文件编号：GZ-DEV-PULSAR-INSTALL

文件版本：1.0

保密级别：保密

Pulsar集群部署
(跨主机基于Docker)

文件创建人	陈宗朗	**文件创建时间**	2023-07-01
文件最后编辑人	陈宗朗	**文件最后编辑时间**	2023-07-01
文件审阅人		**文件审阅时间**	
文件批准人		**文件批准时间**	
文件发布范围	内部研发&运维		

文档修订记录

版本号	修订日期	修订人	修订说明	修订状态	审核人	批准人
V1.0	2023-07-01	陈宗朗	初版	A		

修订状态：A--增加，M--修改，D--删除

日期格式：YYYY-MM-DD

Pulsar集群部署（跨主机基于Docker）

跨主机基于Docker 部署 Pulsar 集群（ zookeeper、bookkeeper、broker 集群）。

方案

- 采用三台服务器实现整个 Pulsar 集群。
- 在每台服务器分别部署一个 zookeeper、bookkeeper、broker 节点，即 zookeeper、bookkeeper、broker 集群均有 3 个节点，确保整个集群的高可用性。
- 采用挂载配置文件的方式配置集群，不在 docker 启动命令指定集群参数，尽量避免误操作，免得破坏集群环境，导致需要重建集群。
- 集群相关配置不固定IP，采用 host 的方式。
- 与二进制安装部署类似（二进制安装需要另行配置 Java 环境）。
- 更换到其他服务器环境安装，理论上采用本方案安装，只需要更改 host 的 ip 映射以及修改 docker 挂载的宿主机目录即可。
- 暂时未实践 Docker-Swarm 的部署方式。

准备

操作系统

- RELEASE: `CentOS Linux release 7.9.2009 (Core)`
- SOFTWARE: `minimal`

角色分配

- node-01: `192.168.159.151`
- node-02: `192.168.159.152`
- node-03: `192.168.159.153`

软件包裹

- docker
- apachepulsar/pulsar-all:2.11.1

Host配置

在 `node-01`、`node-02`、`node-03` 的 `/etc/hosts` 中追加如下内容：

```
cat >> /etc/hosts << EOF
192.168.159.151   zookeeper01     bookkeeper01    broker01
192.168.159.152   zookeeper02     bookkeeper02    broker02
192.168.159.153   zookeeper03     bookkeeper03    broker03
EOF
```

firewalld配置

在所有主机上按照如下规则修改 firewalld 的配置：

```
firewall-cmd --zone=internal --add-port=3888/tcp --permanent
firewall-cmd --zone=internal --add-port=2888/tcp --permanent
firewall-cmd --zone=internal --add-port=2181/tcp --permanent
firewall-cmd --zone=internal --add-port=8080/tcp --permanent
firewall-cmd --zone=internal --add-port=8443/tcp --permanent
firewall-cmd --zone=internal --add-port=6650/tcp --permanent
firewall-cmd --zone=internal --add-port=6651/tcp --permanent
firewall-cmd --zone=internal --add-port=9990/tcp --permanent
firewall-cmd --zone=internal --add-port=3181/tcp --permanent
firewall-cmd --zone=internal --add-port=8000/tcp --permanent
firewall-cmd --zone=internal --add-port=8001/tcp --permanent
firewall-cmd --zone=internal --add-port=8002/tcp --permanent
firewall-cmd --reload
```

注意事项

需要注意在网卡配置中是否存在 `ZONE=internal`，否则不生效。端口策略可按照如下思路设计：

- 内网：`ZONE=internal`
- 公网：`ZONE=public`

> 注意：严禁直接暴露端口在公网环境。

创建用户

为便于管理，创建 `pulsar` 用户用于安装 Pulsar 集群：

```
useradd pulsar
passwd pulsar
```

为便于使用，给数据库用户添加 `sudo` 权限：

```
vim /etc/sudoers
```

在 `root    ALL=(ALL)       ALL` 增加如下内容并保存：

```
pulsar   ALL=(ALL)       ALL
```

将新添加的用户添加到 docker 用户组，并重启 docker 服务：

```
usermod -aG docker pulsar
systemctl restart docker
```

创建目录

为便于数据管理，切换到 `pulsar` 用户创建数据目录：

```
su - pulsar
mkdir -p zookeeper/{conf,data,logs} bookkeeper/{conf,journal,ledgers}
broker/{conf,data,logs}
```

配置

提取配置文件模板

运行容器

运行临时容器用于提取文件模板（仅用于提取配置文件模板，提取后，即可删除该容器）：

```
docker run  -itd --privileged=true --net host --name pulsar-temp
apachepulsar/pulsar-all:2.11.1
```

提取模板

拷贝配置文件模板到本地目录：

```
docker cp pulsar-temp:/pulsar/conf ./
```

修改Zookeeper配置模板

修改拷贝出来的 Zookeeper 配置模板：

```
sed -i 's#dataDir=data/zookeeper#dataDir=/pulsar/data/zookeeper#g'
conf/zookeeper.conf
sed -i '29 a dataLogDir=/pulsar/logs' conf/zookeeper.conf
cat >> conf/zookeeper.conf << EOF
server.1=zookeeper01:2888:3888
server.2=zookeeper02:2888:3888
server.3=zookeeper03:2888:3888
EOF
```

修改Bookkeeper配置模板

修改拷贝出来的 Bookkeeper 配置模板：

```
sed -i "s/#
extraServerComponents=/extraServerComponents=org.apache.bookkeeper.stream.server.
StreamStorageLifecycleComponent/g" conf/bookkeeper.conf
sed -i
"s#zkServers=localhost:2181#zkServers=zookeeper01:2181,zookeeper02:2181,zookeeper
03:2181#g" conf/bookkeeper.conf
sed -i 's/prometheusStatsHttpPort=8000/prometheusStatsHttpPort=8001/g'
conf/bookkeeper.conf
sed -i 's/httpServerPort=8000/httpServerPort=8002/g' conf/bookkeeper.conf
```

修改Broker配置模板

修改拷贝出来的 Broker 配置模板：

```
sed -i 's/clusterName=/clusterName=pulsar-cluster/g' conf/broker.conf
sed -i
"s/zookeeperServers=/zookeeperServers=zookeeper01:2181,zookeeper02:2181,zookeeper
03:2181/g" conf/broker.conf
sed -i
"s/configurationStoreServers=/configurationStoreServers=bookkeeper01:2181,bookkee
per02:2181,bookkeeper03:2181/g" conf/broker.conf
```

分发配置

将修改完成的 `conf` 目录内的所有文件，分别拷贝到三台节点的 `zookeeper`、`bookkeeper`、`broker` 的 `conf` 目录下。

部署

Zookeeper集群

创建MYID文件

在每台主机上，需要在 `myid` 文件中指定节点的 ID。

- node-01

  ```
  echo 1 > zookeeper/data/myid
  ```

- node-02

  ```
  echo 2 > zookeeper/data/myid
  ```

- node-03

```
echo 3 > zookeeper/data/myid
```

启动容器

在创建好 `myid` 文件后，分别在三台节点启动 zookeeper 容器：

```
sudo docker run -itd --user=root \
--privileged=true \
-v /home/pulsar/zookeeper/conf:/pulsar/conf \
-v /home/pulsar/zookeeper/data:/pulsar/data/zookeeper \
-v /home/pulsar/zookeeper/logs:/pulsar/logs \
--restart=always \
--name zookeeper \
--net host \
apachepulsar/pulsar-all:2.11.1 \
bash -c "bin/apply-config-from-env.py conf/zookeeper.conf && bin/generate-
zookeeper-config.sh conf/zookeeper.conf && exec bin/pulsar zookeeper"
```

在三台节点都启动后，在任一节点执行 `docker logs -f zookeeper` 可看到类似如下关键日志输出：

```
INFO  org.apache.zookeeper.server.quorum.Learner - Successfully connected to
leader, using address: zookeeper02/192.168.159.152:2888
.
.
.
INFO  org.apache.zookeeper.server.quorum.QuorumCnxManager - Received connection
request from /192.168.159.153:57246
.
.
.
INFO  org.apache.zookeeper.server.quorum.FastLeaderElection - Notification: my
state:FOLLOWING; n.sid:3, n.state:LOOKING, n.leader:3, n.round:0x1,
n.peerEpoch:0x0, n.zxid:0x0, message format version:0x2, n.config version:0x0
```

初始化集群元数据

在任一节点执行初始化元数据操作：

```
sudo docker run --net host \
    --name initialize-pulsar-cluster-metadata \
    apachepulsar/pulsar-all:2.11.1 bash -c "bin/pulsar initialize-cluster-
metadata \
--cluster pulsar-cluster \
--zookeeper zookeeper01:2181 \
--configuration-store zookeeper01:2181 \
--web-service-url broker01:8080,broker02:8080,broker03:8080 \
--web-service-url-tls https://broker01:8443,broker02:8443,broker03:8443 \
--broker-service-url pulsar://broker01:6650,broker02:6650,broker03:6650 \
--broker-service-url-tls pulsar+ssl://broker01:6651,broker02:6651,broker03:6651"
```

等待一会可看到如下信息输出，说明初始化完毕：

```
INFO  org.apache.bookkeeper.stream.storage.impl.cluster.ZkClusterInitializer -
Successfully initialized the stream cluster :
num_storage_containers: 16

INFO  org.apache.pulsar.PulsarClusterMetadataSetup - Cluster metadata for
'pulsar-cluster' setup correctly
```

Bookkeeper集群

修改节点信息

- node-01

  ```
  sed -i "s/advertisedAddress=/advertisedAddress=bookkeeper01/g"
  bookkeeper/conf/bookkeeper.conf
  ```

- node-02

  ```
  sed -i "s/advertisedAddress=/advertisedAddress=bookkeeper02/g"
  bookkeeper/conf/bookkeeper.conf
  ```

- node-03

  ```
  sed -i "s/advertisedAddress=/advertisedAddress=bookkeeper03/g"
  bookkeeper/conf/bookkeeper.conf
  ```

启动容器

在三台节点上启动 bookkeeper 容器：

```
sudo docker run -itd --user=root \
--privileged=true \
-v /home/pulsar/bookkeeper/conf:/pulsar/conf \
-v /home/pulsar/bookkeeper/journal:/pulsar/data/bookkeeper/journal \
-v /home/pulsar/bookkeeper/ledgers:/pulsar/data/bookkeeper/ledgers \
--restart=always \
--name bookkeeper \
--net host \
apachepulsar/pulsar-all:2.11.1 \
bash -c "bin/apply-config-from-env.py conf/bookkeeper.conf && exec bin/pulsar
bookie"
```

在三台节点都启动后，在任一节点执行 `docker logs -f bookkeeper` 可看到类似如下关键日志输出：

```
INFO  org.apache.bookkeeper.proto.PerChannelBookieClient - Successfully connected
to bookie: bookkeeper01:3181 [id: 0x1b50b0c7, L:/192.168.159.151:43612 -
R:bookkeeper01/192.168.159.151:3181]
INFO  org.apache.bookkeeper.proto.PerChannelBookieClient - Successfully connected
to bookie: bookkeeper02:3181 [id: 0xbd2b0a87, L:/192.168.159.151:46280 -
R:bookkeeper02/192.168.159.152:3181]
INFO  org.apache.bookkeeper.proto.PerChannelBookieClient - Successfully connected
to bookie: bookkeeper03:3181 [id: 0xaf051716, L:/192.168.159.151:45760 -
R:bookkeeper03/192.168.159.153:3181]
```

初始化元数据

在任一节点执行初始化元数据操作：

```
sudo docker exec -i --user=root \
--privileged=true \
bookkeeper \
bash -c "bin/bookkeeper shell metaformat"
```

等待执行过程中，根据提示输入 `Y` 即可看到如下 `Successfully formatted BookKeeper metadata` 日志信息：

```
2023-07-01T23:57:45,753+0000 [main-EventThread] INFO
org.apache.bookkeeper.zookeeper.ZooKeeperWatcherBase - ZooKeeper client is
connected now.
Ledger root already exists. Are you sure to format bookkeeper metadata? This may
cause data loss. (Y or N) Y
2023-07-01T23:58:16,037+0000 [main] INFO
org.apache.bookkeeper.discover.ZKRegistrationManager - Successfully formatted
BookKeeper metadata
2023-07-01T23:58:16,176+0000 [main] INFO  org.apache.zookeeper.ZooKeeper -
Session: 0x200013ad91a0006 closed
2023-07-01T23:58:16,176+0000 [main-EventThread] INFO
org.apache.zookeeper.ClientCnxn - EventThread shut down for session:
0x200013ad91a0006
```

Broker集群

修改节点信息

- node-01

  ```
  sed -i "s/bindAddress=0.0.0.0/bindAddress=broker01/g" conf/broker.conf
  sed -i "s/advertisedAddress=/advertisedAddress=broker01/g" conf/broker.conf
  ```

- node-02

  ```
  sed -i "s/bindAddress=0.0.0.0/bindAddress=broker02/g" conf/broker.conf
  sed -i "s/advertisedAddress=/advertisedAddress=broker02/g" conf/broker.conf
  ```

- node-03

```
sed -i "s/bindAddress=0.0.0.0/bindAddress=broker03/g" conf/broker.conf
sed -i "s/advertisedAddress=/advertisedAddress=broker03/g" conf/broker.conf
```

启动容器

在三台节点上启动 broker 容器：

```
sudo docker run -itd --user=root \
--privileged=true \
-v /home/pulsar/broker/conf:/pulsar/conf \
-v /home/pulsar/broker/logs:/pulsar/logs \
--restart=always \
--name broker \
--net host \
apachepulsar/pulsar-all:2.11.1 \
bash -c "bin/apply-config-from-env.py conf/broker.conf && exec bin/pulsar broker"
```

等待启动完成后，可看到 pulsar 服务已启动完成的日志输出：

```
INFO  org.apache.pulsar.PulsarBrokerStarter - PulsarService started.
```

验证

查看Bookkeeper集群

可执行命令验证是否启动集群：

```
docker exec -i --user=root  --privileged=true bookkeeper \
bash -c "bin/bookkeeper shell bookiesanity"
```

如果不出意外，会看到成功的提示：

```
2023-07-02T00:24:44,606+0000 [bookkeeper-io-3-1] INFO
org.apache.bookkeeper.proto.PerChannelBookieClient - Disconnected from bookie
channel [id: 0x2a3aa818, L:/192.168.159.151:43626 !
R:bookkeeper01/192.168.159.151:3181]
2023-07-02T00:24:46,649+0000 [main-EventThread] INFO
org.apache.zookeeper.ClientCnxn - EventThread shut down for session:
0x300013af7a5000e
2023-07-02T00:24:46,649+0000 [main] INFO  org.apache.zookeeper.ZooKeeper -
Session: 0x300013af7a5000e closed
2023-07-02T00:24:46,650+0000 [main] INFO
org.apache.bookkeeper.tools.cli.commands.bookie.SanityTestCommand - Bookie sanity
test succeeded
```

查看Broker节点

完成启动后，即可通过 `pulsar-admin` 命令查看集群的节点情况：

```
docker exec -i --user=root  --privileged=true broker \
bash -c "bin/pulsar-admin --admin-url http://broker01:8080 brokers list pulsar-
cluster"
```

如不出意外，可看到终端显示的 broker 节点：

```
"broker01:8080"
"broker02:8080"
"broker03:8080"
```

Hello Pulsar

到此，Pulsar 集群已搭建完成，可以进行测试了，使用 `pulsar-client` 命令生产并消费消息。

首先开启一个终端，执行消费监听，这样就可以实时查看到生产的消息，例如计划消费 100 条消息，可在 pulsar01 上面执行如下命令：

```
docker exec -i --user=root  --privileged=true broker \
bash -c "bin/pulsar-client --url pulsar://broker01:6650 consume
persistent://public/default/test   -s 'first-subscription' -n 100"
```

再开一个终端，执行生产消息，例如在 pulsar02 上面执行如下命令：

```
docker exec -i --user=root  --privileged=true broker \
bash -c "bin/pulsar-client --url pulsar://broker01:6650 produce \
  persistent://public/default/test \
  -n 1 \
  -m 'Hello Pulsar'"
```

此时可看到在生产端看到如下输出信息，表示已生产了一条消息：

```
2023-07-02T00:32:54,169+0000 [main] INFO
org.apache.pulsar.client.cli.PulsarClientTool - 1 messages successfully produced
```

回到消费监听端，可看到已经消费了一条消息：

```
----- got message -----
key:[null], properties:[], content:Hello Pulsar
```

至此，已完成 Pulsar 集群的搭建及测试。