Before the inception of [Docker,](https://www.simplilearn.com/tutorials/docker-tutorial) developers predominantly relied on virtual machines. But unfortunately, virtual machines lost their popularity as it was proven to be less efficient. Docker was later introduced and it replaced VMs by allowing developers to solve their issues efficiently and effectively.

Before getting started with what Docker Swarm is, we need to first understand what Docker is as a platform.

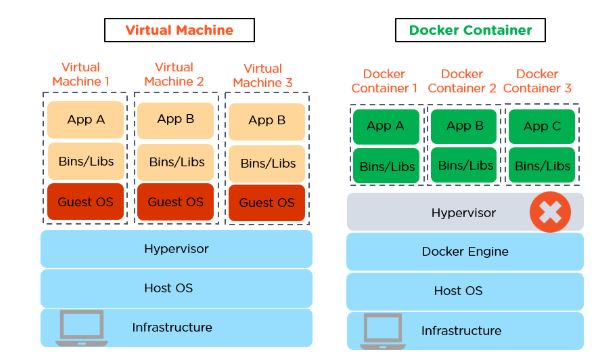
First, let’s dive into what Docker is before moving up to what docker swarm is.

What is Docker and Docker Container?

[Docker](https://www.simplilearn.com/tutorials/docker-tutorial/what-is-docker) is a tool used to automate the deployment of an application as a lightweight container so that the application can work efficiently in different environments.



[Docker container](https://www.simplilearn.com/tutorials/docker-tutorial/what-is-docker-container) is a lightweight software package that consists of the dependencies (code, frameworks, libraries, etc.) required to run an application.

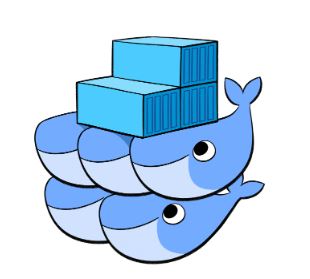


We can use Docker Swarm to make Docker work across multiple nodes, allowing them to share containers with each other. It's an environment where you can have various [Docker images](https://www.simplilearn.com/tutorials/docker-tutorial/docker-images) running on the same host operating system.

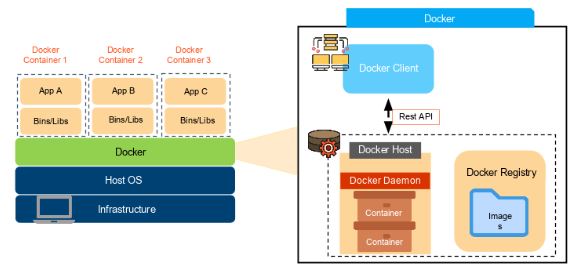
Now, that we have understood what Docker and Docker containers are, let us next look into what Docker swarm is.

What is Docker Swarm?

Docker Swarm is an orchestration management tool that runs on Docker applications. It helps end-users in creating and deploying a cluster of Docker nodes.



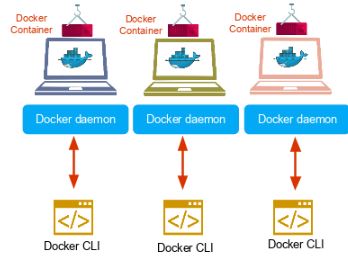
Each node of a Docker Swarm is a Docker daemon, and all Docker daemons interact using the Docker API. Each container within the Swarm can be deployed and accessed by nodes of the same cluster.



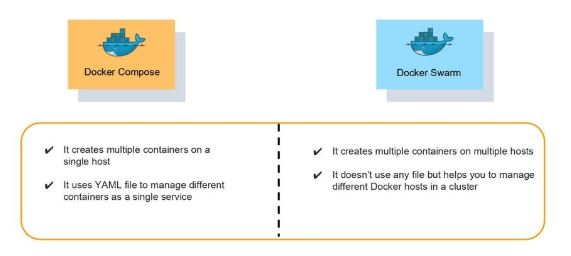
There are five critical elements within a doctor environment:

1. Docker container
2. Docker daemon
3. Docker images
4. Docker client
5. Docker registry

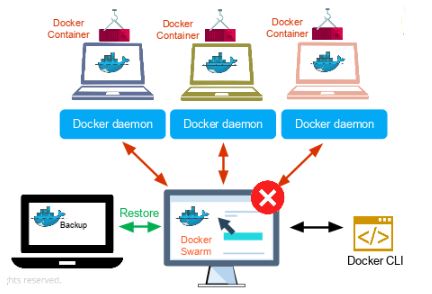
Consider an environment having Docker containers as shown:



If one of the containers fails, we can use the Swarm to correct that failure.



Docker Swarm can reschedule containers on node failures. Swarm node has a backup folder which we can use to restore the data onto a new Swarm.



Features of Docker Swarm

Some of the most essential features of Docker Swarm are:

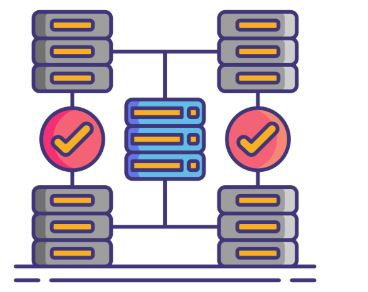
* Decentralized access: Swarm makes it very easy for teams to access and manage the environment
* High security: Any communication between the manager and client nodes within the Swarm is highly secure
* Autoload balancing: There is autoload balancing within your environment, and you can script that 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

Let us extend our learning on what is Docker swarm, let us look into the swarm mode key concepts

Swarm Mode Key Concepts

Service and Tasks

* Docker containers are launched using services.
* Services can be deployed in two different ways - global and replicated.



Global And Replicated Service

* 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.

Node

* A Swarm node is an instance of the Docker engine.
* It is possible to run multiple nodes on a single server. But in production deployments, nodes are distributed across various devices.

How Does Docker Swarm Work?

In Swarm, containers are launched using [services](https://docs.docker.com/engine/swarm/how-swarm-mode-works/services/). A service is a group of containers of the same image that enables the scaling of applications. Before you can deploy a service in Docker Swarm, you must have at least one node deployed.

There are [two types of nodes](https://docs.docker.com/engine/swarm/how-swarm-mode-works/nodes/) in Docker Swarm:

1. Manager node. Maintains cluster management tasks
2. Worker node. Receives and executes tasks from the manager node

Consider a situation where a manager node sends out commands to different worker nodes.



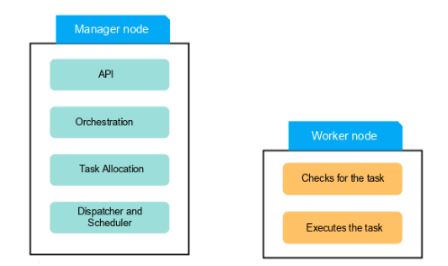
The manager node knows the status of the worker nodes in a cluster, and the worker nodes accept tasks sent from the manager node. Every worker node has an agent that reports on the state of the node's tasks to the manager. This way, the manager node can maintain the desired state of the cluster.

The worker nodes communicate with the manager node using API over HTTP. In Docker Swarm, services can be deployed and accessed by any node of the same cluster. While creating a service, you'll have to specify which container image you're going to use. You can set up commands and services to be either global or replicated: a global service will run on every Swarm node, and on a replicated service, the manager node distributes tasks to worker nodes.

Now a question may arise: don't task and service refer to the same thing? The answer is no.

A service is a description of a task or the state, whereas the actual task is the work that needs to be done. Docker enables a user to create services that can start tasks. When you assign a task to a node, it can't be assigned to another node. It is possible to have multiple manager nodes within a Docker Swarm environment, but there will be only one primary manager node that gets elected by other manager nodes.

Therefore, the working of the Swarm can be summarized as follows:



A service is created based on the [command-line](https://docs.docker.com/swarm/reference/) interface. The API that we connect in our Swarm environment allows us to do orchestration by creating tasks for each service. The task allocation will enable us to allocate work to tasks via their IP address. The dispatcher and scheduler assign and instruct worker nodes to run a task. The Worker node connects to the manager node and checks for new tasks. The final stage is to execute the tasks that have been assigned from the manager node to the worker node.

As we have got a better understanding of what is Docker Swarm, let us next look into the differences between Docker swarm and Kubernetes.

Kubernetes vs. Docker Swarm

The table below illustrates the differences between [Kubernetes](https://www.simplilearn.com/tutorials/kubernetes-tutorial) vs. Docker Swarm:

|  |  |  |
| --- | --- | --- |
| Features | Kubernetes | Docker Swarm |
| Installation | Complex | Simple |
| Load Balancing | Manual intervention is required for load balancing | Automated load balancing |
| Scalability | Scaling and deployment are comparatively slower | Containers are deployed much faster |
| Cluster | Difficult to set-up | Easy to set-up |
| Container Setup | Commands like YAML should be rewritten while switching platforms | A container can be easily deployed to different platforms |
| Logging and monitoring | Consists of built-in tools to manage both processes | Tools are not required for logging and monitoring |
| Availability | High availability when pods are distributed among the nodes | Increases availability of applications through redundancy |
| Data volumes | Shared with containers from the same pod | Can be shared with any container |

To strengthen our understanding of what Docker swarm is, let us look into the demo on the docker swarm.

Demo On Docker Swarm For Beginners

This tutorial “What is Docker Swarm” requires two hosts, which can either be Virtual Machine or AWS EC2.

The demo shows how to build and deploy a Docker Engine, run Docker commands, and install Docker Swarm.

Prerequisites:

* Ubuntu 64-bit operating system
* An ubuntu  account with Sudo privileges
* Command-line terminal
* Docker software repositories (optional)

Step 1: Update Software Repositories

Run the following command on the terminal:

sudo apt-get update

Step 2: Uninstall Old Versions of Docker

Before proceeding, uninstall the old Docker software and use the following command:

sudo apt-get remove docker docker-engine docker.io

Step 3: Install Docker

To install [Docker on Ubuntu](https://www.simplilearn.com/tutorials/docker-tutorial/how-to-install-docker-on-ubuntu),  run the following command:

sudo apt install docker.io

Step 4: Set-up Docker

Set-up and run Docker service by entering the following commands in the terminal window:

sudo systemctl start docker

sudo systemctl enable docker

Step 5: Verify Docker Version

To check the installed Docker version, enter the following command:

sudo docker --version

Step 6: Run Docker Container

To run a Docker container, it’s important to pull a Docker Image (such as MySQL) from Docker Hub.

sudo docker pull mysql

sudo docker run -d -p 0.0.0.0:80:80 mysql:latest

Now, Docker pulls the latest MySQL image from the hub.

List down all the available Docker images on your machine by using the following command:

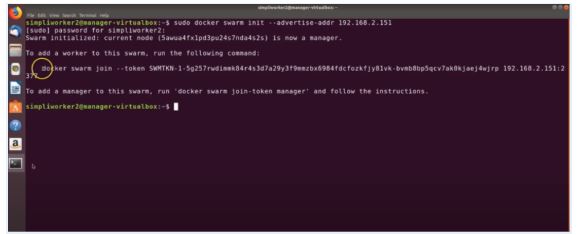
sudo docker ps -a

Step 7: Create Swarm

Here, create a cluster with the IP address of the manager node.

sudo Docker Swarm init --advertise-addr 192.168.2.151

Subsequently, you should see the following output:



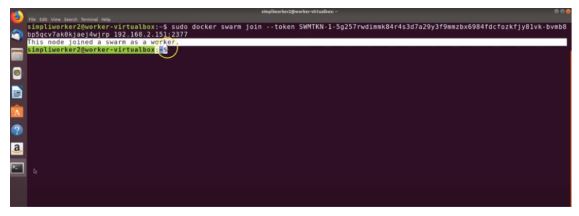
Manager Node

This means that the manager node is successfully configured.

Now, add worker node by copying the command of the “swarm init” and paste the output onto the worker node:

sudo Docker Swarm join --token SWMTKN-1- xxxxx

Your worker node is also created if you see the following output:

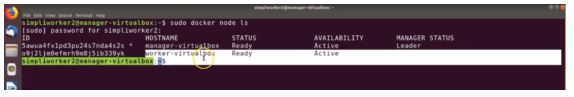


Worker Node

Now, go back to the manager node and execute the following command to list the worker node:

sudo docker node ls

Here, you must see the worker node in the following output:



Swarm Cluster - Docker Swarm

The above image shows you have created the Swarm Cluster successfully. Now, launch the service in Swarm Mode.

Go to your the manager node and execute the command below to deploy a service:

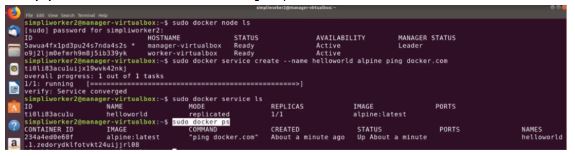
sudo docker service create --name HelloWorld alpine ping docker.com

By executing the above command, you can access the HelloWorld file from the remote system.

To see the output, you can check the services with the following command:

sudo docker service ls

Finally, you should be able to see the following output:



Service Created - Docker Swarm

And that’s it! Well done, you have successfully installed and configured the Swarm cluster on Ubuntu 16.04. Also, whenever required, you can effortlessly scale your application with no performance issues.

This brings us to the conclusion of the article what is Docker Swarm

A swarm is a group of machines that are running Docker and joined into a cluster

Docker Swarm is a tool for Container Orchestration

Let’s take an example

You have 100 containers

You need to do

- Health check on every container

- Ensure all containers are up on every system

- Scaling the containers up or down depending on the load

- Adding updates/changes to all the containers

Orchestration - managing and controlling multiple docker containers as a single service

Tools available - Docker Swarm, Kubernetes, Apache Mesos

Pre-requisites

1. Docker 1.13 or higher

2. Docker Machine (pre installed for Docker for Windows and Docker for Mac)https://docs.docker.com/machine/insta...

https://docs.docker.com/get-started/p...

Step 1 : Create Docker machines (to act as nodes for Docker Swarm) Create one machine as manager and others as workers

docker-machine create --driver hyperv manager1

docker-machine create --driver virtualbox manager1

docker-machine create -d virtualbox --virtualbox-memory=1024 --virtualbox-cpu-count=1 --virtualbox-disk-size=4096 --virtualbox-no-vtx-check manager

docker-machine:Error with pre-create check: “exit status 126”

https://stackoverflow.com/questions/3...

brew cask install virtualbox;

Create one manager machine and other worker machines

docker-machine create -d virtualbox --virtualbox-memory=1024 --virtualbox-cpu-count=1 --virtualbox-disk-size=40960 --virtualbox-no-vtx-check manager1

Step 2 : Check machine created successfully

docker-machine ls

docker-machine ip manager1

Step 3 : SSH (connect) to docker machine

docker-machine ssh manager1

Step 4 : Initialize Docker Swarm

docker swarm init --advertise-addr 192.168.99.109

docker node ls

(this command will work only in swarm manager and not in worker)

Step 5 : Join workers in the swarm

Get command for joining as worker

In manager node run command

docker swarm join-token worker

This will give command to join swarm as worker

docker swarm join-token manager

This will give command to join swarm as manager

SSH into worker node (machine) and run command to join swarm as worker

In Manager Run command - docker node ls to verify worker is registered and is ready

Do this for all worker machines

docker node ls

Step 6 : On manager run standard docker commands

docker info

check the swarm section

no of manager, nodes etc

Now check docker swarm command options

docker swarm

Step 7 : Run containers on Docker Swarm

docker service create --replicas 3 -p 80:80 --name vilas\_nginx\_swarm nginx

Check the status:

docker service ls

docker service ps serviceName

Check the service running on all nodes

Check on the browser by giving ip for all nodes

Step 8 : Scale service up and down

On manager node

docker service scale serviceName=2

Inspecting Nodes (this command can run only on manager node)

docker node inspect nodename

docker node inspect self

docker node inspect worker1

Step 9 : Shutdown node

docker node update --availability drain worker1

Step 10 : Update service

docker service update --image imagename:version web

docker service update --image nginx:1.14.0 serviceName

Step 11 : Remove service

docker service rm serviceName

docker swarm leave : to leave the swarm

docker-machine stop machineName : to stop the machine

docker-machine rm machineName : to remove the machine

REFERENCES:

https://docs.docker.com/get-started/p...

https://rominirani.com/docker-swarm-t...

FAQs & Helpful Tips:

A swarm is a group of machines that are running Docker and joined into a cluster

A cluster is managed by swarm manager

The machines in a swarm can be physical or virtual. After joining a swarm, they are referred to as nodes

Swarm managers are the only machines in a swarm that can execute your commands, or authorise other machines to join the swarm as workers

Open a PowerShell console as an administrator. Install docker-machine using Chocolatey as follows:

PS> choco install -y docker-machine

Create a VM called boot2docker in Hyper-V with the following command:

PS> docker-machine create --driver hyperv --hyperv-virtual-switch "My Internal Switch" boot2docker

Note, you must run the preceding command in administrator mode or it will fail.

You should see ...

docker-machine create -d virtualbox --virtualbox-memory=2048 --virtualbox-cpu-count=1 --virtualbox-disk-size=5120 --virtualbox-no-vtx-check manager

docker-machine create -d virtualbox manager