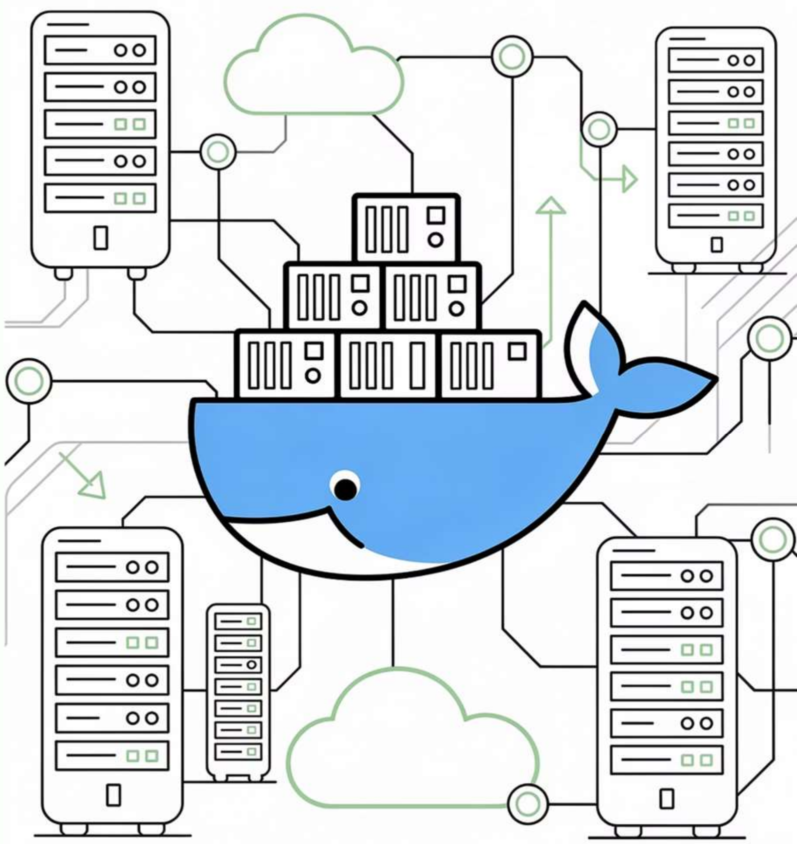
Docker Container Lifecycle & Debugging

Love this topic – this is where learners move from "I can run a container" → "I can operate containers like a pro" 🐳 🔥



Module

Docker Container Lifecycle & Debugging

MODULE TITLE

Docker Container Lifecycle Management

Running, Monitoring & Debugging Containers

Audience: DevOps / Cloud / Platform Engineers

Learning Objectives

By the end of this session, learners will be able to:

01

Understand the complete lifecycle of a Docker container

02

Control container states (start, stop, restart, pause)

03

Inspect container metadata and configuration

04

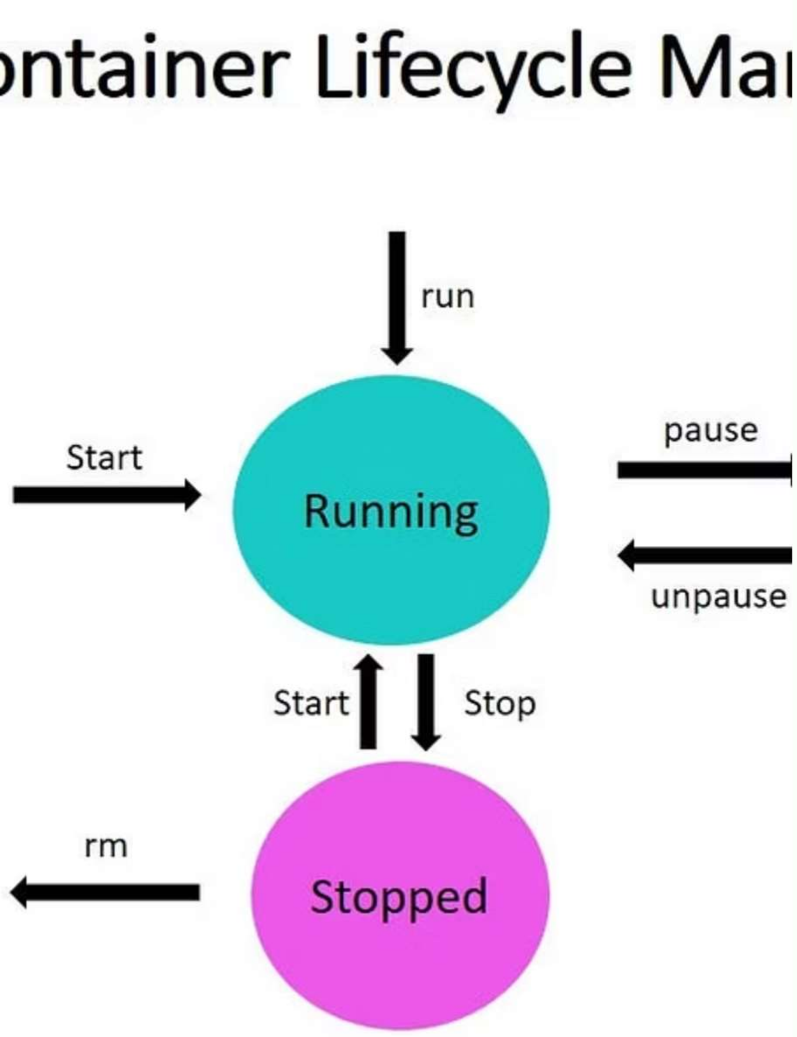
View and analyze container logs

05

Execute commands inside running containers

06

Troubleshoot failed or misbehaving containers



What is a Container Lifecycle?

A Docker container goes through multiple states from creation to removal.

Think of it like a virtual machine but faster and lighter.

Lifecycle Flow:

Create → Start → Run → Pause/Stop → Restart → Remove

Container States Explained

State	Meaning
Created	Container is created but not running
Running	Main process is active
Paused	Processes are frozen
Stopped	Process exited
Restarting	Docker attempting restart
Dead	Failed to start/stop properly
Removed	Container deleted

Command to check state:

```
docker ps -a
```

Creating and Starting Containers

Create only (not running)

```
docker create nginx
```

Create + Start (most common)

```
docker run -d --name web nginx
```

Start a stopped container

```
docker start web
```

Stopping Containers

Graceful Stop (SIGTERM → SIGKILL)

```
docker stop web
```

Force Stop Immediately

```
docker kill web
```

Stop All Running Containers

```
docker stop $(docker ps -q)
```

Key Difference

`stop` gives the container time to shut down gracefully, while `kill` terminates immediately.

Restarting Containers

```
docker restart web
```

Auto-restart Policies

Policy	Behavior
no	Default
always	Always restart
on-failure	Restart if exit code ≠ 0
unless-stopped	Restart unless manually stopped

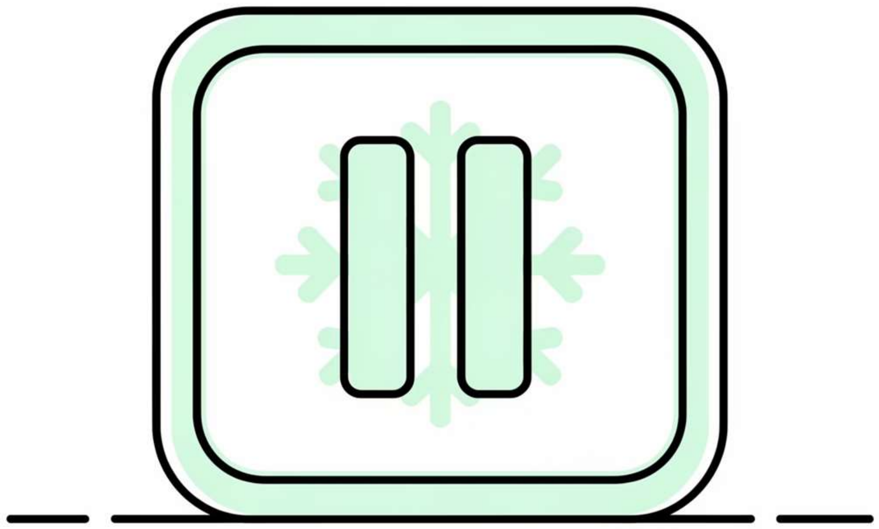
```
docker run -d --restart=always nginx
```

Pausing and Unpausing Containers

Pausing freezes processes without stopping container.

```
docker pause web
docker unpause web
```

Use case: temporarily reduce CPU usage.



Removing Containers

Remove stopped container

```
docker rm web
```

Force remove running container

```
docker rm -f web
```

Remove all stopped containers

```
docker container prune
```

Part 2

Container Inspection & Debugging

Inspecting Container Details

Shows full JSON configuration of container.

```
docker inspect web
```

Useful details:

- IP address
- Mounted volumes
- Environment variables
- Network settings
- Restart policy

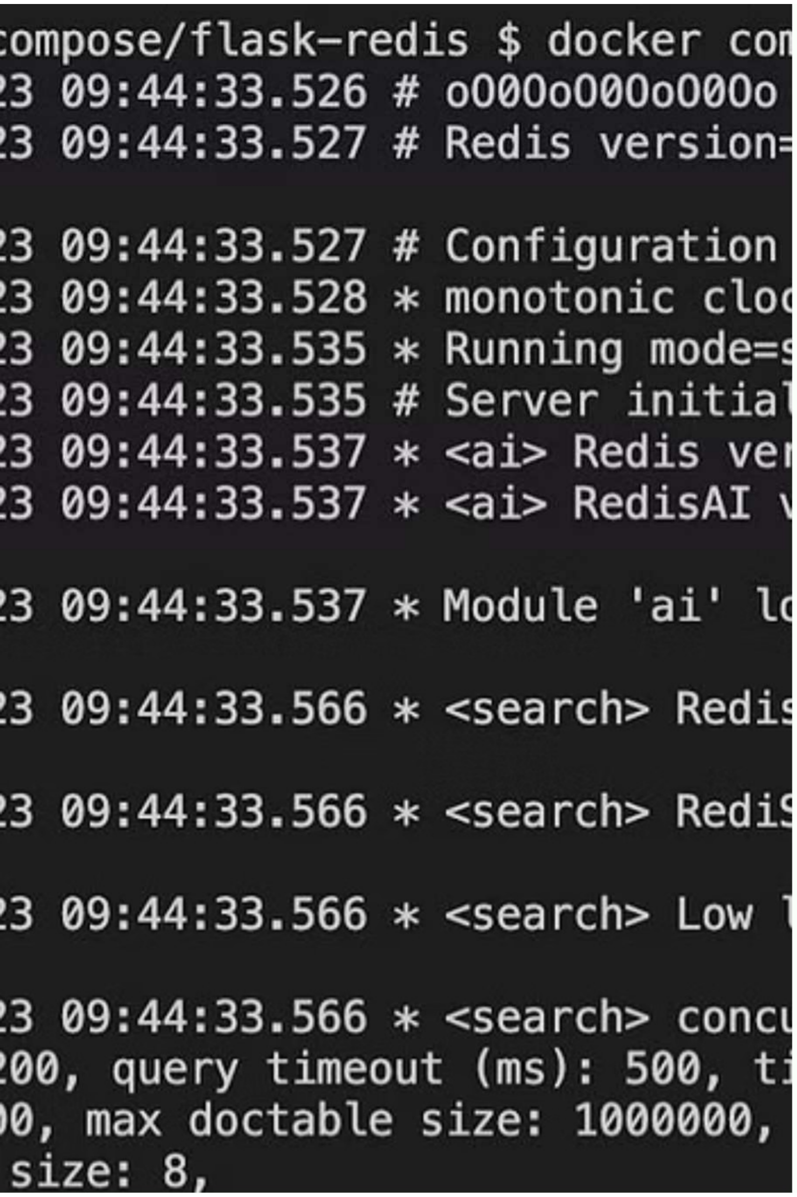
Filtering Inspect Output

Get container status

```
docker inspect -f '{{.State.Status}}' web
```

Get IP address

```
docker inspect -f '{{.NetworkSettings.IPAddress}}' web
```



```
compose/flask-redis $ docker com
23 09:44:33.526 # oO0OoO0OoO0Oo
23 09:44:33.527 # Redis version=

23 09:44:33.527 # Configuration
23 09:44:33.528 * monotonic cloc
23 09:44:33.535 * Running mode=s
23 09:44:33.535 # Server initial
23 09:44:33.537 * <ai> Redis ver
23 09:44:33.537 * <ai> RedisAI v

23 09:44:33.537 * Module 'ai' lo

23 09:44:33.566 * <search> Redis

23 09:44:33.566 * <search> RediS

23 09:44:33.566 * <search> Low l

23 09:44:33.566 * <search> concu
200, query timeout (ms): 500, ti
00, max doctable size: 1000000,
 size: 8,
```

Viewing Container Logs

Logs show STDOUT and STDERR of container process.

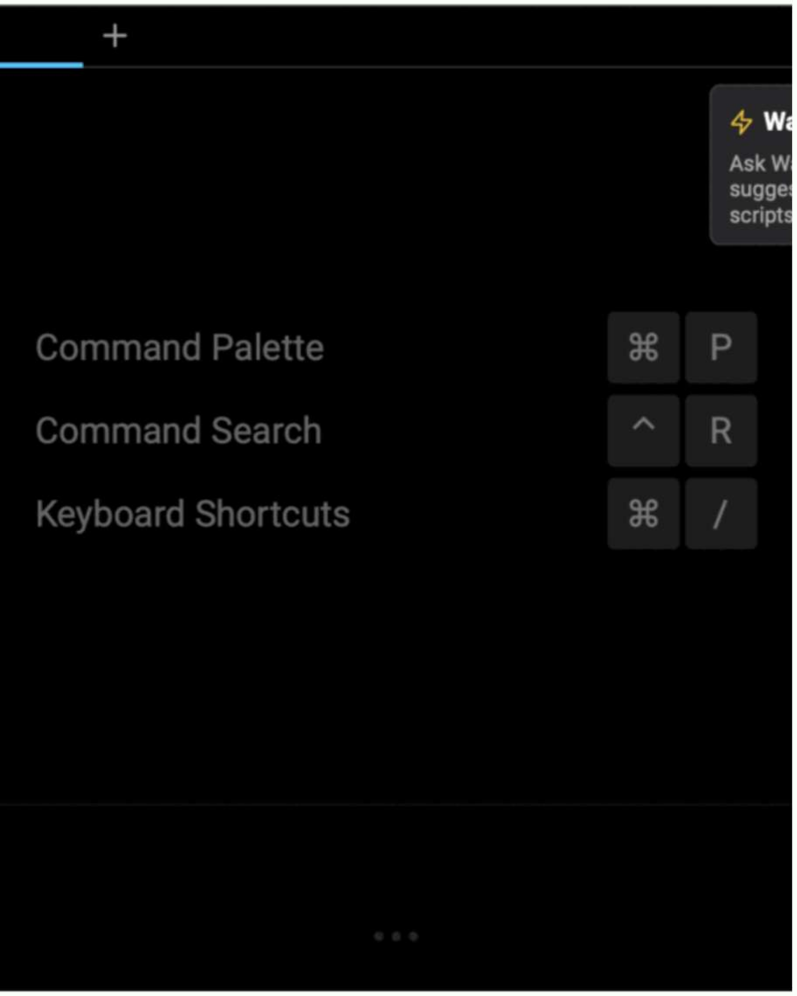
```
docker logs web
```

Live logs:

```
docker logs -f web
```

Last 50 lines:

```
docker logs --tail 50 web
```


Command Palette ⌘ P
Command Search ^ R
Keyboard Shortcuts ⌘ /

Executing Commands Inside Containers

Run commands inside running container.

```
docker exec -it web /bin/bash
```

Other examples:

```
docker exec web ls /usr/share/nginx/html
docker exec -it web sh
```

Difference: run vs exec

`docker run`

- Creates new container
- Runs main process
- New environment

`docker exec`

- Uses existing container
- Runs additional command
- Same container context

Monitoring Container Resource Usage

```
docker stats
```

Shows:

CPU %

Memory usage

Network I/O

Block I/O

Troubleshooting Failed Containers

Check container status

```
docker ps -a
```

Inspect exit code

```
docker inspect web --format='{{.State.ExitCode}}'
```

Common Causes

- App crash
- Missing environment variables
- Port already in use
- Volume permission issues