Docker Swarm

What is a Docker Swarm?

A Docker Swarm is a group of either physical or virtual machines that are running the Docker application and that have been configured to join together in a cluster. Once a group of machines have been clustered together, you can still run the Docker commands that you're used to, but they will now be carried out by the machines in your cluster. The activities of the cluster are controlled by a swarm manager, and machines that have joined the cluster are referred to as nodes.

What is Docker Swarm used for?

Docker swarm is a container orchestration tool, meaning that it allows the user to manage multiple containers deployed across multiple host machines.

One of the key benefits associated with the operation of a docker swarm is the high level of availability offered for applications. In a docker swarm, there are typically several worker nodes and at least one manager node that is responsible for handling the worker nodes' resources efficiently and ensuring that the cluster operates efficiently.

Monitor and analyze Docker containers

Docker Swarm Definitions

Docker Swarm Explained: To contextualize our understanding of a Docker swam, let's take a step back and define some of the more basic terms surrounding containers and the docker application.

Docker is a software platform that enables software developers to easily integrate the use of containers into the software development process. The Docker platform is open source and available for Windows and Mac, making it accessible for developers working on a variety of platforms. The application provides a control interface between the host operating system and containerized applications.

Containers and their utilization and management in the software development process

are the main focus of the docker application. Containers allow developers to package applications with all of the necessary code and dependencies that are necessary for them to function in any computing environment. As a result, containerized applications run reliably when moved from one computing environment to another. In the docker application, a container is launched by running an image.

An Image is a package of executable files that contains all of the code, libraries, runtime, binaries and configuration files necessary to run an application. A container can be described as the runtime instance of an image.

A Dockerfile is the name given to the type of file that defines the contents of a portable image. Imagine you were going to write a program in the Java programming language. Your computer does not understand Java on its own, so you'll need a way to convert your code into machine code. The libraries, configuration files, and programs needed to do this are collectively called the "Java Runtime Environment (JRE)." In Docker, all of these assets would be included in the Dockerfile.

So, instead of installing the JRE onto your computer, you could simply download a portable JRE as an image and include it in the container with your application code. When launching the application from the container, all of the resources necessary for the application to run smoothly will be present in the isolated containerized environment.

Docker Swarm Explained: Docker Swarm 101

With these four definitions, we can cobble together an understand of how software developers are benefiting from the **Docker** application. **Containers** are a desirable method for packaging an application, as they enable consistent performance regardless of the computer platform used to run the application. A **container** is launched by running an **image**, the data for which is stored in a **Dockerfile**.

What are the two types of Docker Swarm mode services?

Docker Swarm has two types of services: replicated and global.

Replicated services: Swarm mode replicated services functions by you specifying a number of replica tasks for the swarm manager to assign to available nodes.

Global services: Global services function by using the swam manager to schedule one task to each available node that meets the services constraints and resource requirements.

What are Docker Swarm Nodes?

A docker swarm is comprised of a group of physical or virtual machines operating in a cluster. When a machine joins the cluster, it becomes a node in that swarm. The docker

swarm function recognizes three different types of nodes, each with a different role within the docker swarm ecosystem:

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.

Leader Node

When a cluster is established, the Raft consensus algorithm is used to assign one of them as the "leader node". The leader node makes all of the swarm management and task orchestration decisions for the swarm. If the leader node becomes unavailable due to an outage or failure, a new leader node can be elected using the Raft consensus algorithm.

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.

Docker Swarm Benefits: Do I need Docker Swarm?

We're seeing an increasing number of developers adopt the Docker engine and use docker swarms to more efficiently produce, update and operate applications. Even software giants like Google are adopting container-based methodologies like docker swarm. Here are three simple reasons why Docker Swarms are becoming more popular:

Leverage the Power of Containers

Developers love using docker swarm because it fully leverages the design advantages offered by containers. Containers allow developers to deploy applications or services in self-contained virtual environments, a task that was previous the domain of virtual machines. Containers are proving a more lightweight version of virtual machines, as their architecture allows them to make more efficient use of computing power.

Docker Swarm Helps Guarantee High Service Availability

One of the main benefits of docker swarms is increasing application availability through redundancy. In order to function, a docker swarm must have a swarm manager that can assign tasks to worker nodes. By implementing multiple managers, developers ensure that the system can continue to function even if one of the manager nodes fails. Docker recommends a maximum of seven manager nodes for each cluster.

Automated Load-Balancing

Docker swarm schedules tasks using a variety of methodologies to ensure that there are enough resources available for all of the containers. Through a process that can be described as automated load balancing, the swarm manager ensures that container workloads are assigned to run on the most appropriate host for optimal efficiency.

Monitor and analyze Docker containers

See how Sumo Logic helps you monitor, troubleshoot and analyze your Docker environment in real time.

Start free trial