Installation and configuration of Docker and Docker Swarm Cluster

Info: Docker is a set of platform as a service (PAAS) product that uses OS level virtualization to deliver software in packages called container. And 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. It is a container orchestration tool, meaning that it allows the user to manage multiple containers deployed across multiple host machines.

Author: Sarvajeet Singh <sarvajeet.singh@50hertz.in>

Date: 19-Apr-2020, revised _____

Introduction

This document explains how Docker with swarm cluster is set up on a maintained version of CentOS 7. The instructions are aimed at any competent SystemAdministrator.

Server

The Docker Application is hosted at 172.16.0.68 (ci.50hertz.in) which is a local server in our office.

The server configuration is summarised below:

CPU	Intel(R) Core(TM) i3-4160 CPU @ 3.60GHz, 4-Core
RAM	16 GB
DISK	500 GB
OS	CentOS Linux release 7.7.1908
IP	172.16.0.68

Docker Swarm Cluster

Manger Node	172.16.0.68
Worker Node 1	172.16.0.69
Worker Node 2	172.16.0.70

Domain

The **ci.50hertz.in** domain is registered through Net4India.

Installation and configuration of Docker & Docker Swarm

Prerequisites

OS requirements :

To install Docker Engine, you need a maintained version of CentOS 7. The centos-extras repository must be enabled. This repository is enabled by default, but if you have disabled it, you need to re-enable it.

Install using the repository :

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

Install the yum-utils package (which provides the yum-config-manager utility) and set up the stable repository.

yum install -y yum-utils

yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

yum install docker-ce docker-ce-cli containerd.io

And Start the Service

systemctl start docker

Configure Docker Swarm :

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

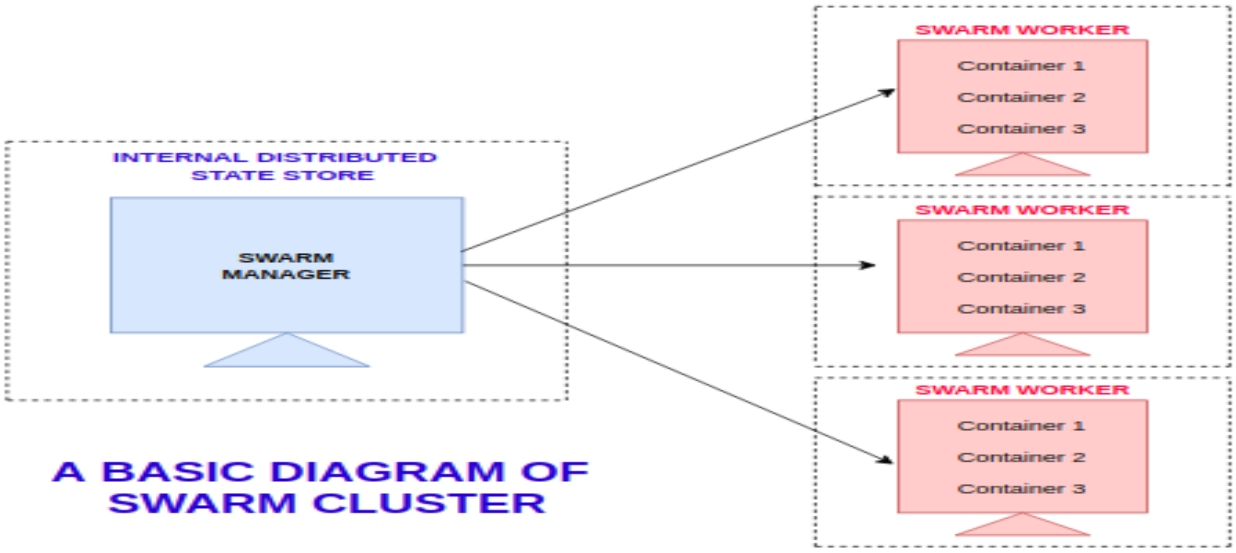
Manager Node, Leader Node, Worker Node

So We choose the two worker node and one manager node for docker swarm and install docker-ce service on them.

Manger or Leader Node : 172.16.0.68

Worker Node : 172.16.0.69

Worker Node : 172.16.0.70



A BASIC DIAGRAM OF SWARM CLUSTER

Step-1:- Create the Manager node in single instance.

docker swarm init --advertise-addr 172.16.0.68

Make sure you see the output below:

Swarm initialized : current node is now a manger .

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

docker swarm join --token SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c <Worker Node IP>:2377

To add a manger to this swarn, run ‘docker swarm join-token manager’ and follow the instruction.

Verify the manager status using the command below:

[root@ip-172 ~]# docker info
Client:
 Debug Mode: false

Server:
 Containers: 15
  Running: 10
  Paused: 0
  Stopped: 5
 Images: 428
 Server Version: 19.03.6

Storage Driver: overlay2
 Backing Filesystem: xfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host ipvlan macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: active
 NodeID: sf9zoulplmm99h6ayxqgnt02u
 Is Manager: true
 ClusterID: xhk2m8fy0f9jkwv2udziqsegh
 Managers: 1
 Nodes: 3
 Default Address Pool: 10.0.0.0/8
 SubnetSize: 24
 Data Path Port: 4789
 Orchestration:
  Task History Retention Limit: 3
 Raft:
  Snapshot Interval: 10000
  Number of Old Snapshots to Retain: 0
  Heartbeat Tick: 1
  Election Tick: 10
 Dispatcher:
  Heartbeat Period: 5 seconds
 CA Configuration:
  Expiry Duration: 3 months
  Force Rotate: 0
 Autolock Managers: false
 Root Rotation In Progress: false
 Node Address: 172.16.0.68
 Manager Addresses:
  172.16.0.68:2377
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: b34a5c8af56e510852c35414db4c1f4fa6172339
runc version: 3e425f80a8c931f88e6d94a8c831b9d5aa481657
init version: fec3683
Security Options:
 seccomp
  Profile: default
Kernel Version: 3.10.0-1062.9.1.el7.x86_64
Operating System: CentOS Linux 7 (Core)
OSType: linux
Architecture: x86_64
CPUs: 4

Total Memory: 15.43GiB
Name: ip-172.16.0.68
ID: K3OX:M6X7:HIS7:YPWS:IVCS:MOGZ:SJN5:IGR2:2D4E:DRTR:5RWO:MZNL
Docker Root Dir: /var/lib/docker
Debug Mode: false
Username: 50hertz
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false

Step 2: - Allow the 2377 port with iptables or firewall.

Step-3: - Now add a worker to this swarm, run the following command:

docker swarm join --token SWMTKN-149nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c **172.16.0.69:2377**

Similarly we can add second node with above command.

docker swarm join --token SWMTKN-149nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c **172.16.0.70:2377**

Step-4: Check the docker node on Manager node

docker node ls

[root@ip-172 ~]# docker node ls
ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION
7hnk4j3urctw95tue1bs9hoil ip-172-16-0-70 Ready Active 19.03.8
sf9zoulplmm99h6ayxqgnt02u * ip-172.16.0.68 Ready Active Leader 19.03.6
xnv2ud7min5xlz61qcgdvxpg3 ip-172.16.0.69 Ready Active 19.03.5

Step-5: Create the single Network for two nodes.

#docker network create --driver overlay --attachable application-network

Step-6: Now it’s time to Create Service of portainer agent for two nodes

Note : The Portainer Agent is a workaround for a Docker API limitation when using the Docker API to manage a Docker environment.Simply you can say its a UI version for dcoker-cli.

#docker service create --name portainer_agent --network portainer_agent_network --publish mode=host,target=9001,published=9001 --mode global --mount type=bind,src=//var/run/docker.sock,dst=/var/run/docker.sock --mount type=bind,src=//var/lib/docker/volumes,dst=/var/lib/docker/volumes portainer/agent

Step-7: Run the Portainer on local system for Access UI.

#docker run -d -p 8000:8000 -p 9100:9000 -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer

Now deploy A jar through the docker as a container :

Step-8: You need to create a image first from the jar file. We are creating the image through the Docker file. Your docker file should be look like as below :

cat Dockerfile

FROM openjdk:8-jdk-alpine

ARG JAR_FILE=/target/schedule-batch-0.0.1-SNAPSHOT.jar

COPY ${JAR_FILE} schedule-batch-0.0.1-SNAPSHOT.jar

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-Duser.timezone=IST","-jar","/schedule-batch-0.0.1-SNAPSHOT.jar"]

docker build -t schedule-batch .

The above command will create the image

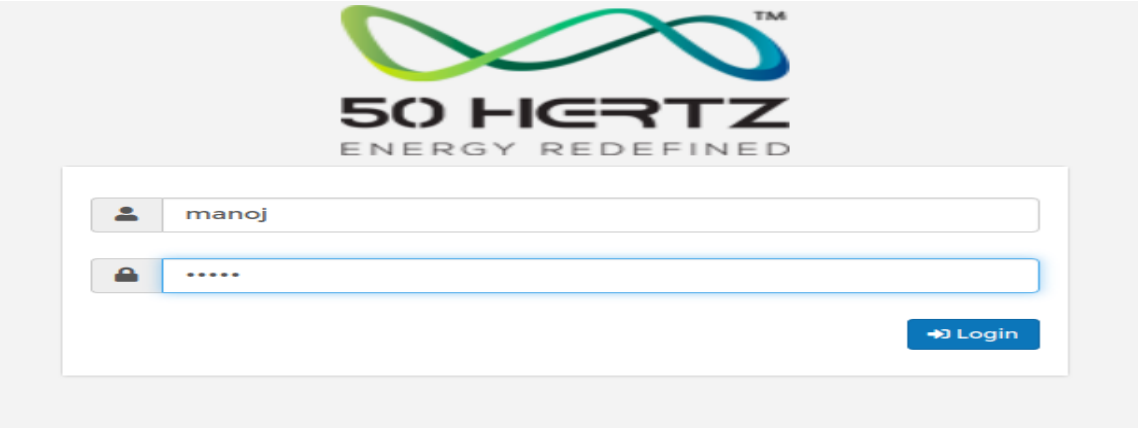
Step-9: Now push the created docker image into docker hub with below command.

docker push 50hertz/energy:schedule-batch

Now it’time to run this service :

Step-10: login the portainer UI and set endpoints of docker swarm

http://172.16.0.68:9100



Step-11:- After login Go to endpoint option and click endpoint.

Step-12:- click the add endpoint and give some information.

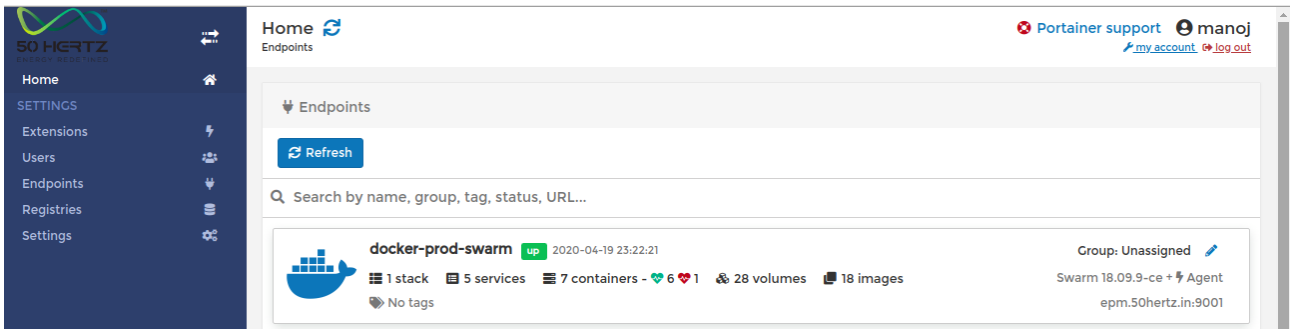
Name:- Docker swarm Prod

Endpoint URL:- Docker swarm IP:9001

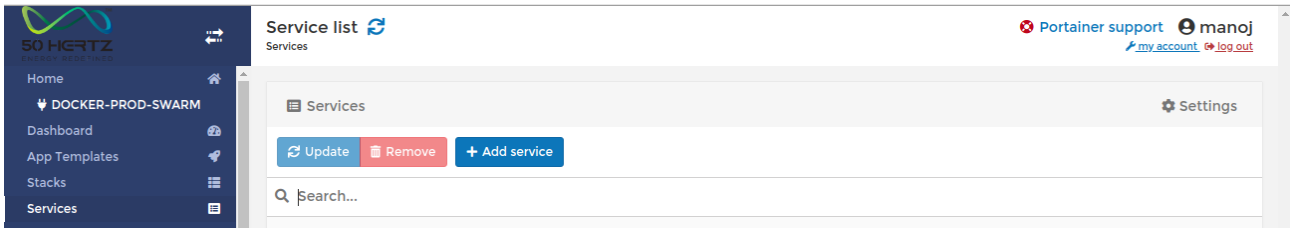
Public IP:- Docker swarm Public IP

=> Click the add endpoint.

Step13:- Again go to home tab and show this page.



Step14:- Now it's time to create the service. Click the docker prod-swarm and create service to Deploy jar image into docker-swarm-prod.



Step15:- Click the Add Service and give the name for create docker image.

Name:- Deploy Application name

Registry:- Docker Hub(docker hub credential set in registries option)

Image:- 50hertz/energy:schedule-batch

Sceduling-mode:- Replicated

Network:- application-network

config:- if you required the create config file under config option.

Optional

Bind Mount *: A bind mount is a file or folder stored anywhere on the container host filesystem, mounted into a running container.*

Volume:- bind volume source to destination path if you have required.

Step16:- Again Click the Service option to check the image.

So these are all steps has taken to deploy a service on docker swarm cluster.