Docker CLI vs Dockerfile vs Docker Compose: Enhanced Cheat Sheet

🚀 Level 1: BASIC FUNDAMENTALS

What Are They?

Tool	Purpose	Format	Scope
Docker CLI	Interactive container management	Command line	Single container operations
Dockerfile	Build custom images	Text file	Image creation blueprint
Docker Compose	Multi-container orchestration	YAML file	Service stack management

Basic Container Operations

Operation	Docker CLI	Dockerfile	Docker Compose
Run Container	`docker run nginx`	N/A	`docker-compose up`
Stop Container	`docker stop myapp`	N/A	`docker-compose stop`
Remove Container	`docker rm myapp`	N/A	`docker-compose down`
View Logs	`docker logs myapp`	N/A	`docker-compose logs`
Exec into Container	`docker exec -it myapp bash`	N/A	`docker-compose exec service bash`

Basic Image Operations

Operation	Docker CLI	Dockerfile	Docker Compose
Use Image	`docker run ubuntu`	`FROM ubuntu`	`image: ubuntu`
Build Image	`docker build -t myapp .`	Instructions in file	`build: .`
List Images	`docker images`	N/A	`docker-compose images`
Pull Image	`docker pull nginx:alpine`	N/A	`docker-compose pull`

🎯 Level 2: ESSENTIAL CONFIGURATION

Container Naming & Basic Settings

Feature	Docker CLI	Dockerfile	Docker Compose
Container Name	`--name webapp`	N/A	`container_name: webapp`
Port Mapping	`-p 8080:80`	`EXPOSE 80`	`ports: - "8080:80"`
Environment Variables	`-e NODE_ENV=prod`	`ENV NODE_ENV=prod`	`environment: NODE_ENV=prod`
Working Directory	`--workdir /app`	`WORKDIR /app`	`working_dir: /app`
Labels	`--label key=value`	`LABEL key=value`	`labels: key: value`

Basic Volume Management

Type	Docker CLI	Dockerfile	Docker Compose
Bind Mount	`-v /host/path:/container/path`	N/A	`- /host/path:/container/path`
Named Volume	`-v myvolume:/data`	`VOLUME /data`	`- myvolume:/data`
Anonymous Volume	`-v /container/path`	`VOLUME /container/path`	`- /container/path`

Basic Examples

Simple Web Server

Docker CLI:

```bash
docker run -d --name nginx-web -p 8080:80 nginx
```

Dockerfile:

```dockerfile
FROM nginx
EXPOSE 80
```

Docker Compose:

```yaml
version: '3.8'
services:
  web:
    image: nginx
    ports:
      - "8080:80"
```

⚙️ Level 3: INTERMEDIATE FEATURES

Advanced Container Configuration

Feature	Docker CLI	Dockerfile	Docker Compose
Memory Limit	`--memory 512m`	N/A	`mem_limit: 512m`
CPU Limit	`--cpus 1.5`	N/A	`cpus: 1.5`
Restart Policy	`--restart unless-stopped`	N/A	`restart: unless-stopped`
Hostname	`--hostname myhost`	N/A	`hostname: myhost`
User	`--user 1000:1000`	`USER 1000:1000`	`user: "1000:1000"`
Init Process	`--init`	N/A	`init: true`

Networking

Feature	Docker CLI	Dockerfile	Docker Compose
Custom Network	`--network mynet`	N/A	`networks: - mynet`
Network Alias	`--network-alias web`	N/A	`aliases: - web`
DNS	`--dns 8.8.8.8`	N/A	`dns: - 8.8.8.8`
Static IP	`--ip 172.18.0.100`	N/A	`ipv4_address: 172.18.0.100`
Extra Hosts	`--add-host host1:127.0.0.1`	N/A	`extra_hosts: - "host1:127.0.0.1"`

Multi-Stage Builds (Dockerfile)

dockerfile

```dockerfile
# Build stage
FROM node:16 AS builder
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

# Production stage
FROM nginx:alpine
COPY --from=builder /app/dist /usr/share/nginx/html
EXPOSE 80
```

Environment Management

Method	Docker CLI	Dockerfile	Docker Compose
Single Variable	`-e KEY=value`	`ENV KEY=value`	`environment:` `KEY=value`
Environment File	`--env-file .env`	N/A	`env_file: .env`
Multiple Variables	`-e KEY1=val1 -e KEY2=val2`	`ENV KEY1=val1 KEY2=val2`	`environment:` (list format)
ARG Variables	`--build-arg KEY=value`	`ARG KEY`	`args: KEY: value`

Intermediate Examples

Node.js Application with Database

Docker CLI:

bash

```bash
# Create network
docker network create myapp-network

# Database
docker run -d --name postgres-db \
  --network myapp-network \
  -e POSTGRES_PASSWORD=secret \
  -v db_data:/var/lib/postgresql/data \
  postgres:13

# Application
docker run -d --name node-app \
  --network myapp-network \
  -p 3000:3000 \
  -e DATABASE_URL=postgres://postgres:secret@postgres-db:5432/myapp \
  node:16
```

Docker Compose:

yaml

```yaml
version: '3.8'
services:
  app:
    image: node:16
    ports:
      - "3000:3000"
    environment:
      DATABASE_URL: postgres://postgres:secret@db:5432/myapp
    depends_on:
      - db
    networks:
      - app-network

  db:
    image: postgres:13
    environment:
      POSTGRES_PASSWORD: secret
    volumes:
      - db_data:/var/lib/postgresql/data
    networks:
      - app-network

volumes:
  db_data:

networks:
  app-network:
```

🔧 Level 4: ADVANCED CONFIGURATION

Security & Privileges

Feature	Docker CLI	Dockerfile	Docker Compose
Read-only Filesystem	`--read-only`	N/A	`read_only: true`
Drop Capabilities	`--cap-drop ALL`	N/A	`cap_drop: - ALL`
Add Capabilities	`--cap-add NET_ADMIN`	N/A	`cap_add: - NET_ADMIN`
Security Options	`--security-opt no-new-privileges`	N/A	`security_opt: - no-new-privileges`
Privileged Mode	`--privileged`	N/A	`privileged: true`
PID Limit	`--pids-limit 100`	N/A	`pids_limit: 100`
AppArmor Profile	`--security-opt apparmor:profile`	N/A	`security_opt: - apparmor:profile`

Advanced Storage

Feature	Docker CLI	Dockerfile	Docker Compose
Tmpfs Mount	`--tmpfs /tmp:size=100m,mode=1777`	N/A	`tmpfs: - /tmp:size=100m,mode=1777`
Volume Driver	`--volume-driver local`	N/A	`driver: local`
Volume Options	`--mount type=volume,src=vol,dst=/data`	N/A	Complex volume configuration
Bind Mount Options	`--mount type=bind,src=/host,dst=/container,readonly`	N/A	`- /host:/container:ro`

Health Checks

Aspect	Docker CLI	Dockerfile	Docker Compose
Command	`--health-cmd "curl -f http://localhost"`	`HEALTHCHECK CMD curl -f http://localhost`	`test: ["CMD", "curl", "-f", "http://localhost"]`
Interval	`--health-interval 30s`	`HEALTHCHECK --interval=30s`	`interval: 30s`
Timeout	`--health-timeout 10s`	`HEALTHCHECK --timeout=10s`	`timeout: 10s`
Retries	`--health-retries 3`	`HEALTHCHECK --retries=3`	`retries: 3`
Start Period	`--health-start-period 60s`	`HEALTHCHECK --start-period=60s`	`start_period: 60s`
Disable	`--no-healthcheck`	`HEALTHCHECK NONE`	`disable: true`

Logging Configuration

Feature	Docker CLI	Dockerfile	Docker Compose
Log Driver	`--log-driver json-file`	N/A	`driver: json-file`
Log Options	`--log-opt max-size=10m`	N/A	`options: max-size: "10m"`
Log Rotation	`--log-opt max-file=3`	N/A	`max-file: "3"`
Syslog	`--log-driver syslog`	N/A	`driver: syslog`
Disable Logging	`--log-driver none`	N/A	`driver: none`

💎 Level 5: EXPERT LEVEL

Advanced Dockerfile Techniques

Multi-Architecture Builds

dockerfile

```dockerfile
# syntax=docker/dockerfile:1
FROM --platform=$BUILDPLATFORM golang:1.19 AS build
ARG TARGETPLATFORM
ARG BUILDPLATFORM
ARG TARGETOS
ARG TARGETARCH
WORKDIR /src
COPY . .
RUN CGO_ENABLED=0 GOOS=${TARGETOS} GOARCH=${TARGETARCH} go build -o app

FROM alpine:latest
RUN apk --no-cache add ca-certificates
COPY --from=build /src/app /app
ENTRYPOINT ["/app"]
```

Advanced Build Arguments

dockerfile

```dockerfile
ARG BUILD_DATE
ARG VCS_REF
ARG VERSION="latest"

LABEL org.opencontainers.image.created=$BUILD_DATE \
      org.opencontainers.image.revision=$VCS_REF \
      org.opencontainers.image.version=$VERSION
```

Docker Compose Advanced Features

Override Files & Profiles

docker-compose.yml:

yaml

```yaml
version: '3.8'
services:
  app:
    image: myapp
    profiles: ["dev", "prod"]
  debug:
    image: myapp:debug
    profiles: ["dev"]
```

docker-compose.override.yml:

yaml

```yaml
version: '3.8'
services:
  app:
    volumes:
      - .:/app
    environment:
      DEBUG: "true"
```

Advanced Networking

yaml

```yaml
version: '3.8'
networks:
  frontend:
    driver: bridge
    ipam:
      config:
        - subnet: 172.16.0.0/24
          gateway: 172.16.0.1
  backend:
    driver: overlay
    attachable: true

services:
  web:
    networks:
      frontend:
        ipv4_address: 172.16.0.100
      backend:
        aliases:
          - web-service
```

Secrets Management

yaml

```yaml
version: '3.8'
services:
  app:
    image: myapp
    secrets:
      - db_password
      - api_key

secrets:
  db_password:
    file: ./secrets/db_password.txt
  api_key:
    external: true
    external_name: myapp_api_key
```

Advanced CLI Operations

Container Inspection & Debugging

bash

```bash
# Detailed inspection
docker inspect --format='{{.NetworkSettings.IPAddress}}' container

# Resource monitoring
docker stats --format "table {{.Container}}\t{{.CPUPerc}}\t{{.MemUsage}}"

# Process monitoring
docker top container_name aux

# File system changes
docker diff container_name

# Export/Import
docker export container_name | gzip > backup.tar.gz
gunzip -c backup.tar.gz | docker import - restored_image

# Copy files
docker cp container:/path/to/file ./local/path
docker cp ./local/file container:/path/to/destination
```

Advanced Volume Operations

bash

```bash
# Volume with specific driver and options
docker volume create --driver local \
  --opt type=nfs \
  --opt o=addr=192.168.1.100,rw \
  --opt device=:/path/to/share \
  nfs_volume

# Backup volume
docker run --rm -v myvolume:/data -v $(pwd):/backup \
  alpine tar czf /backup/backup.tar.gz -C /data .

# Restore volume
docker run --rm -v myvolume:/data -v $(pwd):/backup \
  alpine tar xzf /backup/backup.tar.gz -C /data
```

🛠️ TROUBLESHOOTING & DEBUGGING

Debug Commands by Level

Basic Debugging

```bash
# View logs
docker logs container_name
docker-compose logs service_name

# Check container status
docker ps -a
docker-compose ps

# Inspect configuration
docker inspect container_name
docker-compose config
```

Intermediate Debugging

```bash
# Resource usage
docker stats
docker system df

# Network inspection
docker network ls
docker network inspect network_name

# Volume inspection
docker volume ls
docker volume inspect volume_name

# Check port bindings
docker port container_name
```

Advanced Debugging

```bash
# System events
docker events --filter container=myapp
docker-compose events

# Process inspection
docker top container_name
docker exec container_name ps aux

# File system analysis
docker diff container_name
docker exec container_name find / -name "*.log" -mtime -1

# Performance monitoring
docker exec container_name top
docker exec container_name iostat -x 1
```

📚 BEST PRACTICES BY LEVEL

Basic Best Practices

- Always tag your images with specific versions
- Use `.dockerignore` files to reduce build context
- Clean up unused containers and images regularly
- Use meaningful names for containers and services
- Don't run containers as root user

Intermediate Best Practices

- Use multi-stage builds to reduce image size
- Implement health checks for all services
- Set appropriate resource limits
- Use specific base image versions, avoid `latest`
- Implement proper error handling in scripts

Advanced Best Practices

- Use secrets management for sensitive data
- Implement proper logging strategies with log rotation

- Use read-only filesystems when possible
- Apply security hardening (drop capabilities, use non-root users)
- Implement monitoring and alerting

Expert Best Practices

- Use distroless or minimal base images (Alpine, scratch)
- Implement container scanning in CI/CD pipelines
- Use init systems for proper signal handling
- Implement proper backup and disaster recovery
- Use orchestration platforms (Kubernetes, Docker Swarm) for production

🚀 QUICK REFERENCE COMMANDS

Docker CLI Essentials

bash

```bash
# Container lifecycle
docker run -d --name app nginx    # Create and start
docker start/stop/restart app     # Control container
docker rm app                     # Remove container
docker exec -it app bash          # Access container

# Images
docker build -t myapp .           # Build image
docker images                     # List images
docker rmi myapp                  # Remove image
docker pull nginx:alpine          # Pull image

# System maintenance
docker system prune -a            # Clean everything
docker container prune            # Remove stopped containers
docker image prune                # Remove unused images
docker volume prune               # Remove unused volumes
```

Docker Compose Essentials

bash

```bash
# Service management
docker-compose up -d            # Start services
docker-compose down             # Stop and remove
docker-compose restart          # Restart services
docker-compose logs -f          # Follow logs

# Scaling and building
docker-compose up --scale web=3 # Scale service
docker-compose build            # Build services
docker-compose pull             # Pull images

# Configuration
docker-compose config           # Validate and view config
docker-compose ps              # List services
```

Dockerfile Essentials

dockerfile

```dockerfile
# Base and setup
FROM node:16-alpine             # Base image
WORKDIR /app                    # Working directory
COPY package*.json ./           # Copy dependency files
RUN npm ci --only=production    # Install dependencies

# Application setup
COPY . .                        # Copy source code
EXPOSE 3000                     # Expose port
USER node                       # Run as non-root
HEALTHCHECK --interval=30s \
  CMD curl -f http://localhost:3000/health || exit 1
CMD ["npm", "start"]            # Default command
```

🔍 COMMON ISSUES & SOLUTIONS

Permission Issues

```bash
# Fix permission issues with bind mounts
docker run -u $(id -u):$(id -g) -v $(pwd):/app myapp

# Change ownership inside container
docker exec container chown -R user:group /path
```

Network Connectivity

```bash
# Debug network connectivity
docker exec container ping other-container
docker exec container nslookup service-name
docker network inspect network_name
```

Resource Constraints

```bash
# Check resource usage
docker stats
docker system df

# Set memory and CPU limits
docker run --memory=512m --cpus=1.0 myapp
```

This enhanced cheat sheet provides more comprehensive coverage, fixes inconsistencies, and includes additional practical examples and troubleshooting guidance.