

DCMP 5.2 容器离线部署与维护手册

文档版本：2026-08-06
适用宿主机：CentOS 8、麒麟 V10 SP3
适用架构：x86_64 (Docker amd64) 、aarch64 (Docker arm64)

1. 适用范围

本文只说明 DCMP 5.2 容器离线交付包的现场安装、配置、启动、停止、重启、状态检查、日志查看、备份恢复、镜像更新和故障排查。

2. 技术支持

遇到安装、启动、访问、数据库、搜索、文件转换、备份或升级问题时，请先保留现场，再联系：

- 联系人：王先平
- 电话：13641931142
- 邮箱：wangxianping@dcmp.cn

不要在收集现场前删除容器、镜像、 /usr/dcmp/dbdata 或 /usr/dcmp/dbfiles 。

3. 交付版本、架构和文件名

版本	最终镜像	amd64 交付包	arm64 交付包
社区版	dcmp-community:5.2	dcmp-community-container-offline-amd64-5.2.tar.gz	dcmp-community-container-offline-arm64-5.2.tar.gz

镜像归档名：

版本	amd64	arm64
社区版	images/dcmp-community-5.2-amd64.tar.gz	images/dcmp-community-5.2-arm64.tar.gz

同一架构的镜像可部署到 CentOS 8 或麒麟 V10 SP3 Docker 宿主机。x86_64 主机只能使用 amd64 包，aarch64 主机只能使用 arm64 包。

交付目录结构：

```
<delivery-dir>/
├─ ARCH
├─ images/<image>-5.2-<arch>.tar.gz
├─ docker/docker-26.1.4.tgz
├─ config/config.json
├─ backups/
├─ config.env
└─ install-docker.sh
```

```
|— deploy.sh
|— start.sh
|— stop.sh
|— status.sh
|— backup.sh
|— restore.sh
|— README.md
|— DCMP_CONTAINER_DEPLOYMENT_MAINTENANCE_MANUAL.md
```

`config/config.json` 是从镜像导出的只读参考副本。修改这个副本不会修改镜像，也不会自动修改正在运行的容器。

4. 安装前检查

4.1 主机和资源

使用 root 执行：

```
uname -m
cat /etc/os-release
free -h
df -h / /usr /opt
command -v docker || true
```

建议：

- 根分区至少保留 40 GiB 可用空间；
- 默认容器内存上限为 12 GiB；
- 内核支持 overlay2、cgroup、bridge 和 IPv4 转发；
- 没有同名 `dcmp` 容器，或已确认可以替换现有容器；
- `dcmp_net` 的默认网段不与现场网络冲突；

4.2 解包

交付包可放在 `/opt`、`/root` 或其他安装临时目录，但不要解压到 `/usr/dcmp` 数据目录内。

正式版 amd64 示例：

```
mkdir -p /opt/dcmp-install
cd /opt/dcmp-install
tar -xzf /path/to/dcmp-container-offline-formal-amd64-5.2.tar.gz
cd dcmp-container-offline-formal-amd64-5.2
chmod 0755 ./*.sh
cat ARCH
```

社区版或 ARM 环境使用对应的交付包和目录名。

5. 配置 config.env

首次部署前检查：

```
vi config.env
```

当前默认值：

```
DCMP_CONTAINER_NAME=dcmp
DCMP_NETWORK_NAME=dcmp_net
DCMP_NETWORK_SUBNET=192.168.1.0/24
DCMP_CONTAINER_IP=192.168.1.22
DCMP_DATA_ROOT=/usr/dcmp
DCMP_MEMORY_LIMIT=12g
MYSQL_BUFFER_POOL_SIZE=4G
DCMP_JAVA_MAX_MB=2048
DCMP_RESTART_POLICY=always
DCMP_IMAGE=dcmp:5.2          # 社区版交付包中为 dcmp-community:5.2
DCMP_ES_HOST=http://127.0.0.1:9200
```

5.1 网络和 IP

默认网络为 `dcmp_net`，网段为 `192.168.1.0/24`，容器 IP 为 `192.168.1.22`。如果现场冲突，必须同时修改网段和 IP，例如：

```
DCMP_NETWORK_SUBNET=192.168.50.0/24
DCMP_CONTAINER_IP=192.168.50.22
```

`deploy.sh` 会把 `DCMP_CONTAINER_IP` 同时作为容器内的 `DCMP_HOST_IP`。不要使用历史错误地址 `192.16.3.22`。

6. Docker 离线安装

宿主机没有可用 Docker 时执行：

```
bash install-docker.sh
```

脚本安装交付包内同架构的 Docker 26.1.4，并写入 Docker/containerd systemd 服务、overlay2 配置、日志轮转和 IPv4 转发设置。

如果宿主机已有 Docker 并承载其他容器，先检查脚本对 `/etc/docker/daemon.json` 和 systemd 服务文件的影响；确认现有 Docker 正常时可以跳过安装脚本。

安装后检查：

```
docker version
docker info | grep -E 'Storage Driver|Cgroup Version'
systemctl is-active docker containerd
```

```
systemctl is-enabled docker containerd
sysctl -n net.ipv4.ip_forward
```

预期 Docker/containerd 均为 `active`、`enabled`，`net.ipv4.ip_forward` 为 `1`。

7. 首次部署

```
cd <delivery-dir>
bash deploy.sh
```

当前 `deploy.sh` 会：

1. 检查 root、Docker、CPU 架构和交付包 `ARCH`；
2. 启用 Docker 端口转发所需的内核参数；
3. 完整执行 `gzip -t` 后导入当前变体镜像；
4. 创建 `${DCMP_DATA_ROOT}/dbdata` 和 `${DCMP_DATA_ROOT}/dbfiles`；
5. 首次部署时从镜像复制两个目录的初始内容，并创建 `dbfiles/.dcmp-seeded`；
6. 删除同名旧容器并按配置创建或调整 `dcmp_net`；
7. 使用静态 IP、`--privileged`、内存上限和 `--restart always` 启动 systemd 容器；
8. 挂载 cgroup、`dbdata` 和 `dbfiles`；
9. 发布全部 TCP/UDP 端口并等待健康检查通过。

部署成功时输出类似：

```
DCMP_HEALTHY
restart=always ip=192.168.1.22
DCMP deployed: container=dcmp persistent=/usr/dcmp/dbdata,/usr/dcmp/dbfiles
```

8. 当前持久化边界

8.1 真正持久化的目录

当前 `deploy.sh` 只挂载两个业务目录：

宿主机路径	容器路径	说明
<code>\${DCMP_DATA_ROOT}/dbdata</code>	<code>/usr/dcmp/dbdata</code>	GreatSQL 数据文件
<code>\${DCMP_DATA_ROOT}/dbfiles</code>	<code>/usr/dcmp/dbfiles</code>	GreatSQL 运行文件及首次初始化标记

默认 `DCMP_DATA_ROOT=/usr/dcmp`，所以宿主机实际路径为：

```
/usr/dcmp/dbdata
/usr/dcmp/dbfiles
```

系统挂载 `/sys/fs/cgroup:/sys/fs/cgroup:ro` 只用于 systemd 容器运行，不是业务数据持久化。

确认实际挂载：

```
docker inspect dcmp --format '{{range .Mounts}}{{println .Source "->" .Destination}}{{end}}'
```

预期只有 cgroup、dbdata 和 dbfiles 三条挂载。

8.2 当前不持久化的内容

以下内容位于容器可写层，停止后再次启动同一容器时仍存在，但删除并重新创建容器后会丢失：

/usr/dcmp/data/elasticsearch	Elasticsearch 索引
/usr/dcmp/upload	上传文件
/usr/dcmp/logs	业务日志
/usr/dcmp/dcmp-agent/logs	Agent 日志
/usr/dcmp/cache	缓存
/usr/dcmp/dbbackup	容器内数据库备份
/usr/dcmp/volume_data	业务卷数据
/usr/dcmp/dcmp-agent/config.json	Agent 现场修改

因此必须注意：

- `bash start.sh`、`bash stop.sh` 只操作同一个容器，不会删除上述内容；
- `bash deploy.sh` 会执行 `docker rm -f` 并重新创建容器，上述内容会随旧容器一起删除；
- 换镜像或重新部署前，如果这些内容需要保留，必须先单独导出；
- `backup.sh` 只备份 dbdata 和 dbfiles，不会备份 ES、upload、日志、Agent 配置或 `/usr/dcmp/dbbackup`。

这是当前交付设计，不要再按旧手册把整个 `/usr/dcmp` 或九个子目录理解为已持久化。

8.3 重新部署前导出非持久化内容

在维护窗口先受控停止容器，再按现场需求复制。示例：

```
cd <delivery-dir>
bash stop.sh
EXPORT=/opt/dcmp-container-export-$(date +%Y%m%d-%H%M%S)
mkdir -p "$EXPORT"
docker cp dcmp:/usr/dcmp/upload "$EXPORT/upload"
docker cp dcmp:/usr/dcmp/dcmp-agent/config.json "$EXPORT/config.json"
docker cp dcmp:/usr/dcmp/dbbackup "$EXPORT/dbbackup"
docker cp dcmp:/usr/dcmp/logs "$EXPORT/logs"
docker cp dcmp:/usr/dcmp/dcmp-agent/logs "$EXPORT/agent-logs"
```

只有确实需要时才导出 ES 数据目录。直接复制运行中的 ES 数据不安全，应先停止容器；恢复 ES 数据也必须保证版本、权限和目录结构一致。

9. 服务、启动和停止

9.1 当前服务集合

`status.sh` 检查以下服务：

```
greatsql elasticsearch dcmp-agent ac1 ap1 dd1 mh1 st1 sa1 crond
```

容器入口和服务控制文件：

```
/usr/local/sbin/dcmp-entrypoint
/usr/local/sbin/dcmp-container-prepare
/usr/local/bin/dcmp-healthcheck
/etc/systemd/system/dcmp-container-prepare.service
/usr/lib/systemd/system/dcmp-container.target
```

9.2 状态检查

```
bash status.sh
docker exec dcmp /usr/local/bin/dcmp-healthcheck
docker exec dcmp systemctl is-active \
    greatsql elasticsearch dcmp-agent ac1 ap1 dd1 mh1 st1 sa1 crond
docker exec dcmp systemctl --failed --no-pager
```

9.3 启动

```
bash start.sh
bash status.sh
```

`start.sh` 会恢复 `config.env` 中的重启策略，再启动已有容器。

9.4 受控停止

```
bash stop.sh
```

当前 `systemd` 容器不会按普通容器方式立即响应 Docker 默认 `SIGTERM`。`stop.sh` 会先停止 DCMP 服务，再结束容器并把重启策略临时设为 `no`。不要用 `docker stop` 或 `docker restart` 替代受控流程。

9.5 重启容器

```
bash stop.sh
bash start.sh
bash status.sh
```

9.6 重启单个服务

```
docker exec dcmp systemctl restart dcmp-agent
docker exec dcmp systemctl restart elasticsearch
```

```
docker exec dcmp systemctl restart ac1
docker exec dcmp systemctl is-active dcmp-agent elasticsearch ac1
```

9.7 宿主机重启

Docker/containerd 设为开机自启，容器使用 `restart=always`。宿主机重启后检查：

```
systemctl is-active docker containerd
docker ps --filter 'name=^/dcmp$'
cd <delivery-dir>
bash status.sh
```

10. Agent 配置

交付目录中的 `config/config.json` 只是参考副本；镜像内默认配置为 `/usr/dcmp/dcmp-agent/config.json`。

临时修改正在运行的容器：

```
docker cp ./config/config.json dcmp:/usr/dcmp/dcmp-agent/config.json
docker exec dcmp chmod 0600 /usr/dcmp/dcmp-agent/config.json
docker exec dcmp systemctl restart dcmp-agent
docker exec dcmp curl -fsS http://127.0.0.1:8808/health
```

该修改保存在当前容器可写层，执行 `deploy.sh` 重新创建容器后会丢失。需要长期保留时，应更新对应社区版或正式版构建源中的 Agent 配置，再重新构建镜像；不要把客户 API Key 或密码写入公共镜像。

11. 端口和访问

当前 `deploy.sh` 发布：

```
TCP: 443 3306 33060 33062
      5000 5001 5002 5003 5004 5005 5006 5021
      8808 9200 9300
UDP: 514 162
```

检查：

```
docker port dcmp
ss -ltnup | grep -E ':(443|3306|33060|33062|5000|5001|5002|5003|5004|5005|5006|5021|8808|9200|9300)\b'
sysctl -n net.ipv4.ip_forward
docker network inspect dcmp_net
```

访问失败时依次检查：容器状态、`status.sh`、单个服务状态、`docker port`、IPv4 转发、防火墙、宿主机路由、Docker 网段冲突和容器 IP。

12. 日志和定时任务

12.1 容器输出

```
docker logs --tail 200 dcmp
docker logs --since 30m dcmp
docker logs -f dcmp
```

Docker 使用 json-file 日志轮转，默认单文件 100 MiB、最多 5 个文件。

12.2 systemd 日志

```
docker exec dcmp journalctl -b --no-pager -n 200
docker exec dcmp journalctl -u greysql -b --no-pager -n 200
docker exec dcmp journalctl -u elasticsearch -b --no-pager -n 200
docker exec dcmp journalctl -u dcmp-agent -b --no-pager -n 200
docker exec dcmp journalctl -u acl -b --no-pager -n 200
docker exec dcmp journalctl -u dcmp-container-prepare -b --no-pager -n 100
```

12.3 文件日志

```
docker exec dcmp du -xhd1 /usr/dcmp/logs
docker exec dcmp sh -c \
    "find /usr/dcmp/logs /usr/dcmp/dcmp-agent/logs -type f -printf '%TY-%Tm-%Td %TH:%TM %s %p\\n'
    2>/dev/null | sort -r | sed -n '1,100p'"
```

这些文件日志当前不持久化；删除并重建容器前需要时先 `docker cp` 导出。

12.4 cron、日志清理和数据库备份任务

```
docker exec dcmp systemctl is-active crond
docker exec dcmp crontab -l
docker exec dcmp /usr/dcmp/bin/logsclean.sh
docker exec dcmp /usr/dcmp/bin/dbbackup.sh --help
```

统一使用 `dbbackup.sh`，不要使用拼写错误或过时的 `sqlbackip.sh`。

13. 备份与恢复

13.1 宿主机持久化备份

```
cd <delivery-dir>
bash backup.sh
```

`backup.sh` 会受控停止容器，只打包 `${DCMP_DATA_ROOT}/dbdata` 和 `${DCMP_DATA_ROOT}/dbfiles`，然后重新启动容器。输出目录：

```
backups/YYYYMMDD-HHMMSS/dcmp-persistent-data.tar.gz
```

检查：

```
BACKUP_DIR=$(find backups -mindepth 1 -maxdepth 1 -type d -printf '%T@ %p\n' | sort -nr | sed -n '1p' | cut -d' ' -f2-)
gzip -t "$BACKUP_DIR/dcmp-persistent-data.tar.gz"
tar -tzf "$BACKUP_DIR/dcmp-persistent-data.tar.gz" | \
  grep -E '^(dbdata|dbfiles)(/|$)' | sed -n '1,20p'
```

`backup.sh` 返回码为 `0` 且 `gzip -t` 通过，才能判定归档生成成功。该备份不包含 ES、上传文件、日志、Agent 配置、容器内 `/usr/dcmp/dbbackup`、缓存或 `volume_data`。

13.2 容器内 XtraBackup

```
docker exec dcmp xtrabackup --version
docker exec dcmp /usr/dcmp/bin/dbbackup.sh full
docker exec dcmp /usr/dcmp/bin/dbbackup.sh inc
```

输出位于容器内 `/usr/dcmp/dbbackup`，当前不持久化。若要在重建容器后保留，必须在重建前导出：

```
docker cp dcmp:/usr/dcmp/dbbackup ./backups/dbbackup-export-$(date +%Y%m%d-%H%M%S)
```

13.3 恢复（高风险）

`restore.sh` 会停止并删除当前容器，删除宿主机现有 `dbdata` 和 `dbfiles`，从归档恢复后重新执行 `deploy.sh`。

执行前必须确认：

- 已完成当前数据的第二份备份；
- 归档来自相同产品变体、相同架构和兼容版本；
- 已安排停机窗口；
- 归档通过 `gzip -t`；
- 已另行保护需要保留的非持久化内容。

```
cd <delivery-dir>
gzip -t backups/YYYYMMDD-HHMMSS/dcmp-persistent-data.tar.gz
bash restore.sh backups/YYYYMMDD-HHMMSS
bash status.sh
```

13.4 真实备份恢复闭环验证

仅在专用测试环境或已批准维护窗口执行。数据库标记和 `dbfiles` 文件都成功恢复，才能证明备份恢复真实有效。

```
cd <delivery-dir>
MARKER=dcmp_br_verify_$(date +%Y%m%d-%H%M%S)_$$
```

```

docker exec dcmp /usr/dcmp/dbfiles/bin/mysql --login-path=backup_user -e \
  "CREATE TABLE IF NOT EXISTS dcmm.dcmp_backup_restore_verify (marker varchar(128) NOT NULL
  PRIMARY KEY, created_at timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP) ENGINE=InnoDB; INSERT INTO
  dcmm.dcmp_backup_restore_verify(marker) VALUES ('$MARKER');"
printf '%s\n' "$MARKER" > "/usr/dcmp/dbfiles/.${MARKER}.txt"

bash backup.sh
BACKUP_DIR=$(find backups -mindepth 1 -maxdepth 1 -type d -printf '%T@ %p\n' | sort -nr | sed -n
'1p' | cut -d' ' -f2-)
gzip -t "$BACKUP_DIR/dcmp-persistent-data.tar.gz"
tar -tzf "$BACKUP_DIR/dcmp-persistent-data.tar.gz" | grep -F "dbfiles/.${MARKER}.txt"

docker exec dcmp /usr/dcmp/dbfiles/bin/mysql --login-path=backup_user -e \
  "DELETE FROM dcmm.dcmp_backup_restore_verify WHERE marker='$MARKER';"
rm -f "/usr/dcmp/dbfiles/.${MARKER}.txt"

bash restore.sh "$BACKUP_DIR"
bash status.sh
docker exec dcmp /usr/dcmp/dbfiles/bin/mysql --login-path=backup_user -Nse \
  "SELECT marker FROM dcmm.dcmp_backup_restore_verify WHERE marker='$MARKER';"
test "$(cat "/usr/dcmp/dbfiles/.${MARKER}.txt")" = "$MARKER"

```

验证结束后只删除本次测试标记：

```

docker exec dcmp /usr/dcmp/dbfiles/bin/mysql --login-path=backup_user -e \
  "DELETE FROM dcmm.dcmp_backup_restore_verify WHERE marker='$MARKER';"
rm -f "/usr/dcmp/dbfiles/.${MARKER}.txt"

```

14. 更新镜像或重新部署

执行新交付包的 `deploy.sh` 会删除并重建同名容器。数据库目录和数据库运行文件因 bind mount 保留，其他容器可写层内容不保留。

更新前：

1. 确认新旧版本均为社区版或均为正式版；
2. 确认 CPU 架构一致；
3. 执行 `backup.sh` 并检查归档；
4. 单独导出需要保留的 upload、ES、Agent 配置、日志和容器内备份；
5. 确认新版本数据库升级 SQL 和 ES 重建/导入方案。

已有 `dbfiles/.dcmp-seeded` 时，`deploy.sh` 不会重新复制镜像中的初始数据库。新镜像带有新的 `dcmm.sql` 并不等于现有数据库会自动升级；数据库结构变更必须执行对应升级脚本并验收。

更新示例：

```

cd <old-delivery-dir>
bash backup.sh

cd <new-delivery-dir>
vi config.env

```

```
bash deploy.sh
bash status.sh
```

更新后至少检查：

```
docker inspect dcmp --format 'image={{.Config.Image}} ip={{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}'
docker exec dcmp /usr/local/bin/dcmp-healthcheck
docker exec dcmp systemctl is-active \
  greysql elasticsearch dcmp-agent acl ap1 dd1 mh1 st1 sa1 crond
docker exec dcmp /usr/dcmp/dbfiles/bin/mysql --login-path=backup_user -Nse 'SELECT 1'
docker exec dcmp curl -fsS http://127.0.0.1:9200
docker exec dcmp curl -fsS http://127.0.0.1:8808/health
docker port dcmp
```

15. 功能验证

15.1 LibreOffice 中文 PDF

只看版本号不算转换成功，必须生成非空 PDF：

```
docker exec dcmp sh -c '
  printf "DCMP 中文转换测试\n" >/tmp/dcmp-lo-test.txt
  rm -f /tmp/dcmp-lo-test.pdf
  /usr/local/bin/soffice --headless --convert-to pdf \
    --outdir /tmp /tmp/dcmp-lo-test.txt
  test -s /tmp/dcmp-lo-test.pdf
'
```

15.2 Elasticsearch 和 Agent

```
docker exec dcmp curl -fsS http://127.0.0.1:9200
docker exec dcmp curl -fsS http://127.0.0.1:9200/warning_index
docker exec dcmp curl -fsS http://127.0.0.1:8808/health
```

ES mapping 应使用内置 `cjk` analyzer，不应出现 `ik_max_word has not been configured`。

15.3 IPMI、SNMP 和备份工具

```
docker exec dcmp ipmitool -V
docker exec dcmp snmpget --version
docker exec dcmp snmpwalk --version
docker exec dcmp xtrabackup --version
```

16. 卸载和数据保留

只删除容器、保留数据库持久化目录：

```
cd <delivery-dir>
bash stop.sh
docker rm dcmp
docker network rm dcmp_net 2>/dev/null || true
```

上述操作不会删除宿主机 `/usr/dcmp/dbdata` 和 `/usr/dcmp/dbfiles`。删除这两个目录会造成数据库数据和运行文件丢失，必须在完成可恢复备份并获得明确授权后操作。

非持久化内容属于容器可写层，执行 `docker rm` 时会一并删除。

17. 运维速查

```
# 状态
bash status.sh
docker ps -a
docker exec dcmp systemctl --failed --no-pager

# 受控启停
bash start.sh
bash stop.sh

# 服务
docker exec dcmp systemctl is-active \
  greatsql elasticsearch dcmp-agent acl ap1 ddl mh1 st1 sa1 crond

# 日志
docker logs --tail 200 dcmp
docker exec dcmp journalctl -u dcmp-agent -b --no-pager -n 100
docker exec dcmp journalctl -u elasticsearch -b --no-pager -n 100
docker exec dcmp journalctl -u dcmp-container-prepare -b --no-pager -n 100

# 宿主机持久化备份
bash backup.sh

# 容器内数据库备份（重建容器前需要另行导出）
docker exec dcmp /usr/dcmp/bin/dbbackup.sh inc

# 挂载和端口
docker inspect dcmp --format '{{range .Mounts}}{{println .Source "->" .Destination}}{{end}}'
docker port dcmp
```

18. 联系方式

- 王先平
- 电话: 13641931142
- 邮箱: wangxianping@dcmp.cn

反馈时请注明：社区版或正式版、镜像 ID、CPU 架构、宿主机系统、容器 IP、发生时间、执行命令、完整错误、挂载信息及相关日志。

PDF 生成时间: 2026/08/06 17:26