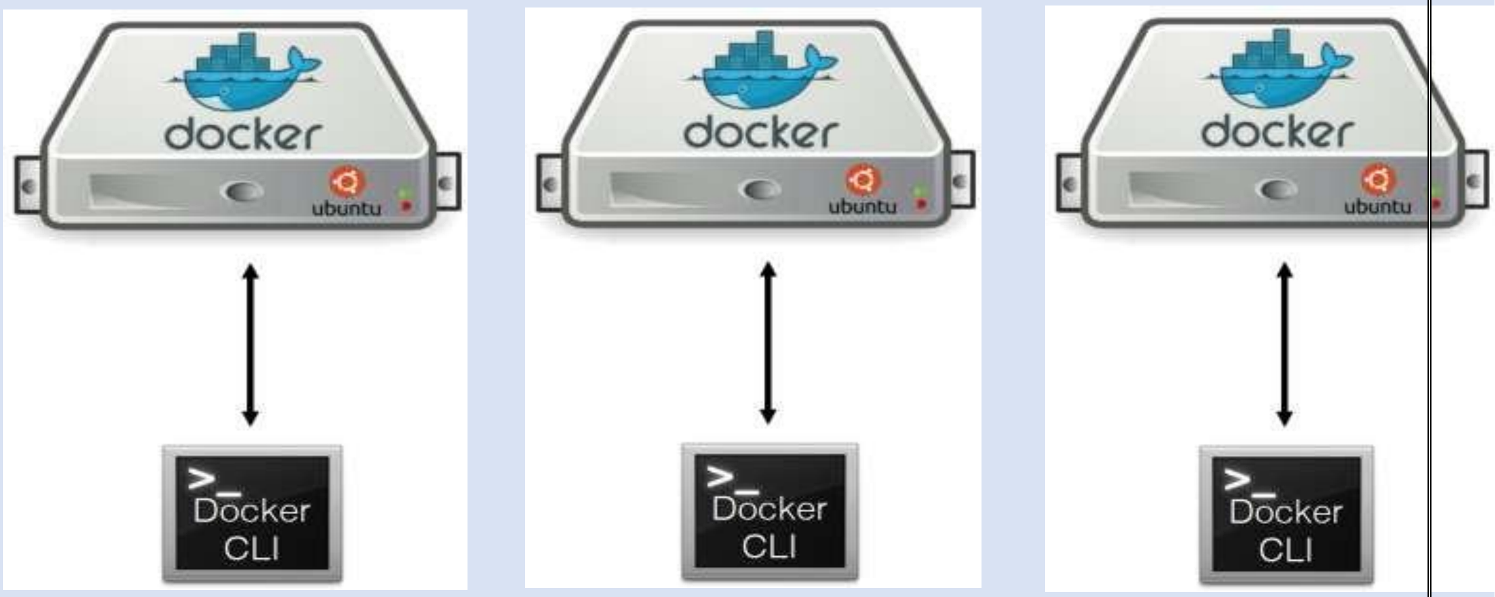
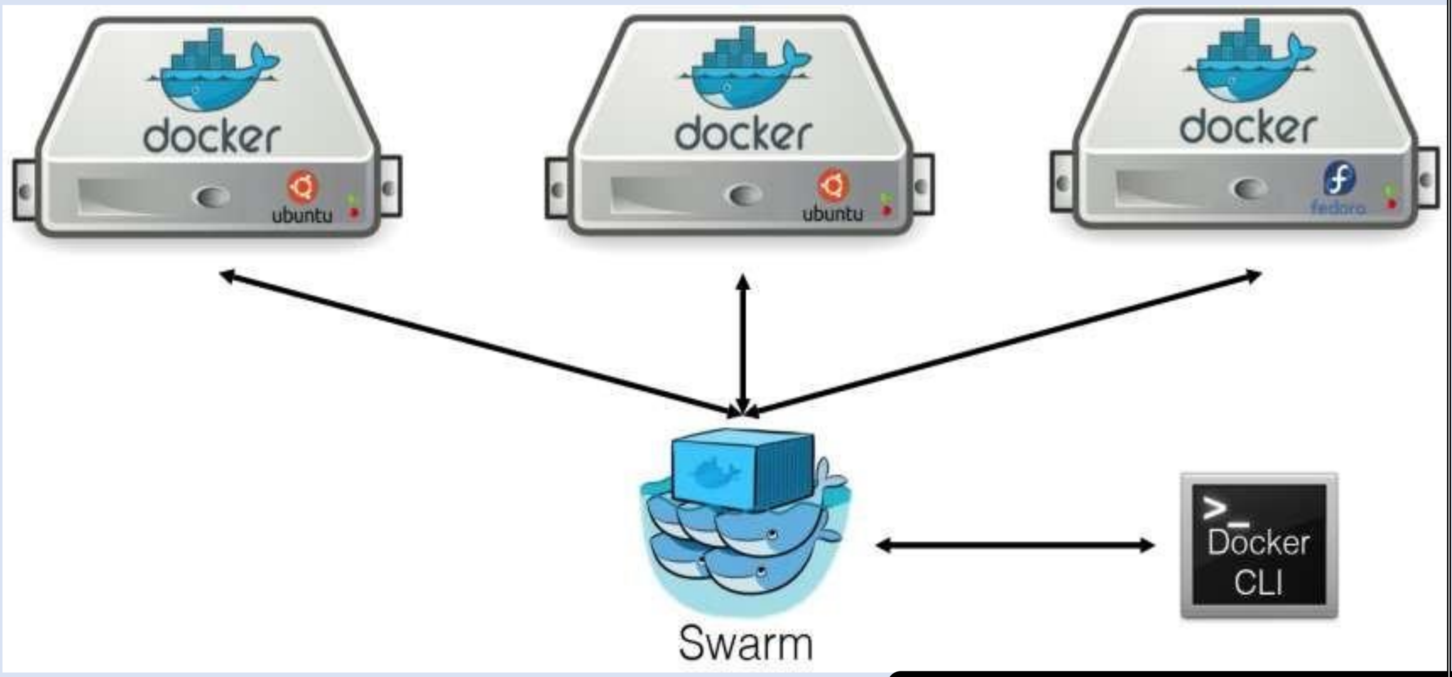
Docker Swarm

What we did till now??

Till now we had created one container on our docker host machine or created multiple container using docker compose on docker host machine. Now suppose our docker host machine goes down, then all the containers will be down, and all our data will be lost present inside the container, which is a big challenge. So, how can we overcome it.

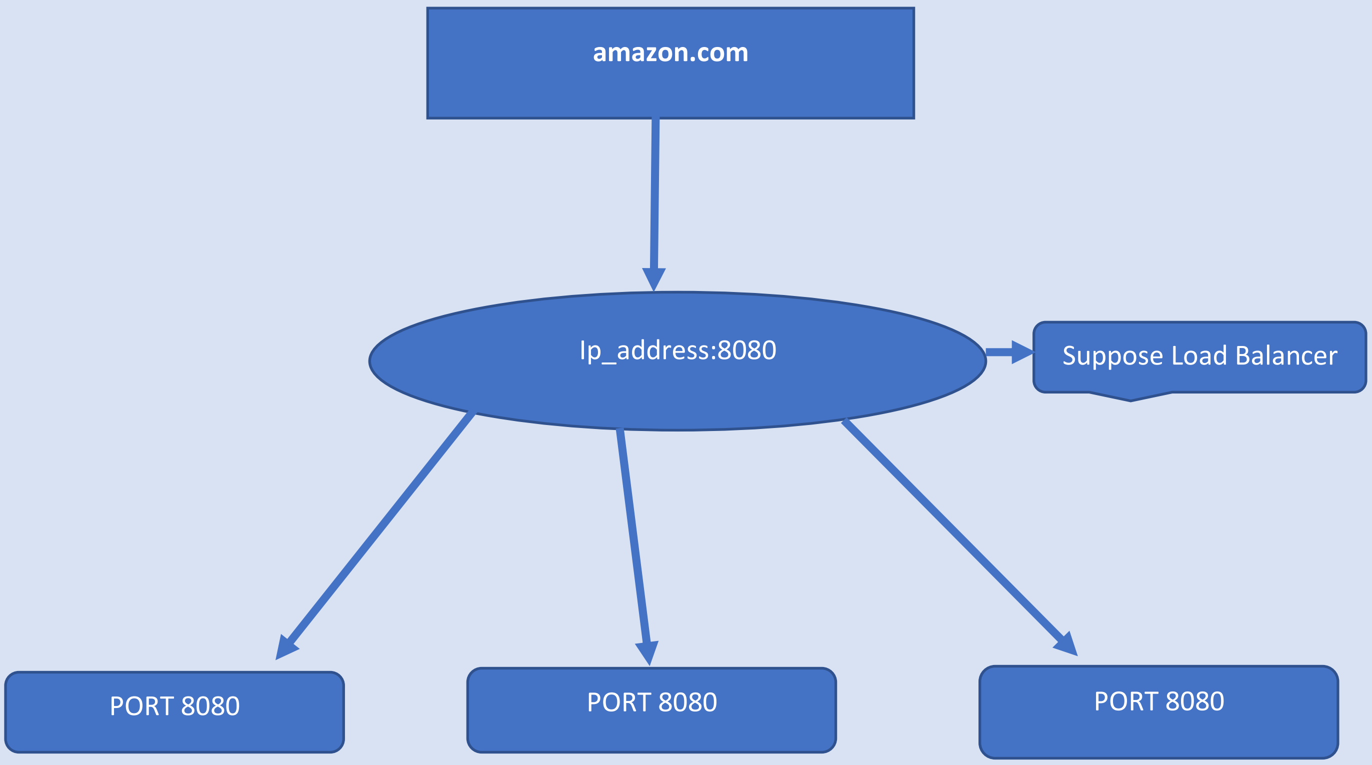


So here comes the solution of cluster, Clustering means we can deploy any of our application inside container in multiple server/instances which would be synchronized with each other, so that if any server goes down it won't affect our application as it would be running in other server/instances in the cluster.



Other advantage of docker swarm is load balancing, In Below given scenario, suppose end user is clicking on amazon.com, then it will get redirected to ip_address of amazon along with the port no. where amazon application is deployed, so now suppose if all application of amazon gets deployed in single server then when the load will increase the server will go down, so docker swarm help us here also to divide the load in multiple server in the docker cluster so that no server would get more load:-

.



Docker Swarm: -

- **Docker Swarm** is a clustering and scheduling tool for **Docker** containers. With **Swarm**, IT administrators and developers can establish and manage a cluster of **Docker** nodes as a single virtual system

- Docker Swarm Mode enables
  - To have multiple Docker Engines / Nodes as a single cluster
  - Also Provides orchestration features

- One manager, and rest workers (More then one manager can also be present, and recommended manager is three).
- Docker Swarm features are comes with Docker Engine – we do not need to install it separately like docker compose. We can just turn our Docker Engine into a Swarm mode.

Docker Swarm Advantages: -

- **Cluster management integrated with Docker Engine:** Use the Docker Engine CLI to create a swarm of Docker Engines where you can deploy application services. You don’t need additional orchestration software to create or manage a swarm.
- **Decentralized design:** Instead of handling differentiation between node roles at deployment time, the Docker Engine handles any specialization at runtime. You can deploy both kinds of nodes, managers and workers, using the Docker Engine.
- **Declarative service model:** Docker Engine uses a declarative approach to let you define the desired state of the various services in your application stack. For example, you might describe an application comprised of a web front end service with message queueing services and a database backend.
- **Scaling:** For each service, you can declare the number of tasks you want to run. When you scale up or down, the swarm manager automatically adapts by adding or removing tasks to maintain the desired state.
- **Desired state reconciliation:** The swarm manager node constantly monitors the cluster state and reconciles any differences between the actual state and your expressed desired state. For example, if you set up a service to run 10 replicas of a container, and a worker machine hosting two of those replicas’ crashes, the manager creates two new replicas to replace the replicas that crashed. The swarm manager assigns the new replicas to workers that are running and available.
- **Multi-host networking:** You can specify an overlay network for your services. The swarm manager automatically assigns addresses to the containers on the overlay network when it initializes or updates the application.
- **Service discovery:** Swarm manager nodes assign each service in the swarm a unique DNS name and load balances running containers. You can query every container running in the swarm through a DNS server embedded in the swarm.
- **Load balancing:** You can expose the ports for services to an external load balancer. Internally, the swarm lets you specify how to distribute service containers between nodes.
- **Secure by default:** Each node in the swarm enforces TLS mutual authentication and encryption to secure communications between itself and all other nodes. You have the option to use self-signed root certificates or certificates from a custom root CA.
- **Rolling updates:** At rollout time you can apply service updates to nodes incrementally. The swarm manager lets you control the delay between service deployment to different sets of nodes. If anything goes wrong, you can roll-back a task to a previous version of the service.