

# Docker ga kirish

Shukurali Rezamonov

# Reja:

1. Docker nima ?
2. Nima uchun Dockerdan foydalanish kerak ?
3. Dockerning asosiy tushunchalari
4. Docker arxitekturasini
5. Docker ni o'rnatish
6. Docker container bilan ishlash
7. Dockerfile qanday yoziladi ?
8. Docker Hub nima ?

# Docker nima ?

1. Dasturni ishga tushirish, test qilish, deploy qilishda foydalaniladigan open-source platforma
2. Dasturni ishlashi uchun kerak bo'ladigan barcha paketlar, dependensilar ni o'zida jamlagan va isolatsiyalangan muhitda dasturni ishga tushira oladigan kontainerlardan foydalanadi

# Nima uchun Dockerdan foydalanish kerak

1. **Portability** - dasturchilarga dasturni ishga tushirishda kerak bo'ladigan kerakli paketlari bilan birga dependenslarini bitta kontainerda paketlash imkoni
2. **Soddalashtirilgan va Tezlashtirilgan Deployment jarayoni**
3. **Izolyatsiya va Resurslarni boshqarish** - bitta kontainerda qilingan o'zgarishlar yoki paket yangilanishlari boshqa kontainerlarga ta'sir qilmaydi, resource limitni ham cheklash imkoniyati borligi
4. **Microservices Architecture** - har bitta service alohida kontainerda ishlaydi va bu horizontal va vertikal kuchaytirishga juda qo'l keladi.
5. **CI/CD Integratsiyasi**
6. **Oson Debug qilish** - local muhitda production muhitni osonlikcha yaratib test qilish orqali xatolikni topish imkoniyati

```
'/System/Volumes/Preboot/Cryptexes/OSlibgobject-2.0-0' (no such file), '/usr/lib/libgobject-2.0-0' (no such file, not in dyld cache), 'libgobject-2.0-0' (no such file). Additionally, ctypes.util.find_library() did not manage to locate a library called 'libgobject-2.0-0'
```



The error indicates that the Python package `weasyprint` is attempting to use the `libgobject-2.0-0` library, but it cannot find it on your system. This is a common issue when running `weasyprint` on macOS because it depends on GTK libraries, which may not be pre-installed.

## Steps to Resolve the Issue

### 1. Install the Required Libraries

- Install `gobject` and other dependencies using [Homebrew](#):

```
bash
```

[Copy](#) [Edit](#)

```
brew install gobject-introspection cairo pango gdk-pixbuf libffi
```

### 2. Set Environment Variables

- After installing the libraries, set the `PKG_CONFIG_PATH` environment variable so that `weasyprint` can locate the required libraries:

```
bash
```

[Copy](#) [Edit](#)

```
export PKG_CONFIG_PATH="/opt/homebrew/lib/pkgconfig:/usr/local/lib/pkgconfig"
```



# Build, Ship, Run, Any App Anywhere

From Dev



To Ops



Any App



Any OS



Windows



Linux

Anywhere



Physical

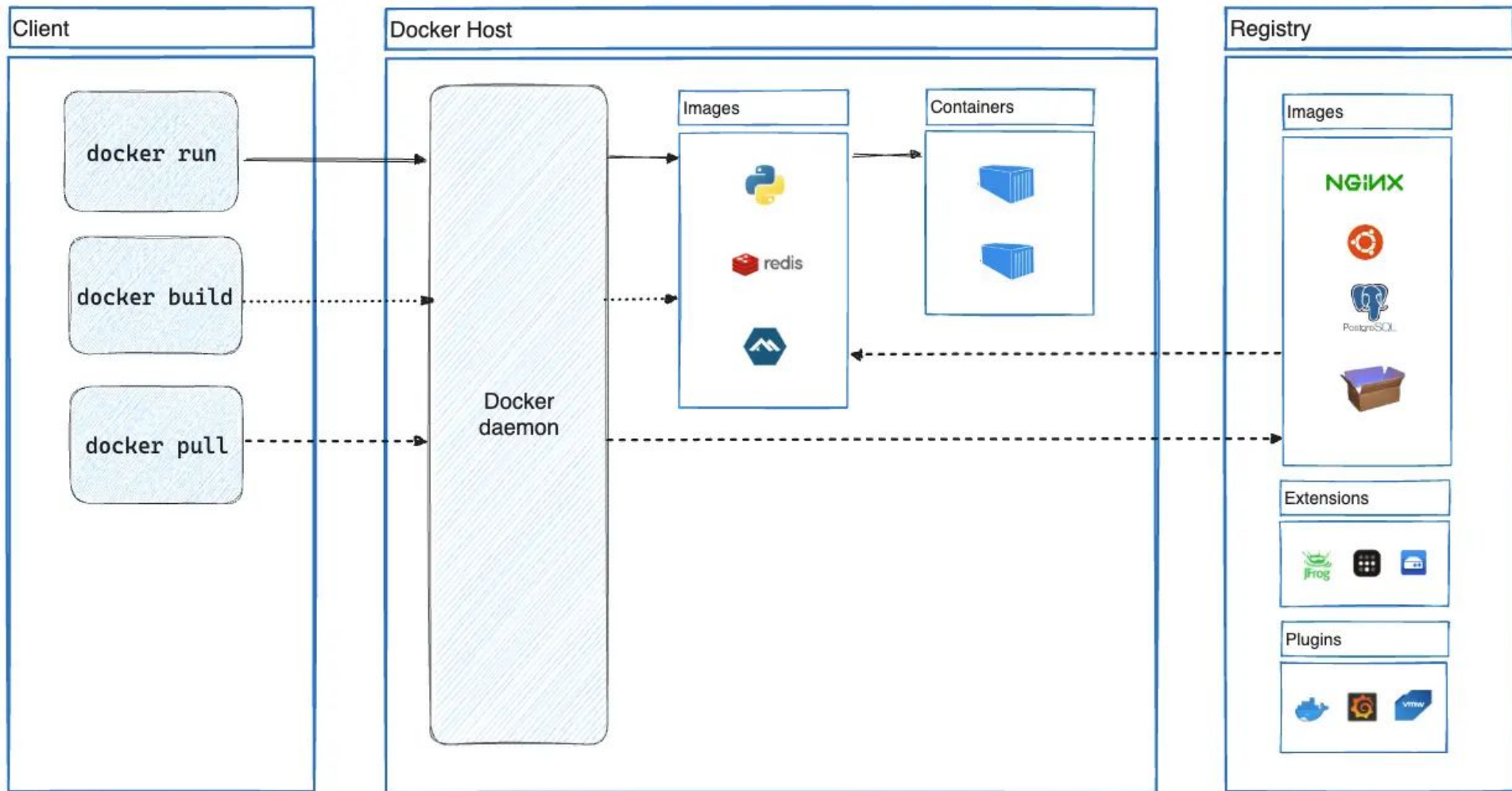


Virtual



Cloud







# Docker asosiy komponentlari

1. **Client** - dockerni resource lari bilan foydalanuvchilarni o'zaro aloqasini yo'lga qo'yadigan interfeys.

\*resource - container, images, volumes, networks

Shuningdek docker host va docker daemon bilan aloqa qilishda ishlatiladigan mahsus komandalarni o'z ichiga oladi:

- **docker run**
- **docker pull**
- **docker build**

Docker daemon bilan bo'ladigan o'zaro aloqa REST API orqali bo'ladi odatda

# Docker Client komandolari

- **docker run** - yangi kontainer yaratadi va uni run qiladi
- **docker start, docker stop** - mavjud kontainer ni ishga tushiradi va to'xtatadi
- **docker ls** - kontainer larni ro'yhatini ko'rsatadi
- **docker rm** - ishlamayotgan kontainerlarni o'chiradi
- **docker pull <image>** - Image ni Docker Hub dan yoki boshqa registry dan yuklab oladi.
- ...

Docker komandolari uchun yo'riqnoma: [Link](#)

# Docker asosiy komponentlari

2. **Docker Host** - Docker injin ishlab turgan fizikal yoki virtual server

**Docker Engine** - host mashina o'rnatiladi va kontainerlarni ishlashini ta'minlaydi

- **Docker daemon** va **Docker CLI** ni o'z ichiga oladi

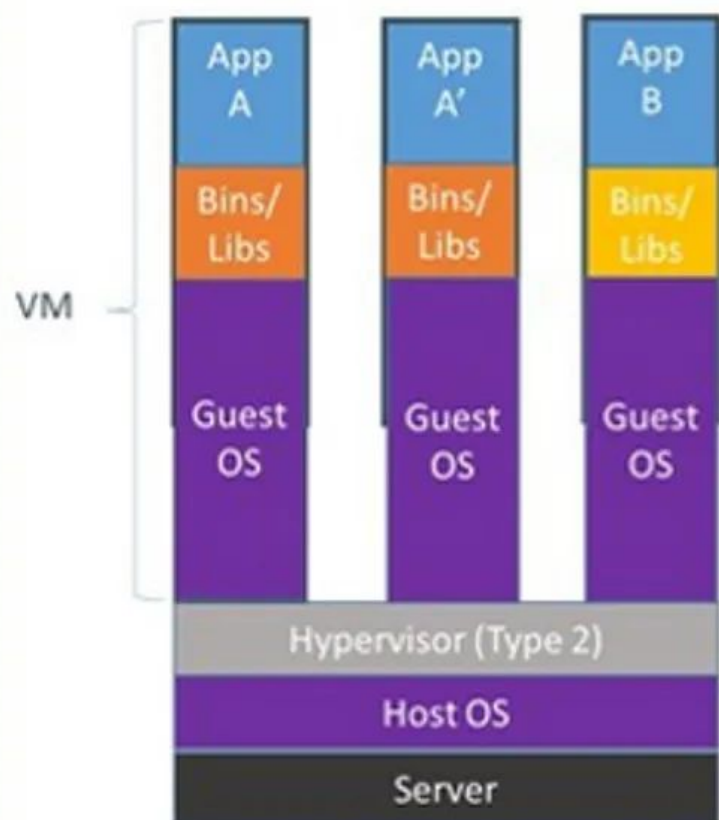
**Docker daemon** - kontainerlar, image, volume va networklarni boshqaradigan va backgroundda(orqa tomonda) ishlaydigan servis

**Docker CLI** - Kommand line interface hisoblanadi, docker daemon bilan foydalanuvchi o'rtasidagi muloqotni ta'minlaydi

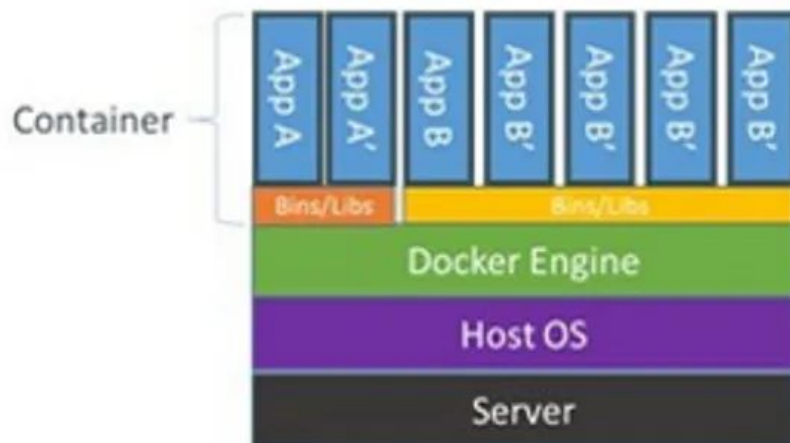
# Docker Host turlari

1. **Local Docker Host** - bu sizni lokalniy kompyuteringiz, development yoki testing uchun foydalaniladi
2. **Cloud Docker Host** - bu turli xil cloudlarda ishlab turgan virtual mashina yoki kontainer service lar  
Misol uchun - AWS EC2 instances, Google Cloud Compute Engine, or Azure VMs  
Docker kontainerlar cloudlarga odatda Kubernetes or Docker Swarm kabi Orchestration toollardan foydalangan holda deploy qilinadi.
3. **Remote Docker Host** - bu har qanday remote Virtual muhit bo'lishi mumkin, ularga ssh orqali ulanish mumkin.

# Containers vs. VMs

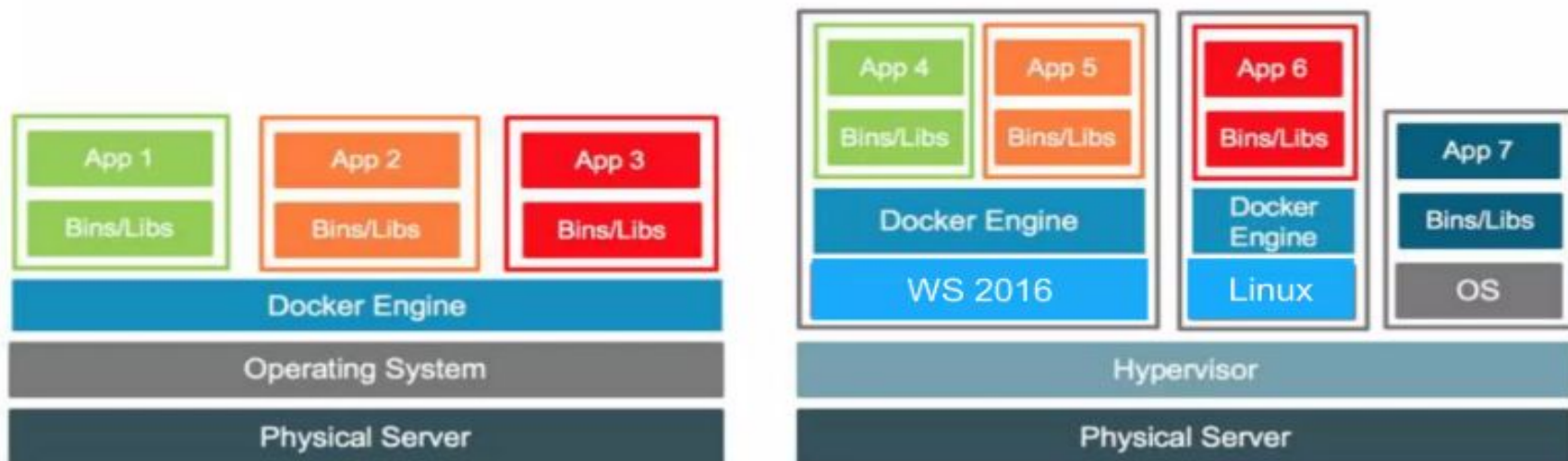


Containers are isolated, but share OS and, where appropriate, bins/libraries



# They're different, not mutually exclusive

Your Datacenter or VPC



## Summary of Differences:

Feature	VMs (Virtual Machines)	Docker Containers
Virtualization	Full hardware virtualization (includes OS)	OS-level virtualization (shares host OS)
Resource Overhead	High (runs a full OS per VM)	Low (shares the host OS kernel)
Isolation	Strong (separate OS per VM)	Weaker (shares kernel, isolated at app level)
Startup Time	Slower (needs to boot the OS)	Faster (no OS booting, just app start)
Performance	Slower due to high resource consumption	Faster and more efficient
Use Case	Running multiple OSes, legacy applications	Microservices, CI/CD, scalable cloud apps
Portability	Less portable (depends on hardware)	Highly portable (runs anywhere with Docker runtime)
Management	Complex (requires hypervisor)	Easier (with Docker and orchestration tools)

# Docker asosiy komponentlari

3. **Docker Registry** - bu markaziy storage(manba) va imagelarni yetkazib beruvchi tizim.

- bu tizim docker image larini saqlovchi, versiyalovchi va foydalanuvchilar uchun image larni mavjudligini ta'minlovchi tizim hisoblanadi. Image pul qilinganda docker registryga murojat qilinadi.

Registry larni 2 xil turi mavjud, public va private.

Public registry - DockerHub([hub.docker.com](https://hub.docker.com))

Private registry - Google Container Register(GCR), Amazon Elastic Container Registry(ECR)



# Docker ni o'rnatish

1. Windows uchun:

<https://docs.docker.com/desktop/setup/install/windows-install/>

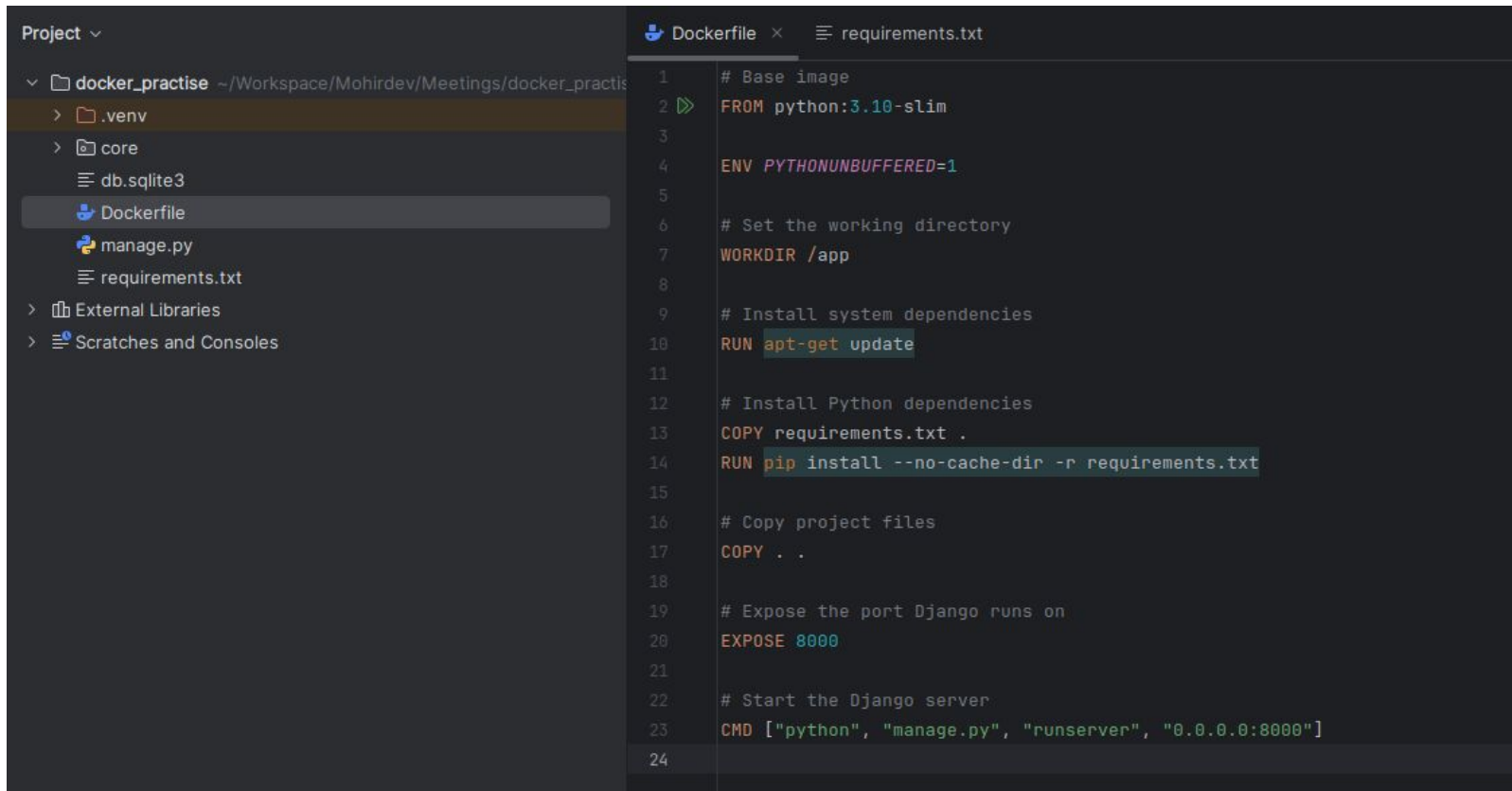
2. Linux asosida ishlaydigan operatsiyin tizimlar uchun Docker Enginer o'rnatilishi kerak:

<https://docs.docker.com/engine/install/>

# Amaliyot

1. Django loyiha yaratib olamiz
2. Dockerfile yaratmiz
3. Docker ni ishga tushiramiz

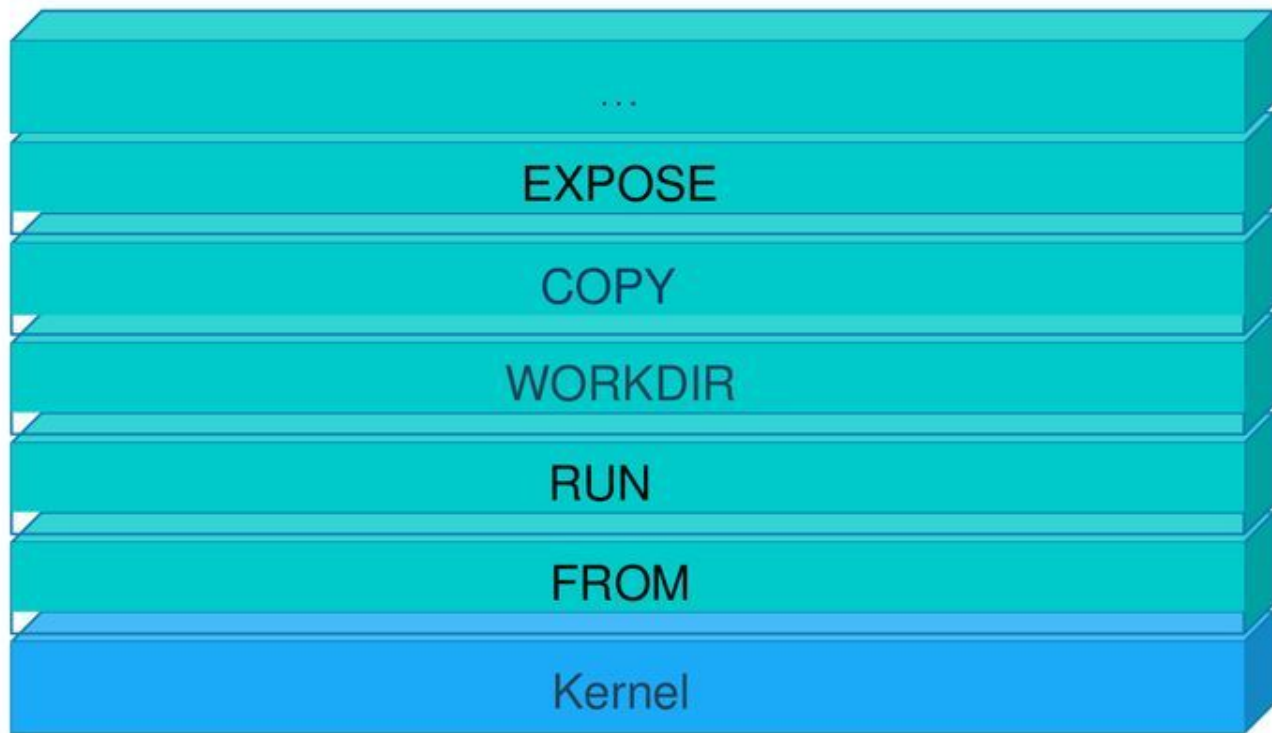
# Dockerfile



The image shows a code editor interface with a project explorer on the left and a code editor on the right. The project explorer shows a project named 'docker\_practise' with a file tree containing '.venv', 'core', 'db.sqlite3', 'Dockerfile', 'manage.py', and 'requirements.txt'. The code editor displays the content of the 'Dockerfile' file, which is a Docker build script for a Django application.

```
1 # Base image
2 FROM python:3.10-slim
3
4 ENV PYTHONUNBUFFERED=1
5
6 # Set the working directory
7 WORKDIR /app
8
9 # Install system dependencies
10 RUN apt-get update
11
12 # Install Python dependencies
13 COPY requirements.txt .
14 RUN pip install --no-cache-dir -r requirements.txt
15
16 # Copy project files
17 COPY . .
18
19 # Expose the port Django runs on
20 EXPOSE 8000
21
22 # Start the Django server
23 CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
24
```

# Each Dockerfile Command Creates a Layer



# Xulosa

Agar jamoa bo'lib ishlasangiz Dockerdan foydalanishga odatlaning

Deployment ni Docker orqali qilishga harakat qiling