



Synapse-iDOCK

CONTAINER ORCHESTRATION

User & Administrator Guide

Sleek, agent-first management for Docker fleets — secure by design.



Contents

1. Introduction	3
Key features	3
2. Architecture Overview	3
3. Getting Started (Administrator)	3
Requirements	3
Deploy with Docker Compose	3
First-run admin setup	3
Configuration	4
4. Deploying Agents	5
One-command install	5
Other options	5
Verifying the connection	6
Removing an agent	6
5. Using the Dashboard (User)	6
Fleet stats	6
Host groups	6
Status & quick actions	7
Container Insight	7
6. Managing Hosts (Manual TCP / TLS / SSH)	8
7. Webhooks & Alerting	9
8. API Tokens & REST API	9
9. Security Best Practices	10
10. Troubleshooting	10
11. Frequently Asked Questions	11



1. Introduction

Synapse-iDOCK is a self-hosted web application for monitoring and controlling Docker containers across many networked hosts from a single, sleek dashboard. It belongs to the Synapse ecosystem and shares its dark, cyan-accented design language.

Its guiding principle is **agent-first** delivery: because Docker daemons normally only listen on a local socket, iDOCK uses a lightweight agent that runs on each host and **pushes** data outbound — no Docker port is ever exposed on the network.

Key features

- **One-command agent deploy** — install on a host with a single, pre-authenticated command.
- **Grouped dashboard** — containers are grouped by host into expandable cards with Online/Offline status.
- **Container Insight** — live CPU / RAM / Network / Disk charts, searchable logs, and full metadata.
- **Quick actions** — Start, Stop, and Update (image pull) inline from the dashboard.
- **Webhooks** — push JSON alerts to Slack, IronNode SOAR, and more on container events.
- **Secure REST API** — token-authenticated `/api/v1` for automation.

2. Architecture Overview

A single container runs the whole service: a FastAPI backend, a SQLite database on a persistent volume, and the compiled React dashboard. Agents on remote hosts connect inbound-free over HTTPS.

```
Docker host Synapse-iDOCK server
+-----+ outbound HTTPS +-----+
| idock-agent | --- POST report ---> | dashboard / metrics |
| reads sock | <-- GET commands --- | webhooks / REST API |
| runs cmds  | --- POST result ---> | SQLite (volume)  |
+-----+ +-----+
```

Persistence lives in one Docker volume (`/data`): the database, the generated session secret, uploaded TLS/SSH material, and the cached agent image.

3. Getting Started (Administrator)

Requirements

- A host with Docker and Docker Compose.
- A published port for the dashboard (default 8800).
- For local management, the host's Docker socket (mounted by the compose file).

Deploy with Docker Compose

```
git clone <repo> synapse-idock && cd synapse-idock
cp .env.example .env # optional – sane defaults apply
docker compose up -d --build
```

First-run admin setup



Open `http://<host>:8800`. On first run you are redirected to **Initial Admin Setup** to create the primary administrator account, then sign in.

Configuration

All settings are environment variables (prefix `IDOCK_`). The most useful ones:

Variable	Default	Purpose
<code>IDOCK_PORT</code>	8800	Published dashboard port.
<code>IDOCK_PUBLIC_URL</code>	(auto)	External URL baked into the agent command; blank auto-detects the LAN IP.
<code>IDOCK_SESSION_HTTPS_ONLY</code>	false	Mark the session cookie Secure (set behind HTTPS).
<code>IDOCK_AGENT_IMAGE</code>	(empty)	Registry ref for the agent image; blank = iDOCK builds & serves it.
<code>IDOCK_DISCOVERY_INTERVAL</code>	30	Seconds between status/metric sweeps.
<code>IDOCK_METRIC_RETENTION</code>	720	Metric samples kept per container.



4. Deploying Agents

The agent is the recommended way to bring a host into iDOCK. It reads the local Docker socket and reports outbound only — nothing inbound is opened.

One-command install

Open **Deploy Agent**, click **Generate token**, and copy the command. The server address (auto-detected to the host's reachable IP) and token are already filled in, and by default iDOCK builds and serves the agent image itself — no registry required:

```
curl -fsSL https://<idock-host>/agent/image.tar.gz | docker load && \
docker run -d --name idock-agent --restart unless-stopped \
-e IDOCK_URL="https://<idock-host>" -e IDOCK_TOKEN="idk_..." \
-e IDOCK_AGENT_NAME="$(hostname)" -e IDOCK_AGENT_ID_FILE=/state/agent_id \
-v /var/run/docker.sock:/var/run/docker.sock -v idock_agent_state:/state \
synapse-idock-agent:1.0.0
```

→ The socket is mounted read-write so Start / Stop / Update work. Append :ro for monitoring only.

Synapse-iDOCK
CONTAINER ORCHESTRATION

Dashboard
Deploy Agent
Remove Agent
Hosts
Webhooks
API Tokens

Signed in as admin
Sign out

Deploy Agent

Install the agent on a host with **one command**. It reports over outbound HTTPS — no Docker port exposed — and appears under **Hosts** within ~20 seconds. No approval needed.

1 - Generate an enrollment token
Shown once, and baked into the command below so the agent is pre-authenticated. Revoke it under API Tokens to disconnect a host.

Regenerate

idk_LvrB9Qu3gZRES4wS3mmGSnxX70cXQzwdXWf3WnvzKzQ

SERVER ADDRESS CLIENTS CONNECT TO

http://192.168.0.50:8259

Auto-detected from this server (http://192.168.0.50:8259) and baked into the command below — no need to edit unless clients reach this host at a different address. Set **IDOCK_PUBLIC_URL** to make it permanent behind a domain or reverse proxy.

2 - Run this one command on the host
Loads the agent image straight from this server and runs it, pre-authenticated — no registry, no extra setup.

Docker (Linux / macOS)
Requires only Docker on the host.

Copy

```
curl -fsSL http://192.168.0.50:8259/agent/image.tar.gz | docker load && \
docker run -d --name idock-agent --restart unless-stopped \
-e IDOCK_URL="http://192.168.0.50:8259" -e IDOCK_TOKEN="idk_LvrB9Qu3gZRES4wS3mmGSnxX70cXQzwdXWf3WnvzKzQ" -e IDOCK_AGENT_NAME="$(hostname)" \
-e IDOCK_AGENT_ID_FILE=/state/agent_id \
-v /var/run/docker.sock:/var/run/docker.sock \
-v idock_agent_state:/state \
synapse-idock-agent:1.0.0
```

► Other hosts & options

The command mounts the Docker socket read-write so the dashboard's Start / Stop / Update buttons work. For monitoring only, append :ro to the socket mount. The idock_agent_state volume keeps the host's identity stable across restarts.

Note: the agent image builds on this server the first time it's requested (a few seconds), then is cached for every host after.

Deploy Agent — generate a token and copy the pre-filled one-line install command.

Other options

- **Docker Compose:** IDOCK_URL=... IDOCK_TOKEN=... docker compose up -d using agent/docker-compose.yml.
- **systemd:** a unit file that runs the Python agent (see agent/README.md).
- **Registry:** publish the image and set IDOCK_AGENT_IMAGE for a plain docker run <ref>.



Verifying the connection

Within ~20 seconds the host appears under **Hosts** → **Connected Agents** and its containers stream onto the dashboard — no approval step, because the token is the trust boundary.

Removing an agent

Use the **Remove Agent** item in the side menu. It removes a **single** agent without revoking the shared token, so any other agents using that token keep working.

- **1. Open Remove Agent** and select the agent from the list.
- **2. Remove this agent** — blocks that agent id (it can no longer reconnect) and clears its host and containers from the dashboard.
- **3. Uninstall on the host** — run the command shown for the host's method (Docker / Compose / systemd / Python).

→ *Because the agent id is blocked, order no longer matters — even a still-running agent is refused. To bring the host back later, reinstall with a wiped state volume so it gets a fresh identity.*

Agent hosts therefore have no inline Remove button on the Hosts page — they show *Managed via Remove Agent*. Manually-added TCP/TLS/SSH hosts still use **Hosts** → **Remove**.

Remove an Agent

AGENT TO REMOVE

prod-db-02 · 2 containers

i Disconnects **only this agent** and blocks it from reconnecting. Other agents sharing the same token keep working — the token is **not** revoked.

1 **Disconnect & clear from iDOCK**
Blocks this agent id and removes its host and containers from the dashboard.

Remove this agent

2 **Uninstall on the host**

Docker Compose systemd Python

`docker rm -f idock-agent && docker volume rm idock_agent_state` **Copy**

Remove Agent — select one agent, remove it (blocks that id), and copy the host uninstall command.

5. Using the Dashboard (User)

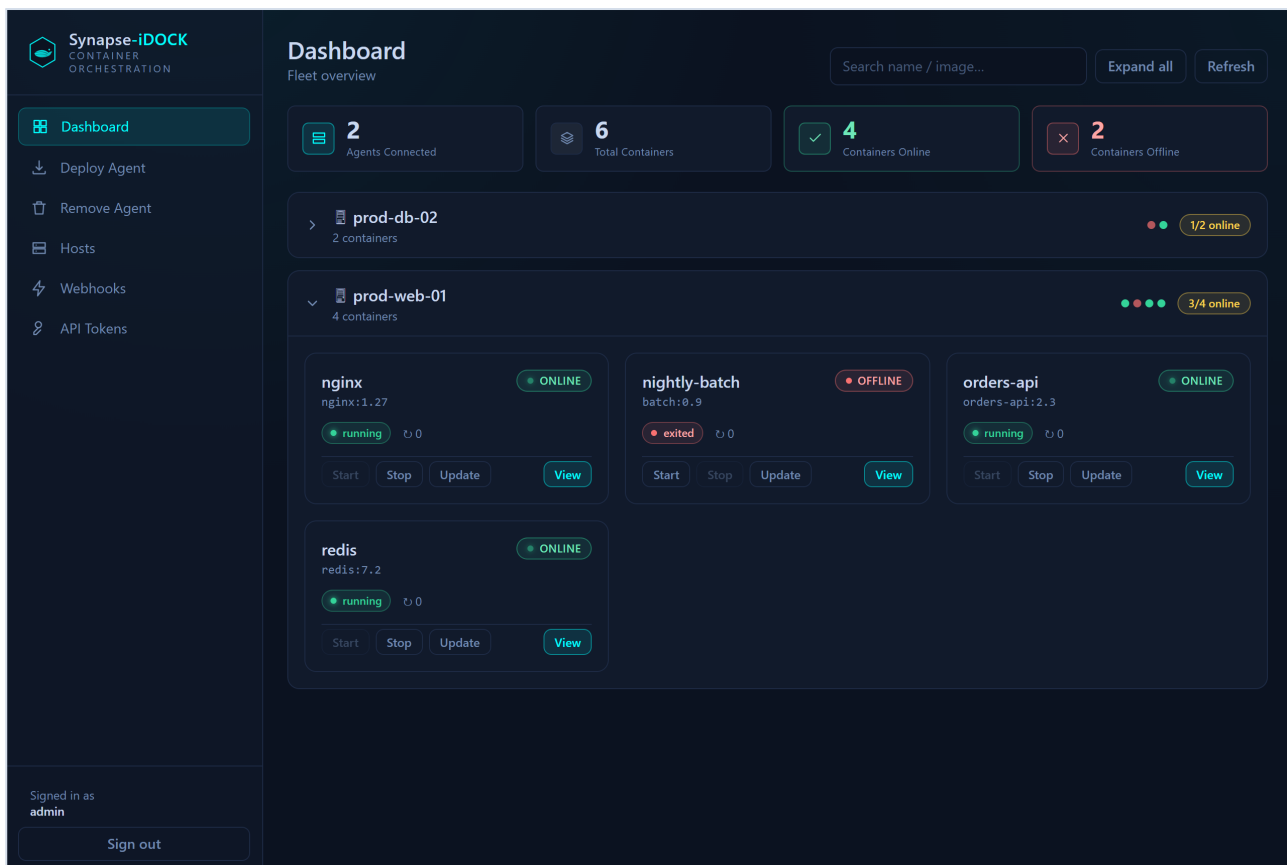
Fleet stats

Four stat cards across the top give an at-a-glance fleet summary: **Agents Connected** (agents currently reporting), **Total Containers**, **Containers Online** and **Containers Offline**. They refresh automatically with the dashboard.

Host groups



Containers are grouped by host. Each host is a collapsible card showing its name, container count, a status dot-strip, and an **online / total** summary. Click a host card to expand it and reveal the individual container cards within.



The dashboard: fleet stat cards on top, with a host group expanded to reveal its containers.

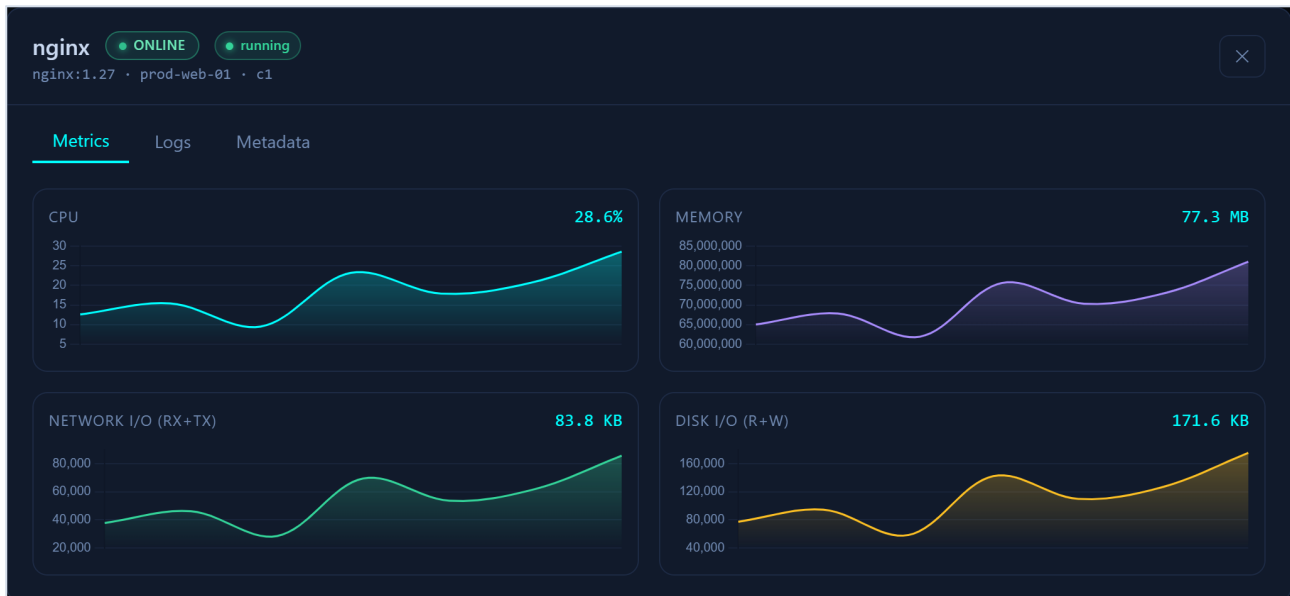
Status & quick actions

Every container shows an **Online / Offline** pill (green when running and its host is reachable). Inline buttons run **Start**, **Stop**, and **Update** (pull the latest image). Use the search box to filter by name or image; matching groups expand automatically.

Container Insight

Click **View** on any container to open the Insight modal:

- **Metrics** — time-series charts for CPU, RAM, Network I/O and Disk I/O.
- **Logs** — the last 100 lines, searchable and downloadable.
- **Metadata** — health, restart count, uptime, port mappings and environment variables.



Container Insight — live CPU, memory, network and disk charts for a container.

6. Managing Hosts (Manual TCP / TLS / SSH)

For daemons deliberately exposed over the network, add them under **Hosts** → **Add Host** and choose a security posture:

- **None** — plain `tcp://host:2375` or a local socket; trusted networks only.
- **TLS** — upload CA, client certificate and key for a mutually-authenticated daemon on 2376.
- **SSH** — `ssh://user@host` with a username and private key.

→ *Prefer agents. Exposing the Docker API over TCP is rarely necessary and widens your attack surface.*

Agent hosts also appear on this page under **Connected Agents** and in the hosts table, but are read-only here (labelled *Managed via Remove Agent*) — use the **Remove Agent** menu to remove one. The inline **Test** / **Remove** actions apply to manual hosts only.



Synapse-iDOCK
CONTAINER ORCHESTRATION

Dashboard

Deploy Agent

Remove Agent

Hosts

Webhooks

API Tokens

Signed in as
admin

Sign out

Hosts

Manage Docker endpoints and their credentials.

+ Add Host

Connected Agents (2)
Push-based hosts reporting over outbound HTTPS — no exposed Docker port required.

AGENT HOST	OS	VERSION	LAST REPORT
prod-db-02	Linux 6.1.0-x86_64	1.0.0	11/07/2026, 11:36:47
prod-web-01	Linux 6.1.0-x86_64	1.0.0	11/07/2026, 11:36:47

HOST	ENDPOINT	SECURITY	STATUS	CONTAINERS	ACTIONS
legacy-tls-01 manual · approved	tcp://10.20.0.5:2376	TLS	● unreachable	0	Test Remove
prod-db-02 agent · approved	agent://agent-db	AGENT	● reachable (26.1.4)	2	Managed via Remove Agent
prod-web-01 agent · approved	agent://agent-web	AGENT	● reachable (26.1.4)	4	Managed via Remove Agent

The Hosts page: Connected Agents plus the hosts table (agent rows are managed via Remove Agent).

7. Webhooks & Alerting

Under **Webhooks**, define endpoints that receive a JSON payload when a trigger fires during a sweep. Triggers: **Container Down**, **Container Recovered**, **High CPU** (per-webhook threshold) and **Restart Loop**. Add custom headers for authenticated targets such as Slack or IronNode SOAR, and use **Send test** to validate connectivity.

```
{ "source": "synapse-idock", "trigger": "container_down",  
  "host": "prod-web-01", "container": "billing-api", "state": "exited" }
```

8. API Tokens & REST API

Generate a token under **API Tokens** (shown once). Authenticate machine calls with `Authorization: Bearer idk_...`. The same tokens enrol agents.

Endpoint		Description
GET /api/v1/hosts	—	List approved hosts.
GET /api/v1/containers	—	List approved containers.
POST /api/v1/containers/{id}/start	—	Start a container.
POST /api/v1/containers/{id}/stop	—	Stop a container.
POST /api/v1/containers/{id}/pull	—	Pull the latest image.
GET /api/v1/containers/{id}/logs	—	Fetch recent logs.



9. Security Best Practices

- Serve iDOCK behind HTTPS (reverse proxy) and set `IDOCK_SESSION_HTTPS_ONLY=true`.
- Passwords are hashed with bcrypt; API tokens are stored only as SHA-256 hashes.
- Give each host its own token so you can revoke one without affecting others.
- Mount the agent's Docker socket read-only unless you need remote Start/Stop/Update.
- Prefer agents or TLS/SSH over unauthenticated `tcp://:2375` daemons.

10. Troubleshooting

- **Agent not appearing:** confirm Docker is running on the host and the agent container is up; check outbound reachability to `IDOCK_URL`.
- **401 in agent logs:** the token is wrong or revoked — generate a fresh one on Deploy Agent.
- **Command shows localhost:** set `IDOCK_PUBLIC_URL` or edit the Server Address field on the Deploy Agent page.
- **Start/Stop do nothing:** the agent's socket is mounted read-only — remount read-write.
- **Removed host reappears:** use the **Remove Agent** menu (which blocks the agent id) rather than deleting the host directly — a blocked agent cannot re-register.



11. Frequently Asked Questions

Do I need to expose the Docker API (port 2375/2376) on my hosts?

No. That's the whole point of the agent — it reads the local socket and reports outbound over HTTPS, so no inbound Docker port is opened. Exposing the TCP API is only needed if you choose to add a host manually as a TCP/TLS/SSH endpoint.

Is there an approval step before containers show up?

No. Agents authenticate with a revocable token, which is the trust boundary, so their hosts and containers appear on the dashboard automatically — usually within ~20 seconds.

How do I remove one agent without disconnecting the others?

Use the Remove Agent menu item and pick the agent. It blocks just that agent id — so it cannot re-register — and clears its host and containers, while the shared token stays valid for every other agent. Then run the uninstall command it shows for the host. (Revoking a token, by contrast, would disconnect every agent that uses it.)

The install command shows 'localhost' — will it work on another machine?

The command auto-fills the server's reachable LAN IP even when you browse from the server itself. If your clients reach iDOCK at a different address (e.g. a domain), set IDOCK_PUBLIC_URL or edit the editable Server Address field before copying.

Can I control containers from the dashboard, or only monitor them?

Both. Start, Stop and Update work inline. For agent hosts these are queued and executed by the agent on its next poll, provided the Docker socket was mounted read-write.

Where is my data stored, and does it survive restarts?

Everything persists in the /data Docker volume: the SQLite database, the session secret, uploaded certificates, and metric history. Keep that volume to retain your configuration.

Do agents work across CPU architectures?

The host-built image matches the server's architecture. For a mixed fleet (e.g. amd64 server, arm64 clients), publish a multi-arch image to a registry and set IDOCK_AGENT_IMAGE.

How are alerts delivered?

Via webhooks. Configure a URL and triggers (Container Down/Recovered, High CPU, Restart Loop); iDOCK POSTs a JSON payload with optional custom headers for services like Slack or IronNode SOAR.

Can I automate iDOCK from scripts or a SOAR platform?

Yes. Use the token-authenticated REST API under /api/v1 to list hosts and containers and to run start/stop/restart/pull/logs.

Is Synapse-iDOCK free to self-host?

It is delivered as a self-hosted application within the Synapse ecosystem; run it on your own infrastructure with the provided Docker Compose stack.

Thank you for choosing Synapse-iDOCK — orchestration that stays out of your way.