Experiment-1: Setup of Ubuntu Virtual Machine using Vagrant and VMware Fusion and Deployment of Nginx & Docker

Date @January 21, 2026

Experiment No. - 1

Course Name- Containerisation and DevOps

Student Name- Ananya Karn

Sap_ID - 500125205

Roll No. - R2142231061

Semester - 6th

Instructor - Prateek Gautam

Aim/ Objective

</> The aim of this experiment is to provision an Ubuntu 22.04 virtual machine using Vagrant with VMware Fusion on Apple Silicon architecture, install and configure the Nginx web server, and deploy Docker Engine to validate containerized application execution.

Software and hardware requirements:

Component	Details
Host Operating System	macOS (Apple Silicon – ARM64)
Virtualization Platform	VMware Fusion
VM Automation Tool	Vagrant
Guest OS	Ubuntu 22.04 LTS (ARM64)
Web Server	Nginx
Container Platform	Docker Engine
Code Editor	Visual Studio Code
System Architecture	ARM64

Theory/BackGround

Virtualization

Virtualization is a technology that allows multiple operating systems to run on a single physical machine by abstracting the underlying hardware resources. Instead of dedicating one physical system to one operating system, virtualization enables efficient utilization of CPU, memory, and storage by creating isolated virtual environments known as virtual machines. Each virtual machine behaves like a real computer with its own operating system and applications, while sharing the same physical hardware.

Vagrant

Vagrant is an open-source tool used for automating the creation and management of virtual machines. It provides a simple and reproducible workflow for setting up development environments using configuration files known as *Vagrantfiles*. With Vagrant, virtual machines can be created, configured, started, stopped, and destroyed using simple command-line instructions. This eliminates manual setup and ensures consistency across different systems, making it highly useful in DevOps and cloud-based environments.

VMware Fusion

VMware Fusion is a virtualization platform used on macOS to run virtual machines efficiently. On Apple Silicon (ARM64) architecture, VMware Fusion is preferred because it provides native support for ARM-based processors. Other virtualization tools such as VirtualBox have limited or unstable support on Apple Silicon, whereas VMware Fusion is optimized for performance and compatibility. Therefore, VMware Fusion is an ideal provider for running Ubuntu virtual machines on macOS systems with Apple Silicon.

Nginx

Nginx is a high-performance web server widely used for serving static and dynamic web content. It is known for its lightweight architecture, high concurrency handling, and low memory usage. Nginx is commonly used as a web server, reverse proxy, and load balancer in modern web applications. In this experiment, Nginx is installed inside the Ubuntu virtual machine to demonstrate the deployment and management of a web service.

Docker

Docker is an open-source containerization platform that allows applications to be packaged along with their dependencies into lightweight containers. Containers ensure that applications run consistently across different environments by isolating them from the underlying system. Docker is widely used in DevOps practices because it simplifies application deployment, improves scalability, and reduces configuration issues. In this experiment, Docker Engine is installed inside the Ubuntu virtual machine to validate container execution using a sample container.

Virtual Machines vs Containers

The main difference between virtual machines and containers lies in their architecture and resource usage. Virtual machines include a full operating system along with the application, making them heavier and slower to start. Containers, on the other hand, share the host operating system kernel and only include the application and its dependencies, making them lightweight and faster. While virtual machines provide strong isolation at the hardware level, containers offer efficient and scalable application deployment. Both technologies are essential in modern infrastructure, and this experiment demonstrates their combined usage.

System Architecture/Setup Description

In this experiment, macOS acts as the host operating system. VMware Fusion is used as the virtualization provider, while Vagrant automates the creation and management of the Ubuntu 22.04 ARM virtual machine. Inside the virtual machine, Nginx is deployed as a web server and Docker Engine is installed to run containerized applications.

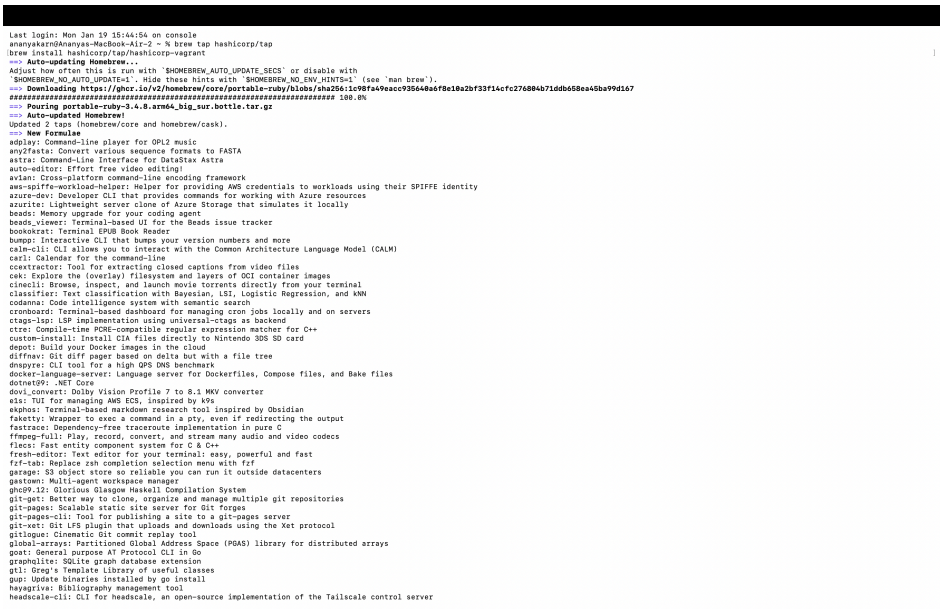
Procedure-

Step 1 : Installation and verification of Vagrant:

Vagrant is used to automate the creation and management of virtual machines.

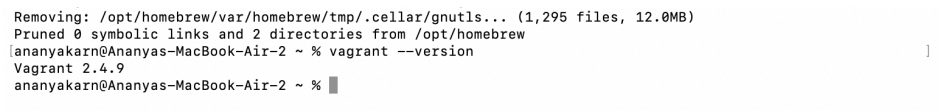
To install Vagrant, run these command in the local terminal of the host system:

```
brew tap hashicorp/tap
brew install hashicorp/tap/hashicorp-vagrant
```



First, the installation of Vagrant is verified on the host system using the terminal.

```
vagrant --version
```



Explanation:

This command checks whether Vagrant is correctly installed on the macOS system and displays the installed version.

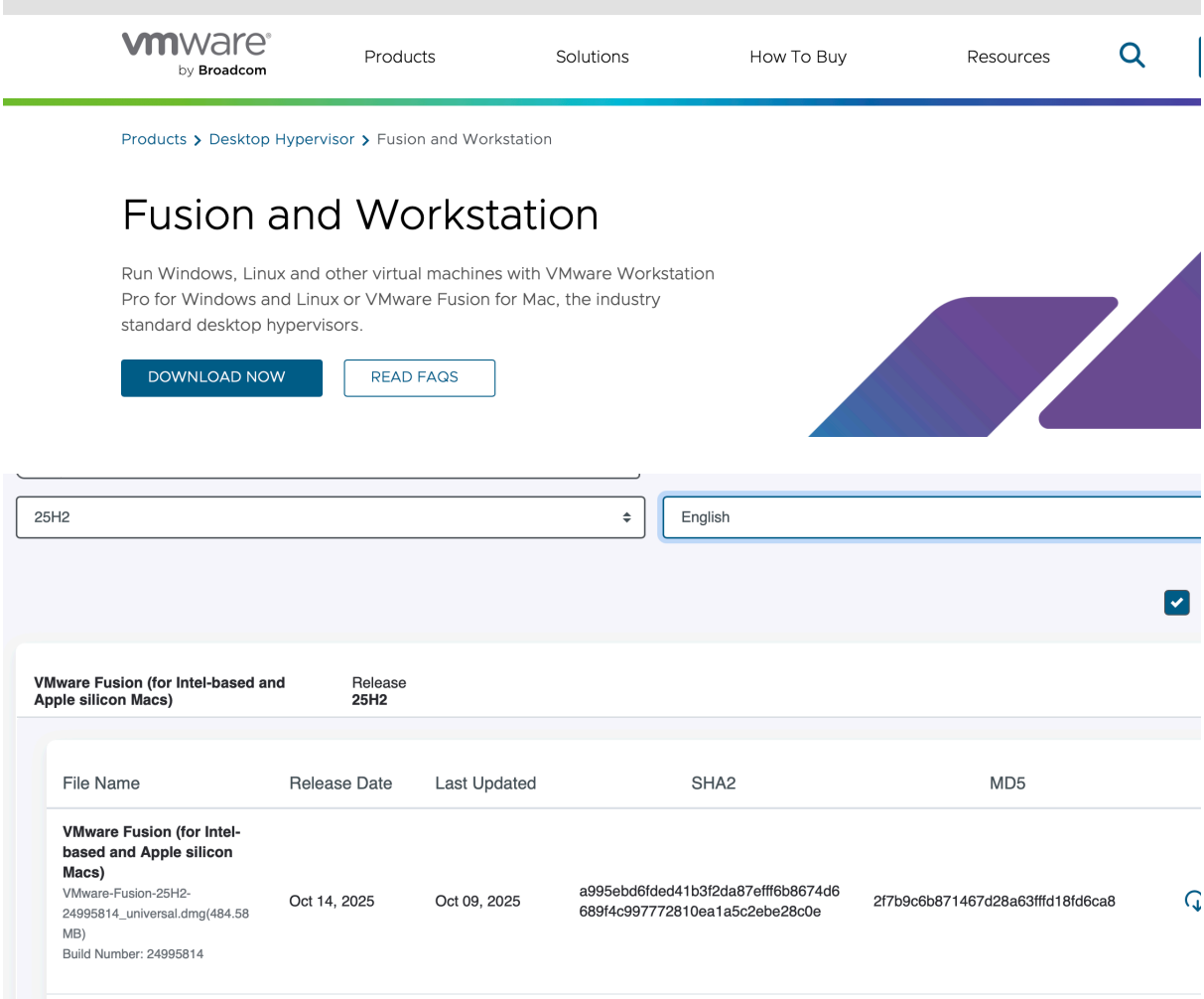
Step 2: Installation and Setup of VMware Fusion

VMware Fusion is used as the virtualization provider for Apple Silicon (ARM64) architecture.

1. VMware Fusion is downloaded from the official VMware website.
2. The application is installed by dragging it into the Applications folder.
3. VMware Fusion is opened once to allow system permissions.
4. Required permissions such as system extensions and network access are granted.

Explanation:

VMware Fusion provides native ARM support on Apple Silicon, making it suitable for running Ubuntu virtual machines efficiently.



Step 3: Creation of Project Directory

A dedicated directory is created to store Vagrant configuration files.

```
mkdir ubuntu-vagrant
cd ubuntu-vagrant
```

```
nanyakarn@Ananyas-MacBook-Air-2 desktop % mkdir ubuntu-vagrant
nanyakarn@Ananyas-MacBook-Air-2 desktop % cd ubuntu-vagrant
```

Explanation:

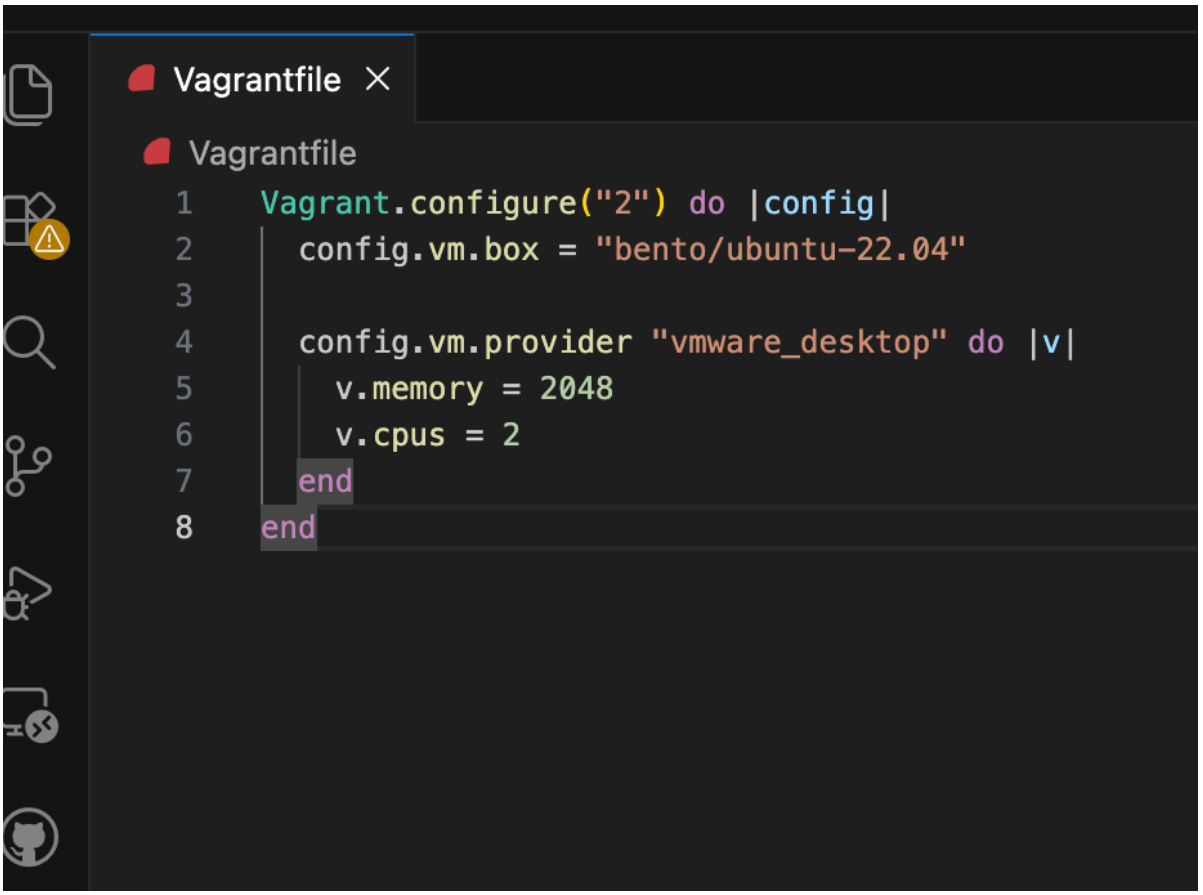
This directory contains the Vagrantfile, which defines the virtual machine configuration.

Step 4: Configuration of Vagrantfile

The virtual machine configuration is defined using a Vagrantfile.

```
Vagrant.configure("2") do |config|
  config.vm.box = "bento/ubuntu-22.04"

  config.vm.provider "vmware_desktop" do |v|
    v.memory = 2048
    v.cpus = 2
  end
end
```



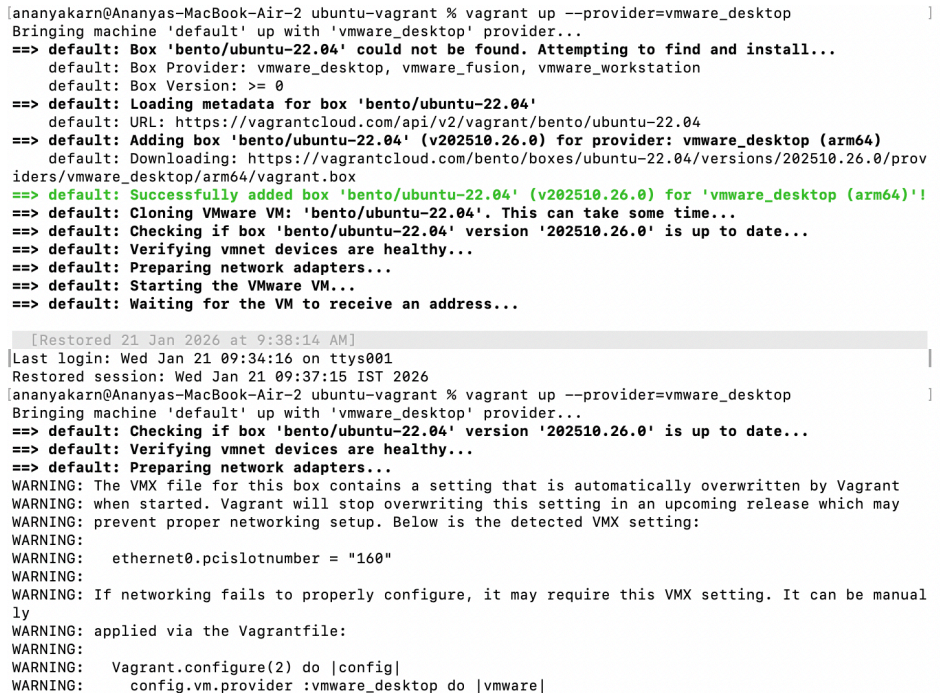
Explanation:

This configuration specifies the Ubuntu 22.04 box compatible with ARM architecture and assigns CPU and memory resources using VMware Fusion as the provider.

Step 5: Starting the Virtual Machine

The Ubuntu virtual machine is started using the following command:

```
vagrant up --provider=vmware_desktop
```



Explanation:

This command downloads the required Ubuntu box and boots the virtual machine using VMware Fusion.

Step 6: Accessing the Virtual Machine

Secure shell access to the virtual machine is established using:

```
vagrant ssh
```

```
ananyakarn@Ananyas-MacBook-Air-2 ubuntu-vagrant % vagrant ssh
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-160-generic aarch64)

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

 System information as of Wed Jan 21 04:13:18 AM UTC 2026

  System load:  0.01               Processes:             247
  Usage of /:   19.3% of 29.82GB   Users logged in:       0
  Memory usage: 11%                IPv4 address for eth0: 192.168.95.128
  Swap usage:   0%


This system is built by the Bento project by Chef Software
More information can be found at https://github.com/chef/bento

Use of this system is acceptance of the OS vendor EULA and License Agreements.
Last login: Wed Jan 21 04:11:18 2026 from 192.168.95.1
vagrant@vagrant:~$ exit
logout
ananyakarn@Ananyas-MacBook-Air-2 ubuntu-vagrant % vagrant up
Bringing machine 'default' up with 'vmware_desktop' provider...
==> default: Checking if box 'bento/ubuntu-22.04' version '202510.26.0' is up to date...
==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked to run always will still run.
ananyakarn@Ananyas-MacBook-Air-2 ubuntu-vagrant % vagrant ssh
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-160-generic aarch64)

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

 System information as of Wed Jan 21 04:14:29 AM UTC 2026
```

Explanation:

This command allows the user to interact with the Ubuntu virtual machine through the terminal.

Step 7: Installation of Nginx Web Server

The Nginx web server is installed inside the Ubuntu virtual machine.

```
sudo apt update
sudo apt install -y nginx
```

```
vagrant@vagrant:~$ sudo apt update
Hit:1 http://ports.ubuntu.com/ubuntu-ports jammy InRelease
Get:2 http://ports.ubuntu.com/ubuntu-ports jammy-updates InRelease [128 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports jammy-backports InRelease [127 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports jammy-security InRelease [129 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 Packages [2,993 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main Translation-en [485 kB]
Get:7 http://ports.ubuntu.com/ubuntu-ports jammy-updates/restricted arm64 Packages [4,875 kB]
Get:8 http://ports.ubuntu.com/ubuntu-ports jammy-updates/restricted Translation-en [944 kB]
Get:9 http://ports.ubuntu.com/ubuntu-ports jammy-updates/universe arm64 Packages [1,274 kB]
Get:10 http://ports.ubuntu.com/ubuntu-ports jammy-updates/universe Translation-en [311 kB]
Get:11 http://ports.ubuntu.com/ubuntu-ports jammy-updates/multiverse arm64 Packages [38.6 kB]
Get:12 http://ports.ubuntu.com/ubuntu-ports jammy-backports/main arm64 Packages [69.0 kB]
Get:13 http://ports.ubuntu.com/ubuntu-ports jammy-backports/universe arm64 Packages [30.1 kB]
Get:14 http://ports.ubuntu.com/ubuntu-ports jammy-backports/universe Translation-en [16.9 kB]
Get:15 http://ports.ubuntu.com/ubuntu-ports jammy-security/main arm64 Packages [2,743 kB]
Get:16 http://ports.ubuntu.com/ubuntu-ports jammy-security/main Translation-en [418 kB]
Get:17 http://ports.ubuntu.com/ubuntu-ports jammy-security/restricted arm64 Packages [4,723 kB]
Get:18 http://ports.ubuntu.com/ubuntu-ports jammy-security/restricted Translation-en [917 kB]
Get:19 http://ports.ubuntu.com/ubuntu-ports jammy-security/universe arm64 Packages [1,043 kB]
Get:20 http://ports.ubuntu.com/ubuntu-ports jammy-security/universe Translation-en [222 kB]
Fetched 21.5 MB in 56s (382 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
54 packages can be upgraded. Run 'apt list --upgradable' to see them.
vagrant@vagrant:~$ Connection to 127.0.0.1 closed by remote host.
ananyakarn@Ananyas-MacBook-Air-2 ubuntu-vagrant % vagrant ssh
ananyakarn@Ananyas-MacBook-Air-2 ubuntu-vagrant % vagrant up
Bringing machine 'default' up with 'vmware_desktop' provider...
==> default: Checking if box 'bento/ubuntu-22.04' version '202510.26.0' is up to date...
==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked to run always will still run.
ananyakarn@Ananyas-MacBook-Air-2 ubuntu-vagrant % vagrant ssh
ananyakarn@Ananyas-MacBook-Air-2 ubuntu-vagrant % vagrant up --provider=vmware_desktop
Bringing machine 'default' up with 'vmware_desktop' provider...
==> default: Checking if box 'bento/ubuntu-22.04' version '202510.26.0' is up to date...
==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> default: flag to force provisioning. Provisioners marked to run always will still run.
ananyakarn@Ananyas-MacBook-Air-2 ubuntu-vagrant % vagrant ssh
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-160-generic aarch64)

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

```
vagrant@vagrant:/vagrant$ sudo apt install -y nginx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  fontconfig-config fonts-dejavu-core libdeflate0 libfontconfig1 libgd3 libjbig0 libjpeg-turbo8
  libjpeg8 libnginx-mod-http-geoip2 libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter
  libnginx-mod-mail libnginx-mod-stream libnginx-mod-stream-geoip2 libtiff5 libwebp7 libx11-6
  libx11-data libxau6 libxcb1 libxdmcp6 libxpm4 nginx-common nginx-core
Suggested packages:
  libgd-tools fcgiwrap nginx-doc ssl-cert
The following NEW packages will be installed:
  fontconfig-config fonts-dejavu-core libdeflate0 libfontconfig1 libgd3 libjbig0 libjpeg-turbo8
  libjpeg8 libnginx-mod-http-geoip2 libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter
  libnginx-mod-mail libnginx-mod-stream libnginx-mod-stream-geoip2 libtiff5 libwebp7 libx11-6
  libx11-data libxau6 libxcb1 libxdmcp6 libxpm4 nginx nginx-common nginx-core
0 upgraded, 25 newly installed, 0 to remove and 54 not upgraded.
Need to get 3,526 kB of archives.
After this operation, 11.0 MB of additional disk space will be used.
Get:1 http://ports.ubuntu.com/ubuntu-ports jammy/main arm64 libxau6 arm64 1:1.0.9-1build5 [7,624 B]
Get:2 http://ports.ubuntu.com/ubuntu-ports jammy/main arm64 libxdmcp6 arm64 1:1.1.3-0ubuntu5 [10.8 kB
]
Get:3 http://ports.ubuntu.com/ubuntu-ports jammy/main arm64 libxcb1 arm64 1.14-3ubuntu3 [49.0 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 libx11-data all 2:1.7.5-1ubuntu0.
3 [120 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 libx11-6 arm64 2:1.7.5-1ubuntu0.3
 [661 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports jammy/main arm64 fonts-dejavu-core all 2.37-2build1 [1,041
 kB]
Get:7 http://ports.ubuntu.com/ubuntu-ports jammy/main arm64 fontconfig-config all 2.13.1-4.2ubuntu5 [
29.1 kB]
Get:8 http://ports.ubuntu.com/ubuntu-ports jammy/main arm64 libdeflate0 arm64 1.10-2 [69.1 kB]
Get:9 http://ports.ubuntu.com/ubuntu-ports jammy/main arm64 libfontconfig1 arm64 2.13.1-4.2ubuntu5 [1
35 kB]
Get:10 http://ports.ubuntu.com/ubuntu-ports jammy/main arm64 libjpeg-turbo8 arm64 2.1.2-0ubuntu1 [129
 kB]
Get:11 http://ports.ubuntu.com/ubuntu-ports jammy/main arm64 libjpeg8 arm64 8c-2ubuntu10 [2,264 B]
Get:12 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 libjbig0 arm64 2.1-3.1ubuntu0.22
.04.1 [29.1 kB]
Get:13 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 libwebp7 arm64 1.2.2-2ubuntu0.22
.04.2 [193 kB]
Get:14 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 libtiff5 arm64 4.3.0-6ubuntu0.12
 [181 kB]
Get:15 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 libxpm4 arm64 1:3.5.12-1ubuntu0.
22.04.2 [35.5 kB]
Get:16 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 libgd3 arm64 2.3.0-2ubuntu2.3 [1
27 kB]
```

The Nginx service is started and enabled:

```
sudo systemctl start nginx
sudo systemctl enable nginx
```

The status of the service is checked using:

```
systemctl status nginx
```

```
vagrant@vagrant:/vagrant$ sudo systemctl start nginx
sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable nginx
vagrant@vagrant:/vagrant$ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2026-01-21 06:33:50 UTC; 18s ago
       Docs: man:nginx(8)
   Main PID: 3642 (nginx)
      Tasks: 3 (limit: 2198)
     Memory: 3.8M
        CPU: 19ms
     CGroup: /system.slice/nginx.service
             ├─3642 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             ├─3644 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "">
             └─3645 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "">

Jan 21 06:33:50 vagrant systemd[1]: Starting A high performance web server and a reverse proxy serve>
Jan 21 06:33:50 vagrant systemd[1]: Started A high performance web server and a reverse proxy server.
...skipping...
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2026-01-21 06:33:50 UTC; 18s ago
       Docs: man:nginx(8)
   Main PID: 3642 (nginx)
      Tasks: 3 (limit: 2198)
     Memory: 3.8M
        CPU: 19ms
     CGroup: /system.slice/nginx.service
             ├─3642 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             ├─3644 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "">
             └─3645 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "">

Jan 21 06:33:50 vagrant systemd[1]: Starting A high performance web server and a reverse proxy serve>
Jan 21 06:33:50 vagrant systemd[1]: Started A high performance web server and a reverse proxy server.
~
~
~
~
```

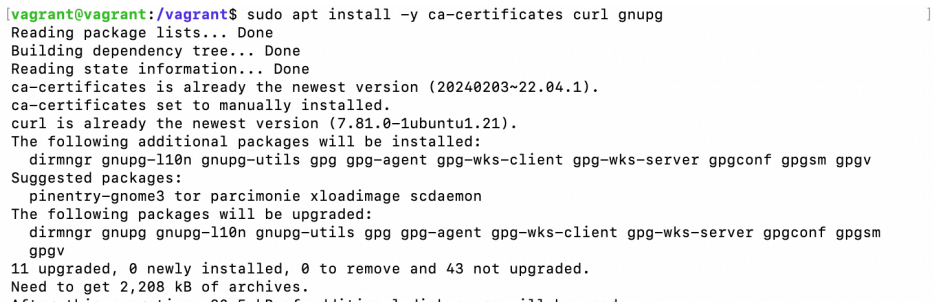
Explanation:

These commands install and configure Nginx to run as a web server inside the virtual machine.

Step 8: Installation of Docker Engine

Docker Engine is installed inside the Ubuntu virtual machine to enable containerization.

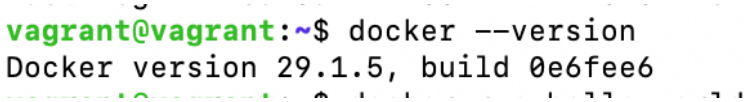
```
sudo apt update
sudo apt install -y ca-certificates curl gnupg
sudo apt install -y docker.io
```



Docker repository is added and Docker Engine is installed.

After installation, the Docker service is verified by running:

```
docker --version
```



Step 9: Verification of Docker Installation

A test container is executed to verify successful installation of Docker.

```
docker run hello-world
```

```
vagrant@vagrant:~$ docker run hello-world

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.
    (arm64v8)
 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 i
    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/
```

Explanation:

This command downloads and runs a test Docker image, confirming that Docker is installed and functioning correctly.

This procedure successfully completes:

- Virtual machine provisioning using Vagrant
- Virtualization using VMware Fusion
- Deployment of Nginx web server
- Installation and verification of Docker Engine

> Result:

> Ubuntu virtual machine was successfully created using Vagrant and VMware Fusion. Nginx web server and Docker Engine were installed and verified successfully.

Observations

- ARM64 architecture requires compatible virtualization support
- Vagrant simplifies VM provisioning
- Docker runs inside the virtual machine
- Nginx service runs successfully on Ubuntu

Conclusion

The experiment successfully demonstrated the creation of a DevOps-ready environment using Vagrant and VMware Fusion. The installation of Nginx and Docker validated service deployment and container execution within the virtual machine.
