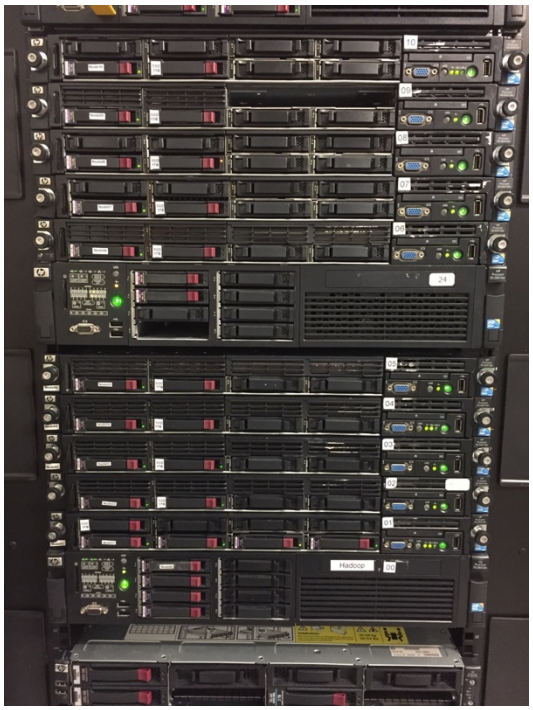
Hands-On Exercise 3-2 Deploying Multi-Container Apps on multi-node with Swarm

1 CSCluster at UIS



11-node Hadoop nodes

- CPU: Intel Xeon **4C/8T** per node
- Memory: **266 GB** in total
  - Master (head) node: 74GB RAM
  - 2nd Master node: 48GB RAM
  - 9 worker nodes: 16GB/node
- Storage: **10 TB SSD** in total
  - Master node: 400GB,
  - 10 worker nodes: 1TB SSD/node
- Hadoop: **Cloudera CDH 6.3**
  - HDFS, MapReduce
  - Spark 2, HBase, Hive
  - Pig, HUE, and etc.

VMs for Docker/K8S (You will use these VMs)

- CPU: Intel Xeon **8** core/node
- Memory: **16GB** RAM/node
- Storage: **100GB**

3-node Cassandra cluster (NoSQL-Column family DB)

- CPU: Intel Xeon **4C/8T** per node
- Memory: **48GB** RAM (16GB/node)
- Storage: **3TB SSD** (1TB/node)

PostgreSQL node (Relational DB)

- CPU: Intel Xeon **4C/8T** per node
- Memory: **18 GB RAM**
- Storage: **1TB SSD**

MongoDB node (NoSQL-Document DB)

- CPU: Intel Xeon **4C/8T** per node
- Memory: **16 GB RAM**
- Storage: **1TB SSD**

2 Accessing your VM for Docker and Kubernetes

2.1 Accessing campus network

If you are in UIS campus, you are fine. If you are not in UIS campus, you should install a Cisco VPN client software. The VPN client software gives remote users a secure and encrypted VPN (Virtual Private Network) connection to the UIS campus network. Please see below website, https://www.uis.edu/informationtechnologyservices/connect/vpn/

2.2 Accessing your VM using SSH

After you install the VPN client software and make a VPN connection to UIS network, you can access your VM using a terminal (Mac), PowerShell (Windows), or Putty (Windows).

- Check the IP address for your VM in the 'Course Information' under 'Modules' in Canvas

Type below command in your preferred SSH shell client:

```
ssh your-login@10.92.128.36
```

your-login: your UIS NetID (for example, **slee675** from slee675@uis.edu)
Initial password & IPs: See the Virtual Machine IPs page in the 'Course Information' under 'Modules' in Canvas.

After you logged in to your VM, you will see below prompt. The 'us2004lts' is a name of your VM and stands for 'Ubuntu Sever 20.04 LTS' that we are using as an OS.

```
your-login@us2004lts:~$
```

Example1: using terminal in Mac

```
LeeMBP15:~ sslee777$ ssh sslee777@10.92.128.36
The authenticity of host '10.92.128.36 (10.92.128.36)' can't be established.
ECDSA key fingerprint is SHA256:dXhKfHsYIXe/53hvU+HOK2V6fVrTbz/QxmUhpnPXpzA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.92.128.36' (ECDSA) to the list of known hosts.
sslee777@10.92.128.36's password: <== Use initial password until you change it
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-33-generic x86_64)
...

sslee777@us2004lts:~$
```

3 Deploying Multi-container Apps with Docker Swarm

3.1 Overview

We've learned how to deploy multi-container apps -- popular open source DBMS PostgreSQL, a web-based admin/development tool, pgAdmin, and a Docker management environment, Portainer -- using Docker compose.

Now we want to learn how to deploy multi-container apps on **multi-nodes**. For this, we will build three virtual nodes with Docker machine and VirtualBox, configure a Docker Swarm cluster with the three virtual nodes, and finally build a 3-node Cassandra cluster on top of the Swarm cluster.

You will have a containerized environment hosting a 3-node Cassandra cluster for managing Bigdata, performing Bigdata analytics.

Related Chapters in textbook

- Chapter 10: Docker Swarm
  - Docker Swarm – The Deep Dive
- Chapter 14: Deploying apps with Docker Stacks
  - Deploying the app
- Chapter 11: Docker Networking
  - Multi-host overlay networks
- Chapter 9: Deploying Apps with Docker Compose

3.2 Apache Cassandra cluster (vs. PostgreSQL)

The Apache Cassandra database is the right choice when you need scalability and high availability without compromising performance. Linear scalability and proven fault-tolerance on commodity hardware or cloud infrastructure make it the perfect platform for mission-critical data. Cassandra's support for replicating across multiple datacenters is best-in-class, providing lower latency for your users and the peace of mind of knowing that you can survive regional outages. See, https://cassandra.apache.org/

You may take the CSC561 NoSQL course, taught by me (Sunshin Lee), if you are interested in NoSQL and Cassandra. Some slides about Cassandra are shown below.

Apache Cassandra

- Apache Cassandra
  - a free, open source, distributed data storage system that differs sharply from relational database management systems (RDBMSs)
- Originally developed at Facebook
  - Ex-employees from Amazon and Microsoft
  - Written in Java
- Open-sourced and exists within the Apache family

Source: Jeff Carpenter, Eben Hewitt, Cassandra: the definitive guide 2nd Edition, O'reilly

Source: NoSQL, Perry Hoekstra, Technical Consultant, Perficient, Inc

UNIVERSITY OF ILLINOIS SPRINGFIELD

Apache Cassandra

- Dynamo-style replication model
  - With no single point of failure
  - Uses the Dynamo's eventual consistency model
- Column Family data model
  - Follows Google's BigTable data model
- API
  - Apache Thrift → Cassandra QL (CQL)

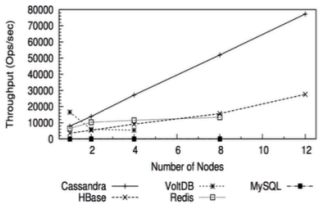
Source: Jeff Carpenter, Eben Hewitt, Cassandra: the definitive guide 2nd Edition, O'reilly

Source: NoSQL, Perry Hoekstra, Technical Consultant, Perficient, Inc

UNIVERSITY OF ILLINOIS SPRINGFIELD

Motivations

- High Scalability
  - Read/write throughput increases linearly when number of nodes increases
- High Availability
  - Cassandra treats failures as normal
  - Decentralized architecture
- High write throughput
  - By efficient disk access policy and flexible consistency level



T. Rabl, S. Gómez-Villamor, M. Sadoghi, V. Muntés-Mulero, H.-A. Jacobsen, and S. Mankovskii, "Solving Big Data Challenges for Enterprise Application Performance Management," Proc. VLDB Endow., vol. 5, no. 12, pp. 1724–1735, Aug. 2012.

UNIVERSITY OF ILLINOIS SPRINGFIELD

DB-Engines Ranking (https://db-engines.com/en/ranking) shows Cassandra is #10 out of 365 DBs, however it's #3 among NoSQL DBs and #1 among column-family DBs. See a picture below. Note: PostgreSQL is #4, but it's #1 non-commercial RDBMS.

356 systems in ranking, June 2020

Rank			DBMS	Database Model	Score		
Jun 2020	May 2020	Jun 2019			Jun 2020	May 2020	Jun 2019
1.	1.	1.	Oracle	Relational, Multi-model	1343.59	-1.85	+44.37
2.	2.	2.	MySQL	Relational, Multi-model	1277.89	-4.75	+54.26
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1067.31	-10.99	-20.45
4.	4.	4.	PostgreSQL	Relational, Multi-model	522.99	+8.19	+46.36
5.	5.	5.	MongoDB	Document, Multi-model	437.08	-1.92	+33.17
6.	6.	6.	IBM Db2	Relational, Multi-model	161.81	-0.83	-10.39
7.	7.	7.	Elasticsearch	Search engine, Multi-model	149.69	+0.56	+0.86
8.	8.	8.	Redis	Key-value, Multi-model	145.64	+2.17	-0.48
9.	9.	↑ 11.	SQLite	Relational	124.82	+1.78	-0.07
10.	↑ 11.	10.	Cassandra	Wide column	119.01	-0.15	-6.17

Recommend you to watch a YouTube video: Docker Tutorial - Getting Started with Cassandra on Docker in less than 10 mins, https://www.youtube.com/watch?v=7gSEXJI8Krg

3.3 Preparing multiple virtual nodes using Docker Machine

We want to prepare multiple nodes for a Swarm cluster, so we will create multiple virtual machines using Docker machine. Docker Machine is a tool that lets you install Docker Engine on virtual hosts and manage the hosts with docker-machine commands. You can use Docker machine to create Docker hosts on your local Mac or Windows box, on your company network, in your data center, or on cloud providers like Azure, AWS, or DigitalOcean. See Docker Machine Overview: https://docs.docker.com/machine/overview/ and Get started with Docker Machine and a local VM: https://docs.docker.com/machine/get-started/

3.3.1 Installing docker machine

To install Docker Machine binaries, following the instructions https://docs.docker.com/machine/install-machine/

Download the Docker Machine binary and extract it to your PATH.

```
sslee777@us2004lts:~$ base=https://github.com/docker/machine/releases/download/v0.16.
0

sslee777@us2004lts:~$ curl -L $base/docker-machine-$(uname -s)-$(uname -m)
>/tmp/docker-machine
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   638  100   638    0     0   2680      0 --:--:-- --:--:-- --:--:--  2680
100 26.8M  100 26.8M    0     0  16.7M      0  0:00:01  0:00:01 --:--:-- 27.1M
```

```
sslee777@us2004lts:~$ ls /tmp/.
docker-machine
snap.lxd
systemd-private-f011dcbfbad541d8a80e99c0694011b0-systemd-logind.service-C7uTdi
systemd-private-f011dcbfbad541d8a80e99c0694011b0-systemd-resolved.service-IV5uvg
systemd-private-f011dcbfbad541d8a80e99c0694011b0-systemd-timesyncd.service-lhfuui
vmware-root_833-3979642945

sslee777@us2004lts:~$ sudo mv /tmp/docker-machine /usr/local/bin/docker-machine
[sudo] password for sslee777:

sslee777@us2004lts:~$ chmod +x /usr/local/bin/docker-machine

sslee777@us2004lts:~$ ls -alF /usr/local/bin/
total 27516
drwxr-xr-x  2 root     root         4096 Jun 28 19:58 ./
drwxr-xr-x 10 root     root         4096 Apr 23 07:32 ../
-rwxrwxr-x  1 sslee777 sslee777 28164576 Jun 28 19:58 docker-machine*
sslee777@us2004lts:~$
```

Let's check versions of Docker, Docker compose, and Docker machine.

```
sslee777@us2004lts:~$ docker version
Client:
 Version:           19.03.8
 API version:       1.40
 Go version:        go1.13.8
 Git commit:        afacb8b7f0
 Built:             Wed Mar 11 23:42:35 2020
 OS/Arch:           linux/amd64
 Experimental:      false

Server:
 Engine:
  Version:          19.03.8
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.13.8
  Git commit:       afacb8b7f0
  Built:            Wed Mar 11 22:48:33 2020
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.3.3-0ubuntu2
  GitCommit:
 runc:
  Version:          spec: 1.0.1-dev
  GitCommit:
 docker-init:
  Version:          0.18.0
  GitCommit:
sslee777@us2004lts:~$
```

```
sslee777@us2004lts:~$ docker-compose -version
docker-compose version 1.25.0, build unknown

sslee777@us2004lts:~$ docker-machine -version
docker-machine version 0.16.0, build 702c267f

sslee777@us2004lts:~$
```

3.3.2 Install VirtualBox

To build virtual nodes using Docker Machine, we need a hypervisor, VirtualBox. We will install a VirtualBox in our VM. https://phoenixnap.com/kb/install-virtualbox-on-ubuntu

Before you install a virtualbox, it's recommended to update the Ubuntu package lists. 'apt-get updates' updates the list of available packages and their versions, but it does not install or upgrade any packages.
Note: Ignore the PostgreSQL GPG error. All are updated except PostgreSQL lists.

```
sslee777@us2004lts:~$ sudo apt-get update
Hit:1 http://us.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease [107 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease [98.3 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu focal-security InRelease [107 kB]
Get:5 http://apt.postgresql.org/pub/repos/apt focal-pgdg InRelease [66.8 kB]
Err:5 http://apt.postgresql.org/pub/repos/apt focal-pgdg InRelease
  The following signatures couldn't be verified because the public key is not
available: NO_PUBKEY 7FCC7D46ACCC4CF8
Reading package lists... Done
W: GPG error: http://apt.postgresql.org/pub/repos/apt focal-pgdg InRelease: The
following signatures couldn't be verified because the public key is not available:
NO_PUBKEY 7FCC7D46ACCC4CF8
E: The repository 'http://apt.postgresql.org/pub/repos/apt focal-pgdg InRelease' is
not signed.
N: Updating from such a repository can't be done securely, and is therefore disabled
by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
sslee777@us2004lts:~$
```

Let's install VirtualBox.

```
sslee777@us2004lts:~$ sudo apt-get install virtualbox
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
...
Suggested packages:
...
The following NEW packages will be installed:
...
0 upgraded, 182 newly installed, 0 to remove and 24 not upgraded.
Need to get 140 MB of archives.
After this operation, 837 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu focal/main ...
...
Get:182 http://us.archive.ubuntu.com/ubuntu focal/multiverse amd64 virtualbox-qt
amd64 6.1.6-dfsg-1 [21.7 MB]
Fetched 140 MB in 3s (41.3 MB/s)
Extracting templates from packages: 100%
Selecting previously unselected package gcc-9-base:amd64.
...
Setting up libgraphite2-3:amd64 (1.3.13-11build1) ...
...
Loading new virtualbox-6.1.6 DKMS files...
Building for 5.4.0-33-generic 5.4.0-39-generic
Building initial module for 5.4.0-33-generic
Done.
...
DKMS: install completed.
Building initial module for 5.4.0-39-generic
Done.
...
Setting up virtualbox (6.1.6-dfsg-1) ...
...
Processing triggers for libgdk-pixbuf2.0-0:amd64 (2.40.0+dfsg-3) ...
sslee777@us2004lts:~$
```

3.3.3 Create virtual nodes with docker machine

Let's start with docker-machine basic command, ls, which lists virtual machines/nodes. We have no VMs yet.

```
sslee777@us2004lts:~$ docker-machine ls
NAME   ACTIVE   DRIVER   STATE   URL   SWARM   DOCKER   ERRORS
sslee777@us2004lts:~$
```

Creating virtual nodes is very straightforward! Just execute following command:
We will use virtual box as a hypervisor (-d virtualbox) and name the node as vm01.

```
sslee777@us2004lts:~$ docker-machine create -d virtualbox vm01
Running pre-create checks...
(vm01) Image cache directory does not exist, creating it at
/home/sslee777/.docker/machine/cache...
(vm01) No default Boot2Docker ISO found locally, downloading the latest
release...
(vm01) Latest release for github.com/boot2docker/boot2docker is v19.03.5
(vm01) Downloading /home/sslee777/.docker/machine/cache/boot2docker.iso from
https://github.com/boot2docker/boot2docker/releases/download/v19.03.5/boot2docker
.iso...
(vm01) 0%....10%....20%....30%....40%....50%....60%....70%....80%....90%....100%
Creating machine...
(vm01) Copying /home/sslee777/.docker/machine/cache/boot2docker.iso to
/home/sslee777/.docker/machine/machines/vm01/boot2docker.iso...
(vm01) Creating VirtualBox VM...
(vm01) Creating SSH key...
(vm01) Starting the VM...
(vm01) Check network to re-create if needed...
(vm01) 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: docker-machine env vm01
sslee777@us2004lts:~$
```

Docker is downloading Boot2Docker ISO image from GitHub. Boot2Docker is a lightweight Linux distribution made specifically to run Docker containers. It runs completely from RAM, is a ~45MB download and boots quickly, See https://github.com/boot2docker/boot2docker

Note: You may need to wait about 1-2 minute when you see "(vm01) Waiting for an IP..."

Docker is up and running inside a virtual node on VirtualBox.

Let's verfiy your new virtual node with command 'docker-machine ls'. You may have a different IP address, e.g. 192.168.99.100

```
sslee777@us2004lts:~$ docker-machine ls
NAME   ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER
ERRORS
vm01   -        virtualbox   Running   tcp://192.168.99.102:2376           v19.03.5
sslee777@us2004lts:~$
```

You still have plenty of memory spaces, more than 13 GiB. You should see a VBoxHeadless process -- VirtualBox headless mode which means no display or screen -- for your virtual node. Note: Type ‘q’ to quit.

```
sslee777@us2004lts:~$ top
```

```
top - 23:28:36 up  2:29,  1 user,  load average: 0.71, 0.39, 0.27
Tasks: 272 total,   1 running, 271 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.8 sy,  0.2 ni, 98.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  16012.8 total,  13934.2 free,   1280.3 used,    798.3 buff/cache
MiB Swap:   4096.0 total,   4096.0 free,      0.0 used.  14434.7 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   8846 sslee777  20   0 2139088  78556  45008 S   8.3   0.5   5:30.18 VBoxHeadless
  10118 sslee777  20   0    8524   4212   3316 R   0.7   0.0   0:00.12 top
```

We still have enough storage, 98 GiB out of 100 GiB

```
sslee777@us2004lts:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            7.8G     0  7.8G   0% /dev
tmpfs           1.6G  1.4M  1.6G   1% /run
/dev/sda2        98G   13G   81G  14% /
tmpfs           7.9G     0  7.9G   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           7.9G     0  7.9G   0% /sys/fs/cgroup
/dev/loop1       72M   72M     0 100% /snap/lxd/15724
/dev/loop2       55M   55M     0 100% /snap/core18/1754
/dev/loop0       55M   55M     0 100% /snap/core18/1705
/dev/loop3       31M   31M     0 100% /snap/snapd/7777
/dev/loop4       30M   30M     0 100% /snap/snapd/8140
/dev/loop5       72M   72M     0 100% /snap/lxd/15753
tmpfs           1.6G     0  1.6G   0% /run/user/1001
sslee777@us2004lts:~$
```

Something went worng? You may need below commands:

```
docker-machine stop vm01
docker machine rm vm01
```

Next create another virtual node using docker-machine, vm02
If you see an error like below, you may need to wait several minutes and/or regenerate TLS machine certs.

```
sslee777@us2004lts:~$ docker-machine create -d virtualbox vm02
Running pre-create checks...
Creating machine...
(vm02) Copying /home/sslee777/.docker/machine/cache/boot2docker.iso to
/home/sslee777/.docker/machine/machines/vm02/boot2docker.iso...
(vm02) Creating VirtualBox VM...
(vm02) Creating SSH key...
(vm02) Starting the VM...
(vm02) Check network to re-create if needed...
(vm02) 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...
Error creating machine: Error running provisioning: Unable to verify the Docker
daemon is listening: Maximum number of retries (10) exceeded
```

You may also see an error like below when you try to see your virtual nodes lists. Also need to regenerate TLC machine certs.

```
sslee777@us2004lts:~$ docker-machine ls
NAME   ACTIVE   DRIVER       STATE     URL                         SWARM
DOCKER     ERRORS
vm01   -        virtualbox   Running   tcp://192.168.99.102:2376
v19.03.5
vm02   -        virtualbox   Running   tcp://192.168.99.103:2376
Unknown    Unable to query docker version: Get
https://192.168.99.103:2376/v1.15/version: x509: certificate signed by
unknown authority
sslee777@us2004lts:~$
```

To regenerate TLS machine certs, use 'docker-machine regenerate-certs' command.
You need to specify virtual node's name, vm02.

```
sslee777@us2004lts:~$ docker-machine regenerate-certs vm02
Regenerate TLS machine certs?  Warning: this is irreversible. (y/n): y
Regenerating TLS certificates
Waiting for SSH to be available...
Detecting the provisioner...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
```

Now the problem has been resolved. You have two virtual nodes, vm01 and vm02.

```
sslee777@us2004lts:~$ docker-machine ls
NAME   ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER
ERRORS
vm01   -        virtualbox   Running   tcp://192.168.99.102:2376           v19.03.5
vm02   -        virtualbox   Running   tcp://192.168.99.103:2376           v19.03.5
sslee777@us2004lts:~$
```

You may have an error regarding VT-x/AMD-v. To use VirtualBox, VT-X/AMD-v option of CPU should be enabled in your Host VM (instructor has already enabled it)

```
sslee777@us2004lts:~$ docker-machine create --driver virtualbox docker-
sandbox
Creating CA: /home/sslee777/.docker/machine/certs/ca.pem
Creating client certificate: /home/sslee777/.docker/machine/certs/cert.pem
Running pre-create checks...
Error with pre-create check: "This computer doesn't have VT-X/AMD-v enabled.
Enabling it in the BIOS is mandatory"
sslee777@us2004lts:~$
```

Continue to create 3rd node, vm03.

3.4 Setting up Swarm cluster with Docker Swarm

Now you have three virtual nodes for a Swarm cluster. Check the names and IP addresses.

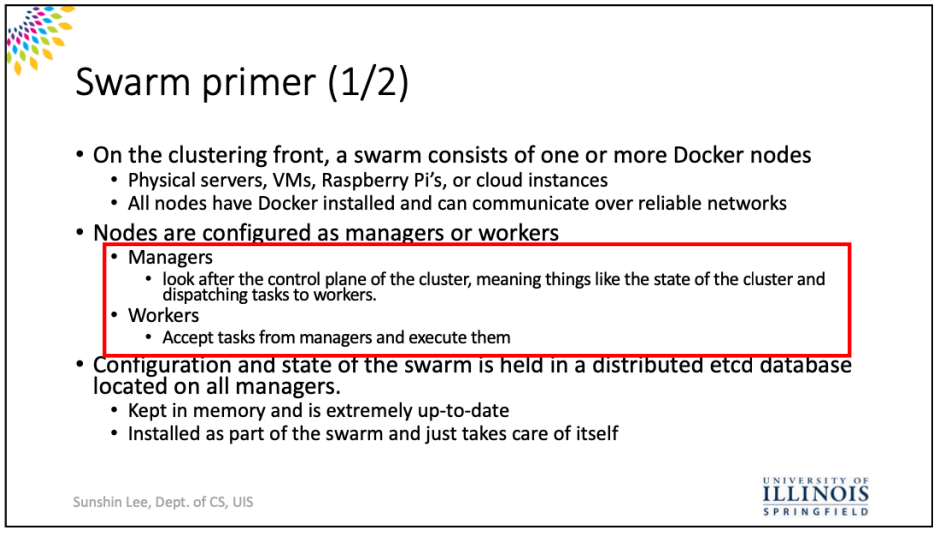
```
sslee777@us2004lts:~$ docker-machine ls
NAME   ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER     ERRORS
vm01   -        virtualbox   Running   tcp://192.168.99.102:2376           v19.03.5
vm02   -        virtualbox   Running   tcp://192.168.99.103:2376           v19.03.5
vm03   -        virtualbox   Running   tcp://192.168.99.104:2376           v19.03.5
sslee777@us2004lts:~$
```

To configure your Docker Swarm cluster, you will ssh to vm01.
Note: Always check your prompt. You need to know whether you are in HostVM, a container, or a virtual node.

```
sslee777@us2004lts:~$ docker-machine ssh vm01
   ( '>')
  /) TC (\   Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\)           www.tinycorelinux.net

docker@vm01:~$
```

Once you're inside the node, create a cluster manager in vm01.



Execute 'docker swarm init' command. Pass the --advertise-addr argument with the IP address of the vm01

```
docker@vm01:~$ docker swarm init --advertise-addr 192.168.99.102
Swarm initialized: current node (5elb7i9ul5w7l20i9t26i6glm) is now a manager.

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

    docker swarm join --token SWMTKN-1-
56frl8rlop5xysxf85qs5jqhgkzffb1sh8pktzvhp8u5gghevp-48scxieq75hyp3m7pgl4cxirs
192.168.99.102:2377

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

docker@vm01:~$ exit
```

Now, we have one node in our cluster and it is a manager. Meaning, we can schedule tasks from this node. Copy the docker swarm join command since we'll use it later.

Before we continue, we need to add vm02 and vm03 as workers to the cluster.

Note: Don't forget to exit from vm01

SSH to these nodes and execute. Token and IP address is for the manager node (vm01)

```
sslee777@us2004lts:~$ docker-machine ssh vm02
   ( '>')
  /) TC (\   Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\)           www.tinycorelinux.net

docker@vm02:~$ docker swarm join --token SWMTKN-1-
56frl8rlop5xysxf85qs5jqhgkzffb1sh8pktzvhp8u5gghevp-48scxieq75hyp3m7pgl4cxirs
192.168.99.102:2377

This node joined a swarm as a worker.
docker@vm02:~$
```

To make vm3 as a worker node, you SSH to the vm3. To simplify the process, you may pass the command to ssh command. It's much simpler and faster. Note: Don't forget to use double quote "..."

```
sslee777@us2004lts:~$ docker-machine ssh vm03 "docker swarm join --token SWMTKN-1-
56frl8rlop5xysxf85qs5jqhgkzffb1sh8pktzvhp8u5gghevp-48scxieq75hyp3m7pgl4cxirs
192.168.99.102:2377"
This node joined a swarm as a worker.
sslee777@us2004lts:~$
```

After joining vm02 and vm03 as workers, you can check from vm01 if those nodes are available from the manager. Again, you may pass the command to ssh. You have three nodes and vm01 is a manager or leader.

```
sslee777@us2004lts:~$ docker-machine ssh vm01 "docker node ls"
ID                            HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION
5elb7i9ul5w7l20i9t26i6glm *   vm01     Ready  Active       Leader         19.03.5
wsqs89geamfhh33d1fzt76s18     vm02     Ready  Active                      19.03.5
rlmbmuh3oo1grhjn6rry05rol     vm03     Ready  Active                      19.03.5
sslee777@us2004lts:~$
```

Let's look at some information about the node inside the vm.

```
sslee777@us2004lts:~$ docker-machine ssh vm01 "docker info"
Client:
 Debug Mode: false

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 19.03.5
 Storage Driver: overlay2
  Backing Filesystem: extfs
  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: 5elb7i9ul5w7l20i9t26i6glm
  Is Manager: true
  ClusterID: ygyskqkt2suwr7wzn380a8bh7
  Managers: 1
  Nodes: 3
  Default Address Pool: 10.0.0.0/8
  SubnetSize: 24
  Data Path Port: 4789
  Orchestration:
   Task History Retention Limit: 5
  Raft:
   Snapshot Interval: 10000
   Number of Old Snapshots to Retain: 0
   Heartbeat Tick: 1
   Election Tick: 10
  ...
  Node Address: 192.168.99.102
  Manager Addresses:
   192.168.99.102:2377
 ...
 Operating System: Boot2Docker 19.03.5 (TCL 10.1)
 OSType: linux
 Architecture: x86_64
 CPUs: 1
 Total Memory: 989.5MiB
 Name: vm01
 ...
sslee777@us2004lts:~$
```

3.5 Building a Cassandra cluster on a Swarm cluster

Official Cassandra Docker image can be found here:
https://hub.docker.com/_/cassandra
https://github.com/docker-library/cassandra

3.5.1 Creating a docker-compose file for deploying a Cassandra cluster

Login to vm01 and create a directory, casscluster.

```
sslee777@us2004lts:~$ docker-machine ssh vm01
   ( '>')
  /) TC (\   Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\)           www.tinycorelinux.net

docker@vm01:~$ mkdir casscluster
docker@vm01:~$ cd casscluster/
docker@vm01:~/casscluster$
```

Inside the node, vm01, create a docker compose file. We will have three services: cassandra01, cassandra02, and cassandra03. We want deploy multi-container apps on multi-node with Swarm , so we will put cassandra01, 02, and 03 on vm01, vm02, and vm03 respectively, see placement: constraints:-node.hostname in the compose file.

```
docker@vm01:~/casscluster$ vi docker-compose.yml
version: "3.7"
services:
  # Node01
  cassandra01:
    image: cassandra:3.11
    environment:
      CASSANDRA_BROADCAST_ADDRESS: "cassandra01"
    deploy:
      restart_policy:
        condition: on-failure
        max_attempts: 3
        window: 120s
      placement:
        constraints:
          - node.hostname == vm01
    ports:
      - 9042
    volumes:
        - cass-data:/var/lib/cassandra
    networks:
      - cass-net
```

```
  # Node02
  cassandra02:
    image: cassandra:3.11
    environment:
      CASSANDRA_BROADCAST_ADDRESS: "cassandra02"
      CASSANDRA_SEEDS: "cassandra01"
    depends_on:
      - cassandra-1
    deploy:
      restart_policy:
        condition: on-failure
        max_attempts: 3
        window: 120s
      placement:
        constraints:
          - node.hostname == vm02
    volumes:
        - cass-data:/var/lib/cassandra
    networks:
      - cass-net

  # Node03
  cassandra03:
    image: cassandra:3.11
    environment:
      CASSANDRA_BROADCAST_ADDRESS: "cassandra03"
      CASSANDRA_SEEDS: "cassandra01"
    depends_on:
      - cassandra-1
    deploy:
      restart_policy:
        condition: on-failure
        max_attempts: 3
        window: 120s
      placement:
        constraints:
          - node.hostname == vm03
    volumes:
        - cass-data:/var/lib/cassandra
    networks:
      - cass-net

networks:
  cass-net:

volumes:
  cass-data:
```

3.5.2 Deploy a stack to a swarm

We will deploy the compose file to the Swarm cluster using Docker stack. https://docs.docker.com/engine/swarm/stack-deploy/. You may also see Chapter 14. Deploying apps with Docker Stacks in the textbook.

```
docker@vm01:~/casscluster$ docker stack deploy --compose-file docker-compose.yml
casscluster
Creating network casscluster_cass-net
Creating service casscluster_cassandra01
Creating service casscluster_cassandra02
Creating service casscluster_cassandra03
docker@vm01:~/casscluster$
```

Verify with 'docker stack ls' command. We have the casscluster with 3 services orchestrated by Swarm.

```
docker@vm01:~/casscluster$ docker stack ls
NAME                SERVICES            ORCHESTRATOR
casscluster         3                   Swarm
docker@vm01:~/casscluster$
```

You may also look at each process. You see cassandra01, 02, and 03 are running on vm01, 02, and 03 respectively

```
docker@vm01:~/casscluster$ docker stack ps casscluster
ID            NAME                     IMAGE          NODE DESIRED STATE CURRENT STATE     ERROR  PORTS
i7ixv1ux4x31  casscluster_cassandra01.1  cassandra:3.11  vm01 Running       Preparing 23 seconds ago
papxffskhhnz  casscluster_cassandra03.1  cassandra:3.11  vm03 Running       Preparing 5 minutes ago
kbim33smg3s6  casscluster_cassandra02.1  cassandra:3.11  vm02 Running       Preparing 5 minutes ago
docker@vm01:~/casscluster$
```

You may list the services in the stack

```
docker@vm01:~/casscluster$ docker stack services casscluster
ID                  NAME                      MODE                REPLICAS            IMAGE
PORTS
5eq2riat0m7g        casscluster_cassandra02   replicated          1/1                 cassandra:3.11
9545uzwh7xtm        casscluster_cassandra01   replicated          1/1                 cassandra:3.11
x9phgcbd5qp6        casscluster_cassandra03   replicated          1/1                 cassandra:3.11
docker@vm01:~/casscluster$
```

Further information regarding manage nodes in a Swarm, see https://docs.docker.com/engine/swarm/manage-nodes/

There is another similar command, listing nodes in the swarm

```
docker@vm01:~$ docker node ls
ID                            HOSTNAME    STATUS      AVAILABILITY     MANAGER STATUS      ENGINE VERSION
5elb7i9ul5w7l20i9t26i6glm *   vm01        Ready       Active           Leader              19.03.5
wsqs89geamfhh33d1fzt76s18     vm02        Ready       Active                               19.03.5
rlmbmuh3oo1grhjn6rry05rol     vm03        Ready       Active                               19.03.5
docker@vm01:~$
```

You can run docker node inspect <NODE-ID> on a manager node to view the details for an individual node. The output defaults to JSON format, but you can pass the --pretty flag to print the results in human-readable format. For example:

```
docker@vm01:~$ docker node inspect self --pretty
ID:                     5elb7i9ul5w7l20i9t26i6glm
Hostname:               vm01
Joined at:              2020-06-29 00:04:34.540997757 +0000 utc
Status:
 State:                 Ready
 Availability:          Active
 Address:               192.168.99.102
Manager Status:
 Address:               192.168.99.102:2377
 Raft Status:           Reachable
 Leader:                Yes
Platform:
 Operating System:      linux
 Architecture:          x86_64
Resources:
 CPUs:                  1
 Memory:                989.5MiB
Plugins:
 Log:         awslogs, fluentd, gcplogs, gelf, journald, json-file, local, logentries,
splunk, syslog
 Network:               bridge, host, ipvlan, macvlan, null, overlay
 Volume:                local
Engine Version:         19.03.5
Engine Labels:
 - provider=virtualbox
TLS Info:
...
docker@vm01:~$
```

If something went wrong, you may want to delete the casscluster by

```
docker stack rm casscluster
```

Let's check if Cassandra cluster is working fine.
Note: It may take a couple of minutes, because Cassandra nodes need to be initialized and communicate each other.

First we will get the name of the cluster.

```
docker@vm01:~/casscluster$ docker ps
CONTAINER ID...PORTS                                        NAMES
774ec02dd5e6...7000-7001/tcp,7199/tcp, 042/tcp,9160/tcp casscluster_cassandra01.1.ip4vxazwk08p54a0ke0206w46
```

And then, run docker exec command to run 'nodetool status' command. It shows cluster information (state, load, IDs, ...)
https://cassandra.apache.org/doc/latest/tools/nodetool/status.html

We see three nodes are working and their status are UN which means Up and Normal. Cassnadra NoSQL database cluster is working fine.

```
docker@vm01:~/casscluster$ docker exec
casscluster_cassandra01.1.ip4vxazwk08p54a0ke0206w46 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load       Tokens  Owns (effective)  Host ID                               Rack
UN  10.0.5.2   281.49 KiB 256     68.7%             9e8c1ffa-d24d-476c-8600-9e76ed3b9b8a  rack1
UN  10.0.5.5   92.61 KiB  256     68.0%             f8b442d6-c83d-4aae-88e1-c2eb1c70f66c  rack1
UN  10.0.5.8   92.92 KiB  256     63.3%             eddd931d-54c8-44da-b202-950ad2d05fc6  rack1
```

We also check the cluster by running cqlsh. cqlsh is a command line shell for interacting with Cassandra through CQL (the Cassandra Query Language), see
https://cassandra.apache.org/doc/latest/tools/cqlsh.html

```
docker@vm01:~/casscluster$ docker exec -it
casscluster_cassandra01.1.ip4vxazwk08p54a0ke0206w46 cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.6 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> quit
docker@vm01:~/casscluster$
```

If you can see cqlsh prompt, you are connected to the Cassandra cluster using cqlsh and ready to run some queries to store or analyze Bigdata.

4 Submit

Submit a word document (docx, doc, or PDF) to Canvas.
Assignments

- Run your code/scripts, take screenshots, and explain about it.
    a. Section 3.3 Preparing multiple virtual nodes using Docker Machine
    b. Section 3.4 Setting up Swarm cluster with Docker Swarm
    c. Section 3.5 Building a Cassandra cluster

If you have any problems or questions regarding this exercise, post messages in the 'Discussions' in Canvas. Posting exact codes or links to an article that have exact codes are not allowed.

5 Reference (Note: Links may not work in PDF, try to copy&paste it)

1. DB-Engines Ranking, https://db-engines.com/en/ranking
2. How to Use Docker Machine to Create a Swarm Cluster, https://www.linux.com/topic/cloud/how-use-docker-machine-create-swarm-cluster/
3. How To Provision and Manage Remote Docker Hosts with Docker Machine on Ubuntu 16.04, https://www.digitalocean.com/community/tutorials/how-to-provision-and-manage-remote-docker-hosts-with-docker-machine-on-ubuntu-16-04
4. How To Install VirtualBox On Ubuntu, https://phoenixnap.com/kb/install-virtualbox-on-ubuntu
5. boot2docker, https://github.com/boot2docker/boot2docker
6. Docker: Orchestration of multi-container apps with Swarm and Compose, https://www.ionos.com/digitalguide/server/know-how/docker-orchestration-with-swarm-and-compose/
7. How to Create a Cluster of Docker Containers with Docker Swarm and DigitalOcean on Ubuntu 16.04, https://www.digitalocean.com/community/tutorials/how-to-create-a-cluster-of-docker-containers-with-docker-swarm-and-digitalocean-on-ubuntu-16-04
8. Docker Swarm - How to create a minimal cluster running a service using VirtualBox, https://www.infralovers.com/en/articles/2018/09/11/docker-swarm-how-to-create-a-minimal-cluster-running-a-service-using-virtualbox/
9. Docker Swarm Tutorial, https://rominirani.com/docker-swarm-tutorial-b67470cf8872
10. Create Cluster using docker swarm, https://medium.com/tech-tajawal/create-cluster-using-docker-swarm-94d7b2a10c43
11. Create a swarm cluster with Docker 1.12 swarm mode, https://medium.com/lucjuggery/create-a-swarm-cluster-with-docker-1-12-swarm-mode-541449114c27
12. How To Setup a Docker Swarm Cluster on Ubuntu 16.04 VPS or Dedicated Server, https://hostadvice.com/how-to/how-to-setup-a-docker-swarm-cluster-on-ubuntu-16-04/
13. Install and configure Docker swarm, https://itnext.io/install-and-monitor-docker-swarm-e9db8a8fbbc4
14. Official Cassandra Docker: https://hub.docker.com/_/cassandra
15. Docker Tutorial - Getting Started with Cassandra on Docker in less than 10 mins, https://www.youtube.com/watch?v=7gSEXJI8Krg
16. Docker Meet Cassandra. Cassandra Meet Docker, https://thelastpickle.com/blog/2018/01/23/docker-meet-cassandra.html

17. Apache Cassandra: Begins with Docker, https://albertusk95.github.io/posts/2019/08/install-cassandra-docker/
18. Cassandra and Docker-Swarm, https://ralph.blog.imixs.com/2019/06/24/cassandra-and-docker-swarm/
19. Cassandra cluster on Docker Swarm and Overlay Networking using Docker Experimental 1.9, http://sirile.github.io/2015/09/30/cassandra-cluster-on-docker-swarm-and-overlay-networking-using-docker-experimental-1.9.html
20. Creating a multiple-node Swarm cluster using Docker Machine, https://www.melvinvivas.com/create-a-swarm-cluster-using-docker-machine/
21. Docker Swarm Part I: Multi-Host Cassandra Cluster, https://dzone.com/articles/swarmweek-part-1-multi-host-cassandra-cluster-with
22. Five Minute Guide: Getting Started with Cassandra on Docker, https://medium.com/@michaeljpr/five-minute-guide-getting-started-with-cassandra-on-docker-4ef69c710d84
23. Apache Cassandra: Begins with Docker, https://albertusk95.github.io/posts/2019/08/install-cassandra-docker/