

Docker 建立容器集群

資工二 S1254013 楊敦傑

1. 引言

在現代應用的部署中，容器化技術已成為不可或缺的一部分，特別是當應用需要在多節點的環境中運行時。雖然 Kubernetes 是最常見的容器編排工具之一，但其配置和管理過程相對複雜，許多教學資料建議使用虛擬機技術來建立 Kubernetes 集群(cluster)，然而，這樣的方式對於希望專注於輕量化容器環境的我來說並不理想。本報告基於這樣的挑戰，將嘗試基於 Docker 容器的 Kubeadm 初始化構建 k8s，並說明為何改用 Docker 內建的 Swarm。

2. Kubeadm 集群構建

為了模擬虛擬機環境，我們基於 Docker 定義了多個容器節點，其中包含 Kubernetes 主節點和工作節點，並進行以下操作：

1. 製作鏡像容器，具備 Kubeadm 初始化必備條件。

```
# Use the official Ubuntu base image
FROM ubuntu:latest

# Set environment variable for non-interactive installation
ENV DEBIAN_FRONTEND=noninteractive

# Install systemd and other necessary packages
RUN apt-get update && \
    apt-get install -y systemd systemd-sysv netcat-traditional sudo conntrack ethtool socat net-tools curl iputils-ping nano open \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# Set the default command to run systemd as init
STOPSIGNAL SIGRTMIN+3
CMD ["/lib/systemd/systemd"]
```

2. Kubeadm 初始化:嘗試在容器中進行 Kubeadm 的初始化

- 初始化過程遇到了問題，容器能解析域名，但無法拉取鏡像，奇怪的是 docker 卻能拉取鏡像，因此參考 AI 給出的答案：鏡像解壓過程中出現 operation not supported，可能與底層存儲驅動（如 aufs 或 overlay）不兼容有關。

```
[ERROR ImagePull]: failed to pull image registry.k8s.io/kube-apiserver:v1.31.2: failed to pull image
registry.k8s.io/kube-apiserver:v1.31.2: failed to pull and unpack image "registry.k8s.io/kube-apiserver:v1.
31.2": failed to extract layer sha256:5b1efae7f7c35b14e825385dec30a0aa235a85b9469c7effb720b08c76ea6: fail
ed to convert whiteout file "usr/local/.wh..wh..opq": operation not supported: unknown
[ERROR ImagePull]: failed to pull image registry.k8s.io/kube-controller-manager:v1.31.2: failed to p
ull image registry.k8s.io/kube-controller-manager:v1.31.2: failed to pull and unpack image "registry.k8s.io/
kube-controller-manager:v1.31.2": failed to extract layer sha256:c5c90b3a07ad6879d9511c3d60de1ef1816b4aba5af
b47722d931442200c2e94: failed to convert whiteout file "usr/local/.wh..wh..opq": operation not supported: un
known
[ERROR ImagePull]: failed to pull image registry.k8s.io/kube-scheduler:v1.31.2: failed to pull image
registry.k8s.io/kube-scheduler:v1.31.2: failed to pull and unpack image "registry.k8s.io/kube-scheduler:v1.
31.2": failed to extract layer sha256:c23f607019c9db903dc43508224077f462eb938ecfb303c892fef0f8643bc1ed5: fail
ed to convert whiteout file "usr/local/.wh..wh..opq": operation not supported: unknown
[ERROR ImagePull]: failed to pull image registry.k8s.io/kube-proxy:v1.31.2: failed to pull image reg
istry.k8s.io/kube-proxy:v1.31.2: failed to pull and unpack image "registry.k8s.io/kube-proxy:v1.31.2": faile
d to extract layer sha256:54cddbc5055ab0b98dbd20afc3859c9b0ead4562123811a174205a4fde4b148: failed to conver
t whiteout file "usr/local/.wh..wh..opq": operation not supported: unknown
```

在基於 Docker 容器構建 Kubernetes 集群的過程中，網路問題成為了主要的挑戰之一。特別是在執行 `kubeadm init` 前，Kubernetes 要求其他容器節點能夠正確連接到主節點的指定端口。即使我的容器之間可以相互 ping 通 IP 地址，但連接到指定的端口卻失敗，導致初始化過程無法順利完成。

3. Docker Swarm

礙於以下兩個問題遲遲無法解決

1. 容器網路隔離：默認的 Docker 網路設置可能導致容器之間的端口未正確暴露或隔離。
2. 路由配置不完整：容器之間的路由可能無法完全覆蓋跨容器通信所需的端口範圍。

經過查詢後發現，Docker 自帶了一種輕量化的集群管理工具—Docker Swarm，其內建於 Docker 引擎中，不需要額外安裝，配置更是方便。Docker Swarm 內建了對容器的網路管理，特別是它的 Ingress 網路，這對於解決網路隔離問題至關重要。Swarm 提供的 ingress 網路是一種多主機的虛擬網路，所有部署在 Swarm 上的服務容器都能夠通過這個網路進行通信。當啟動一個服務時，Docker 會自動為這些服務分配網路端口，並將其暴露給其他節點。這不僅解決了容器之間的隔離問題，亦保證了跨主機的容器通信不會出現問題。結果如下圖：

容器執行狀態：

```
root@LAPTOP-30V3PCQM:/home/grason# docker ps
```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS |
|--------------|-------------|-------------------------|-------------|------------|---------------|
| 8df3bb7d9963 | docker:dind | "dockerd-entrypoint..." | 2 hours ago | Up 2 hours | 2375-2376/tcp |
| cp_worker2 | docker:dind | "dockerd-entrypoint..." | 2 hours ago | Up 2 hours | 2375-2376/tcp |
| a511414d6bbd | docker:dind | "dockerd-entrypoint..." | 2 hours ago | Up 2 hours | 2375-2376/tcp |
| cp_worker1 | docker:dind | "dockerd-entrypoint..." | 2 hours ago | Up 2 hours | 2375-2376/tcp |
| bc2ac9ea41f5 | docker:dind | "dockerd-entrypoint..." | 2 hours ago | Up 2 hours | 2375-2376/tcp |
| cp_manager | docker:dind | "dockerd-entrypoint..." | 2 hours ago | Up 2 hours | 2375-2376/tcp |

集群建立:

1. 集群初始化
2. 獲取相關 token
3. 在 node(worker) 中將該節點加入集群
4. 獲取配置服務
5. 查看配置服務之負載均衡結果
6. 確認 nginx server 可連線

The image shows four terminal windows illustrating the initial steps of Docker Swarm setup:

- Step 1:** A terminal window showing the command `docker swarm init` being executed. The output indicates that the current node is now a manager.
- Step 2:** A terminal window showing the command `docker swarm join --token SWMTKN-1-2azyoum5ihjz4bo2m64g1ori7q4lpg8v12wruk0p87a98xvzu-7qir5j4ds15jei3u4duor1sut 172.17.0.2:2377`. The output shows the node joining the swarm as a worker.
- Step 3:** A terminal window showing the command `docker network create --driver overlay swarm-net`. The output shows the network being created.
- Step 4:** A terminal window showing the command `docker network ls`. The output lists the available networks, including the newly created `swarm-net`.

The image shows a terminal window with the following commands and output:

```
grason@LAPTOP-3OV3PCQM:~$ docker exec -it manager sh
/ # docker network ls
NETWORK ID      NAME                DRIVER  SCOPE
61267b51373d    bridge              bridge  local
321ca0c3173f    docker_gwbridge     bridge  local
f4feeb5dfc89    host                host    local
x9f0fs4xa5h9    ingress             overlay  swarm
bbf285b35a57    none                null    local
rlurt3ueitbq    swarm-net           overlay  swarm

/ # docker service ls
ID                NAME                MODE                REPLICAS  IMAGE                PORTS
c08li4qeyuf1      test-service         replicated          5/3        nginx:latest         *:8080->80/tcp

/ # docker service ps test-service
ID                NAME                IMAGE                NODE        DESIRED STATE  CURRENT STATE        ERROR
td0abygp2c6w      test-service.1      nginx:latest         manager     Running         Running 17 minutes ago
qp6s0nm7e229      \_ test-service.1    nginx:latest         worker2     Shutdown        Running 3 hours ago
82fr7a0mxixc      test-service.2      nginx:latest         manager     Running         Running 17 minutes ago
8m0fq9lpd35h      \_ test-service.2    nginx:latest         manager     Shutdown        Failed 17 minutes ago  "task: non-zero ex
it (255)"
66n6hc054g54      test-service.3      nginx:latest         manager     Running         Running 17 minutes ago
gtt8bosf0xis      \_ test-service.3    nginx:latest         worker1     Shutdown        Running 3 hours ago

/ # |
```

```
grason@LAPTOP-30V3PCQM:~$ docker exec -it worker1 sh
/ # curl 172.17.0.2:8080
curl: (7) Failed to connect to 172.17.0.2 port 8080 after 0 ms: Could not connect to server
/ # curl 172.17.0.4:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ # █
```

6

4. 總結

Docker Swarm 提供了一個簡單、高效的解決方案來處理容器網路隔離和路由配置問題。對於小型群集或開發測試環境，Swarm 內建的網路和服務發現機制能夠快速解決常見的網路配置挑戰。實作 Swarm 後，忽然瞭解 k8s 的 node，與 swarm 中 worker 其實是同樣的概念，同樣藉由內部 docker(docker in docker)的方式進行的容器排程工作。