# Docker Swarm

## Objective

* Understand what Docker Swarm is
* Why Docker Swarm is used?
* How to implement and use a Docker Swarm

### **Pre-requisites**

* 3 x Ubuntu VM's running version 18.04 LTS   
  (Spun up with your choice of cloud provider).
* Start a Terminal window on your main PC to simulate a node server and then run the following command to start an Ubuntu server with Docker capabilities (see the **–privileged** option). You will then a second command to shell into the server in an interactive session. **docker run --privileged --name ubuntu1 -d docker:20.10-dind  
    
  docker exec -it ubuntu1 sh**
* Start a Terminal window to start another server called **ubuntu2   
    
  docker run --privileged --name ubuntu2 -d docker:20.10-dind  
    
  docker exec -it ubuntu2 sh**
* Start a Terminal window to start another server called **ubuntu3  
    
  docker run --privileged --name ubuntu2 -d docker:20.10-dind  
    
  docker exec -it ubuntu2 sh**
* Docker is installed on Ubuntu1-3 because of the image **dind**

### **Create Cluster**

Put together a cluster with 3 nodes

* On your main PC which you have chosen to be the Swarm Manager run the following command:

**docker swarm init**

This will initialise the Swarm Manager and Docker will output a Join Token. You will need this command to add workers to your Swarm.

A black screen with white text

Description automatically generated

* Copy the line **docker swarm join –token** …….  
    
  **Note**: If you can’t obtain the join command anymore, run the join-token command below on the Swarm Manager VM (your main PC in this lab) to retrieve the token: **docker swarm join-token worker**
* Log on to the Virtual Machine's that will be worker node's in the Swarm (ubuntu 1,2 & 3), run the **join command** that you copied above to join the Swarm as a worker.
* Now you have 3 nodes in the Swarm! To check if the Swarm has been configured properly run the following command on your main PC:  
    
  **docker node ls**

### **Create a Service**

* Create a service that uses the **bobcrutchley/python-http-server:latest** image and is named **python-http-server**.
* Run the following command (all on one line) on your manager node   
  (your main PC):  
    
  **docker service create --name python-http-server bobcrutchley/python-http-server:latest**

### **Update the Service**

* Update the service so that there are 10 replicas and the port 9000 (inside the container) has been published to 80 (outside the container). Run the following Command to update the service:  
    
  **docker service update --replicas 10 --publish-add 80:9000 python-http-server**  
    
  The port numbers in the command above 80:9000 represent the Published Port which is 80 and the Container Port 9000.

### **Access the Service**

Use the curl CLI tool to view the info.json file served by the application, this file shows the name of the host that it is running on.

* Run the following command on your main PC:  
  Type **ipconfig**  and copy the IP address  
    
  A screen shot of a computer

  Description automatically generated  
    
  **curl http:*//[YOUR\_PRIVATE\_IP]/info.json***

Substitute the private IP address here with your PC’s private IP address Curl the file multiple times, what do you notice about the output?

### **Remove the Worker Nodes**

Update the amount of replicas to 2. Run the command Below on the manager VM to do this:

**docker service update --replicas 2 python-http-server**

Type **docker node ls** to get the IDs of the running nodes

Drain one of the worker nodes by typing:  
**docker node update --availability drain [NODE\_NAME/id]**

You can get the node names by running **docker node ls**

## **Clean up**

1. Remove Service (Run on Swarm Manager):

**docker service rm python-http-server**

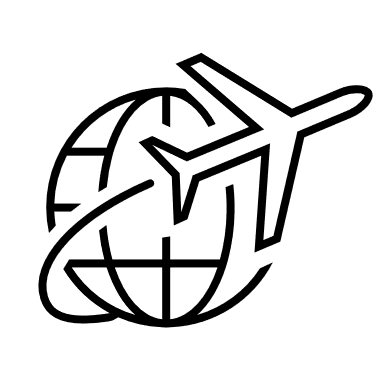
1. On each worker node run the command below to shut it down:

**docker swarm leave**

1. Remove the nodes using the command below on the Swarm Manager: **docker node rm [YOUR NODE NAME]**

Remember to always shut down and remove all VM resources on your cloud provider platform to avoid unexpected charges and fees.

# Go much further!



<https://app.qa.com/lab/docker-swarm-playground/>

**Duration:** 4h

This Docker swarm playground lab provides you with a Docker swarm cluster running in Microsoft Azure. The cluster is comprised of one manager node and two worker nodes. The nodes have docker, docker-compose, and command-line completions pre-installed and ready for you to play with. The cluster allows public access on ports 80, 443, and 8080.