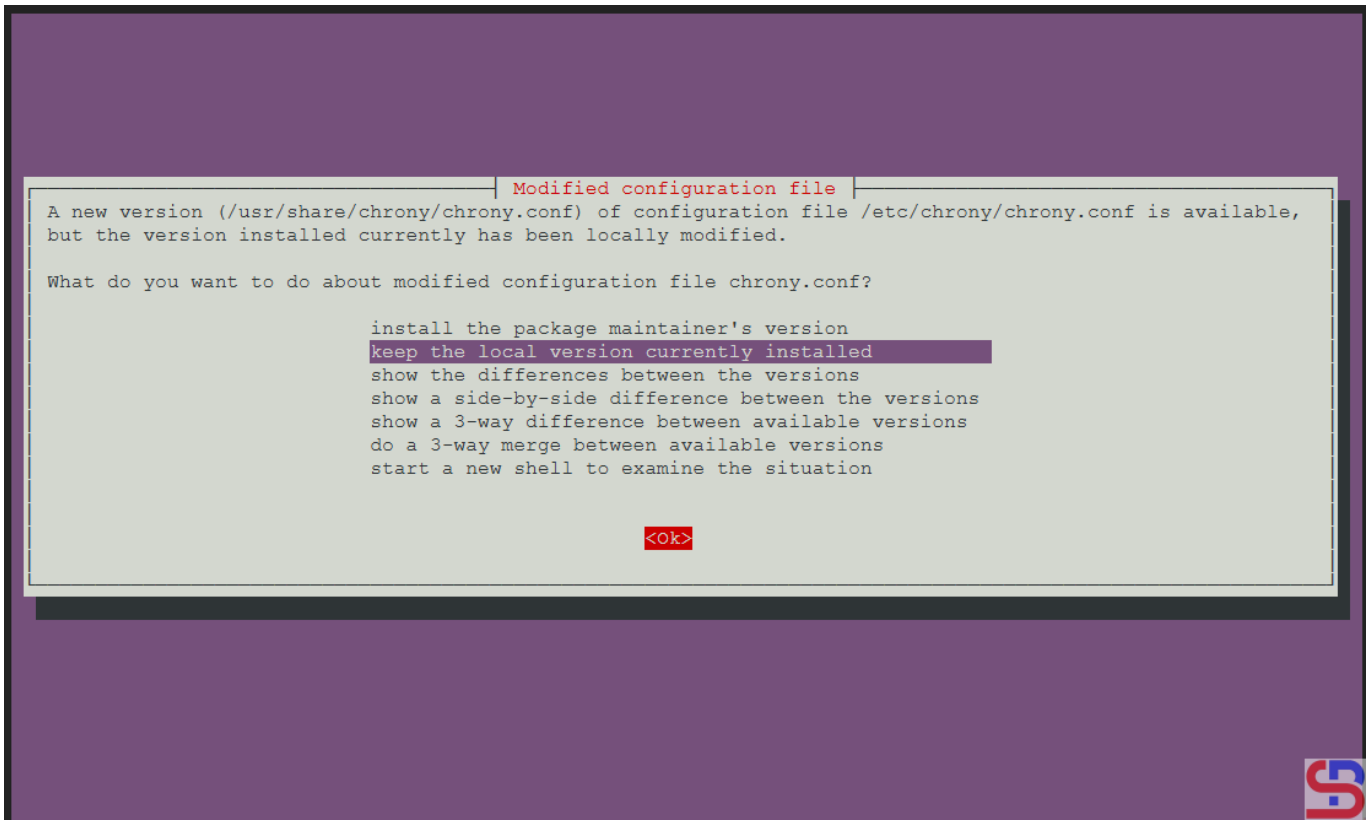
The root cause is in the postgresql-12 package, so I removed the package and then ran the command below to carry out the upgrade process again:

```
sudo do-release-upgrade
```

The Ubuntu upgrade process should be completed until it is finished.

## Note

When you upgrade Ubuntu, you have to answer the questions from the Ubuntu system, like in the image below:




Choose the answer when upgrading to Ubuntu

If you don’t want to be bothered by the questions asked by the Linux system during the upgrade process, then use the command below:

```
sudo do-release-upgrade -f DistUpgradeViewNonInteractive
```

## References


ubuntu.com
serverpilot.io
jumpcloud.com
askubuntu.com