Docker Container Management - Essential Commands Guide

Quick Reference Card

🚀 Most Common Commands You'll Use Daily

```bash
# Start everything
docker compose up -d

# Stop everything
docker compose down

# Restart everything
docker compose restart

# See what's running
docker compose ps

# View logs
docker compose logs -f
```

📍 First Things First: Where to Run Commands

IMPORTANT: Always run these commands from your project directory:

```bash
# Navigate to your project directory first
cd ~/my-working-prototype-dashbaord

# Or if it's in a different location
cd /path/to/your/project
```

🟢 Starting Containers (Bringing Them Up)

1. Start All Containers (Most Common)

```bash
# Start all services defined in docker-compose.yml
docker compose up -d
```

What this does:

- `up` = Create and start containers
- `-d` = Run in detached mode (background)
- Starts nginx, api, mongodb, redis, etc. all at once

2. Start Specific Container Only

```bash
# Start just nginx
docker compose up -d nginx

# Start just the API
docker compose up -d api

# Start multiple specific services
docker compose up -d nginx api
```

3. Start and Watch Logs (Useful for Debugging)

```bash
# Start and see logs in real-time (not detached)
docker compose up

# Press Ctrl+C to stop when running like this
```

4. Force Rebuild Before Starting

```bash
# Rebuild images before starting (useful after code changes)
docker compose up -d --build

# Force recreate containers even if config hasn't changed
docker compose up -d --force-recreate
```

5. Start with Specific Configuration File

```bash
# If you have multiple compose files
docker compose -f docker-compose.yml up -d
docker compose -f docker-compose.prod.yml up -d
```

🔴 Stopping Containers (Taking Them Down)

1. Stop All Containers (Keeps Data)

```bash
# Stop all running containers
docker compose stop
```

What this does:

- Stops containers but doesn't remove them
- Data in volumes is preserved
- Can restart quickly with `docker compose start`

2. Stop and Remove Containers (Clean Stop)

```bash
# Stop and remove containers, networks
docker compose down
```

What this does:

- Stops all containers
- Removes containers
- Removes networks
- **Keeps volumes (your data is safe)**

3. Stop and Remove Everything (Including Data)

```bash
# WARNING: This deletes your data!
docker compose down -v
```

What this does:

- Stops all containers
- Removes containers
- Removes networks
- **REMOVES VOLUMES (deletes all data!)**

4. Stop Specific Container

```bash
# Stop just nginx
docker compose stop nginx

# Stop multiple specific services
docker compose stop nginx api
```

5. Emergency Force Stop

```bash
# Force stop if containers are stuck
docker compose kill

# Force remove if normal removal fails
docker compose rm -f
```

🔄 Restarting Containers

1. Restart All Containers

```bash
# Restart all services
docker compose restart
```

2. Restart Specific Container

```bash
# Restart just nginx
docker compose restart nginx

# Restart with a delay
docker compose restart -t 30 nginx  # 30 second timeout
```

3. Stop Then Start (Clean Restart)

```bash
# Sometimes more effective than restart
docker compose stop
docker compose start
```

📊 Checking Container Status

1. See What's Running

```bash
# Show status of all services
docker compose ps

# Show all containers (including stopped)
docker compose ps -a
```

2. Detailed Container Information

```bash
# See all Docker containers on system
docker ps

# See resource usage (CPU, Memory)
docker stats

# See container details
docker inspect cyber-trust-nginx
```

3. Check Container Health

```bash
# See health status
docker compose ps --format json | jq '.[].Health'

# Simple health check
curl http://localhost/health
```

📜 Viewing Logs

1. View All Logs

```bash
```

```bash
# Show logs from all containers
docker compose logs

# Follow logs in real-time (like tail -f)
docker compose logs -f

# Show last 100 lines
docker compose logs --tail=100
```

2. View Specific Container Logs

```bash
# View nginx logs
docker compose logs nginx

# Follow nginx logs in real-time
docker compose logs -f nginx

# View last 50 lines of API logs
docker compose logs --tail=50 api
```

3. View Logs with Timestamps

```bash
# Show logs with timestamps
docker compose logs -t

# Follow logs with timestamps
docker compose logs -tf
```

🛠️ Maintenance Commands

1. Update Containers

```bash
# Pull latest images
docker compose pull

# Pull and restart with new images
docker compose pull && docker compose up -d
```

2. Clean Up Resources

```bash
# Remove stopped containers
docker compose rm

# Remove unused images, networks, volumes
docker system prune

# Remove everything unused (aggressive cleanup)
docker system prune -a --volumes
```

3. Execute Commands in Container

```bash
# Open shell in nginx container
docker compose exec nginx sh

# Run command in api container
docker compose exec api npm list

# Open shell in specific container
docker exec -it cyber-trust-nginx sh
```

🚨 Troubleshooting Commands

1. When Containers Won't Start

```bash
# Check what's wrong
docker compose logs

# Try removing and recreating
docker compose down
docker compose up -d

# Force recreate
docker compose up -d --force-recreate
```

2. When Ports Are Blocked

```bash
```

```bash
# Check what's using port 80
sudo lsof -i :80

# Check Docker port bindings
docker ps --format "table {{.Names}}\t{{.Ports}}"
```

3. When Out of Space

```bash
# Check Docker space usage
docker system df

# Clean up everything unused
docker system prune -a
```

📝 Common Workflows

Morning Startup Routine

```bash
# 1. Navigate to project
cd ~/my-working-prototype-dashbaord

# 2. Pull latest changes (if working with team)
git pull

# 3. Start all services
docker compose up -d

# 4. Check everything is running
docker compose ps

# 5. Check logs for errors
docker compose logs --tail=50
```

End of Day Shutdown

```bash
```

```
# 1. Navigate to project
cd ~/my-working-prototype-dashbaord

# 2. Stop all containers
docker compose stop
# OR remove them completely
docker compose down
```

After Making Code Changes

bash

```
# 1. If you changed the frontend
npm run build
docker compose restart nginx

# 2. If you changed the API
docker compose restart api

# 3. If you changed docker-compose.yml
docker compose down
docker compose up -d

# 4. If you changed Dockerfile
docker compose build
docker compose up -d
```

Debugging When Something's Wrong

bash

```
# 1. Check what's running
docker compose ps

# 2. Check logs for errors
docker compose logs --tail=100

# 3. Try restarting
docker compose restart

# 4. If still broken, recreate
docker compose down
docker compose up -d

# 5. Check specific service logs
docker compose logs -f nginx
```

⚡ Quick Copy-Paste Commands

Start Everything

```
bash

cd ~/my-working-prototype-dashbaord && docker compose up -d
```

Stop Everything

```
bash

cd ~/my-working-prototype-dashbaord && docker compose down
```

Restart Everything

```
bash

cd ~/my-working-prototype-dashbaord && docker compose restart
```

View Status and Logs

```
bash

cd ~/my-working-prototype-dashbaord && docker compose ps && docker compose logs --tail=50
```

Emergency Reset

```
bash
```

```
cd ~/my-working-prototype-dashbaord && docker compose down && docker compose up -d --force-recreate
```

📋 Command Comparison Table

Action	Soft Method	Hard Method	Nuclear Method
Start	`docker compose start`	`docker compose up -d`	`docker compose up -d --force-recreate`
Stop	`docker compose stop`	`docker compose down`	`docker compose down -v`
Restart	`docker compose restart`	`docker compose stop && start`	`docker compose down && up -d`
Update	`docker compose pull`	`docker compose pull && up -d`	`docker compose build --no-cache && up -d`
Clean	`docker system prune`	`docker system prune -a`	`docker system prune -a --volumes`

🎯 Best Practices

1. Always Use `-d` Flag

```bash
# Good - runs in background
docker compose up -d

# Avoid - ties up your terminal
docker compose up
```

2. Check Before Stopping

```bash
# Always check what's running first
docker compose ps

# Then stop
docker compose down
```

3. Use Specific Names When Possible

```bash
```

```
# More efficient - only affects nginx
docker compose restart nginx

# Less efficient - restarts everything
docker compose restart
```

4. Keep Your Data Safe

bash

```
# Safe - preserves volumes
docker compose down

# DANGEROUS - deletes data
docker compose down -v
```

5. Monitor Logs After Changes

bash

```
# After any restart, check logs
docker compose up -d
docker compose logs -f

# Press Ctrl+C to stop watching logs
```

🆘 Emergency Commands

If Everything is Broken

bash

```bash
#!/bin/bash
# Emergency reset script

echo "Emergency reset starting..."

# Stop everything
docker compose kill 2>/dev/null || true
docker compose rm -f 2>/dev/null || true

# Clean up
docker system prune -f

# Restart Docker daemon
sudo systemctl restart docker

# Wait for Docker to be ready
sleep 5

# Start fresh
docker compose up -d --force-recreate

echo "Reset complete. Checking status..."
docker compose ps
```

If You Can't Access Your Application

```bash
# 1. Check if containers are running
docker compose ps

# 2. Check if ports are accessible
curl http://localhost/health

# 3. Check nginx logs
docker compose logs nginx --tail=100

# 4. Restart nginx
docker compose restart nginx

# 5. If still broken, check firewall
sudo ufw status
sudo ufw allow 80/tcp
```

📖 Understanding What Happens

When You Run `docker compose up -d`:

1. Docker reads your `docker-compose.yml` file
2. Creates networks if they don't exist
3. Creates/starts containers in dependency order
4. Binds ports (like 80:80)
5. Mounts volumes for persistent data
6. Returns control to your terminal

When You Run `docker compose down`:

1. Stops all running containers
2. Removes containers
3. Removes networks
4. Keeps volumes (your data) unless you use `-v`

When You Run `docker compose restart`:

1. Sends stop signal to containers
2. Waits for graceful shutdown
3. Starts containers again
4. Preserves all data and configurations

💡 Pro Tips

1. **Create Aliases** for common commands:

```bash
# Add to ~/.bashrc or ~/.zshrc
alias dcup='docker compose up -d'
alias dcdown='docker compose down'
alias dcps='docker compose ps'
alias dclogs='docker compose logs -f'
```

2. **Use Watch** for monitoring:

```bash
# Auto-refresh container status every 2 seconds
watch -n 2 docker compose ps
```

3. **Quick Health Check**:

```bash
# Create a health check function
health() {
    echo "Checking services..."
    docker compose ps
    echo -e "\nChecking web..."
    curl -s http://localhost/health || echo "Web is down"
    echo -e "\nChecking API..."
    curl -s http://localhost/api/health || echo "API is down"
}
```

Remember: The most important commands are just:

- `docker compose up -d` (start)
- `docker compose down` (stop)
- `docker compose ps` (check)
- `docker compose logs -f` (debug)

Everything else is for specific situations!