Инструкция по установке экземпляра программного обеспечения

РУБИН-Т

Москва, 2026

Термины и определения

Docker – набор продуктов, использующих виртуализацию на уровне операционной системы для доставки программного обеспечения в пакетах, называемых контейнерами.

Docker Compose – инструмент для определения и запуска многоконтейнерных приложений Docker с использованием одного конфигурационного файла YAML.

Harbor Registry – реестр для хранения Docker-образов с открытым исходным кодом, который предоставляет доступ к образам с помощью политик, а также умеет сканировать образы на наличие уязвимостей.

Ubuntu – дистрибутив GNU/Linux, основанный на Debian.

Содержание

1 Системные требования для работы ПО РУБИН-Т

Требования зависят от размеров и наполнения данными, с которыми будет работать ПО РУБИН-Т.

Таблица 1 – Минимальные системные требования

	Рабочие места пользователей	***Серверное оборудование***
Операционная система (ОС)	ОС, поддерживающие современные браузеры	Ubuntu 24.04
Процессор	2-х ядерный	4-х ядерный с тактовой частотой от 2.4 Гц
Оперативная память	от 4 Гб	от 32 Гб
Жесткий диск	от 50 Гб	от 1 Тб SSD3
Веб-браузер	Яндекс.Браузер, Google Chrome, Mozilla Firefox, Microsoft Edge (версии не старше одного года)	-

2 Установка ПО РУБИН-Т

При использовании ПО РУБИН-Т бесплатно или по подписке установка не требуется. Разработчик ПО предоставляет пользователям доступ по логину и паролю после регистрации.

Установка ПО РУБИН-Т осуществляется только при коммерческой установке программного обеспечения на сервере заказчика.

Программное обеспечение РУБИН-Т распространяется в виде контейнеров Docker.

3 Разворачивание ПО РУБИН-Т на сервер

Развертывание выполняется администратором системы на сервере под управлением операционной системы Ubuntu.

Для установки программного обеспечения необходимо выполнить действия, описанные в пп. 3.1 – 3.3.

3.1 Предварительные требования

Перед началом установки на сервере должны быть установлены:

- Docker;
- Docker Compose.

Необходим доступ к Harbor Registry, из которого будут загружаться контейнерные образы.

3.2 Авторизация в Harbor Registry

Перед загрузкой образов необходимо выполнить авторизацию в Harbor с помощью команды: (пароль необходимо запросить и подставить)

```
echo "PASSWORD_HERE" | docker login harbor.zone-it.dev -u
robot\$registration --password-stdin
```

3.3 Подготовка файлов

На сервере создать файл конфигурации `docker-compose.yml` со следующим содержимым:

```
x-rubin-db-env: &rubin-db-env
  POSTGRES_DB: rubin_t
  POSTGRES_USER: rubin_t
  POSTGRES_PASSWORD: rubin_t

x-rt-staging-env: &rt-staging-env
  COOKIE_SECURE: false
  ADMIN_PASSWORD: password
  RAILS_MASTER_KEY: 2351d67ec6d6ef38aa985c82f8c30c23
  RAILS_ENV: staging
  DATABASE_URL: postgis://rubin_t:rubin_t@database:5432/rubin_t
  RABBITMQ_URL: amqp://rabbitmq:rabbitmq@rabbitmq:5672/app
  TRACCAR_API_BASE: http://traccar:9999
  TRACCAR_USERNAME: admin
  TRACCAR_PASSWORD: admin
  OSM_SSO_SECRET: change_me_osm_sso_secret

x-rt-staging-common: &rt-staging-common
  image: harbor.zone-it.dev/rubin-t/sl-api:main
  restart: always
  environment:
    <<: *rt-staging-env

x-osm-env: &osm-env
  PIDFILE: /tmp/pids/server.pid
  RAILS_ENV: production
  DATABASE_URL: postgresql://osm:osm@database-osm:5432/osm
  OSM_SSO_SECRET: change_me_osm_sso_secret
  RAILS_RELATIVE_URL_ROOT: /osm

x-osm-common: &osm-common
  image: harbor.zone-it.dev/rubin-t/openstreetmap-website:zone-it-master
  restart: always
  stdin_open: true
  tty: true
  environment:
    <<: *osm-env
  tmpfs:
    - /tmp/pids

services:
```

```
  database-traccar:
    image: harbor.zone-it.dev/rubin-t/timescale-pg16:main
    restart: always
    container_name: db_traccar
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U traccar -d traccar"]
      interval: 5s
      timeout: 5s
      retries: 20
      start_period: 30s
    environment:
      POSTGRES_DB: traccar
      POSTGRES_USER: traccar
      POSTGRES_PASSWORD: traccar
      TIMESCALEDB_TELEMETRY: "off"

  database-osm:
    image: harbor.zone-it.dev/rubin-t/timescale-pg16:main
    restart: always
    container_name: db_osm
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U osm -d osm"]
      interval: 5s
      timeout: 5s
      retries: 20
      start_period: 30s
    environment:
      POSTGRES_DB: osm
      POSTGRES_USER: osm
      POSTGRES_PASSWORD: osm
      TIMESCALEDB_TELEMETRY: "off"

  database:
    image: harbor.zone-it.dev/rubin-t/postgis:16-master
    restart: always
    container_name: db
    environment:
      <<: *rubin-db-env
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U rubin_t -d rubin_t"]
      interval: 20s
      timeout: 5s
      retries: 20
      start_period: 30s

  rabbitmq:
    image: harbor.zone-it.dev/rubin-t/rabbitmq:main
    container_name: rabbitmq
    restart: always
    command: >
      sh -c "rabbitmq-plugins enable --offline rabbitmq_management
rabbitmq_mqtt &&
      rabbitmq-server"
    environment:
      RABBITMQ_DEFAULT_USER: rabbitmq
      RABBITMQ_DEFAULT_PASS: rabbitmq
      RABBITMQ_DEFAULT_VHOST: app

  rt-migrate:
    image: harbor.zone-it.dev/rubin-t/sl-api:main
```

```
    container_name: rt-migrate
    restart: "no"
    depends_on:
      database:
        condition: service_healthy
      rabbitmq:
        condition: service_started
    environment:
      <<: *rt-staging-env
    command: >
      sh -c "
        until pg_isready -h database -p 5432 -U rubin_t -d rubin_t; do
          echo 'Waiting for rt database...';
          sleep 2;
        done;
        bundle exec rails db:prepare
      "

  osm-migrate:
    image: harbor.zone-it.dev/rubin-t/openstreetmap-website:zone-it-master
    container_name: osm-migrate
    restart: "no"
    depends_on:
      database-osm:
        condition: service_healthy
    environment:
      <<: *osm-env
    tmpfs:
      - /tmp/pids
    command: >
      sh -c "
        until pg_isready -h database-osm -p 5432 -U osm -d osm; do
          echo 'Waiting for osm database...';
          sleep 2;
        done;
        bundle exec rails db:prepare
      "

  traccar:
    image: harbor.zone-it.dev/rubin-t/traccar:main
    restart: always
    depends_on:
      database-traccar:
        condition: service_healthy
    environment:
      CONFIG_USE_ENVIRONMENT_VARIABLES: "true"
      DATABASE_DRIVER: org.postgresql.Driver
      DATABASE_URL: jdbc:postgresql://database-traccar:5432/traccar
      DATABASE_USER: traccar
      DATABASE_PASSWORD: traccar
    healthcheck:
      test: ["CMD", "wget", "-q", "--spider",
"http://localhost:8082/api/health"]
      interval: 2m
      timeout: 5s
      start_period: 1h
      retries: 3

  osm-web:
    <<: *osm-common
```

```
  container_name: osm-web
  depends_on:
    osm-migrate:
      condition: service_completed_successfully
    database-osm:
      condition: service_healthy
  ports:
    - "3000:3000"
  command: bundle exec rails s -p 3000 -b 0.0.0.0

frigate:
  image: harbor.zone-it.dev/rubin-t/frigate:stable
  container_name: frigate
  restart: always
  volumes:
    - frigate_media:/media/frigate
    - frigate_config:/config
  environment:
    FRIGATE_RTSP_USER: admin
    FRIGATE_RTSP_PASSWORD: admin
    FRIGATE_MQTT_USER: rabbitmq
    FRIGATE_MQTT_PASSWORD: rabbitmq

rt-rails:
  <<: *rt-staging-common
  container_name: rt-rails
  depends_on:
    rt-migrate:
      condition: service_completed_successfully
    database:
      condition: service_healthy
    rabbitmq:
      condition: service_started
  command: bundle exec rails server -b 0.0.0.0 -p 3000
  healthcheck:
    test: ["CMD", "curl", "-f", "http://localhost:3000/up"]
    interval: 30s
    timeout: 10s
    retries: 10
    start_period: 20s

rt-worker:
  <<: *rt-staging-common
  container_name: rt-worker
  depends_on:
    rt-migrate:
      condition: service_completed_successfully
    database:
      condition: service_healthy
    rabbitmq:
      condition: service_started
  command: bundle exec rails solid_queue:start

rt-position-consume:
  <<: *rt-staging-common
  container_name: rt-position-consume
  depends_on:
    rt-migrate:
      condition: service_completed_successfully
    database:
```

```
      condition: service_healthy
    rabbitmq:
      condition: service_started
  command: bundle exec rake rabbitmq:consume:positions

rt-position-cache-consume:
  <<: *rt-staging-common
  container_name: rt-position-cache-consume
  depends_on:
    rt-migrate:
      condition: service_completed_successfully
    database:
      condition: service_healthy
    rabbitmq:
      condition: service_started
  command: bundle exec rake rabbitmq:consume:position_cache

rt-event-consume:
  <<: *rt-staging-common
  container_name: rt-event-consume
  depends_on:
    rt-migrate:
      condition: service_completed_successfully
    database:
      condition: service_healthy
    rabbitmq:
      condition: service_started
  command: bundle exec rake rabbitmq:consume:events

rt-frigate-consume:
  <<: *rt-staging-common
  container_name: rt-frigate-consume
  depends_on:
    rt-migrate:
      condition: service_completed_successfully
    database:
      condition: service_healthy
    rabbitmq:
      condition: service_started
  command: bundle exec rake rabbitmq:consume:frigate

swagger:
  image: harbor.zone-it.dev/rubin-t/swagger-ui:main
  container_name: swagger
  restart: always
  environment:
    SWAGGER_JSON_URL: https://app.swiftlogic.dev/api/openapi.json
    BASE_URL: /swagger

rt-web:
  image: harbor.zone-it.dev/rubin-t/sl-web:main
  container_name: rt-web
  restart: always
  depends_on:
    rt-rails:
      condition: service_healthy
    osm-web:
      condition: service_started
  environment:
    RAILS_APP: rt-rails:3000
```

```
      FRIGATE_BACKEND: frigate:5000
      RABBITMQ_BACKEND: rabbitmq:15672
      TRACCAR_BACKEND: traccar:9999
      GO2RTC_BACKEND: frigate:1984
      SWAGGER_BACKEND: swagger:8080
      OSM_BACKEND: osm-web:3000
    ports:
      - "8080:80"

volumes:
  frigate_config:

  frigate_media:
```

4 Запуск программного обеспечения

Для запуска программного обеспечения необходимо перейти в каталог, содержащий файл `docker-compose.yml`, и выполнить команду *docker compose up -d*.

Команда:

- загрузит требуемые образы из Harbor Registry;
- создаст и запустит контейнеры в фоновом режиме.

5 Запуск веб-интерфейса

1. В браузере ввести *http://ip:8080*, где *ip* – IP адреса сервера.
2. В появившемся окне нажать **Продолжить с почтой**.
3. Ввести логин и пароль (*admin@zone-it.studio* и *password*).

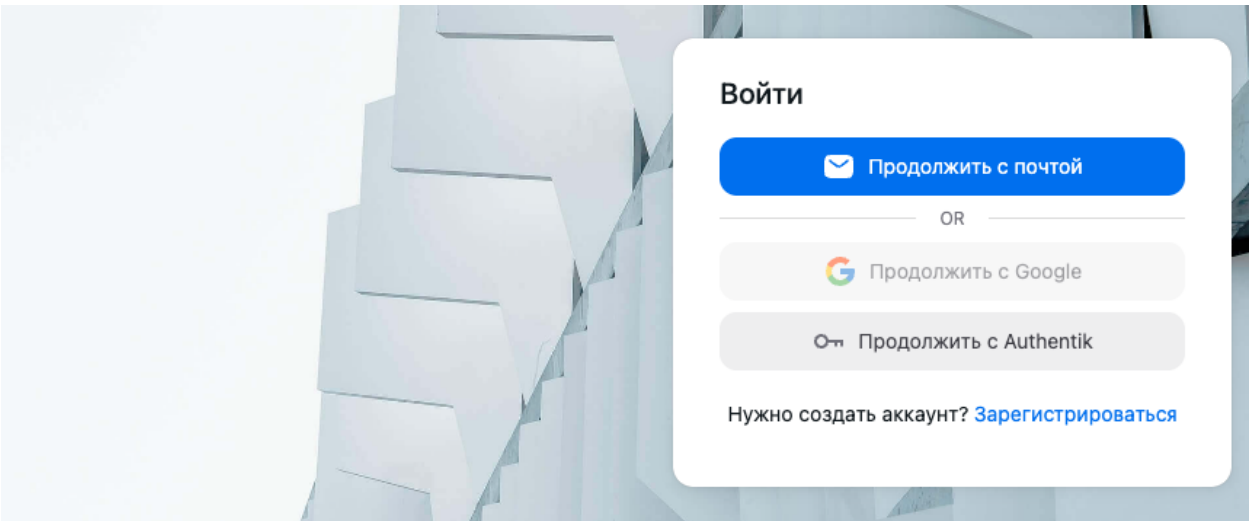

Рисунок 1 – Окно входа в ПО РУБИН-Т