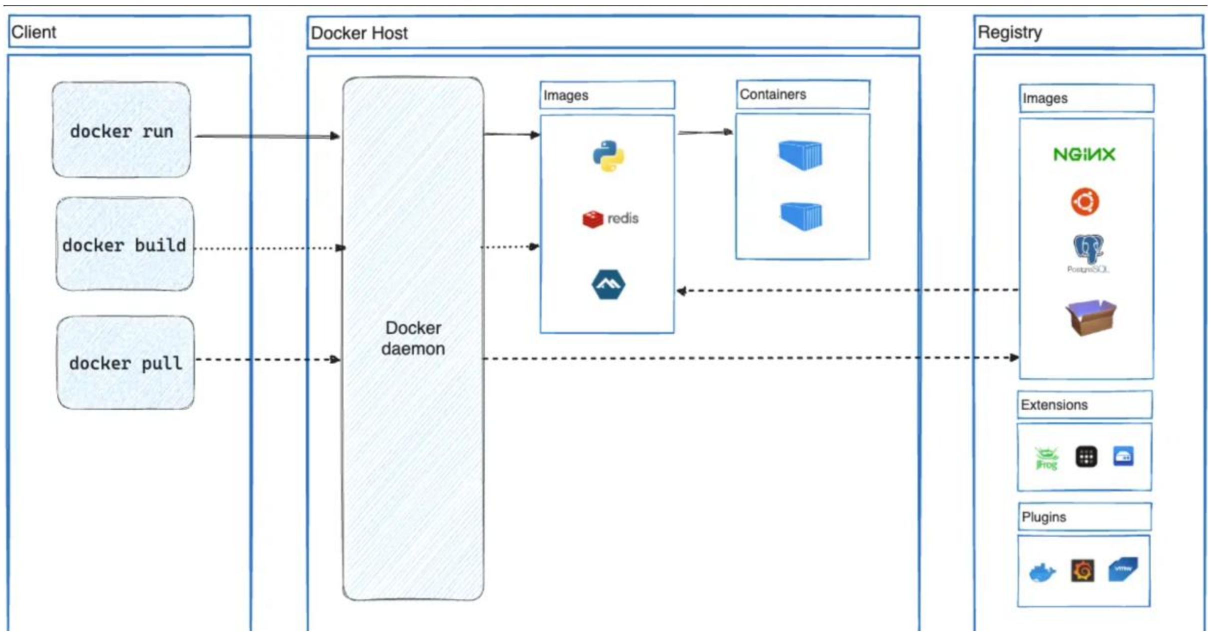
1. Docker Architecture:

Docker follows a **client-server architecture**, which includes the following components:

- **Docker Client:**
  The interface through which users interact with Docker (e.g., CLI commands like docker run, docker build).
- **Docker Daemon (dockerd):**
  The server running in the background that manages Docker objects like images, containers, networks, and volumes.
- **Docker Images:**
  Read-only templates used to create containers. Each image is built from a Dockerfile.
- **Docker Containers:**
  Running instances of Docker images that package the application and its dependencies.
- **Docker Registry:**
  A repository (like Docker Hub) that stores Docker images.
  Users can **pull** images from and **push** images to the registry.



2. **Installation Steps**

# Update packages

sudo apt update

# Install Docker

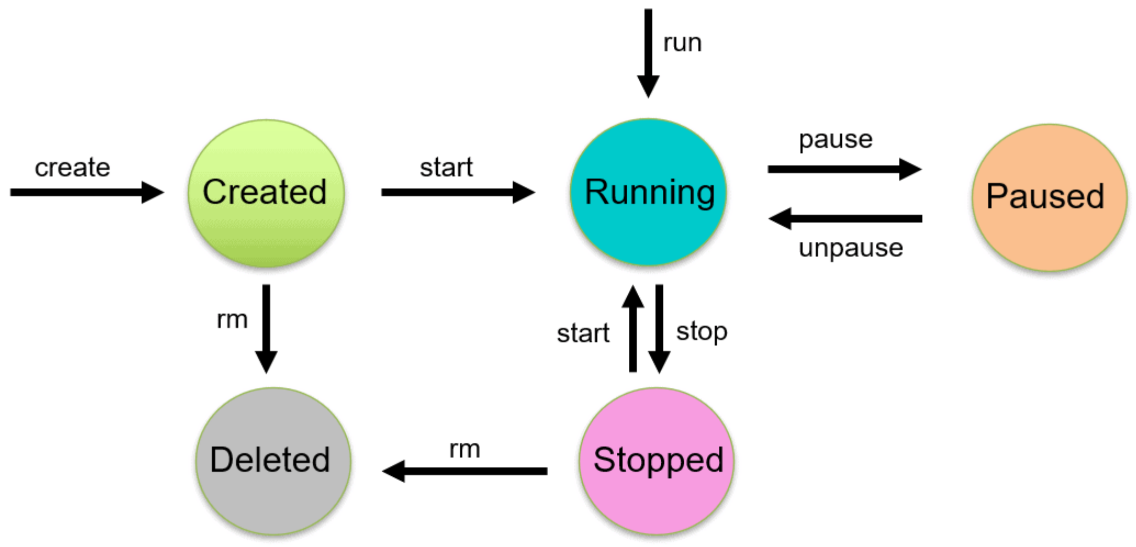
sudo apt install docker.io -y

```
# Start and enable Docker service

sudo systemctl start docker

sudo systemctl enable docker


# Verify installation

docker –version
```

3. Docker Commands to Manage Images and Containers:

```
# Pull an image from Docker Hub

docker pull ubuntu


# List available images

docker images


# Remove an image

docker rmi <image_id>
```

Docker Container Life Cycle:



EXPERIMENT-7

eval $(minikube docker-env -u) if you did 8 and then again want to do 7

```
mkdir k8s-demo-app
cd k8s-demo-app
```

vim app.js

```
const express = require("express");
const app = express();
const port = 3000;

app.get("/", (req, res) => {
  res.send("Hello from Kubernetes App!");
});

app.listen(port, () => {
  console.log(App running on port ${port});
});
```

Vim package.json

```
{
  "name": "k8s-demo-app",
  "version": "1.0.0",
  "main": "app.js",
  "scripts": {
    "start": "node app.js"
  },
  "dependencies": {
    "express": "^4.18.2"
  }
}
```

Vim dockerfile

```
FROM node:18-alpine

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "start"]
```

```
docker build -t k8s-demo-app:v1 --no-cache .

docker run -p 3000:3000 k8:v1

docker login
```

8th

```
cd ~/flask-app

vim app.py

from flask import Flask

app = Flask(__name__)

@app.route('/')

def home():

    return "Hello from Flask App running on Kubernetes! 🚀 "

if __name__ == '__main__':

    app.run(host='0.0.0.0', port=5000)

vim requirements.txt

Flask==2.2.5
```

Vim dockerfile

```
FROM python:3.9-slim

WORKDIR /app
COPY . .

RUN pip install -r requirements.txt

EXPOSE 5000
CMD ["python", "app.py"]
```

```
minikube start

eval $(minikube docker-env)

docker build -t flask-k8s-app:1.0 .
```

vim deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: flask-app
  template:
    metadata:
      labels:
        app: flask-app
    spec:
      containers:
      - name: flask-app
        image: flask-k8s-app:1.0   # Local image in Minikube
        imagePullPolicy: Never     # <--- This line is REQUIRED for local images
        ports:
        - containerPort: 5000
```

Vim service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: flask-service
spec:
  type: NodePort
  selector:
    app: flask-app
  ports:
  - protocol: TCP
    port: 5000
    targetPort: 5000
    nodePort: 31001
```

```
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
```

```
kubectl get pods
```

```
minikube service flask-service
```

9EXPERIMENT

```
wget https://apt.puppet.com/puppet-release-bionic.deb
sudo dpkg -i puppet-release-bionic.deb
sudo apt update
sudo apt install puppetserver -y
sudo systemctl start puppetserver
sudo systemctl enable puppetserver
sudo nano /etc/puppet/puppet.conf
```

put the below on in it

```
[agent]
server = daivik
environment = production
runinterval = 30m
```

ctrl+x , Y, enter

```
# Start Puppet Server
sudo systemctl start puppetserver
sudo systemctl enable puppetserver

# Start Puppet Agent
sudo systemctl start puppet
sudo systemctl enable puppet

sudo puppet agent --test
```

10

```
puppet –version

mkdir -p ~/puppet-demo/modules/webserver/{manifests,lib/puppet/functions/webserver}

nano ~/puppet-demo/modules/webserver/manifests/init.pp

class webserver {
  package { 'apache2':
    ensure => installed,
  }

  service { 'apache2':
    ensure => running,
    enable => true,
  }

  file { '/var/www/html/index.html':
    ensure  => file,
    content => "<h1>Hello from Puppet Webserver!</h1>",
  }

  # Call custom function (Puppet 8)
  $message = webserver::greet()
  notify { $message: }
}

nano ~/puppet-demo/modules/webserver/lib/puppet/functions/webserver/greet.rb

Puppet::Functions.create_function(:'webserver::greet') do
  def greet()
    "Webserver setup done!"
  end
end

nano ~/puppet-demo/site.pp
```

```
include webserver
```

```
sudo /usr/bin/puppet apply /home/bharath/puppet-demo/site.pp --
modulepath=/home/bharath/puppet-demo/modules --debug
```