**Phase 2: Convert to Docker Swarm Setup (Single Node Initially)**

**Convert the setup to Docker Swarm mode on the same machine.**

1> Initialize Swarm.

2> Create required volumes and networks.

3> Deploy services (n8n and PostgreSQL) using a Docker Swarm-compatible compose file.

4> Ensure 2 n8n containers (replicas) are running and connected to the same Postgres database (for clustering).

**Initialize Docker Swarm (on the same EC2 machine)**

What is Docker swarm : Is provide native container orchestration capablities within the docker eco system.

Docker swarm deploy and manage applications as a service, So its automatically handling the task, Such as service creation, Update and rollbacks.

Easy to scalling of services by adding or removing container replicas, and it include built in load balancing to distribute incomming traffic evenly across the running containers.

incase node failure the docker swarm automatically reschedules containers in to healthy node.

Minimize downtime.

It allows user to create and manage a cluster of docker host (nodes) as a single, unified system, enabling the deployment, **scaling and management of containerized applications across multiple machines.**

-> Cluster Management

-> Sevice deployment and management

-> Scaling and Load balancing

-> High availablity and fault tolerance.

-> Service discovery

-> Simplified and Orchestration.

**1> Initialize Swarm.**

The purpose of docker swarm initialization is to transform standalone docker engine in to swarm manager node and establish a new docker swarm cluster.

**docker swarm init**

This will output a join token (We can ignore it for now — it's for multi-node setup).



**Created Docker Volumes (for persistence) in Phase 1 with the help of Docker-Compose.yml file.**

**If we want to create docker volume by using the below command**

**docker volume create**



**Created named volumes that will be reused in the stack.**



Thse are Docker-named volumes, not bind mounts.

**n8n\_storage** → persists data for the n8n container (/home/node/.n8n)

**postgres\_storage** → stores PostgreSQL database files (/var/lib/postgresql/data)

**ollama\_storage** → used by the ollama containers (/root/.ollama)

**qdrant\_storage** → persists vector storage for Qdrant (/qdrant/storage)

Named volumes are ideal for persistent data that containers need to retain between restarts or rebuilds. (Even the container removed the data will be persistent)





Need to Create an Overlay Network (for service communication), But we already creted default bridge network through the Docker-compose.yml file.







Docker bridge network mode is a default network type of containers.

Its creates a private, Internal networ on the docker host, allowing containers on the same host to communicate with each other.

But in Containers are in the bridge network also can communicate with external networks with the help of NAT.

Offering for isolation.

Here we are going to use multiple container deployments so we are gong to create Overlaty network.

Enable communication between running on different docker host.

So now we are going to change networks in Overlay mode in Docker-compose.yml file.

Just edit from

networks:

demo:

To:

networks:

demo:

driver: overlay

**Convert Your docker-compose.yml to Swarm Format**

**Building a complex and flexible stack (n8n + PostgreSQL + Qdrant + Ollama with multi-device support**

Rename your file to docker-stack.yml. Then modify it with Swarm-specific options.



**Deploy the Stack**

We need to do adjustments for Docker Swarm from docker-compose.yml file.

This creates and starts all services under the stack name n8n\_stack.

docker stack deploy -c docker-stack.yml n8n\_stack