

به نام خدا

basybox image پروژه

استاد مربوطه: جناب خانم دکتر طاهری

مقدمه:

یک ابزار قدرتمند و سبک وزن است که به عنوان **BusyBox** "چاقوی سوئیسی لینوکس" شناخته می‌شود. این ابزار مجموعه‌ای از ابزارهای استاندارد یونیکس را در یک فایل اجرایی کوچک ترکیب می‌کند و به همین دلیل در محیط‌هایی با منابع محدود مانند سیستم‌های نهفته (Embedded Systems)، توزیع‌های کوچک لینوکسی و ابزارهای خط فرمان بسیار پرکاربرد است.

شامل نسخه‌های ساده‌شده‌ای از دستورات و ابزارهایی مانند `ls`, `grep`, `cp`, `vi`, و بسیاری دیگر است. این ابزار به دلیل اندازه کوچک و کارایی بالا، برای استفاده در سیستم‌هایی با حافظه کم یا در شرایطی که سرعت و ساده‌سازی اهمیت دارد، بسیار مناسب است.

در دنیای مدرن، **BusyBox** به ویژه در حوزه‌هایی مانند IoT (اینترنت اشیا)، دستگاه‌های نهفته، و حتی سیستم‌های مجازی و کانتینرها مانند Docker استفاده می‌شود. این ابزار با ارائه

عملکردهای اصلی و حیاتی لینوکس، به توسعه دهندگان و مهندسان اجازه می‌دهد تا سیستم‌هایی سبک و کارآمد ایجاد کنند.

در این پروژه، ما به نصب، معرفی و آموزش استفاده از **BusyBox** خواهیم پرداخت تا با کاربردهای آن در توسعه سیستم‌های کوچک و نهفته بیشتر آشنا شویم.

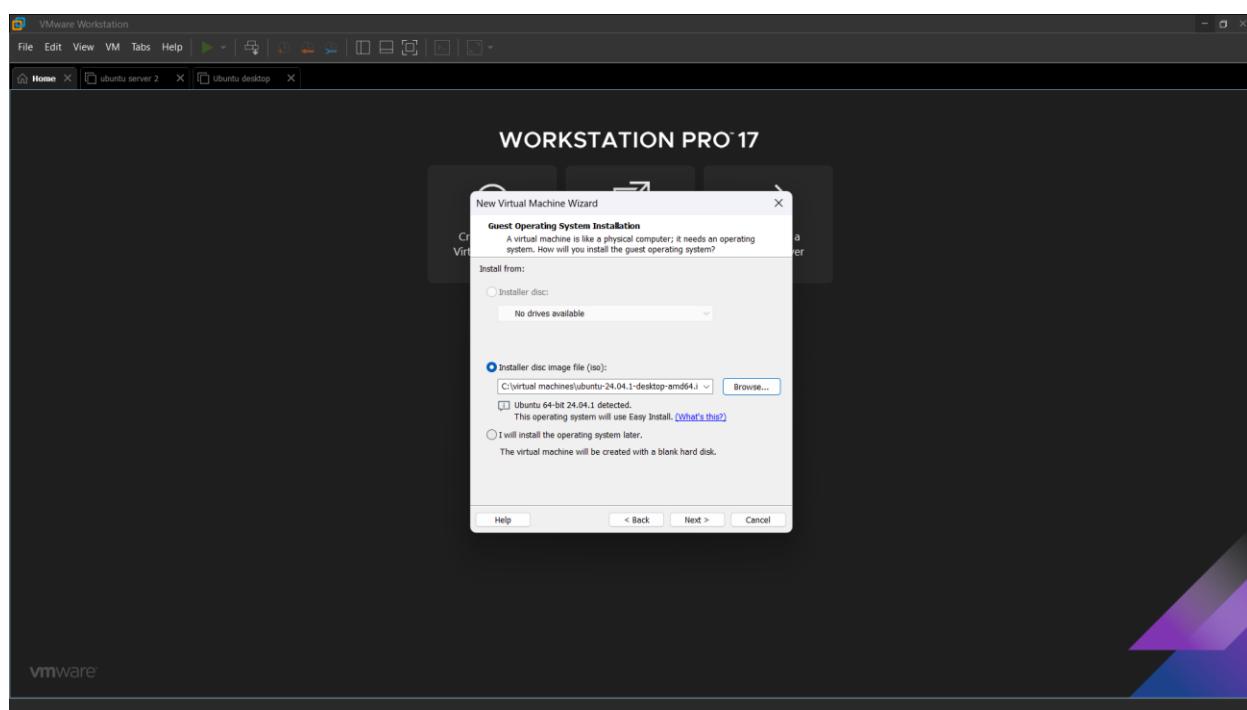
گام اول: نصب لینوکس

ما برای این پروژه به یک بستر لینوکس یا یونیکس نیاز داریم؛ که برای سهولت کار میتوانیم از یک بستر سرور و یک بستر دسکتاپ لینوکس استفاده کنیم.

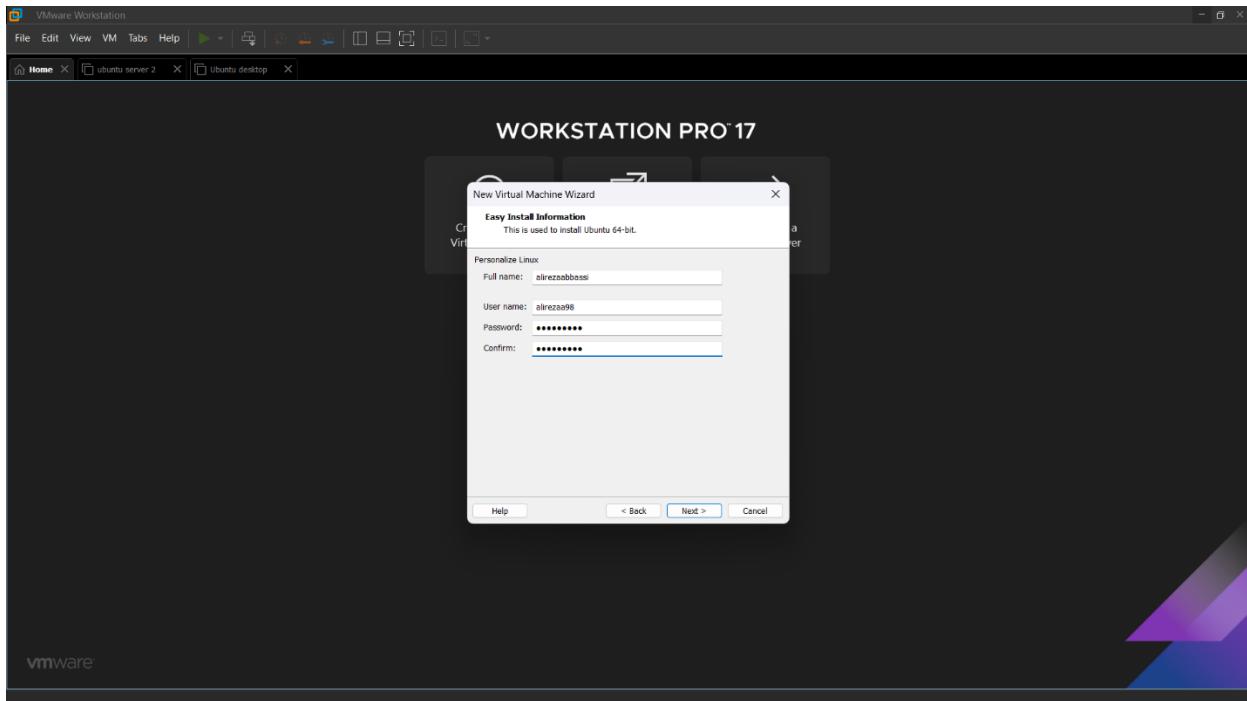
من برای انجام این پروژه از یک اوبونتو سرور و یک اوبونتو دسکتاپ استفاده کردم.

1. دانلود فایل <https://ubuntu.com>. از سایت .iso

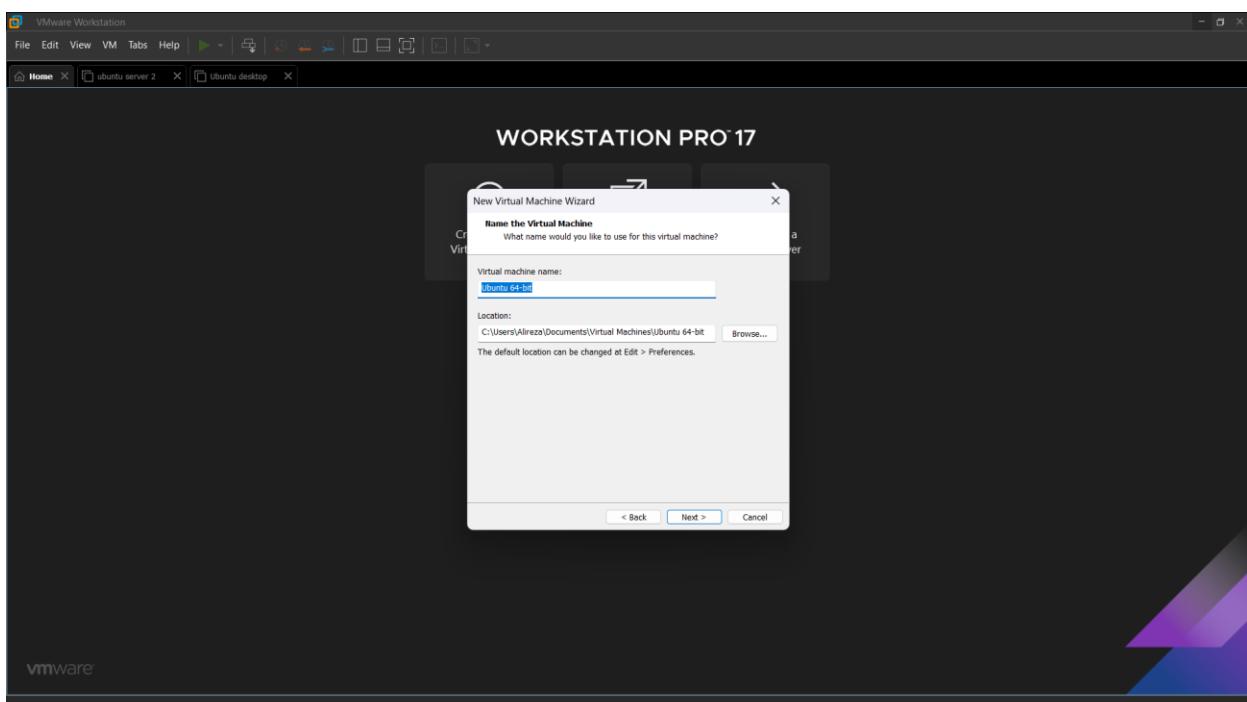
2. نصب فایل .iso روی بستر vm
من برای این پروژه از vm ware استفاده کردم

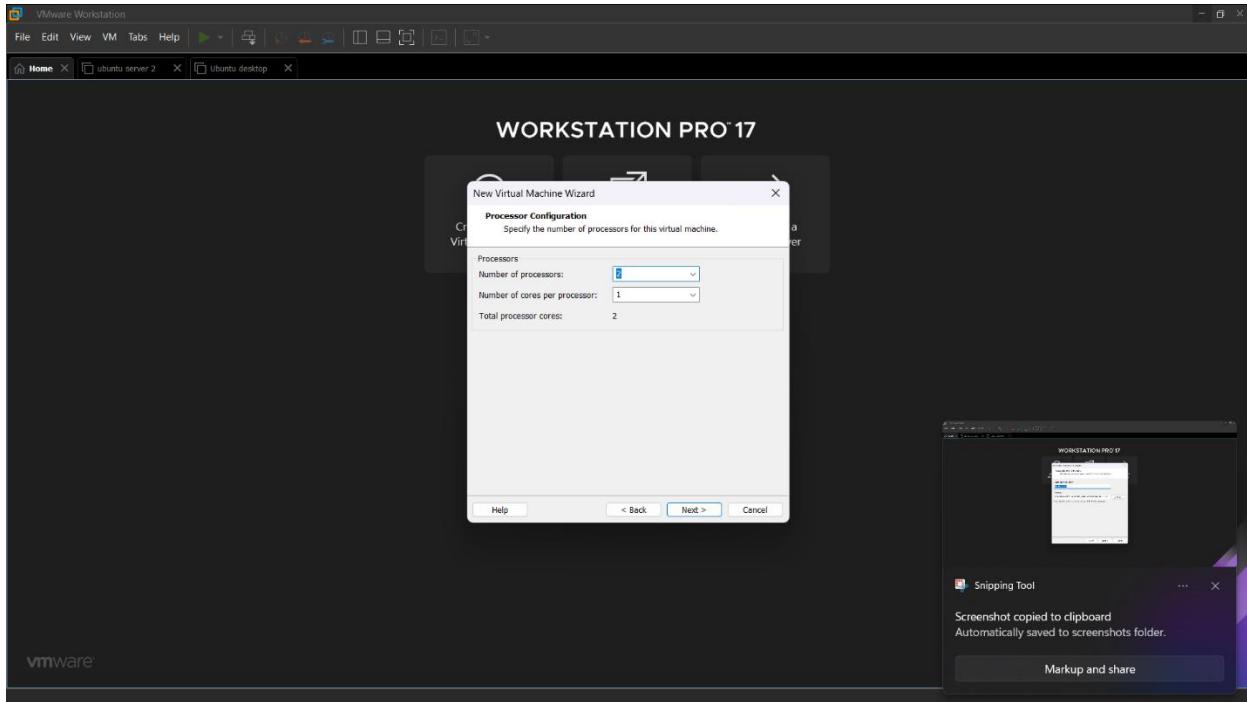


در مرحله اول نصب باید با اضافه کردن یک ماشین مجازی به vm ware و انتخاب فایل ios دانلود شده دیسک vm ware iso را به vm ware اضافه کنیم.

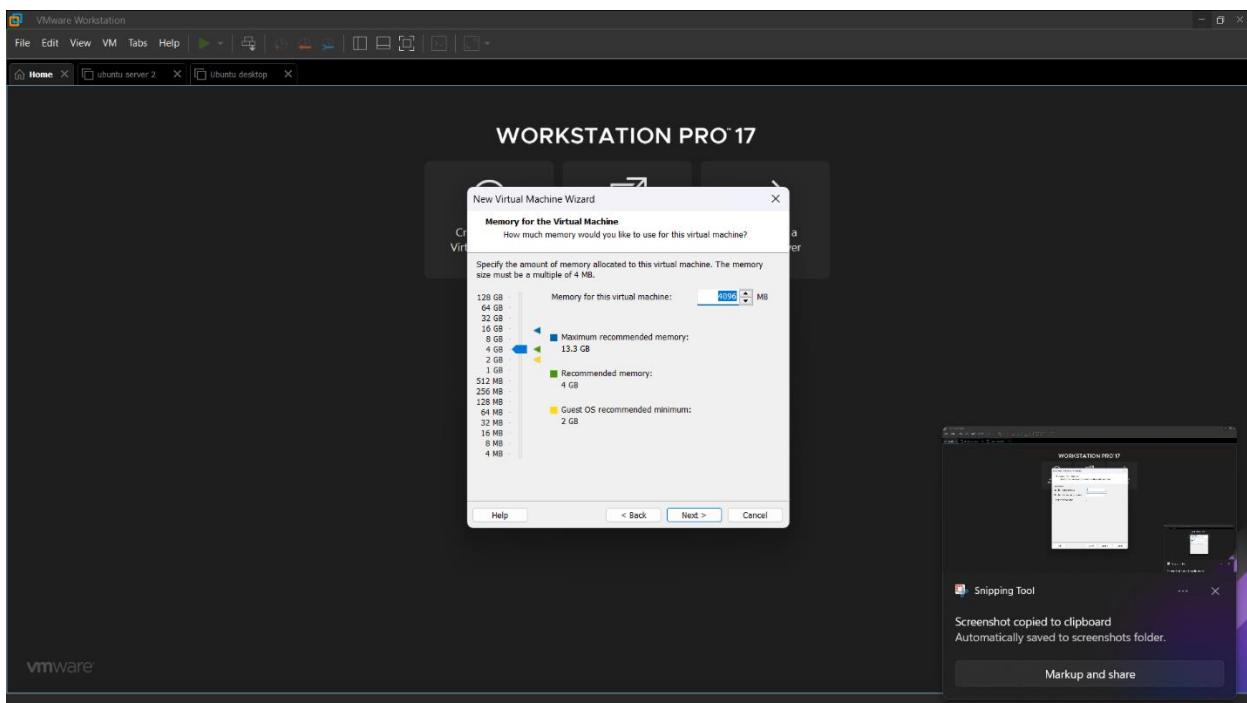


در مرحله دوم باید فرم خواسته شده را پر کنیم

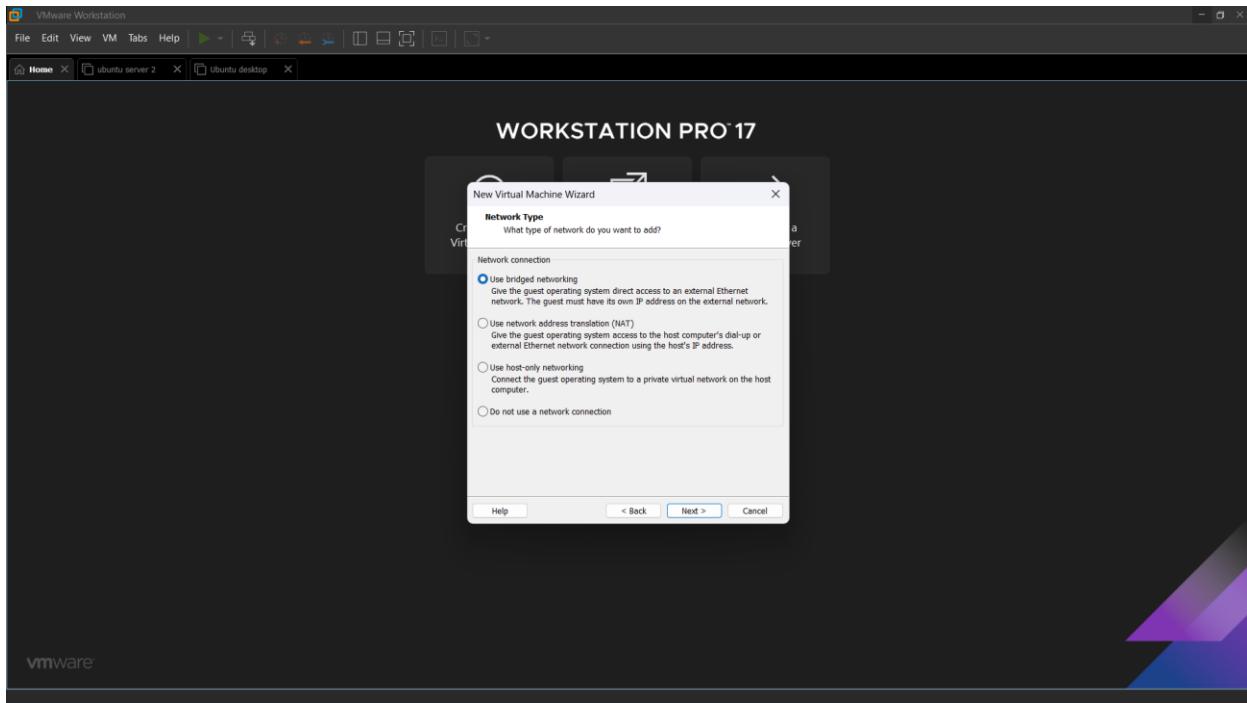




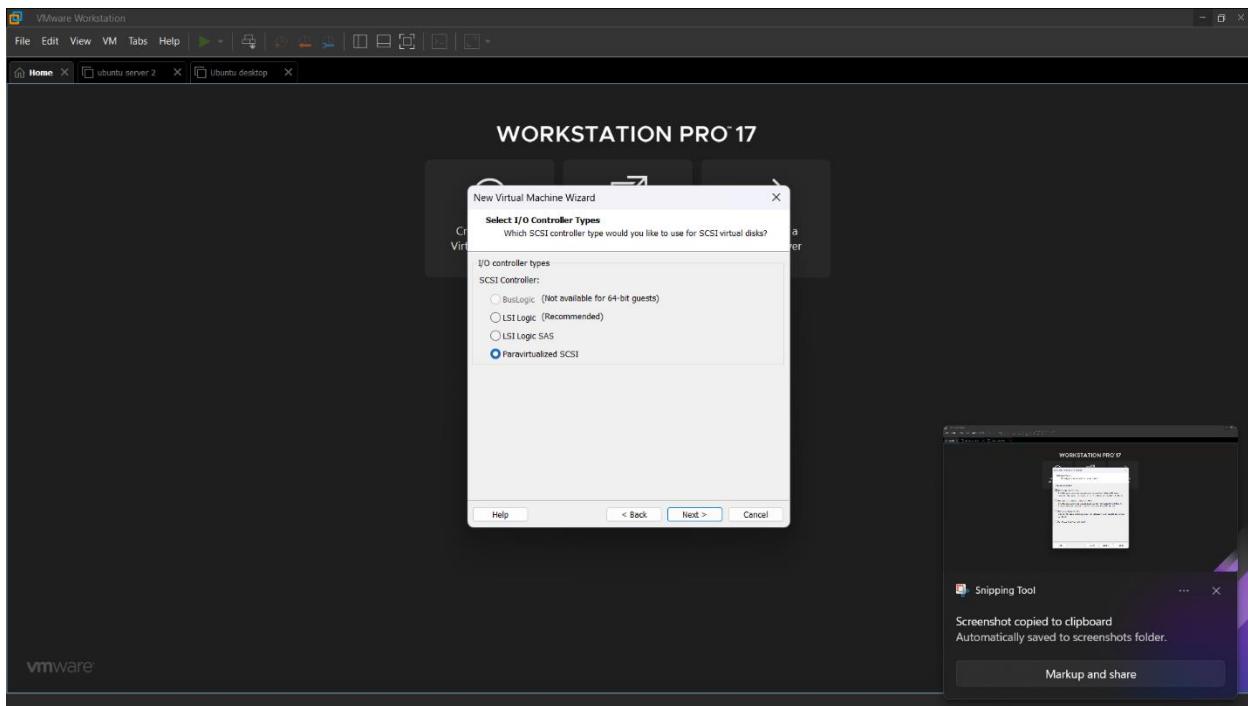
در مرحله چهارم باید مقدار CPU و تعداد هسته های مجازی آن را که برای پردازش ماشین مجازی خود در نظر داریم را مشخص کنیم.



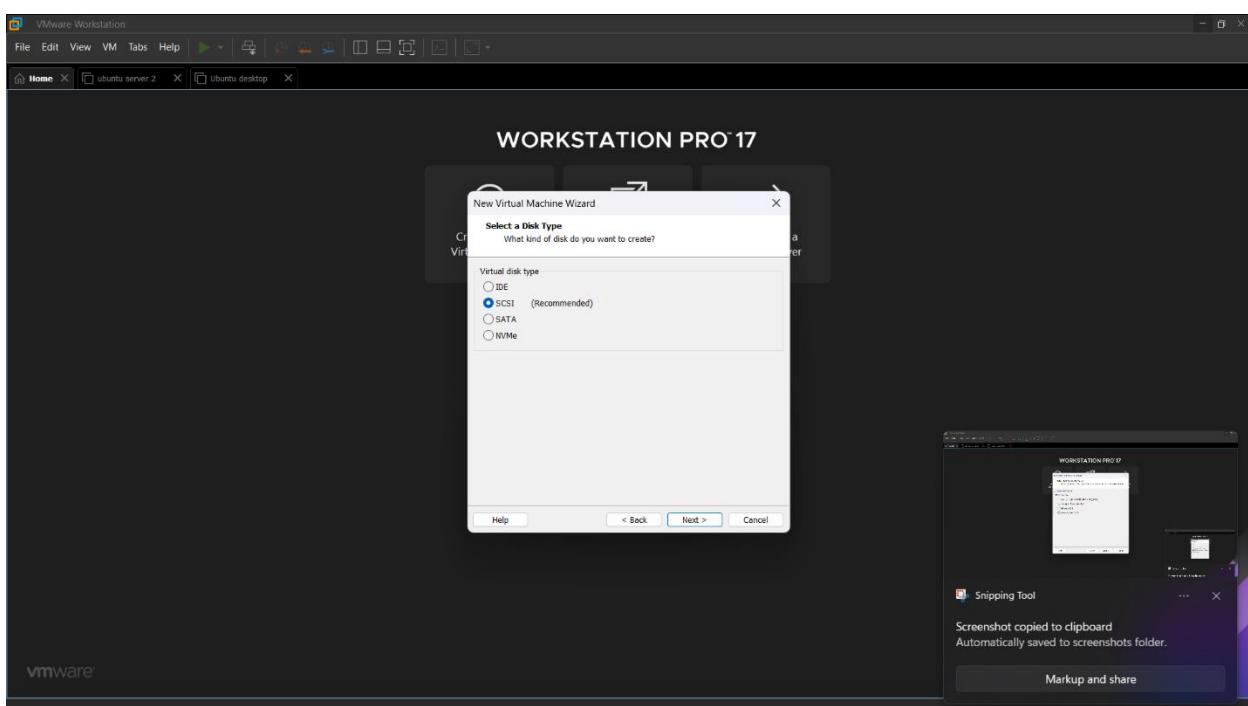
در مرحله پنجم باید مقدار رم مموری ای که برای ماشین خود در نظر داریم انتخاب کنیم. برای **ubuntu** مقدار پیشنهادی آن 4gb است.



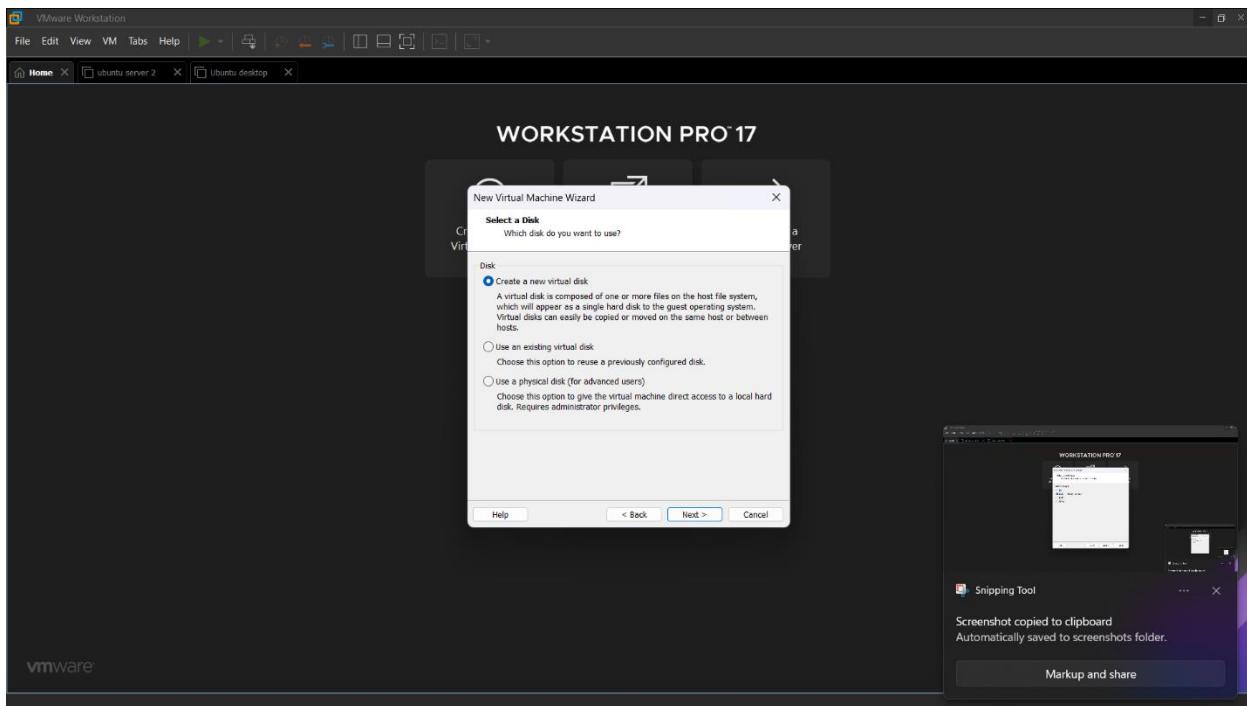
در مرحله پنجم باید شیوه اتصال ماشین مجازی به شبکه خود را انتخاب کنیم. که دو راه پیشنهادی آن **nat** (اتصال از شبکه سیستم) و **bridged** (اتصال مستقیم از روتر یا مودم).



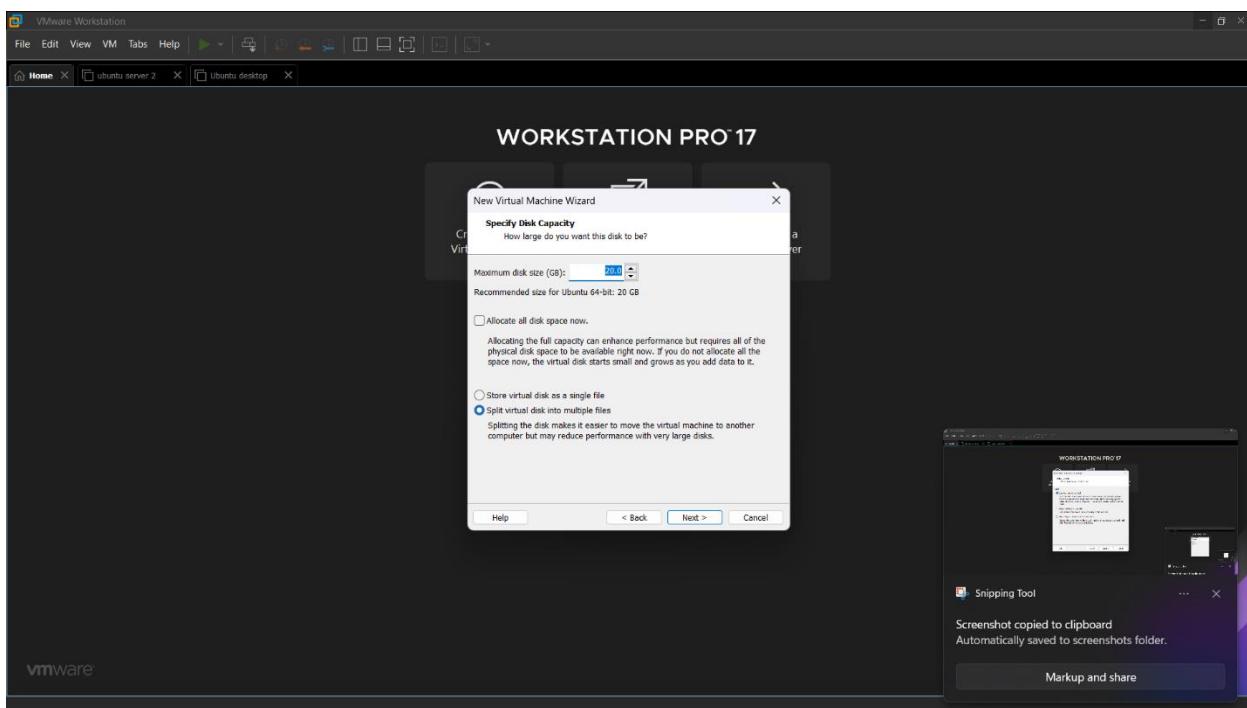
در این مرحله نوع کنترلر I/O خود را انتخاب میکنیم.



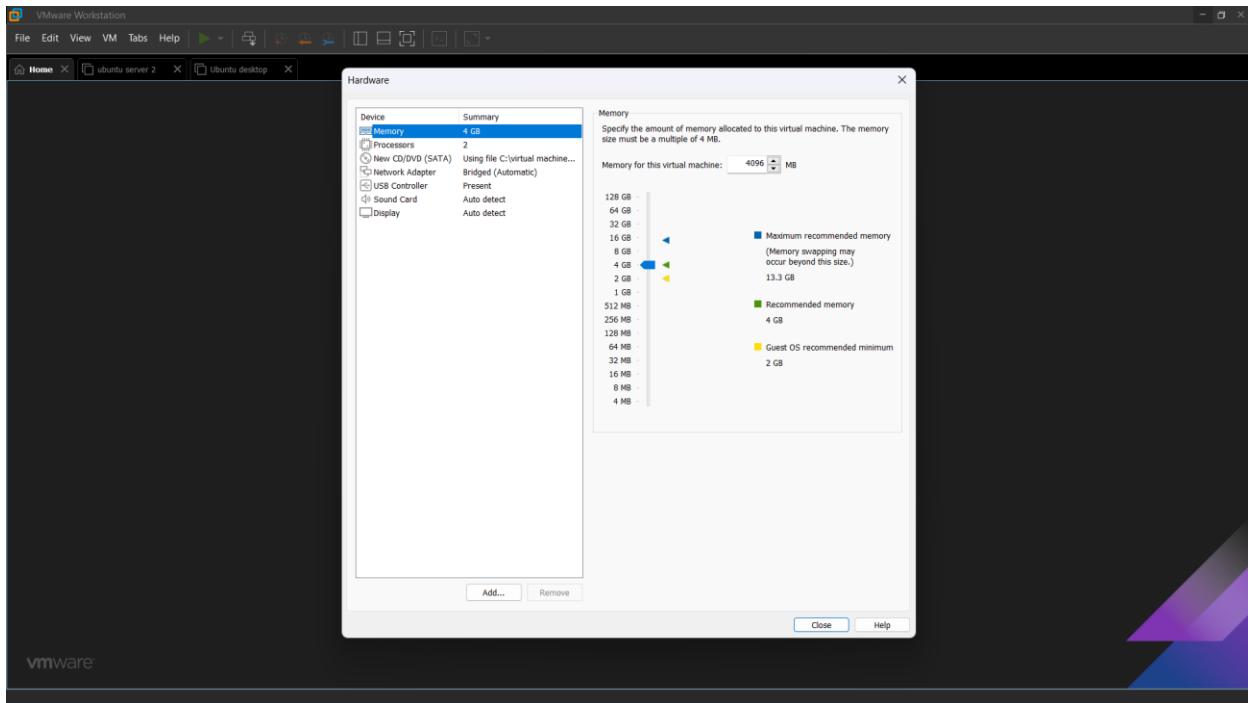
در این مرحله نوع هارد دیسک خود را مشخص میکنیم.



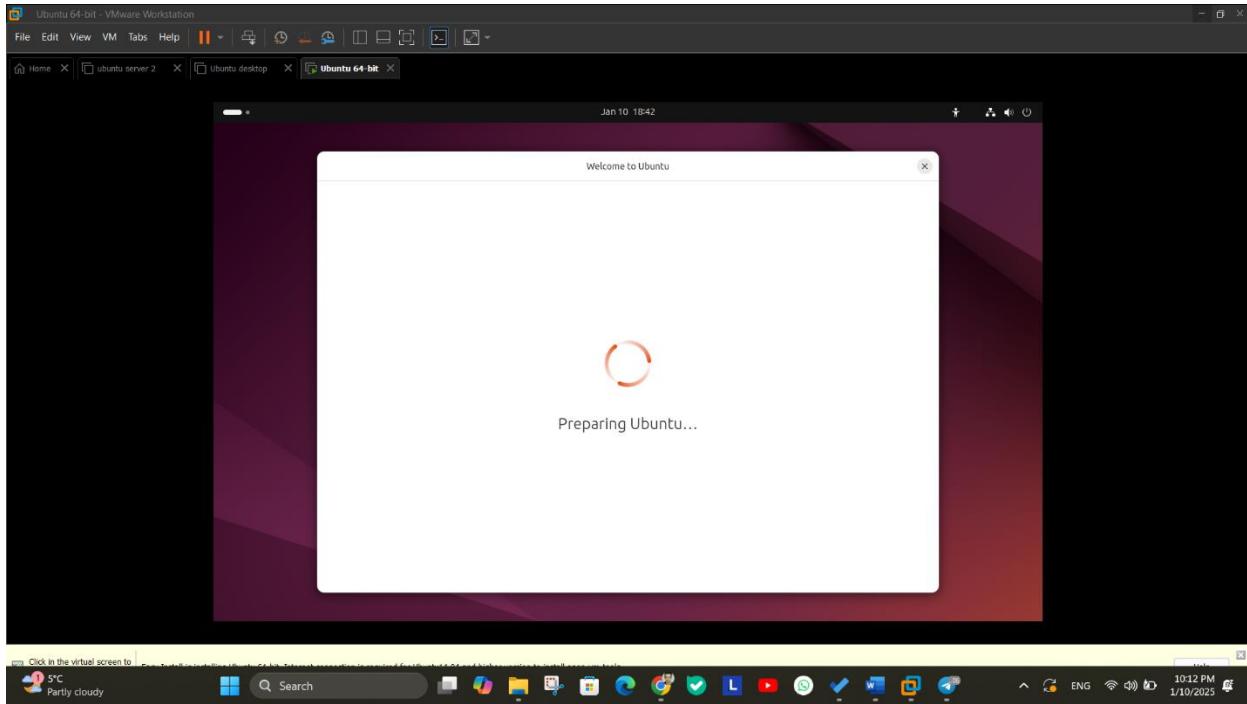
در این مرحله شیوه نصب ماشین مجازی را روی هارد دیسک را انتخاب میکنیم. (ماشین مجازی روی دیسک مجازی نصب شود یا روی دیسک مجازی قابل جایه جایی یا مستقیم روی دیسک فیزیکی)



در این مرحله میزان حجم هارد دیسکی که موردنیاز ماشین مجازی است را
انتخاب میکنیم.



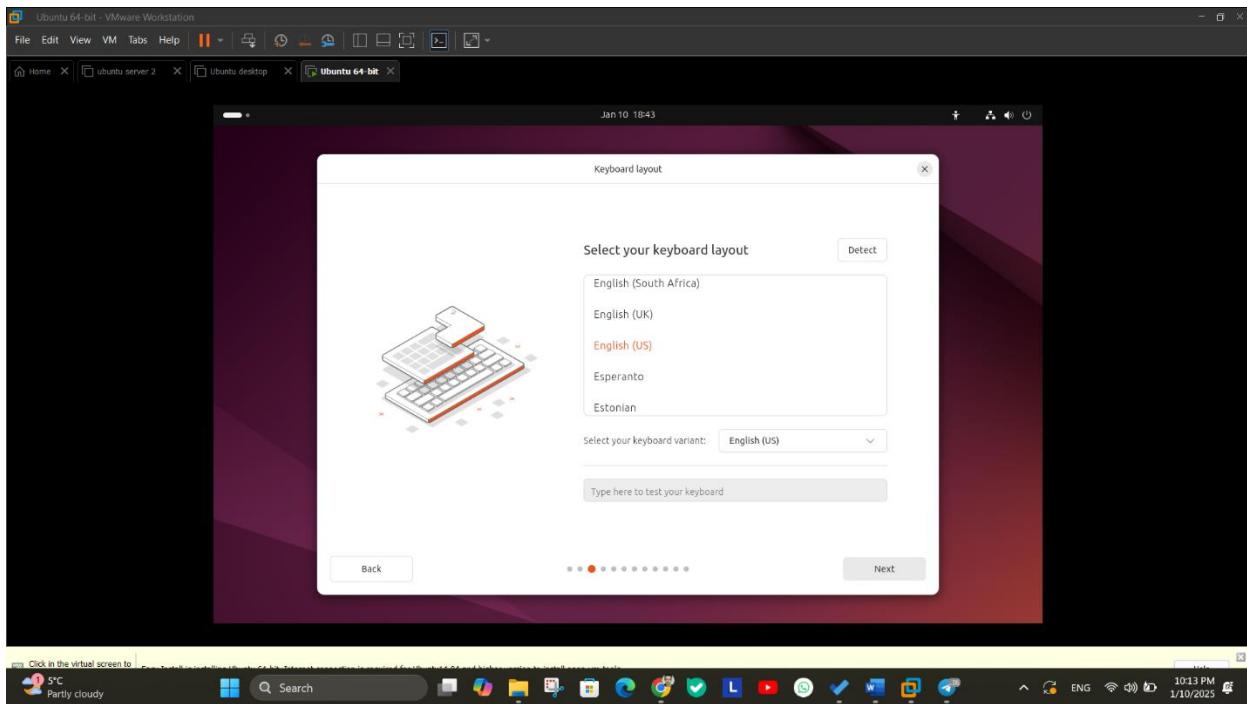
و در اخر به قسمت مدیریت فایل سخت افزار میرسیم که میتوانیم سخت
افزاری ماشین مجازی خود را تغییر بدھیم یا سخت افزاری به آن اضافه کنیم.

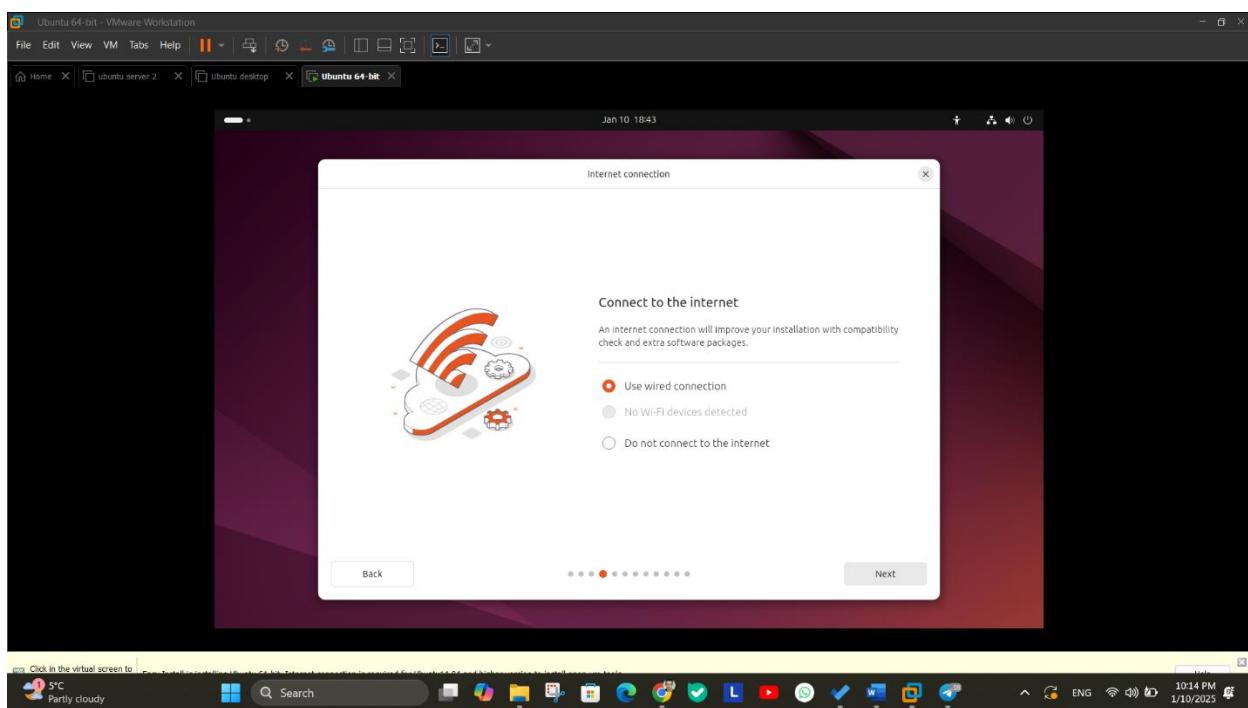
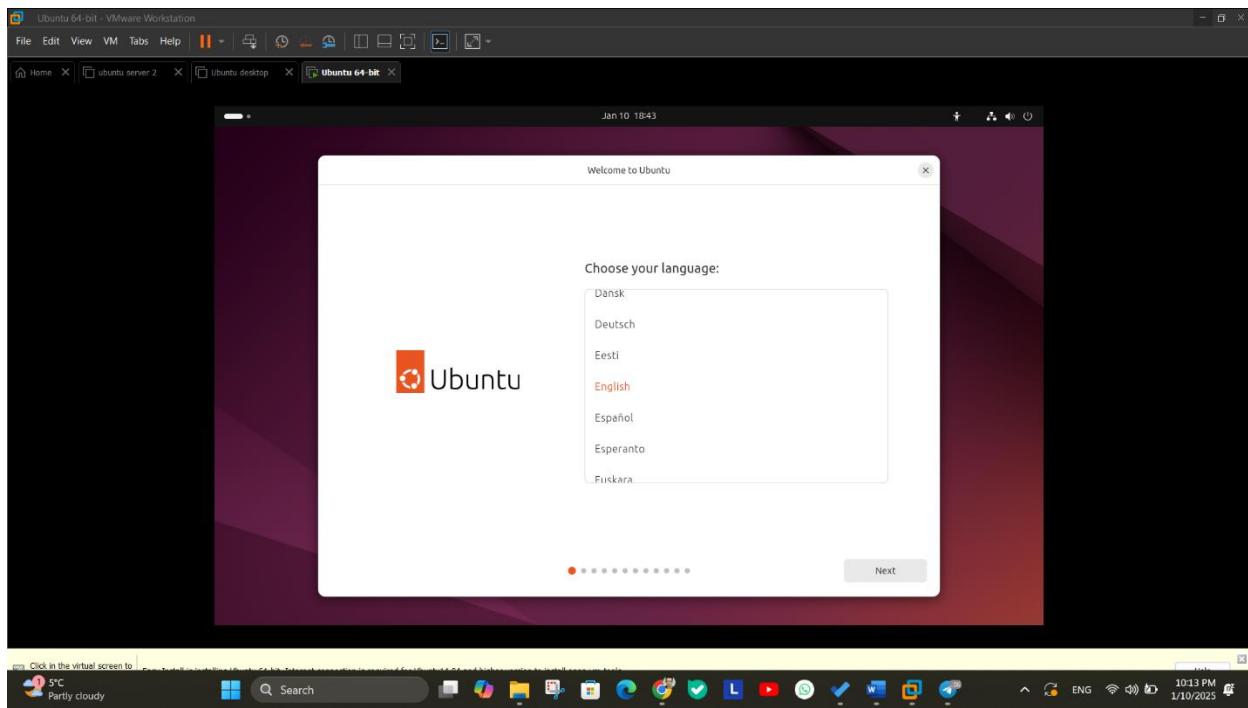


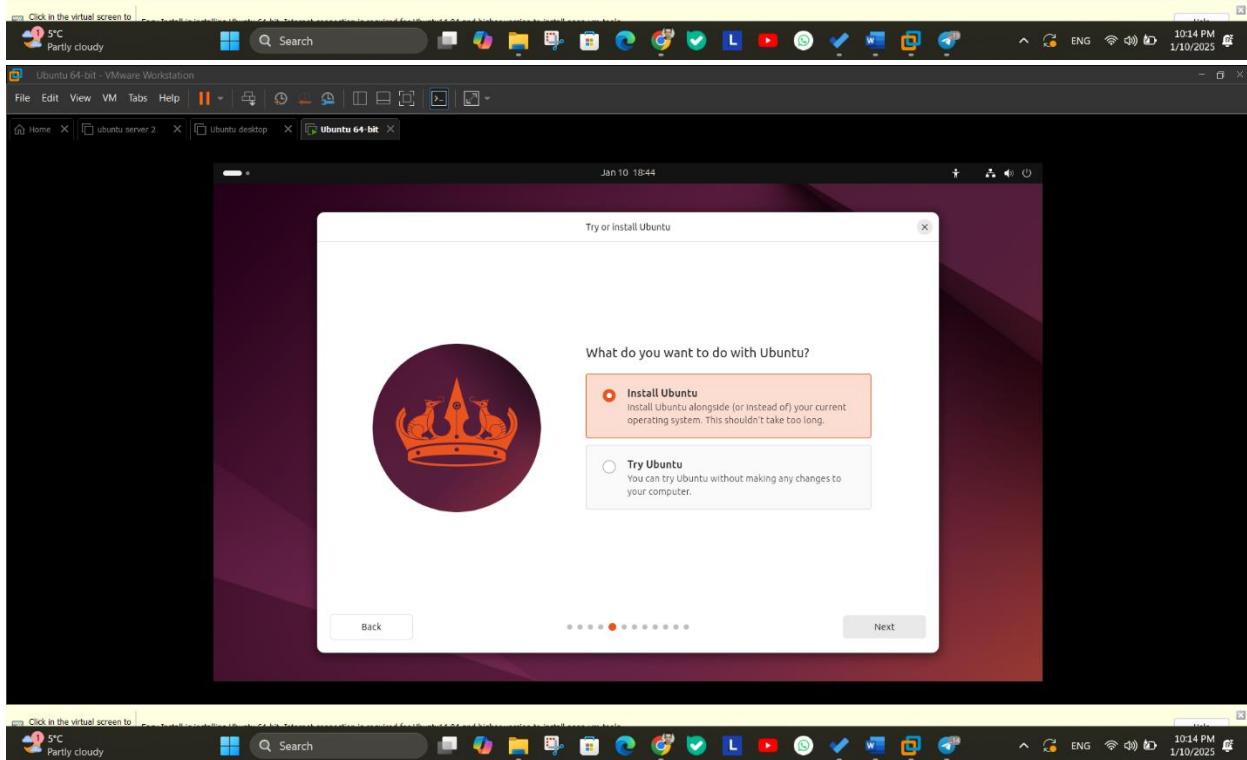
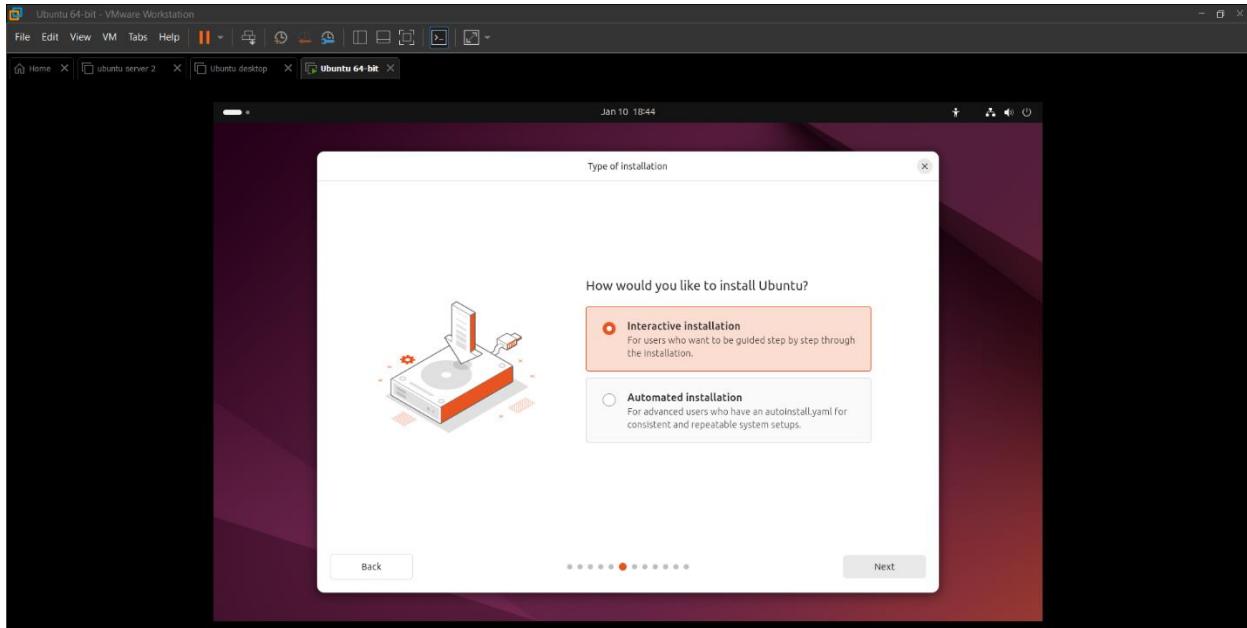
سپس با زدن `install` ماشین شروع به نصب شدن میکند.

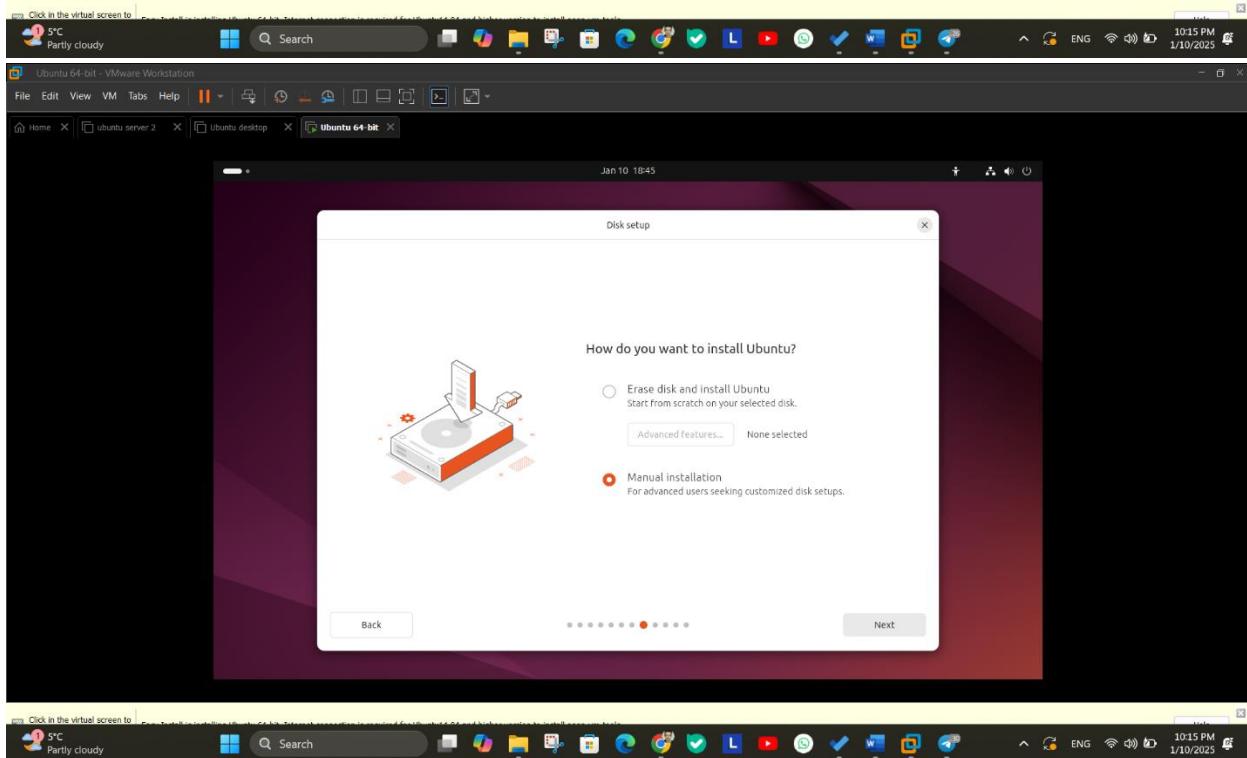
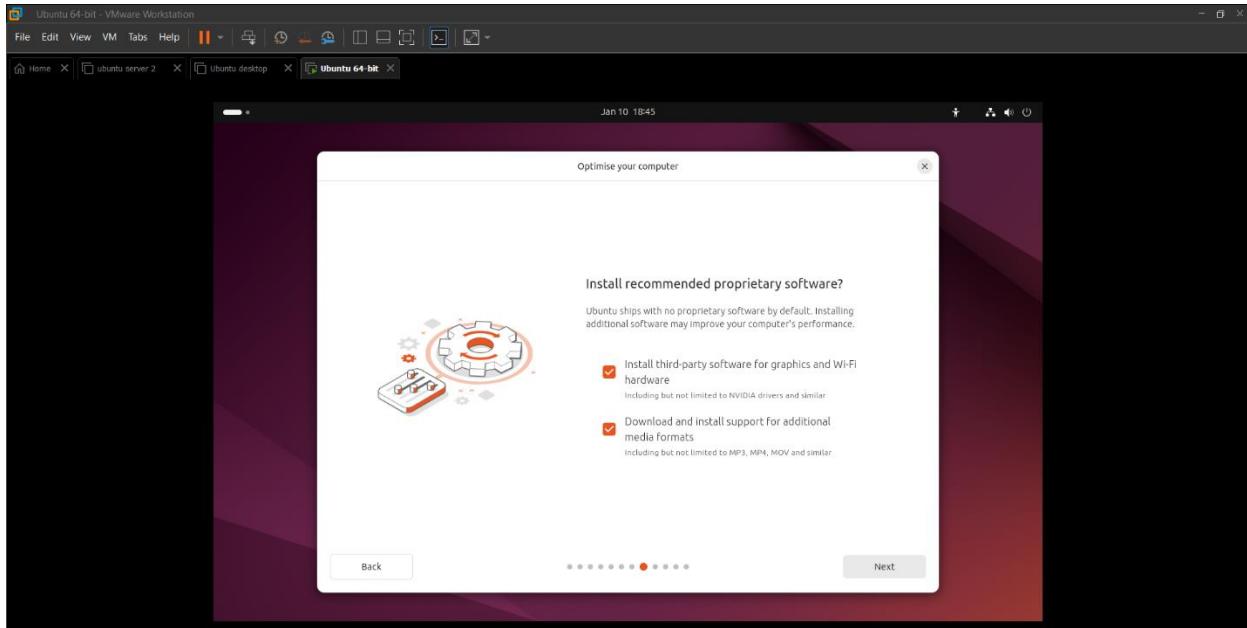
بعد از نصب `vm` نوبت نصب `ubuntu desktop` است؛ که به صورت تصاویر

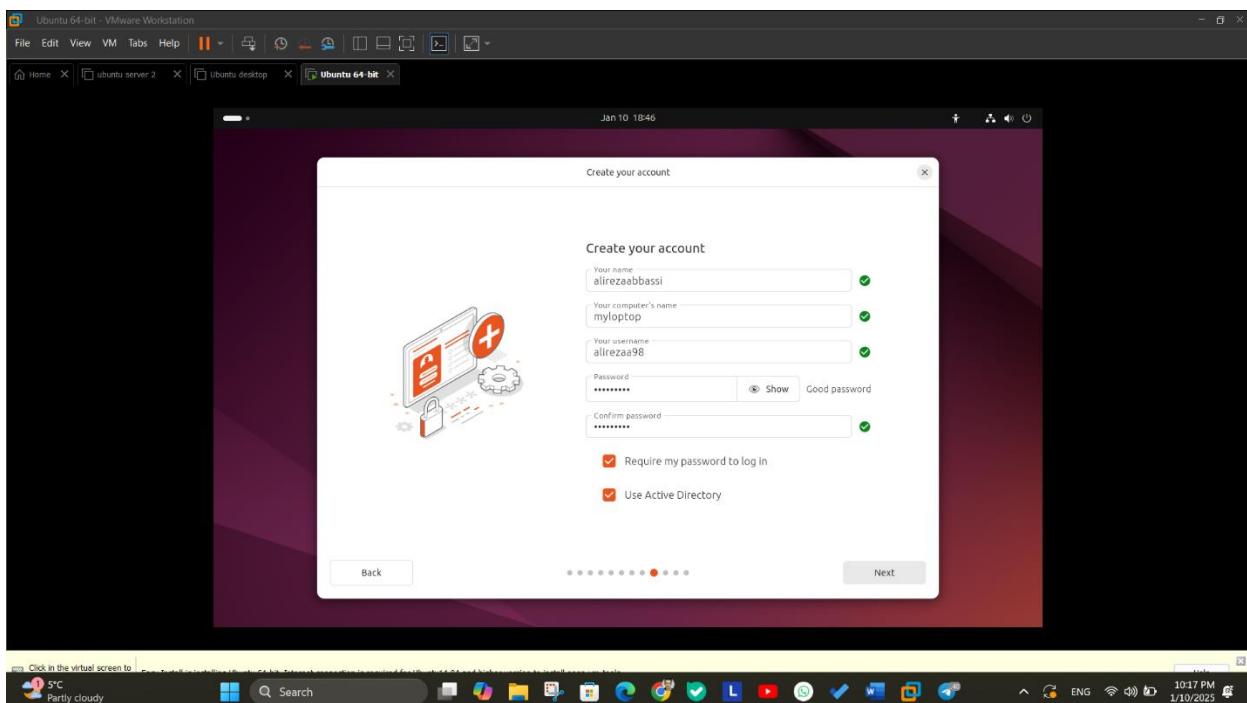
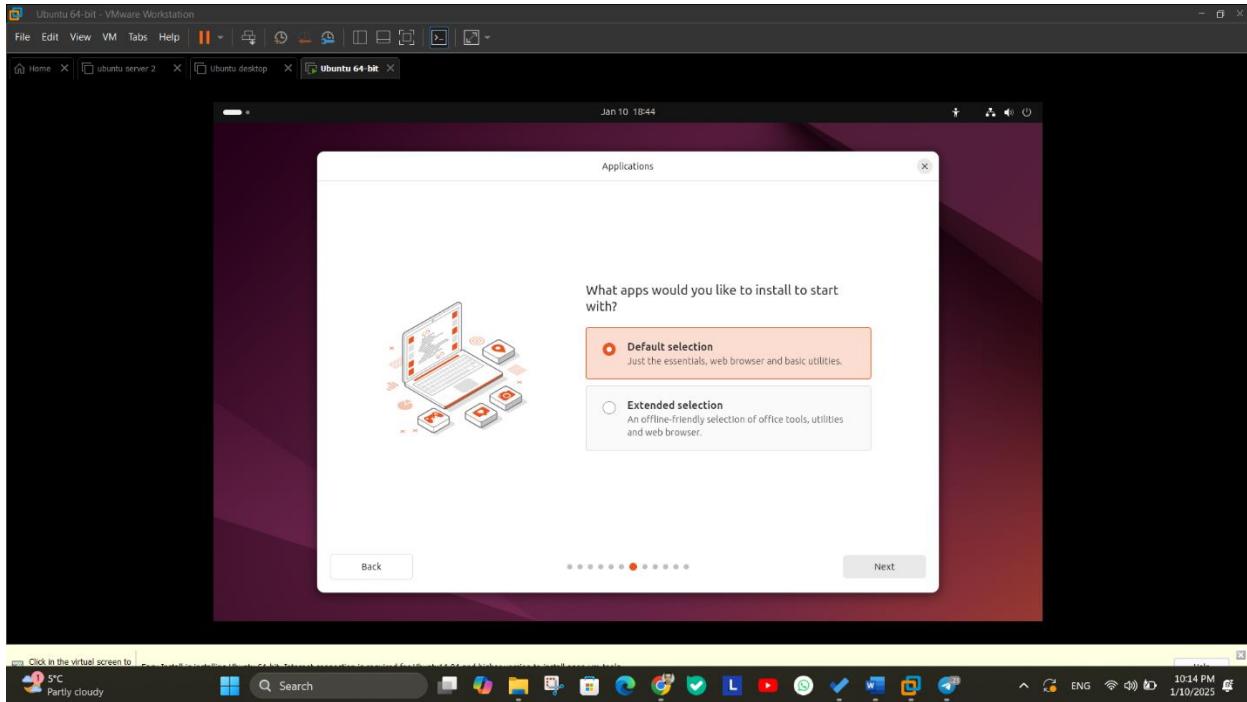
زیر عمل میکنیم

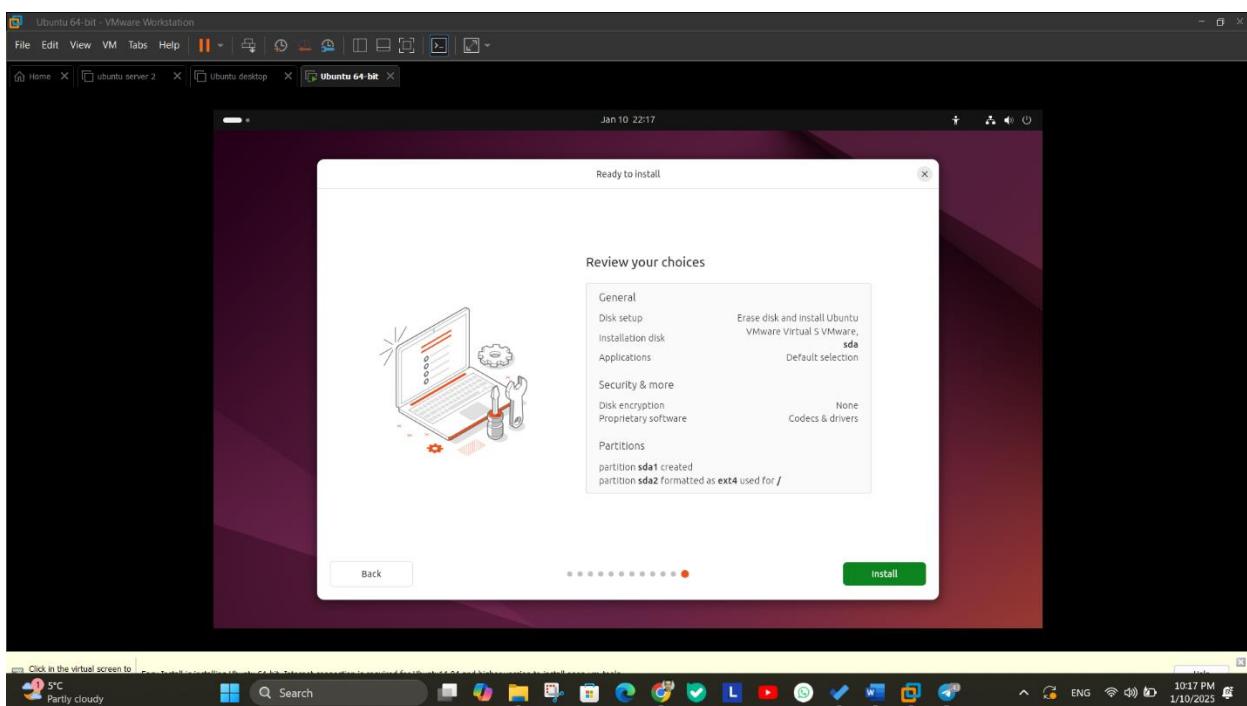
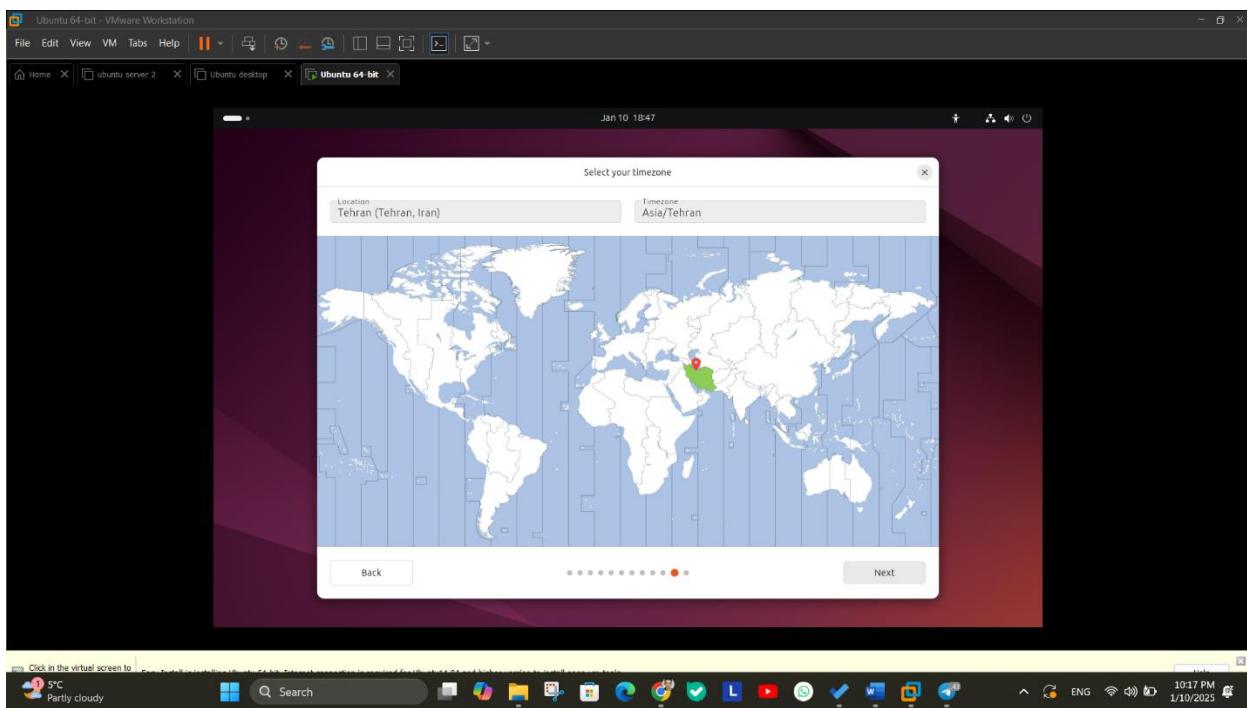












و در انتها با زدن **install** منتظر نصب **ubuntu** میشویم.

گام دوم: نصب docker

مقدمه ای درباره داکر:

یک پلتفرم منبع باز (Open Source) است که توسعه دهنده‌گان و مهندسان عملیات (DevOps) را قادر می‌سازد تا برنامه‌ها و سرویس‌های خود را در قالب واحدهای مستقل به نام کانتینر (Container) بسته‌بندی، اجرا و مدیریت کنند. این ابزار با بهره‌گیری از فناوری مجازی‌سازی در سطح سیستم‌عامل (OS-Level Virtualization)، محیطی یکپارچه، سبک و کارآمد برای اجرای برنامه‌ها فراهم می‌کند.

کانتینرها به گونه‌ای طراحی شده‌اند که شامل تمام اجزای موردنیاز یک برنامه، از جمله کد، کتابخانه‌ها و وابستگی‌ها، باشند. این ویژگی به توسعه دهنده‌گان اجازه می‌دهد تا برنامه‌های خود را به صورت قابل حمل و مستقل از محیط زیرساختی اجرا کنند. به عبارت دیگر، یک کانتینر Docker می‌تواند روی هر سیستمی که از Docker پشتیبانی می‌کند (مانند سرورها، لپ‌تاپ‌ها یا حتی محیط‌های ابری) بدون نیاز به تغییر اجرا شود.

برخی از ویژگی‌های بر جسته Docker عبارت‌اند از:

- انعطاف‌پذیری و مقیاس‌پذیری: امکان ایجاد و مدیریت هزاران کانتینر در مقیاس بزرگ.

- قابل حمل بودن: برنامه‌ها می‌توانند بدون وابستگی به زیرساخت اجرا شوند.

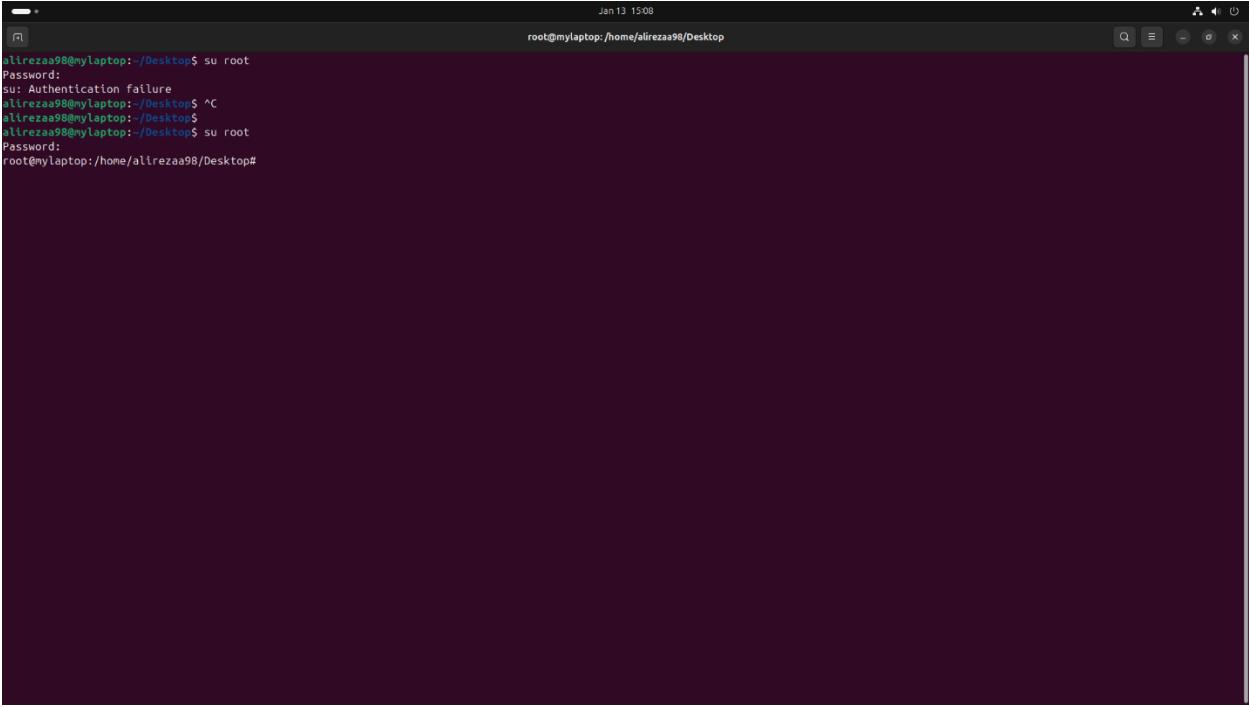
- کاهش پیچیدگی‌ها: استفاده از تصاویر (Images) آماده و ابزارهای ساده‌سازی عملیات.

Docker به‌طور گسترده در توسعه نرم‌افزار، تست، و اجرای برنامه‌ها در محیط‌های تولیدی و ابری مورد استفاده قرار می‌گیرد. این ابزار تحول بزرگی در فرایندهای توسعه نرم‌افزار ایجاد کرده و به تیم‌ها امکان همکاری بهتر و ارائه سریع‌تر محصولات را داده است.

مراحل نصب:

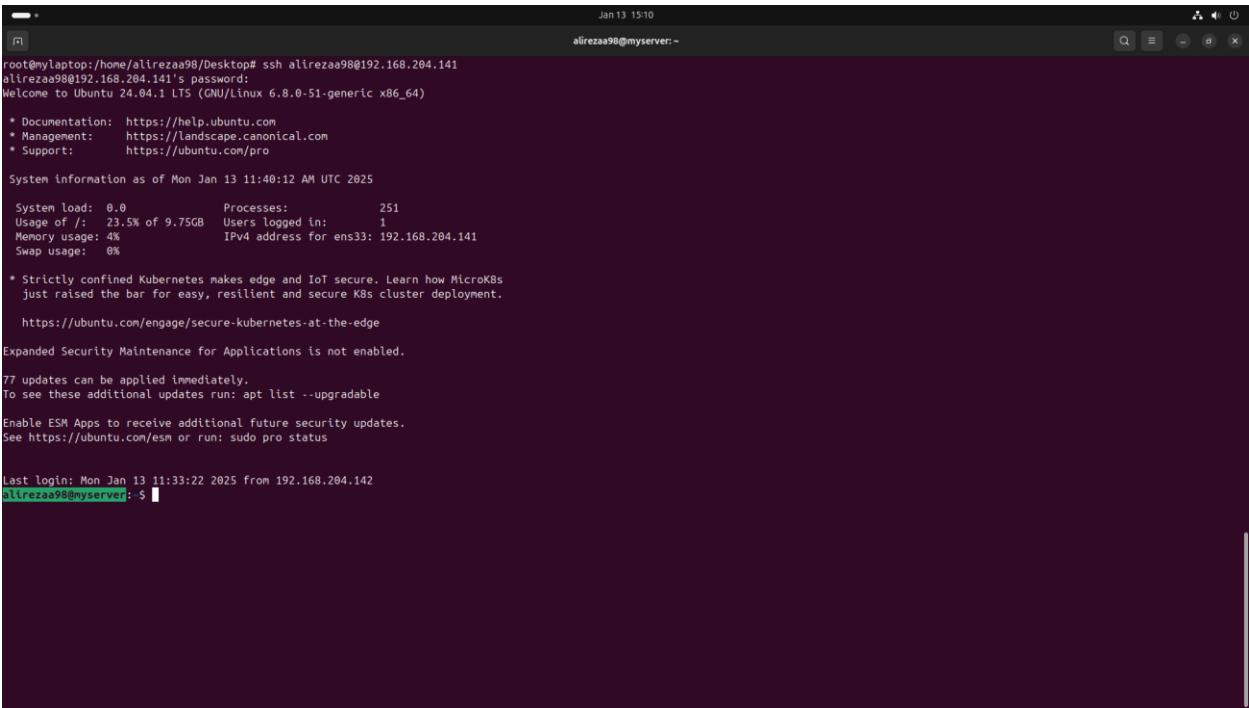
برای نصب داکر روش‌های زیادی وجود دارد که من از طریق نصب ریپازیتوری داکر آن را نصب می‌کنم.

مراحل نصب:



```
alirezaa98@mylaptop:~/Desktop$ su root
Password:
su: Authentication failure
alirezaa98@mylaptop:~/Desktop$ ^C
alirezaa98@mylaptop:~/Desktop$ alirezaa98@mylaptop:~/Desktop$ su root
Password:
root@mylaptop:/home/alirezaa98/Desktop#
```

ابتدا وارد مود super user میشویم



```
root@mylaptop:/home/alirezaa98/Desktop# ssh alirezaa98@192.168.204.141
alirezaa98@192.168.204.141's password:
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-51-generic x86_64)

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

System information as of Mon Jan 13 11:40:12 AM UTC 2025

System load: 0.0 Processes: 251
Usage of /: 23.5% of 9.75GB Users logged in: 1
Memory usage: 4% IPv4 address for ens33: 192.168.204.141
Swap usage: 0%

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

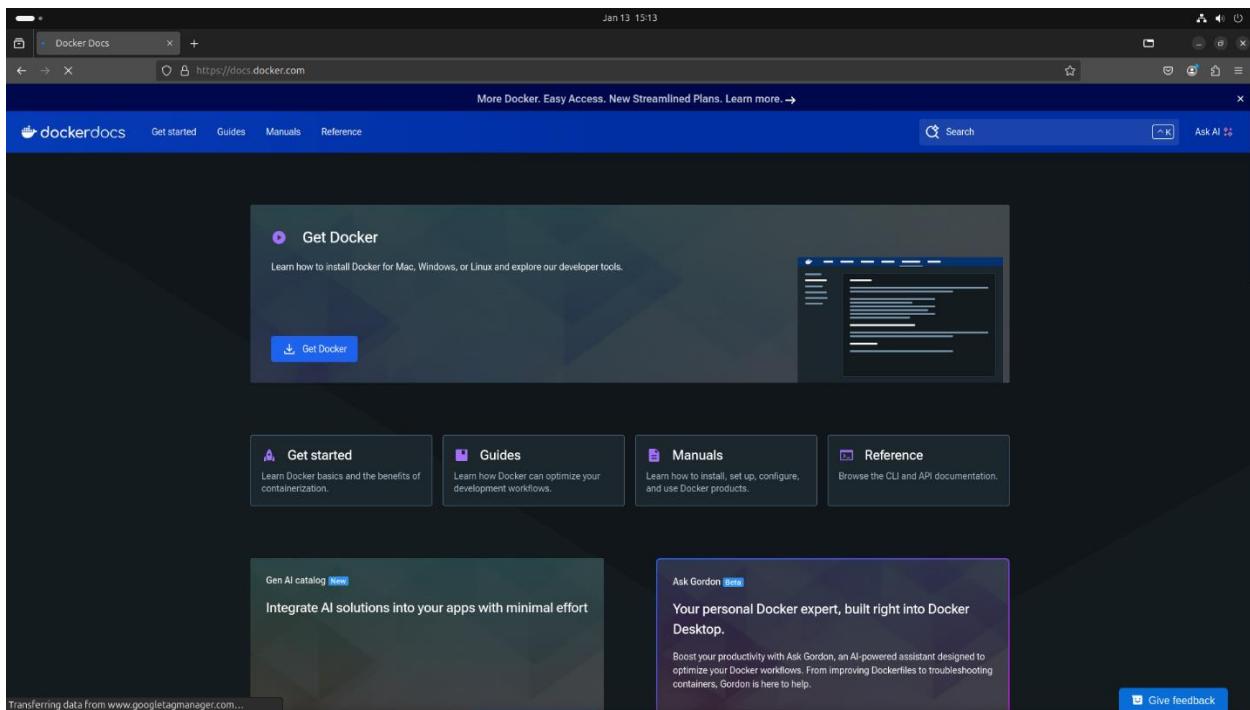
Expanded Security Maintenance for Applications is not enabled.

77 updates can be applied immediately,
To see these additional updates run: apt list --upgradable

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

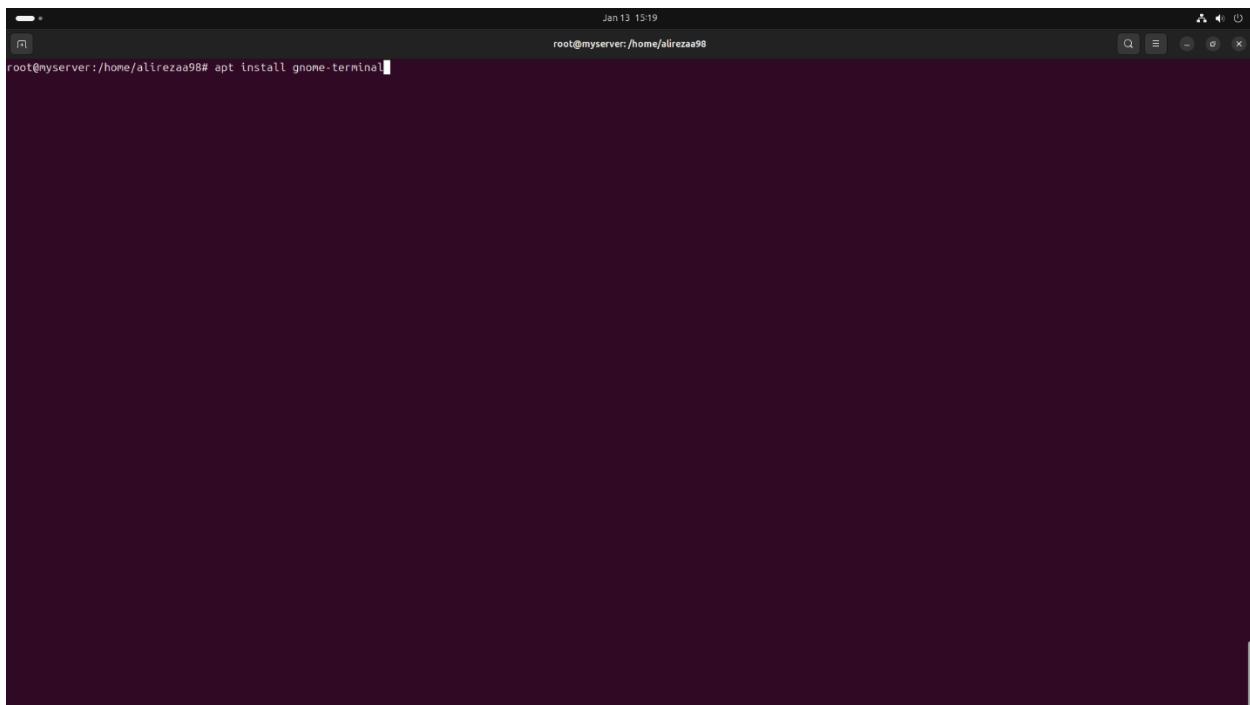
Last login: Mon Jan 13 11:33:22 2025 from 192.168.204.142
alirezaa98@myserver: ~
```

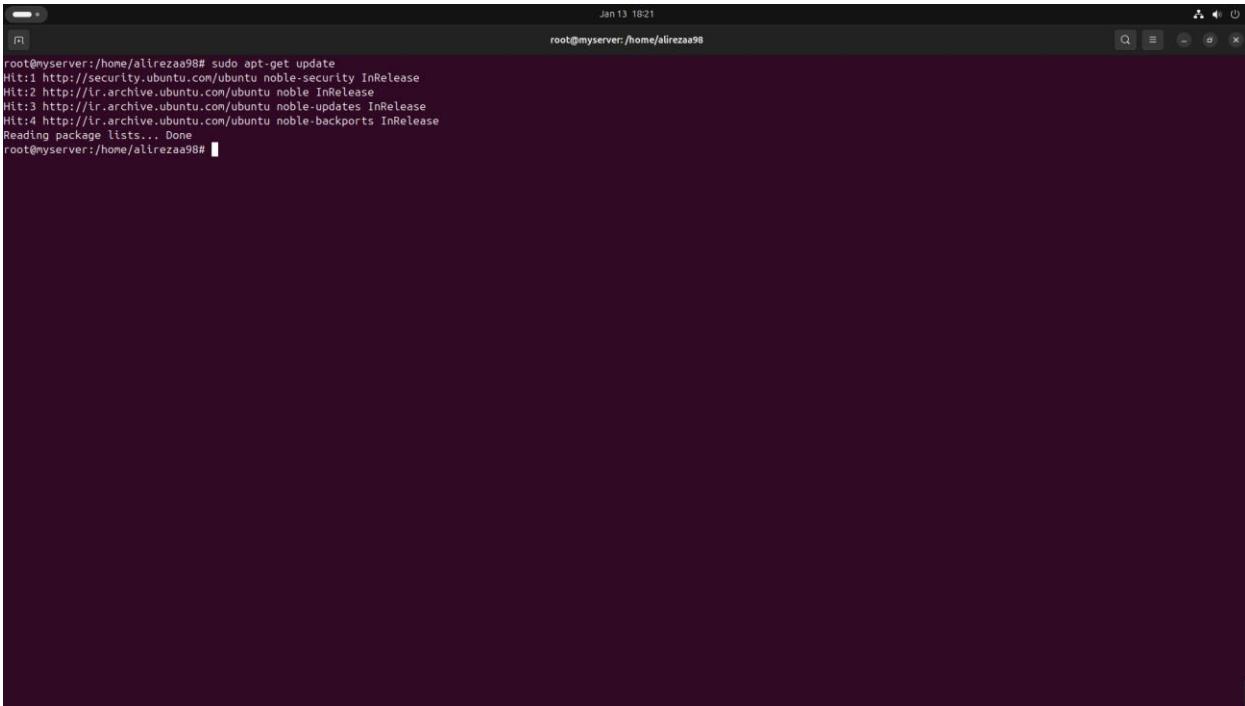
سپس از طریق پروتکل ssh به سرور خود متصل میشویم



سپس از طریق سایت docs.docker.com و راهنمای نصب ریپازیتوری داکر شروع به نصب آن میکنیم.

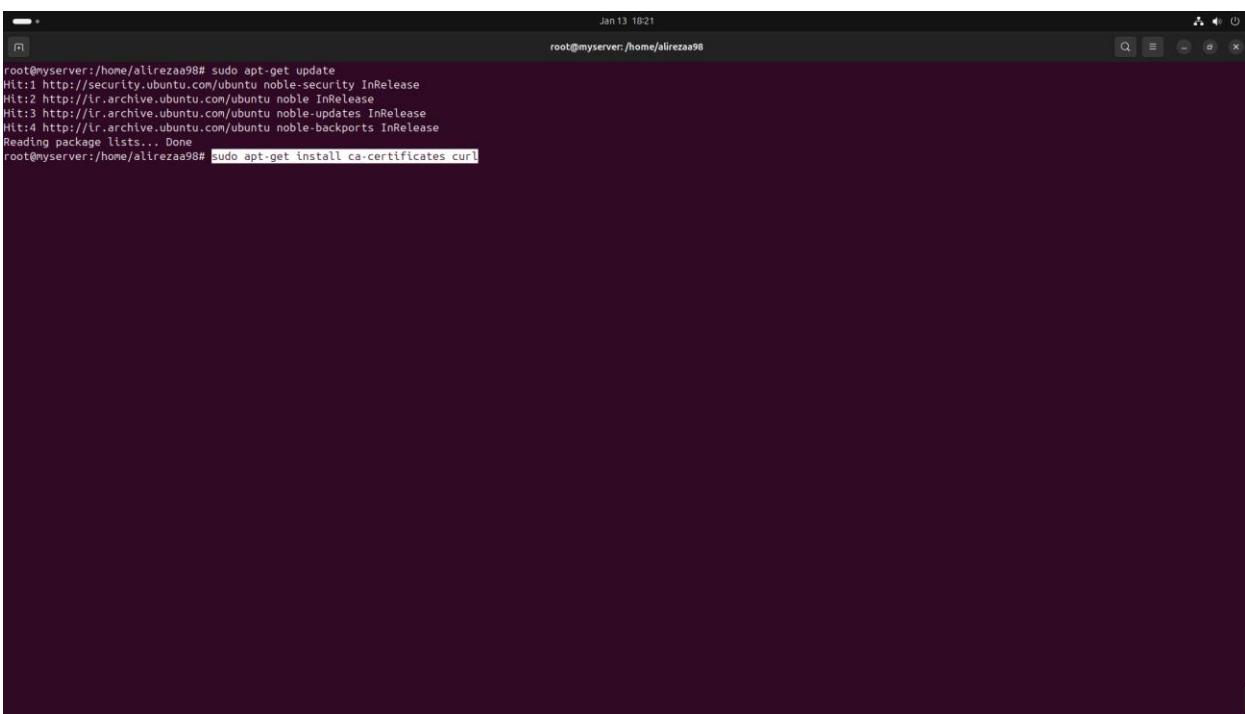
برای این کار در توزیع اوبونتو باید ابتدا gnome-terminal را نصب کنیم.





```
Jan 13 18:21
root@myserver:/home/airezaa98# sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://fr.archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://fr.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://fr.archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
root@myserver:/home/airezaa98#
```

ابتدا با ران دستور ریپازیتوری خود OS را آپدیت میکنیم

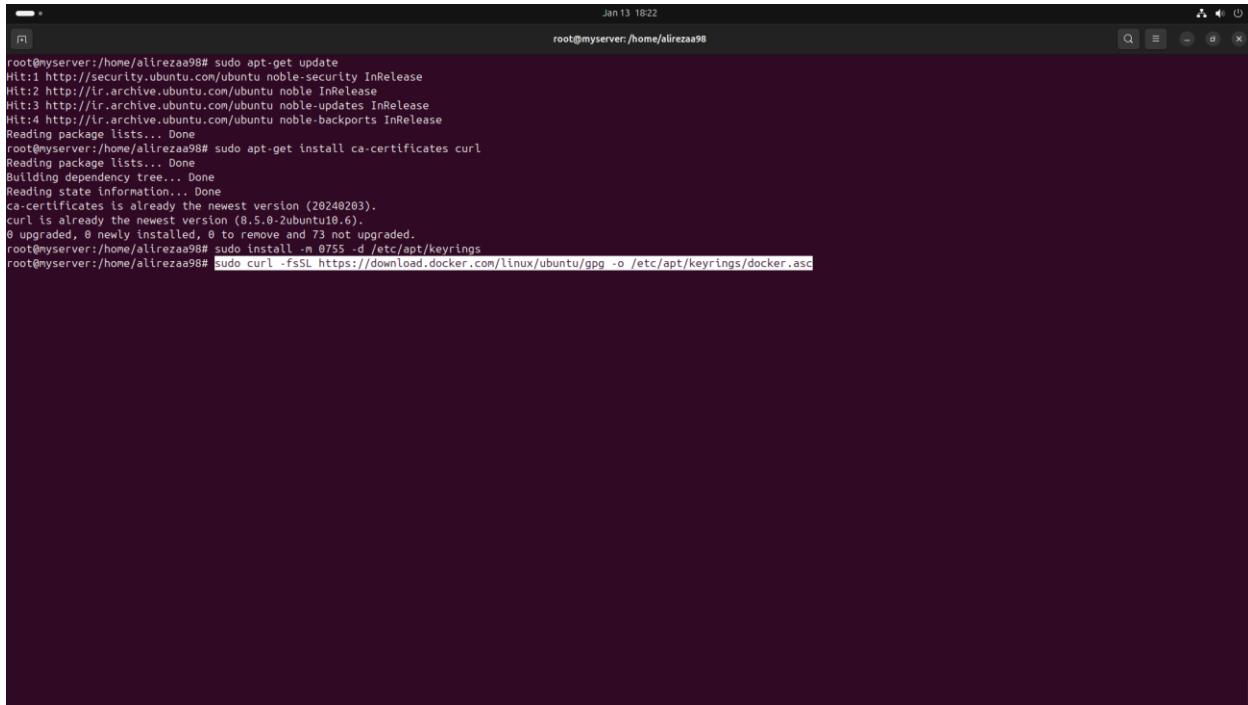


```
Jan 13 18:21
root@myserver:/home/airezaa98# sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://fr.archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://fr.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://fr.archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
root@myserver:/home/airezaa98# sudo apt-get install ca-certificates curl
```

با این دستور این موارد را نصب میکنیم:

: گواهی‌های امنیتی مورد نیاز برای برقراری ارتباط امن (SSL/TLS) با سرورهای مختلف.

: ابزاری برای انتقال داده‌ها از طریق پروتکل‌های مختلف مانند HTTP و HTTPS که در اینجا برای دانلود فایل‌ها استفاده می‌شود.



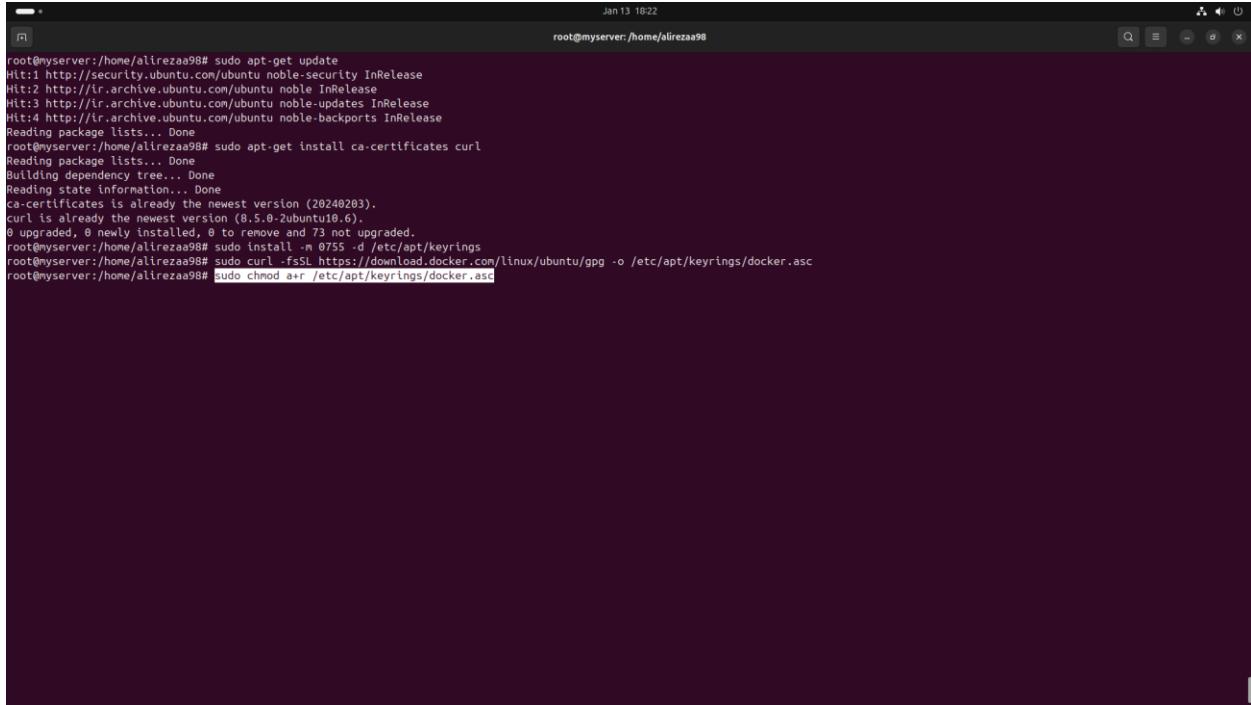
```
root@myserver:/home/alirezaa98# sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
root@myserver:/home/alirezaa98# sudo apt-get install ca-certificates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
curl is already the newest version (8.5.0-2ubuntu10.6).
0 upgraded, 0 newly installed, 0 to remove and 73 not upgraded.
root@myserver:/home/alirezaa98# sudo install -m 0755 -d /etc/apt/keyrings
root@myserver:/home/alirezaa98# sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
```

با این دو دستور ابتدا یک دایرکتوری جدید به نام **/etc/apt/keyrings** ایجاد می‌کند و مجوزهای آن را تنظیم می‌کند:

-m 0755: تعیین مجوزهای دایرکتوری (خواندن، نوشتن و اجرا برای مالک؛ خواندن و اجرا برای سایرین).

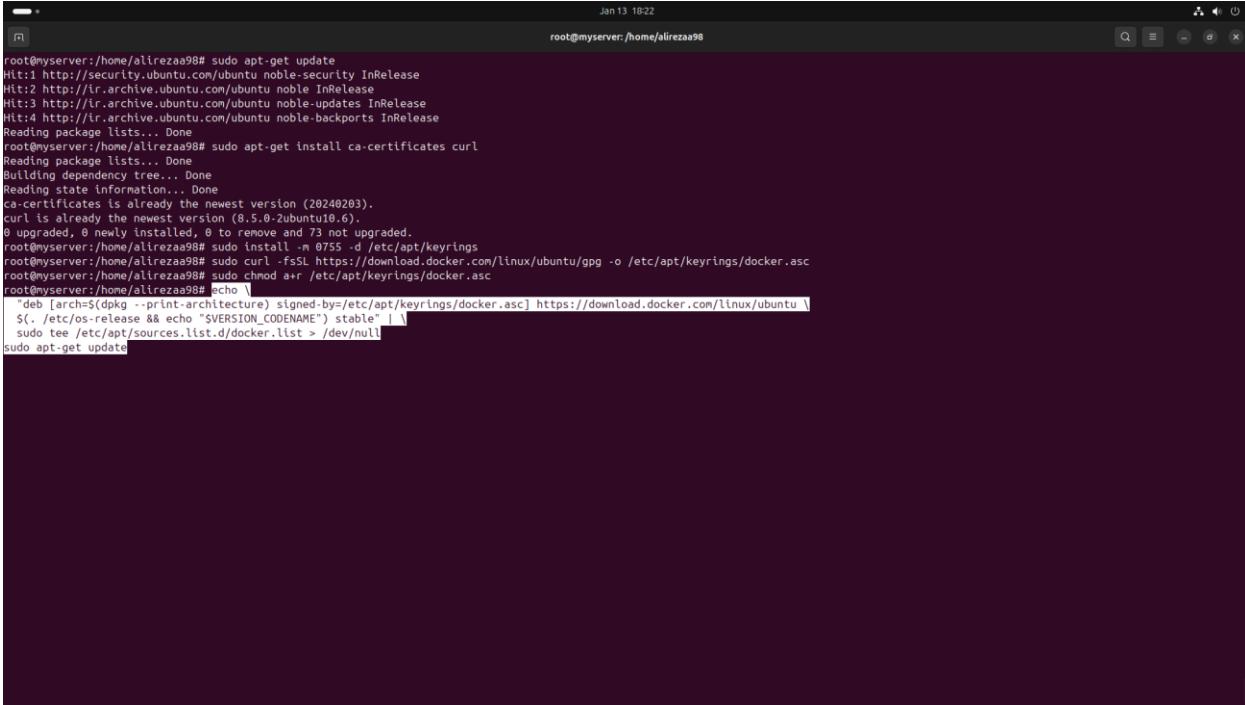
سپس دومین دستور یک کلید GPG را از آدرس مورد نظر (https://download.docker.com/linux/ubuntu/gpg) دانلود

می‌کند و آن را در مسیر /etc/apt/keyrings/docker.asc ذخیره می‌کند.



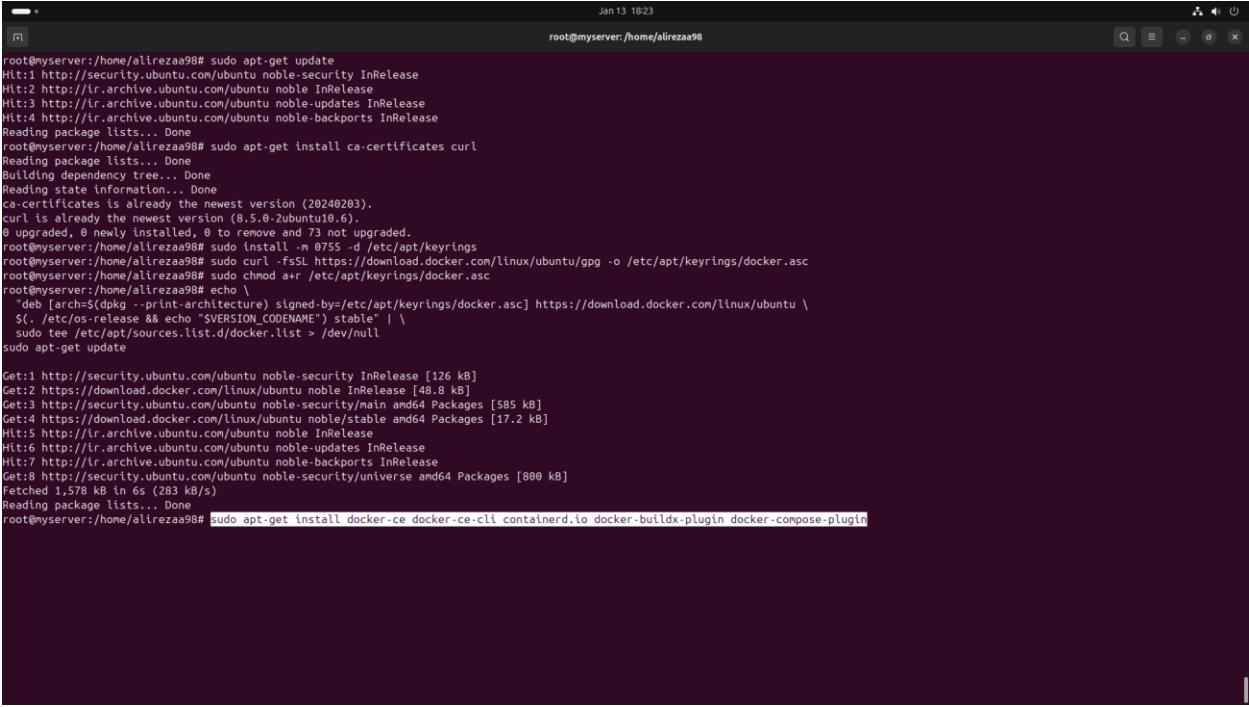
```
root@myserver:/home/alirezaa98# sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://fr.archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://fr.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://fr.archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
root@myserver:/home/alirezaa98# sudo apt-get install ca-certificates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
curl is already the newest version (8.5.0-2ubuntu10.6).
0 upgraded, 0 newly installed, 0 to remove and 73 not upgraded.
root@myserver:/home/alirezaa98# sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
root@myserver:/home/alirezaa98# sudo chmod a+r /etc/apt/keyrings/docker.asc
```

این دستور اجازه خواندن (read) را به همه کاربران برای فایل etc/apt/keyrings/docker.asc می‌دهد.



```
Jan 13 16:22
root@myserver:/home/alirezaa98#
root@myserver:/home/alirezaa98# sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://fr.archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://fr.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://fr.archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
root@myserver:/home/alirezaa98# sudo apt-get install ca-certificates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
curl is already the newest version (8.5.0-2ubuntu10.6).
0 upgraded, 0 newly installed, 0 to remove and 73 not upgraded.
root@myserver:/home/alirezaa98# sudo install -m 0755 -d /etc/apt/keyrings
root@myserver:/home/alirezaa98# sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | /etc/apt/keyrings/docker.asc
root@myserver:/home/alirezaa98# sudo chmod a+r /etc/apt/keyrings/docker.asc
root@myserver:/home/alirezaa98# gpg --debu
deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

این دستورات ریپازیتوری Docker را با توجه به معماری و نسخه سیستم عامل شما به سیستم اضافه می‌کنند. سپس اطلاعات بسته‌ها را از این ریپازیتوری دریافت می‌کنند و در نتیجه، شما می‌توانید مستقیماً نسخه‌های پایدار Docker را نصب کنید.



```
root@myserver:/home/alirezaa98# sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://fr.archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://fr.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://fr.archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
root@myserver:/home/alirezaa98# sudo apt-get install ca-certificates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
curl is already the newest version (8.5.0-2ubuntu10.6).
0 upgraded, 0 newly installed, 0 to remove and 73 not upgraded.
root@myserver:/home/alirezaa98# sudo install -m 0755 -d /etc/apt/keyrings
root@myserver:/home/alirezaa98# sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
root@myserver:/home/alirezaa98# sudo chmod a+r /etc/apt/keyrings/docker.asc
root@myserver:/home/alirezaa98# echo 'deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "VERSION_CODENAME") stable' | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:2 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:3 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [585 kB]
Get:4 https://download.docker.com/linux/ubuntu/noble/stable amd64 Packages [17.2 kB]
Hit:5 http://fr.archive.ubuntu.com/ubuntu noble InRelease
Hit:6 http://fr.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:7 http://fr.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:8 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [800 kB]
Fetched 1,578 kB in 6s (283 kB/s)
Reading package lists... Done
root@myserver:/home/alirezaa98# sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

و در نهایت داکر و بسته های مورد نیاز آن را نصب میکنیم.

```**docker-ce**```

(Docker Community Edition)

شامل موتور Docker (Docker Engine) می باشد.

**b. docker-ce-cli**

رابط خط فرمان (CLI) برای تعامل با Docker Engine که به شما اجازه docker build و docker run مانند Docker می دهد تا از دستورات استفاده کنید.

**containerd.io**

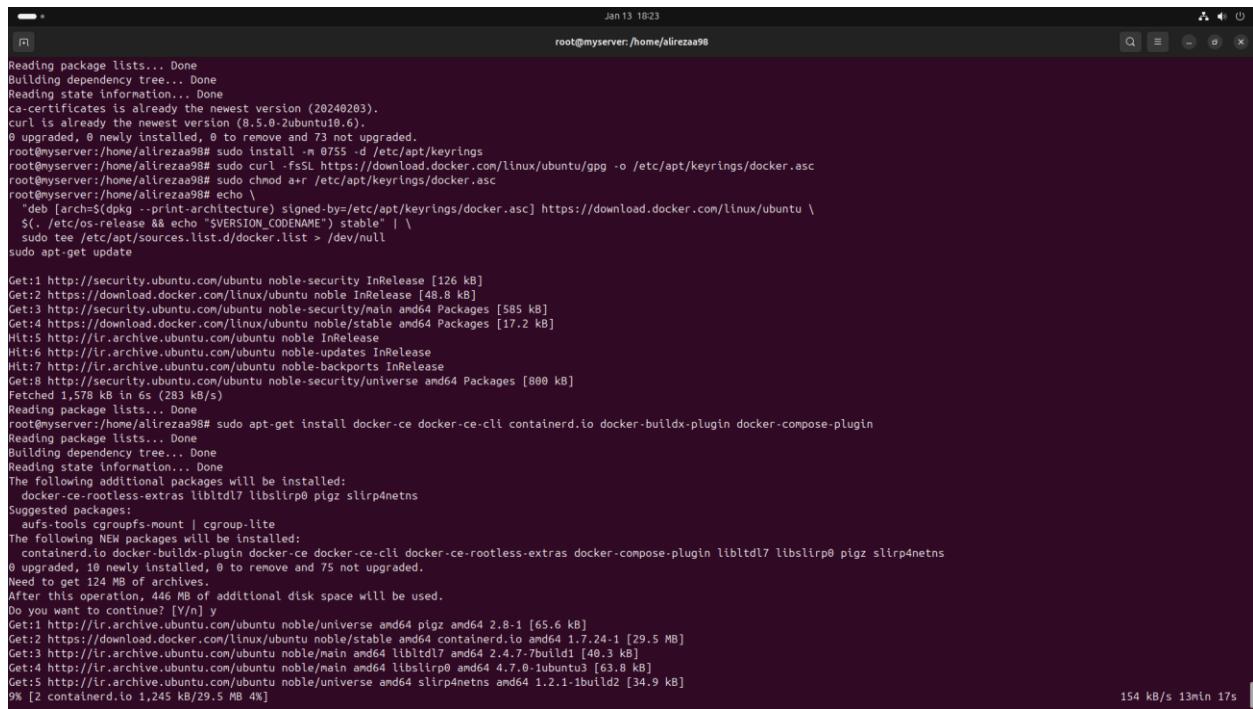
یک Docker runtime container مستقل است که توسط استفاده می‌شود که مسئول اجرای و مدیریت کانتینرها است.

## docker-buildx-plugin

افزونه Docker image برای ساخت Buildx ها

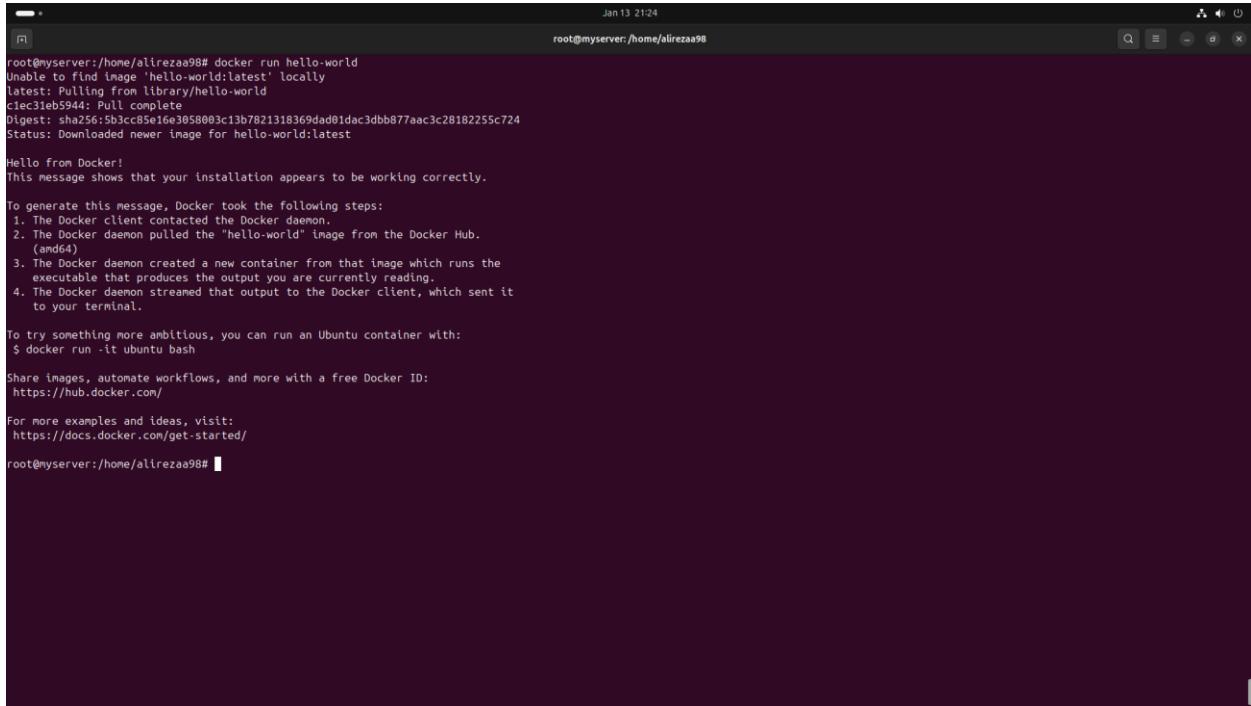
## docker-compose-plugin

نسخه جدید Docker Compose به صورت افزونه که اجازه می‌دهد از فایل‌های YAML برای تعریف و اجرای سرویس‌های چندکانتینری استفاده کنید.



```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
curl is already the newest version (8.5.0-2ubuntu10.6).
0 upgraded, 0 newly installed, 0 to remove and 73 not upgraded.
root@myserver:/home/alirezaa98# sudo install -m 0755 -d /etc/apt/keyrings
root@myserver:/home/alirezaa98# sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
root@myserver:/home/alirezaa98# sudo chmod a+r /etc/apt/keyrings/docker.asc
root@myserver:/home/alirezaa98# echo '
 "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
 $(/etc/os-release && echo "$VERSION_CODENAME") stable" | \
 sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
'
root@myserver:/home/alirezaa98# sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:2 https://download.docker.com/linux/ubuntu noble InRelease [49.8 kB]
Get:3 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [585 kB]
Get:4 https://download.docker.com/linux/ubuntu/noble/stable amd64 Packages [17.2 kB]
Hit:5 http://fr.archive.ubuntu.com/ubuntu noble InRelease
Hit:6 http://fr.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:7 http://fr.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:8 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [800 kB]
Fetched 1,578 kB in 6s (283 kB/s)
Reading package lists... Done
root@myserver:/home/alirezaa98# sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
 docker-ce-rootless-extras libltdl7 libslirp0 pigz slirp4netns
Suggested packages:
 aufs-tools cgroups-mount | cgroup-lite
The following NEW packages will be installed:
 containerd.io docker-buildx-plugin docker-ce docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 75 not upgraded.
Need to get 124 MB of archives.
After this operation, 446 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://fr.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 https://download.docker.com/linux/ubuntu/noble/stable amd64 containerd.io amd64 1.7.24-1 [29.5 MB]
Get:3 http://fr.archive.ubuntu.com/ubuntu/noble/main amd64 libltdl7 amd64 2.4.7-7build1 [40.3 kB]
Get:4 http://fr.archive.ubuntu.com/ubuntu/noble/main amd64 libslirp0 amd64 4.7.0-1ubuntu3 [63.8 kB]
Get:5 http://fr.archive.ubuntu.com/ubuntu/noble/universe amd64 slirp4netns amd64 1.2.1-1build2 [34.9 kB]
9% [2 containerd.io 1,245 kB/29.5 MB 4%]
154 kB/s 13min 17s
```

و در نهایت با پول کردن Hello World image داکر را تست میکنیم.



```
Jan 13 21:24
root@myserver:/home/alirezaa98# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:5b3cc85e16e3058003c13b7821318369dad01dac3dbb877aac3c2818225c724
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 (amd64)
 3. The Docker daemon created a new container from that image which runs the
 executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
 to your terminal.

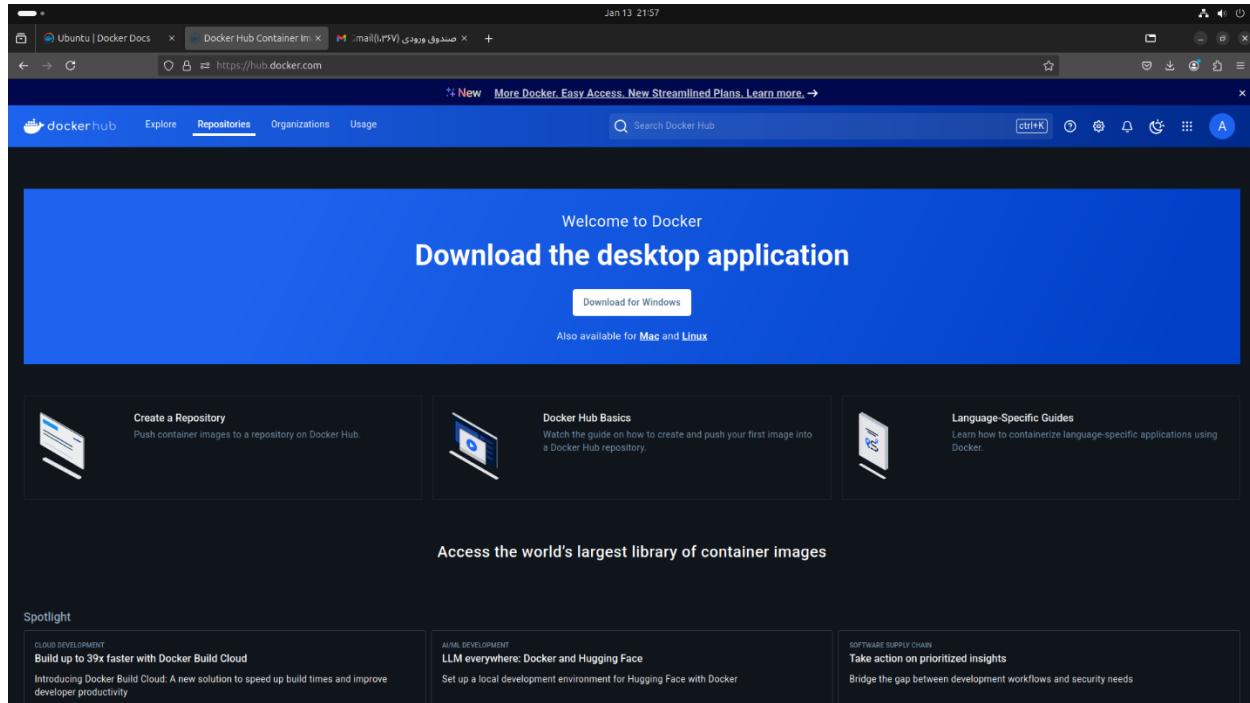
To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

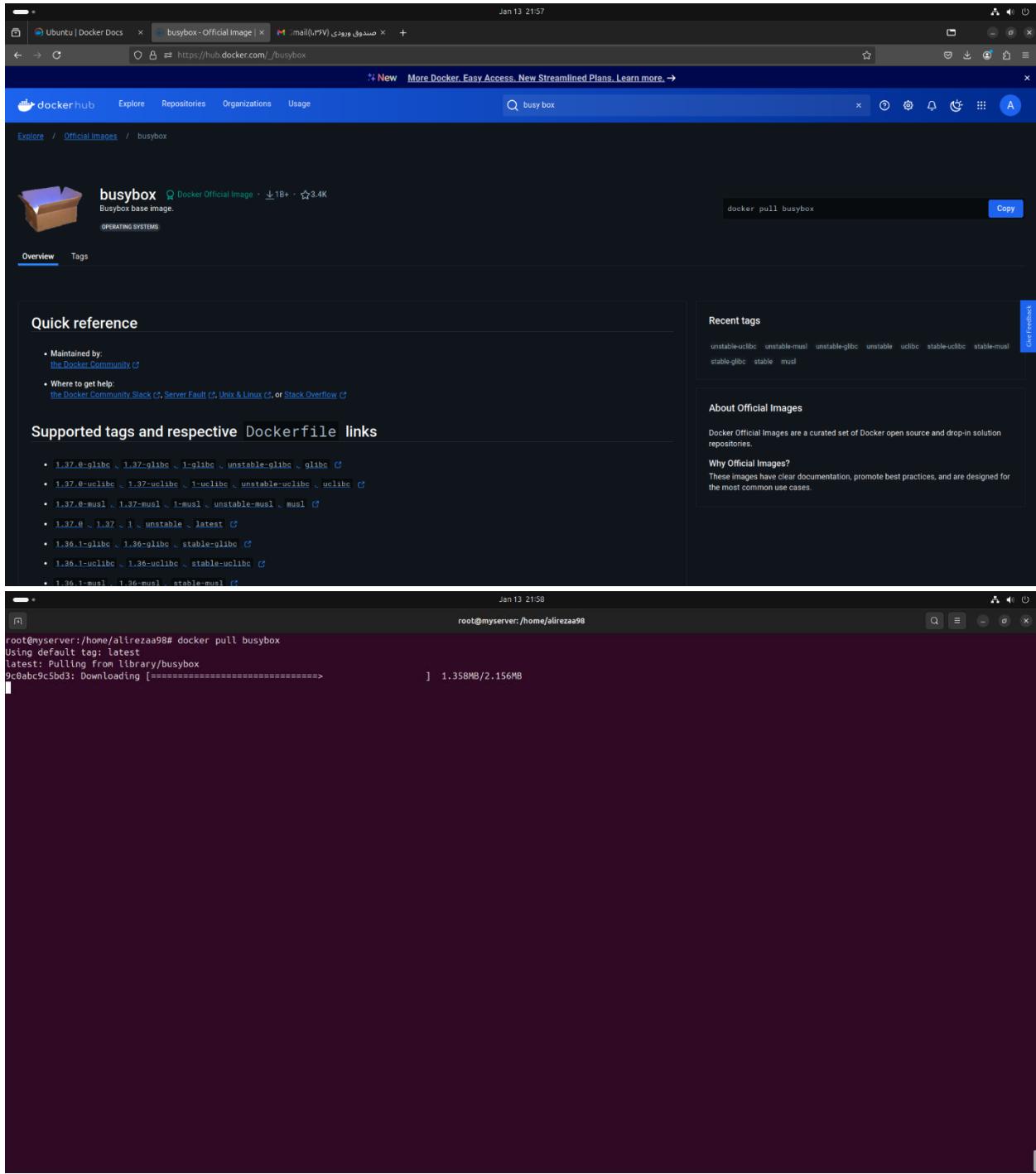
For more examples and ideas, visit:
https://docs.docker.com/get-started/
root@myserver:/home/alirezaa98#
```

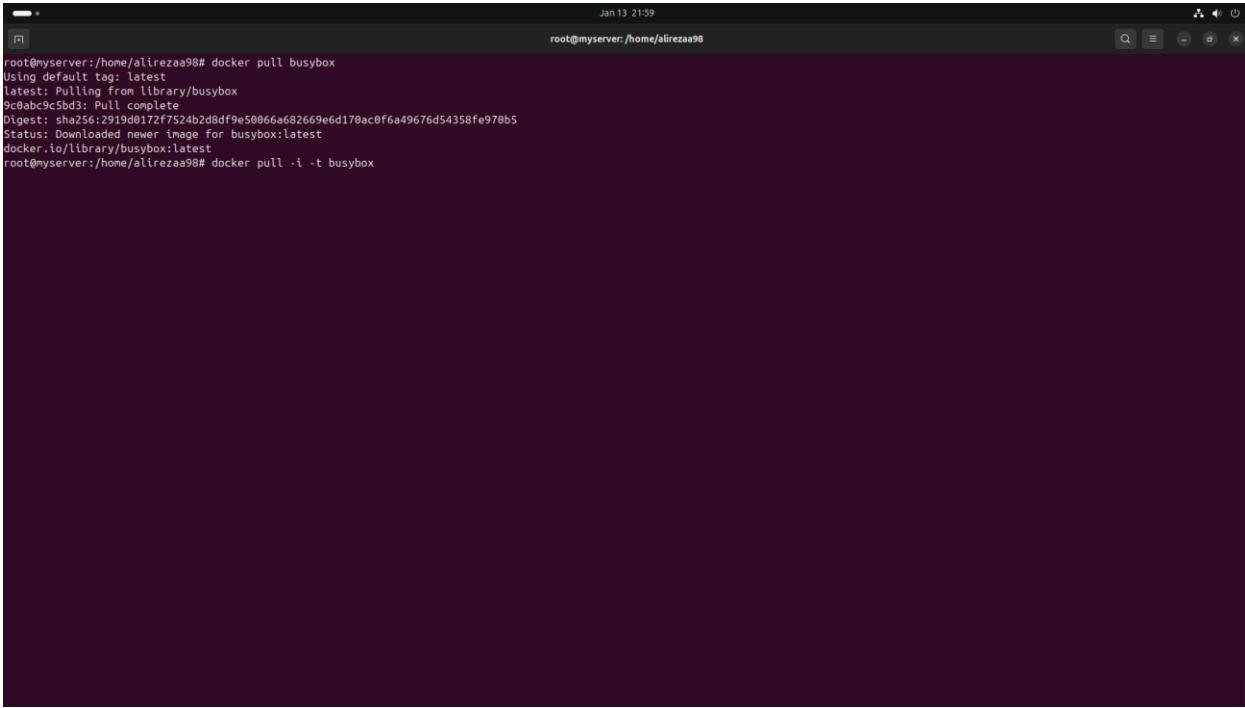
برای نصب ابتدا ریپازیتوری خود OS بررسی میشود که با پیغام عدم یافتن این بسته روبرو میشویم؛ و سپس از ریپازیتوری داکر استفاده میشود و از طریق آن این ایمیج بصورت کانتینرایز نصب شده و با پیغام Hello World روبرو میشویم؛ که نشان نصب موفقیت آمیز داکر و ریپازیتوری آن است.

بعد از نصب داکر ما به [hub.docker.com](https://hub.docker.com) لایگین کرده و از آنجا به تمام ایمیج های دسترسی خواهیم داشت.



سپس با جستجوی آن را یافته و با دستور سایت شروع به نصب آن میکنیم.





```
root@myserver:/home/alirezaa98# docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
9c8abc9c5bd3: Pull complete
Digest: sha256:2919d0172f7524bzd8df9e50066a682669e6d170ac0f6a49676d54358fe970b5
Status: Downloaded newer image for busybox:latest
docker.lo/library/busybox:latest
root@myserver:/home/alirezaa98# docker pull -i -t busybox
```

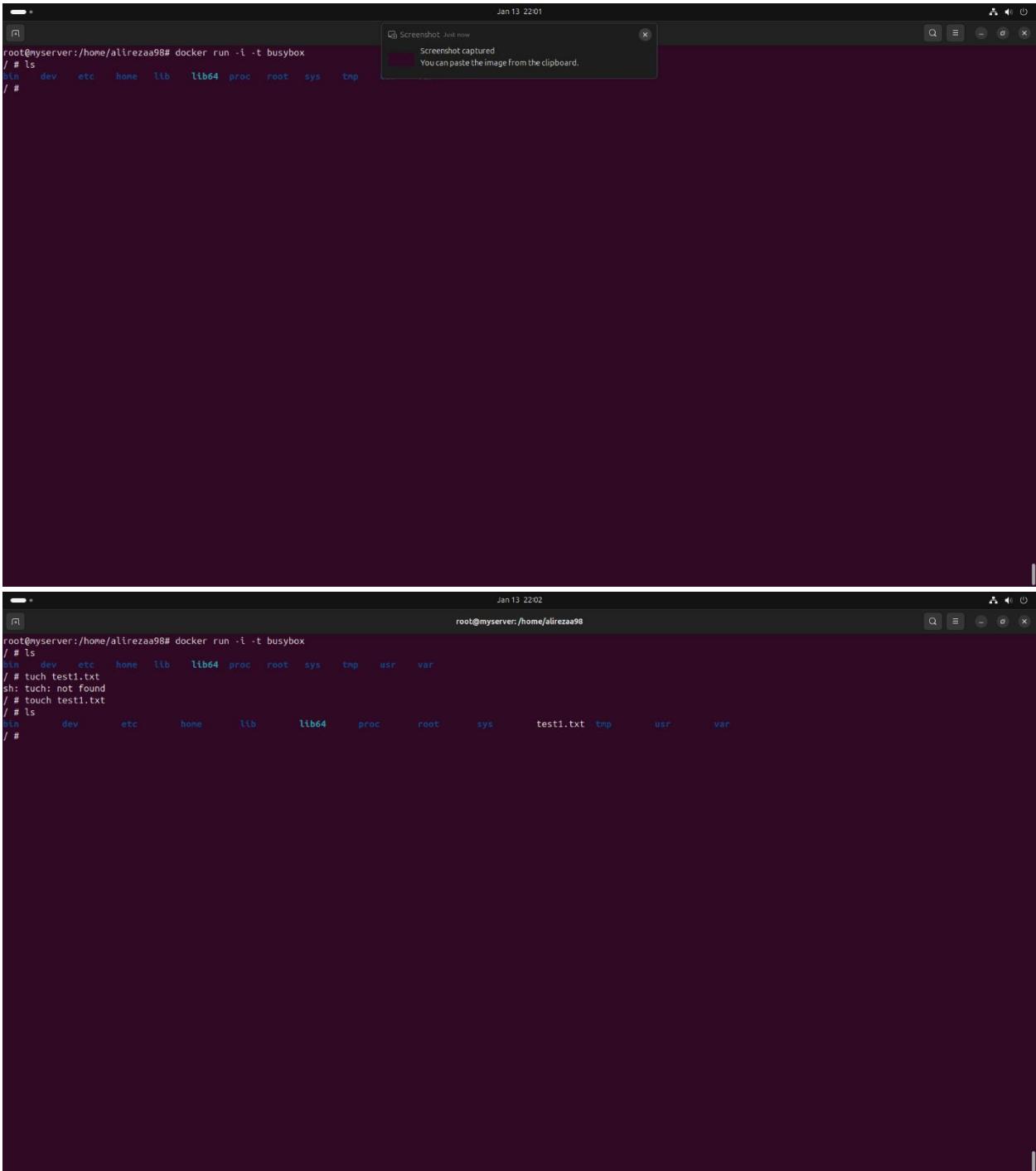
آپشن ها:

-: برای وارد شدن به ایمیج به صورت **interactive**  
-t: برای وارد کردن ورودی و خروجی  
با اضافه کردن این دو آپشن میتوان به صورت **interactive** به **busybox** ورودی و خروجی بدهیم.

The image consists of two vertically stacked screenshots of a terminal window. Both screenshots show a dark-themed terminal interface with a light-colored text area. The top screenshot shows the command `root@myserver:/home/alirezaa98# docker run -i -t busybox` being entered at the prompt. The bottom screenshot shows the command completed, and the user has typed `/ # ls` to list the contents of the root directory.

```
root@myserver:/home/alirezaa98# docker run -i -t busybox
root@myserver:/home/alirezaa98#
root@myserver:/home/alirezaa98# docker run -i -t busybox
root@myserver:/home/alirezaa98#
/ # ls
```

همانطور که در مقدمه گفتیم **busybox** یک ورژن ساده شده لینوکس است که میتوانیم در آن کامند های ساده ای مثل `ls`, `grep`, `cp` و `vi` بدهیم.

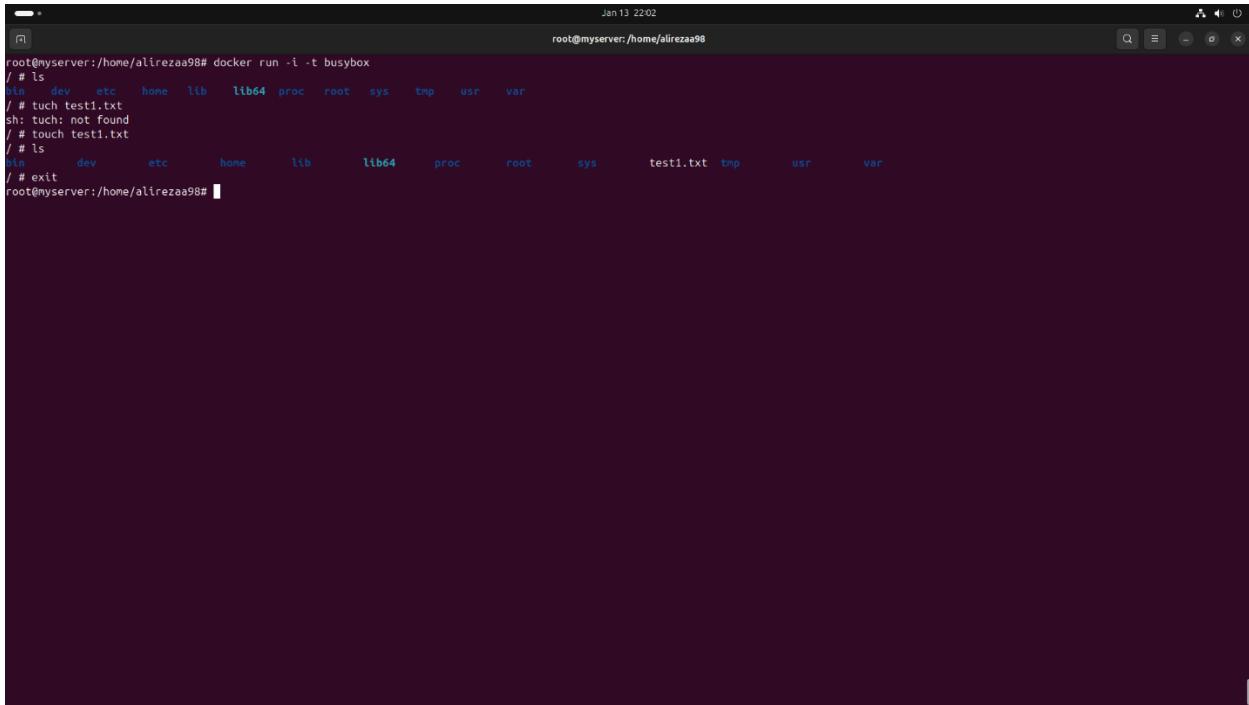


```
root@myserver:/home/alirezaa98# docker run -i -t busybox
/ # ls
bin dev etc home lib lib64 proc root sys tmp
/ #
root@myserver:/home/alirezaa98# docker run -i -t busybox
/ # ls
bin dev etc home lib lib64 proc root sys tmp usr var
/ # tuch test1.txt
sh: tuch: not found
/ # touch test1.txt
/ # ls
bin dev etc home lib lib64 proc root sys test1.txt tmp usr var
/ #
```

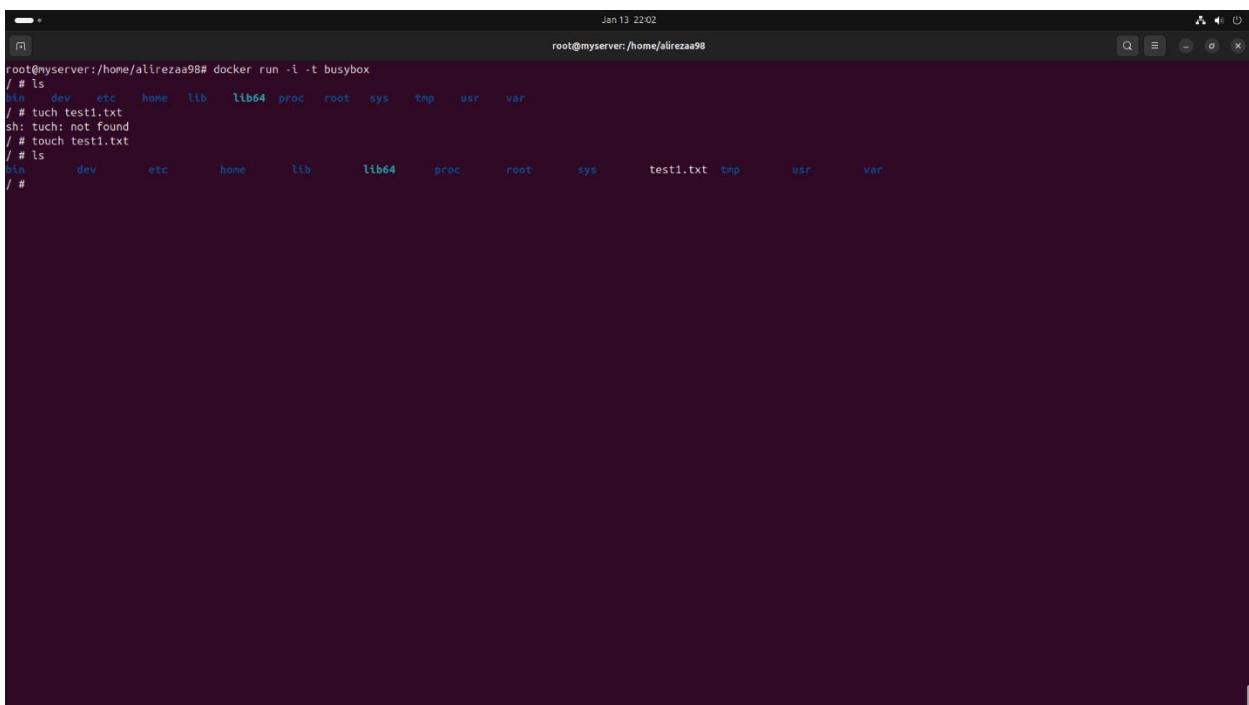
و اما یک نکته مهم:

تغییراتی که در **busybox** اعمال میشود موقتی هستند و بعد از هر بار خروج از ایمیج از بین میروند. برای اینکه این اتفاق صورت نگیرد ما باید با آپشن **-d**

برنامه را در background در اجرا نگهداشیم تا تغییرات اعمال شده حذف نشوند.



```
root@myserver:/home/alirezaa98# docker run -i -t busybox
/ # ls
bin dev etc home lib lib64 proc root sys tmp usr var
/ # touch test1.txt
sh: touch: not found
/ # touch test1.txt
/ # ls
bin dev etc home lib lib64 proc root sys test1.txt tmp usr var
/ # exit
root@myserver:/home/alirezaa98#
```



```
root@myserver:/home/alirezaa98# docker run -i -t busybox
/ # ls
bin dev etc home lib lib64 proc root sys tmp usr var
/ # touch test1.txt
sh: touch: not found
/ # touch test1.txt
/ # ls
bin dev etc home lib lib64 proc root sys test1.txt tmp usr var
/ #
```

```
root@myserver:/home/alirezaa98# docker run -i -t busybox
/ # ls
bin dev etc home lib lib64 proc root sys tmp usr var
/ # touch test1.txt
sh: touch: not found
/ # touch test1.txt
/ # ls
bin dev etc home lib lib64 proc root sys test1.txt tmp usr var
/ #
/ # exit
root@myserver:/home/alirezaa98# docker run -i -t busybox
/ # ls
bin dev etc home lib lib64 proc root sys tmp usr var
/ #
```

```
root@myserver:/home/alirezaa98# docker run -i -t busybox
/ # ls
bin dev etc home lib lib64 proc root sys tmp usr var
/ # touch test1.txt
sh: tuch: not found
/ # touch test1.txt
/ # ls
bin dev etc home lib lib64 proc root sys test1.txt tmp usr var
/ # exit
root@myserver:/home/alirezaa98# docker run -i -t busybox
/ # ls
bin dev etc home lib lib64 proc root sys tmp usr var
/ # exit
root@myserver:/home/alirezaa98# docker run -d busybox sleep 100
```

```
Jan 13 22:04
root@myserver:/home/alirezaa98# docker run -i -t busybox
/ # ls
bin dev etc home lib lib64 proc root sys tmp usr var
/ # touch test1.txt
sh: touch: not found
/ # touch test1.txt
/ # ls
bin dev etc home lib lib64 proc root sys test1.txt tmp usr var
/ # exit
root@myserver:/home/alirezaa98# docker run -i -t busybox
/ # ls
bin dev etc home lib lib64 proc root sys tmp usr var
/ # exit
root@myserver:/home/alirezaa98# docker run -d busybox sleep 100
7bc4ebbc8fda659f2ded02bd5d3604a359db7a02a340e9765e5b0bf488cd29
root@myserver:/home/alirezaa98# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
7bc4ebbc8fda busybox "sleep 100" 27 seconds ago Up 27 seconds blissful_ritchie
root@myserver:/home/alirezaa98#
```

و در نهایت همانطور که میبینید با دستور `docker ps` همانطور که میبینید با کامند `sleep 100(s)` ایمیج در حال اجرا است.

شروع پروژه اصلی:

## شبیه‌سازی شبکه با داکر و BusyBox

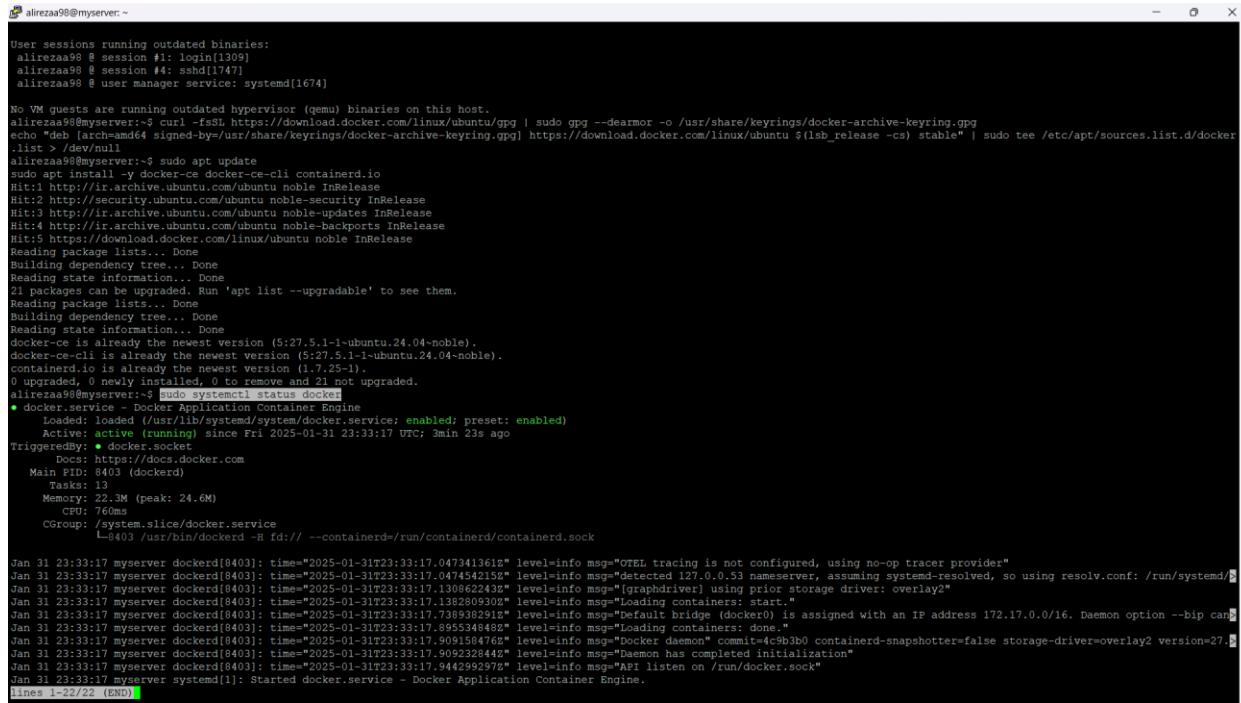
در دنیای فناوری، مجازی‌سازی و شبیه‌سازی شبکه نقش مهمی در توسعه، تست و استقرار سرویس‌های توزیع شده ایفا می‌کند. داکر (Docker) به عنوان یکی از محبوب‌ترین ابزارهای کانتینرسازی، امکان ایجاد محیط‌های ایزوله و مدیریت شبکه‌های مجازی را فراهم می‌کند.

در این پروژه، با استفاده از داکر و BusyBox، یک محیط شبکه‌ای مجازی شبیه‌سازی می‌کنیم. یک سیستم سبک وزن است که ابزارهای اساسی لینوکس را در یک بسته کوچک ارائه می‌دهد و برای تست و دیباگ شبکه در کانتینرها گزینه‌ای ایده‌آل محسوب می‌شود.

### اهداف این پروژه

- ◆ آشنایی با مدیریت شبکه در داکر
- ◆ ایجاد یک شبکه مجازی و اتصال کانتینرها به آن
- ◆ تست ارتباط بین کانتینرها با دستورات BusyBOX
- ◆ بررسی DNS داخلی داکر و باز بودن پورت‌ها
- ◆ شبیه‌سازی قطع و وصل ارتباط شبکه‌ای

## گام اول: بررسی وضعیت داکر:

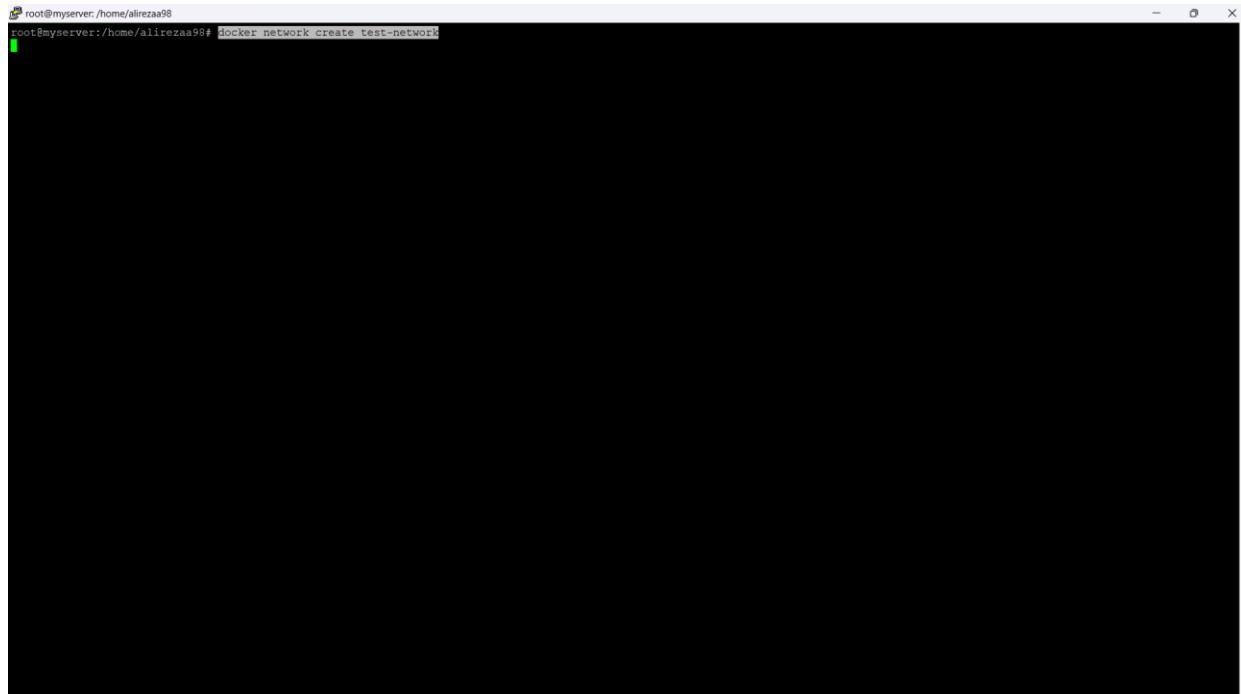


```
alirezaa98@myserver:~$ User sessions running outdated binaries:
alirezaa98 @ session #1: login[1309]
alirezaa98 @ session #4: sshd[1747]
alirezaa98 @ user manager service: systemd[1674]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
alirezaa98@myserver:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
alirezaa98@myserver:~$ sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io
Hit:1 http://ir.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:3 http://ir.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://ir.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
22 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker-ce is already the newest version (5:27.5.1-1~ubuntu.24.04-noble).
docker-ce-cli is already the newest version (5:27.5.1-1~ubuntu.24.04-noble).
containerd.io is already the newest version (1.7.25-1).
0 upgraded, 0 newly installed, 0 to remove and 21 not upgraded.
alirezaa98@myserver:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
 Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
 Active: active (running) since Fri 2025-01-31 23:33:17 UTC; 3min 23s ago
 Docs: https://docs.docker.com
Main PID: 8403 (dockerd)
 Tasks: 13
 Memory: 22.3M (peak: 24.6M)
 CPU: 760ms
 CGroup: /system.slice/docker.service
 └─8403 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

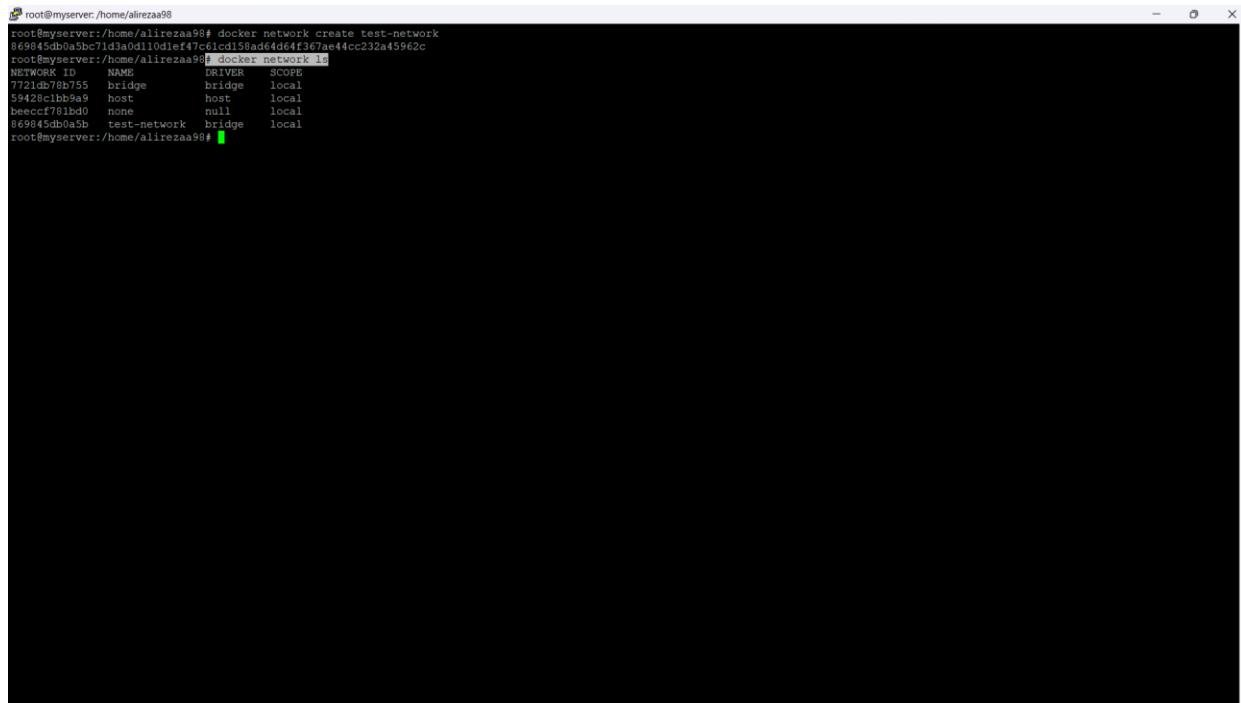
Jan 31 23:33:17 myserver dockerd[8403]: time="2025-01-31T23:33:17.047341361Z" level=info msg="OTEL tracing is not configured, using no-op tracer provider"
Jan 31 23:33:17 myserver dockerd[8403]: time="2025-01-31T23:33:17.047454252Z" level=info msg="detected 127.0.0.53 nameserver, assuming systemd-resolved, so using resolv.conf: /run/systemd/hostname"
Jan 31 23:33:17 myserver dockerd[8403]: time="2025-01-31T23:33:17.130063303Z" level=info msg="graphdriver[graphdriver] using prior storage driver: overlay2"
Jan 31 23:33:17 myserver dockerd[8403]: time="2025-01-31T23:33:17.130063303Z" level=info msg="Loading containers: start."
Jan 31 23:33:17 myserver dockerd[8403]: time="2025-01-31T23:33:17.130063303Z" level=info msg="No containers found. This node is assigned with an IP address 172.17.0.0/16. Daemon option --bip can be used to change this."
Jan 31 23:33:17 myserver dockerd[8403]: time="2025-01-31T23:33:17.895552482Z" level=info msg="Loading containers: done."
Jan 31 23:33:17 myserver dockerd[8403]: time="2025-01-31T23:33:17.909150476Z" level=info msg="Docker daemon" commit=4c9b3b0 containerd-snapshotter=false storage-driver=overlay2 version=27.5.1-1~ubuntu.24.04-noble
Jan 31 23:33:17 myserver dockerd[8403]: time="2025-01-31T23:33:17.909232844Z" level=info msg="Daemon has completed initialization"
Jan 31 23:33:17 myserver dockerd[8403]: time="2025-01-31T23:33:17.944299297Z" level=info msg="API listen on /run/docker.sock"
Jan 31 23:33:17 myserver systemd[1]: Started docker.service - Docker Application Container Engine.
lines 1-22/22 (ENDO)
```

## گام دوم: ایجاد یک شبکه مجازی در داکر



```
root@myserver:/home/alirezaa98# docker network create test-network
```

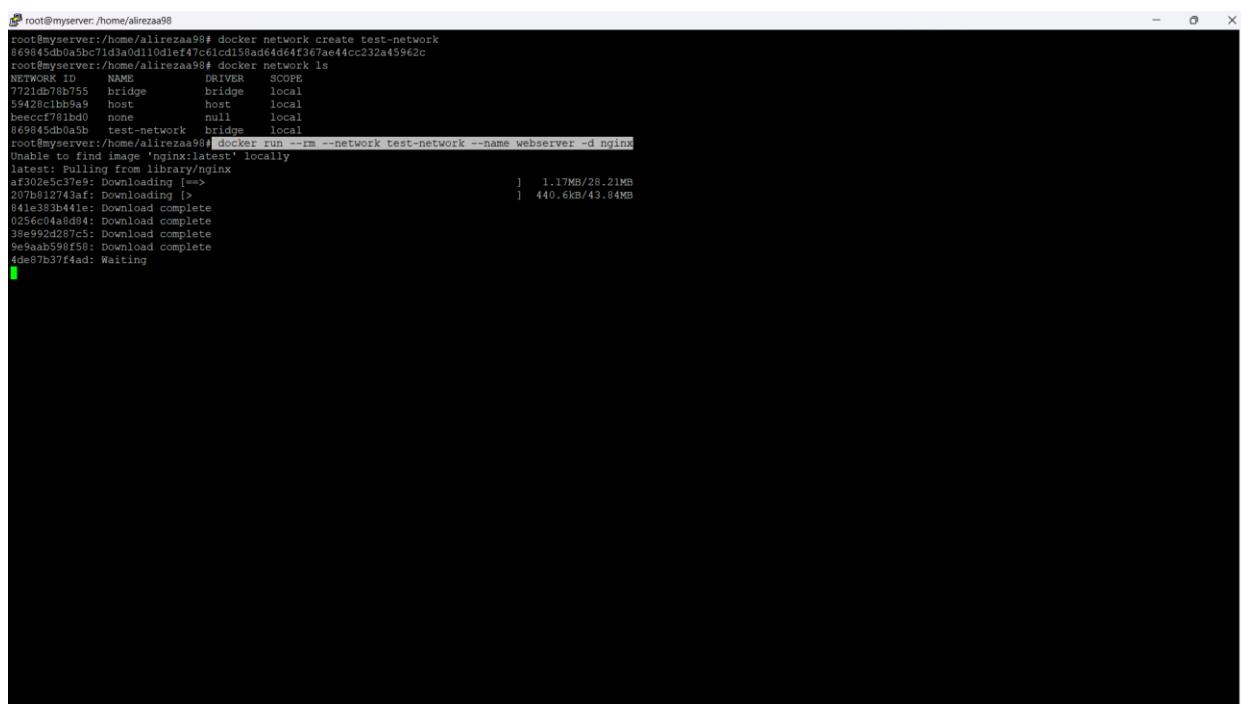
و بررسی لیست شبکه های داکر:



```
root@myserver:/home/alirezaa98# docker network create test-network
root@myserver:/home/alirezaa98# docker network ls
NETWORK ID NAME DRIVER SCOPE
7721db78b755 bridge bridge local
59428cbb9a9 host host local
beeccf781bd0 none null local
869845db0a5b test-network bridge local
root@myserver:/home/alirezaa98#
```

گام سوم: اجرای وبسروور در شبکه

یک کانتینر Nginx را در شبکه **test-network** اجرا می کنیم:



```
root@myserver:/home/alirezaa98# docker network create test-network
root@myserver:/home/alirezaa98# docker network ls
NETWORK ID NAME DRIVER SCOPE
7721db78b755 bridge bridge local
59428cbb9a9 host host local
beeccf781bd0 none null local
869845db0a5b test-network bridge local
root@myserver:/home/alirezaa98# docker run --rm --network test-network --name webserver -d nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
af302e5c17e9: Downloading [==>]
207b18132841: Downloading [>] 1.17MB/28.21MB
8413b441c1: Download complete
0256c04a8d84: Download complete
38a992d237c5: Download complete
4e99ab59bf58: Download complete
4de87b3774ad: Waiting
```

## توضیح دستورات:

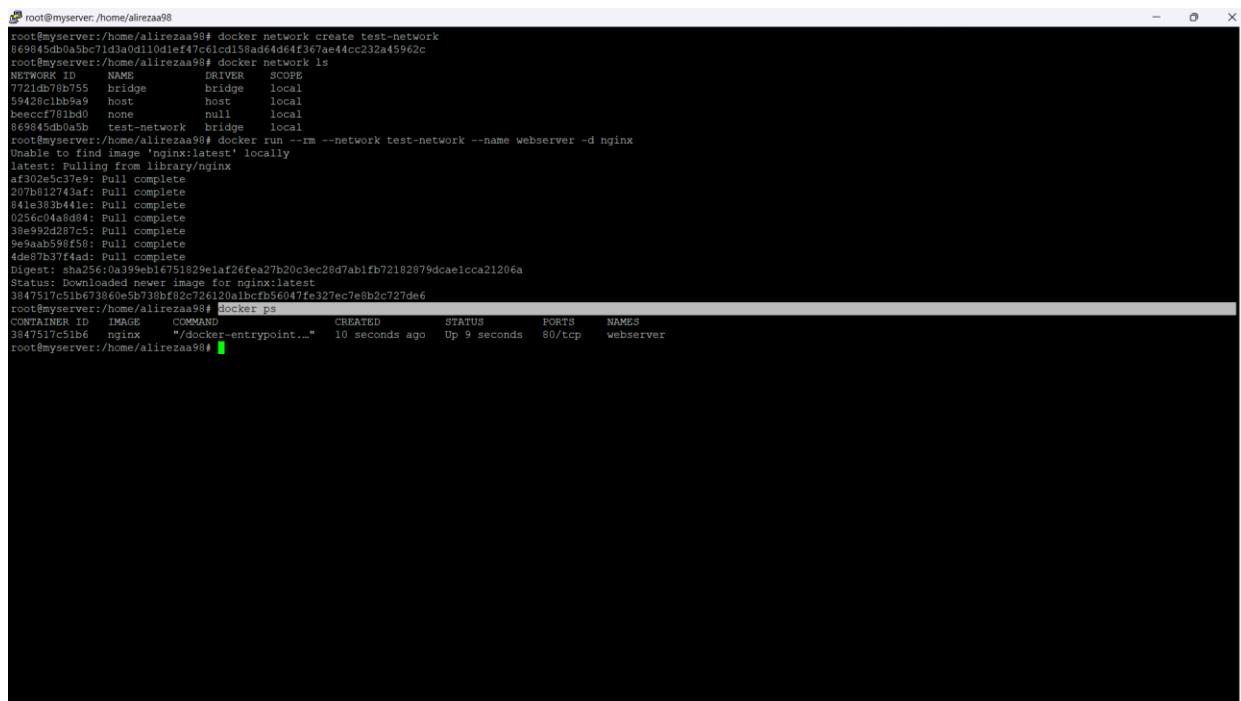
--rm → کانتینر بعد از توقف حذف شود.

--network test-network → متصل شدن به شبکه‌ای که ساختیم

--name webserver → تعیین نام وب سرور

.(background) → اجرای کانتینر در پس زمینه

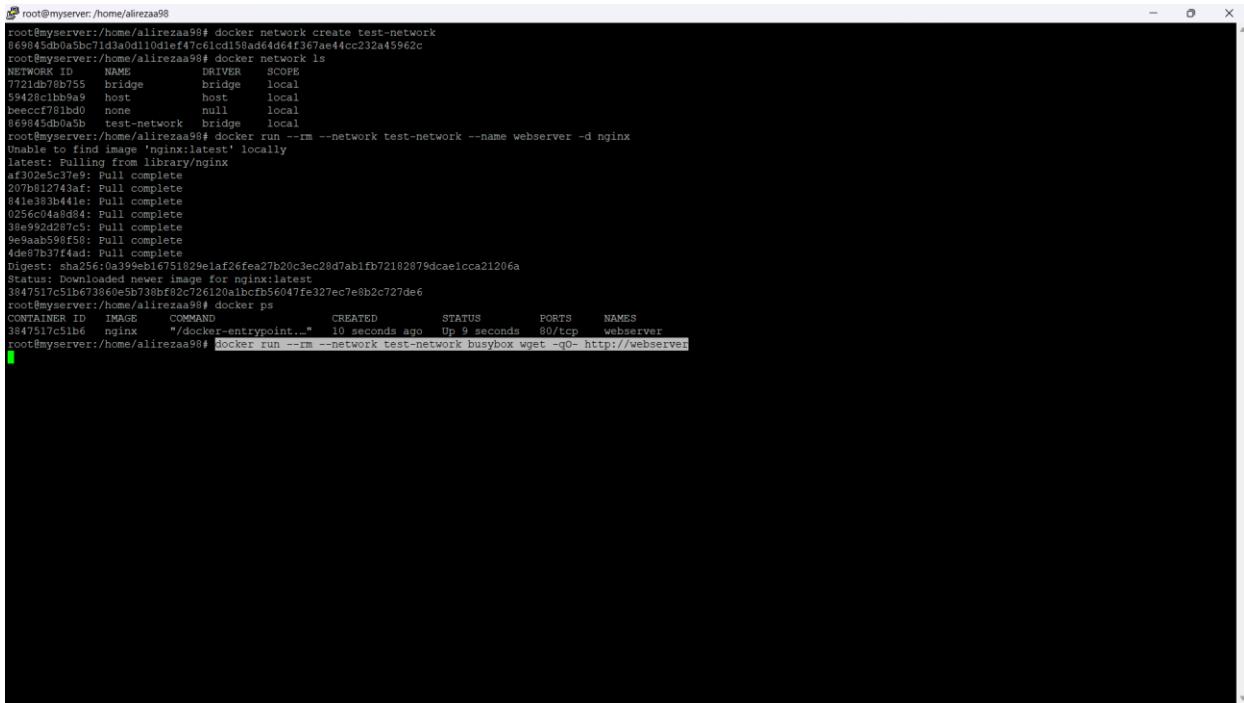
## بررسی وضعیت کانتینر:



```
root@myserver:/home/alirezaaa98# docker network create test-network
869845db0a5bc71d3a0d1010lef47c61cd158ad64d64f36ae44cc232a45962c
root@myserver:/home/alirezaaa98# docker network ls
NETWORK ID NAME DRIVER SCOPE
7721db78b755 bridge bridge local
59428cbb9a9 host host local
beeccff781bd0 none null local
869845db0a5b test-network bridge local
root@myserver:/home/alirezaaa98# docker run --rm --network test-network --name webserver -d nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
af302e5c57e9: Pull complete
207b812743af: Pull complete
841e383b441e: Pull complete
0256c04a0d84: Pull complete
38e992d287c5: Pull complete
9e9a9ab599f58: Pull complete
4495a1a1a1a1: Pull complete
Digest: sha256:0a399eb167518291a1f26fea27b20c3ec28d7ab1fb72182879dcae1cca21206a
Status: Downloaded newer image for nginx:latest
3847517c5b1673860e5b738bf82c726120a1bcfb56047fe327ec7e8b2c727de6
root@myserver:/home/alirezaaa98# Docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
3847517c5b16 nginx "/docker-entrypoint..." 10 seconds ago Up 9 seconds 80/tcp webserver
root@myserver:/home/alirezaaa98#
```

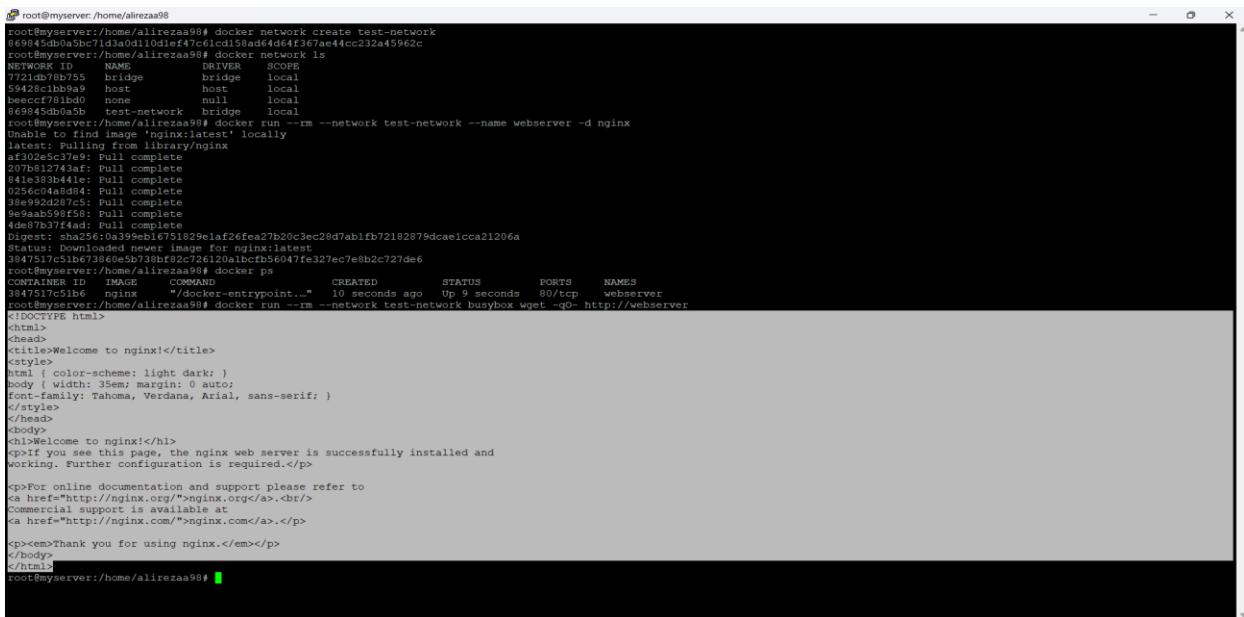
## گام چهارم: اجرای BusyBox و تست ارتباط

یک کانتینر BusyBox را اجرا کرده و بررسی می‌کنیم که آیا می‌تواند به webserver متصل شود:



```
root@myserver:/home/alirezaa98# docker network create test-network
069845dba05bc71d3a0d10dlef4f7c1cd158ad64d64f367ae44cc232a45962c
root@myserver:/home/alirezaa98# docker network ls
NETWORK ID NAME DRIVER SCOPE
7721db78b755 bridge bridge local
59428c1bb9a9 host host local
beeccf781bd0 none null local
069845dba05b test-network bridge local
root@myserver:/home/alirezaa98# docker run --rm --network test-network -d nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
af302e5c5c7e: Pull complete
207b12743af: Pull complete
841e383b441e: Pull complete
0256c04a4d94: Pull complete
38e992d287de: Pull complete
9e9aab59ff58: Pull complete
4de87b37f4ad: Pull complete
Digest: sha256:0a39eb16751829elaf26fea27b20c3ec28d7ab1fb72182879dcac1206a
Status: Downloaded newer image for nginx:latest
3847517c51b673860e5b738fb02c726120a1bcfb56047fe327ec7e8b2c727de6
root@myserver:/home/alirezaa98# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
3847517c51b6 nginx "/docker-entrypoint..." 10 seconds ago Up 9 seconds 80/tcp webserver
root@myserver:/home/alirezaa98# docker run --rm --network test-network busybox wget -qO- http://webserver
```

اگر ارتباط برقرار باشد، صفحه پیشفرض Nginx نمایش داده می‌شود:

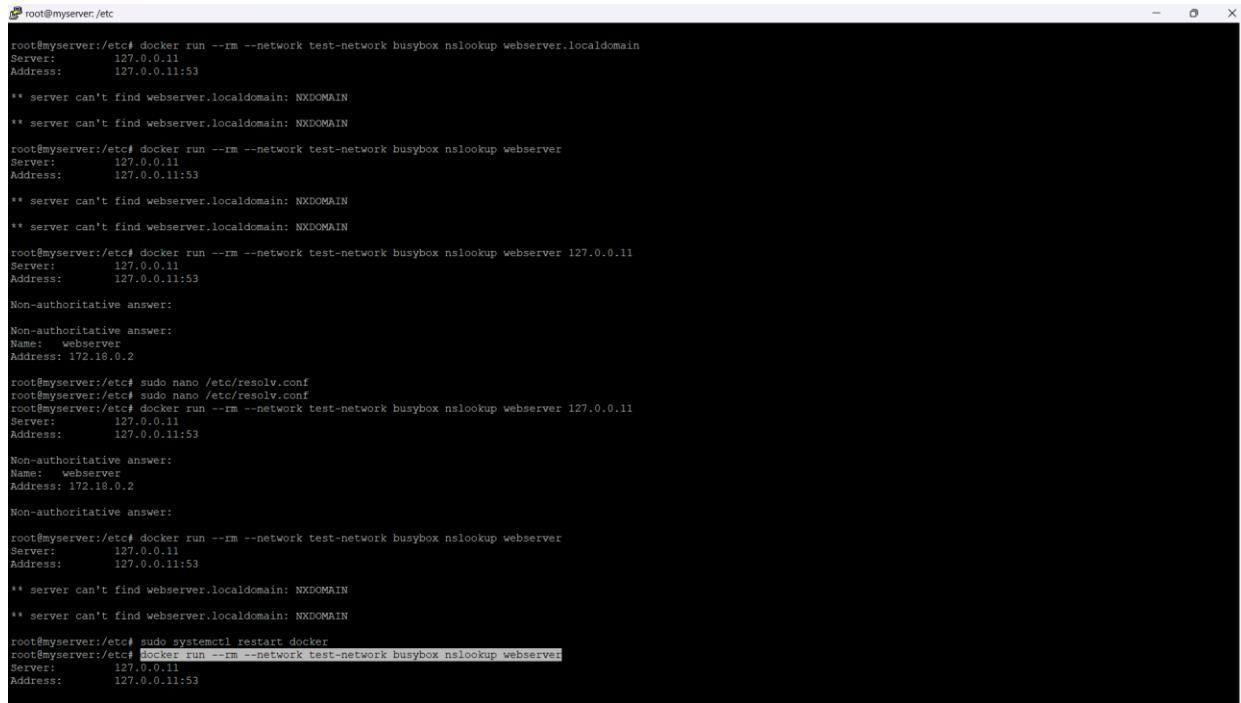


```
root@myserver:/home/alirezaa98# docker network create test-network
069845dba05bc71d3a0d10dlef4f7c1cd158ad64d64f367ae44cc232a45962c
root@myserver:/home/alirezaa98# docker network ls
NETWORK ID NAME DRIVER SCOPE
7721db78b755 bridge bridge local
59428c1bb9a9 host host local
beeccf781bd0 none null local
069845dba05b test-network bridge local
root@myserver:/home/alirezaa98# docker run --rm --network test-network -d nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
af302e5c5c7e: Pull complete
207b12743af: Pull complete
841e383b441e: Pull complete
0256c04a4d94: Pull complete
38e992d287de: Pull complete
9e9aab59ff58: Pull complete
4de87b37f4ad: Pull complete
Digest: sha256:0a39eb16751829elaf26fea27b20c3ec28d7ab1fb72182879dcac1206a
Status: Downloaded newer image for nginx:latest
3847517c51b673860e5b738fb02c726120a1bcfb56047fe327ec7e8b2c727de6
root@myserver:/home/alirezaa98# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
3847517c51b6 nginx "/docker-entrypoint..." 10 seconds ago Up 9 seconds 80/tcp webserver
root@myserver:/home/alirezaa98# docker run --rm --network test-network busybox wget -qO- http://webserver
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p>For online documentation and support please refer to
http://nginx.org/.

Commercial support is available at
http://nginx.com/.</p>
<p>Thank you for using nginx.</p>
</body>
</html>
root@myserver:/home/alirezaa98#
```

## گام پنجم: بررسی DNS داخلی داکر

تست می کنیم که سیستم DNS داخلی داکر به درستی کار می کند:



```
root@myserver:/etc# docker run --rm --network test-network busybox nslookup webserver.localdomain
Server: 127.0.0.11
Address: 127.0.0.11:53

** server can't find webserver.localdomain: NXDOMAIN
** server can't find webserver.localdomain: NXDOMAIN

root@myserver:/etc# docker run --rm --network test-network busybox nslookup webserver
Server: 127.0.0.11
Address: 127.0.0.11:53

** server can't find webserver.localdomain: NXDOMAIN
** server can't find webserver.localdomain: NXDOMAIN

root@myserver:/etc# docker run --rm --network test-network busybox nslookup webserver 127.0.0.11
Server: 127.0.0.11
Address: 127.0.0.11:53

Non-authoritative answer:

Non-authoritative answer:
Name: webserver
Address: 172.18.0.2

root@myserver:/etc# sudo nano /etc/resolv.conf
root@myserver:/etc# sudo nano /etc/resolv.conf
root@myserver:/etc# docker run --rm --network test-network busybox nslookup webserver 127.0.0.11
Server: 127.0.0.11
Address: 127.0.0.11:53

Non-authoritative answer:
Name: webserver
Address: 172.18.0.2

Non-authoritative answer:
root@myserver:/etc# docker run --rm --network test-network busybox nslookup webserver
Server: 127.0.0.11
Address: 127.0.0.11:53

** server can't find webserver.localdomain: NXDOMAIN
** server can't find webserver.localdomain: NXDOMAIN

root@myserver:/etc# sudo systemctl restart docker
root@myserver:/etc# docker run --rm --network test-network busybox nslookup webserver
Server: 127.0.0.11
Address: 127.0.0.11:53
```

سرور و دسکتاپ بنده تا این مرحله اجرا شد ولی با این خطا مواجه شد:

server can't find webserver.localdomain: NXDOMAIN

که بخاطر مشکل در nslookup dns به وجود امد و من نتوانستم حلش کنم اما ادامه پروژه را

بصورت تئوری توضیح خواهم داد:

## گام ششم : تست باز بودن پورت‌ها

برای بررسی باز بودن پورت 80 روی webserver استفاده می کنیم:

**docker run --rm --network test-network busybox telnet webserver 80**

اگر پورت باز باشد، اتصال برقرار می شود.

## گام هفتم شبیه‌سازی قطع ارتباط و اتصال مجدد

قطع ارتباط **webserver** از شبکه:

```
docker network disconnect test-network webserver
```

تلash برای اتصال مجدد از **BusyBox** (باید شکست بخورد):

```
docker run --rm --network test-network busybox wget -qO- http://webserver
```

اتصال مجدد **webserver** به شبکه:

```
docker network connect test-network webserver
```

حالا دوباره ارتباط باید برقرار باشد.

## گام هشتم: راهاندازی سرور **HTTP** با **BusyBox**

می‌توانیم از **BusyBox** به عنوان یک سرور **HTTP** موقت استفاده کنیم:

اجرای سرور **HTTP** روی پورت **8080**:

```
docker run --rm -v $(pwd):/www --network test-network busybox httpd -f -p 8080
```

این سرور فایل‌های موجود در پوشه جاری را به اشتراک می‌گذارد.

دانلود فایل از کانتینر دیگر:

```
docker run --rm --network test-network busybox wget -qO-
http://<IP>:8080/yourfile.txt
```

جای **<IP>** را با آدرس IP سرور **BusyBox** جایگزین کنید.

## گام نهم: تست اتصال اینترنت از داخل کانتینر

بررسی می‌کنیم که آیا کانتینرهای ما به اینترنت متصل هستند:

تست با **ping**

```
docker run --rm busybox ping -c 4 google.com
```

تست با **wget**:

```
docker run --rm busybox wget -qO- https://example.com
```

اگر محتوا دریافت شد، اینترنت داخل کانتینر فعال است.

## جمع‌بندی پروژه

در این پروژه یاد گرفتیم:

چگونه یک شبکه اختصاصی در داکر ایجاد کنیم.

چگونه یک وب‌서ور **Nginx** اجرا کنیم.

چگونه با **BusyBox** تست‌های شبکه‌ای انجام دهیم.

بررسی **DNS** داخلی داکر و تست باز بودن پورت‌ها.

شبیه‌سازی قطع و وصل ارتباط کانتینرها به شبکه.

استفاده از **HTTP** به عنوان سرور.

تست اتصال اینترنت از داخل کانتینر.