DOCKER SWARM

PRESENTATION BY NGUYEN NHU THUONG

NỘI DUNG CHÍNH

SWARM CLUSTER

Tìm hiểu về khái niệm cluster là gì? Tại sao lại phải cần triển khai cluster và ứng dụng nó như thế nào?

SERVICE & TASK

Tìm hiểu về các đơn vị triển khai trong một cụm server, các thành phần này khác gì so với Docker thông thường?

INGRESS NETWORK

Tìm hiểu về network, load balancer trong Docker Swarm. Các triển khai dịch vụ sử dung ingress network.

INTERNAL OVERLAY NETWORK

Tìm hiểu cách giao tiếp giữa 2 service được triển khai trên nhiều server khác nhau mà vẫn đảm bảo bảo mật.

SWARM CLUSTER

Standalone Docker Host

- Nhiều container chạy trên cùng một docker host.
- Bị giới hạn computing resources, phần cứng (CPU, RAM) chỉ mở rộng tới một điểm nào đó nhất định.
- Khó khăn mở rộng hệ thống, rủi ro chết hệ thống (khả dụng thấp).

Multiple Docker hosts

- Các container được triển khai trên nhiều docker host phù hợp.
- Các docker host được gom cụm (clustering) để trở thành 1 server khổng lồ.
- Tận dụng computing resources của nhiều server (mở rộng theo chiều ngang).
- Phù hợp cho hệ thống sử dụng kiến trúc microservices.

Container Orchestration

- Container orchestration là các công cụ, dịch vụ dùng để điều phối và quản lý nhiều container sao cho chúng làm việc một cách hiệu quả nhất.
- Tự động triển khai, mở rộng, quản lý các containers.

TASK #1

TẠO SWARM CLUSTER

mô hình cluster

các bước tiến hành

- Chuẩn bị 03 máy ảo đã cài sẵn Docker
- SSH vào máy 01, khởi tạo Swarm Cluster bằng lệnh:

```
$ docker swarm init --advertise-addr IP-của-máy-ảo
```

- Hệ thống tạo ra 1 đoạn code để thêm các worker node vào Swarm cluster. Copy paste đoạn code đó và chạy trên 2 node còn lại.
- Trên cả 03 máy, chạy lệnh dưới đây để kiểm tra cluster:

```
$ docker node ls
```

Swarm Nodes

- Mỗi node trong Swarm Cluster ~ 1 docker host.
- Có 2 loại node: manager và worker.
- Manager:
  - Quản lý trạng thái của cluster
  - Điều phối các container
  - Chạy các container
- Worker:
  - Chạy các container do manager chỉ định

SERVICE & TASK

Stand-alone container vs Swarm service

Stand-alone container	Swarm service
App được triển khai theo từng container	App được triển khai dưới dạng service
Mỗi App là một container chạy trên một node	Mỗi service gồm một hoặc nhiều task chạy trên cluster
	Mỗi task khởi tạo một container
Docker run … image	Docker service create … image

tand-alone container vs Swarm service

Stand-alone container

Nginx Container

Swarn service

TASK #2

TRIỂN KHAI 1 SERVICE CHẠY IMAGE NGINX TRÊN SWARM CLUSTER

các bước tiến hành

Trên manager node:

- Sử dụng **docker service create** với image nginx:alpine, expose ra cổng 8080, đặt tên service là nginx-app
- Liệt kê danh sách service: **docker service** ls
- Liệt kê các task có trong service nginx-app: **docker service** ps nginx-app

Trên tất cả các node:

- Kiểm tra port 8080 có được mở hay chưa:

sudo netstat –tulpn | grep LISTEN | grep 8080

Service mô tả desired state của app

Desired state của service nginx-app:

- Sử dụng image **nginx:latest**
- Expose ra cổng **8080** của cluster
- Số lượng task: 1
- Chạy trên 1 node bất kỳ

Service mô tả trạng thái chúng ta mong muốn (**What**), còn thế nào (**How**) để duy trì trạng thái đó là nhiện vụ của Swarm Managers

TASK #3

CẬP NHẬT DESIRED STATE CHO APP NGINX

yêu cầu

Cập nhật desired state mới cho service nginx-app:

- Số lượng task: 6 (**--replicas**)
- Không publish ra cổng 8080 của cluster nữa (**--publish-rm 8080:80**)
- Thay vào đó publish ra cổng 8081 (**--publish-add 8081:80**)
- Kiểm tra trạng thái service nginx-app

Sử dụng lệnh: **docker service update [OPTIONS]** tên-service

Tham khảo các option:

https://docs.docker.com/engine/reference/commandline/service_update

TASK #4

GỠ MỘT NODE RA KHỎI CLUSTER

yêu cầu

- Trên 1 worker node có các task đang chạy, dung lệnh sau để gỡ node ra khỏi cluster:

docker swarm leave

- Kiểm tra trạng thái của service nginx-app

TASK #5

THÊM NODE VÀO CLUSTER

yêu cầu

- Tạo them Docker host instance
- Trên node manager, chạy lệnh sau để lấy join token vào Swarm:

docker swarm join-token worker

- Copy lệnh mà node manager tạo ra để chạy trên instance mới
- Kiểm tra trạng thái service nginx-app

Golbal vs replicated service

TASK #6

TẠO GLOBAL SERVICE MONITORING CÁC NODE

yêu cầu

- Tạo global service cho image google/cadvisor:latest
- Kiểm tra trên các node
- Sau khi tạo global service, add them 1 worker node vào cluster

Tạo service có tên monitor, mode global, expose ra cổng 9005

```
docker service create \
--mount type=bind,source=/,destination=/rootfs:ro \
--mount type=bind,source=/var/run,destination=/var/run:rw \
--mount type=bind,source=/var/run/docker.sock,destination=/var/run/docker.sock \ --mount
type=bind,source=/sys,destination=/sys:ro \
--mount type=bind,source=/var/lib/docker/,destination=/var/lib/docker:ro \
--mode global \
--name monitor \
-p 9005:8080 \
google/cadvisor:latest
```

INGRESS NETWORK

Ingress Network

TASK #7

TRIỂN KHAI SERVICE SỬ DỤNG INGERSS NETWORK

các bước tiến hành

Trên node manager:

- Tạo service từ image thuongnn1997/nginx:debug gồm 2 tasks
- Expose cổng 80 của image ra cổng 8085 của cluster
- Đặt tên service là my-nginx
- Chỉ chạy các tasks trên các node worker

Gọi vào cổng 8085 từ một node bất kỳ trong Swarm

Tham khảo cách đặt constraint cho service:

https://docs.docker.com/engine/reference/commandline/service_create/#specify-service-constraints---constraint

Host mode published port

TASK #8

HOST-MODE PUBLISHED PORT CHO GLOBAL SERVICE

các bước tiến hành

- Remove published port:

  docker service update --publish-rm 9005:8080 monitor

- Kiểm tra trạng thái service: docker service ps monitor
- Add host-mode published port:

  docker service update --publish-add mode=host,published=9006,target=8080 monitor

- Kiểm tra trạng thái service monitor
- Truy cập cổng 9006 từ các node trong cluster

INTERNAL OVERLAY NETWORK

Ingress Network

TASK #9

TRIỂN KHAI ỨNG DỤNG.NODEJS VỚI MONGODB

các bước tiến hành

- Viết Dockerfile cho source code NodeJS:

  https://github.com/handuy/nodejs-mongodb
- Build thành docker image có tên <docker-hub-repo>/demo-service
- Push image lên Docker Hub

triển khai MongoDB

Tạo 1 internal overlay network: **docker network create -d overlay myapp**

Triển khai service MongoDB:

- Sử dụng image **mongo:latest**
- Tên service là **mongodb**
- Mount thư mục **/data/db** ra 1 volume có tên dbdata
- Chạy ở 1 node cố định
- Gắn vào network myapp

triển khai service NodeJS

Với service NodeJS:

- Chạy sau service mongodb và scale thành 3 tasks
- Sử dụng image **demo-service** vừa build ở slide trước
- Expose cổng 3000 của ứng dụng ra cổng 8000 của cluser
- Định nghĩa 2 biến môi trường: **MONGODB_URI**=mongodb://mongodb:27017/demo, **PORT**=3000
- Gắn vào network myapp

Thanks for watching

Email: thuongnn6666@gmail.com