本小节将展示如何使用[管理Docker引擎的Swarm工具](https://docs.docker.com/engine/swarm/)，来编排部署和管理一个由3个节点构成的以安全模式启动的CockroachDB集群。

如果集群只是用做测试环境而非生产环境，对安全性没有要求，可以选择非安全模式。

前言

相关术语

英文|中文|简介 -|- instance|实例|物理或虚拟的机器。本小节将使用3台机器，每台机器运行一个CockroachDB节点。 [Docker Engine](https://docs.docker.com/engine/)|Docker引擎| Docker核心应用，创建和运行容器。本小节将为每台机器安装和启动Docker引擎。 [swarm](https://docs.docker.com/engine/swarm/key-concepts/#/swarm)|-|由一组Docker引擎组成的一个虚拟的主机 [swarm node](https://docs.docker.com/engine/swarm/how-swarm-mode-works/nodes/)|swarm节点|swarm的每个成员视为一个节点，本小节每台机器为一个swarm节点，其中一个主节点，两个工作节点。用户需要向主节点提交服务内容，主节点再将工作分发给工作节点。 [service](https://docs.docker.com/engine/swarm/how-swarm-mode-works/services/)|服务|执行在swarm节点上的任务内容。本小节将定义3个服务，每个服务在容器中启动一个CockroachDB节点并将节点组成一个集群。每个服务还需要通过可解析的DNS名称确保重启时的稳定网络身份。 [secret](https://docs.docker.com/engine/swarm/secrets/)|-|secret是Docker用于管理容器在运行时需要的敏感数据的机制。 由于CockroachDB使用TLS证书对节点间和客户端/节点通信进行身份验证和加密，用户需要为每个证书创建secret并在服务中使用。 [overlay network](https://docs.docker.com/engine/userguide/networking/#/an-overlay-network-with-docker-engine-swarm-mode)|覆盖网络|实现集群节点之间的通信。本小节将创建一个覆盖网络，用于每个服务。

Step 1: 创建实例

集群由3个节点组成，需要为集群的每个节点创建一个实例机器。

* 针对GCE的创建流程，可以参照[Deploy CockroachDB on GCE](https://www.cockroachlabs.com/docs/stable/deploy-cockroachdb-on-google-cloud-platform-insecure.html)的相关内容
* 针对AWS的创建流程，可以参照[Deploy CockroachDB on AWS](https://www.cockroachlabs.com/docs/stable/deploy-cockroachdb-on-aws-insecure.html)的相关内容

确保每台机器的以下端口能被CockroachDB使用：

* 26257用于集群节点之间和客户端/节点之间的通讯
* 8080用于登陆Admin界面

Step 2: 安装Docker引擎

在每台实例机器上：

* [安装并启动Docker引擎](https://docs.docker.com/engine/installation/)
* 确认Docker daemon在后台运行

sudo docker **version**

Step 3: 启动swarm

* 选择一台实例机器作为管理节点，[初始化swarm](https://docs.docker.com/engine/swarm/swarm-tutorial/create-swarm/)

sudo docker swarm init *--advertise-addr 10.142.0.2*

Swarm initialized: current node (414z67gr5cgfalm4uriu4qdtm) is now a manager

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

$ docker swarm join \

*--toke SWMTKN-1-5vwxyi6zl3cc62lqlhi1jrweyspi8wblh2i3qa7kv277fgy74n-e5eg5c7ioxypjxlt3rpqorh15 \*

10.142.0.2:237

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

* 在另外两台机器上，[创建工作节点](https://docs.docker.com/engine/swarm/swarm-tutorial/add-nodes/)并通过docker swarm join命令将节点加入到swarm

sudo docker swarm join --to SWMTKN-1-5vwxyi6zl3cc62lqlhi1jrweyspi8wblh2i3qa7kv277fgy74n-e5eg5c7ioxypjxlt3rpqorh15 10.142.0.2:2377

This node joined a swarm as a worker.

* 在管理节点上检查swarm的运行状态

sudo docker node ls

ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS

414z67gr5cgfalm4uriu4qdtm \* instance-1 Ready Active Leader

ae144s35dx1p1lcegh6bblyed instance-2 Ready Active

aivjg2joxyvzvbksjsix27khy instance-3 Ready Active

Step 4: 创建覆盖网路

在管理节点上创建一个覆盖网络，用于swarm内容器之间相互进行通讯。

sudo docker network create --driver overlay --attachable cockroachdb

--attachable标签使得在Docker上运行的非swarm容器能够访问网络上的服务，使得服务更易于交互使用。

Step 5: 创建安全资源

若选择非安全模式，则跳过此步骤。

安全模式下的CockroachDB集群使用TLS证书进行为节点间和客户端/节点认证和通信进行加密。 在此步骤中用户将在管理节点的实例机器上安装CockroachDB，使用[cockroach cert](http://doc.cockroachchina.baidu.com/#deploy/create-security-certificates/)命令生成CA证书、集群节点、客户端证书和密钥对，并使用[docker secret create](https://docs.docker.com/engine/reference/commandline/secret_create/)命令分配这些文件到Docker [secret](https://docs.docker.com/engine/swarm/secrets/)，供您的Docker服务使用。

准备工作

* 在管理节点上安装最新版本的CockroachDB

*# Get the latest CockroachDB tarball:*

wget https://binaries.cockroachdb.com/cockroach-v2.0.5.linux-amd64.tgz

*# Extract the binary:*

tar -xf cockroach-v2.0.5.linux-amd64.tgz --strip=1 cockroach-v2.0.5.linux-amd64/cockroach

*# Move the binary:*

sudo mv cockroach /usr/local/bin

* 创建certs目录和保存CA密钥的安全目录

mkdir certs

mkdir my-safe-directory

* 创建CA证书和密钥

cockroach cert create-ca --certs-dir=certs --ca-key=my-safe-directory/ca.key

*#检查是否有ca.crt文件*

ls certs

* 使用docker secret create命令为ca.crt创建Docker secret

**WARNING:** 需要确保ca.crt文件的安全，并做必要的备份，一旦丢失则无法为集群添加节点和客户端。

sudo docker secret create **ca**-crt certs/**ca**.crt

该命令为secret（ca-crt）分配名称，并标记cockroach命令生成的CA证书文件的位置。用户根据需要可修改自定义的名称，并在下述步骤进行相应的名称替换。

为节点证书和密钥创建secret

* 为第一个节点创建证书和密钥

cockroach cert create-node cockroachdb-1 localhost 127.0.0.1 --certs-**dir**=certs --**ca**-key=my-safe-directory/**ca**.key

#检查是否生成**ca**.crt、node.crt、node.key三个文件

**ls** certs

该命令将证书/密钥对发布到用于节点（cockroachdb-1）的服务名称以及本地地址，使用本地地址能在容器内方便地运行内置SQL shell和其他CockroachDB客户端命令。

* 为第一个节点的证书和密钥创建Docker secrets

sudo docker secret **create** cockroachdb-1-crt certs/node.crt

sudo docker secret **create** cockroachdb-1-**key** certs/node.**key**

该命令为secret（cockroachdb-1-crt和cockroachdb-1-key）命名，并标记cockroach命令生成的证书和密钥的地址。

* 为第二个节点创建证书和密钥，使用--overwrite标签覆盖第一个节点对应的文件

cockroach cert create-node --overwrite cockroachdb-2 localhost 127.0.0.1 --certs-**dir**=certs --**ca**-key=my-safe-directory/**ca**.key

#检查是否生成**ca**.crt、node.crt、node.key三个文件

**ls** certs

* 为第二个节点的证书和密钥创建Docker secrets

sudo docker secret **create** cockroachdb-2-crt certs/node.crt

sudo docker secret **create** cockroachdb-2-**key** certs/node.**key**

* 为第三个节点创建证书和密钥，使用--overwrite标签覆盖第二个节点对应的文件

cockroach cert create-node --overwrite cockroachdb-3 localhost 127.0.0.1 --certs-**dir**=certs --**ca**-key=my-safe-directory/**ca**.key

#检查是否生成**ca**.crt、node.crt、node.key三个文件

**ls** certs

* 为第三个节点的证书和密钥创建Docker secrets

sudo docker secret **create** cockroachdb-3-crt certs/node.crt

sudo docker secret **create** cockroachdb-3-**key** certs/node.**key**

为用户证书和密钥创建secret

* 为root用户创建客户端证书和密钥

cockroach cert create-client root --certs-**dir**=certs --**ca**-key=my-safe-directory/**ca**.key

* 为root用户的客户端证书和密钥创建Docker secrets

sudo docker secret create cockroachdb-root-crt certs/client.root.crt

sudo docker secret create cockroachdb-root-key certs/client.root.key

Step 6: 启动CockroachDB集群

创建swarm服务

在管理节点上为每个CockroachDB节点创建swarm服务

非安全模式

*# Start the first service:*

sudo docker service create \

--replicas 1 \

--name cockroachdb-1 \

--hostname cockroachdb-1 \

--network cockroachdb \

--mount type=volume,source=cockroachdb-1,target=/cockroach/cockroach-data,volume-driver=local \

--stop-grace-period 60s \

--publish 8080:8080 \

cockroachdb/cockroach:v2.0.5 start \

--join=cockroachdb-1:26257,cockroachdb-2:26257,cockroachdb-3:26257 \

--cache=.25 \

--max-sql-memory=.25 \

--logtostderr \

--insecure

*# Start the second service:*

sudo docker service create \

--replicas 1 \

--name cockroachdb-2 \

--hostname cockroachdb-2 \

--network cockroachdb \

--mount type=volume,source=cockroachdb-2,target=/cockroach/cockroach-data,volume-driver=local \

--stop-grace-period 60s \

cockroachdb/cockroach:v2.0.5 start \

--join=cockroachdb-1:26257,cockroachdb-2:26257,cockroachdb-3:26257 \

--cache=.25 \

--max-sql-memory=.25 \

--logtostderr \

--insecure

*# Start the third service:*

sudo docker service create \

--replicas 1 \

--name cockroachdb-3 \

--hostname cockroachdb-3 \

--network cockroachdb \

--mount type=volume,source=cockroachdb-3,target=/cockroach/cockroach-data,volume-driver=local \

--stop-grace-period 60s \

cockroachdb/cockroach:v2.0.5 start \

--join=cockroachdb-1:26257,cockroachdb-2:26257,cockroachdb-3:26257 \

--cache=.25 \

--max-sql-memory=.25 \

--logtostderr \

--insecure

安全模式

*# Create the first service:*

sudo docker service create \

--replicas 1 \

--name cockroachdb-1 \

--hostname cockroachdb-1 \

--network cockroachdb \

--mount type=volume,source=cockroachdb-1,target=/cockroach/cockroach-data,volume-driver=local \

--stop-grace-period 60s \

--publish 8080:8080 \

--secret source=ca-crt,target=ca.crt \

--secret source=cockroachdb-1-crt,target=node.crt \

--secret source=cockroachdb-1-key,target=node.key,mode=0600 \

--secret source=cockroachdb-root-crt,target=client.root.crt \

--secret source=cockroachdb-root-key,target=client.root.key,mode=0600 \

cockroachdb/cockroach:v2.0.5 start \

--join=cockroachdb-1:26257,cockroachdb-2:26257,cockroachdb-3:26257 \

--cache=.25 \

--max-sql-memory=.25 \

--logtostderr \

--certs-dir=/run/secrets

*# Create the second service:*

sudo docker service create \

--replicas 1 \

--name cockroachdb-2 \

--hostname cockroachdb-2 \

--network cockroachdb \

--stop-grace-period 60s \

--mount type=volume,source=cockroachdb-2,target=/cockroach/cockroach-data,volume-driver=local \

--secret source=ca-crt,target=ca.crt \

--secret source=cockroachdb-2-crt,target=node.crt \

--secret source=cockroachdb-2-key,target=node.key,mode=0600 \

--secret source=cockroachdb-root-crt,target=client.root.crt \

--secret source=cockroachdb-root-key,target=client.root.key,mode=0600 \

cockroachdb/cockroach:v2.0.5 start \

--join=cockroachdb-1:26257,cockroachdb-2:26257,cockroachdb-3:26257 \

--cache=.25 \

--max-sql-memory=.25 \

--logtostderr \

--certs-dir=/run/secrets

*# Create the third service:*

sudo docker service create \

--replicas 1 \

--name cockroachdb-3 \

--hostname cockroachdb-3 \

--network cockroachdb \

--mount type=volume,source=cockroachdb-3,target=/cockroach/cockroach-data,volume-driver=local \

--stop-grace-period 60s \

--secret source=ca-crt,target=ca.crt \

--secret source=cockroachdb-3-crt,target=node.crt \

--secret source=cockroachdb-3-key,target=node.key,mode=0600 \

--secret source=cockroachdb-root-crt,target=client.root.crt \

--secret source=cockroachdb-root-key,target=client.root.key,mode=0600 \

cockroachdb/cockroach:v2.0.5 start \

--join=cockroachdb-1:26257,cockroachdb-2:26257,cockroachdb-3:26257 \

--cache=.25 \

--max-sql-memory=.25 \

--logtostderr \

--certs-dir=/run/secrets

上述每个命令各自创建一个服务，安全地启动容器，将其连接到覆盖网络，并在挂载了用于持久存储的本地卷的容器上启动CockroachDB节点。

* sudo docker service create用于创建服务的Docker的命令
* --replicas服务包含的容器的数量。此处每个服务包含一个容器，容器内运行一个CockroachDB节点，因此取值为1。
* --name服务的名称
* --hostname容器的主机名，作为地址监听容器的网络连接。
* --network容器连接的覆盖网络，详见**Step4**
* --mount将挂载一个与服务同名的本地卷。 在此容器中运行的节点的数据和日志将存储在对应机器上的/cockroach/cockroach-data目录下并在重启容器时复用，但不保证重新启动时使用同一机器。

**NOTE:** 如果用户计划更换或添加实例机器，建议使用远程存储而不是本地硬盘。 选择卷驱动程序为每个CockroachDB实例[创建一个远程卷](https://docs.docker.com/engine/reference/commandline/volume_create/)，然后在上面命令中指定远程卷，替换volume-driver=local部分。具体地，例如使用GCE卷驱动，则替换内容为volume-driver=gce。

* --stop-grace-period设置一个缓冲时间，在可能的情况下为CockroachDB提供足够的时间进行优雅地关闭。
* --publish使Admin用户界面可以通过任何运行swarm节点实例的IP地址和指定的端口8080去访问。值得注意的是，即使仅在第一个节点的服务中设置了该标记，swarm也会通过路由网格在每个swarm节点上开放此端口。 详见[Publishing ports](https://docs.docker.com/engine/swarm/services/#publish-ports)
* --secret指定一个secret用于保护节点。对于节点和客户端的证书和密钥的secrets，source配置项指定secrets的名称，target配置项指定了cockroach start和cockroach sql标签对应的值。此外，对于节点和客户端的密钥secrets，mode配置项设置了文件的访问权限为0600。默认情况下Docker文件权限是0400，此时无法和CockroachDB内置SQL客户端配合使用。
* cockroachdb/cockroach:v2.0.5 start ...用于在容器中以非安全模式[启动节点](http://doc.cockroachchina.baidu.com/#deploy/start-a-node/)的命令，并通知其他节点通过于服务名称匹配的、确定的网络地址，相互之间进行通信。

验证服务

sudo docker service ls

ID NAME MODE REPLICAS IMAGE

a6g0ur6857j6 cockroachdb-1 replicated 1/1 cockroachdb/cockroach:v2.0.5

dr81a756gaa6 cockroachdb-2 replicated 1/1 cockroachdb/cockroach:v2.0.5

il4m7op1afg9 cockroachdb-3 replicated 1/1 cockroachdb/cockroach:v2.0.5

**TIPS:** 服务指定CockroachDB节点将输出重定向到stderr，因此如果需要通过节点的日志进行问题定位，可以在容器对应的机器上执行[sudo docker logs <container id>](https://docs.docker.com/engine/reference/commandline/logs/)命令。

初始化集群

现在所有的CockroachDB节点都处于运行状态，此时用户需要显示地借助sudo docker run命令在其中一个节点上运行cockroach init命令去初始化集群。 cockroach init命令将初始化集群，使其进入可用状态。

*# 非安全模式*

sudo docker run -it --rm --network=cockroachdb cockroachdb/cockroach:v2.0.5 init --host=cockroachdb-1 --insecure

*# 安全模式*

sudo docker run -it --rm --network cockroachdb --mount type=bind,source="$(pwd)/certs",target=/cockroach/certs,readonly cockroachdb/cockroach:v2.0.5 init --host=cockroachdb-1 --certs-dir=certs

该命令挂载了包含了root用户客户端证书和密钥的certs目录，作为容器的一个卷，并在访问集群的时候使用。

Step 7: 使用内置SQL客户端

* 使用sudo docker run命令启动一个容器连接CockroachDB网络，执行内置SQL shell，连接到集群：

*#非安全模式*

sudo docker run -it --rm --network=cockroachdb cockroachdb/cockroach:v2.0.5 sql --host=cockroachdb-1 --insecure

*#安全模式*

sudo docker run -it --rm --network cockroachdb --mount type=bind,source="$(pwd)/certs",target=/cockroach/certs,readonly cockroachdb/cockroach:v2.0.5 sql --host=cockroachdb-1 --certs-dir=certs

* 创建nodetest数据库

**CREATE** **DATABASE** nodetest;

* 使用CTRL+D、CTRL+C或\q退出SQL shell

Step 8: 监控集群

访问https://<any node's external IP address>:8080，登陆Admin界面来监控集群。

**NOTE:** 第一个节点服务配置了对外服务的端口8080，使得用户能够在swarm集群外部访问到Admin界面。访问Admin界面时浏览器会将CockroachDB创建的证书视作无效的，需要忽略警告。

登陆界面后，可以验证：

* 在**Node List**中查看所有节点是否在集群中正常运行。
* 在**Databases**标签页中检查是否存在nodetest。

Step 9: 模拟节点丢失

先前我们配置了三个服务，每个节点运行其中一个。Docker Swarm将确保始终有三个节点在运行。 如果节点出现故障，Docker Swarm将自动创建具有相同网络身份和存储的新节点。

* 使用sudo docker ps命令获取运行CockroachDB节点的容器ID

sudo docker ps | grep cockroachdb

32769a6dd664 cockroachdb/cockroach:v2.0.5 "/cockroach/cockroach" 10 minutes ago Up 10 minutes 8080/tcp, 26257/tcp cockroachdb-2.1.0wigdh8lx0ylhuzm4on9bbldq

* 使用sudo docker kill停止节点，移除容器

sudo docker kill <container ID>

* 验证Docker Swarm是否自动创建新的容器，重启节点

sudo docker ps | grep cockroachdb

4a58f86e3ced cockroachdb/cockroach:v2.0.5 "/cockroach/cockroach" 7 seconds ago Up 1 seconds 8080/tcp, 26257/tcp cockroachdb-2.1.cph86kmhhcp8xzq6a1nxtk9ng

在Admin界面的**Node List**查看是否有3个节点在运行。

Step 10: 拓展集群

拓展集群的节点数量：

* 创建新的机器（参考**Step 1**）
* 在新机器上安装Docker引擎（参考**Step 2**）
* 将机器添加到swarm集群作为工作节点（参考**Step 3.2**）
* 创建节点的安全资源（参考**Step 5.7和5.8**）
* 创建新服务，启动一个新的CockroachDB节点并添加到集群（参考**Step 6.1**）

Step 11: 停止集群

在管理节点上删除服务，关闭CockroachDB集群

sudo docker service rm cockroachdb-1 cockroachdb-2 cockroachdb-3

cockroachdb-1

cockroachdb-2

cockroachdb-3

如需同时移除集群节点使用的持久卷和secrets，请执行：

*# Identify the name of the local volume:*

sudo docker volume ls

*# Remove the local volume:*

sudo docker volume rm cockroachdb-1

*# Identify the name of secrets:*

sudo docker secrets ls

*# Remove the secrets if necessary:*

sudo docker secret rm ca-crt cockroachdb-1-crt cockroachdb-1-key