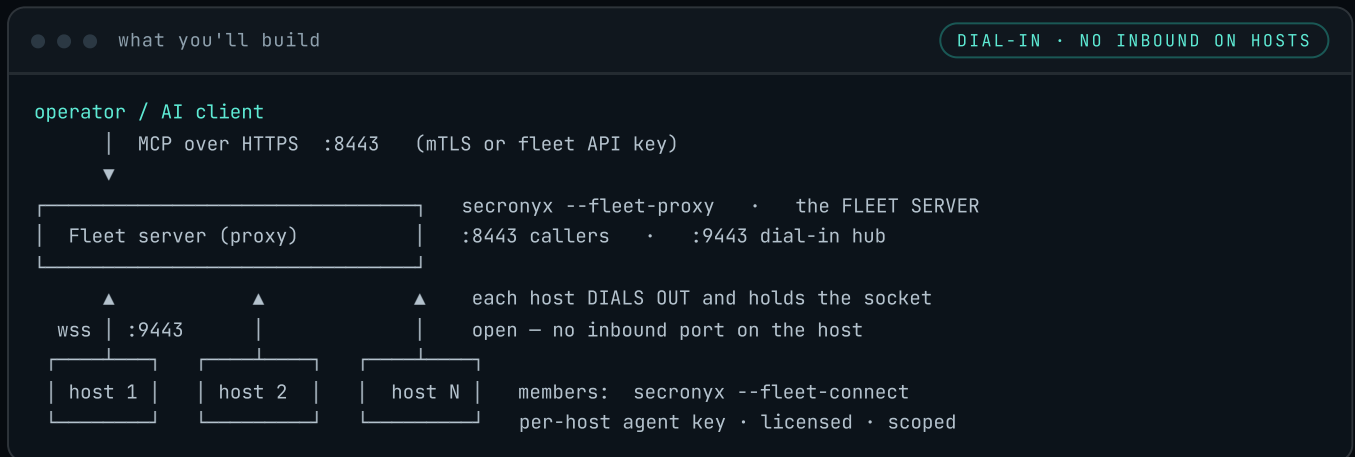


• SELF-HOSTED FLEET SERVER • DIAL-IN MEMBERS

Stand up the fleet. Enrol your estate.

This guide installs the **Secronyx fleet server** (the proxy), enrolls your hosts as **dial-in members** that open no inbound ports, and connects a client that can query the whole estate — up to 5,000 hosts — from one authenticated endpoint. Four steps, read-only throughout.



1

FLEET SERVER

2

PORTS (8443 · 9443)

5,000

HOSTS, ONE ENDPOINT

4

STEPS TO LIVE

YOU WILL NEED

A verified Secronyx binary

A valid licence

TLS cert + key for the proxy host

Caller auth (mTLS CA or API keys)

Outbound :9443 from each host → proxy

Three roles, one authenticated door.

A fleet has three parts. Callers authenticate **once, at the fleet server**; members **dial out** to it and answer over that connection, so no endpoint exposes an inbound port. Every host still runs read-only and enforces its own scopes.

THE COMPONENTS

Fleet server proxy

One host runs **secronyx --fleet-proxy**. Serves callers on **:8443** and the dial-in hub on **:9443**; fans each call out and aggregates replies.

Members your hosts

Each host runs **secronyx --fleet-connect** as a fleet-only member — no listener, licensed, answering only over its outbound socket.

Client operator / AI

Claude Code or any MCP client points at the fleet server and calls **fleet_run** / **fleet_ask** across the estate.

NETWORK & PORTS

PORT	DIRECTION	SOURCE	PURPOSE
:8443/tcp	inbound → fleet server	operator CIDR	MCP for callers (mTLS or fleet API key)
:9443/tcp	inbound → fleet server	each member's address	dial-in hub (per-host agent key)
:9443/tcp	outbound ← each member	member → fleet server	the one connection a member needs
—	inbound → members	none	members expose no inbound port

PREREQUISITES

On every host

A **verified** Secronyx binary (Step 1) and a **valid licence** — the release build refuses to start without one.

On the fleet server

A **TLS cert + key** for its hostname, plus caller auth: a client-cert **CA** (mTLS) and/or an **API-key store**.

READ-ONLY

The proxy only ever sends **tools/list** and **tools/call**. A caller's scopes carry through every forwarded call, and nothing is ever written to a host.



Install & verify the binary.

Put a verified binary and a licence on **every host**, including the one that will be the fleet server.

Downloads are open; verify the checksum, signature and provenance before you run.

● ● ● install (Linux example)

VERIFY BEFORE YOU RUN

```
# 1 · download the binary + its proofs from secronyx.com/downloads
curl -fLO https://secronyx.com/downloads/latest/secronyx-linux-amd64
curl -fLO https://secronyx.com/downloads/latest/secronyx-linux-amd64.cosign.bundle
curl -fLO https://secronyx.com/downloads/latest/SHA256SUMS

# 2 · verify — checksum and signature (cosign)
sha256sum -c SHA256SUMS --ignore-missing
cosign verify-blob --bundle secronyx-linux-amd64.cosign.bundle secronyx-linux-amd64

# 3 · install the binary and the licence
sudo install -m 0755 secronyx-linux-amd64 /usr/local/bin/secronyx
sudo mkdir -p /etc/secronyx && sudo cp your.license /etc/secronyx/license

# 4 · confirm it runs (version is always allowed, licensed or not)
secronyx --version
```

DEFAULT LICENCE PATHS

OS	BINARY	LICENCE (DEFAULT PATH)
Linux	/usr/local/bin/secronyx	/etc/secronyx/license
macOS	/opt/secronyx/bin/secronyx	/Library/Application Support/Secronyx/license
Windows	C:\Program Files\Secronyx\secronyx.exe	C:\ProgramData\Secronyx\license

LICENCE

Every release artefact is **enforced** — a host with no valid licence refuses to start. Place the licence before you start the service, or pass `--license-file <path>` explicitly.

RENEWALS

Your licence is a single signed file issued by Levantar AI. For more hosts, renewals or a replacement licence, contact sales@levantar.ai.

Configure & run the fleet server.

On the proxy host, mint the keys callers and members will use, declare the inventory, then start the server listening for both callers (:8443) and dial-in members (:9443).

● ● ● a · mint keys on the proxy

```
# caller key — needs the fleet scope plus the tool scopes they may use
secronyx apikey create --file callers.json --name soc-analyst --scopes fleet,core,triage,logs,security
# one dial-in agent key per host — name it after the host (repeat per host)
secronyx apikey create --file agent-keys.json --name web-1 # prints msk_... → that host
secronyx apikey create --file agent-keys.json --name dc-1
```

● ● ● b · fleet-inventory.json — a host with no url is expected to dial in

```
{
  "version": 1,
  "defaults": { "tags": { "env": "prod" } },
  "hosts": [
    { "name": "web-1", "os": "linux", "tags": { "role": "web" } },
    { "name": "dc-1", "os": "windows", "tags": { "role": "dc" } }
  ]
}
```

● ● ● c · run the fleet server

CALLERS :8443 · HUB :9443

```
secronyx --fleet-proxy --fleet-inventory /etc/secronyx/fleet-inventory.json \
  --transport http --listen 0.0.0.0:8443 --server-url https://fleet.example.com:8443 \
  --tls-cert /etc/secronyx/tls.crt --tls-key /etc/secronyx/tls.key \
  --api-keys-file /etc/secronyx/callers.json \
  --fleet-agent-listen 0.0.0.0:9443 \
  --fleet-agent-tls-cert /etc/secronyx/tls.crt --fleet-agent-tls-key /etc/secronyx/tls.key \
  --fleet-agent-api-keys-file /etc/secronyx/agent-keys.json \
  --audit-output /var/log/secronyx/fleet-proxy-audit.jsonl --enable-sensitive
```

MTLS OPTION

Prefer certificates? Swap `--api-keys-file` for `--tls-client-ca` + `--mtls-identity-file` for callers, and `--fleet-agent-client-ca` + `--fleet-agent-identities` for members.

SCOPES

A caller reaches a tool only if its key holds that tool's scope; the proxy's own `--scopes` / `--enable-sensitive` cap what it will forward at all.

3

STEP 3 · EACH HOST

Enrol each host as a dial-in member.

On every host, drop in its **agent key** (printed in Step 2a) and its licence, then start the member. It opens one outbound websocket to the fleet server and serves its tools over it — no inbound port, nothing to firewall.

● ● ● run a member (foreground, to test)

FLEET-ONLY · NO LISTENER

```
# place the host's agent key (the msk_... printed for THIS host) + licence
printf '%s' "msk_..." | sudo tee /etc/secronyx/fleet-agent-key >/dev/null

secronyx --fleet-connect wss://fleet.example.com:9443/fleet/connect \
  --fleet-name web-1 --fleet-tags env=prod,os=linux,role=web \
  --fleet-key-file /etc/secronyx/fleet-agent-key \
  --license-file /etc/secronyx/license \
  --audit-output /var/log/secronyx/audit.jsonl
```

FLEET-ONLY

The name a member announces must match its agent key. Passing `--transport` alongside `--fleet-connect` is refused; `secronyx --query ...` still works locally for troubleshooting.

RUN IT AS A SERVICE

LINUX · SYSTEMD

```
# /etc/systemd/system/secronyx.service
[Service]
ExecStart=/usr/local/bin/secronyx --
fleet-connect
wss://fleet.example.com:9443/fleet/conn
ect --fleet-name web-1 --fleet-key-file
/etc/secronyx/fleet-agent-key --
license-file /etc/secronyx/license
Restart=always

systemctl enable --now secronyx
```

WINDOWS · TASK

```
# SYSTEM, at startup (PowerShell)
$a = New-ScheduledTaskAction `
-Execute "C:\Program
Files\Secronyx\secronyx.exe" `
-Argument "--fleet-connect
wss://fleet.example.com:9443/fleet/conn
ect --fleet-name dc-1 --fleet-key-file
C:\ProgramData\Secronyx\fleet-agent-
key"
Register-ScheduledTask Secronyx -Action
$a `
-Trigger (New-ScheduledTaskTrigger -
AtStartup) -User SYSTEM
```

MACOS · LAUNCHD

```
#
/Library/LaunchDaemons/com.secronyx.pli
st
ProgramArguments:
/opt/secronyx/bin/secronyx
--fleet-connect
wss://fleet.example.com:9443/fleet/conn
ect
--fleet-name mac-1
--fleet-key-file /etc/secronyx/fleet-
agent-key
--license-file /etc/secronyx/license

sudo launchctl load -w
/Library/LaunchDaemons/com.secronyx.pli
st
```

SCALE

Repeat per host — the server holds one websocket each and fans out **500 in flight** by default (`--fleet-concurrency`), tuned for up to 5,000 hosts.

4

STEP 4 · THE CLIENT

Connect your client & verify.

Point an MCP client at the fleet server, confirm your members are connected, then ask one question of the whole estate. Tool arguments are JSON; omit **targets** to hit every host, or filter by os, tags or names.

client config — one MCP server for the whole fleet

```
// Claude Code — HTTP to the fleet server, fleet API key
{ "mcpServers": { "fleet": {
  "type": "http",
  "url": "https://fleet.example.com:8443/",
  "headers": { "X-API-Key": "msk_your_caller_key" } } } }
```

verify

READ-ONLY

```
# who's connected? (members that dialled in, with tags + reachability)
fleet_hosts { }
# run one read-only tool across hosts — omit "targets" for all; or filter by os/tags/names
fleet_run { "tool": "get_os_info" }
fleet_run { "tool": "get_os_info", "targets": { "os": ["windows"] } }
# or ask in natural language — the proxy picks the tools offline
fleet_ask { "prompt": "which hosts have Defender disabled?" }
```

SECURE & OPERATE

Firewall

:8443 from operator networks only;
:9443 from member addresses only.
Members need no inbound rule.

Scopes carry through

A caller can reach only the tools its key
is scoped for — enforced again on
each host.

Audit both ends

Proxy logs fleet/fanout; each host
logs the call under the proxy identity
with the caller + run id.

Resilience

One retry on transport failure; a host
failing 3× is skipped for a minute
(circuit_open).

Rotate keys

```
secronyx apikey revoke --file
... --id ... to cut a caller or member
without a restart.
```

Licence & updates

Keep licences current; roll a new
verified binary to hosts and restart the
service.

DONE

Your fleet server is live. Full reference: secronyx.com/docs.