readme.md for @serve.zone/onebox

> □□ Self-hosted Docker Swarm platform with Caddy reverse proxy, automatic SSL, and real-time WebSocket updates

Onebox transforms any Linux server into a powerful container hosting platform. Deploy Docker Swarm services with automatic HTTPS, DNS configuration, and Caddy reverse proxy running as a Docker service - all managed through a beautiful Angular web interface with real-time updates.

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

What Makes Onebox Different? □□

- **Caddy Reverse Proxy in Docker** - Production-grade HTTP/HTTPS proxy running as a Swarm service with native service discovery, HTTP/2, HTTP/3, and bidirectional WebSocket proxying
- **Docker Swarm First** - All workloads (including the reverse proxy!) run as Swarm services on the overlay network for seamless service-to-service communication
- **Real-time Everything** - WebSocket-powered live updates for service status, logs, and metrics across all connected clients
- **Single Executable** - Compiles to a standalone binary - just run it, no dependencies
- **Private Registry Included** - Built-in Docker registry with token-based auth and auto-deploy on push
- **Zero Config SSL** - Automatic Let's Encrypt certificates with inline `load_pem` (no volume mounts needed)
- **Cloudflare Integration** - Automatic DNS record management and zone synchronization

- **Modern Stack** - Deno runtime + SQLite database + Angular 19 UI

Features

Core Platform

- **Docker Swarm Management** - Deploy, scale, and orchestrate services with Swarm mode
- **Caddy Reverse Proxy** - Production-grade proxy running as Docker service with SNI, HTTP/2, HTTP/3
- **Automatic SSL Certificates** - Let's Encrypt integration with hot-reload and renewal monitoring
- ☁ **Cloudflare DNS Integration** - Automatic DNS record creation and zone synchronization
- **Built-in Registry** - Private Docker registry with per-service tokens and auto-update
- **Real-time WebSocket Updates** - Live service status, logs, and system events

Monitoring & Management

- **Metrics Collection** - Historical CPU, memory, and network stats (every 60s)
- **Centralized Logging** - Container logs with streaming and retention policies
- **Angular Web UI** - Modern, responsive interface with real-time updates
- **Multi-user Support** - Role-based access control (admin/user)
- **SQLite Database** - Embedded, zero-configuration storage

Developer Experience

- **Auto-update on Push** - Push to registry and services update automatically
- **Private Registry Support** - Use Docker Hub, Gitea, or custom registries
- **Systemd Integration** - Run as a daemon with auto-restart
- **Full CLI & API** - Manage everything from terminal or HTTP API

Quick Start

Installation

```
# One-line install (recommended)
curl -sSL https://code.foss.global/serve.zone/onebox/raw/branch/main/install.sh | sudo bash

# Install a specific version
curl -sSL https://code.foss.global/serve.zone/onebox/raw/branch/main/install.sh | sudo bash -s
-- --version v1.11.0

# Or install from npm
pnpm install -g @serve.zone/onebox
```

First Run

```
# Start the server in development mode
onebox server --ephemeral

# In another terminal, deploy your first service
onebox service add myapp \
  --image nginx:latest \
  --domain app.example.com \
  --port 80
```

Access the Web UI

Open `http://localhost:3000` in your browser.

Default credentials:

- Username: `admin`
- Password: `admin`

⚠ **Change the default password immediately after first login!**

Production Setup

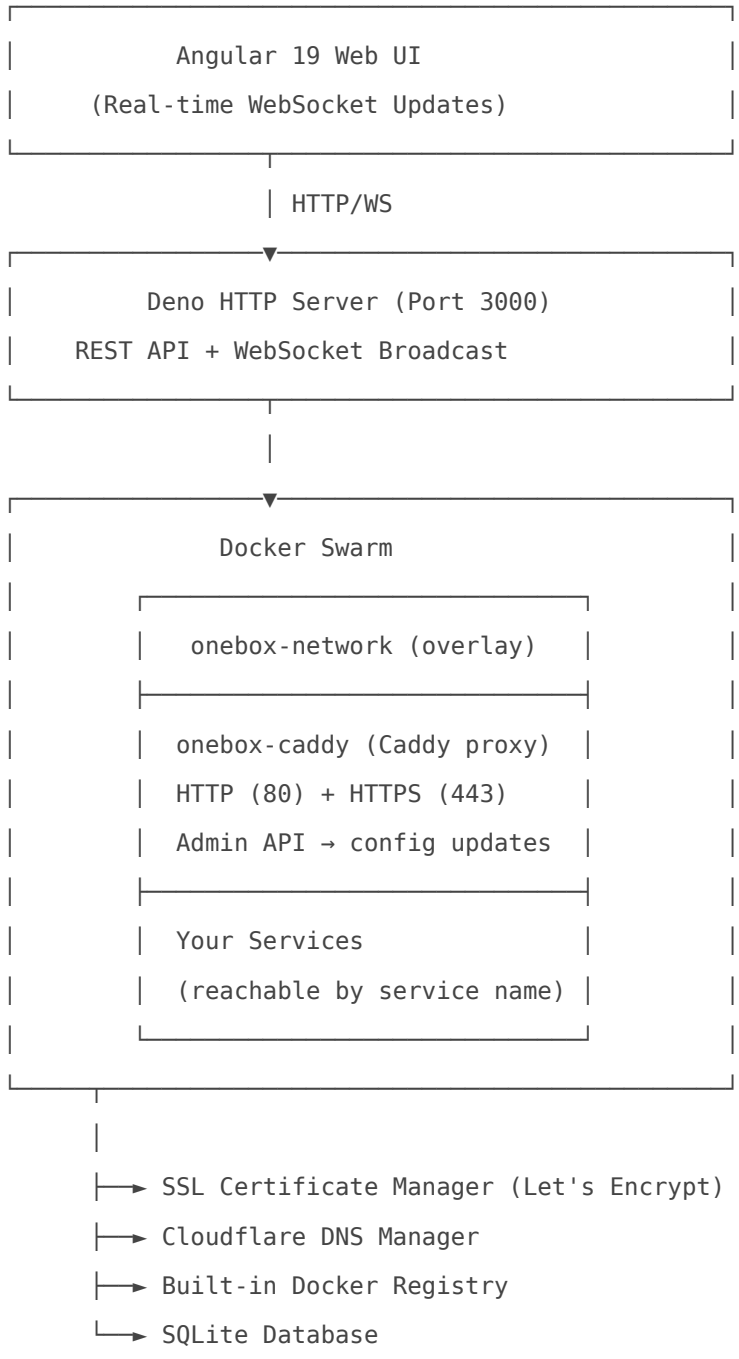
```
# Install as systemd service
sudo onebox daemon install
```

```
# Start the daemon
sudo onebox daemon start

# View logs
sudo onebox daemon logs
```

Architecture □□

Onebox is built with modern technologies for performance and developer experience:



Core Components

Component	Description
Deno Runtime	Modern TypeScript with built-in security
Caddy Reverse Proxy	Docker Swarm service with HTTP/2, HTTP/3, SNI, and WebSocket support
Docker Swarm	Container orchestration (all workloads run as services)
SQLite Database	Configuration, metrics, and user data
WebSocket Server	Real-time bidirectional communication
Let's Encrypt	Automatic SSL certificate management
Cloudflare API	DNS record automation

CLI Reference □□

Service Management

```
# Deploy a service
onebox service add <name> --image <image> --domain <domain> [--port <port>] [--env KEY=VALUE]

# Deploy with Onebox Registry (auto-update on push)
onebox service add myapp --use-onebox-registry --domain myapp.example.com

# List services
onebox service list

# Control services
onebox service start <name>
onebox service stop <name>
onebox service restart <name>

# Remove service
onebox service remove <name>

# View logs
onebox service logs <name>
```

Server Management

```
# Start server (development)
onebox server --ephemeral          # Runs in foreground with monitoring

# Start server (production)
onebox daemon install              # Install systemd service
onebox daemon start                # Start daemon
onebox daemon stop                 # Stop daemon
onebox daemon logs                 # View logs
```

Registry Management

```
# Add external registry credentials
onebox registry add --url registry.example.com --username user --password pass

# List registries
onebox registry list

# Remove registry
onebox registry remove <url>
```

DNS Management

```
# Add DNS record (requires Cloudflare config)
onebox dns add <domain>

# List DNS records
onebox dns list

# Sync from Cloudflare
onebox dns sync

# Remove DNS record
onebox dns remove <domain>
```

SSL Management

```
# Renew expiring certificates
onebox ssl renew

# Force renew specific domain
onebox ssl force-renew <domain>

# List certificates
onebox ssl list
```

Configuration

```
# Show all settings
onebox config show

# Set configuration value
onebox config set <key> <value>

# Example: Configure Cloudflare
onebox config set cloudflareAPIKey your-api-key
onebox config set cloudflareEmail your@email.com
onebox config set cloudflareZoneID your-zone-id
```

System Status

```
# Get full system status
onebox status
```

Upgrade

```
# Upgrade to the latest version (requires root)
sudo onebox upgrade
```

Configuration □□

System Requirements

- **Linux** (x64 or ARM64)
- **Docker** installed and running
- **Docker Swarm** initialized (`docker swarm init`)
- **Root/sudo access** for ports 80/443
- **(Optional) Cloudflare account** for DNS automation

Data Locations

Data	Location
Database	`./onebox.db` (or custom path)
SSL Certificates	Managed by CertManager
Registry Data	`./.nogit/registry-data`

Environment Variables

```
# Database location
ONEBOX_DB_PATH=/path/to/onebox.db

# HTTP server port (default: 3000)
ONEBOX_HTTP_PORT=3000

# Enable debug logging
ONEBOX_DEBUG=true
```

Development □□

Setup

```
# Clone repository
git clone https://code.foss.global/serve.zone/onebox
cd onebox

# Start development server (auto-restart on changes)
pnpm run watch
```

Tasks

```
# Development server (auto-restart on changes)
deno task dev

# Run tests
deno task test

# Watch mode for tests
deno task test:watch

# Compile binaries for all platforms
deno task compile
```

Project Structure

```
onebox/
├── ts/
│   ├── classes/                    # Core implementations
│   │   ├── onebox.ts               # Main coordinator
│   │   ├── reverseproxy.ts         # Reverse proxy orchestration
│   │   ├── caddy.ts                # Caddy Docker service management
│   │   ├── docker.ts               # Docker Swarm API
│   │   ├── httpserver.ts           # REST API + WebSocket
│   │   ├── services.ts             # Service orchestration
│   │   ├── certmanager.ts          # SSL certificate management
│   │   ├── cert-requirement-manager.ts  # Certificate requirements
│   │   ├── ssl.ts                  # SSL utilities
│   │   ├── registry.ts             # Built-in Docker registry
│   │   ├── registries.ts           # External registry management
```

```
│   │   ├── dns.ts                # DNS record management
│   │   ├── cloudflare-sync.ts    # Cloudflare zone sync
│   │   ├── daemon.ts             # Systemd daemon management
│   │   └── apiclient.ts          # API client utilities
│   ├── database/                 # Database layer (repository pattern)
│   │   ├── index.ts              # Main OneboxDatabase class
│   │   ├── base.repository.ts    # Base repository class
│   │   └── repositories/         # Domain-specific repositories
│   │       ├── service.repository.ts
│   │       ├── certificate.repository.ts
│   │       ├── auth.repository.ts
│   │       ├── metrics.repository.ts
│   │       └── ...
│   ├── cli.ts                    # CLI router
│   ├── types.ts                  # TypeScript interfaces
│   ├── logging.ts                # Logging utilities
│   └── plugins.ts                # Dependency imports
├── ui/                           # Angular 19 web interface
├── test/                         # Test files
├── mod.ts                        # Main entry point
└── deno.json                     # Deno configuration
```

API Endpoints

The HTTP server exposes a comprehensive REST API:

Authentication

Method	Endpoint	Description
`POST`	`/api/auth/login`	User authentication (returns token)

Services

Method	Endpoint	Description
`GET`	`/api/services`	List all services
`POST`	`/api/services`	Create/deploy service
`GET`	`/api/services/:name`	Get service details
`PUT`	`/api/services/:name`	Update service

Method	Endpoint	Description
`DELETE`	`/api/services/:name`	Delete service
`POST`	`/api/services/:name/start`	Start service
`POST`	`/api/services/:name/stop`	Stop service
`POST`	`/api/services/:name/restart`	Restart service
`GET`	`/api/services/:name/logs`	Get service logs
`WS`	`/api/services/:name/logs/stream`	Stream logs via WebSocket

SSL Certificates

Method	Endpoint	Description
`GET`	`/api/ssl/list`	List all certificates
`GET`	`/api/ssl/:domain`	Get certificate details
`POST`	`/api/ssl/obtain`	Request new certificate
`POST`	`/api/ssl/:domain/renew`	Force renew certificate

Domains

Method	Endpoint	Description
`GET`	`/api/domains`	List all domains
`GET`	`/api/domains/:domain`	Get domain details
`POST`	`/api/domains/sync`	Sync domains from Cloudflare

DNS Records

Method	Endpoint	Description
`GET`	`/api/dns`	List DNS records
`POST`	`/api/dns`	Create DNS record
`DELETE`	`/api/dns/:domain`	Delete DNS record
`POST`	`/api/dns/sync`	Sync DNS from Cloudflare

Registry

Method	Endpoint	Description
`GET`	`/api/registry/tags/:service`	Get registry tags for service
`GET`	`/api/registry/tokens`	List registry tokens

Method	Endpoint	Description
`POST`	`/api/registry/tokens`	Create registry token
`DELETE`	`/api/registry/tokens/:id`	Delete registry token

System

Method	Endpoint	Description
`GET`	`/api/status`	System status
`GET`	`/api/settings`	Get settings
`PUT`	`/api/settings`	Update settings
`WS`	`/api/ws`	WebSocket for real-time updates

WebSocket Messages

Real-time updates are broadcast via WebSocket:

```
// Service lifecycle updates
{
  type: 'service_update',
  action: 'created' | 'updated' | 'deleted' | 'started' | 'stopped',
  service: { id, name, status, ... }
}

// Service status changes
{
  type: 'service_status',
  service: { id, name, status, ... }
}

// System status updates
{
  type: 'system_status',
  status: { docker, reverseProxy, services, ... }
}
```

Advanced Usage □□

Using the Built-in Registry

```
# Deploy a service with Onebox Registry
onebox service add myapp \
  --use-onebox-registry \
  --domain myapp.example.com \
  --auto-update-on-push

# Get the registry token for pushing images
# (Token is automatically created and stored in database)

# Push your image
docker tag myimage:latest localhost:4000/myapp:latest
docker push localhost:4000/myapp:latest

# Service automatically updates! □□
```

Registry Token Management

```
# Create a CI/CD token via API
curl -X POST http://localhost:3000/api/registry/tokens \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"name": "github-actions", "type": "ci", "scope": ["myapp"], "expiresIn": "90d"}'

# Use token for docker login
docker login localhost:4000 -u ci -p <token>
```

Cloudflare DNS Integration

```
# Configure Cloudflare (one-time setup)
onebox config set cloudflareAPIKey your-api-key
onebox config set cloudflareEmail your@email.com
onebox config set cloudflareZoneID your-zone-id

# Deploy with automatic DNS
```

```
onebox service add myapp \
  --image nginx:latest \
  --domain myapp.example.com

# DNS record is automatically created!

# Sync all domains from Cloudflare
onebox dns sync
```

SSL Certificate Management

SSL certificates are automatically obtained and renewed:

- □ Certificates are requested when a service with a domain is deployed
- □ Renewal happens automatically 30 days before expiry
- □ Certificates are hot-reloaded without downtime
- □ Force renewal: `onebox ssl force-renew <domain>`

Monitoring and Metrics

Metrics are collected every 60 seconds (configurable):

```
# Set metrics interval (milliseconds)
onebox config set metricsInterval 30000

# View in web UI or query database directly
sqlite3 onebox.db "SELECT * FROM metrics WHERE service_id = 1 ORDER BY timestamp DESC LIMIT
10"
```

Troubleshooting □□

Docker Swarm Not Initialized

```
# Initialize Docker Swarm
docker swarm init
```

```
# Verify swarm mode
docker info | grep "Swarm: active"
```

Port Already in Use

```
# Check what's using port 80/443
sudo lsof -i :80
sudo lsof -i :443

# Kill the process or change Onebox ports
onebox config set httpPort 8080
```

SSL Certificate Issues

```
# Check certificate status
onebox ssl list

# Verify DNS is pointing to your server
dig +short yourdomain.com

# Force certificate renewal
onebox ssl force-renew yourdomain.com
```

WebSocket Connection Issues

- Ensure firewall allows WebSocket connections
- Check browser console for connection errors
- Verify `/api/ws` endpoint is accessible

Service Not Starting

```
# Check Docker logs
docker service logs <service-name>

# Check Onebox logs
onebox daemon logs
```

```
# Verify image exists
docker images | grep <image-name>
```

License and Legal Information

This repository contains open-source code that is licensed under the MIT License. A copy of the MIT License can be found in the license file within this repository.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH and are not included within the scope of the MIT license granted herein. Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines, and any usage must be approved in writing by Task Venture Capital GmbH.

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

Company Information

Task Venture Capital GmbH Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

Revision #3
Created 2026-03-28 11:14:37 UTC by foss.global Team
Updated 2026-03-28 12:21:24 UTC by foss.global Team