

Technical Documentation: Docker Installation on AWS

Table of Contents

1. Introduction
2. Getting Started with Docker
 - a. Benefits of Docker
 - b. Docker Components
3. Installation Steps
 - a. Pre-installation Steps
 - b. Installing Docker Engine
 - c. Verifying Docker Installation
4. Resources

1. Introduction

Docker, a ground-breaking tool that is revolutionizing the development, shipping, and operation of applications. Regardless of your level of experience with containerization, Docker provides unmatched efficiency and flexibility for software deployment. We'll go over the basics of Docker, covering everything from what containers are to how to create and manage them practically down to the installation process. Let's get started!

2. Getting Started with Docker

a. Benefits of Docker

Docker offers many advantages. Some key advantages include portability, high isolation, resource efficiency, scalability, and fast development and deployment.

- **Portability:** Docker containers encapsulate applications and their dependencies which makes it easier to seamlessly migrate applications between environments. Regardless of where you are working, the docker container will consistently run the same way.
- **Isolation:** Docker containers offer a high level of isolation between applications and their dependencies. This means that since each container runs independently, it will not interfere with other containers on the same machine or server.
- **Resource efficiency:** Containers share the underlying OS kernel, which makes them more efficient than traditional virtual machines. This can lead to a more efficient use of resources, and for cloud computing, cost savings because you can run more containers on the same existing hardware.
- **Scalability:** Docker makes applications scalable with docker images which include everything required to run and even deploy an application. Docker images can be pulled and run on containers across different systems and devices, increasing the scalability of the applications run and developed via Docker.
- **Fast Development and Deployment:** Containers can help improve the speed and reliability of deployments, whether on traditional infrastructure or via the cloud. By packaging applications into containers, developers can be confident that their applications will run consistently on any cloud platform and on any different system. This can help to reduce the time it takes to deploy new applications and to minimize the risk of problems.

b. Docker Components

To understand Docker, we must first understand some of its key components. Docker uses a client-server architecture where the Docker client communicates with the Docker daemon.

- **Docker Client:** The client is the first component of Docker. It allows the users to communicate with Docker. For example, when a user gives a command to docker, the docker client sends the desired command to the host, which further fulfills the command using Docker API. The client can interact with multiple hosts.
- **Docker Image:** Docker images are used to build containers. These images are read-only binary templates in YAML.

- **Docker Daemon:** Docker Daemon is one of the most essential components of Docker as it is directly responsible for fulfilling the actions related to containers. The Docker daemon mainly runs as a background process responsible for creating, running and distributing containers.
- **Docker Networking:** Docker networking helps in establishing communication between containers.
- **Docker Registry:** Docker images require a location where they can be stored, and the Docker registry is that location. Docker Hub is the default storage location of images that stores the public registry.
- **Docker Container:** A Docker container is the instance of an image that can be created, started, moved, or deleted through a Docker API. Containers are a lightweight and independent method of running applications. They can be connected to one or more networks and create a new image depending on the current state.

3. Installation Steps

a. Pre-installation Steps

Login to your AWS Console



Sign in

☒ **Root user**

Account owner that performs tasks requiring unrestricted access. [Learn more](#)

☐ **IAM user**

User within an account that performs daily tasks. [Learn more](#)

Root user email address

[Redacted email address]

Next

Here I am logging in as a Root user. It is not recommended to regularly login as root user as it can pose security risks. IAM user login suggested.



Multi-factor authentication

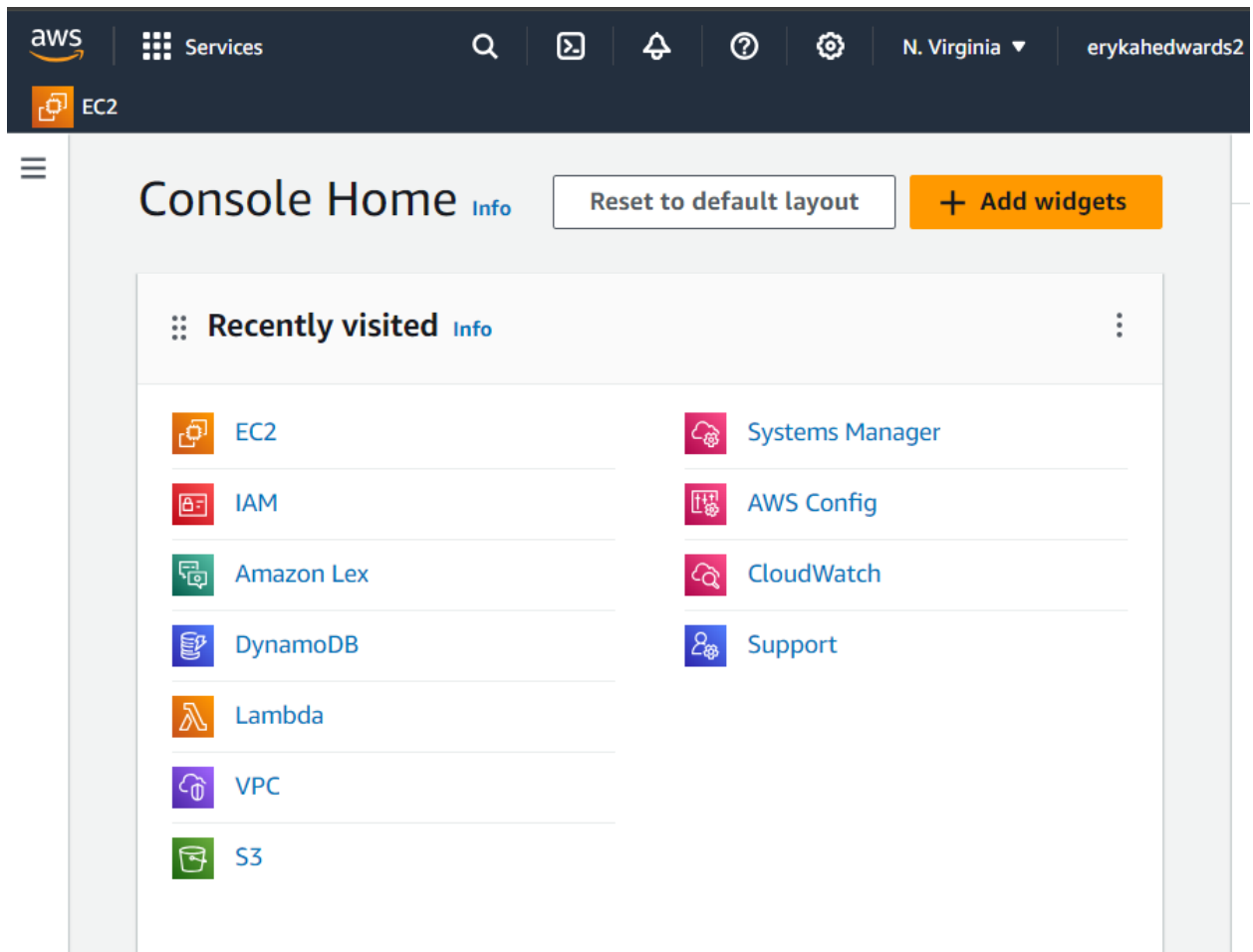
Your account is secured using multi-factor authentication (MFA). To finish signing in, turn on or view your MFA device and type the authentication code below.

Email address:

[REDACTED]

MFA code

Here I am using my authenticator app code as a method of MFA.



Now, we are at the Console home page. Let's create an EC2 instance. Search and Select EC2.

EC2

EC2 Dashboard

EC2 Global View

Events

Console-to-Code

Preview

Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Instances

Info

Refresh

Connect

Instance state

Actions

Launch instances

Find Instance by attribute or tag (case-sensitive)

Any state

< 1 >

Name

Instance ID

Instance state

No instances

You do not have any instances in this region

Launch instances

< 1 >

Select Instances and then Launch Instance.

Launch an instance

Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Info

Name

Docker-ec2

Add additional tags

Give your instance a name


▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below


 Search our full catalog including 1000s of application and OS images

Quick Start


Amazon
Linux




macOS




Ubuntu



Window





[Browse more AMIs](#)

Including AMIs from
AWS, Marketplace and
the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type Free tier eligible
ami-07d9b9ddc6cd8dd30 (64-bit (x86)) / ami-0568072f574d822a4 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

We're going to select the Ubuntu Amazon Machine Image.

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro

Free tier eligible

Family: t2 1 vCPU 1 GiB Memory

Current generation: true

On-Demand Windows base pricing: 0.0162 USD per Hour

On-Demand SUSE base pricing: 0.0116 USD per Hour

On-Demand RHEL base pricing: 0.0716 USD per Hour

On-Demand Linux base pricing: 0.0116 USD per Hour

☐ All generations

[Compare instance types](#)

[Additional costs apply for AMIs with pre-installed software](#)


For instance type, we are going to keep t2.micro

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

Select

 [Create new key pair](#)

We use the Key pair to securely login to our ec2 instances. You can create or import a key pair that allows us to securely connect to the instance via SSH. Select generate a new key pair. The key pair consists of a public key that you place on the instance and a private key that you keep safe and secure.

Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type


☒ RSA
RSA encrypted private and public key pair

☐ ED25519
ED25519 encrypted private and public key pair

Private key file format

☒ .pem
For use with OpenSSH

☐ .ppk
For use with PuTTY

 When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** [Learn](#)

Cancel

Create key pair

Create a name for the Key pair. RSA Key pair type and private key file format leave as .pem and click create new key pair. A .pem file should show in your downloads. We will need to open the file later on.

▼ Network settings [Info](#)

VPC - *required* [Info](#)

vpc-0c8b101705441d97c
172.31.0.0/16

(default) ▼



Subnet [Info](#)

subnet-0747cfb59a9e63ba1

VPC: vpc-0c8b101705441d97c Owner: 992382453401
Availability Zone: us-east-1a IP addresses available: 4091
CIDR: 172.31.16.0/20



[Create new subnet](#)

Auto-assign public IP [Info](#)

Enable ▼

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group

☐ Select existing security group

Security group name - *required*

sg-dockerec2

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0)

Remove

Type [Info](#)

ssh ▼

Protocol [Info](#)

TCP

Port range [Info](#)

22

Source type [Info](#)

Anywhere ▼

Source [Info](#)

🔍 Add CIDR, prefix list or security group

0.0.0.0/0 ✕

Description - optional [Info](#)

e.g. SSH for admin desktop


Security groups should be enabled so you can SSH from your local machine. Here we are selecting an existing security group (launch-wizard-1). This security group type is SSH, protocol tcp and is using Port range 22.

▼ Configure storage [Info](#)

[Advanced](#)

1x GiB ▼ Root volume


(Not encrypted)

 Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage



Add new volume

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

 Click refresh to view backup information



The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

0 x File systems

[Edit](#)

► **Advanced details** [Info](#)

We will leave the storage settings and advanced details as is.

▼ Summary

Number of instances [Info](#)

1

Software Image (AMI)

Canonical, Ubuntu, 22.04 LTS, ...[read more](#)
ami-07d9b9ddc6cd8dd30

Virtual server type (instance type)

t2.micro

Firewall (security group)

launch-wizard-1

Storage (volumes)

1 volume(s) - 8 GiB

Cancel

Launch instance

[Review commands](#)

Click on Launch instance.

[EC2](#) > [Instances](#) > Launch an instance

✓ **Success**
Successfully initiated launch of instance ([i-043735bc33116b793](#))

Instances (1) Info				
	Connect	Instance state ▼	Actions ▼	Launch instances ▼
<input type="text" value="Find Instance by attribute or tag (case-sensitive)"/>				
Any state ▼				
< 1 >				
Name	Instance ID	Instance state ▼	Instance type ▼	Status check
Docker-ec2	i-043735bc33116b793	✓ Running	t2.micro	✓ 2/2 checks

The instance has been successfully created and is running and we can now SSH to our instance from our local machines. Click on Connect.



Connect to instance [Info](#)

Connect to your instance i-043735bc33116b793 (Docker-ec2) using any of these options


< | **EC2 Instance Connect** | **Session Manager** | **SSH client** | EC2 sel | >

Instance ID

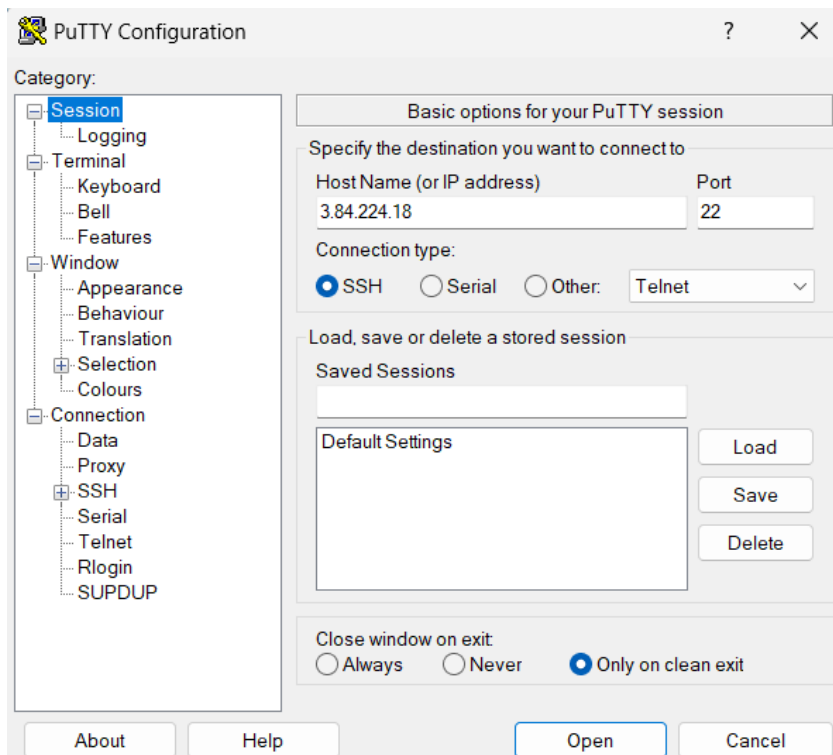
 **i-043735bc33116b793** (Docker-ec2)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is docker-ec2.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
 `chmod 400 "docker-ec2.pem"`
4. Connect to your instance using its Public DNS:
 `ec2-54-80-102-17.compute-1.amazonaws.com`

Example:

 `ssh -i "docker-ec2.pem" ubuntu@ec2-54-80-102-17.compute-1.amazonaws.com`

We will use the example command that displays “docker-ec2.pem” which is the name of your key pair file that we created and downloaded onto our local host. The rest of the command shows the name of the user@the publicDNS.



We are going to use Putty as our SSH client. Input your instance IP address. Use this link to install putty onto your host machine if you do not already have it installed
<https://www.puttygen.com/download.php?val=4>

PutTY Key Generator

File Key Conversions Help

Key

No key.

Actions

Generate a public/private key pair Generate

Load an existing private key file Load

Save the generated key Save public key Save private key

Parameters

Type of key to generate:

☒ RSA ☐ DSA ☐ ECDSA ☐ EdDSA ☐ SSH-1 (RSA)

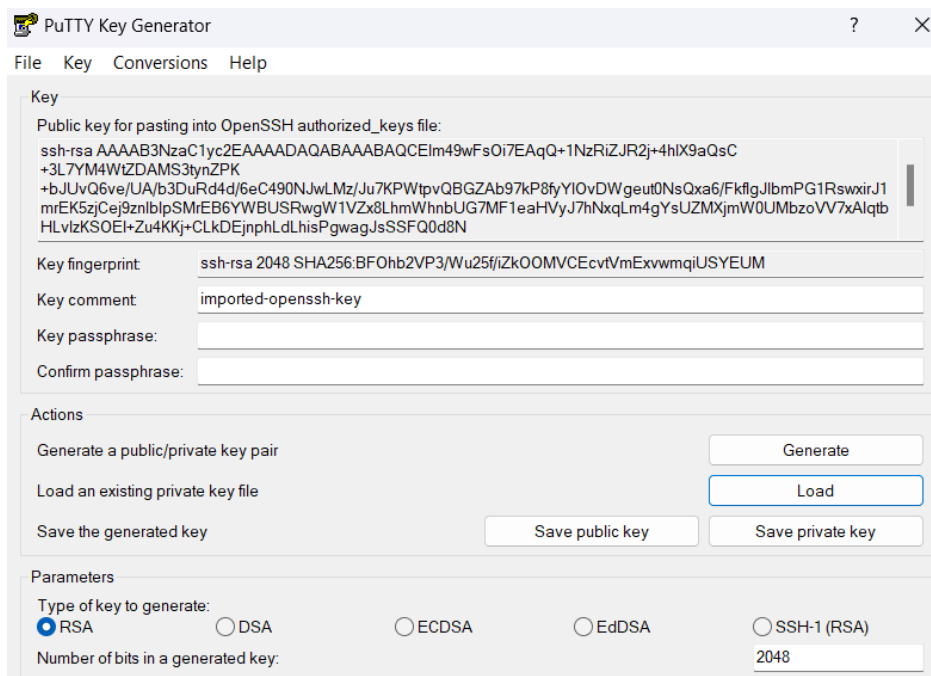
Number of bits in a generated key:

Open the Putty key generator and load your .pem file to convert it into a .ppk file for Putty

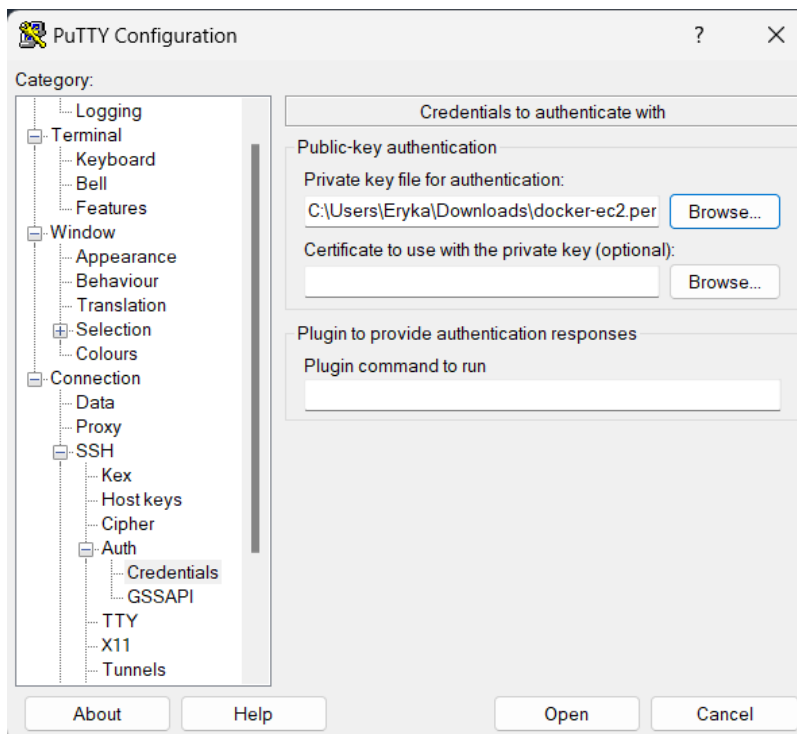
PutTYgen Notice

Successfully imported foreign key
(OpenSSH SSH-2 private key (old PEM format)).
To use this key with PuTTY, you need to
use the "Save private key" command to
save it in PuTTY's own format.

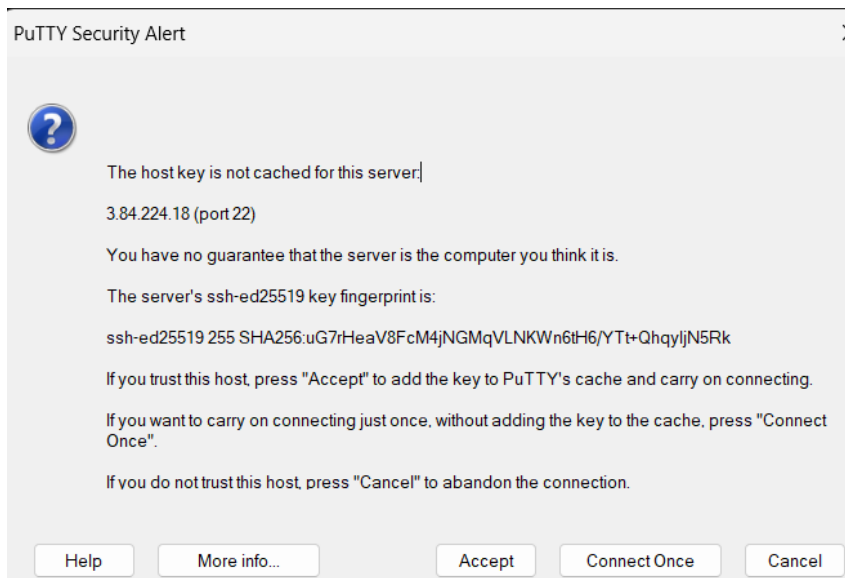
OK



Save the key onto your host machine. You should see the file now show as a .ppk file which can now be uploaded onto Putty



Navigate to SSH ---> Auth ---> Credentials. Then browse for the key pair converted .pem file that was downloaded onto your host machine.



Click Accept

```
ubuntu@ip-172-31-9-149: ~  
login as: ubuntu  
Authenticating with public key "imported-openssh-key"  
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1018-aws x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/pro  
  
System information as of Thu Mar  7 03:22:28 UTC 2024  
  
System load:  0.0                Processes:            95  
Usage of /:   20.3% of 7.57GB     Users logged in:     0  
Memory usage: 20%                IPv4 address for eth0: 172.31.  
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 list of available updates is more than a week old.
To check for new updates run: `sudo apt update`

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.

```
ubuntu@ip-172-31-9-149:~$
```

```
ubuntu@ip-172-31-9-149:~$
```

```
ubuntu@ip-172-31-9-149:~$ pwd  
/home/ubuntu
```

```
ubuntu@ip-172-31-9-149: ~  
Get:31 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [1495 kB]  
Get:32 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [248 kB]  
Get:33 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [847 kB]  
Get:34 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [161 kB]  
Get:35 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [16.8 kB]  
Get:36 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [37.1 kB]  
Get:37 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [7476 B]  
Get:38 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [260 B]  
Fetched 29.9 MB in 5s (5491 kB/s)  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
36 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

b.

```

ubuntu@ip-172-31-9-149: ~
Processing triggers for initramfs-tools (0.140ubuntu13.4) ...
update-initramfs: Generating /boot/initrd.img-6.5.0-1014-aws
Scanning processes...
Scanning candidates...
Scanning linux images...

Restarting services...
/etc/needrestart/restart.d/systemd-manager
systemctl restart packagekit.service ssh.service systemd-journald.service systemd-networkd.service systemd-resolved.service systemd-udev.service
Service restarts being deferred:
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service
systemctl restart user@1000.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-9-149:~$

```

After logging in to the Ubuntu virtual machine, open a terminal window and run the following commands to update the package lists and upgrade existing packages:

```

sudo apt update
sudo apt upgrade

```

```

ubuntu@ip-172-31-9-149:~$ 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 (20230311ubuntu0.22.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.81.0-1ubuntu1.15).
curl set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ubuntu@ip-172-31-9-149:~$

```

```

ubuntu@ip-172-31-9-149:~$ sudo apt install docker.io

```

Run the command “sudo apt install docker.io” in order to install docker.

```

ubuntu@ip-172-31-9-149:~$ sudo docker --version
Docker version 24.0.5, build 24.0.5-0ubuntu1~22.04.1
ubuntu@ip-172-31-9-149:~$

```

Run the command “sudo docker --version” to display the version of docker installed.

```
ubuntu@ip-172-31-9-149:~$ sudo service docker start
ubuntu@ip-172-31-9-149:~$
```

```
ubuntu@ip-172-31-9-149: ~
ubuntu@ip-172-31-9-149:~$ systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enable>
   Active: active (running) since Thu 2024-03-07 04:39:38 UTC>
 TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Main PID: 13741 (dockerd)
     Tasks: 7
    Memory: 32.0M
      CPU: 344ms
   CGroup: /system.slice/docker.service
           └─13741 /usr/bin/dockerd -H fd:// --containerd=/ru>

Mar 07 04:39:37 ip-172-31-9-149 systemd[1]: Starting Docker App>
Mar 07 04:39:37 ip-172-31-9-149 dockerd[13741]: time="2024-03-0>
Mar 07 04:39:37 ip-172-31-9-149 dockerd[13741]: time="2024-03-0>
Mar 07 04:39:37 ip-172-31-9-149 dockerd[13741]: time="2024-03-0>
Mar 07 04:39:37 ip-172-31-9-149 dockerd[13741]: time="2024-03-0>
Mar 07 04:39:38 ip-172-31-9-149 dockerd[13741]: time="2024-03-0>
Mar 07 04:39:38 ip-172-31-9-149 dockerd[13741]: time="2024-03-0>
Mar 07 04:39:38 ip-172-31-9-149 dockerd[13741]: time="2024-03-0>
Mar 07 04:39:38 ip-172-31-9-149 dockerd[13741]: time="2024-03-0>
Mar 07 04:39:38 ip-172-31-9-149 systemd[1]: Started Docker Appl>
lines 1-21/21 (END)
```

To start docker we will run the command “sudo service docker start”. Then run “systemctl status docker”. If Docker has started successfully, you should see output indicating that the service is active and running.

```
ubuntu@ip-172-31-9-149:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d000bc569937abbe195e20322a0bde6b2922d805332fd6d8a
68b19f524b7d21d
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 Do
```

This command also shows that docker is successfully running.

```
ubuntu@ip-172-31-9-149:~$ sudo docker pull centos
Using default tag: latest
latest: Pulling from library/centos
ald0c7532777: Pull complete
Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Downloaded newer image for centos:latest
docker.io/library/centos:latest
```

We have successfully pulled the centos image.

```
ubuntu@ip-172-31-9-149:~$ sudo docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
bccd10f490ab: Pull complete
Digest: sha256:77906da86b60585ce12215807090eb327e7386c8fafb5402369e421f44eff17e
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

We have successfully pulled the Ubuntu image.