**1. What is a Docker Swarm?**

Docker Swarm is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual Docker host. Docker Swarm serves the standard Docker API, any tool that already communicates with a Docker daemon can use Swarm to transparently scale to multiple hosts

**2. Explain the Orchestration.**

The portability and reproducibility of a containerized process mean we have an opportunity to move and scale our containerized applications across clouds and datacenters. Containers effectively guarantee that those applications run the same way anywhere, allowing us to quickly and easily take advantage of all these environments. Furthermore, as we scale our applications up, we’ll want some tooling to help automate the maintenance of those applications, able to replace failed containers automatically, and manage the rollout of updates and reconfigurations of those containers during their lifecycle.

Tools to manage, scale, and maintain containerized applications are called orchestrators, and the most common examples of these are Kubernetes and Docker Swarm. Development environment deployments of both of these orchestrators are provided by Docker Desktop, which we’ll use throughout this guide to create our first orchestrated, containerized application.

**3. What is the Importance of Orchestration?**

Container orchestration is used to automate the following tasks at scale:

- Provisioning and deployments of containers

- Availability of containers

- Load balancing, traffic routing and service discovery of containers

- Health monitoring of containers

- Securing the interactions between containers.

- Configuring and scheduling of containers

- The configuration of applications in terms of the containers that they run in

- Scaling of containers to equally balance application workloads across infrastructure

- Allocation of resources between containers

**4. What is the difference between Docker and Docker Swarm?**

The purpose of designing Docker was to create an environment for running containers. Whereas Docker Swarm runs on top of many Docker hosts to orchestrate containers on many machines. Docker Swarm also adds some capabilities to docker, like cross-machine docker networking, clustering, and simplify management on large container ecosystem. It can run on a single machine, but for study purposes. Another difference is that Docker is one of the container technologies while Docker Swarm is a container scheduler similar to Kubernetes.

**5. What are the main differences between Docker Swarm and Kubernetes?**

Docker and Kubernetes share many differences between them, to you help understand the differences between the two, the below-mentioned list would help understand the differences:

- Docker Swarm does not support auto-scaling, while Kubernetes allows the auto-scaling feature

- Installing Kubernetes requires you to spend more time configuring the files, on the other hand, Docker swarm are easy to install

- Docker swarm supports only 2000 nodes, whereas Kubernetes supports up to 5000 nodes

**6. What are Docker Swarm nodes?**

Manager Node: The primary function of manager nodes is to assign tasks to worker nodes in the swarm. Manager nodes also help to carry out some of the managerial tasks needed to operate the swarm. Docker recommends a maximum of seven manager nodes for a swarm.

Worker Node: In a docker swarm with numerous hosts, each worker node functions by receiving and executing the tasks that are allocated to it by manager nodes. By default, all manager modes are also worker nodes and are capable of executing tasks when they have the resources available to do so.

**7. What are benefits of Docker Swarm?**

- Decentralized design

- Declarative service model

- Scaling

- Desired state reconciliation

- Multi-host networking

- Service discovery

- Load balancing

- Secure by default

- Rolling updates

**8. How do you enable Docker Swarm?**

Docker Swarm Mode comes integrated with Docker Platform. Starting 1.12, Docker Swarm Mode is rightly integrated which means that you don’t need to install anything outside to run Docker Swarm.Docker Desktop runs primarily on Docker Engine, which has everything you need to run a Swarm built-in. Follow the setup and validation instructions appropriate for your operating system:

Open a powershell, and initialize Docker Swarm mode:

docker swarm init

**9. What are features of Docker Swarm?**

Some of the most essential features of Docker Swarm are:

- Decentralized access: Swarm makes it very easy for teams to access and manage the environment

- High security: Any communication between the manager and client nodes within the Swarm is highly secure

- Autoload balancing: There is autoload balancing within your environment, and you can script that into how you write out and structure the Swarm environment

- High scalability: Load balancing converts the Swarm environment into a highly scalable infrastructure

- Roll-back a task: Swarm allows you to roll back environments to previous safe environments

**10. What are Service and Tasks in swarm?**

- Docker containers are launched using services.

- Services can be deployed in two different ways - global and replicated.

- Global services are responsible for monitoring containers that want to run on a Swarm node. In contrast, replicated services specify the number of identical tasks that a developer requires on the host machine.

- Services enable developers to scale their applications.

- Before deploying a service in Swarm, the developer should implement at least a single node.

- Services can be used and accessed by any node of the same cluster.

- A service is a description of a task, whereas a task performs the work.

- Docker helps a developer in creating services, which can start tasks. However, when a task is assigned to a node, the same task cannot be attributed to another node.

**11. How do I scale a service in Docker Swarm?**

1 - If you haven’t already, open a terminal and ssh into the machine where you run your manager node. For example, the tutorial uses a machine named manager1.

2 - Run the following command to change the desired state of the service running in the swarm:

$ docker service scale <SERVICE-ID>=<NUMBER-OF-TASKS>

**12. How do I run a docker service?**

1 - Open a terminal and ssh into the machine where you run your manager node. For example, the tutorial uses a machine named manager1.

2 - Run the following command:

$ docker service create --replicas 1 --name helloworld alpine ping docker.com

9uk4639qpg7npwf3fn2aasksr

- The docker service create command creates the service.

- The --name flag names the service helloworld.

- The --replicas flag specifies the desired state of 1 running instance.

- The arguments alpine ping docker.com define the service as an Alpine Linux container that executes the command ping docker.com.

**13. What is service scale?**

The scale command enables you to scale one or more replicated services either up or down to the desired number of replicas. This command cannot be applied on services which are global mode. The command will return immediately, but the actual scaling of the service may take some time. To stop all replicas of a service while keeping the service active in the swarm you can set the scale to 0.

Usage

docker service scale SERVICE=REPLICAS [SERVICE=REPLICAS...]

**14. How do you inspect a service in swarm?**

Run docker service inspect --pretty <SERVICE-ID> to display the details about a service in an easily readable format. To return the service details in json format, run the same command without the --pretty flag.

**15. What are cons of using Docker Swarm?**

- Docker is platform dependent: Docker Swarm is a Linux agonistic platform. Although Docker supports Windows and Mac OS X, it utilizes virtual machines to run on a non-Linux platform. An application which is designed to run in a docker container on Windows can’t run on Linux and vice versa.

- Doesn’t provide storage option: Docker Swarm doesn’t provide a hassle-free way to connect containers to storage and this is one of the major disadvantages. Its data volumes require a lot of improvising on the host and manual configurations. If you’re expecting Docker Swarm to solve the storage issues, it may get done but not in an efficient and user-friendly way.

- Poor monitoring: Docker Swarm provides the basic information about the container and if you are looking for the basic monitoring solution than Stats command is suffice. If you are looking for the advanced monitoring than Docker Swarm is never an option. Although there are third-party tools available like CAdvisor which offers more monitoring. It is not feasible to collect more data about containers in real-time with Docker itself.