Application of Docker and Apache for Web-Hosting using the Microsoft Azure Platform

BY

Soham Hait

Soupayan Ghosh

TITLE : Application of Docker and Apache for Web-Hosting using the Microsoft Azure Platform.

- **Project Category: Cloud Computing with Azure.**

- **Uses:** After learning how to use **Docker** and **Apache** for web hosting on the **Microsoft Azure** platform, we will be able to:

1. **Containerize Applications**: Use Docker to package your web applications and their dependencies into containers, ensuring consistent performance across different environments.
2. **Automate Deployment:** Automate the deployment of your web applications using Docker images and Azure services, reducing manual setup and configuration.
3. **Scalability**: Easily scale your web applications horizontally by deploying multiple container instances across Azure's infrastructure.
4. **Continuous Integration and Continuous Deployment (CI/CD):** Implement CI/CD pipelines to streamline the process of building, testing, and deploying applications using Azure DevOps or other CI/CD tools.
5. **Enhanced Security**: Improve security by isolating applications within containers and utilizing Azure's security features to protect your web hosting environment.
6. **Efficient Resource Management**: Optimize resource usage by running multiple Docker containers on a single VM to manage container orchestration.
7. **Load Balancing**: Implement load balancing using Azure Load Balancer or Azure Application Gateway to distribute incoming traffic across multiple container instances.
8. **Backup and Disaster Recovery**: Set up automated backups and disaster recovery strategies to ensure the availability and integrity of your web applications.
9. **Monitoring and Logging:** Utilize Azure Monitor and other logging tools to track the performance and health of your containerized applications.
10. **Hybrid Deployments**: Integrate on-premises infrastructure with Azure to create hybrid deployments, allowing seamless interaction between local and cloud-hosted resources.

- **Tools and Software Required**:

1. Command prompt
2. Linux OS(Ubuntu)
3. Docker
4. GitHub
5. Git
6. Apache2
7. MongoDB

- **Platform: Microsoft Azure Services**

❖ Goals Of Implementation:

The goal of this project is to leverage Docker and Apache on the Microsoft Azure platform to create a robust, scalable, and efficient web hosting environment. By containerizing applications with Docker, deploying them on Azure, and utilizing Apache as the web server, the project aims to streamline deployment processes, enhance application performance, ensure high availability, and provide seamless scalability. This approach will enable continuous integration and continuous deployment (CI/CD), optimize resource management, and bolster security, ultimately delivering a resilient and cost-effective web hosting solution.

❖ Steps Taken:

- **Create a Linux virtual machine in the Azure portal:**

Azure virtual machines (VMs) can be created through the Azure portal. The Azure portal is a browser-based user interface to create Azure resources. The following steps shows us how to use the Azure portal to deploy a Linux virtual machine (VM) running Ubuntu Server 22.04 LTS.

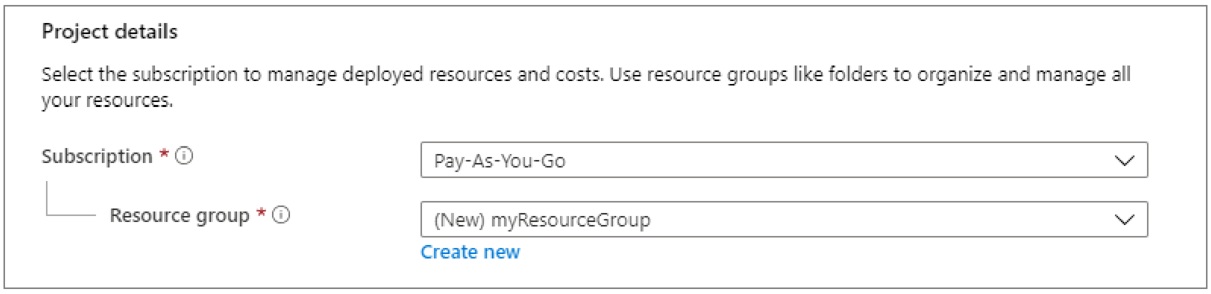
If you don't have an Azure subscription, create a free account before you begin.

- **Sign in to Azure:**

Sign in to the Azure portal.

- **Creating a virtual machine:**

1. Enter *virtual machines* in the search.
2. Under **Services**, select **Virtual machines**.
3. In the **Virtual machines** page, select **Create** and then **Virtual machine**. The **Create a virtual machine** page opens.
4. In the **Basics** tab, under **Project details**, make sure the correct subscription is selected and then choose to **Create new** resource group. Enter *myResourceGroup* for the name.



Project details

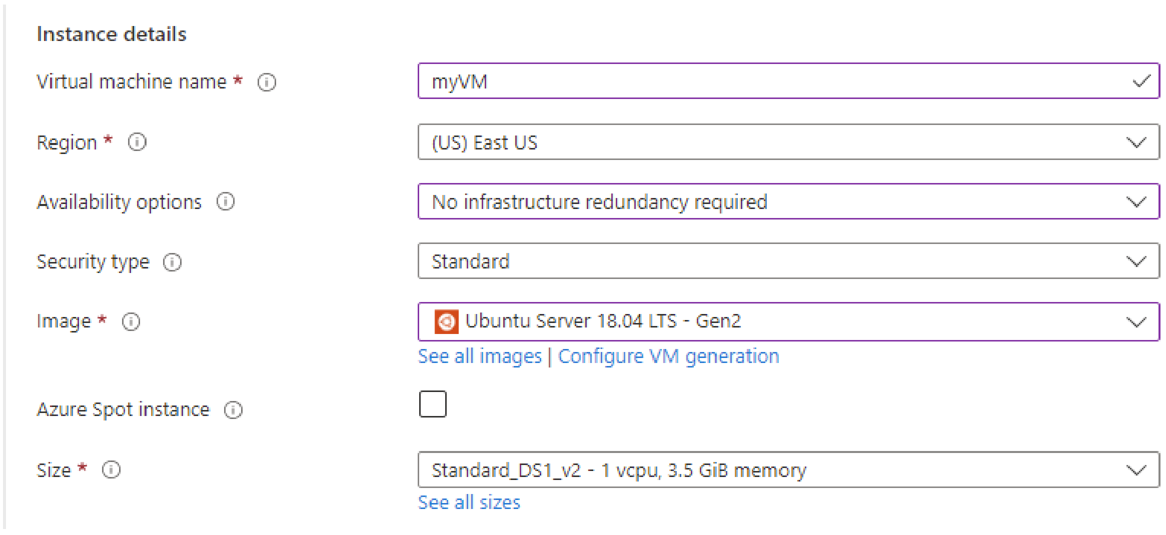
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ: Pay-As-You-Go

Resource group * ⓘ: (New) myResourceGroup

Create new

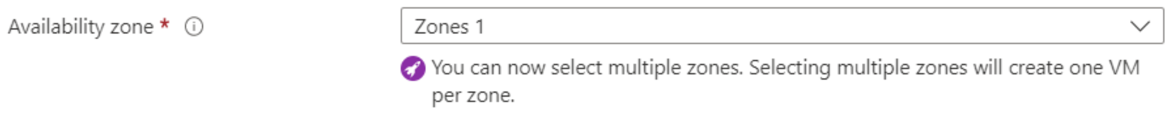
5. Under **Instance details**, enter *JVM* for the **Virtual machine name**, and choose *Ubuntu Server 22.04 LTS - Gen2* for your **Image**. Leave the other defaults. The default size and pricing are only shown as an example. Size availability and pricing are dependent on your region and subscription.



Instance details	
Virtual machine name *	myVM
Region *	(US) East US
Availability options	No infrastructure redundancy required
Security type	Standard
Image *	Ubuntu Server 18.04 LTS - Gen2
Azure Spot instance	☐
Size *	Standard_DS1_v2 - 1 vcpu, 3.5 GiB memory

Note:

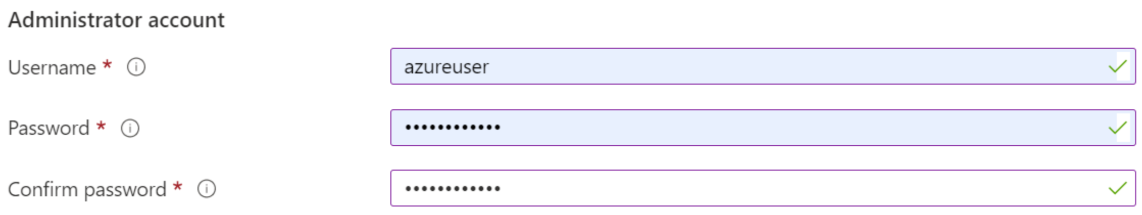
Some users will now see the option to create VMs in multiple zones.



Availability zone * : Zones 1

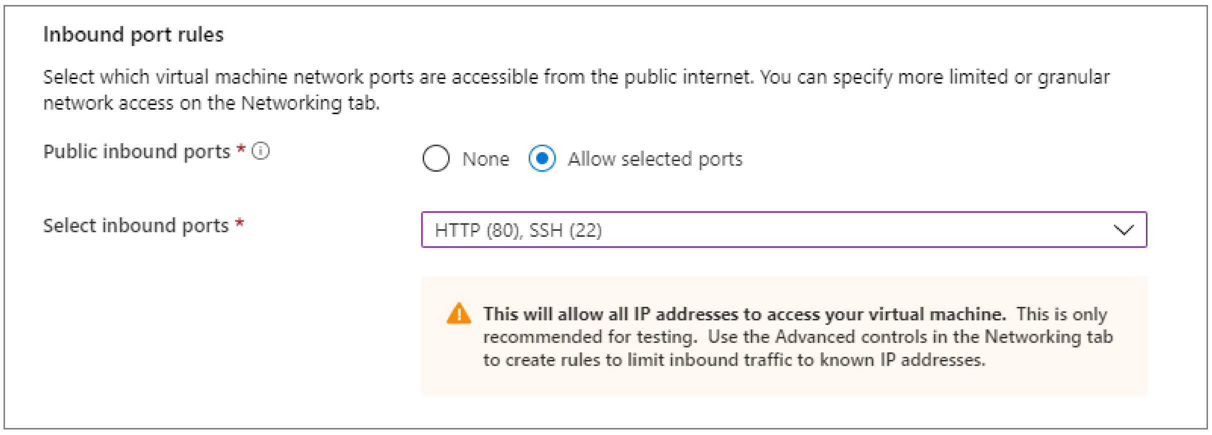
You can now select multiple zones. Selecting multiple zones will create one VM per zone.

6. Under **Administrator account**, provide a username, such as *azureuser* and a password. The password must be at least 12 characters long and meet the defined complexity requirements.



Administrator account	
Username *	azureuser
Password *	••••••••••••
Confirm password *	••••••••••••

7. Under **Inbound port rules** > **Public inbound ports**, choose **Allow selected ports** and then select **SSH (22)** and **HTTP (80)** from the drop-down.



Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports *	None [✓ Allow selected ports]
Select inbound ports *	HTTP (80), SSH (22)

This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

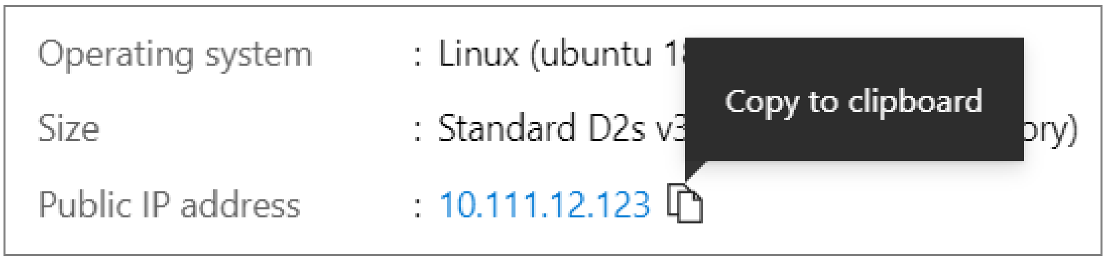
8. Leave the remaining defaults and then select the **Review + create** button at the bottom of the page.

9. On the **Create a virtual machine** page, you can see the details about the VM you are about to create. When you are ready, select **Create**.

10.

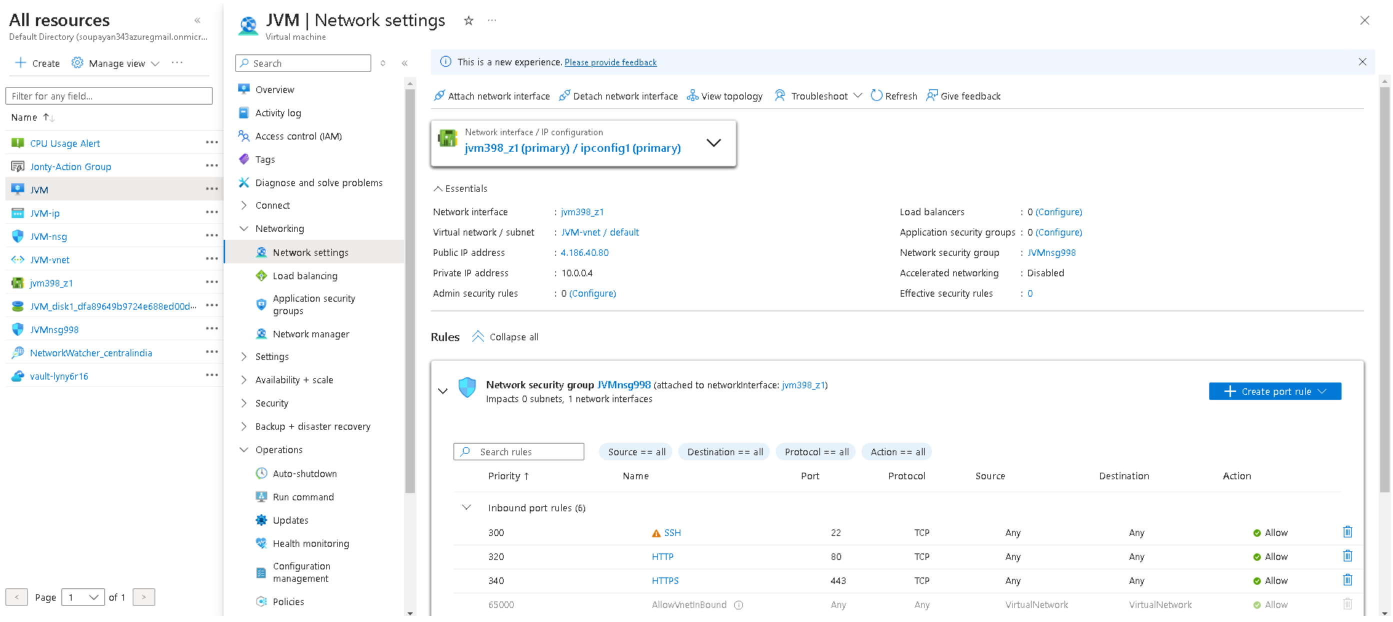
11. When the deployment is finished, select **Go to resource**.

12. On the page for your new VM, select the public IP address and copy it to your clipboard.



Operating system	:	Linux (ubuntu 1
Size	:	Standard D2s v3 ory)
Public IP address	:	10.111.12.123

13. In the VM menu, under the "**Settings**" section, click on "**Networking**". This will display the network interface settings for the selected VM.



Network interface / IP configuration
jvm398_z1 (primary) / ipconfig1 (primary)

Essentials

Network interface	: jvm398_z1	Load balancers	: 0 (Configure)
Virtual network / subnet	: JVM-vnet / default	Application security groups	: 0 (Configure)
Public IP address	: 4.186.40.80	Network security group	: JVMnsg998
Private IP address	: 10.0.0.4	Accelerated networking	: Disabled
Admin security rules	: 0 (Configure)	Effective security rules	: 0

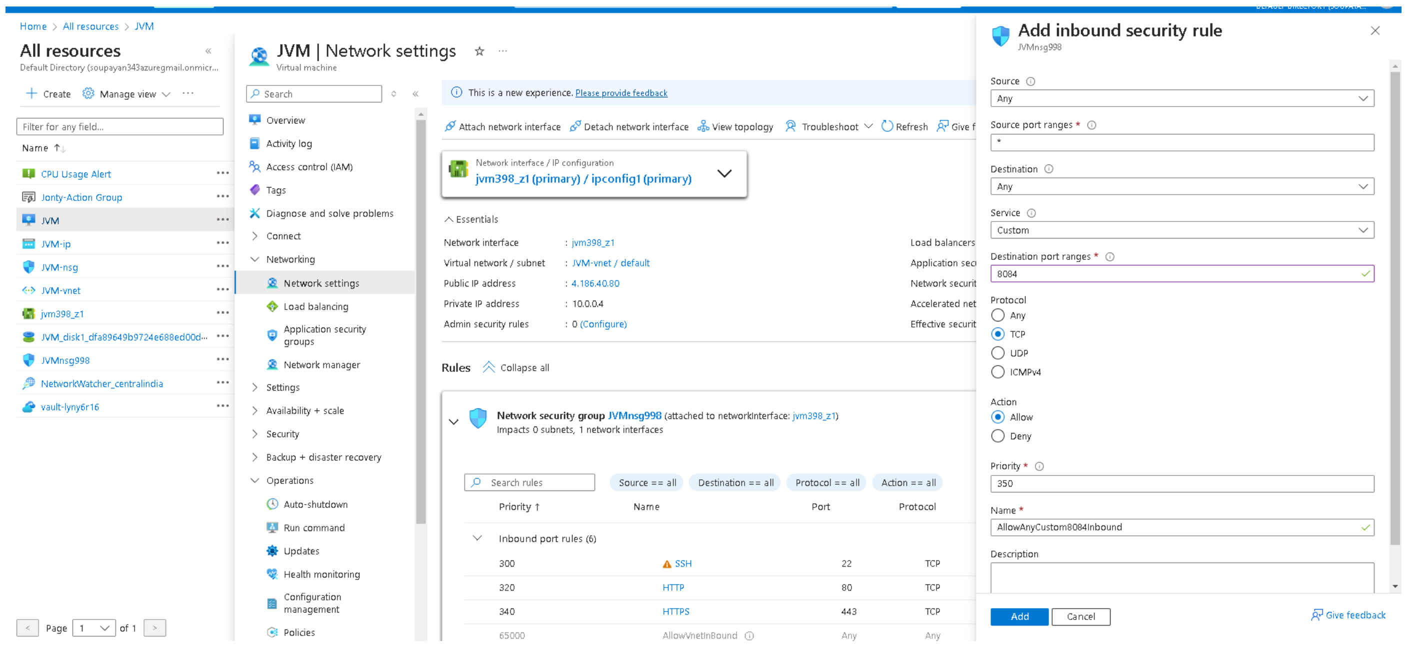
Rules

Network security group JVMnsg998 (attached to networkInterface: jvm398_z1)
Impacts 0 subnets, 1 network interfaces

Search rules | Source == all | Destination == all | Protocol == all | Action == all

Priority ↑	Name	Port	Protocol	Source	Destination	Action
Inbound port rules (6)						
300	SSH	22	TCP	Any	Any	Allow
320	HTTP	80	TCP	Any	Any	Allow
340	HTTPS	443	TCP	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow

14. Click on add new port rule(inbound). Set the destination port range to '8084' and protocol to TCP. This will allow incoming traffic from other specific sources when we make the docker container.



Add inbound security rule
JVMnsg998

Source	Any
Source port ranges *	*
Destination	Any
Service	Custom
Destination port ranges *	8084
Protocol	Any [✓ TCP] UDP ICMPv4
Action	[✓ Allow] Deny
Priority *	350
Name *	AllowAnyCustom8084Inbound
Description	

Add | Cancel | Give feedback

- ## Inside the Linux Virtual Machine:

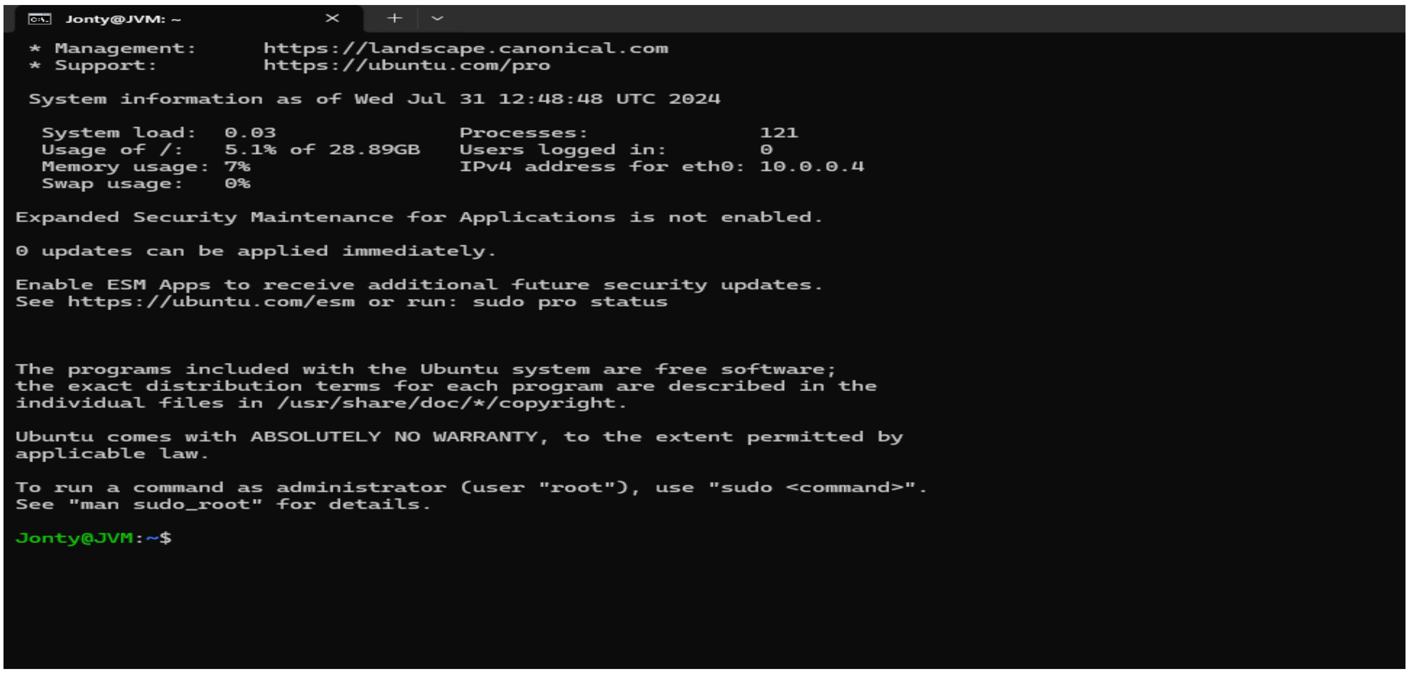
1. **ssh user@host:**

ssh: This is the command for Secure Shell (SSH), a protocol used to securely log into a remote machine and execute commands.

user: This is the username on the remote machine that you want to log into.

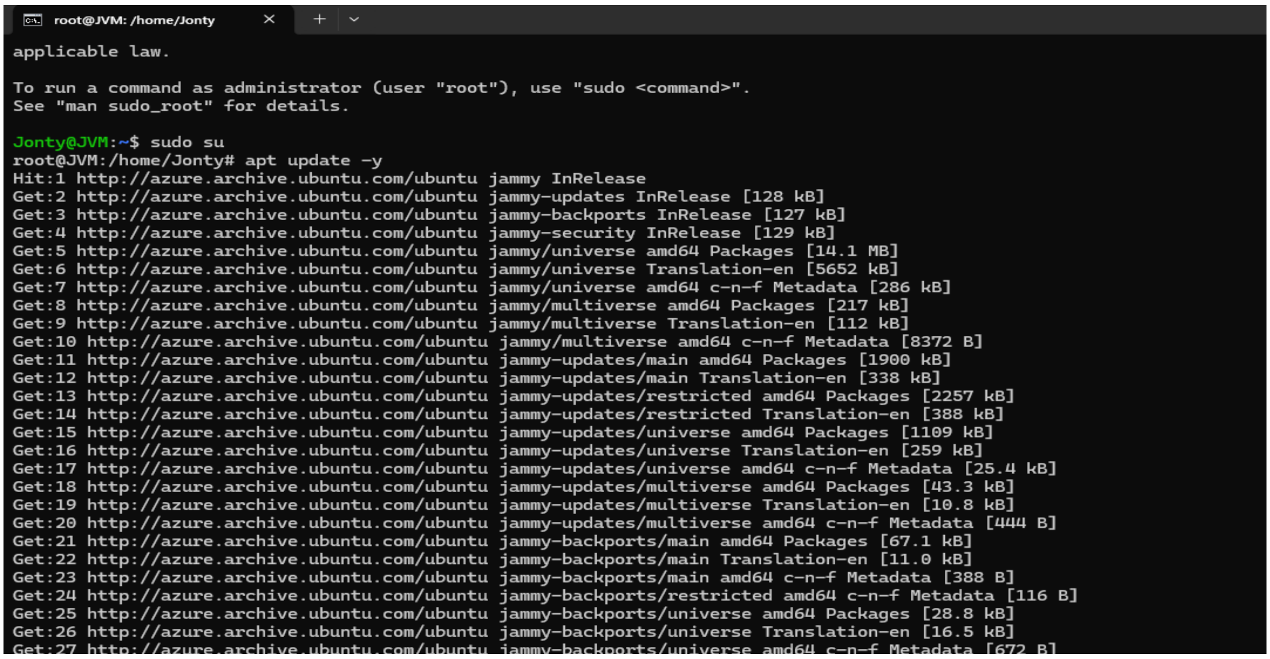
@: This symbol separates the username from the hostname or IP address.

host: This is the hostname or IP address of the remote machine you want to access.



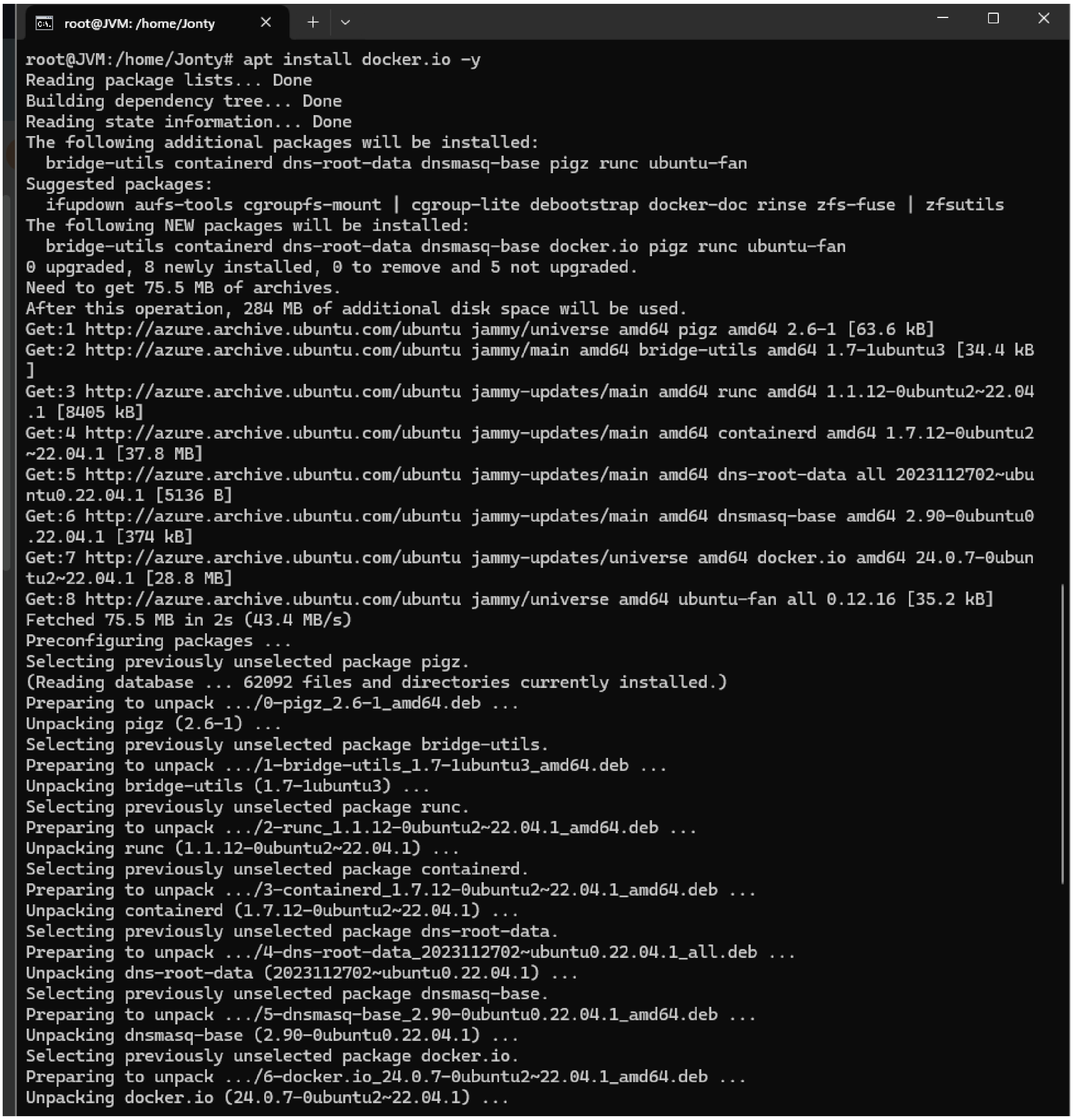
```
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Wed Jul 31 12:48:48 UTC 2024

  System load:  0.03               Processes:             121
  Usage of /:   5.1% of 28.89GB    Users logged in:       0
  Memory usage: 7%                 IPv4 address for eth0: 10.0.0.4
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

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


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

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

Jonty@JVM:~$
```

2. **sudo su:** The command sudo su allows a user to switch to the root user (superuser) with administrative privileges. By using sudo, the user temporarily gains root access without needing the root password, provided they have sudo privileges.
3. **apt update -y**: This command updates the package lists for the package manager apt (Advanced Package Tool) on a Debian-based system. The -y flag automatically confirms any prompts, making the process non-interactive.



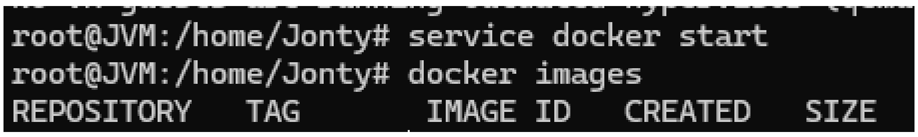
```
applicable law.

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

Jonty@JVM:~$ sudo su
root@JVM:/home/Jonty# apt update -y
Hit:1 http://azure.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://azure.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://azure.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://azure.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://azure.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://azure.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1900 kB]
Get:12 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [338 kB]
Get:13 http://azure.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [2257 kB]
Get:14 http://azure.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [388 kB]
Get:15 http://azure.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1109 kB]
Get:16 http://azure.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [259 kB]
Get:17 http://azure.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [25.4 kB]
Get:18 http://azure.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [43.3 kB]
Get:19 http://azure.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [10.8 kB]
Get:20 http://azure.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [444 B]
Get:21 http://azure.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [67.1 kB]
Get:22 http://azure.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [11.0 kB]
Get:23 http://azure.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [388 B]
Get:24 http://azure.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 B]
Get:25 http://azure.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [28.8 kB]
Get:26 http://azure.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.5 kB]
Get:27 http://azure.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [672 B]
```

4. **apt install docker.io -y**: This command installs the Docker engine from the package repository. Docker is a platform used for developing, shipping, and running applications in containers. The -y flag, as before, auto-confirms prompts.



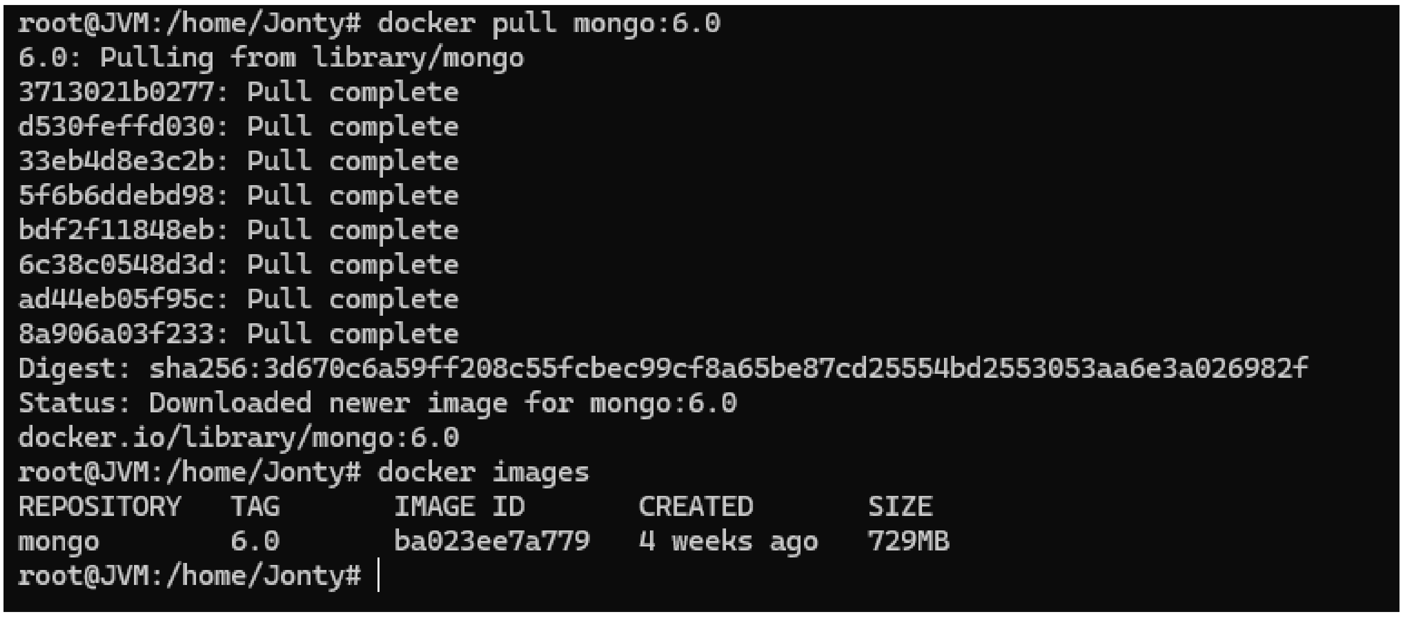
```
root@JVM:/home/Jonty# apt install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 5 not upgraded.
Need to get 75.5 MB of archives.
After this operation, 284 MB of additional disk space will be used.
Get:1 http://azure.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://azure.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB
]
Get:3 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.12-0ubuntu2~22.04
.1 [8405 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.7.12-0ubuntu2
~22.04.1 [37.8 MB]
Get:5 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dns-root-data all 2023112702~ubu
ntu0.22.04.1 [5136 B]
Get:6 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dnsmasq-base amd64 2.90-0ubuntu0
.22.04.1 [374 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 24.0.7-0ubun
tu2~22.04.1 [28.8 MB]
Get:8 http://azure.archive.ubuntu.com/ubuntu jammy/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
Fetched 75.5 MB in 2s (43.4 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 62092 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.6-1_amd64.deb ...
Unpacking pigz (2.6-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7-1ubuntu3_amd64.deb ...
Unpacking bridge-utils (1.7-1ubuntu3) ...
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.1.12-0ubuntu2~22.04.1_amd64.deb ...
Unpacking runc (1.1.12-0ubuntu2~22.04.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../3-containerd_1.7.12-0ubuntu2~22.04.1_amd64.deb ...
Unpacking containerd (1.7.12-0ubuntu2~22.04.1) ...
Selecting previously unselected package dns-root-data.
Preparing to unpack .../4-dns-root-data_2023112702~ubuntu0.22.04.1_all.deb ...
Unpacking dns-root-data (2023112702~ubuntu0.22.04.1) ...
Selecting previously unselected package dnsmasq-base.
Preparing to unpack .../5-dnsmasq-base_2.90-0ubuntu0.22.04.1_amd64.deb ...
Unpacking dnsmasq-base (2.90-0ubuntu0.22.04.1) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../6-docker.io_24.0.7-0ubuntu2~22.04.1_amd64.deb ...
Unpacking docker.io (24.0.7-0ubuntu2~22.04.1) ...
```

5. **service docker start**: This command starts the Docker service, allowing you to manage and run Docker containers.
6. **docker images**: This command lists all Docker images that are currently available on your system. Docker images are templates used to create Docker containers.



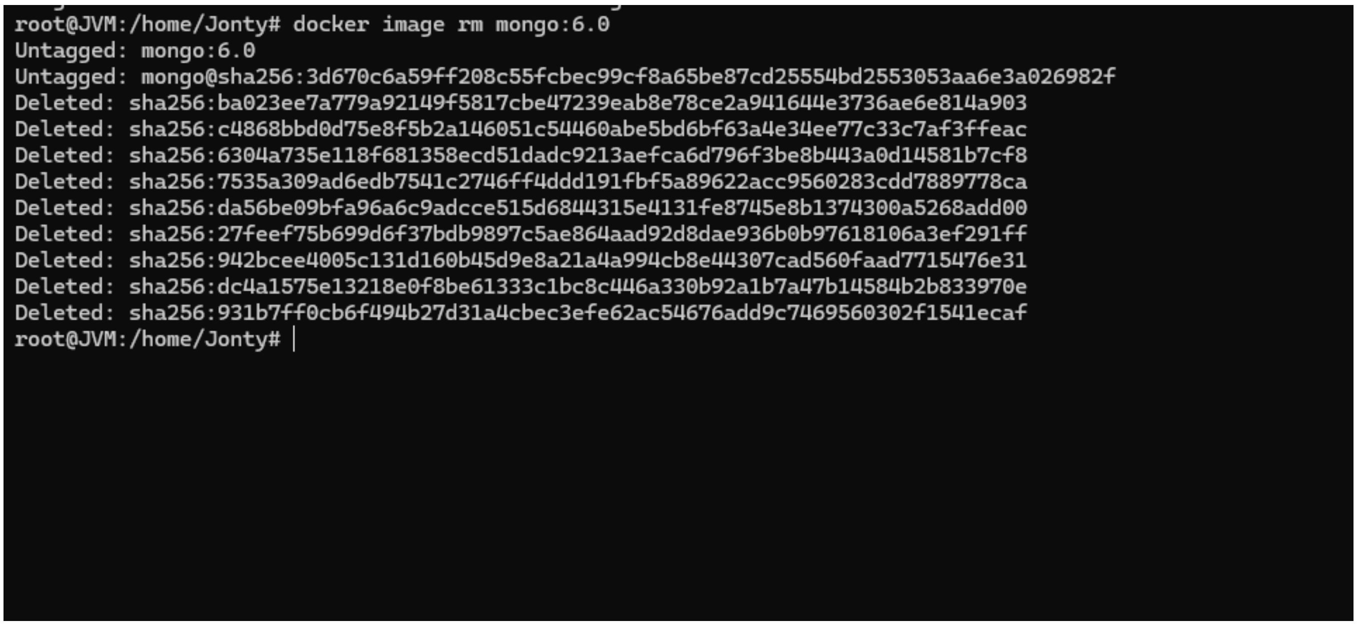
```
root@JVM:/home/Jonty# service docker start
root@JVM:/home/Jonty# docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

7. **docker pull mongo:6.0**: This command downloads the MongoDB Docker image with the tag 6.0 from the Docker Hub repository. MongoDB is a popular NoSQL database.
8. **docker images**: This command, repeated here, will now list the Docker images available on your system, including the newly pulled MongoDB image.



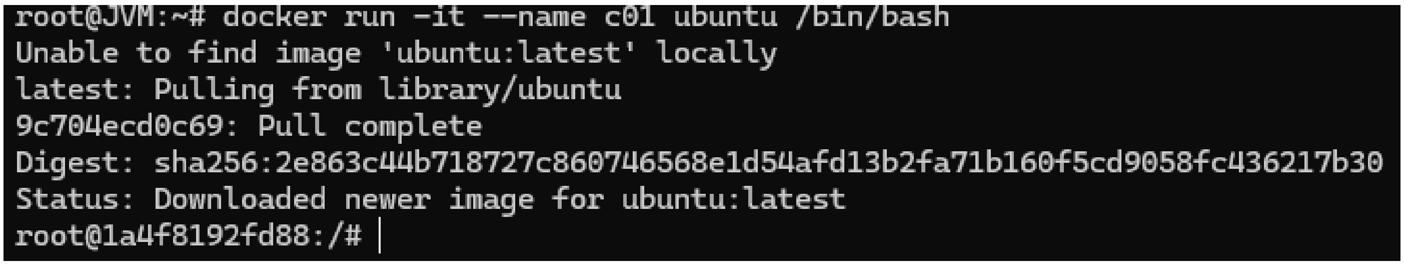
```
root@JVM:/home/Jonty# docker pull mongo:6.0
6.0: Pulling from library/mongo
3713021b0277: Pull complete
d530feffd030: Pull complete
33eb4d8e3c2b: Pull complete
5f6b6ddebd98: Pull complete
bdf2f11848eb: Pull complete
6c38c0548d3d: Pull complete
ad44eb05f95c: Pull complete
8a906a03f233: Pull complete
Digest: sha256:3d670c6a59ff208c55fcbec99cf8a65be87cd25554bd2553053aa6e3a026982f
Status: Downloaded newer image for mongo:6.0
docker.io/library/mongo:6.0
root@JVM:/home/Jonty# docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
mongo        6.0       ba023ee7a779   4 weeks ago   729MB
root@JVM:/home/Jonty#
```

9. **docker images**: This command, repeated here, will now list the Docker images available on your system, including the newly pulled MongoDB image.



```
root@JVM:/home/Jonty# docker image rm mongo:6.0
Untagged: mongo:6.0
Untagged: mongo@sha256:3d670c6a59ff208c55fcbec99cf8a65be87cd25554bd2553053aa6e3a026982f
Deleted: sha256:ba023ee7a779a92149f5817cbe47239eab8e78ce2a941644e3736ae6e814a903
Deleted: sha256:c4868bbd0d75e8f5b2a146051c54460abe5bd6bf63a4e34ee77c33c7af3ffeac
Deleted: sha256:6304a735e118f681358ecd51dadc9213aefca6d796f3be8b443a0d14581b7cf8
Deleted: sha256:7535a309ad6edb7541c2746ff4ddd191fbf5a89622acc9560283cdd7889778ca
Deleted: sha256:da56be09bfa96a6c9adcce515d6844315e4131fe8745e8b1374300a5268add00
Deleted: sha256:27feef75b699d6f37bdb9897c5ae864aad92d8dae936b0b97618106a3ef291ff
Deleted: sha256:942bcee4005c131d160b45d9e8a21a4a994cb8e44307cad560faad7715476e31
Deleted: sha256:dc4a1575e13218e0f8be61333c1bc8c446a330b92a1b7a47b14584b2b833970e
Deleted: sha256:931b7ff0cb6f494b27d31a4cbec3efe62ac54676add9c7469560302f1541ecaf
root@JVM:/home/Jonty#
```

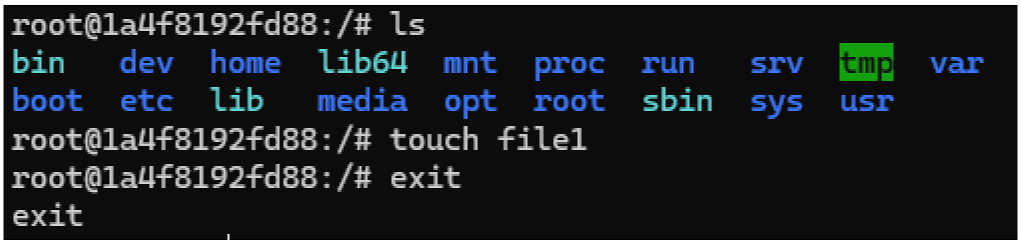
10. **docker run -it --name c01 ubuntu /bin/bash**: This command creates and starts a new Docker container from the ubuntu image, names it c01, and provides an interactive terminal session within the container. The -it flag enables interactive mode and allocates a pseudo-TTY. The /bin/bash argument specifies that the Bash shell should be run inside the container.



```
root@JVM:~# docker run -it --name c01 ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
9c704ecd0c69: Pull complete
Digest: sha256:2e863c44b718727c860746568e1d54afd13b2fa71b160f5cd9058fc436217b30
Status: Downloaded newer image for ubuntu:latest
root@1a4f8192fd88:/#
```

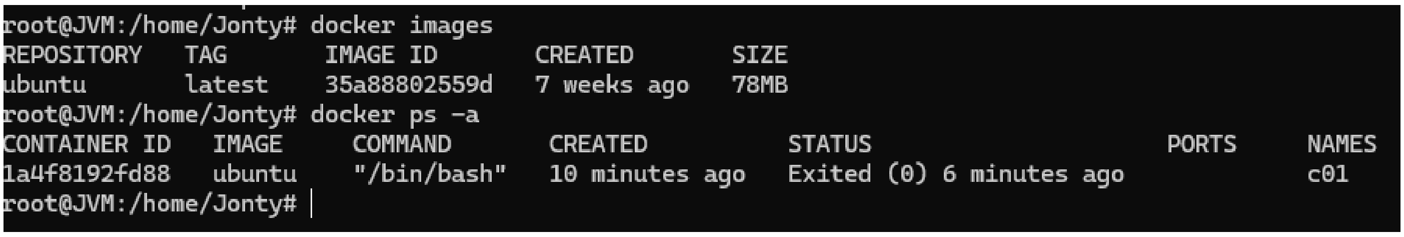
11. **ls**: This command lists the contents of the current directory within the Docker container. It helps to see the files and directories present at that location.

12. **touch file1**: This command creates a new empty file named file1 inside the Docker container. The touch command updates the timestamp of the file if it already exists or creates a new file if it doesn't.
13. **exit**: This command exits the Bash shell within the Docker container, effectively stopping the interactive session and stopping the container unless it was run in detached mode.



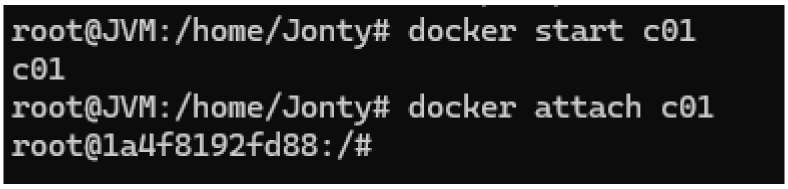
```
root@1a4f8192fd88:/# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
root@1a4f8192fd88:/# touch file1
root@1a4f8192fd88:/# exit
exit
```

14. **docker images**: This command lists all Docker images available on the local system. It shows details like the repository name, tags, image ID, creation date, and size of each image.
15. **docker ps -a**: This command lists all Docker containers on the system, including those that are running, stopped, and exited. It provides details like container ID, names, status, and the command that was run in the container.



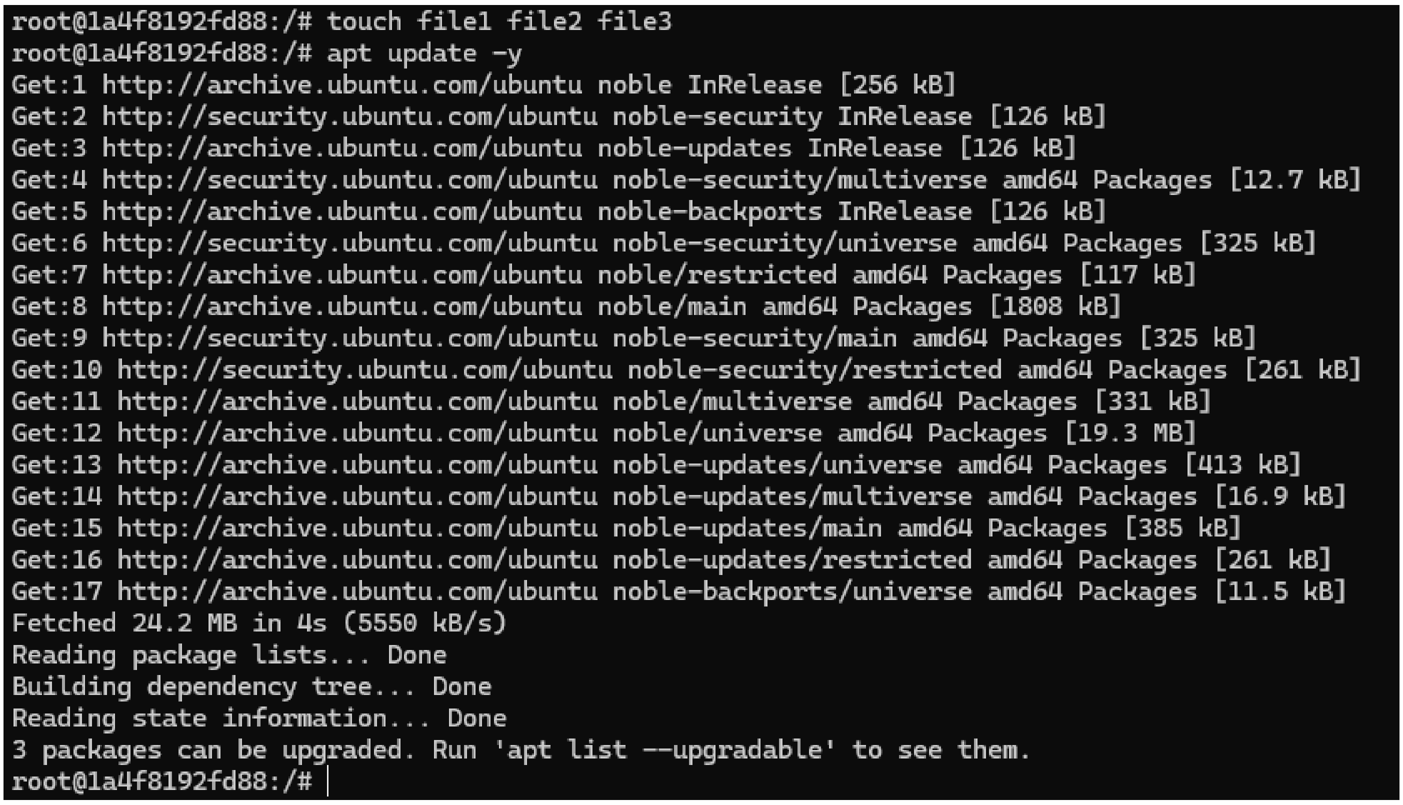
```
root@JVM:/home/Jonty# docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    35a88802559d   7 weeks ago   78MB
root@JVM:/home/Jonty# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS                      PORTS     NAMES
1a4f8192fd88   ubuntu    "/bin/bash"   10 minutes ago   Exited (0) 6 minutes ago              c01
root@JVM:/home/Jonty#
```

16. **docker start c01**: This command starts the previously created and stopped Docker container named c01. The container will run in the background.
17. **docker attach c01**: This command attaches your terminal to a running container named c01. It allows you to interact with the container's shell as if you were inside it.



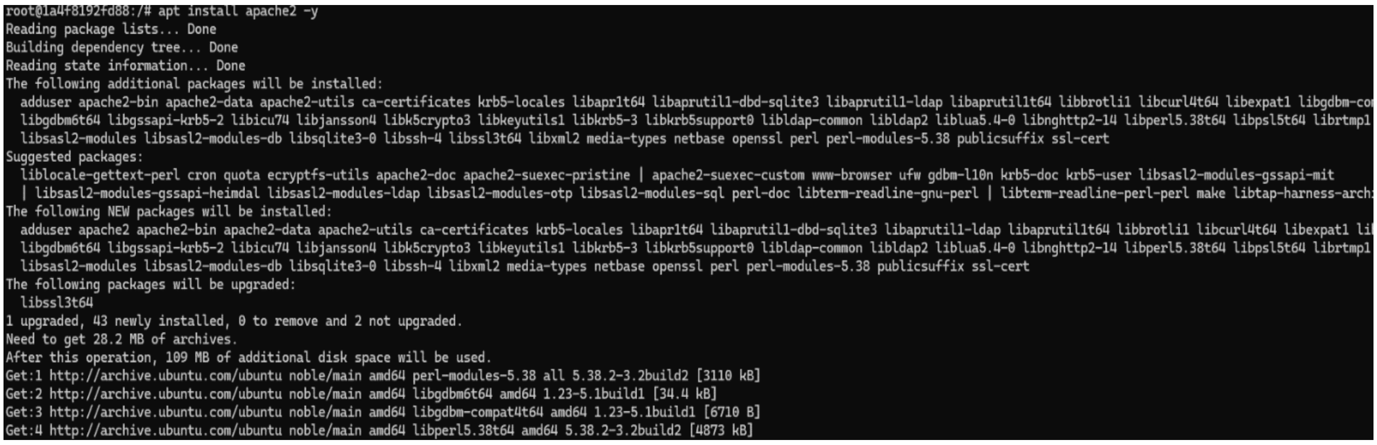
```
root@JVM:/home/Jonty# docker start c01
c01
root@JVM:/home/Jonty# docker attach c01
root@1a4f8192fd88:/#
```

18. **touch file1 file2 file3**: This command creates three new empty files named file1, file2, and file3 inside the attached Docker container (c01).
19. **apt update -y**: This command updates the package lists inside the Docker container for the apt package manager. The -y flag automatically confirms any prompts, making the process non-interactive.



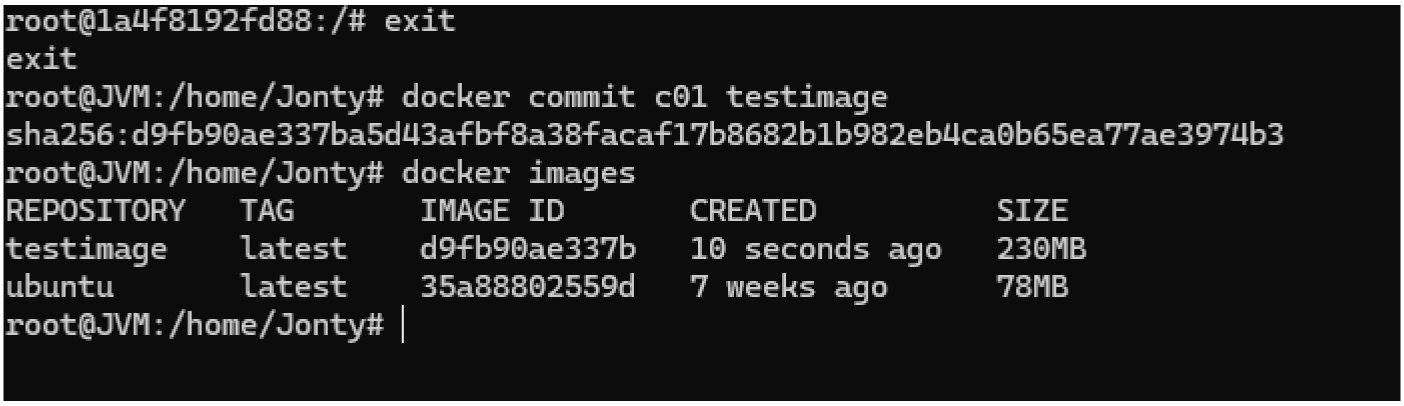
```
root@1a4f8192fd88:/# touch file1 file2 file3
root@1a4f8192fd88:/# apt update -y
Get:1 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [12.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [325 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble/restricted amd64 Packages [117 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages [1808 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [325 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [261 kB]
Get:11 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [331 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
Get:13 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [413 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [16.9 kB]
Get:15 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [385 kB]
Get:16 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [261 kB]
Get:17 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [11.5 kB]
Fetched 24.2 MB in 4s (5550 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
3 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@1a4f8192fd88:/# |
```

20. **apt install apache2 -y**: This command installs the Apache HTTP server inside the Docker container. Apache2 is a widely used web server. The -y flag, as before, auto-confirms prompts, allowing the installation to proceed without manual intervention.



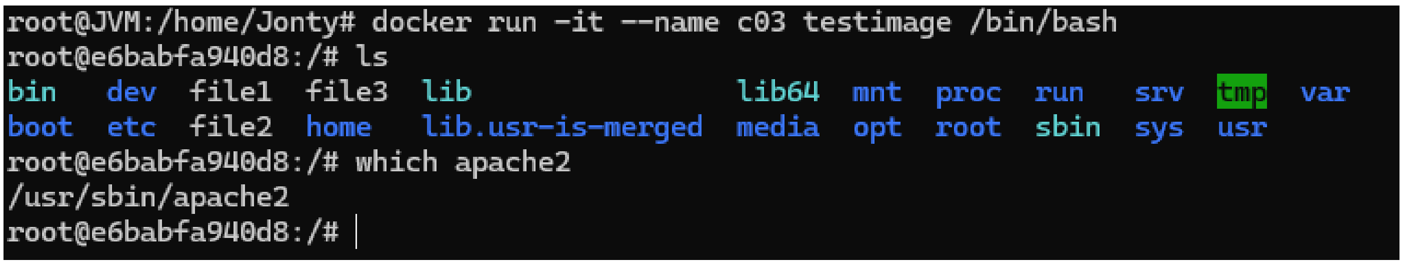
```
root@1a4f8192fd88:/# apt install apache2 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  adduser apache2-bin apache2-data apache2-utils ca-certificates krb5-locales libapr1t64 libaprutil1-dbd-sqlite3 libaprutil1-ldap libaprutil1t64 libbrotli1 libcurl4t64 libexpat1 libgdbm-co
  libgdbm6t64 libgssapi-krb5-2 libicu74 libjansson4 libk5crypto3 libkeyutils1 libkrb5-3 libkrb5support0 libldap-common libldap2 liblua5.4-0 libnghttp2-14 libperl5.38t64 libpsl5t64 librtmp1
  libsasl2-modules libsasl2-modules-db libsqlite3-0 libssh-4 libssl3t64 libxml2 media-types netbase openssl perl perl-modules-5.38 publicsuffix ssl-cert
Suggested packages:
  liblocale-gettext-perl cron quota ecryptfs-utils apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser ufw gdbm-l10n krb5-doc krb5-user libsasl2-modules-gssapi-mit
  | libsasl2-modules-gssapi-heimdal libsasl2-modules-ldap libsasl2-modules-otp libsasl2-modules-sql perl-doc libterm-readline-gnu-perl | libterm-readline-perl-perl make libtap-harness-arch
The following NEW packages will be installed:
  adduser apache2 apache2-bin apache2-data apache2-utils ca-certificates krb5-locales libapr1t64 libaprutil1-dbd-sqlite3 libaprutil1-ldap libaprutil1t64 libbrotli1 libcurl4t64 libexpat1 li
  libgdbm6t64 libgssapi-krb5-2 libicu74 libjansson4 libk5crypto3 libkeyutils1 libkrb5-3 libkrb5support0 libldap-common libldap2 liblua5.4-0 libnghttp2-14 libperl5.38t64 libpsl5t64 librtmp1
  libsasl2-modules libsasl2-modules-db libsqlite3-0 libssh-4 libxml2 media-types netbase openssl perl perl-modules-5.38 publicsuffix ssl-cert
The following packages will be upgraded:
  libssl3t64
1 upgraded, 43 newly installed, 0 to remove and 2 not upgraded.
Need to get 28.2 MB of archives.
After this operation, 109 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble/main amd64 perl-modules-5.38 all 5.38.2-3.2build2 [3110 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble/main amd64 libgdbm6t64 amd64 1.23-5.1build1 [34.4 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble/main amd64 libgdbm-compat4t64 amd64 1.23-5.1build1 [6710 B]
Get:4 http://archive.ubuntu.com/ubuntu noble/main amd64 libperl5.38t64 amd64 5.38.2-3.2build2 [4873 kB]
```

21. **exit**: This command exits the Bash shell within the Docker container, effectively stopping the interactive session and stopping the container unless it was run in detached mode.
22. **docker commit c01 testimage**: This command creates a new Docker image from the container c01, saving its current state. The new image is named testimage. This is useful for preserving the state of a container with all its modifications.
23. **docker commit c01 testimage**: This command creates a new Docker image from the container c01, saving its current state. The new image is named testimage. This is useful for preserving the state of a container with all its modifications.



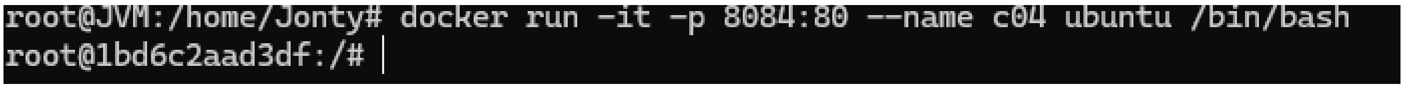
```
root@1a4f8192fd88:/# exit
exit
root@JVM:/home/Jonty# docker commit c01 testimage
sha256:d9fb90ae337ba5d43afbf8a38facaf17b8682b1b982eb4ca0b65ea77ae3974b3
root@JVM:/home/Jonty# docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
testimage    latest    d9fb90ae337b   10 seconds ago   230MB
ubuntu       latest    35a88802559d   7 weeks ago      78MB
root@JVM:/home/Jonty# 
```

24. **docker run -it --name c03 testimage /bin/bash**: This command creates and starts a new Docker container named c03 from the testimage image. It opens an interactive terminal session inside the container with the Bash shell.
25. **ls**: This command lists the contents of the current directory within the Docker container c03. It helps to see the files and directories present at that location.
26. **which apache2**: This command locates the binary of the Apache2 executable inside the Docker container c03. It shows the path where apache2 is installed if it exists.



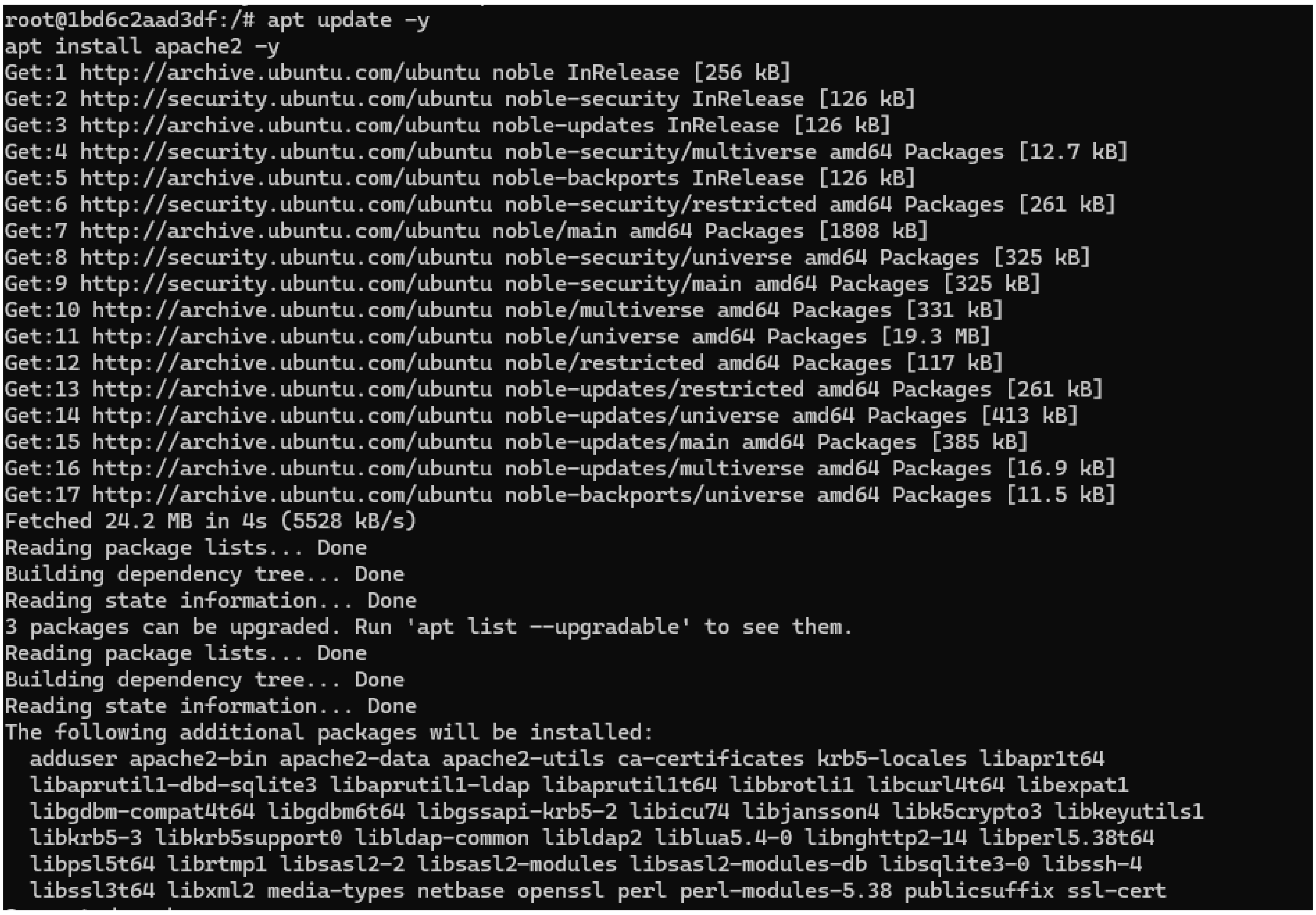
```
root@JVM:/home/Jonty# docker run -it --name c03 testimage /bin/bash
root@e6babfa940d8:/# ls
bin   dev  file1  file3  lib                lib64  mnt  proc  run   srv  tmp  var
boot  etc  file2  home   lib.usr-is-merged  media  opt  root  sbin  sys  usr
root@e6babfa940d8:/# which apache2
/usr/sbin/apache2
root@e6babfa940d8:/# 
```

27. **docker run -it -p 8084:80 --name c04 ubuntu /bin/bash**: This command creates and starts a new Docker container named c04 from the ubuntu image. It maps port 8084 on the host to port 80 on the container, allowing access to services running on port 80 of the container via port 8084 on the host. The interactive terminal session inside the container runs the Bash shell.



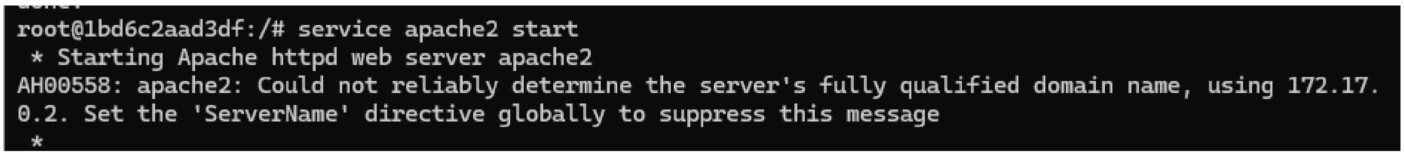
```
root@JVM:/home/Jonty# docker run -it -p 8084:80 --name c04 ubuntu /bin/bash
root@1bd6c2aad3df:/# 
```

28. **apt update -y**: This command updates the package lists inside the Docker container c04 for the apt package manager. The -y flag automatically confirms any prompts, making the process non-interactive.
29. **apt install apache2 -y**: This command installs the Apache HTTP server inside the Docker container c04. Apache2 is a widely used web server. The -y flag auto-confirms prompts, allowing the installation to proceed without manual intervention.



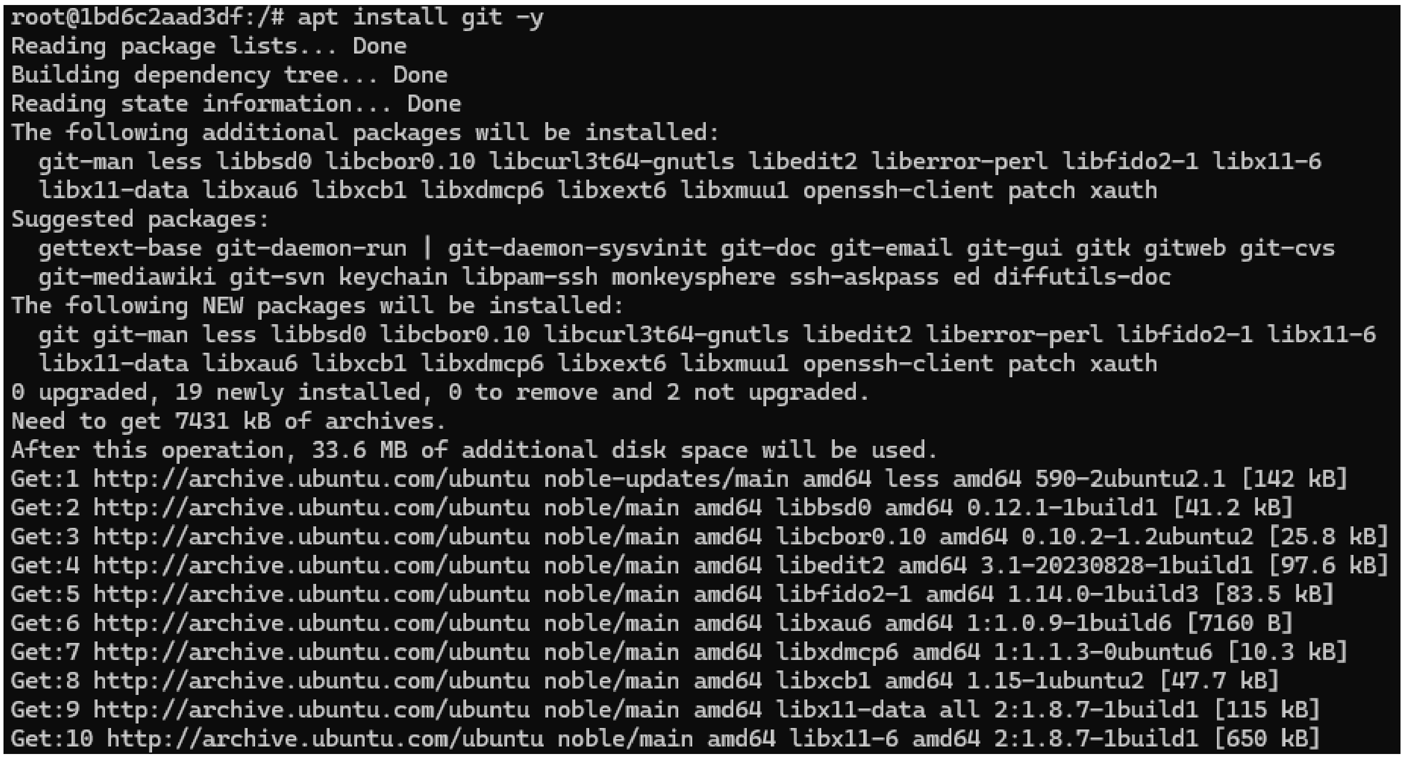
```
root@1bd6c2aad3df:/# apt update -y
apt install apache2 -y
Get:1 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [12.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [261 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages [1808 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [325 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [325 kB]
Get:10 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [331 kB]
Get:11 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
Get:12 http://archive.ubuntu.com/ubuntu noble/restricted amd64 Packages [117 kB]
Get:13 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [261 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [413 kB]
Get:15 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [385 kB]
Get:16 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [16.9 kB]
Get:17 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [11.5 kB]
Fetched 24.2 MB in 4s (5528 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
3 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  adduser apache2-bin apache2-data apache2-utils ca-certificates krb5-locales libapr1t64
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libaprutil1t64 libbrotli1 libcurl4t64 libexpat1
  libgdbm-compat4t64 libgdbm6t64 libgssapi-krb5-2 libicu74 libjansson4 libk5crypto3 libkeyutils1
  libkrb5-3 libkrb5support0 libldap-common libldap2 liblua5.4-0 libnghttp2-14 libperl5.38t64
  libpsl5t64 librtmp1 libsasl2-2 libsasl2-modules libsasl2-modules-db libsqlite3-0 libssh-4
  libssl3t64 libxml2 media-types netbase openssl perl perl-modules-5.38 publicsuffix ssl-cert
```

30. **service apache2 start**: This command starts the Apache2 service inside the Docker container c04, allowing it to serve web pages.



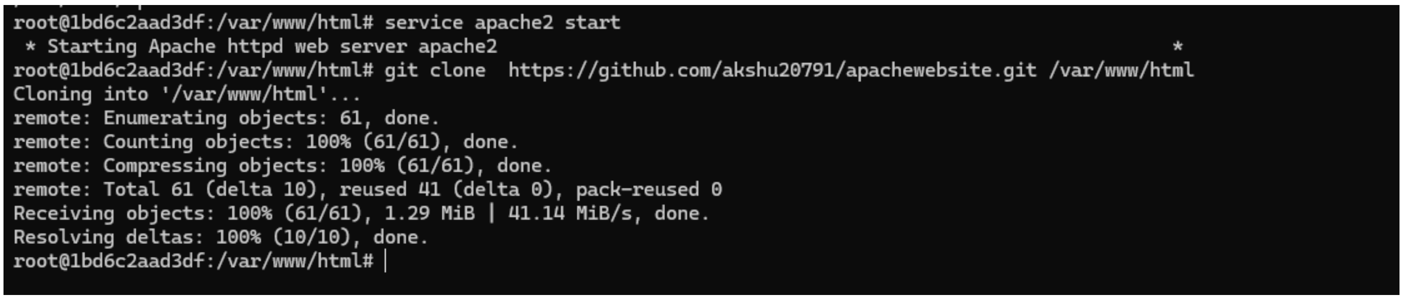
```
root@1bd6c2aad3df:/# service apache2 start
 * Starting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.
0.2. Set the 'ServerName' directive globally to suppress this message
 *
```

31. **apt install git -y**: This command installs Git, a version control system, inside the Docker container c04. The -y flag auto-confirms prompts, making the installation non-interactive.



```
root@1bd6c2aad3df:/# apt install git -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  git-man less libbsd0 libcbor0.10 libcurl3t64-gnutls libedit2 liberror-perl libfido2-1 libx11-6
  libx11-data libxau6 libxcb1 libxdmcp6 libxext6 libxmuu1 openssh-client patch xauth
Suggested packages:
  gettext-base git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs
  git-mediawiki git-svn keychain libpam-ssh monkeysphere ssh-askpass ed diffutils-doc
The following NEW packages will be installed:
  git git-man less libbsd0 libcbor0.10 libcurl3t64-gnutls libedit2 liberror-perl libfido2-1 libx11-6
  libx11-data libxau6 libxcb1 libxdmcp6 libxext6 libxmuu1 openssh-client patch xauth
0 upgraded, 19 newly installed, 0 to remove and 2 not upgraded.
Need to get 7431 kB of archives.
After this operation, 33.6 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 less amd64 590-2ubuntu2.1 [142 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble/main amd64 libbsd0 amd64 0.12.1-1build1 [41.2 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble/main amd64 libcbor0.10 amd64 0.10.2-1.2ubuntu2 [25.8 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble/main amd64 libedit2 amd64 3.1-20230828-1build1 [97.6 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble/main amd64 libfido2-1 amd64 1.14.0-1build3 [83.5 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble/main amd64 libxau6 amd64 1:1.0.9-1build6 [7160 B]
Get:7 http://archive.ubuntu.com/ubuntu noble/main amd64 libxdmcp6 amd64 1:1.1.3-0ubuntu6 [10.3 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble/main amd64 libxcb1 amd64 1.15-1ubuntu2 [47.7 kB]
Get:9 http://archive.ubuntu.com/ubuntu noble/main amd64 libx11-data all 2:1.8.7-1build1 [115 kB]
Get:10 http://archive.ubuntu.com/ubuntu noble/main amd64 libx11-6 amd64 2:1.8.7-1build1 [650 kB]
```

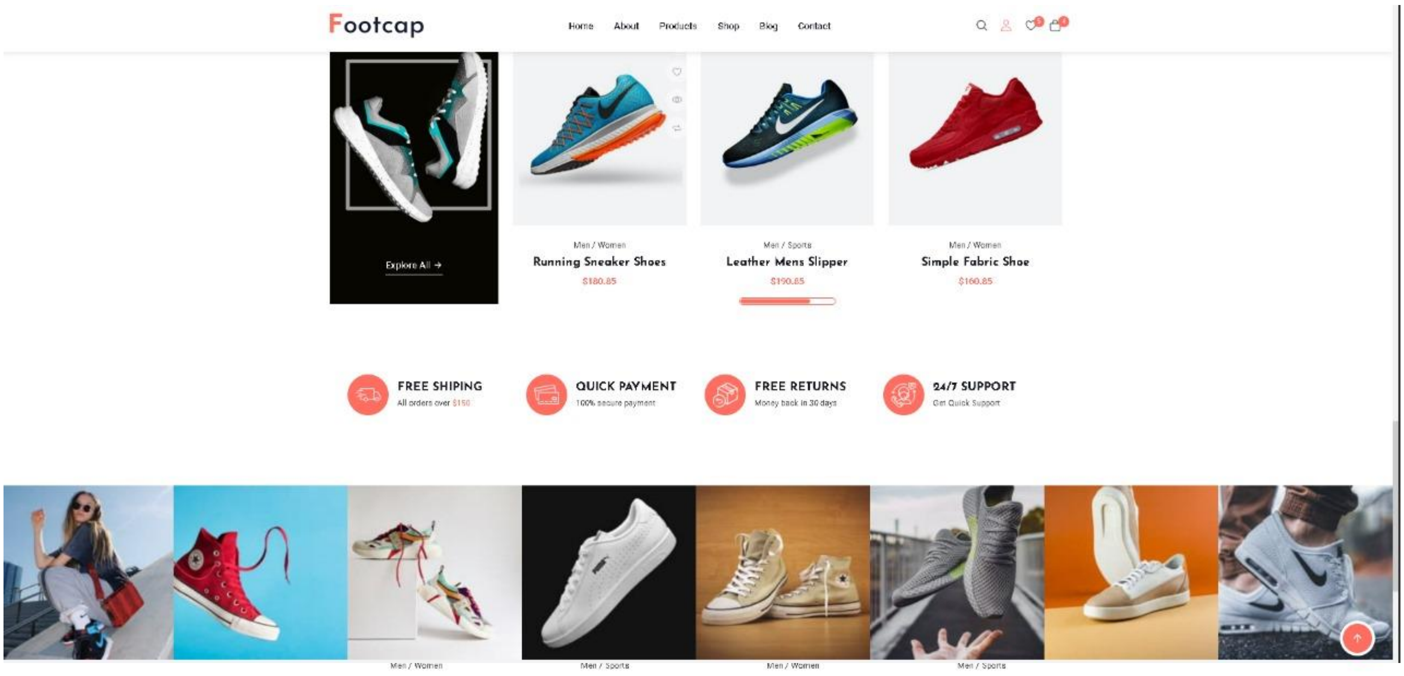
32. **cd /var/www/html**: This command changes the current directory to /var/www/html inside the Docker container c04, which is the default directory where Apache serves web content.
33. **service apache2 start**: This command starts the Apache2 service inside the Docker container c04, allowing it to serve web pages.
34. **git clone https://github.com/akshu20791/apachewebsite.git /var/www/html**: This command clones the Git repository located at the provided URL into the /var/www/html directory inside the Docker container c04. This populates the directory with the contents of the repository, which in this case contains the files for a website.

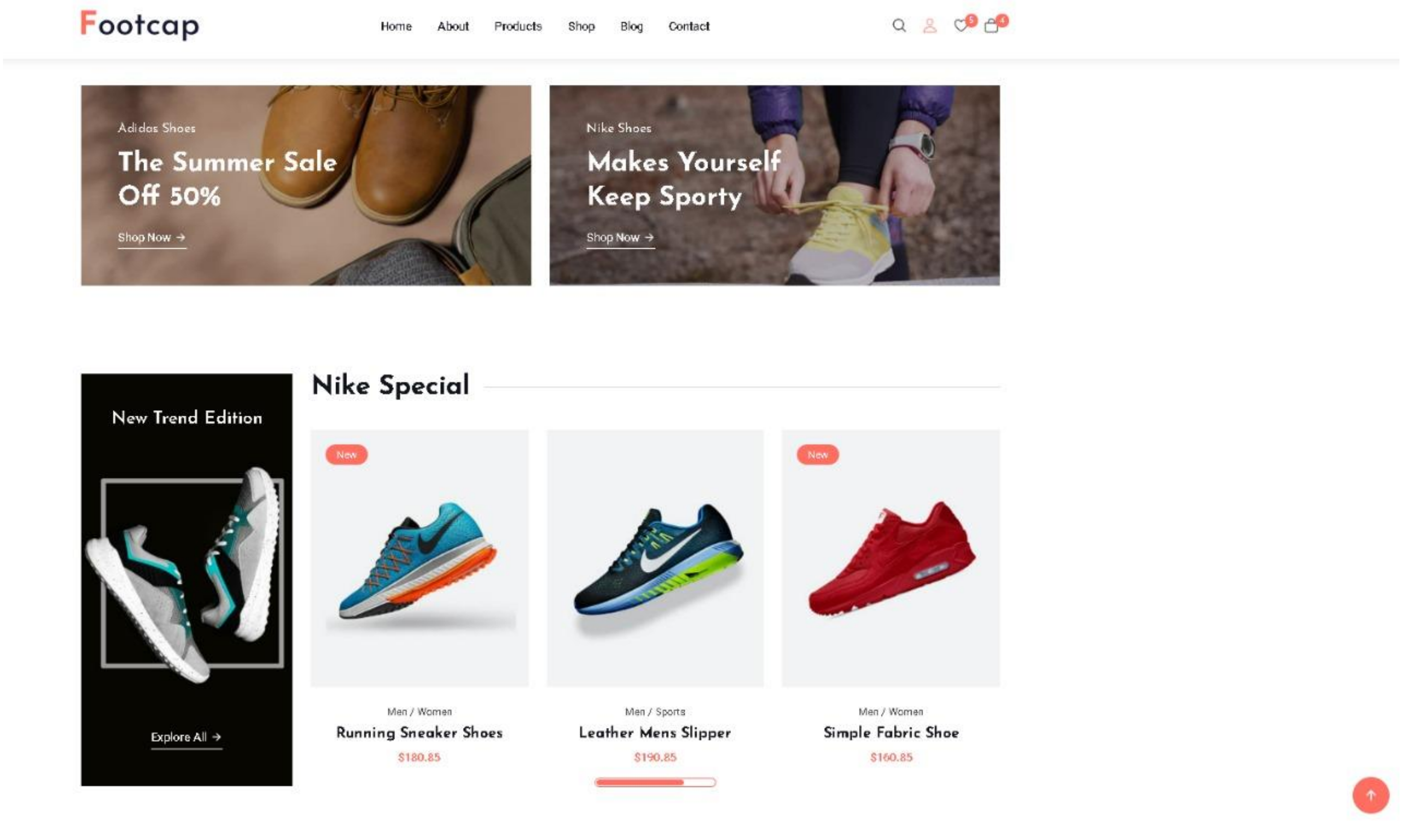


```
root@1bd6c2aad3df:/var/www/html# service apache2 start
 * Starting Apache httpd web server apache2                                          *
root@1bd6c2aad3df:/var/www/html# git clone  https://github.com/akshu20791/apachewebsite.git /var/www/html
Cloning into '/var/www/html'...
remote: Enumerating objects: 61, done.
remote: Counting objects: 100% (61/61), done.
remote: Compressing objects: 100% (61/61), done.
remote: Total 61 (delta 10), reused 41 (delta 0), pack-reused 0
Receiving objects: 100% (61/61), 1.29 MiB | 41.14 MiB/s, done.
Resolving deltas: 100% (10/10), done.
root@1bd6c2aad3df:/var/www/html# |
```

Final Website:

- To Use the website , go to your local browser and put in "**http://<server-ip-or-domain>:<port number>".**
- We will be using http://localhost:8084





❖ Components:

1. Container based application

A container-based application is a software application that is packaged with its dependencies, libraries, and configuration files into a container image. Containers provide a lightweight, consistent, and portable environment for running applications across different computing environments, such as development, testing, and production. Containerized applications are applications run in isolated packages of code called containers. Containers include all the dependencies that an application might need to run on any host operating system, such as libraries, binaries, configuration files, and frameworks, into a single lightweight executable.

I. Virtualization

1. Definition: Virtualization involves creating multiple virtual machines (VMs) on a single physical hardware host. Each VM includes a full operating system (OS), virtual hardware, and an application.

2. Hypervisor: VMs are managed by a hypervisor, which is a software layer that allows multiple VMs to run on the same physical machine. Common hypervisors include VMware ESXi, Microsoft Hyper-V, and open-source options like KVM.

3. Isolation: Each VM is fully isolated from others, with its own OS and resources. This strong isolation provides security and stability but can result in higher overhead.

4. Resource Usage: VMs require more resources (CPU, memory, storage) because each VM runs its own OS. This can lead to inefficiencies and higher overhead.

5. Use Cases: Virtualization is ideal for running multiple, diverse applications on a single physical server, consolidating hardware, and supporting legacy applications.

II. Containerization

1. Definition: Containerization involves packaging an application with its dependencies, libraries, and configuration files into a container. Containers share the host OS kernel but run in isolated user spaces.

2. Container Engine: Containers are managed by a container engine (such as Docker), which provides tools for creating, running, and managing containers.

3. Isolation: Containers provide process and filesystem isolation but share the host OS kernel. This lighter isolation allows for faster performance but can be less secure compared to VMs.

4. Resource Usage: Containers are more lightweight and efficient because they share the host OS kernel and require fewer resources. This allows for higher density and faster startup times.

5. Use Cases: Containerization is ideal for developing, testing, and deploying microservices and cloud-native applications. It's well-suited for environments where rapid scaling and portability are important.

Key Differences

1. Architecture:

- Virtualization: Full OS per VM.
- Containerization: Shared OS kernel, isolated user space.

2. Performance:

- Virtualization: Higher overhead due to multiple OS instances.
- Containerization: Lower overhead, more efficient resource usage.

3. Startup Time:

- Virtualization: Slower startup times (booting an OS).
- Containerization: Faster startup times (starting a container).

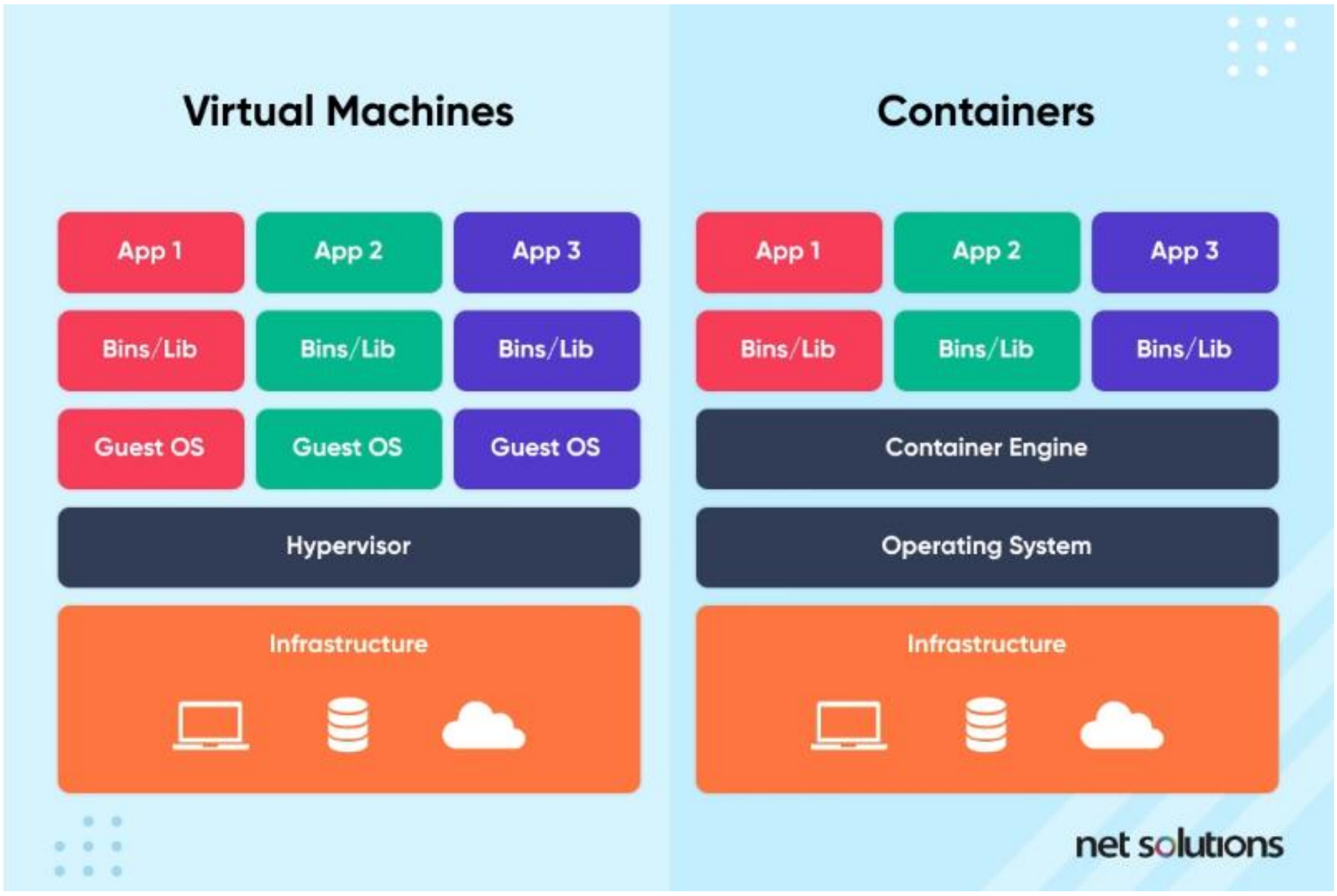
4. Isolation:

- Virtualization: Stronger isolation with full OS instances.
- Containerization: Lighter isolation, sharing the host OS kernel.

5. Management:

- Virtualization: Managed by hypervisors.
- Containerization: Managed by container engines (e.g., Docker) and orchestration tools (e.g., Kubernetes).

Both technologies have their own strengths and are often used together in modern IT environments to leverage the benefits of each. Virtualization provides strong isolation and compatibility with a wide range of OSs, while containerization offers lightweight, efficient, and portable environments for applications.



2. <u>**Subscription Plans**</u>

Azure offers a variety of subscription plans tailored to different needs, ranging from individual developers to large enterprises. These subscriptions provide access to a broad range of Azure services, including virtual machines, databases, storage, AI, and more. Below is an overview of the main types of Azure subscriptions:

Azure Subscription Plans

1. Free Account

- **Target Audience**: Individuals and small teams looking to explore Azure services.
- **Features:**
  - $200 credit for the first 30 days.
  - 12 months of free access to popular services.
  - Always-free services, like Azure Functions, Logic Apps, and more.
- **Limitations:** Limited by the credit amount and free tier service quotas.

2. Pay-As-You-Go

- **Target Audience**: Individuals, startups, and businesses needing flexible payment options.
- **Features:**
  - No upfront costs.

  - Pay only for what you use.
  - Access to all Azure services.
- **Billing**: Monthly billing based on actual usage.

3. Azure Dev/Test

- **Target Audience**: Development and testing teams.
- **Features:**
  - Discounted rates on Azure services for development and testing.
  - Access to Windows 10 and Windows 11 VMs.
  - Ability to provision services without incurring production costs.
- **Limitations**: Not for production use.

4. Enterprise Agreement (EA)

- **Target Audience**: Large organizations and enterprises.
- **Features:**
  - Custom pricing and payment options.
  - Enterprise-level support and SLAs.
  - Access to the Azure portal for centralized management.
  - Reserved instances and additional discounts.
- **Billing:** Annual or multi-year commitments with customized billing options.

5. Microsoft Customer Agreement

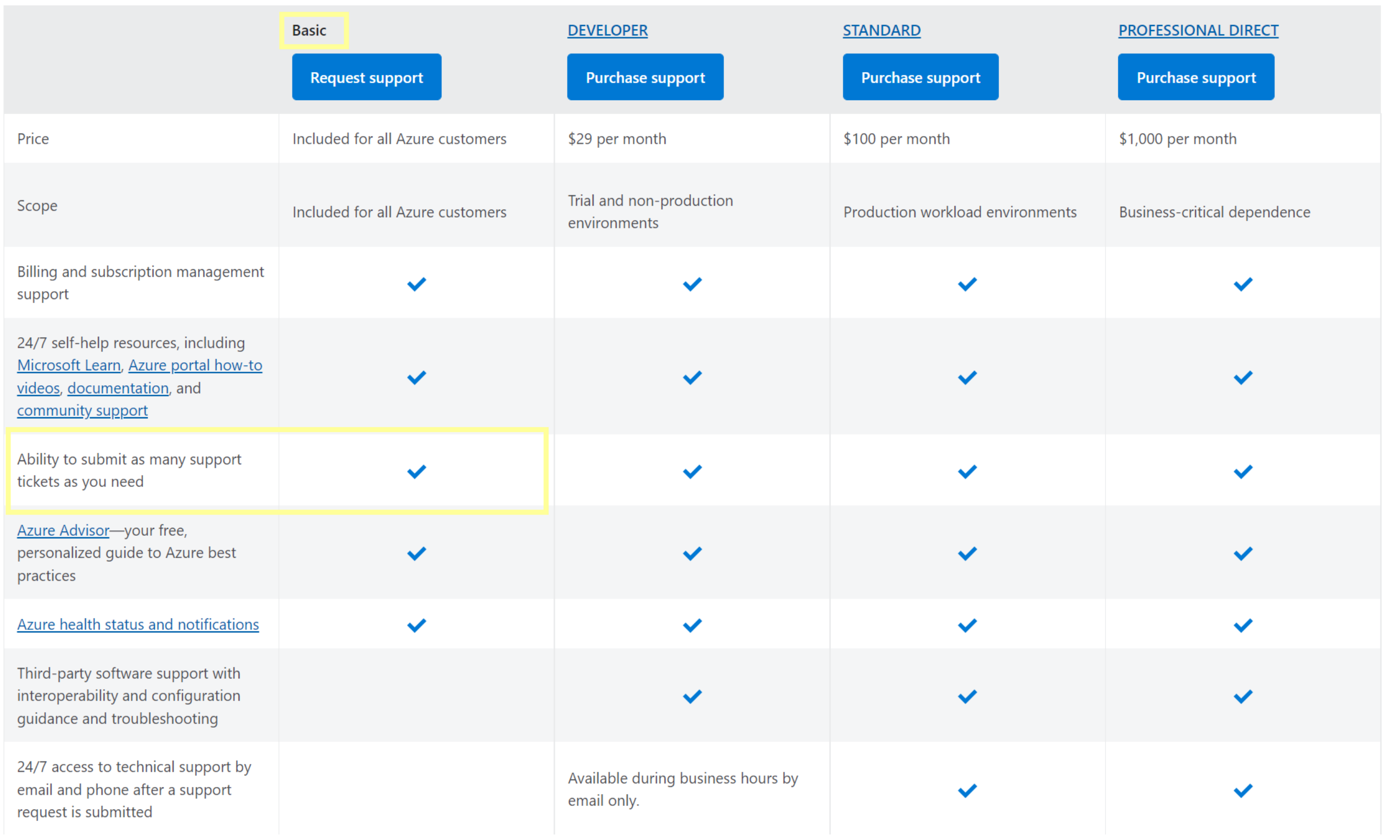
- **Target Audience**: Medium to large organizations.
- **Features:**
  - Simplified purchasing and billing.
  - Flexible terms and scalable options.
  - Comprehensive access to all Azure services.
  - Management and billing through the Azure portal.
- **Billing:** Monthly billing with pay-as-you-go or reserved options.

6. CSP (Cloud Solution Provider)

- **Target Audience**: Organizations working with Microsoft partners.
- **Features:**

- Partner-managed billing and support.
- Access to all Azure services through a partner.
- Custom pricing and service bundles.

- **Billing**: Managed by the CSP partner, with customized billing options.

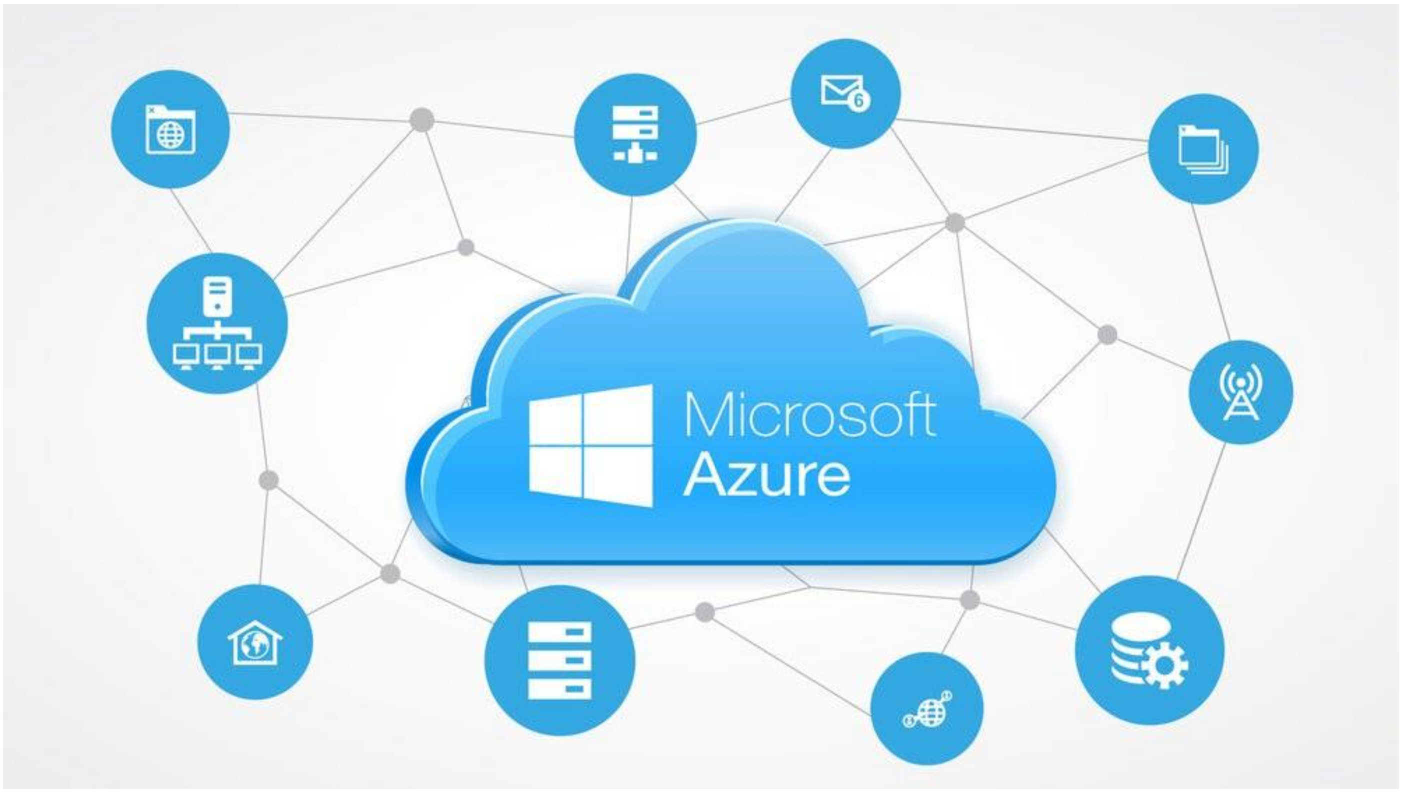


	Basic Request support	DEVELOPER Purchase support	STANDARD Purchase support	PROFESSIONAL DIRECT Purchase support
Price	Included for all Azure customers	$29 per month	$100 per month	$1,000 per month
Scope	Included for all Azure customers	Trial and non-production environments	Production workload environments	Business-critical dependence
Billing and subscription management support	✓	✓	✓	✓
24/7 self-help resources, including Microsoft Learn, Azure portal how-to videos, documentation, and community support	✓	✓	✓	✓
Ability to submit as many support tickets as you need	✓	✓	✓	✓
Azure Advisor—your free, personalized guide to Azure best practices	✓	✓	✓	✓
Azure health status and notifications	✓	✓	✓	✓
Third-party software support with interoperability and configuration guidance and troubleshooting		✓	✓	✓
24/7 access to technical support by email and phone after a support request is submitted		Available during business hours by email only.	✓	✓

Key Benefits of Azure Subscriptions

- **Scalability**: Easily scale resources up or down based on demand.
- **Flexibility**: Wide range of services and pricing options to fit various needs and budgets.
- **Global Reach**: Azure's global network of data centres provides low-latency access and redundancy.
- **Security and Compliance**: Comprehensive security features and compliance with various industry standards.
- **Integration**: Seamless integration with other Microsoft products and services, such as Office 365, Dynamics 365, and Active Directory.

3. Resources



1. Compute

- **Virtual Machines:** Provision Windows and Linux virtual machines in seconds.
- **App Services:** Build, deploy, and scale web apps and APIs.

2. Storage

- **Azure Blob Storage:** Scalable object storage for unstructured data.
- **Azure File Storage:** Fully managed file shares in the cloud.
- **Azure Queue Storage:** Messaging service for reliable messaging between application components.
- **Azure Disk Storage:** High-performance, durable block storage for Azure VMs.

3. Networking

- **Virtual Network:** Provision private networks and optionally connect to on-premises datacenters.
- **Azure Load Balancer:** Distribute traffic across multiple VMs.

4. Security

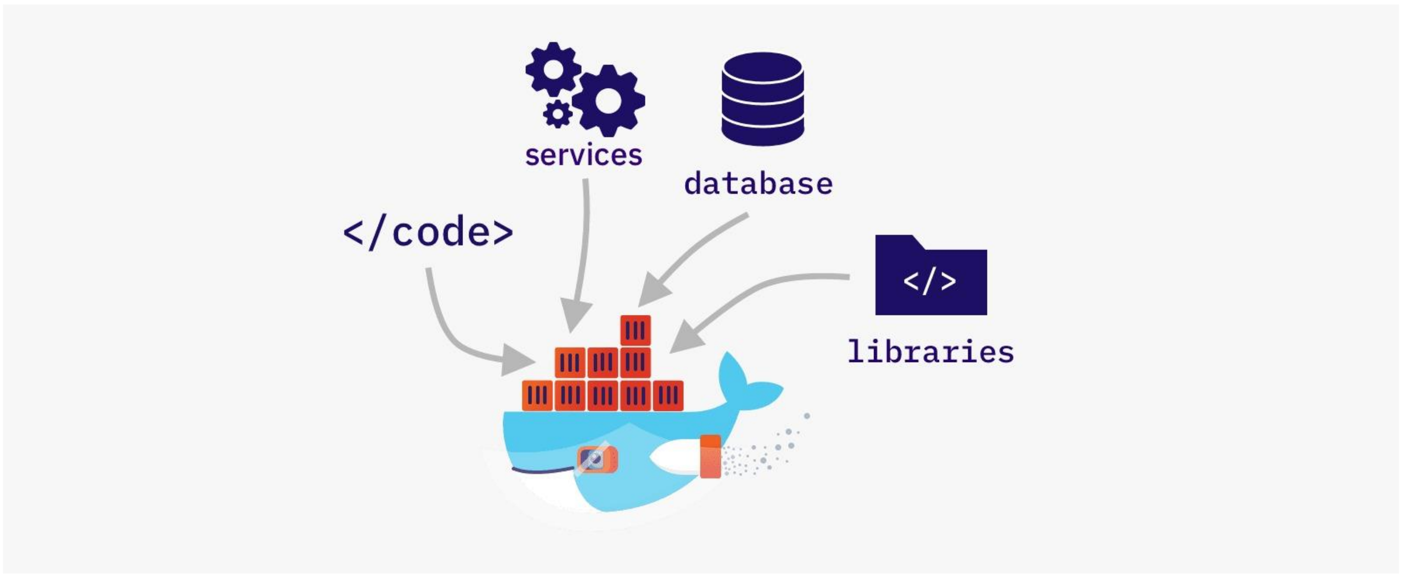
- **Azure Active Directory (AAD):** Identity and access management service.
- **Azure Key Vault:** Safeguard cryptographic keys and secrets.
- **Azure Security Center:** Unified security management and threat protection.
- **Azure DDoS Protection:** Protect your applications from distributed denial-of-service (DDoS) attacks.

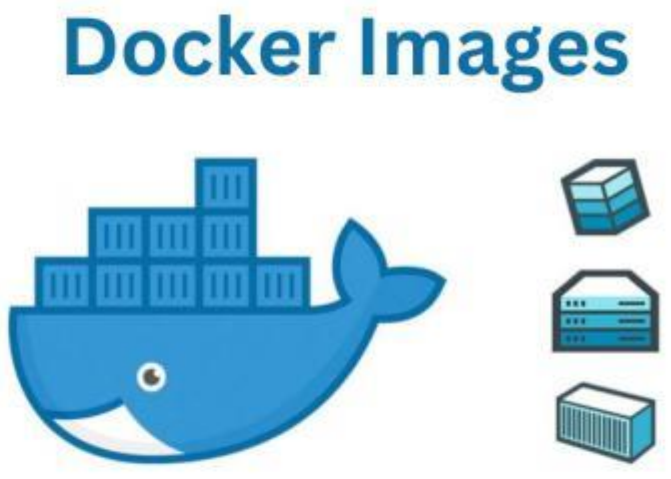
5. Management and Governance

- **Azure Policy:** Enforce organizational standards and assess compliance.
- **Azure Monitor:** Full observability into your applications, infrastructure, and network.
- **Azure Cost Management:** Optimize cloud spend and maximize cloud investment.

4. Docker

Docker is a popular platform for developing, shipping, and running applications inside containers. It enables developers to package applications with all their dependencies into a standardized unit called a container. Here's a comprehensive overview of Docker:





Key Components

1. **Docker Engine**: The core part of Docker that creates and runs containers.

  - **Docker Daemon**: A background service that manages Docker images, containers, networks, and storage volumes.

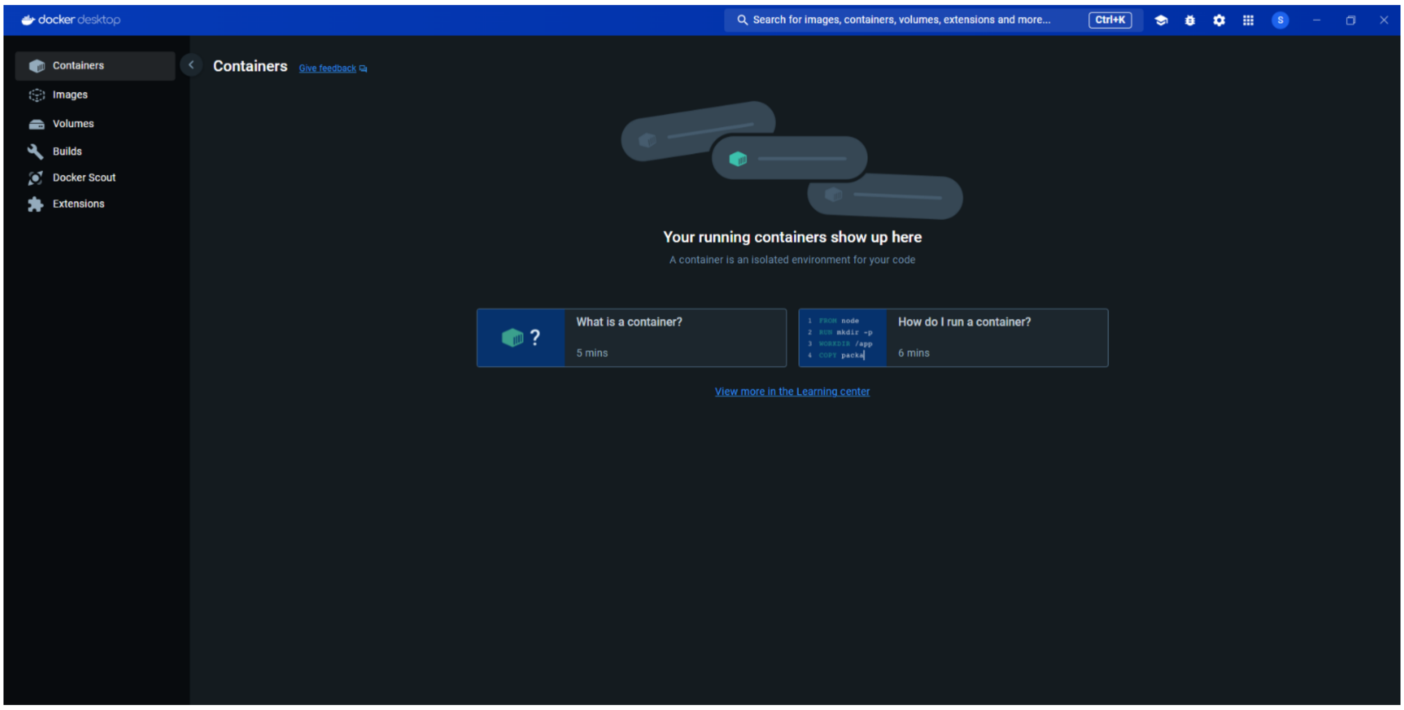
  - **Docker CLI**: A command-line interface that allows users to interact with the Docker daemon.

2. **Docker Images**: Immutable, read-only files containing the application code, runtime, libraries, environment variables, and configuration files required to run an application.

3. **Docker Containers**: Instances of Docker images that run in isolated environments. Containers share the host OS kernel but operate independently.

4. **Dockerfile**: A text file containing instructions on how to build a Docker image. It defines the base image, application code, dependencies, and commands to run the application.

5**. Docker Hub**: A public repository for storing and sharing Docker images. Users can pull pre-built images or push their own images to Docker Hub.


docker desktop
Search for images, containers, volumes, extensions and more...
Ctrl+K
Containers
Images
Volumes
Builds
Docker Scout
Extensions
Containers
Give feedback
Your running containers show up here
A container is an isolated environment for your code
What is a container?
5 mins
How do I run a container?
6 mins
View more in the Learning center

```
FROM node
RUN mkdir -p
WORKDIR /app
COPY packa
```

Basic Docker Commands

1**. docker run**: Create and start a new container from an image.

    docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

2. **docker build**: Build an image from a Dockerfile.

    docker build [OPTIONS] PATH | URL | -

3**. docker pull**: Download an image from a registry (e.g., Docker Hub).

    docker pull IMAGE

4. **docker push**: Upload an image to a registry.

    docker push IMAGE

5. **docker ps**: List running containers.

    docker ps

6**. docker images**: List available images.

```
docker images
```

7. **docker stop**: Stop a running container.

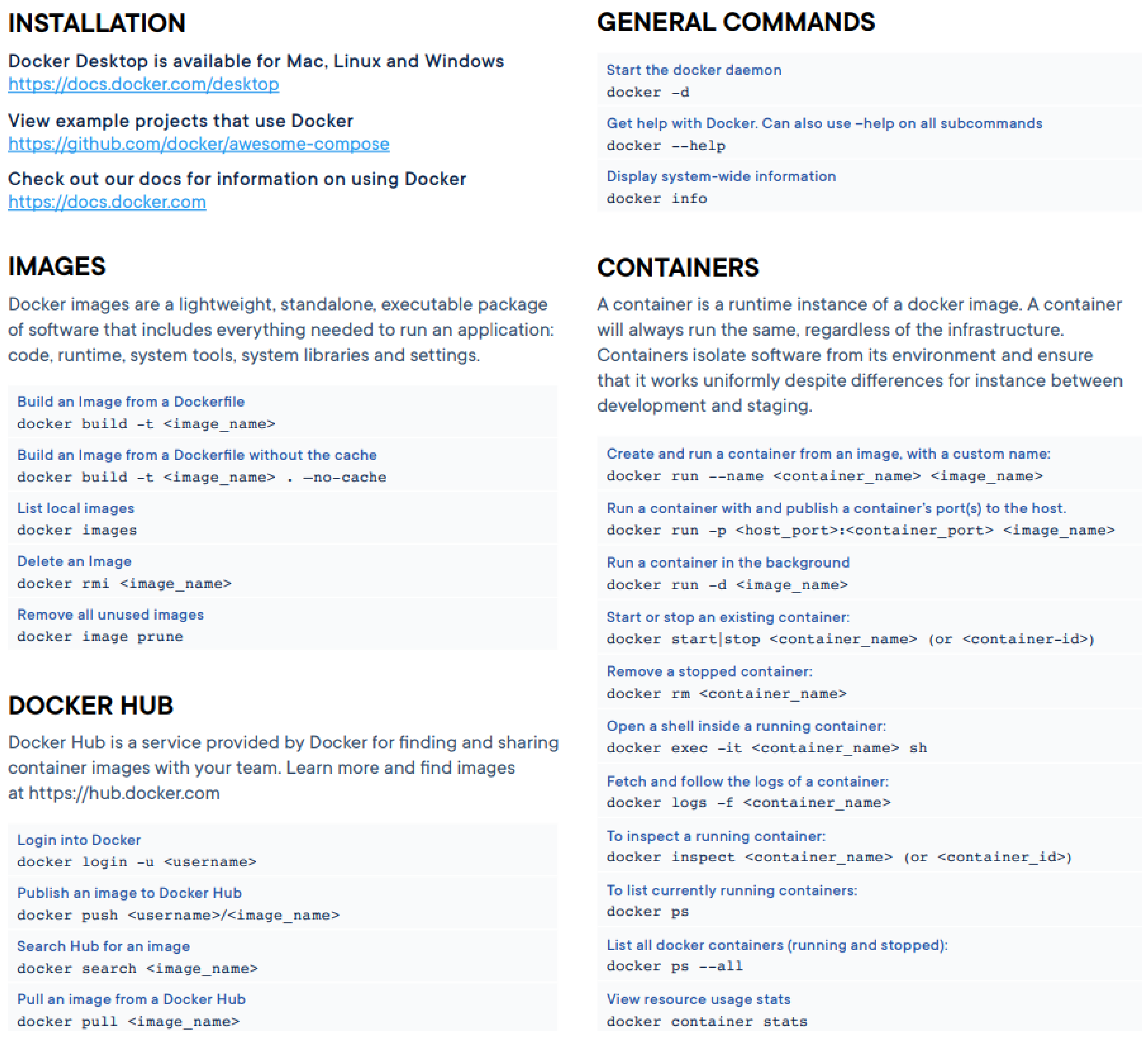
```
docker stop CONTAINER
```

8. **docker rm**: Remove a stopped container.

```
docker rm CONTAINER
```

9. **docker rmi**: Remove an image.

```
docker rmi IMAGE
```



INSTALLATION

Docker Desktop is available for Mac, Linux and Windows
https://docs.docker.com/desktop

View example projects that use Docker
https://github.com/docker/awesome-compose

Check out our docs for information on using Docker
https://docs.docker.com

IMAGES

Docker images are a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Build an Image from a Dockerfile
```
docker build -t <image_name>
```
Build an Image from a Dockerfile without the cache
```
docker build -t <image_name> . –no-cache
```
List local images
```
docker images
```
Delete an Image
```
docker rmi <image_name>
```
Remove all unused images
```
docker image prune
```

DOCKER HUB

Docker Hub is a service provided by Docker for finding and sharing container images with your team. Learn more and find images at https://hub.docker.com

Login into Docker
```
docker login -u <username>
```
Publish an image to Docker Hub
```
docker push <username>/<image_name>
```
Search Hub for an image
```
docker search <image_name>
```
Pull an image from a Docker Hub
```
docker pull <image_name>
```

GENERAL COMMANDS

Start the docker daemon
```
docker -d
```
Get help with Docker. Can also use –help on all subcommands
```
docker --help
```
Display system-wide information
```
docker info
```

CONTAINERS

A container is a runtime instance of a docker image. A container will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

Create and run a container from an image, with a custom name:
```
docker run --name <container_name> <image_name>
```
Run a container with and publish a container's port(s) to the host.
```
docker run -p <host_port>:<container_port> <image_name>
```
Run a container in the background
```
docker run -d <image_name>
```
Start or stop an existing container:
```
docker start|stop <container_name> (or <container-id>)
```
Remove a stopped container:
```
docker rm <container_name>
```
Open a shell inside a running container:
```
docker exec -it <container_name> sh
```
Fetch and follow the logs of a container:
```
docker logs -f <container_name>
```
To inspect a running container:
```
docker inspect <container_name> (or <container_id>)
```
To list currently running containers:
```
docker ps
```
List all docker containers (running and stopped):
```
docker ps --all
```
View resource usage stats
```
docker container stats
```

Docker installation process

Docker is a platform for developing, shipping, and running applications inside containers. Here's a step-by-step guide to install Docker on different operating systems.

Install Docker on Ubuntu

I. **Update Package Index**:

    a. sudo apt-get update

II. **Install Prerequisites**:
    a. sudo apt-get install apt-transport-https ca-certificates curl software-properties-common

III. **Add Docker's Official GPG Key**:
    a. curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

IV. **Set Up the Stable Repository**:
    a. sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

V. **Update the Package Index Again**:
    a. sudo apt-get update

VI. **Install Docker**:
    a. sudo apt-get install docker-ce

VII. **Verify Docker Installation**:
    a. sudo systemctl status docker
    b. sudo docker --version

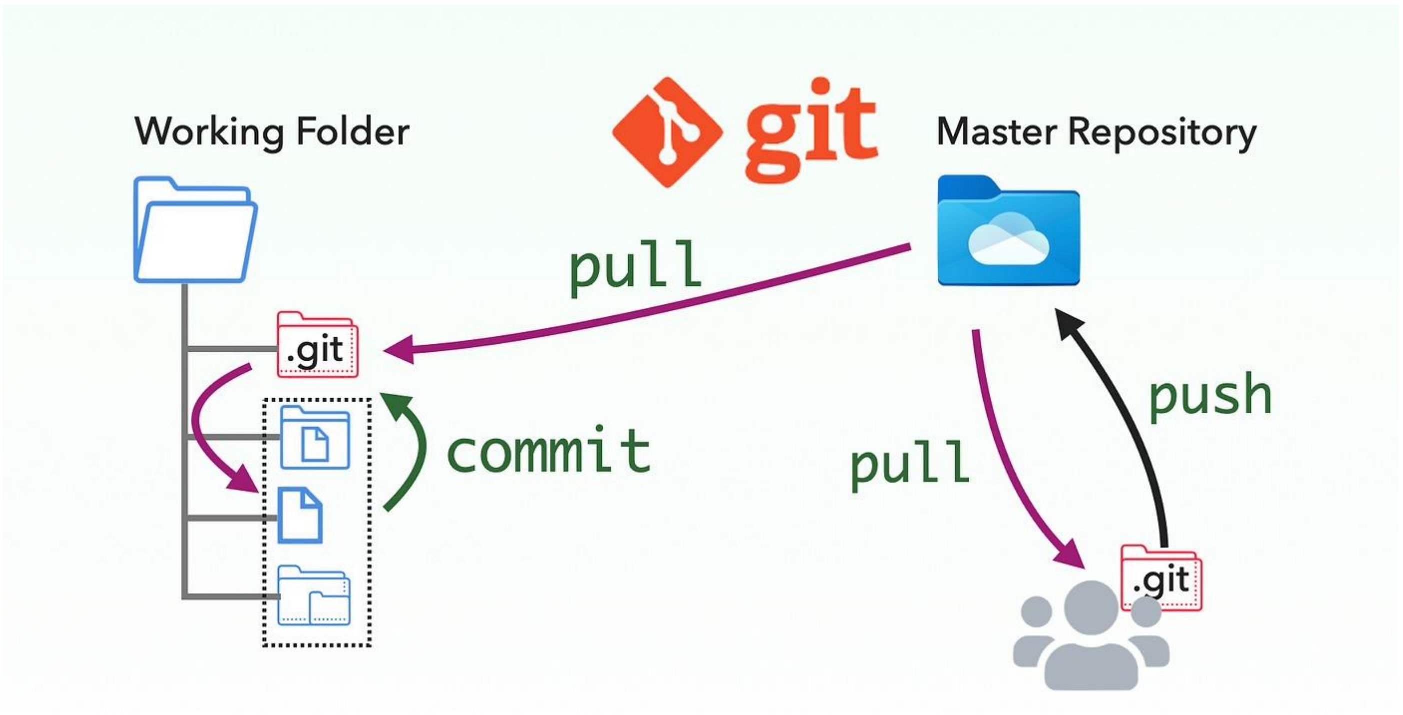
5. <u>Computing engine and Ubuntu virtual machines</u>

Using computing engines with Ubuntu virtual machines is a common approach for deploying scalable and flexible computing resources. Here's how Microsoft azure cloud providers facilitate this:

Microsoft Azure:

- **Overview**: Azure Virtual Machines provides on-demand, scalable computing resources.
- **Ubuntu Support**: Azure offers a variety of Ubuntu images that can be deployed as virtual machines.
- **How to Use**:
    1. **Create a VM**: Use the Azure Portal, CLI, or API to create a new VM.
    2. **Select an Image**: Choose an Ubuntu image from the Azure Marketplace.
    3. **Configure VM**: Set up the VM size, storage options, networking, and other settings.
    4. **Deploy and Connect**: Deploy the VM and connect via SSH for further configuration.

6. <u>GitHub Repository</u>



GitHub is a web-based platform used for version control and collaborative software development. It hosts Git repositories, where developers can store and manage their code. Here's how to create and manage a GitHub repository, along with some common practices:

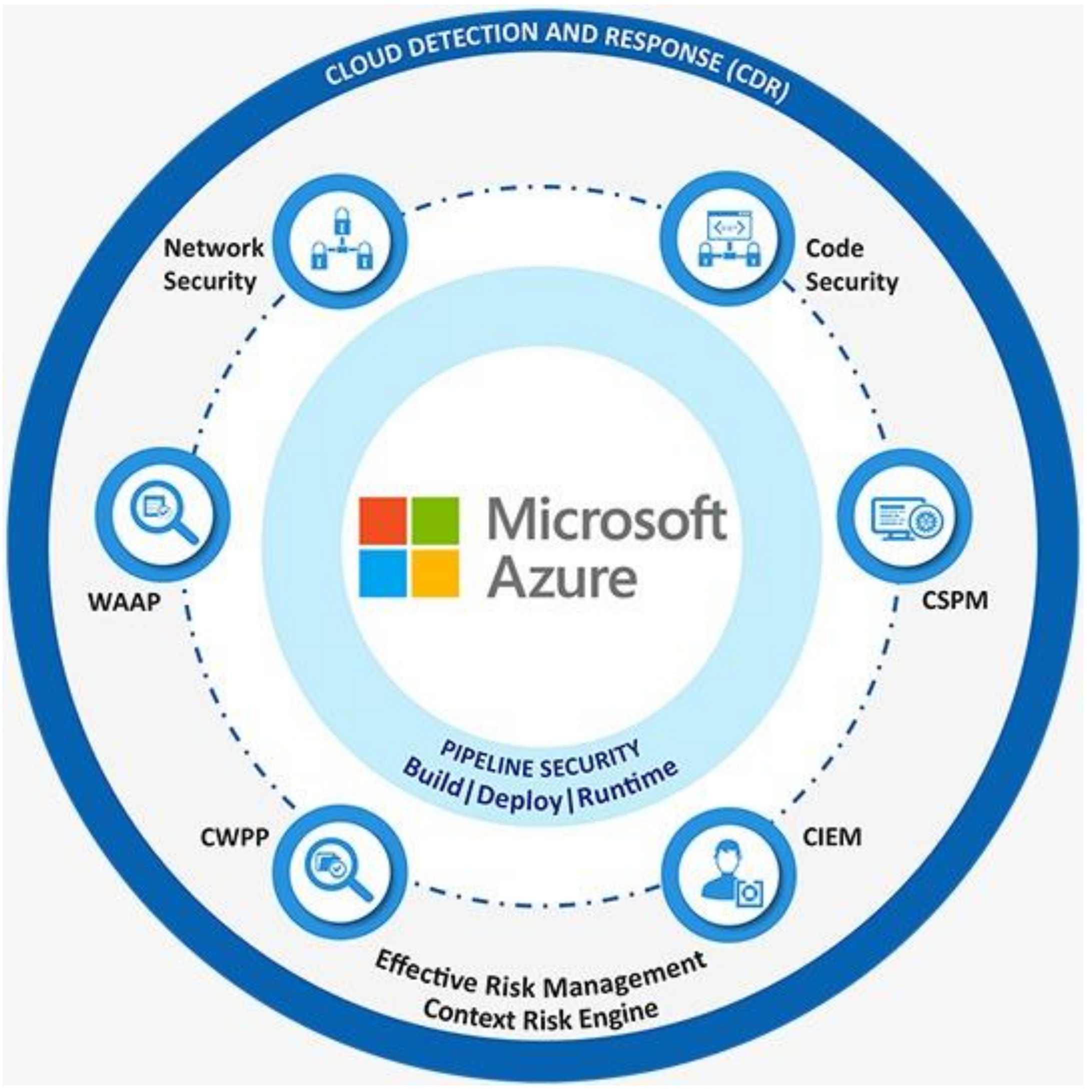
Creating a GitHub Repository

1. **Sign Up/Log In**:
   - Go to GitHub and sign up or log in to your account.
2. **Create a New Repository**:
   - Click the "+" icon in the upper-right corner and select "New repository".
   - Fill in the repository name, description (optional), and choose its visibility (public or private).
   - Optionally, you can initialize the repository with a README file, gitignore, and a license.
   - Click "Create repository".

7. Virtual machine security group

A security group acts as a virtual firewall for your virtual machine (VM) instances to control inbound and outbound traffic. Security groups are used to secure VMs by defining rules that specify the allowed traffic to and from the instances.

Microsoft Azure:



1. **Create a Network Security Group (NSG)**:
    - Go to the Azure portal at https://portal.azure.com/.
    - In the left-hand menu, select "Create a resource".
    - Search for "Network Security Group" and select it.
    - Click "Create".
    - Enter the required details, such as name, subscription, resource group, and region.
    - Click "Review + create" and then "Create".
2. **Configure Inbound Security Rules**:
    - Navigate to the NSG you created.
    - Select "Inbound security rules".
    - Click "Add".
    - Define the rule by specifying the source, source port ranges, destination, destination port ranges, protocol, and action.
    - Click "Add".
3. **Configure Outbound Security Rules**:
    - Select "Outbound security rules".
    - Click "Add".
    - Define the rule by specifying the destination, destination port ranges, protocol, and action.
    - Click "Add".
4. **Associate NSG with a VM**:
    - Navigate to the VM you want to secure.
    - In the VM's menu, select "Networking".
    - Under "Network interfaces", select the network interface.
    - Under "Settings", select "Network security group".
    - Select the NSG you created and click "Save".

References

- https://www.geeksforgeeks.org/introduction-to-github/
- https://aws.amazon.com/docker/
- https://www.geeksforgeeks.org/beginners-guide-to-using-ubuntu-on-google-cloud-platform-gcp/#:~:text=Step%203%3A%20Create%20a%20Virtual,the%20option%20%E2%80%9CCreate%20Instance%E2%80%9D.
- https://docs.docker.com/desktop/install/windows-install/
- https://www.google.com/search?sca_esv=6649c81aa00f7f94&sca_upv=1&rlz=1C1UEAD_enIN997IN997&udm=2&sxsrf=ADLYWILGEkkDah3gm9bTalgTGZhUol8cZA:1722575626282&q=git&spell=1&sa=X&ved=2ahUKEwjUoraixtWHAxV82DQHHSxHN0oQBSgAegQIBxAB&biw=1536&bih=776&dpr=1.25#vhid=0pt1uI_gJecMJM&vssid=mosaic
- https://education.github.com/git-cheat-sheet-education.pdf