

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «Б5ГИС»

**ИНСТРУКЦИЯ ПО УСТАНОВКЕ ЭКЗЕМПЛЯРА ПРОГРАММНОГО
ОБЕСПЕЧЕНИЯ, ПРЕДОСТАВЛЕННОГО ДЛЯ ПРОВЕДЕНИЯ
ЭКСПЕРТНОЙ ПРОВЕРКИ**

Москва 2024

СОДЕРЖАНИЕ

1. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА.....	3
2. РАЗВОРАЧИВАНИЕ СИСТЕМЫ	4

1. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Технические средства Б5ГИС разделены на 3 структурных части: сервер приложений, Сервер ГИС, сервер БД.

В следующей таблице представлены стандартный комплекс технических средств для стабильного функционирования программного обеспечения:

№	Сервер	Роль	CPU	RAM	SSD\HDD
1	VM_APP	Сервер приложений	8	16GB	SSD: 600GB
2	VM_BD	Сервер БД	16	64GB	SSD: 4TB, HDD: 2TB
3	VM_GIS	Сервер ГИС	16	48GB	SSD: 4TB

2. РАЗВОРАЧИВАНИЕ СИСТЕМЫ

Установка службы Docker

```
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

Проверить, что служба запущена:

```
sudo systemctl status docker
```

Во избежание конфликтов в сети рекомендуется создать Docker сеть, отличную от текущих сетей.

```
sudo docker network create project-network --subnet=10.19.91.0/24
```

Имя сети project-network в дальнейшем будет задаваться в файле docker-compose.yml для каждого сервиса.

Установка службы Nginx

```
sudo apt-get install nginx
```

Проверить, что служба запущена:

```
sudo systemctl status nginx
```

Установка утилит PostgreSQL 16

```
sudo sh -c 'echo "deb https://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-
pgdg main" > /etc/apt/sources.list.d/pgdg.list'
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key
add -
```

```
sudo apt-get update
sudo apt-get -y install postgresql-client-16
```

Настройка и запуск Docker контейнеров

Redis Service

Распаковать архив с сервисом.

```
sudo tar xf redis.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd redis-docker
```

Отредактировать docker-compose.yml:

```
version: '3'
services:
  redis:
    image: redis:latest
    ports:
      - '16379:6379'
    volumes:
      - ./data:/var/lib/redis/data
      - ./config/redis.conf:/usr/local/etc/redis/redis.conf
    command: redis-server --requirepass "<Установить_пароль>"
    restart: always
networks:
  default:
    name: "project-network"
    external: true
```

Задать пароль для подключения к БД Redis.

Загрузить образ из публичного Docker репозитория.

```
sudo docker compose pull
```

Запустить Docker контейнер

```
sudo docker compose up -d
```

Проверить, что контейнер запущен

```
sudo docker compose ps
```

PostgreSQL 16

Распаковать архив с БД.

```
sudo tar xf pgsql-16.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd pgsql-16
```

Отредактировать docker-compose.yml:

```
version: '3'
services:
  pgsql-16:
    image: registry.oxity.ru:5000/svc/pgsql-16-ubuntu
    build: ./postgres
    user: postgres
    container_name: pgsql-16
    command: /usr/lib/postgresql/16/bin/postgres -D /var/lib/pgsql/16/main
    shm_size: '4gb'
    volumes:
      - ./postgres/initdb.d:/docker-entrypoint-initdb.d:ro
      - ./data/pgdata:/var/lib/pgsql/16/main
      - ./data/backups:/var/lib/pgsql/16/backups
      - /etc/localtime:/etc/localtime:ro
      - ../files/files:/home/files
    ports:
      - "65432:5432"
    restart: always
networks:
  default:
    name: "project-network"
    external: true
```

Предположим, что рабочий каталог с данными PostgreSQL должен находиться на отдельном диске.

В этом случае, необходимо перенести директорию с данными **pgdata** и отредактировать путь в блоке **volumes**. Как пример: - /mnt/pgdata:/var/lib/pgsql/16/main **Важно.** БД должна видеть рабочую директорию сервиса **files**, если сервис расположен на другом узле, необходимо подключить каталог, как пример, через **smb/sshfs** и т.п. и изменить соответствующий путь в блоке **volumes**. Как пример, рабочая папка монтирована в директорию **/mnt/files**, значит путь необходимо задать как - **/mnt/files:/home/files**.

Установить ID 1001 пользователя и группы для рабочего каталога PostgreSQL **pgdata**.

```
sudo chown 1001:1001 -R /путь/до/директории/pgdata
```

Загрузить образ из Docker репозитория registry.oxity.ru:5000.

```
sudo docker compose pull
```

Запустить Docker контейнер

```
sudo docker compose up -d
```

Проверить, что контейнер запущен

```
sudo docker compose ps
```

ClickHouse Server

Распаковать архив с БД ClickHouse Server.

```
sudo tar xf clickhouse.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd clickhouse-docker
```

Отредактировать docker-compose.yml:

```
version: "3"
services:
  clickhouse-server:
    image: yandex/clickhouse-server
    ports:
      - "8123:8123"
      - "9000:9000"
    volumes:
      - ./clickhouse-config:/etc/clickhouse-server
      - ./clickhouse-data:/var/lib/clickhouse
      - /etc/localtime:/etc/localtime:ro
    environment:
      TZ: "Europe/Moscow"
    restart: always
networks:
  default:
```

```
name: "project-network"  
external: true
```

Предположим, что рабочий каталог с данными ClickHouse должен находиться на отдельном диске.

В этом случае, необходимо перенести директорию с данными **clickhouse-data** в нужное место и отредактировать путь в блоке **volumes**. Как пример: - /mnt/clickhouse-data:/var/lib/clickhouse

Загрузить образ из публичного Docker репозитория.

```
sudo docker compose pull
```

Запустить Docker контейнер.

```
sudo docker compose up -d
```

Проверить, что контейнер запущен.

```
sudo docker compose ps
```

API Service

Распаковать архив с сервисом API.

```
sudo tar xf api_service.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd api_service
```

Отредактировать docker-compose.yml.

```
version: '3.2'  
services:  
  api_service:  
    container_name: api_service  
    build:  
      context: .  
      dockerfile: ./docker/Dockerfile  
    image: registry.oxity.ru:5000/svc/api_service  
    ports:  
      - '8104:80'  
      - '8106:80'  
      - '3001:80'
```



```

restart: always
volumes:
  - ./nginx:/usr/local/openresty/nginx/conf
  - ./core:/core
  - ./logs:/usr/local/openresty/nginx/logs
  - /etc/localtime:/etc/localtime:ro
  - ./envs.lua:/envs.lua
  - ./oauth2-proxy/configs/oauth2-proxy-alpha-config.yaml:/oauth2-proxy-alpha-
config.yaml:ro

oauth2-proxy:
  container_name: oauth2-proxy
  build:
    context: oauth2-proxy/oauth2-proxy
    dockerfile: ./Dockerfile
  image: registry.oxity.ru:5000/svc/oauth2-proxy
  restart: always
  ports:
    - '3000:3000'
  command:
    --skip-provider-button
    --alpha-config /oauth2-proxy-alpha-config.yaml
    --cookie-httponly=false
    --cookie-name=_oauth2_proxy_cp
    --cookie-refresh=1m
    --email-domain=*
    --cookie-secret=<Задаётся_секрет_куки_случайным_значением>
    --cookie-secure=false
    --session-store-type=redis
    --redis-connection-
url=redis://<задат_IP_адрес_к_БД_redis>:<задать_порт_к_БД_redis>
    --redis-password='задать_пароль_к_БД_redis'
  volumes:
    - ./oauth2-proxy/configs/oauth2-proxy-alpha-config.yaml:/oauth2-proxy-alpha-
config.yaml:ro

networks:
  default:
    name: "rusgis-network"
    external: true

```

oauth2-proxy сервис используется для интеграции с системами аутентификации и авторизации, использующих технологии OAuth2 или OpenID Connect для аутентификации пользователя.

Если интеграция не используется, все значения остаются по умолчанию. Если используется интеграция, необходимо задать значения: --cookie-secret – задать значение секрет Cookie, можно установить любое произвольное значение.

--redis-connection-url – задать адрес и порт подключения к БД Redis. Значение задаётся в виде ссылки redis://127.0.0.1:6379

--redis-password – задать пароль для подключения к БД Redis

А также отредактировать файл конфигурации сервиса, расположенный по следующему пути ./api_service/oauth2-proxy/configs/oauth2-proxy-alpha-config.yaml.

Как пример, конфигурация, где в качестве провайдера используется KeyCloak:

```

- provider: keycloak-oidc
  name: keycloak

```

```

label: Keycloak
icon: C5385971-A43A-4750-BC1F-2FD2000C022C
id: 4
code_challenge_method: S256
clientSecret: <задать_значение_установленное_провайдером>
clientId: <задать_значения_установленное_провайдером>
scope: openid email profile roles
apiConfig:
  f_create_user: config.pr_user_keycloak
oidcConfig:
  issuerURL: <задать_URL_провайдера_идентификации>
  insecureAllowUnverifiedEmail: true
  emailClaim: email
  userIDClaim: email
  audienceClaims:
    - aud

```

clientSecret должен быть создан и предоставлен администратором при регистрации и настройке клиентского приложения или сервиса в провайдере идентификации. В зависимости от провайдера идентификации это может быть сгенерированный секретный ключ или пароль.

issuerURL указывает на URL-адрес провайдера идентификации, который должен быть предоставлен администратором.

clientId значение переменной выпускается или настраивается администратором системы. Как правило, оно предоставляется провайдером идентификации при регистрации и настройке клиентского приложения или сервиса.

Отредактировать файл **envs.lua**.

```

local Envs = {
  SESSION_TIME = 1440, -- Время сессии в минутах
  TIMEOUT_DB = 99999999, -- Время запроса в БД в миллисекундах
  TIMEOUT_REQUEST = 600, -- Время запроса в секундах
  REDIS_KEYS_TTL = 172800, -- Время жизни ключей редиса в секундах
  SESSION_EXPIRE = true, -- Истечение срока действия сессии по времени
  REDIS_KEYS_EXPIRE = true, -- Истечение срока действия ключей редиса по времени
  connect = {
    POSTGRES_CONNECTIONS = {
      name = "БД connections",
      str = {
        host = "<задать_IP_адрес_БД>",
        port = "<задать_Порт_БД>",
        database = "connections",
        user = "<задать_пользователя_БД>",
        password = "<задать_пароль_БД>"
      }
    },
  },
  CONFIG_DSCONFIG = {
    name = "БД config_19",
    str = {
      host = ""<задать_IP_адрес_БД>",
      port = ""<задать_Порт_БД>",
      database = "config_19",
      user = "<задать_пользователя_БД>",
      password = "<задать_пароль_БД>"
    }
  },
},

```

```

    REDIS = {
        name = "Redis",
        str = {
            host = "<задать_адрес_подключения_к_БД_Redis>",
            port = <задать_порт_подключения_к_БД_Redis>,
            pass = "<задать_пароль_для_подключения_к_БД_Redis>"
        },
    },
}

function Envs:is(code_system)
    return Envs.connect[string.upper(code_system)] ~= nil
end
function Envs:get(code_system)
    return Envs.connect[string.upper(code_system)].str
end
function Envs:getName(code_system)
    return Envs.connect[string.upper(code_system)].name
end
return Envs

```

Задать значения подключения для каждого блока БД:

host – адрес подключения к БД

port – порт подключения к БД

user – пользователь БД

password – пароль пользователя БД

Загрузить образ из Docker репозитория registry.oxity.ru:5000.

```
sudo docker compose pull
```

Запустить Docker контейнер.

```
sudo docker compose up -d
```

Проверить, что контейнер запущен.

```
sudo docker compose ps
```

Сервис **TMS/Composite**

Распаковать архив с сервисом TMS.

```
sudo tar xf composite.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd composite
```

Отредактировать envs.lua.

```

local Envs = {
  SESSION_TIME = 1440, -- Время сессии в минутах
  TIMEOUT_DB = 600000, -- Время запроса в БД в миллисекундах
  TIMEOUT_REQUEST = 600, -- Время запроса в секундах
  REDIS_KEYS_TTL = 172800, -- Время жизни ключей редиса в секундах
  SESSION_EXPIRE = true, -- Истечение срока действия сессии по времени
  REDIS_KEYS_EXPIRE = false, -- Истечение срока действия ключей редиса по времени
  connect = {
    POSTGRES_CONNECTIONS = {
      name = "Доступ база",
      str = {
        host = "<задать_IP_адрес_БД>",
        port = "<задать_Порт_БД>",
        database = "connections",
        user = "<задать_пользователя_БД>",
        password = "<задать_пароль_БД>"
      }
    },
    POSTGRES_DB_NAME = {
      name = "POSTGRES_DB_NAME",
      str = {
        host = "<задать_IP_адрес_БД>",
        port = "<задать_Порт_БД>",
        database = "POSTGRES_DB_NAME",
        user = "<задать_пользователя_БД>",
        password = "<задать_пароль_БД>"
      }
    },
    REDIS = {
      name = "Redis",
      str = {
        host = "<задать_адрес_подключения_к_БД_Redis>",
        port = "<задать_порт_подключения_к_БД_Redis>",
        pass = "<задать_пароль_для_подключения_к_БД_Redis>"
      }
    },
  },
}

function Envs:is(code_system)
  return Envs.connect[string.upper(code_system)] ~= nil
end
function Envs:get(code_system)
  return Envs.connect[string.upper(code_system)].str
end
function Envs:getName(code_system)
  return Envs.connect[string.upper(code_system)].name
end
return Envs

```

Задать значения подключения для каждого блока БД, где POSTGRES_DB_NAME блок с названием БД PostgreSQL.

host – адрес подключения к БД

port – порт подключения к БД

user – пользователь БД

password – пароль пользователя БД

Загрузить образ из Docker репозитория registry.oxity.ru:5000.

```
sudo docker compose pull
```

Запустить Docker контейнер.

```
sudo docker compose up -d
```

Проверить, что контейнер запущен.

```
sudo docker compose ps
```

Сервис Файлов/Files

Распаковать архив с сервисом Файлов.

```
sudo tar xf files.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd files
```

Отредактировать docker-compose.yml.

```
version: '2'
services:
  files:
    container_name: f1
    build: ./docker
    image: files
    ports:
      - '8102:8102'
    volumes:
      - ./nginx:/usr/local/openresty/nginx/conf
      - ./core:/core
      - ./logs:/usr/local/openresty/nginx/logs
      - ./files:/usr/local/openresty/nginx/files
      - ./libs:/usr/local/libs
      - /etc/localtime:/etc/localtime:ro
      - ./envs.lua:/envs.lua
    restart: always
networks:
  default:
    name: "project-network"
    external: true
```

Предположим, что рабочий каталог **files** с данными сервиса Файлов должен находиться на отдельном диске, так как загруженные файлы в информационную систему будут расположены в этой директории, а значит, объём свободного пространства должен быть достаточным.

В этом случае, необходимо перенести/создать директорию **files** в нужное место и отредактировать путь в блоке **volumes**. Как пример: - /mnt/files:/usr/local/openresty/nginx/files.

Отредактировать envs.lua.

```
local Envs = {
  SESSION_TIME = 1440, -- Время сессии в минутах
  TIMEOUT_DB = 99999999, -- Время запроса в БД в миллисекундах
  TIMEOUT_REQUEST = 600, -- Время запроса в секундах
  REDIS_KEYS_TTL = 172800, -- Время жизни ключей редиса в секундах
  SESSION_EXPIRE = true, -- Истечение срока действия сессии по времени
  REDIS_KEYS_EXPIRE = true, -- Истечение срока действия ключей редиса по времени
  connect = {
    POSTGRES_DB_NAME = {
      name = "POSTGRES_DB_NAME",
      str = {
        host = "<задать_IP_адрес_БД>",
        port = "<задать_Порт_БД>",
        database = "POSTGRES_DB_NAME",
        user = "<задать_пользователя_БД>",
        password = "<задать_пароль_БД>"
      }
    },
  },
  REDIS = {
    name = "Redis",
    str = {
      host = "<задать_адрес_подключения_к_БД_Redis>",
      port = "<задать_порт_подключения_к_БД_Redis>",
      pass = "<задать_пароль_для_подключения_к_БД_Redis>"
    }
  },
  API_SERVICE = {
    name = "Данные к DATASET'у",
    str = {
      host = "<задать_адрес_сервиса_API>",
      port = "<задать_порт_сервиса_API>",
      token = ''
    }
  },
  EFISZSN = {
    name = "Данные для подключения к ЕФИС ЗСН",
    str = {
      url = "<задать_ссылку_ЕФИС_ЗСН>",
      params = '{"email": "<задать_email_адрес>", "password": "<задать_пароль>"}'
    }
  },
  STIMULSOFT = {
    name = "Данные для взаимодействия со стимулсофт",
    str = {
      url = "http://<задать_адрес>:<задать_порт>"
    }
  },
}
```

```

    path = "/usr/local/openresty/nginx/files/4", -- Путь к файлам
}
function Envs:is(code_system)
    return Envs.connect[string.upper(code_system)] ~= nil
end
function Envs:get(code_system)
    return Envs.connect[string.upper(code_system)].str
end
function Envs:getName(code_system)
    return Envs.connect[string.upper(code_system)].name
end
return Envs

```

Задать значения подключения для каждого блока БД, где POSTGRES_DB_NAME:

Блок с названием БД PostgreSQL.

host – адрес подключения к БД

port – порт подключения к БД

user – пользователь БД

password – пароль пользователя БД

Блок API_SERVICE, задаются данные для обращения к сервису API

host – адрес сервиса API

port – порт сервиса API

token – токен сервиса API, по умолчанию отсутствует.

Блок EFISZSN, задаются данные для подключения к сервису ЕФИС, если данное подключение используется.

Блок STEMULSOFT, задаются данные для обращения к сервису отчетов, используется http-ссылка, где задаётся адрес и порт сервиса, как пример: http://127.0.0.1:8102

Загрузить образ из Docker репозитория registry.oxity.ru:5000.

```

sudo docker compose pull

```

Запустить Docker контейнер.

```

sudo docker compose up -d

```

Проверить, что контейнер запущен.

```

sudo docker compose ps

```

Сервис очереди RabbitMQ

Распаковать архив с сервисом RabbitMQ.

```
sudo tar xf rabbitmq.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd rabbitmq
```

Отредактировать .env файл

```
RABBITMQ_ERLANG_COOKIE=<задать_значение_cookie>  
RABBITMQ_DEFAULT_USER=<задать_пользователя>  
RABBITMQ_DEFAULT_PASS=<задать_пароль>  
RABBITMQ_DEFAULT_VHOST=/  

```

RABBITMQ_ERLANG_COOKIE – задать значение Cookie, можно установить любое произвольное значение.

RABBITMQ_DEFAULT_USER – задать пользователя сервиса.

RABBITMQ_DEFAULT_PASS – задать пароль пользователя сервиса.

Загрузить образ из публичного Docker репозитория.

```
sudo docker compose pull
```

Запустить Docker контейнер.

```
sudo docker compose up -d
```

Проверить, что контейнер запущен.

```
sudo docker compose ps
```

Сервис разбора очереди Worker

Распаковать архив с сервисом Worker.

```
sudo tar xf worker.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd worker
```


Отредактировать docker-compose.yml.

```
version: '3.7'
services:
  worker:
    container_name: worker
    build:
      context: .
      dockerfile: ./docker/Dockerfile
    image: registry.oxity.ru:5000/svc/worker
    restart: always
    volumes:
      - ./configs/supervisord:/etc/supervisor/conf.d
      - ./app:/app
      - ./logs:/var/log/main_log
    environment:
      db_user: "<задать_пользователь_БД>"
      db_password: "<задать_пароль_пользователя_БД>"
      db_host: "<задать_адрес_БД>"
      db_port: <задать_порт_БД>
      db_name: "bpm"
      mq_host: "<адрес_сервиса_rabbitmq>"
      mq_port: <задать_порт_сервиса_rabbitmq>
      mq_user: "<пользователь_сервиса_rabbitmq>"
      mq_password: "<пароль_сервиса_rabbitmq>"
      mq_queues_names: "queue_timer,queue_bpmn,queue_launch"
      mq_vhost: "/"
      mq_prefetch_count: 20
      time_sleep: 30
      load_percentage: 60
networks:
  default:
    name: "project-network"
    external: true
```

Загрузить образ из Docker репозитория registry.oxity.ru:5000.

```
sudo docker compose pull
```

Запустить Docker контейнер.

```
sudo docker compose up -d
```

Проверить, что контейнер запущен.

```
sudo docker compose ps
```

Сервис логов upload_logs_to_db

Распаковать архив с сервисом upload_logs_to_db.

```
sudo tar xf upload_logs_to_db.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd upload_logs_to_db
```

Отредактировать docker-compose.yml.

```
version: '3.2'
services:
  upload_logs_to_db:
    container_name: upload_logs_to_db
    build:
      context: .
      dockerfile: ./docker/Dockerfile
    image: registry.oxity.ru:5000/svc/upload_logs_to_db
    restart: always
    volumes:
      - ./logs:/var/log/main_log
      - ./configs/supervisord:/etc/supervisor/conf.d
      - ./app:/app
      - /opt/rusgis/services/api_service/logs:/app/dir_for_pars/api_service
      - /opt/rusgis/services/composite/logs:/app/dir_for_pars/composite
      - /opt/rusgis/services/files/logs:/app/dir_for_pars/files
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    environment:
      config_host: "<задать_адрес_БД_PostgreSQL_config_19>"
      config_port: <задать_порт_БД_PostgreSQL_config_19>
      config_user: "<задать_пользователя_БД_PostgreSQL>"
      config_password: "<задать_пароль_пользователя_БД_PostgreSQL>"
      config_name: "config_19"
      timeout: 60
      host: "<задать_адрес_БД_ClickHouse>"
      port: "<задать_порт_БД_ClickHouse>"
      max_threads: "10"
      dbname: "METRICS"
      table_name: "logs"
      user: "<задать_пользователя_БД_ClickHouse>"
      password: "<задать_пароль_пользователя_БД_ClickHouse>"
      redis_host: "<задать_адрес_БД_Redis>"
      redis_port: <задать_порт_БД_Redis>
      redis_pass: "<задать_пароль_БД_Redis>"
networks:
  default:
    name: "project-network"
    external: true
```

config_host – адрес подключения к БД PostgreSQL где развернута база config_19
config_port – порт подключения к БД PostgreSQL где развернута база config_19
config_user – пользователя БД PostgreSQL где развернута база config_19
config_password – пароль БД PostgreSQL где развернута база config_19
host – адрес подключения к БД ClickHouse
port – порт подключения к БД ClickHouse
user – пользователь БД ClickHouse
password – пароль пользователя БД ClickHouse
redis_host – адрес подключения БД Redis
redis_port – порт подключения БД Redis
redis_pass – пароль подключения БД Redis

Загрузить образ из Docker репозитория registry.oxity.ru:5000.

```
sudo docker compose pull
```

Запустить Docker контейнер.

```
sudo docker compose up -d
```

Проверить, что контейнер запущен.

```
sudo docker compose ps
```

Сервис отображения растров mapserver

Распаковать архив с сервисом mapserver.

```
sudo tar xf mapserver.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd mapserver
```

Отредактировать docker-compose.yml.

```
version: '3.7'

services:
  tilerserver:
    container_name: tilerserver
    build:
      context: .
```

```

    dockerfile: ./docker/Dockerfile
    image: registry.oxity.ru:5000/svc/tilerserver
    command: gunicorn main:app --workers 4 --worker-class
uvicorn.workers.UvicornWorker --bind 0.0.0.0:8077
    restart: always
    volumes:
      - ./app:/app
      - ./app/tiles:/app/tiles
      - ./app/tiles2:/app/tiles2
      - ../files:/app/files:rw
    ports:
      - '8077:8077'

networks:
  default:
    name: "project-network"
    external: true

```

Предположим, что рабочий каталог **tiles** и **tiles2** с данными сервиса растров должен находиться на отдельном диске, так как загруженные файлы растров в информационную систему будут расположены в этой директории, а значит, объём свободного пространства должен быть достаточным.

В этом случае, необходимо перенести/создать директорию **tiles** и **tiles2** в нужное место и отредактировать путь в блоке **volumes**. Как пример: - /mnt/tiles:/app/tiles и соответственно для директории **tiles2**.

Важно. Сервис должен видеть директорию с данными сервиса **files**, если сервис расположен на другом узле, необходимо подключить каталог, как пример, через **smb/sshfs** и т.п. и изменить соответствующий путь в блоке **volumes**. Как пример, рабочая папка смонтирована в директорию /mnt/files, значит, путь необходимо задать как - /mnt/files:/app/files.

Загрузить образ из Docker репозитория **registry.oxity.ru:5000**.

```

sudo docker compose pull

```

Запустить Docker контейнер.

```

sudo docker compose up -d

```

Проверить, что контейнер запущен.

```

sudo docker compose ps

```

Сервис работы с растровыми слоями **bands_service**

Распаковать архив с сервисом **bands_service**.

```

sudo tar xf bands_service.tar.gz

```

Перейти в директорию распакованного сервиса.

```
cd bands_service
```

Отредактировать docker-compose.yml.

```
version: '3.7'
services:
  bands_service:
    container_name: bands_service
    build:
      context: .
      dockerfile: ./docker/Dockerfile
    image: registry.oxity.ru:5000/svc/bands_service
    command: uvicorn main:app --host 0.0.0.0 --port 8074
    restart: always
    volumes:
      - ./app:/app
      - ../mapserver/app/tiles:/app/tiles
      - ../mapserver/app/tiles2:/app/tiles2
      - ../files/files:/app/files:rw
    ports:
      - '8074:8074'
    environment:
      USER_PSQL: "<задать_пользователя_БД>"
      PASSWORD_PSQL: "<задать_пароль_БД>"
      HOST_PSQL_PAGES: "<задать_адрес_подключения_БД_PostgreSQL_pages>"
      PORT_PSQL_PAGES: 65432
      DATABASE_PSQL_PAGES: "pages"

networks:
  default:
    name: "project-network"
    external: true
```

Важно. Сервис должен видеть директории с данными сервиса files и сервиса mapserver, если сервис расположен на другом узле, необходимо подключить каталог, как пример, через smb/sshfs и т.п. и изменить соответствующий путь в блоке volumes. Как пример, рабочая папка сервиса файлс монтирована в директорию /mnt/files, а рабочая папка сервиса mapserver в /mnt/tiles2, значит, путь необходимо задать как - /mnt/files:/app/files и соответственно для сервиса mapserver - /mnt/tiles2:/app/tiles2.

USER_PSQL – пользователь БД PostgreSQL

PASSWORD_PSQL – пароль пользователя БД PostgreSQL

HOST_PSQL_PAGES – адрес БД PostgreSQL где развернута база pages

PORT_PSQL_PAGES – порт БД PostgreSQL где развернута база pages

Загрузить образ из Docker репозитория registry.oxity.ru:5000.

```
sudo docker compose pull
```

Запустить Docker контейнер.

```
sudo docker compose up -d
```

Проверить, что контейнер запущен.

```
sudo docker compose ps
```

Сервис работы с растровыми слоями clipper_service

Распаковать архив с сервисом bands_service.

```
sudo tar xf clipper_service.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd clipper_service
```

Отредактировать docker-compose.yml.

```
version: '3.7'
services:
  clipper_service:
    container_name: clipper_service
    build:
      context: .
      dockerfile: ./docker/Dockerfile
    image: registry.oxity.ru:5000/svc/clipper_service
    command: uvicorn main:app --host 0.0.0.0 --port 8073
    restart: always
    volumes:
      - ./app:/app
      - ../files/files/4:/app/files:rw
      - ../mapserver/app/tiles:/app/tiles:rw
      - ../mapserver/app/tiles2:/app/tiles2:rw
    expose:
      - 8073
    ports:
      - 8073:8073
    environment:
      USER_PSQL: "<задать_пользователя_БД_PostgreSQL>"
      PASSWORD_PSQL: "<задать_пароль_пользователя_БД_PostgreSQL>"
      HOST_PSQL_BASE: "<задать_адрес_БД_PostgreSQL_base_60>"
      PORT_PSQL_BASE: <задать_порт_БД_PostgreSQL_base_60>
      DATABASE_PSQL_BASE: "base_60"
      HOST_PSQL_PAGES: "<задать_адрес_БД_PostgreSQL_pages>"
      PORT_PSQL_PAGES: <задать_порт_БД_PostgreSQL_pages>
```

```

DATABASE_PSQL_PAGES: "pages"
HOST_PSQL_GKN: "<задать_адрес_БД_PostgreSQL_gkn_18>"
PORT_PSQL_GKN: <задать_порт_БД_PostgreSQL_gkn_18>
DATABASE_PSQL_GKN: "gkn_18"
HOST_PSQL_LAYERS: "<задать_адрес_БД_PostgreSQL_layers>"
PORT_PSQL_LAYERS: <задать_порт_БД_PostgreSQL_layers>
DATABASE_PSQL_LAYERS: "layers"

networks:
  default:
    name: "project-network"
    external: true

```

Важно. Сервис должен видеть директории с данными сервиса `files` и сервиса `mapserver`, если сервис расположен на другом узле, необходимо подключить каталог, как пример, через `smb/sshfs` и т.п. и изменить соответствующий путь в блоке `volumes`. Как пример, рабочая папка сервиса `файлс` монтирована в директорию `/mnt/files`, а рабочая папка сервиса `mapserver` в `/mnt/tiles2`, значит, путь необходимо задать как - `/mnt/files/4:/app/files` и соответственно для сервиса `mapserver` - `/mnt/tiles2:/app/tiles2`.

`USER_PSQL` – пользователь БД PostgreSQL

`PASSWORD_PSQL` – пароль пользователя БД PostgreSQL

`HOST_PSQL_BASE` – адрес БД PostgreSQL где развернута база `base_60`

`PORT_PSQL_BASE` – порт БД PostgreSQL где развернута база `base_60`

`HOST_PSQL_PAGES` – адрес БД PostgreSQL где развернута база `pages`

`PORT_PSQL_PAGES` – порт БД PostgreSQL где развернута база `pages`

`HOST_PSQL_GKN` – адрес БД PostgreSQL где развернута база `gkn_18`

`PORT_PSQL_GKN` – порт БД PostgreSQL где развернута база `gkn_18`

`HOST_PSQL_LAYERS` – адрес БД PostgreSQL где развернута база `layers`

`PORT_PSQL_LAYERS` – порт БД PostgreSQL где развернута база `layers`

Загрузить образ из Docker репозитория `registry.oxity.ru:5000`.

```

sudo docker compose pull

```

Запустить Docker контейнер.

```

sudo docker compose up -d

```

Проверить, что контейнер запущен.

```

sudo docker compose ps

```

Сервис работы с растровыми слоями `onda_service`

Распаковать архив с сервисом `bands_service`.

```

sudo tar xf clipper_service.tar.gz

```

Перейти в директорию распакованного сервиса.

```
cd clipper_service
```

Отредактировать docker-compose.yml.

```
version: '3.7'
services:
  onda_service_fastapi:
    container_name: onda_service_fastapi
    build:
      context: .
      dockerfile: ./docker/Dockerfile
    image: registry.rusgis.com:5000/svc/onda_service_fastapi
    command: uvicorn main:app --host 0.0.0.0 --port 8076
    restart: always
    volumes:
      - ./app:/app
      - ./app/onda_archives:/app/onda_archives
      - ../files/files/./app/files
    environment:
      ONDA_USERNAME: "<задать_пользователя_для_подключения_к_ONDA_сервису>"
      ONDA_PASSWORD: "<задать_пароль_пользователя>"
      GEO_USERNAME: "<задать_пользователя_для_подключения_к_GeoServer>"
      GEO_PASSWORD: "<задать_пароль_пользователя_GeoServer>"
      USER_PSQL: "<задать_пользователя_БД_PostgreSQL>"
      PASSWORD_PSQL: "<задать_пароль_пользователя_БД_PostgreSQL>"
      HOST_PSQL: "<задать_адрес_БД_PostgreSQL_pages>"
      PORT_PSQL: "<задать_порт_БД_PostgreSQL_pages>"
      DATABASE_PSQL: "pages"
      PROJ_LIB: "/usr/local/lib/python3.7/site-packages/rasterio/proj_data/proj.db"
    ports:
      - '8076:8076'

networks:
  default:
    name: "project-network"
    external: true
```

ONDA_USERNAME – пользователя к сервису ONDA
ONDA_PASSWORD – пароль к пользователю от сервиса ONDA
GEO_USERNAME – пользователь к GeoServer
GEO_PASSWORD – пароль к пользователю от GeoServer
USER_PSQL – пользователь БД PostgreSQL
PASSWORD_PSQL – пароль пользователя БД PostgreSQL
HOST_PSQL – адрес БД PostgreSQL где развернута база pages
PORT_PSQL – порт БД PostgreSQL где развернута база pages

Важно. Сервис должен видеть директорию с данными сервиса files, если сервис расположен на другом узле, необходимо подключить каталог, как пример, через smb/sshfs и т.п. и изменить соответствующий путь в блоке volumes. Как пример,

рабочая папка монтирована в директорию /mnt/files, значит, путь необходимо задать как - /mnt/files:/app/files.

Загрузить образ из Docker репозитория registry.oxity.ru:5000.

```
sudo docker compose pull
```

Запустить Docker контейнер.

```
sudo docker compose up -d
```

Проверить, что контейнер запущен.

```
sudo docker compose ps
```

Сервис работы с растровыми слоями pansharpening_service

Распаковать архив с сервисом pansharpening_service.

```
sudo tar xf pansharpening_service.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd pansharpening_service
```

Отредактировать docker-compose.yml

```
version: '3.7'

services:
  pansharpening_service:
    container_name: pansharpening_service
    build:
      context: .
      dockerfile: ./docker/Dockerfile
    image: registry.rusgis.com:5000/svc/pansharpening_service
    command: uvicorn main:app --host 0.0.0.0 --port 8072
    restart: always
    volumes:
      - ./app:/app
      - ../files/files:/app/files
      - ../mapserver/app/tiles:/app/tiles
      - ../mapserver/app/tiles2:/app/tiles2
    ports:
      - '8072:8072'
    environment:
```

```

USER_PSQL: "<задать_пользователя_БД_PostgreSQL>"
PASSWORD_PSQL: "<задать_пароль_пользователя_БД_PostgreSQL>"
HOST_PSQL_CONFIG: "<задать_адрес_БД_PostgreSQL_config_19>"
PORT_PSQL_CONFIG: <задать_порт_БД_PostgreSQL_config_19>
DATABASE_PSQL_CONFIG: "config_19"
HOST_PSQL_BASE: "<задать_адрес_БД_PostgreSQL_base_60>"
PORT_PSQL_BASE: <задать_порт_БД_PostgreSQL_base_60>
DATABASE_PSQL_BASE: "base_60"
HOST_PSQL_PAGES: "<задать_адрес_БД_PostgreSQL_pages>"
PORT_PSQL_PAGES: <задать_порт_БД_PostgreSQL_pages>
DATABASE_PSQL_PAGES: "pages"

networks:
  default:
    name: "project-network"
    external: true

```

USER_PSQL – пользователь БД PostgreSQL

PASSWORD_PSQL – пароль пользователя БД PostgreSQL

HOST_PSQL_CONFIG – адрес БД PostgreSQL где развернута база config_19

PORT_PSQL_CONFIG – порт БД PostgreSQL где развернута база config_19

HOST_PSQL_BASE – адрес БД PostgreSQL где развернута база base_60

PORT_PSQL_BASE – порт БД PostgreSQL где развернута база base_60

HOST_PSQL_PAGES – адрес БД PostgreSQL где развернута база pages

PORT_PSQL_PAGES – порт БД PostgreSQL где развернута база pages

Важно. Сервис должен видеть директории с данными сервиса files и сервиса mapserver, если сервис расположен на другом узле, необходимо подключить каталог, как пример, через smb/sshfs и т.п. и изменить соответствующий путь в блоке volumes. Как пример, рабочая папка сервиса файле смонтирована в директорию /mnt/files, а рабочая папка сервиса mapserver в /mnt/tiles2, значит, путь необходимо задать как - /mnt/files:/app/files и соответственно для сервиса mapserver - /mnt/tiles2:/app/tiles2.

Загрузить образ из Docker репозитория registry.oxity.ru:5000.

```
sudo docker compose pull
```

Запустить Docker контейнер.

```
sudo docker compose up -d
```

Проверить, что контейнер запущен.

```
sudo docker compose ps
```

Сервис публикации растров publisher_service

Распаковать архив с сервисом publisher_service.

```
sudo tar xf publisher_service.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd publisher_service
```

Отредактировать docker-compose.yml.

```
version: '3.7'

services:
  publisher_service_celery:
    container_name: publisher_service_celery
    build:
      context: .
      dockerfile: ./docker/Dockerfile
    image: registry.oxity.ru:5000/svc/publisher_service_celery
    command: uvicorn main:app --host 0.0.0.0 --port 8084
    restart: always
    volumes:
      - ./app:/app
      - ../mapserver/app/tiles:/app/tiles:rw
      - ../mapserver/app/tiles2:/app/tiles2:rw
      - ../files/files:/app/files:rw
    environment:
      CELERY_BROKER_URL:
amqp://<пользователь_сервиса_rabbitmq>:<пароль_сервиса_rabbitmq>@<адрес_сервиса_rabbi
tmq>:<порт_сервиса_rabbitmq>//
      CELERY_RESULT_BACKEND:
db+postgresql://<пользователь_БД_PostgreSQL_celery>:<пароль_БД_PostgreSQL_celery>@<ад
рес_БД_PostgreSQL_celery>:<порт_БД_PostgreSQL_celery>/celery
    environment:
      USER_PSQL: "<задать_пользователя_БД_PostgreSQL>"
      PASSWORD_PSQL: "<задать_пароль_пользователя_БД_PostgreSQL>"
      HOST_PSQL_OLAP: "<задать_адрес_БД_PostgreSQL_celery>"
      PORT_PSQL_OLAP: <задать_порт_БД_PostgreSQL_celery>
      DATABASE_PSQL_OLAP: "celery"
    ports:
      - '8084:8084'

  publisher_worker:
    build:
      context: .
      dockerfile: ./docker/Dockerfile
    image: registry.oxity.ru:5000/svc/publisher_worker
    command: celery -A celery_worker worker --concurrency=${CONCURRENCY} --max-tasks-
per-child=1 --max-memory-per-child=2500000 --loglevel=INFO -Q publisher_queue
    restart: always
    volumes:
      - ./app:/app
      - ../mapserver/app/tiles:/app/tiles:rw
      - ../mapserver/app/tiles2:/app/tiles2:rw
      - ../files/files:/app/files:rw
    environment:
```

```

    - CELERY_BROKER_URL=
amqp://<пользователь_сервиса_rabbitmq>:<пароль_сервиса_rabbitmq>@<адрес_сервиса_rabbi
tmq>:<порт_сервиса_rabbitmq>//
    - CELERY_RESULT_BACKEND=
db+postgresql://<пользователь_БД_PostgreSQL_celery>:<пароль_БД_PostgreSQL_celery>@<ад
рес_БД_PostgreSQL_celery>:<порт_БД_PostgreSQL_celery>/celery
    depends_on:
    - publisher_service_celery

flower:
    build:
        context: .
        dockerfile: ./docker/Dockerfile
    image: registry.oxity.ru:5000/svc/flower
    command: celery -A celery_worker flower --loglevel=info --
purge_offline_workers=10 --timezone=EAT --inspect_timeout=10000
    restart: always
    environment:
    - CELERY_BROKER_URL=
amqp://<пользователь_сервиса_rabbitmq>:<пароль_сервиса_rabbitmq>@<адрес_сервиса_rabbi
tmq>:<порт_сервиса_rabbitmq>//
    - CELERY_RESULT_BACKEND=
db+postgresql://<пользователь_БД_PostgreSQL_celery>:<пароль_БД_PostgreSQL_celery>@<ад
рес_БД_PostgreSQL_celery>:<порт_БД_PostgreSQL_celery>/celery
    volumes:
    - ./app:/app
    depends_on:
    - publisher_service_celery
    - publisher_worker
    ports:
    - '5555:5555'

networks:
    default:
        name: "project-network"
        external: true

```

USER_PSQL – пользователь БД PostgreSQL

PASSWORD_PSQL – пароль пользователя БД PostgreSQL

HOST_PSQL_OLAP – адрес БД PostgreSQL где развернута база celery

PORT_PSQL_OLAP – порт БД PostgreSQL где развернута база celery

CELERY_BROKER_URL – подключение к сервису очереди RabbitMQ задаётся в виде ссылки, где указываются данные подключения пользователь:пароль@узел:порт. Как пример: amqp://user:PassW0rd@127.0.0.1:35672//.

CELERY_RESULT_BACKEND – подключение к БД PostgreSQL где развернута база celery задаётся в виде ссылки, где указываются данные подключения пользователь:пароль@узел:порт/имя_базы. Как пример:

db+postgresql://user:PassW0rd@127.0.0.1:65432/celery

Важно. Сервисы должны видеть директории с данными сервиса files и сервиса mapserver, если сервис расположен на другом узле, необходимо подключить каталог, как пример, через smb/sshfs и т.п. и изменить соответствующий путь в блоках volumes. Как пример, рабочая папка сервиса файлс смонтирована в директорию /mnt/files, а рабочая папка сервиса mapserver в /mnt/tiles2, значит, путь необходимо

здать как - /mnt/files:/app/files и соответственно для сервиса mapserver - /mnt/tiles2:/app/tiles2.

Загрузить образы из Docker репозитория registry.oxity.ru:5000.

```
sudo docker compose pull
```

Запустить Docker контейнеры.

```
sudo docker compose up -d
```

Проверить, что контейнеры запущен.

```
sudo docker compose ps
```

Сервис работы с растровыми слоями merger_service

Распаковать архив с сервисом merger_service.

```
sudo tar xf merger_service.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd merger_service
```

Отредактировать docker-compose.yml.

```
version: '3.7'
services:
  merger_service:
    container_name: merger_service
    build:
      context: .
      dockerfile: ./docker/Dockerfile
    image: registry.oxity.ru:5000/svc/merger_service
    command: uvicorn main:app --host 0.0.0.0 --port 8078
    restart: always
    volumes:
      - ./app:/app
      - ../mapserver/app/tiles:/app/tiles:rw
      - ../mapserver/app/tiles2:/app/tiles2:rw
      - ../files/files:/app/files:rw
    ports:
      - 8078:8078
    environment:
      USER_PSQL: "<здать_пользователя_БД_PostgreSQL>"
```

```

PASSWORD_PSQL: "<задать_пароль_пользователя_БД_PostgreSQL>"
HOST_PSQL_PAGES: "<задать_адрес_БД_PostgreSQL_pages>"
PORT_PSQL_PAGES: <задать_порт_БД_PostgreSQL_pages>
DATABASE_PSQL_PAGES: "pages"

networks:
  default:
    name: "project-network"
    external: true

```

USER_PSQL – пользователь БД PostgreSQL

PASSWORD_PSQL – пароль пользователя БД PostgreSQL

HOST_PSQL_PAGES – адрес БД PostgreSQL где развернута база pages

PORT_PSQL_PAGES – порт БД PostgreSQL где развернута база pages

Важно. Сервис должен видеть директории с данными сервиса files и сервиса mapserver, если сервисы расположен на другом узле, необходимо подключить каталог, как пример, через smb/sshfs и т.п. и изменить соответствующий путь в блоке volumes. Как пример, рабочая папка сервиса файле смонтирована в директорию /mnt/files, а рабочая папка сервиса mapserver в /mnt/tiles2, значит, путь необходимо задать как - /mnt/files:/app/files и соответственно для сервиса mapserver - /mnt/tiles2:/app/tiles2.

Загрузить образ из Docker репозитория registry.oxity.ru:5000.

```
sudo docker compose pull
```

Запустить Docker контейнер.

```
sudo docker compose up -d
```

Проверить, что контейнер запущен.

```
sudo docker compose ps
```

Сервис работы с растровыми слоями raster_composer

Распаковать архив с сервисом raster_composer.

```
sudo tar xf raster_composer.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd raster_composer
```

Отредактировать docker-compose.yml.

```
version: '3.7'
services:
  raster_composer:
    container_name: raster_composer
    build:
      context: .
      dockerfile: ./docker/Dockerfile
    image: registry.oxity.ru:5000/svc/raster_composer
    command: uvicorn main:app --host 0.0.0.0 --port 8075 --workers 3
    restart: always
    volumes:
      - ./app:/app
      - ../mapserver/app/tiles:/app/tiles/:rw
      - ../mapserver/app/tiles2:/app/tiles2/:rw
      - ../files/files:/app/files:rw
    ports:
      - 8075:8075
    environment:
      USER_PSQL: "<задать_пользователя_БД_PostgreSQL>"
      PASSWORD_PSQL: "<задать_пароль_пользователя_БД_PostgreSQL>"
      HOST_PSQL: "<задать_адрес_БД_PostgreSQL>"
      PORT_PSQL: <задать_порт_БД_PostgreSQL >
      DATABASE_PSQL_PAGES: "pages"
      DATABASE_PSQL_BASE: "base_60"

networks:
  default:
    name: "project-network"
    external: true
```

USER_PSQL – пользователь БД PostgreSQL

PASSWORD_PSQL – пароль пользователя БД PostgreSQL

HOST_PSQL – адрес БД PostgreSQL

PORT_PSQL – порт БД PostgreSQL

Важно. Сервис должен видеть директории с данными сервиса files и сервиса mapserver, если сервис расположен на другом узле, необходимо подключить каталог, как пример, через smb/sshfs и т.п. и изменить соответствующий путь в блоке volumes. Как пример, рабочая папка сервиса файлс смонтирована в директорию /mnt/files, а рабочая папка сервиса mapserver в /mnt/tiles2, значит, путь необходимо задать как - /mnt/files:/app/files и соответственно для сервиса mapserver - /mnt/tiles2:/app/tiles2.

Загрузить образ из Docker репозитория registry.oxity.ru:5000.

```
sudo docker compose pull
```

Запустить Docker контейнер.

```
sudo docker compose up -d
```

Проверить, что контейнер запущен.

```
sudo docker compose ps
```

Сервис работы отчётов reports

Распаковать архив с сервисом reports-docker.

```
sudo tar xf reports-docker.tar.gz
```

Перейти в директорию распакованного сервиса.

```
cd reports-docker
```

Установить ID 1001 пользователя и группы для рабочего каталога reports-data.

```
sudo chown 1001:1001 -R ./reports-docker/reports-data
```

Загрузить образ из Docker репозитория registry.oxity.ru:5000.

```
sudo docker compose pull
```

Запустить Docker контейнер.

```
sudo docker compose up -d
```

Проверить, что контейнер запущен.

```
sudo docker compose ps
```

Фронт - core

Распаковать архив с веб-приложением.

```
sudo tar xf core.tar.gz
```

Перейти в директорию распакованного сервиса.


```
cd core
```

Загрузить образ из Docker репозитория registry.oxity.ru:5000.

```
sudo docker compose pull
```

Запустить Docker контейнер.

```
sudo docker compose up -d
```

Проверить, что контейнер запущен.

```
sudo docker compose ps
```

Разворачивание базы в СУБД PostgreSQL

Добавить необходимые роли в БД.

```
CREATE ROLE confluence NOREPLICATION LOGIN;  
CREATE ROLE "ANALYST" NOINHERIT NOREPLICATION LOGIN;  
CREATE ROLE geoanalytic NOINHERIT NOREPLICATION LOGIN IN ROLE "ANALYST";  
CREATE ROLE iportal SUPERUSER CREATEDB NOREPLICATION LOGIN IN ROLE "ANALYST";  
CREATE ROLE iportal6 NOINHERIT NOREPLICATION LOGIN;  
CREATE ROLE khd NOINHERIT NOREPLICATION LOGIN;  
CREATE ROLE rc7postgres CREATEDB CREATEROLE NOREPLICATION LOGIN IN ROLE "ANALYST";  
CREATE ROLE log NOREPLICATION LOGIN;
```

Создание БД и расширение

Подключится к БД, например, с помощью утилиты psql.

```
psql -h 127.0.0.1 -p 65432 -U postgres
```

-h 127.0.0.1 - задает хост базы данных (в данном примере - адрес 127.0.0.1)

-p 65432 - задает порт базы данных (в данном примере - порт 65432)

-U postgres - задает имя пользователя для подключения к базе данных (в данном примере - имя пользователя postgres)

Создание БД – config_19

```
CREATE DATABASE config_19 WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE =  
'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;  
\c config_19  
CREATE EXTENSION postgis;  
CREATE EXTENSION dblink;
```

```
CREATE EXTENSION postgres_fdw;  
ALTER DATABASE config_19 SET postgres.enable_outdb_rasters TO '1';  
ALTER DATABASE config_19 SET postgres.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Создание БД – connections

```
CREATE DATABASE connections WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE =  
'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;  
\c connections  
CREATE EXTENSION postgis;  
CREATE EXTENSION dblink;  
CREATE EXTENSION postgres_fdw;  
CREATE EXTENSION plpython3u;  
CREATE EXTENSION adminpack;  
ALTER DATABASE connections SET postgres.enable_outdb_rasters TO '1';  
ALTER DATABASE connections SET postgres.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Создание БД – pages

```
CREATE DATABASE pages WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF-8'  
LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;  
\c pages  
CREATE EXTENSION postgis;  
CREATE EXTENSION plpython3u;  
CREATE EXTENSION dblink;  
CREATE EXTENSION postgres_fdw;  
ALTER DATABASE pages SET postgres.enable_outdb_rasters TO '1';  
ALTER DATABASE pages SET postgres.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Создание БД – gis

```
CREATE DATABASE gis WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF-8'  
LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;  
\c gis  
CREATE EXTENSION postgis;  
CREATE EXTENSION dblink;  
CREATE EXTENSION postgres_fdw;  
CREATE EXTENSION plpython3u;  
CREATE EXTENSION pldbapi;  
ALTER DATABASE gis SET postgres.enable_outdb_rasters TO '1';  
ALTER DATABASE gis SET postgres.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Создание БД – history

```
CREATE DATABASE history WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF-  
8' LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;  
\c history  
CREATE EXTENSION postgis;  
CREATE EXTENSION dblink;  
CREATE EXTENSION postgres_fdw;
```

```
ALTER DATABASE history SET postgis.enable_outdb_rasters TO '1';
ALTER DATABASE history SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Создание БД – gkn_18

```
CREATE DATABASE gkn_18 WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF-8'
LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;
\c gkn_18
CREATE EXTENSION plpython3u;
CREATE EXTENSION dblink;
CREATE EXTENSION postgis;
CREATE EXTENSION postgres_fdw;
CREATE EXTENSION pg_trgm;
ALTER DATABASE gkn_18 SET postgis.enable_outdb_rasters TO '1';
ALTER DATABASE gkn_18 SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Создание БД – gkn_history

```
CREATE DATABASE gkn_history WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE =
'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;
\c gkn_history
CREATE EXTENSION postgis;
CREATE EXTENSION lo;
CREATE EXTENSION xml2;
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION postgis_tiger_geocoder;
CREATE EXTENSION plpython3u;
CREATE EXTENSION pg_trgm;
CREATE EXTENSION dblink;
CREATE EXTENSION postgis_topology;
CREATE EXTENSION pg_stat_statements;
CREATE EXTENSION pgcrypto;
CREATE EXTENSION postgres_fdw;
ALTER DATABASE gkn_history_new SET postgis.enable_outdb_rasters TO '1';
ALTER DATABASE gkn_history_new SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Создание БД – layers

```
CREATE DATABASE layers WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF-8'
LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;
\c layers
CREATE EXTENSION postgis;
CREATE EXTENSION postgis_topology;
CREATE EXTENSION lo;
CREATE EXTENSION xml2;
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION postgis_tiger_geocoder;
CREATE EXTENSION postgis_raster;
CREATE EXTENSION pgrouting;
CREATE EXTENSION plpython3u;
CREATE EXTENSION pg_trgm;
CREATE EXTENSION dblink;
CREATE EXTENSION pg_stat_statements;
```

```

CREATE EXTENSION pgcrypto;
CREATE EXTENSION postgres_fdw;
CREATE EXTENSION pldbgapi;
CREATE EXTENSION mysql_fdw;
CREATE EXTENSION tablefunc;
CREATE EXTENSION kmeans;
ALTER DATABASE layers SET postgis.enable_outdb_rasters TO '1';
ALTER DATABASE layers SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';

```

Создание БД – base_60

```

CREATE DATABASE base_60 WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;
\c base_60
CREATE EXTENSION postgis;
CREATE EXTENSION postgis_topology;
CREATE EXTENSION lo;
CREATE EXTENSION xml2;
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION postgis_tiger_geocoder;
CREATE EXTENSION postgis_raster;
CREATE EXTENSION pgrouting;
CREATE EXTENSION plpython3u;
CREATE EXTENSION pg_trgm;
CREATE EXTENSION dblink;
CREATE EXTENSION pg_stat_statements;
CREATE EXTENSION pgcrypto;
CREATE EXTENSION postgres_fdw;
CREATE EXTENSION pldbgapi;
CREATE EXTENSION mysql_fdw;
CREATE EXTENSION tablefunc;
CREATE EXTENSION kmeans;
ALTER DATABASE base_60 SET postgis.enable_outdb_rasters TO '1';
ALTER DATABASE base_60 SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';

```

Создание БД – bpm

```

CREATE DATABASE bpm WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;
\c bpm
CREATE EXTENSION postgis;
CREATE EXTENSION plpython3u;
CREATE EXTENSION dblink;
CREATE EXTENSION pg_stat_statements;
ALTER DATABASE bpm SET postgis.enable_outdb_rasters TO '1';
ALTER DATABASE bpm SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';

```

Создание БД – bpmconfig

```

CREATE DATABASE bpmconfig WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;
\c bpmconfig
CREATE EXTENSION pldbgapi;

```

```
CREATE EXTENSION plpython3u;
CREATE EXTENSION dblink;
CREATE EXTENSION postgis;
ALTER DATABASE bpmconfig SET postgis.enable_outdb_rasters TO '1';
ALTER DATABASE bpmconfig SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Создание БД – olap

```
CREATE DATABASE olap WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF-8'
LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;
\c olap
CREATE EXTENSION adminpack;
CREATE EXTENSION dblink;
CREATE EXTENSION postgis;
CREATE EXTENSION postgis_raster;
CREATE EXTENSION postgres_fdw;
CREATE EXTENSION pldbgapi;
CREATE EXTENSION plpython3u;
ALTER DATABASE olap SET postgis.enable_outdb_rasters TO '1';
ALTER DATABASE olap SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Создание БД – osm

```
CREATE DATABASE osm WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF-8'
LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;
\c osm
CREATE EXTENSION plpython3u;
CREATE EXTENSION postgis;
CREATE EXTENSION pgcrypto;
CREATE EXTENSION dblink;
ALTER DATABASE osm SET postgis.enable_outdb_rasters TO '1';
ALTER DATABASE osm SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Создание БД – geocode

```
CREATE DATABASE geocode WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF-8'
LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;
\c geocode
CREATE EXTENSION dblink;
CREATE EXTENSION postgis;
ALTER DATABASE geocode SET postgis.enable_outdb_rasters TO '1';
ALTER DATABASE geocode SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Создание БД – iot

```
CREATE DATABASE iot WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF-8'
LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;
\c iot
CREATE EXTENSION postgis;
```

```

CREATE EXTENSION pg_trgm;
CREATE EXTENSION dblink;
CREATE EXTENSION pg_stat_statements;
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION xml2;
CREATE EXTENSION lo;
CREATE EXTENSION postgres_fdw;
CREATE EXTENSION pgcrypto;
ALTER DATABASE iot SET postgis.enable_outdb_rasters TO '1';
ALTER DATABASE iot SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';

```

Создание БД – graph

```

CREATE DATABASE graph WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF-8'
LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;
\c graph
CREATE EXTENSION postgis;
CREATE EXTENSION pgrouting;
CREATE EXTENSION postgres_fdw;
CREATE EXTENSION pg_stat_statements;
CREATE EXTENSION pgcrypto;
CREATE EXTENSION postgis_raster;
CREATE EXTENSION postgis_topology;
CREATE EXTENSION xml2;
CREATE EXTENSION postgis;
CREATE EXTENSION tablefunc;
CREATE EXTENSION plpgsql;
CREATE EXTENSION dblink;
CREATE EXTENSION pg_trgm;
CREATE EXTENSION kmeans;
CREATE EXTENSION mysql_fdw;
ALTER DATABASE graph SET postgis.enable_outdb_rasters TO '1';
ALTER DATABASE graph SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';

```

Создание БД – gko

```

CREATE DATABASE gko WITH OWNER postgres ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF-8'
LC_CTYPE = 'en_US.UTF-8' TEMPLATE = template0;
\c gko
CREATE EXTENSION postgis;
CREATE EXTENSION postgis_topology;
CREATE EXTENSION lo;
CREATE EXTENSION xml2;
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION postgis_tiger_geocoder;
CREATE EXTENSION postgis_raster;
CREATE EXTENSION pgrouting;
CREATE EXTENSION plpython3u;
CREATE EXTENSION pg_trgm;
CREATE EXTENSION dblink;
CREATE EXTENSION pg_stat_statements;
CREATE EXTENSION pgcrypto;
CREATE EXTENSION postgres_fdw;
CREATE EXTENSION pldbapi;
CREATE EXTENSION mysql_fdw;
CREATE EXTENSION tablefunc;

```

```
CREATE EXTENSION kmeans;  
ALTER DATABASE gko SET postgis.enable_outdb_rasters TO '1';  
ALTER DATABASE gko SET postgis.gdal_enabled_drivers TO 'ENABLE_ALL';
```

Разворачивание базы из резервной копии

```
pg_restore -h 127.0.0.1 -p 65432 -U postgres -j 6 -Fc base_name.dump -d base_name
```

-h 127.0.0.1 - задает хост базы данных (в данном примере - адрес 127.0.0.1)
-p 65432 - задает порт базы данных (в данном примере - порт 65432)
-U postgres - задает имя пользователя для подключения к базе данных (в данном примере - имя пользователя postgres)
-j 6 - задает количество параллельных восстановлений (в данном примере - 6 параллельных восстановлений)
-Fc - указывает формат архива (в данном примере - формат "custom")
base_name.dump - имя файла дампа базы данных
-d base_name - задает имя базы данных для восстановления (в данном примере - имя базы данных base_name)

Настройка конфигурации в базах

Подключится к БД как пример с помощью утилиты psql.

```
psql -h 127.0.0.1 -p 65432 -U postgres -d db_name
```

-h 127.0.0.1 - задает хост базы данных (в данном примере - адрес 127.0.0.1)
-p 65432 - задает порт базы данных (в данном примере - порт 65432)
-U postgres - задает имя пользователя для подключения к базе данных (в данном примере - имя пользователя postgres)
-d db_name - задает имя целевой базы.

Во всех развернутых БД выполнить UPDATE с указанием: адрес узла БД, порт БД, пользователь БД, пароль БД. Как пример.

```
update config.connections set host = '127.0.0.1', port = '5432', "user" = 'user_pg',  
password = 'PassW0rd' where dbname = 'db_name';
```

host – адрес целевой БД PostgreSQL

port – порт целевой БД PostgreSQL

user – пользователь целевой БД PostgreSQL

password – пароль пользователя целевой БД PostgreSQL

dbname – название целевой БД PostgreSQL

В базе данных **config_19** выполнить запрос на UPDATE с параметрами подключения к ClickHouse сервер.

```
update config.connections set host = '127.0.0.1', "user" = 'user_ch', password =  
'PassW0rd' where port = '8123' or port = '9000';
```

host – адрес целевой БД ClickHouse

user – пользователь целевой БД ClickHouse

password – пароль пользователя целевой БД ClickHouse

В базе данных **config_19** выполнить запрос на UPDATE с параметрами подключения к GeoServer.

```
update config.connections c set host = '127.0.0.1:8888', "user" = 'user_geoserver',  
password = 'PassW0rd' where c.connecttype_id = 3;
```

host – адрес:port развернутого GeoServer

user – пользователь GeoServer

password – пароль пользователя GeoServer

В базе данных **connections** выполнить запрос на UPDATE с указанием сервиса API.

```
update config.connections set protocol = 'http', host = 'http://127.0.0.1:8101/api'  
where id = '4294'
```

host – http ссылка с адресом и портом сервиса API

В базе данных **connections** выполнить запрос на INSERT.

```
INSERT INTO config.portal_domain AS d (name, portal_id, is_main, protocol) VALUES  
( 'gis.portal.ru', 100, TRUE, https);
```

name – задать домен развернутой информационной системы

portal_id – задать ID портала

is_main – задать является ли домен основным TRUE или FALSE

protocol – задать протокол http/https по которому доступен домен

В базе данных **connections** выполнить запрос на UPDATE с параметрами подключения к GeoServer.

```
update config.connections set host = '127.0.0.1:8888/geoserver', "user" =  
'user_geoserver', password = 'PassW0rd' where code like '%ГЕОСЕРВЕР%';
```

host – адрес:port развернутого GeoServer

user – пользователь GeoServer

password – пароль пользователя GeoServer

Получить и проверить список баз данных проекта с помощью запроса SELECT в базе данных **connections**.

```
SELECT c.id, c.connect_str, c.code, c.connecttype_id  
FROM config.connections c  
WHERE ( c.connecttype_id = 1 or c.connecttype_id = 9  
or ( c.connecttype_id = 8 and c.region = 60 ) or ( c.connecttype_id = 10 and c.portal_id = 100 ));
```


c.portal_id – задать ID портала

Отключить базу данных от проекта с помощью запроса UPDATE.

```
update config.connections set connecttype_id = '-1' WHERE id = 4222;
```

id – задать ID базы данных для отключения её из проекта, можно получить из списка запроса на SELECT выше.

В базе данных **bpm** выполнить запрос на UPDATE с параметрами подключения к сервису очереди RabbitMQ.

```
update config.connections set host = '127.0.0.1', port = '', "user" = 'user_rabbitmq', password = 'PassW0rd' where id = '4144';
```

host – адрес RabbitMQ

port – порт RabbitMQ

user – пользователь RabbitMQ

password – пароль пользователя RabbitMQ

В базе данных **gis** выполнить запросы на UPDATE с параметрами подключения к сервисам.

```
update config.connections set host = 'http://gis.portal.ru/' where id = '4122';
update config.connections set host = 'http://127.0.0.1:8075/' where id = '4131';
update config.connections set host = 'http://127.0.0.1:8074/' where id = '4133';
update config.connections set host = 'http://127.0.0.1:8073/' where id = '4136';
update config.connections set host = 'http://127.0.0.1:8072/' where id = '4137';
update config.connections set host = 'http://127.0.0.1:8084/' where id = '4139';
update config.connections set host = 'http://127.0.0.1:8078/' where id = '4140';
update config.connections set host = 'http://127.0.0.1:8071/' where id = '4141';
update config.connections set host = 'http://127.0.0.1:8074/' where id = '4143';
update config.connections set host = 'http://127.0.0.1:8069/' where id = '4142';
update config.connections set host = 'http://127.0.0.1:8170/' where id = '4144';
```

host – где id 4122 задать FQDN проекта, в остальных заменить адрес 127.0.0.1 на адрес где развернуты сервисы работы с растрами (bands_service, clipper_service, mapserver, merger_service, onda_service, pansharpening_service, publisher_service, raster_composer).

Конфигурация Nginx

Создать отдельный файл конфигурации для веб-приложения в директории /etc/nginx/conf.d.

Как пример: front-oxity-ru.conf

```
server {
    listen 80;

    server_name oxity.ru;
    access_log /var/log/nginx/nginx.access.log combined;

    client_body_in_single_buffer on;
    client_max_body_size 0;
    proxy_request_buffering off;
```

```

client_header_buffer_size 1024M;

keepalive_timeout 600000s;
proxy_connect_timeout 18000s;
proxy_send_timeout 18000s;
proxy_read_timeout 18000s;
send_timeout 18000s;

large_client_header_buffers 16 64k;

include /etc/nginx/conf.d/services.conf;

location / {
    proxy_pass http://127.0.0.1:8081/;
    proxy_set_header    Host                $http_host;
    proxy_set_header    X-Real-IP           $remote_addr;
    proxy_set_header    X-Forwarded-For    $proxy_add_x_forwarded_for;
    proxy_pass_header    Content-Type;
    proxy_pass_header    Content-Disposition;
    proxy_pass_header    Content-Length;
    proxy_pass_header    Set-Cookie;
    proxy_http_version  1.1;
    proxy_set_header    X-Forwarded-Proto $scheme;
}
}

```

listen 80: сервер будет слушать на порту 80.

server_name oxity.ru: указывает доменное имя сервера.

access_log /var/log/nginx/nginx.access.log combined: определяет файл журнала доступа и формат записей в нем.

client_body_in_single_buffer on: разрешает считывание тела запроса клиента в один буфер.

client_max_body_size 0: устанавливает максимальный размер тела запроса клиента.

proxy_request_buffering off: выключает буферизацию запросов прокси.

client_header_buffer_size 1024M: устанавливает размер буфера для заголовков клиента.

keepalive_timeout 600000s: устанавливает таймаут поддержки активного соединения с клиентом.

proxy_connect_timeout 18000s: устанавливает таймаут соединения с бэкэнд-сервером.

proxy_send_timeout 18000s: устанавливает таймаут отправки данных на бэкэнд-сервер.

proxy_read_timeout 18000s: устанавливает таймаут чтения данных с бэкэнд-сервера.

send_timeout 18000s: устанавливает таймаут отправки данных клиенту.

large_client_header_buffers 16 64k: устанавливает буфер для больших клиентских заголовков.

proxy_pass http://127.0.0.1:8081/; указывает бэкэнд-сервер, к которому будут проксироваться запросы.

proxy_set_header: устанавливает значения заголовков для прокси-запроса, таких как Host, X-Real-IP, X-Forwarded-For, X-Forwarded-Proto.

Создать отдельный файл конфигурации для сервисов в директории /etc/nginx/conf.d. Как пример: services.conf.

```

# Stimulsoft
location ~
(/showReport|/showDesigner|/stimulsoft_webdesigner_action|/stimulsoft_web_resource|/s
timulsoft_webviewer_action) {
    proxy_pass http://127.0.0.1:8104/lauth?url=http://127.0.0.1:8082$request_uri;
    gzip_static on;
    gzip on;
    gzip_disable "msie6";
    gzip_vary on;
    gzip_proxied any;
    gzip_comp_level 5;
    gzip_buffers 16 8k;
    gzip_http_version 1.1;
    gzip_min_length 256;
    gzip_types application/x-www-form-urlencoded;
    proxy_redirect off;
    proxy_set_header Connection close;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

# Прокси
location /proxy {
    proxy_pass http://127.0.0.1:8118;
    proxy_redirect off;
    proxy_set_header Connection close;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_ssl_server_name on;
}

# Composite
location /composite {
    proxy_pass http://127.0.0.1:8075;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_pass_header Content-Type;
    proxy_pass_header Content-Disposition;
    proxy_pass_header Content-Length;
    proxy_pass_header Set-Cookie;
    proxy_http_version 1.1;
    proxy_set_header X-Forwarded-Proto $scheme;
}

# Mapserver
location /tiles {
    proxy_pass http://127.0.0.1:8077;
    proxy_redirect off;
    proxy_set_header Connection close;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

location /translate_to_cog {
    proxy_pass http://127.0.0.1:8077;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
}

```

```

proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_pass_header   Content-Type;
proxy_pass_header   Content-Disposition;
proxy_pass_header   Content-Length;
proxy_pass_header   Set-Cookie;
proxy_http_version  1.1;
proxy_set_header    X-Forwarded-Proto $scheme;
}

# Publisher
location /get_tiles {
    proxy_pass http://127.0.0.1:8076;
    proxy_set_header    Host $http_host;
    proxy_set_header    X-Real-IP $remote_addr;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_pass_header   Content-Type;
    proxy_pass_header   Content-Disposition;
    proxy_pass_header   Content-Length;
    proxy_pass_header   Set-Cookie;
    proxy_http_version  1.1;
    proxy_set_header    X-Forwarded-Proto $scheme;
}

# Загрузка и получение файлов (Files)
location ~ (/downloadFile|/getPortraitImage/uploadFile|/apiFiles) {
    proxy_pass http://127.0.0.1:8102;
    gzip_static on;
    gzip on;
    gzip_vary on;
    gzip_proxied any;
    gzip_comp_level 5;
    gzip_buffers 16 8k;
    gzip_http_version 1.1;
    gzip_min_length 256;
    gzip_types application/octet-stream;
    proxy_redirect off;
    proxy_set_header Connection close;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

# TMS
location /tms {
    proxy_pass http://127.0.0.1:8101;
    gzip_static on;
    gzip on;
    gzip_disable "msie6";
    gzip_vary on;
    gzip_proxied any;
    gzip_comp_level 5;
    gzip_buffers 16 8k;
    gzip_http_version 1.1;
    gzip_min_length 256;
    gzip_types application/vnd.mapbox-vector-tile;
    proxy_redirect off;
    proxy_set_header Connection close;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

```

```

# API_SERVICE
location ~
^/(api|dataset|createSession|update_session|removeSession|logout|providers|info2|init
_api_keys) {
    proxy_pass http://127.0.0.1:8104;
    gzip_static on;
    gzip on;
    gzip_disable "msie6";
    gzip_vary on;
    gzip_proxied any;
    gzip_comp_level 5;
    gzip_buffers 16 8k;
    gzip_http_version 1.1;
    gzip_min_length 256;
    gzip_types application/geo+json application/json;
    proxy_redirect off;
    proxy_set_header Connection close;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

# Прокси-аутентификации
location ~ ^/(info|login_proxy|oauth2) {
    proxy_pass http://127.0.0.1:3000;
    proxy_redirect off;
    proxy_set_header Connection close;
    proxy_set_header Host $http_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-Host $host;
    proxy_set_header X-Forwarded-Proto $scheme;
}

```

Проверить конфигурацию на наличие ошибок.

```
sudo nginx -t
```

Применить конфигурацию.

```
sudo nginx -s reload
```