Hands-on Exercise 9: Docker Volume Management & Data Persistence

Overview

This exercise demonstrates Docker volume management for persistent storage, focusing on a document analysis system with OCR and NLP components.

Exercise 9.1: Volume Management for Document Analysis

Objective

Create and configure persistent storage for document processing applications using Docker volumes.

Tasks

1. Create Volumes for Document Storage

```
# Create named volumes for different storage needs
docker volume create doc-storage
docker volume create ocr-models
docker volume create nlp-models

# List created volumes to verify
docker volume ls

# Inspect volume details
docker volume inspect doc-storage
```

2. Configure Persistence for OCR and NLP Models

```
# Create a container that loads OCR models into persistent volume
docker run -d --name ocr-setup \
  -v ocr-models:/usr/share/tesseract-ocr/4.00/tessdata \
  ubuntu:22.04 \
  /bin/bash -c "apt-get update && apt-get install -y tesseract-ocr && sleep infinity"

# Download additional OCR language models into the volume
docker exec ocr-setup bash -c "apt-get install -y tesseract-ocr-eng tesseract-ocr-urd"

# Create a container that loads NLP models into persistent volume
docker run -d --name nlp-setup \
  -v nlp-models:/models \
  python:3.9-slim \
  /bin/bash -c "pip install --no-cache-dir spacy && python -m spacy download en_core_web_sm && sleep infinity"

# Verify models are downloaded and stored in volumes
docker exec ocr-setup ls -la /usr/share/tesseract-ocr/4.00/tessdata
docker exec nlp-setup ls -la /models
```

3. Test Data Persistence Across Container Restarts

```
# Create test documents
mkdir -p test-docs
echo "This is a test document for OCR processing." > test-docs/test-doc-1.txt
echo "Another test document for NLP analysis." > test-docs/test-doc-2.txt

# Create a container using the document storage volume
docker run -d --name doc-processor \
  -v doc-storage:/data \
  -v $(pwd)/test-docs:/import \
  ubuntu:22.04 \
  /bin/bash -c "cp /import/* /data/ && sleep infinity"

# Verify documents are copied to volume
docker exec doc-processor ls -la /data

# Stop and remove container
docker stop doc-processor
docker rm doc-processor

# Create a new container using the same volume
docker run -d --name doc-processor-new \
  -v doc-storage:/data \
  ubuntu:22.04 \
  /bin/bash -c "sleep infinity"

# Verify data persists in the new container
docker exec doc-processor-new ls -la /data
docker exec doc-processor-new cat /data/test-doc-1.txt

# Test OCR model persistence
docker stop ocr-setup
docker rm ocr-setup

# Create new container using the same OCR models volume
docker run -d --name ocr-setup-new \
  -v ocr-models:/usr/share/tesseract-ocr/4.00/tessdata \
  ubuntu:22.04 \
  /bin/bash -c "apt-get update && apt-get install -y tesseract-ocr && sleep
infinity"

# Verify OCR models persist
docker exec ocr-setup-new ls -la /usr/share/tesseract-ocr/4.00/tessdata
```

Exercise 9.2: Backup and Recovery Implementation

Objective

Create backup procedures for Docker volumes and test recovery scenarios.

Tasks

1. Create Backup Scripts for Document Volumes

```
# Create a backup directory
mkdir -p volume-backups

# Create backup script
cat > backup-volumes.sh << 'EOF'
#!/bin/bash
BACKUP_DIR="./volume-backups"
DATE=$(date +%Y%m%d_%H%M%S)

# Create backup directory if it doesn't exist
mkdir -p $BACKUP_DIR

# Backup document storage volume
echo "Backing up doc-storage volume..."
docker run --rm \
  -v doc-storage:/source:ro \
  -v $(pwd)/$BACKUP_DIR:/backup \
  ubuntu:22.04 \
  tar -czf /backup/doc-storage_$DATE.tar.gz -C /source .

# Backup OCR models volume
echo "Backing up ocr-models volume..."
docker run --rm \
  -v ocr-models:/source:ro \
  -v $(pwd)/$BACKUP_DIR:/backup \
  ubuntu:22.04 \
  tar -czf /backup/ocr-models_$DATE.tar.gz -C /source .

# Backup NLP models volume
echo "Backing up nlp-models volume..."
docker run --rm \
  -v nlp-models:/source:ro \
  -v $(pwd)/$BACKUP_DIR:/backup \
  ubuntu:22.04 \
  tar -czf /backup/nlp-models_$DATE.tar.gz -C /source .

echo "Backup completed. Files saved to $BACKUP_DIR/"
ls -la $BACKUP_DIR
EOF

# Make script executable
chmod +x backup-volumes.sh

# Run backup script
./backup-volumes.sh
```

2. Test Recovery Procedures

```
# Create recovery script
cat > restore-volumes.sh << 'EOF'
#!/bin/bash
BACKUP_DIR="./volume-backups"

# Check if backup file is provided
if [ -z "$1" ]; then
  echo "Usage: $0 <backup-file> <volume-name>"
  echo "Example: $0 doc-storage_20230101_120000.tar.gz doc-storage"
  exit 1
fi

BACKUP_FILE="$1"
VOLUME_NAME="$2"

# Check if backup file exists
if [ ! -f "$BACKUP_DIR/$BACKUP_FILE" ]; then
  echo "Backup file not found: $BACKUP_DIR/$BACKUP_FILE"
  exit 1
fi

# Confirm before proceeding
echo "This will restore $BACKUP_FILE to Docker volume: $VOLUME_NAME"
read -p "Are you sure? (y/n) " -n 1 -r
echo
if [[ ! $REPLY =~ ^[Yy]$ ]]; then
  echo "Operation cancelled."
  exit 1
fi

echo "Restoring backup to $VOLUME_NAME..."
docker run --rm \
  -v $VOLUME_NAME:/destination \
  -v $(pwd)/$BACKUP_DIR:/backup \
  ubuntu:22.04 \
  bash -c "rm -rf /destination/* && tar -xzf /backup/$BACKUP_FILE -C
/destination"

echo "Restore completed."
EOF

# Make script executable
chmod +x restore-volumes.sh

# Test recovery scenario:
# 1. Create a simulated data loss
docker run --rm -v doc-storage:/data ubuntu:22.04 rm -rf /data/*

# 2. Verify data is gone
docker run --rm -v doc-storage:/data ubuntu:22.04 ls -la /data
```

```
# 3. Restore from backup
# Find the latest backup file
LATEST_BACKUP=$(ls -t volume-backups/doc-storage_*.tar.gz | head -1 | xargs
basename)
./restore-volumes.sh $LATEST_BACKUP doc-storage

# 4. Verify data is restored
docker run --rm -v doc-storage:/data ubuntu:22.04 ls -la /data
docker run --rm -v doc-storage:/data ubuntu:22.04 cat /data/test-doc-1.txt
```

3. Implement Automated Backup Solution

```
# Create a cron job file for automated backups
cat > backup-cron << 'EOF'
# Edit this file to schedule automatic backups
# Format: minute hour day month weekday command
# Example: Run backup daily at 2 AM
0 2 * * * /path/to/backup-volumes.sh >> /path/to/backup.log 2>&1
EOF

# Create a Docker container that runs scheduled backups
cat > Dockerfile.backup << 'EOF'
FROM ubuntu:22.04

RUN apt-get update && \
    apt-get install -y cron && \
    rm -rf /var/lib/apt/lists/*

WORKDIR /app

COPY backup-volumes.sh /app/
COPY backup-cron /etc/cron.d/backup-cron

RUN chmod +x /app/backup-volumes.sh && \
    chmod 0644 /etc/cron.d/backup-cron && \
    crontab /etc/cron.d/backup-cron

# Run cron in foreground
CMD ["cron", "-f"]
EOF

# Build backup container
docker build -t document-backup:latest -f Dockerfile.backup .

# Run the backup container with access to Docker socket and volumes
cat > docker-compose.backup.yml << 'EOF'
version: '3.8'

services:
  backup-service:
    image: document-backup:latest
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - doc-storage:/source/doc-storage:ro
      - ocr-models:/source/ocr-models:ro
      - nlp-models:/source/nlp-models:ro
      - ./volume-backups:/backup
    restart: unless-stopped

volumes:
  doc-storage:
    external: true
  ocr-models:
```

```
    external: true
  nlp-models:
    external: true
EOF

echo "Note: In production, you would run this with 'docker-compose -f docker-compose.backup.yml up -d'"
echo "For this demo, we won't actually start it."
```

Clean up

```
# Clean up containers
docker stop doc-processor-new ocr-setup-new nlp-setup
docker rm doc-processor-new ocr-setup-new nlp-setup

# Optional: Remove volumes if no longer needed
# docker volume rm doc-storage ocr-models nlp-models
```

Key Concepts Demonstrated

1. **Volume Creation and Management**
   - Named volumes for specific data types
   - Volume inspection and management
2. **Data Persistence**
   - Container independence from data
   - Models and documents persisting across container lifecycles
3. **Backup Strategies**
   - Volume backup using intermediary containers
   - Timestamped backups for version control
4. **Recovery Procedures**
   - Targeted volume restoration
   - Recovery verification
5. **Automation**
   - Scripted backup procedures
   - Containerized scheduled backups

Next Steps

- Implement remote backup storage (S3, NFS, etc.)

- Add backup encryption for sensitive data
- Configure backup retention policies
- Implement incremental backup strategies for large volumes