By default docker scout , docker init comes in docker desktop if we need extensively in Linux We need to add plugins

docker scout

image scanning tool - docker scout, trivy to scan the vulnerabilities - is any security threats in base image or final image

- docker scout quickview imagename = vulnerability info

- docker scout cves imagename = cves - common vulnerabilities and exposure for depth it provides CVE of vulnerability

---

Docker init - make templates for dockerfile, docker compose, readme file

- docker init

After init we have to select base image - java, python go
then version, main package, port

.dockerignore - we dont push anything which we need for contenarised

---

Container

docker ps - to see running containers

docker ps -a - to see exited containers

docker stop 7405ac804cfc && docker rm 7405ac804cfc - stop and remove container

docker run -d -p 5000:5000 -e MYSQL_HOST=mysql -e MYSQL_USER=root -e MYSQL_PASSWORD=root -e MYSQL_DB=devops two-tier:latest

docker run -d -e MYSQL_ROOT_PASSWORD=root MySQL - MySQL docker run

docker exec -it f27c3aaddc2f bash - to enter inside container

docker logs 3a918e377c07 - to see logs of container

docker system prune

```
WARNING! This will remove:
  - all stopped containers
  - all networks not used by at least one container
  - all dangling images
  - unused build cache

Are you sure you want to continue? [y/N] y
```

Image

docker build -t two-tier . - build image

docker rmi java-app:latest - to remove image

```
docker rmi -f $(docker images -aq)


------------------------------------------------------

## docker network

docker network ls

docker network create two-tier -d bridge - -d is driver here

docker network inspect two-tier - it will show how many containers attach with
the same network

docker run -d -p 5000:5000 --network two-tier -e MYSQL_HOST=mysql -e
MYSQL_USER=root -e MYSQL_PASSWORD=root -e MYSQL_DB=devops two-tier:latest

------------------------------------------------

## docker volume and storage

docker volume ls

docker volume create mysql-data

docker volume inspect mysql-data - it will show the bydefault path where data
will store - /var/lib/docker/volumes/mysql-data/_data

### mysql-data:/var/lib/MySQL persist volume here  mysql-data is named volume

docker run -d --name mysql --network two-tier -v mysql-data:/var/lib/mysql -e
MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=devops MySQL   -


## Alternate method to create volume

create one folder

/home/ubuntu/volumes/MySQL - created volume , /var/lib/mysql - inside sql
bydefault data stored here

docker run -d --name mysql --network two-tier -v
/home/ubuntu/volumes/mysql:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -e
MYSQL_DATABASE=devops MySQL   - mysql-data:/var/lib/MySQL persist volume here
mysql-data is named volume

------------------------------------------------------------------------------
------------------------

### Docker-Compose configuration file


#### two-tier-flask-app

ubuntu@ip-172-31-10-151:~/projects/two-tier-flask-app$ cat docker-compose.yml
version: "3.8"

services:
  mysql:
    container_name: mysql
    image: mysql
    ports:
      - "3306:3306"
    environment:
```

```
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: devops
    volumes:
      - mysql-data:/var/lib/mysql
    networks:
      - two-tier
    restart: always
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost", "-uroot", "-proot"]
      interval: 5s
      timeout: 10s
      retries: 5
      start_period: 60s

  flask:
    container_name: two-tier-app
    build:
      context: .
    ports:
      - "5000:5000"
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: root
      MYSQL_PASSWORD: root
      MYSQL_DB: devops
    networks:
      - two-tier
    depends_on:
      - mysql
    restart: always
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:5000/health || exit 1"]
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 30s



volumes:
  mysql-data:

networks:
  two-tier:

------------------------------------------------------------------------------
---------

### django-notes-app - Deploying a web application within a nginx and MYsql

ubuntu@ip-172-31-10-151:~/projects/django-notes-app$ cat docker-compose.yml
version: "3.8"


services:
  db:
    container_name: db_cont
    image: mysql
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: test_db
    volumes:
```

```
      - mysql-store:/var/lib/mysql
    networks:
      - django-net
    restart: always
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost", "-uroot", "-proot"]
      interval: 5s
      timeout: 10s
      retries: 5
      start_period: 60s


  django_app:
    container_name: django_cont
    build:
      context: .
    image: django_app
    command: sh -c "python manage.py migrate --no-input && gunicorn
notesapp.wsgi --bind 0.0.0.0:8000"
    ports:
      - "8000:8000"
    env_file:
      - ".env"
    networks:
      - django-net
    depends_on:
      - db
    restart: always
    healthcheck:
      test: ["CMD-SHELL", "curl -f htttp://localhost:8000/admin | exit 1"]
      interval: 5s
      timeout: 10s
      retries: 5
      start_period: 60s

  nginx:
    container_name: nginx_cont
    build:
      context: ./nginx
    ports:
      - "80:80"
    networks:
      - django-net
    depends_on:
      - django_app
    restart: always

volumes:
  mysql-store:
networks:
  django-net:


-------------------------------------------------------

### Expenses-Tracker-WebApp

version: "3.8"
services:
  mysql:
    image: mysql:latest
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: Test@123
```

```
      MYSQL_DATABASE: expenses_tracker
    volumes:
      - ./mysql-data:/var/lib/mysql
    networks:
      - appbridge
    healthcheck:
      test: ["CMD","mysqladmin","ping","-h","localhost","-uroot","-pTest@123"]
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 30s
    restart: always

  mainapp:
    image: snehcreate/expensetracker_v3
    container_name: Expensetracker
    environment:
      SPRING_DATASOURCE_USERNAME: root
      SPRING_DATASOURCE_URL: "jdbc:mysql://mysql:3306/expenses_tracker?
allowPublicKeyRetrieval=true&useSSL=false"
      SPRING_DATASOURCE_PASSWORD: Test@123
    ports:
      - "8080:8080"
    networks:
      - appbridge
    depends_on:
      - mysql
    restart: always

networks:
  appbridge:

volumes:
  mysql-data:
```

```
# docker compose commands

Command     Description
docker compose up Starts and builds the containers as needed.
docker compose down     Stops and removes containers, networks, and volumes
(defined in the file).
docker compose build    Builds the images defined in the Compose file.
docker compose start    Starts existing (already created) containers.
docker compose stop     Stops running containers without removing them.
docker compose restart  Restarts all services.

ðŸ”  Inspection & Logs
Command     Description
docker compose ps Lists containers created by the Compose file.
docker compose logs     Shows logs from all services.
docker compose logs -f  Follows the logs in real time.
docker compose exec <service> <cmd> Runs a command inside a running service
container (e.g., bash, sh).
```

docker compose top Shows running processes inside services.

ðŸ§¹ Cleanup & Maintenance
Command Description
docker compose rm Removes stopped service containers.
docker compose down -v Also removes named volumes declared in volumes:.
docker compose down --rmi all Also removes images built by build:.

ðŸ§ª Testing & One-Off Commands
Command Description
docker compose run <service> <cmd> Run a one-off command (e.g., migrations).
docker compose config Validates and prints the merged config.

---

docker registry - docker hub

- docker login = authentication token

- docker pull mysql

- docker tag python-app:latest suraj0221/python-app

- docker push suraj0221/python-app

Multistage docker builds - Two FROM

ubuntu@ip-172-31-10-151:~/projects/flask-app-ecs$ cat Dockerfile

```
#STAGE 1

# Base image (OS)

FROM python:3.9 AS Builder

# Working directory

WORKDIR /app

# Copy src code to container

COPY requirements.txt .

# Run the build commands

RUN pip install -r requirements.txt

# -------------------------

#STAGE 2

FROM python:3.9-slim


# Working directory

WORKDIR /app

# Copy src code to container
```

```
COPY --from=builder /usr/local/lib/python3.9/site-packages/
/usr/local/lib/python3.9/site-packages/

COPY . .

# expose port 80

EXPOSE 80

# serve the app / run the app (keep it running)

CMD ["python","run.py"]
```

Monitoring and logging

```
docker logs d701c57a1ec8

docker attach f6a4e8488918

nohup docker attach f6a4e8488918 &
```

The key components of a Docker architecture are: the Docker Engine, the Docker Registries, and the Docker Objects (Images, Containers, Network, Storage)

https://www.linkedin.com/posts/activity-7356211183990849536-KRe7?utm_source=share&utm_medium=member_desktop&rcm=ACoAADMQgPcBNQRtqrAJl_uE3bGEspjHN2zsHq4

Basic Docker Interview Questions and Answers

What is Docker?

Docker is an open-source platform that automates the deployment, scaling, and management of applications using containerization. Containers package software with all its dependencies into a single unit that runs reliably across environments.

What are containers, and how are they different from virtual machines (VMs)?

Containers share the host OS kernel and isolate applications at the process level, making them lightweight. VMs, in contrast, run full guest operating systems and are heavier. Containers start faster and use fewer resources than VMs.

What is the difference between Docker image and container?

An image is a read-only template that defines a containerâ ™s contents. A container is a running instance of an image with a writable layer on top.

What is the use of Dockerfile?

A Dockerfile is a script containing a set of instructions to build a Docker image. It defines the environment, dependencies, and commands to run the app.

How do you create a Docker container?
Use the command:

bash
Copy
Edit
docker run -it --name my_container my_image
What is the role of Docker Hub?
Docker Hub is a cloud-based registry where Docker users can store and share container images.

What is the difference between CMD and ENTRYPOINT in a Dockerfile?
CMD provides default arguments for the containerâ ™s execution. ENTRYPOINT sets the command to run and makes the container behave like an executable. You can override CMD, but ENTRYPOINT is fixed unless explicitly changed.

How do you list all running Docker containers?

bash
Copy
Edit
docker ps
How do you remove a Docker image or container?

bash
Copy
Edit
docker rm <container_id>   # Remove container
docker rmi <image_id>   # Remove image
How does Docker achieve isolation?
Docker uses Linux kernel features like namespaces (for isolation) and cgroups (for resource control) to isolate containers from each other and the host.

ðŸŸ¡ Intermediate Docker Interview Questions and Answers
How does Docker networking work?
Docker provides network drivers: bridge, host, overlay, and none. The default is bridge, which connects containers on the same host via a virtual bridge interface.

What are Docker volumes, and how are they different from bind mounts?
Volumes are managed by Docker and stored in a specific part of the host filesystem. Bind mounts are dependent on the host directory structure. Volumes are safer and preferred for data persistence.

Explain Docker Compose.
Docker Compose is a tool for defining and running multi-container Docker applications using a YAML file (docker-compose.yml).

How do you scale services using Docker Compose?
Use:

bash
Copy
Edit
docker-compose up --scale web=3
What are some common Dockerfile best practices?

Use multi-stage builds

Minimize layers

Use .dockerignore

Use official base images

Avoid installing unnecessary packages

How do you pass environment variables to a Docker container?

With -e flag:

bash
Copy
Edit
docker run -e ENV_VAR=value my_image
Or via .env file in Docker Compose

How do you handle container logs?
Use docker logs <container_name> or configure log drivers (e.g., json-file, syslog, fluentd) for external logging.

What happens when a container crashes?
If not configured, it stops. Restart policies like --restart=always can automatically restart containers on failure.

Explain multi-stage builds in Docker.
Multi-stage builds use multiple FROM statements to reduce image size by copying only necessary files to the final image.

What is the difference between COPY and ADD in a Dockerfile?

COPY only copies files/directories.

ADD can also extract tar archives and handle remote URLs.

ðŸ”´ Advanced Docker Interview Questions and Answers
What is the difference between Docker Swarm and Kubernetes?
Both are container orchestration tools. Kubernetes is more powerful, with features like auto-scaling and rolling updates. Docker Swarm is simpler and integrated with Docker.

How do you secure Docker containers?

Use minimal base images

Run containers as non-root

Limit container capabilities

Use image scanning tools

Apply network segmentation

How does Docker caching work during the image build process?
Docker caches image layers. If a layer hasn't changed, Docker reuses it to speed up the build. Changes invalidate subsequent cached layers.

How can you optimize the size of a Docker image?

Use smaller base images (e.g., alpine)

Combine RUN commands

Clean up packages

Use multi-stage builds

Explain the layered architecture of Docker images.

Docker images are composed of stacked layers. Each instruction in a Dockerfile creates a new layer. Layers are cached and reused across images.

What are namespaces and cgroups in Docker?

Namespaces provide isolation (PID, NET, IPC, etc.)

Cgroups limit and monitor resources (CPU, memory, etc.)

How do you debug a running container?
Use:

bash
Copy
Edit
docker exec -it <container_id> /bin/sh
What is the difference between a host network and a bridge network?

Bridge: Default; creates a virtual network on the host

Host: Shares hostâ ™s networking namespace; no isolation

How do you perform zero-downtime deployments with Docker?
Use rolling updates with orchestration tools like Kubernetes or Swarm. Load balancers help by routing traffic only to healthy containers.

How do you monitor Docker containers in production?
Tools: Prometheus, Grafana, ELK Stack, cAdvisor, Datadog, etc. Metrics like CPU, memory, disk I/O, and logs are typically monitored.

âœ… Behavioral/Scenario-Based Questions and Sample Answers
Resolved container performance issue?
"I analyzed the container's CPU and memory usage using docker stats and found a memory leak. Rewrote part of the app and added resource limits to avoid future issues."

Docker in CI/CD pipelines?
"Used Docker to containerize builds and tests in Jenkins. This ensured consistent build environments and faster rollbacks."

Containerizing a legacy app?
"Identified dependencies, created a Dockerfile, and used volume mounts for configuration. Gradually migrated the app to Docker without downtime."

Slow image build fix?
"Used multi-stage builds, reduced layers, and used .dockerignore to exclude unnecessary files, improving build speed by ~40%."

Debugging memory leaks in containers?
"Monitored memory usage with Prometheus + Grafana. Profiled the app inside the container using tools like valgrind and fixed inefficient code."