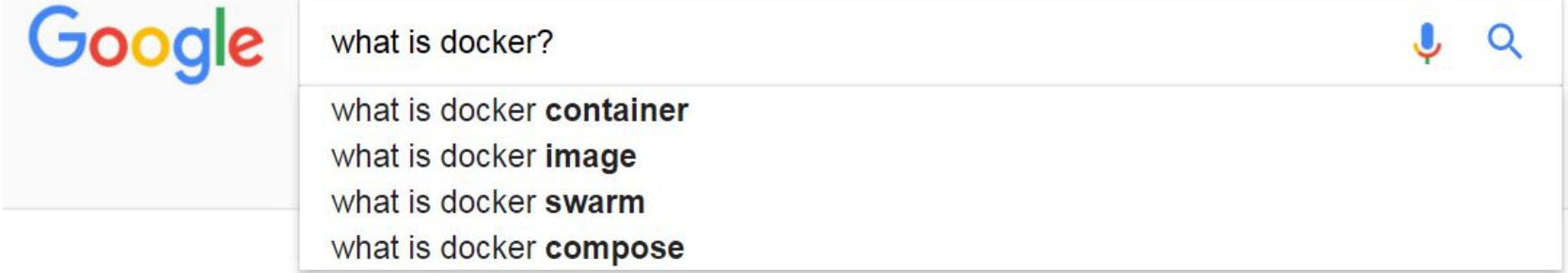
# docker



# Docker

Docker is a platform for developers and sysadmins to develop, deploy, and run applications with containers.

## How does docker work?

- Uses Linux kernel
- Image based deployment

## Where to use docker?

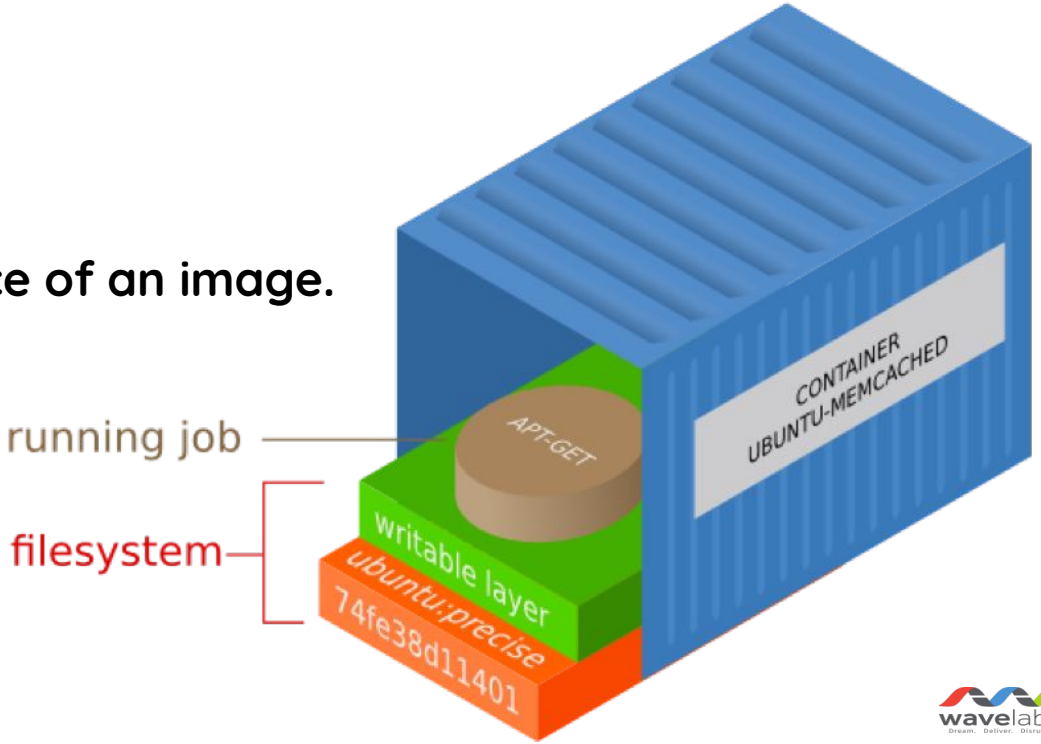
- To eliminate “works on my machine” problems

## Image

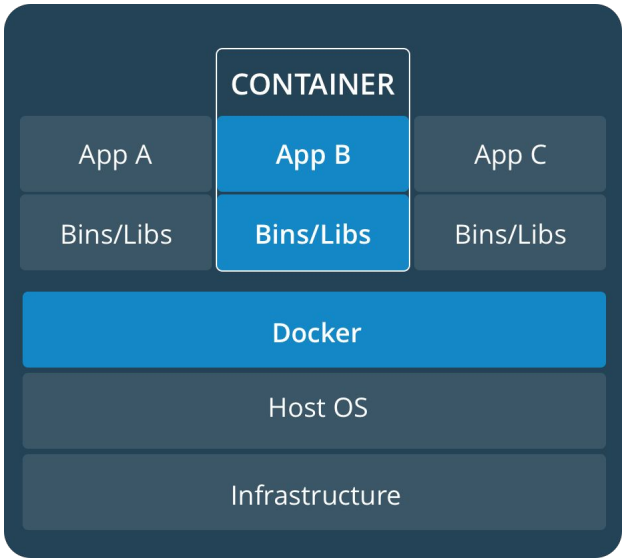
A lightweight, stand-alone, executable package containing code, a runtime, libraries etc.

## Container

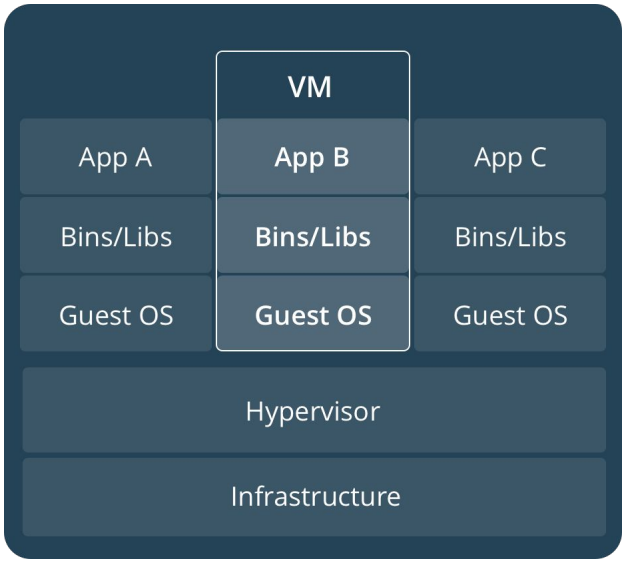
A container is a runtime instance of an image.



# Containers Vs Virtual Machines




Container




Virtual Machine

- Share the OS kernel with other containers
- Take up less space (tens of MBs)

- Each VM includes a full copy of an operating system
- Take up More space (tens of GBs)

# Docker Image Commands

**Display list of Docker images:**

docker images

**Pull a docker Image:**

**Syntax:** docker pull image_name

**Example:** docker pull ubuntu

**Remove a docker image:**

**Syntax:** docker rmi image_name

**Example:** docker rmi ubuntu

**Run a Container with a docker image:**

**Syntax:** docker run [OPTIONS]  image_name

**Example:** docker run --name ubuntu_container -p 9889:9889 ubuntu:14.04

# Docker Container Commands

**Display list of Running containers:**

docker ps

**Stop a Running container:**

docker stop CONTAINER

**Display list of all containers:**

docker ps -a

**Remove a stopped container:**

docker rm CONTAINER

**Run a Command in a running container:**

**Syntax:** docker exec [OPTIONS]  CONTAINER COMMAND [ARGS...]

**Example:** docker exec -it mysql_container /bin/bash

## Dockerfile

File containing instructions  the build the image.


## Docker-compose file

The Compose file is a YAML file defining services, networks and volumes.

## Docker Hub

Docker Hub is a registry service on the cloud that allows

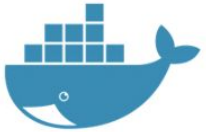
- Download Docker images that are built by other communities
- Can also upload your own Docker built images

## Docker Registry


- Enterprise-grade image storage solution from Docker.

- Install it behind your firewall so that you can securely store and manage the Docker images

# Basic taxonomy in Docker




A **Registry**
Stores many static images


**Images**
Static, persisted container image


**Container**
Image-instance running an app process (service/web)

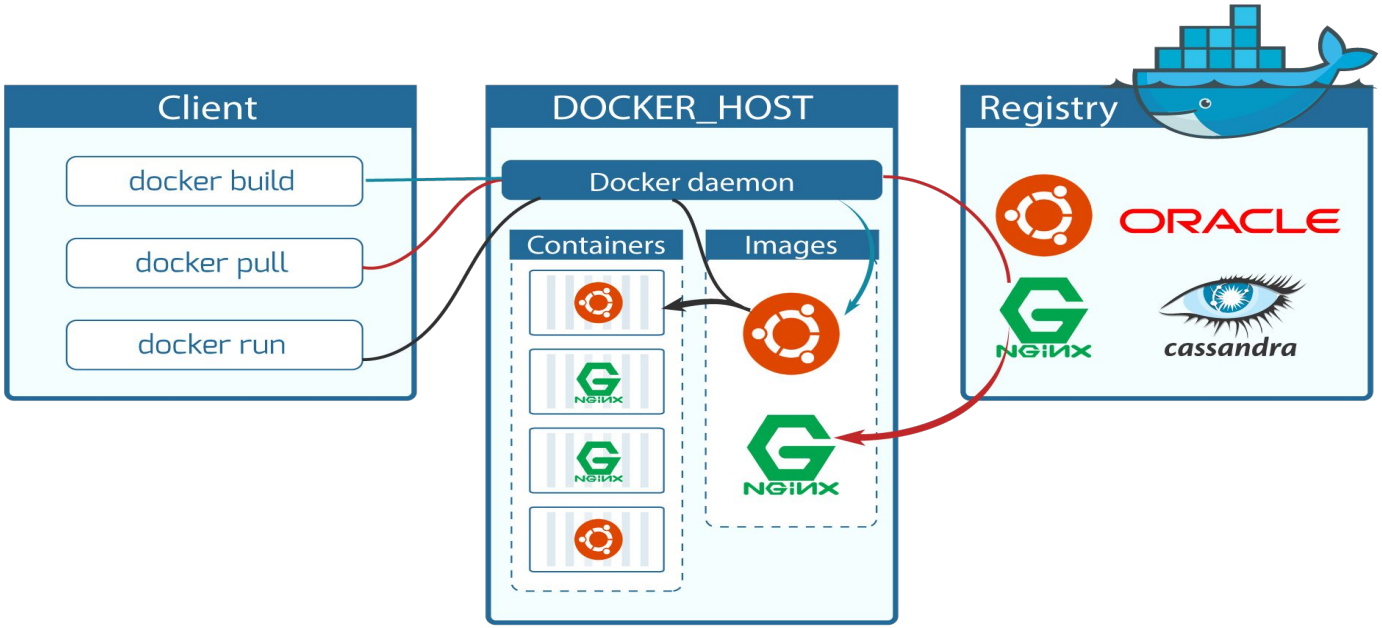
- **Hosted Docker Registry**
- **Docker Trusted Registry on-prem.**

→ **On-premises** (‘n’ private organizations)

- **Docker Hub Registry**
- Docker Trusted Registry on-cloud
- **Azure Container Registry**
- AWS Container Registry
- Google Container Registry
- Quay Registry
- Other Cloud

→ **Public Cloud** (specific vendors)

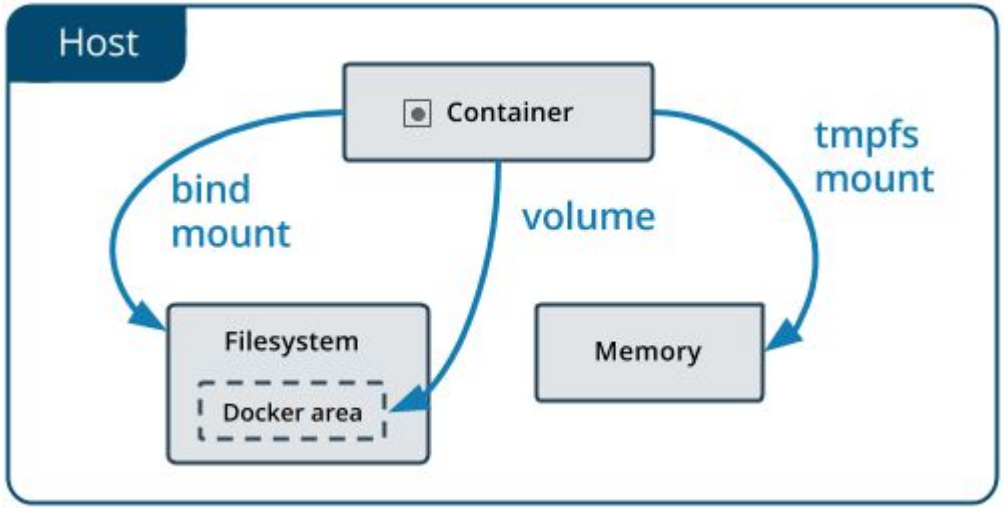
# DOCKER COMPONENTS



# Data Management in Docker

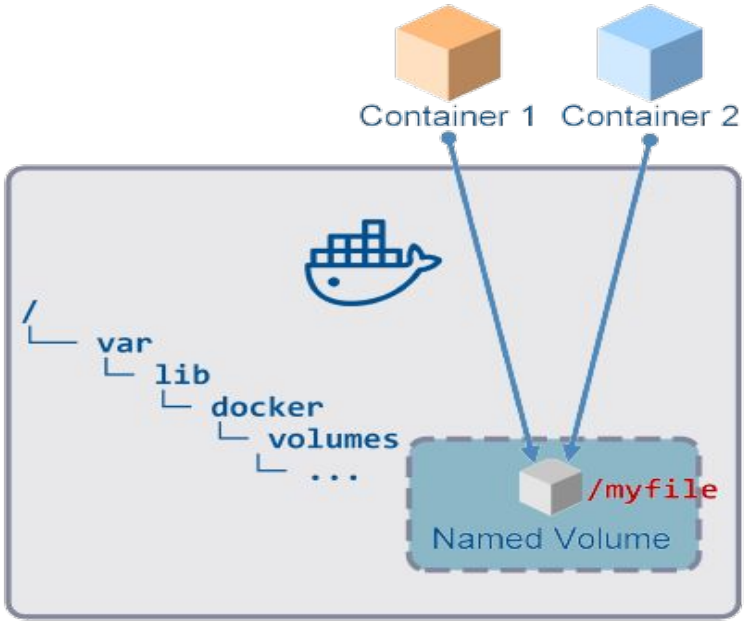
Docker has two options for containers to store files in the Host Machine:

- Volumes
- Bind Mounts



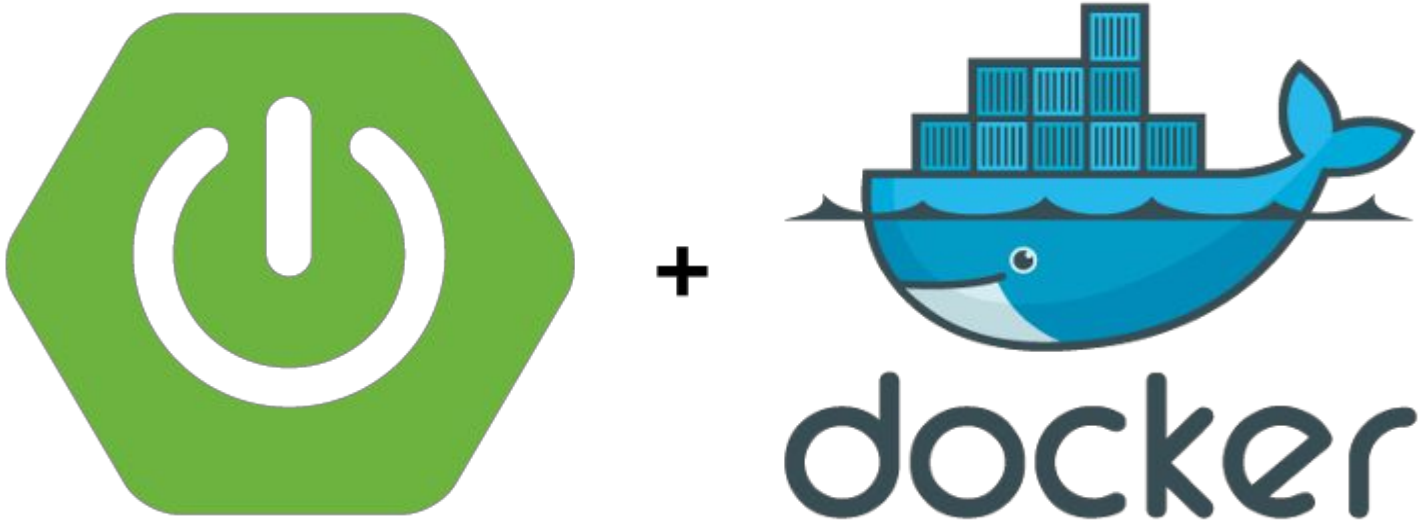
# Docker Volumes

- Volumes are the preferred mechanism for persisting data generated by and used by Docker containers.
- Created and managed by Docker.
- Volumes will be created at /var/lib/docker/volumes/



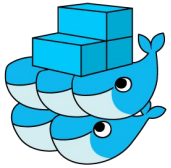
# Docker Bind Mounts

- Bind mounts can be stored anywhere on the host system.
- *tmpfs* mounts are stored in the host system's memory only




# Spring boot Application deployment in Docker

# Docker Swarm



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

- **Swarm Manager**

  Machine which initializes the Swarm

- **Workers**

  Machines which join in the Swarm

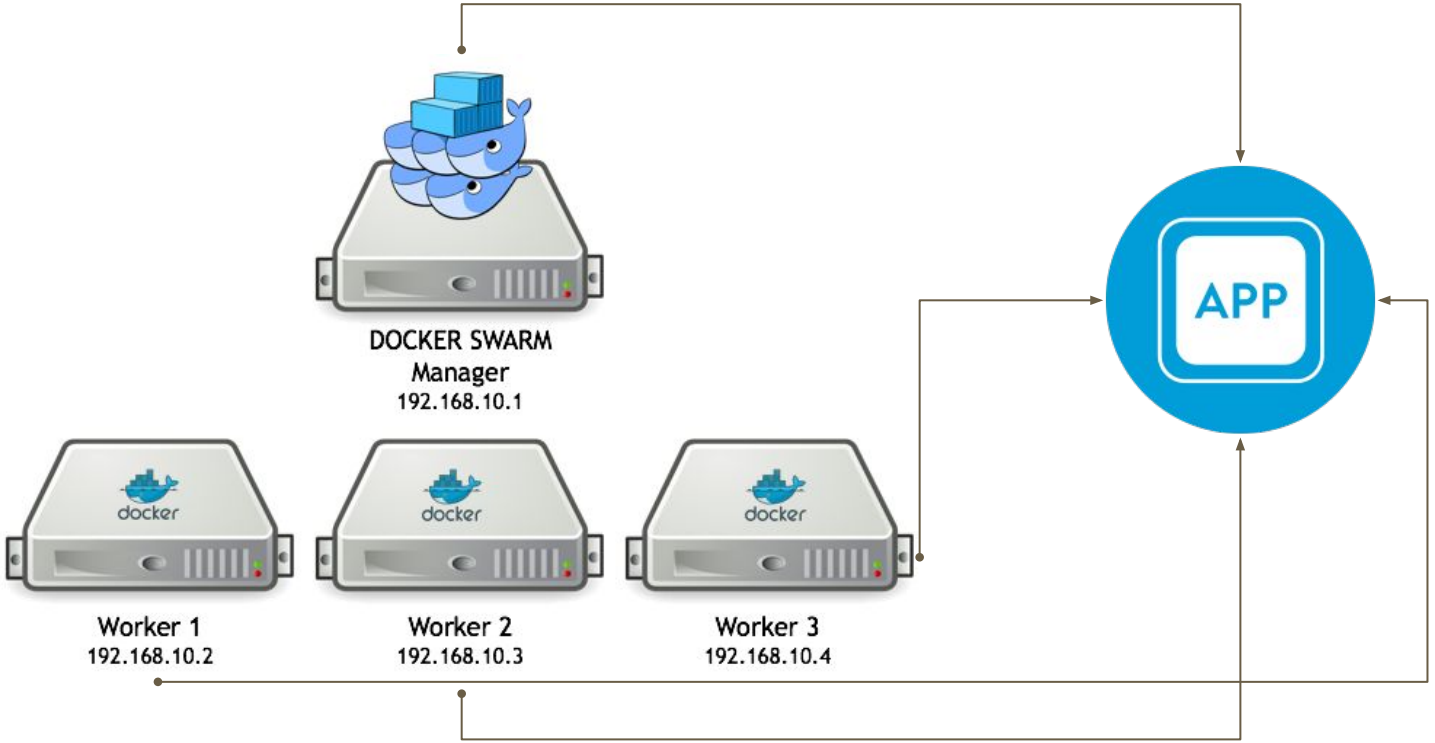
## Steps involved

**Step 1:**
Swarm manager initializes the Swarm and generates a token

**Step 2:**
Worker joins in the Swarm using the token

**Step 3:**
Swarm manager deploys the application

# Docker Swarm Architecture



# Creating a Virtual Machine

```
muralikrishnak@NBTR1 MINGW64 ~
$ docker-machine create --driver virtualbox jarvis
Running pre-create checks...
Creating machine...
(jarvis) Copying C:\Users\muralikrishnak\.docker\machine\cache\boot2docker.iso to C:\Users\muralikrishnak\.docker\machine\machines\jarvis\boot2docker.iso...
(jarvis) Creating VirtualBox VM...
(jarvis) Creating SSH key...
(jarvis) Starting the VM...
(jarvis) Check network to re-create if needed...
(jarvis) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: C:\Program Files\Docker Toolbox\docker-machine.exe env jarvis
```

# Virtual Machines

```
muralikrishnak@NBTR1 MINGW64 ~
$ docker-machine ls
NAME      ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER        ERRORS
default   *        virtualbox   Running   tcp://192.168.99.100:2376           v17.06.0-ce
genisys   -        virtualbox   Running   tcp://192.168.99.104:2376           v17.06.0-ce
jarvis    -        virtualbox   Running   tcp://192.168.99.103:2376           v17.06.0-ce
unicorn   -        virtualbox   Running   tcp://192.168.99.105:2376           v17.06.0-ce
```

## Creating the swarm

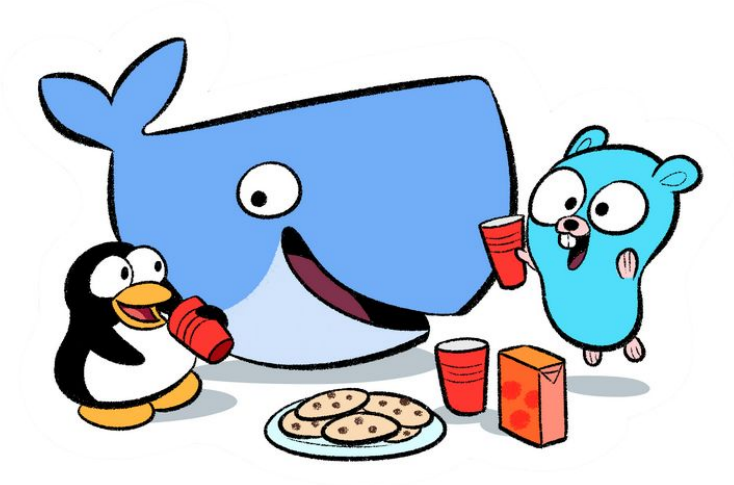
```
muralikrishnak@NBTR1 MINGW64 ~
$ docker-machine ssh jarvis "docker swarm init --advertise-addr 192.168.99.103:2377"
Swarm initialized: current node (8lws5ooacsu16fow8dvakmnyo) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-1wqdf63afpl5y578kgx9rno5v4t9zvit4bs40vk6t03a63a1eg-8lbw0ddl52ucf81w54ys3l6u0 192.168.99.103:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

## Worker Joining in the swarm

```
muralikrishnak@NBTR1 MINGW64 /e/docker/swarm
$ docker-machine ssh genisys "
 docker swarm join --token SWMTKN-1-1wqdf63afpl5y578kgx9rno5v4t9zvit4bs40vk6t03a63a1eg-8lbw0ddl52ucf81w54ys3l6u0 192.168.99.103:2377"
```

## Swarm Nodes

```
muralikrishnak@NBTR1 MINGW64 ~
$ docker-machine ssh jarvis "docker node ls"
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
8lws5ooacsu16fow8dvakmnyo *   jarvis              Ready               Active              Leader
jbjl3qjwnlrtyba6ynlj6pfz3     unicorn             Ready               Active
oz0pfcg94cp9azn7jbw079pgl     genisys             Ready               Active
```



# THANK YOU!