Установка UMNI Server

Инструкции по развертыванию экосистемы UMNI Server в различных окружениях.

- Установка UMNI Server в среде виртуализации Proxmox VE на Debian 13 + Docker
- Установка UNI Server на Ubuntu 26.04

Установка UMNI Server в среде виртуализации Proxmox VE на Debian 13 + Docker

В этой статье пойдет речь об установке UMNI Server в среде Proxmox VE (9.2). Работы будем проводить в контейнере LXC с включенным nesting и выключенным Unprivileged container.

Подготовка окружения

В качестве контейнера используем шаблон Debian 13 (https://docs.docker.com/engine/install/debian/).

После создания LXC обновите пакеты

```
su
apt update
apt upgrade
```

Теперь необходимо установить docker и docker-compose

```
# Add Docker's official GPG key:
apt update
apt install ca-certificates curl
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc
chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
tee /etc/apt/sources.list.d/docker.sources <<EOF
Types: deb
```

```
URIs: https://download.docker.com/linux/debian
Suites: $(. /etc/os-release && echo "$VERSION_CODENAME")
Components: stable
Architectures: $(dpkg --print-architecture)
Signed-By: /etc/apt/keyrings/docker.asc
EOF

apt update
```

Теперь установим Docker, docker-compose

```
apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Проверка, что Docker запущен

```
systemctl status docker
```

Установка UMNI Server

Создайте каталог, в котором будет находиться UMNI Server. Структура файлов для минимального старта такая:

```
umnu-ecosystem/                  <- любой каталог, где лежат файлы ниже
├── docker-compose.release.yml   # обязательно: запускает контейнеры из образов
├── nginx.conf                   # обязательно: монтируется в frontend, без него контейнер не стартует
├── env_config/
│   └── .env                     # обязательно: БД + настройки backend (сюда пишется ENCRYPTION_KEY)
├── .env                         # опционально: интерполяция compose (WEB_PORT, DB_NAME, ...)
├── storage/                     # внутренние данные; создаётся автоматически
├── logs/                        # логи backend; создаётся автоматически
└── plugins/
    └── custom/                  # пользовательские плагины
```

Создадим папку `/opt/umni-ecosystem`:

```
mkdir /opt/umni-ecosystem -p
```

Теперь создадим `docker-compose.release.yml`

```
cd /opt/umni-ecosystem
nano docker-compose.release.yml
```

Вставьте в него следующий код

```
# Развёртывание из готовых образов Docker Hub (без сборки из исходников).
#
# Быстрый старт:
#   1) cp .env.example .env   (при необходимости отредактируйте)
#   2) создайте каталоги:     mkdir env_config storage logs
#   3) положите свой .env бэкенда в env_config/.env (ключи см. DOCKER.md)
#   4) добавьте маунты хранилищ хоста в раздел volumes (см. ниже)
#   5) docker compose -f docker-compose.release.yml up -d
#
# Образы по умолчанию берутся из Docker Hub. Переопределите при необходимости:
#   $env:IMAGE_OWNER="<dockerhub-user>"; $env:IMAGE_TAG="v1.2.3"
#   docker compose -f docker-compose.release.yml up -d
#
# HOST-РЕЖИМ (network_mode: host):
#   backend  в сети хоста — нужен для multicast DNS (поиск устройств _umni_api._tcp.local.)
#            и приёма UDP-широковещательного syslog (порт 514) от устройств.
#   frontend в сети хоста — nginx слушает 80 на хосте, проксирует на backend по loopback.
#   postgres остаётся на bridge app-network, публикуется только на 127.0.0.1:5432.
#
# ВАЖНО: сервис с network_mode: host не может иметь networks:, ports:, expose:.
#   Ошибка "only one instance of 'host' network is allowed" = это правило нарушено.

services:
  postgres:
    image: postgres:16-alpine
    container_name: umni_postgres
    restart: unless-stopped
    env_file:
      - ./env_config/.env
    environment:
      TZ: Europe/Moscow
      POSTGRES_DB: ${DB_NAME:-umni}
      POSTGRES_USER: ${DB_USER:-postgres}
      POSTGRES_PASSWORD: ${DB_PASSWORD:-postgres}
    ports:
```

```
    - "127.0.0.1:5432:5432"
  volumes:
    - postgres_data:/var/lib/postgresql/data
  networks:
    - app-network
  healthcheck:
    test: ["CMD-SHELL", "pg_isready -U ${DB_USER:-postgres}"]
    interval: 10s
    timeout: 5s
    retries: 5

backend:
  image: ${IMAGE_OWNER:-sazanof}/umni-ecosystem-backend:${IMAGE_TAG:-latest}
  container_name: umni_backend
  restart: unless-stopped
  network_mode: host
  command: uvicorn main:app --host 127.0.0.1 --port 8000
  depends_on:
    postgres:
      condition: service_healthy
  env_file:
    - ./env_config/.env
  environment:
    DB_HOST: 127.0.0.1
    DB_USER: ${DB_USER:-postgres}
    DB_PASSWORD: ${DB_PASSWORD:-postgres}
    DB_NAME: ${DB_NAME:-umni}
    DB_PORT: ${DB_PORT:-5432}
    LOG_LEVEL: ${LOG_LEVEL:-INFO}
    LOG_DIR: /app/logs
    STORAGE_HOST_BASE: ${STORAGE_HOST_BASE:-/host}
    TZ: Europe/Moscow
  volumes:
    - ./logs:/app/logs
    # .env монтируется на запись: приложение само хранит ENCRYPTION_KEY.
    - ./env_config:/app/env_config
    - ./storage:/app/storage
    - ./plugins/custom:/app/plugins/custom
    # Хранилища хоста. Путь в storages задаётся ХОСТОВЫЙ (например "D:\CameraData"),
    # backend сам переведёт его в контейнерный: "<STORAGE_HOST_BASE>/D/CameraData".
```

```
      #
      # Windows-хост (Docker Desktop): монтируем весь диск в /host/<буква>.
      # - type: bind
      #   source: 'D:\'
      #   target: /host/D
      #
      # Linux-хост: путь в storages задаётся контейнерный, как в маунте ниже.
      # - type: bind
      #   source: /mnt/rec
      #   target: /mnt/rec

  frontend:
    image: ${IMAGE_OWNER:-sazanof}/umni-ecosystem-frontend:${IMAGE_TAG:-latest}
    container_name: umni_frontend
    restart: unless-stopped
    network_mode: host
    volumes:
      - ./nginx.host.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      backend:
        condition: service_healthy

volumes:
  postgres_data:

networks:
  app-network:
    driver: bridge
```

Монтирование своих директорий и дисков

Если у вас в системе есть несколько дисков и вы хотите добавить их как кастомное хранилище, например для камер, то необходимо добавить соответствующий блок в `docker-compose.release.yml`

Пример: узнаем сколько томов LVM и куда примонтированы (актуально для LXC):

```
root@umni-ecosystem:/opt/umni-ecosystem# df -h
Filesystem                                  Size  Used Avail Use% Mounted on
/dev/mapper/data--ssd--500gb-vm--106--disk--0   20G  1.2G   18G   7% /
/dev/mapper/data--hdd--1tb-vm--106--disk--0     35G  2.1M   33G   1% /mnt/umni-data
```

Редактируем `docker-compose.release.yml`

```
- type: bind
  source: /mnt/umni-data
  target: /mnt/umni-data
```

Создайте недостающие директории и файлы

```
mkdir env_config storage logs
```

Создайте файл `env_config/.env`

```
APP_MODE=production
DB_HOST=postgres
DB_USER=postgres
DB_PASSWORD=postgres
DB_NAME=ecosystem
DB_PORT=5432
LOG_LEVEL=debug
DEBUG_MODE=app,cameras,devices,rules,mqtt,services,storages,systeminfo,scheduler,tasks,websock
ets,notifications,plugins
```

Создайте `.env` на основе `.env.example`

```
# Скопируйте этот файл в .env:  cp .env.example .env
# ДЕРЖИТЕ ЗНАЧЕНИЯ В СИНХРОНЕ С env_config/.env (backend).
# Этот файл используется ТОЛЬКО для интерполяции переменных docker compose
# (имя БД, порт фронтенда и т.п.) и НЕ передаётся в контейнеры напрямую.

# Postgres, создаваемая контейнером postgres
DB_USER=postgres
DB_PASSWORD=postgres
DB_NAME=ecosystem
DB_PORT=5432

# Порт, на котором наружу будет доступен веб-интерфейс (http://localhost:WEB_PORT)
WEB_PORT=80

# Уровень логирования backend (INFO/DEBUG)
LOG_LEVEL=INFO
```

```
# Корень, куда монтируются диски хоста (backend переводит D:\... -> /host/D/...)
STORAGE_HOST_BASE=/host
```

Если есть плагины, скопируйте их в `plugins/custom`, предварительно создав ее

```
mkdir plugins/custom -p
```

Создайте nginx.conf

```
events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    sendfile on;
    keepalive_timeout 65;

    client_max_body_size 100m;

    server {
        listen 80;
        server_name localhost;

        root /usr/share/nginx/html;
        index index.html;

        location / {
            try_files $uri $uri/ /index.html;
        }

        # Роуты бэкенда живут без префикса /api (auth, ws, cameras, ...),
        # поэтому срезаем /api через завершающий слэш в proxy_pass.
        # Хост-режим: backend слушает 127.0.0.1:8000 в сети хоста.
        location /api/ {
            proxy_pass http://127.0.0.1:8000/;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
```

```
            proxy_set_header Connection "upgrade";
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_read_timeout 3600s;
            proxy_send_timeout 3600s;
        }

        location /docs {
            proxy_pass http://127.0.0.1:8000/docs;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
        }

        location /openapi.json {
            proxy_pass http://127.0.0.1:8000/openapi.json;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
        }
    }
}
```

Контейнер backend работает от UID 1000 (appuser).
Без этого он не сможет писать логи и обновлять ENCRYPTION_KEY в .env. Дайте соответствующие права:

```
chown -R 1000:1000 env_config logs storage plugins
```

Итоговый набор папок и файлов

```
root@umni-ecosystem:/opt/umni-ecosystem# tree -fh
[4.0K]  .
|-- [1.9K]  ./docker-compose.release.yml
|-- [4.0K]  ./env_config
|-- [4.0K]  ./logs
|-- [1.3K]  ./nginx.conf
|-- [4.0K]  ./plugins
|   `-- [4.0K]  ./plugins/custom
`-- [4.0K]  ./storage
```

Запуск

```
docker compose -f docker-compose.release.yml pull      # подтянуть образы (рекомендуется)
docker compose -f docker-compose.release.yml up -d
docker compose -f docker-compose.release.yml ps        # все три контейнера Healthy
```

Перейдите по адресу http://IP_ВАШЕГО_LXC/#/

Установка UNI Server на Ubuntu 26.04

install-umni-ubuntu.sh