ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ ФИРМА "ПАЛИТРА СИСТЕМ"
ИНН 7718053570, КПП 772401001,
115201, МОСКВА, КАШИРСКОЕ, ДОМ 22, КОРПУС 3, ЭТ/ПОМ 9/1,1А,
+7 499 754-10-04, metrolog@palitra-system.ru , http://palitra-system.ru

Документация, содержащая информацию, необходимую для установки программного обеспечения «Автоматизированная система управления метрологической службой» (АСУ МС) версия 7

Москва, 2022 г.

Развертывание архитектуры микро-сервисов

Развертывание и настройка ОС

Требования к системе

OS: Linux Ubuntu 20.04 LTS Hardware: минимум @CPU 4, RAM 8GB, HDD(System) 50GB, SSD(Data) 200GB.

Установка системы

При установке ОС необходимо разделять данные системы и данные приложений на разных дисках.

Для диска с данными необходимо использовать журналируемую файловую систему Btrfs.

Настройка системы

- Установка для SSH метода аутентификации через пароль: раскомментировать в файле /etc/ssh/sshd_config

```
PasswordAuthentication yes
```

и перезапустить демон ssh

```
$ systemctl restart sshd
```

- Изменение имени хоста:

```
$ sudo hostnamectl set-hostname prod01
```

- Установка часового пояса:

```
$ sudo timedatectl set-timezone Europe/Moscow
```

Установка локали и формата времени 24h:

```
# генерируем локаль
$ sudo locale-gen ru_RU.UTF-8
# устанавливаем вывод времени в формате 24h
$ sudo update-locale LANG=en_US.UTF-8 LC_TIME="ru_RU.UTF-8"
```

```
# после требуется перелогиниться
# просмотр информации о локали
$ locale
# вывод даты
$ date
```

- Добавление нового пользователя psadmin и соответствующей группы psadmin:

```
$ sudo adduser psadmin
```

- Добавление пользователя psadmin в группу администраторов sudo и adm:

```
$ sudo usermod -G sudo,adm psadmin
```

- Установка пароля для пользователя `psadmin`

```
$ sudo passwd psadmin
```

- Установка необходимых сетевых утилит и приложений:

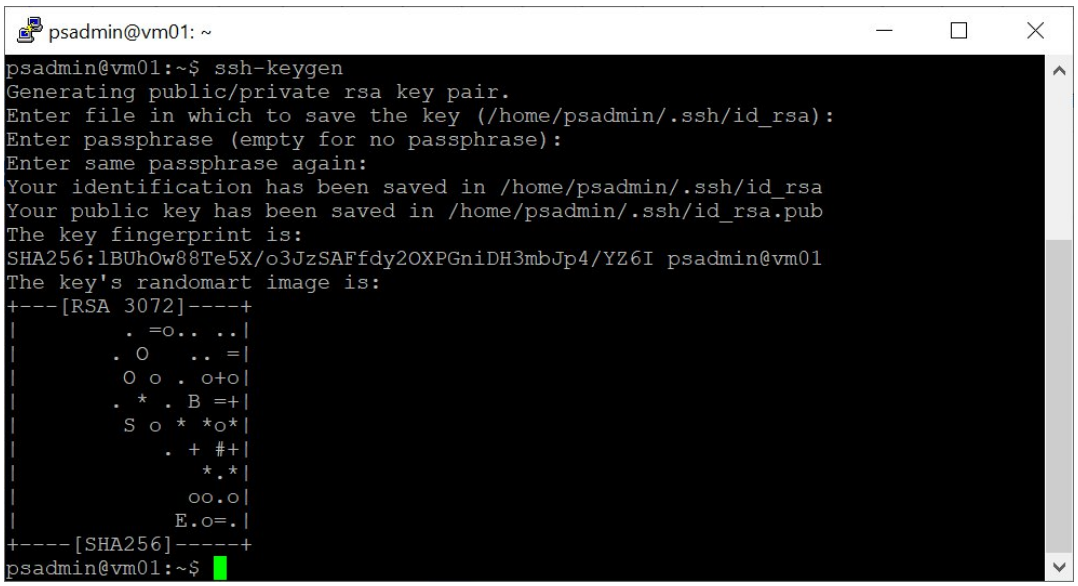
```
$ sudo apt install mc curl inetutils-ping net-tools dnsutils
```

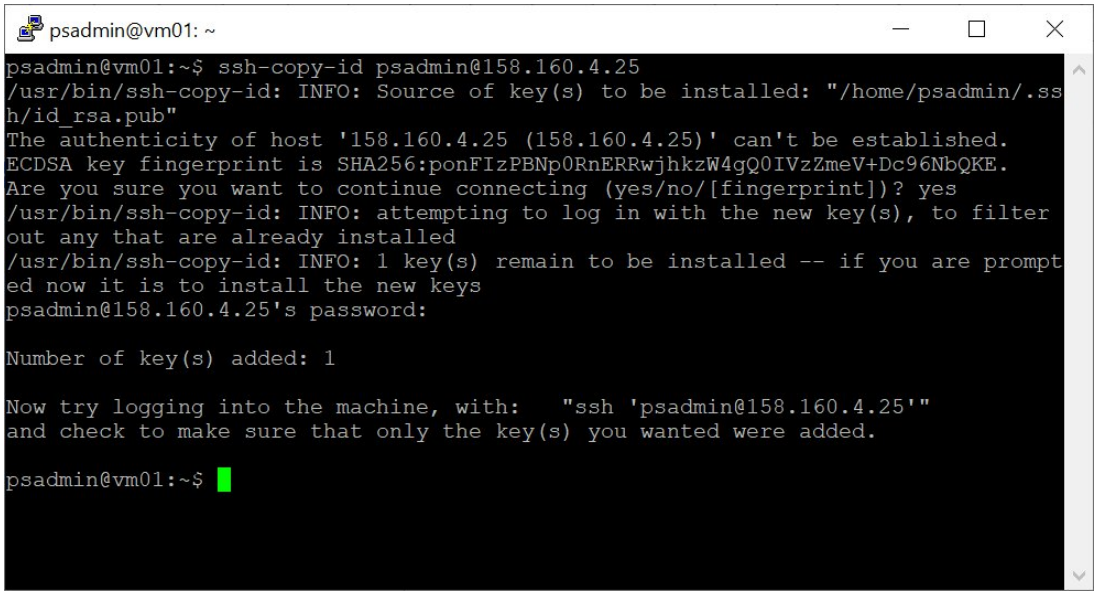
- Установка утилиты для поиска файлов в пакетах:

```
$ sudo apt install apt-file
$ sudo apt-file update
```

- Настройка авторизации между серверами по протоколу SSH:

```
# генерируем ключ
$ sudo ssh-keygen
# копируем ключ на другой сервер
$ ssh-copy-id psadmin@158.160.4.25
```





Теперь авторизовавшись на одном сервере, можно заходить на другой сервер:

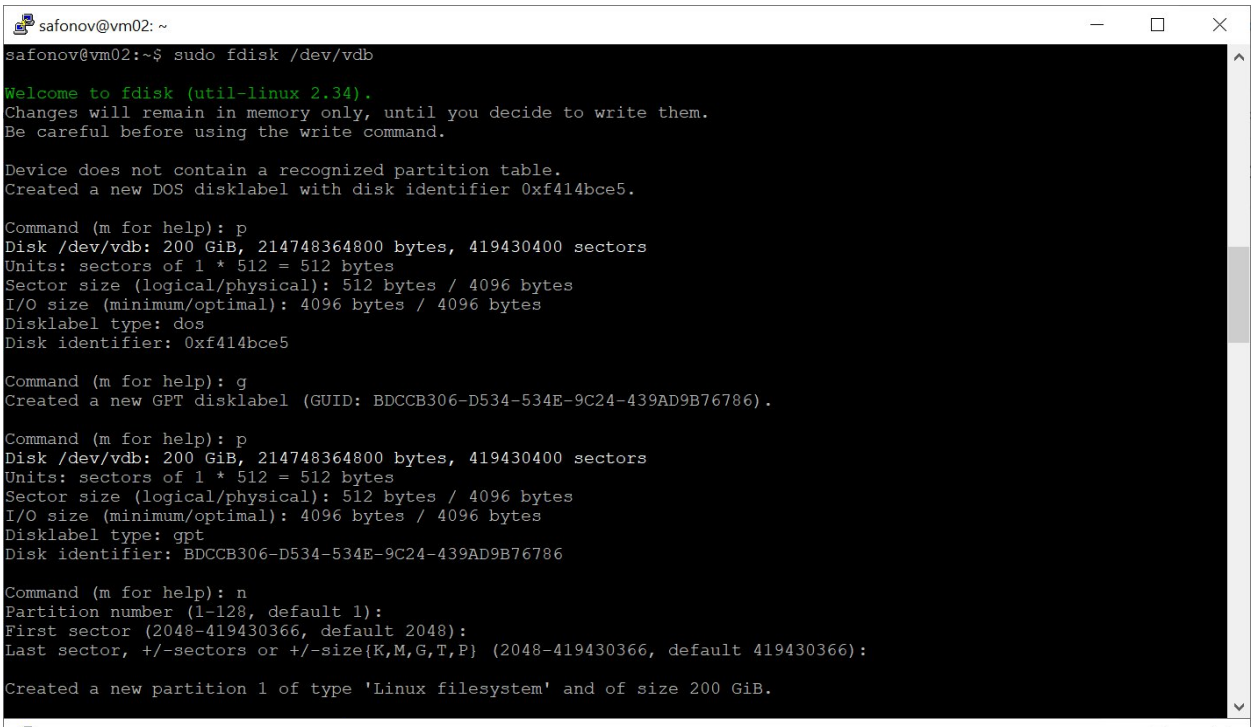
```
$ ssh psadmin@158.160.4.25
```

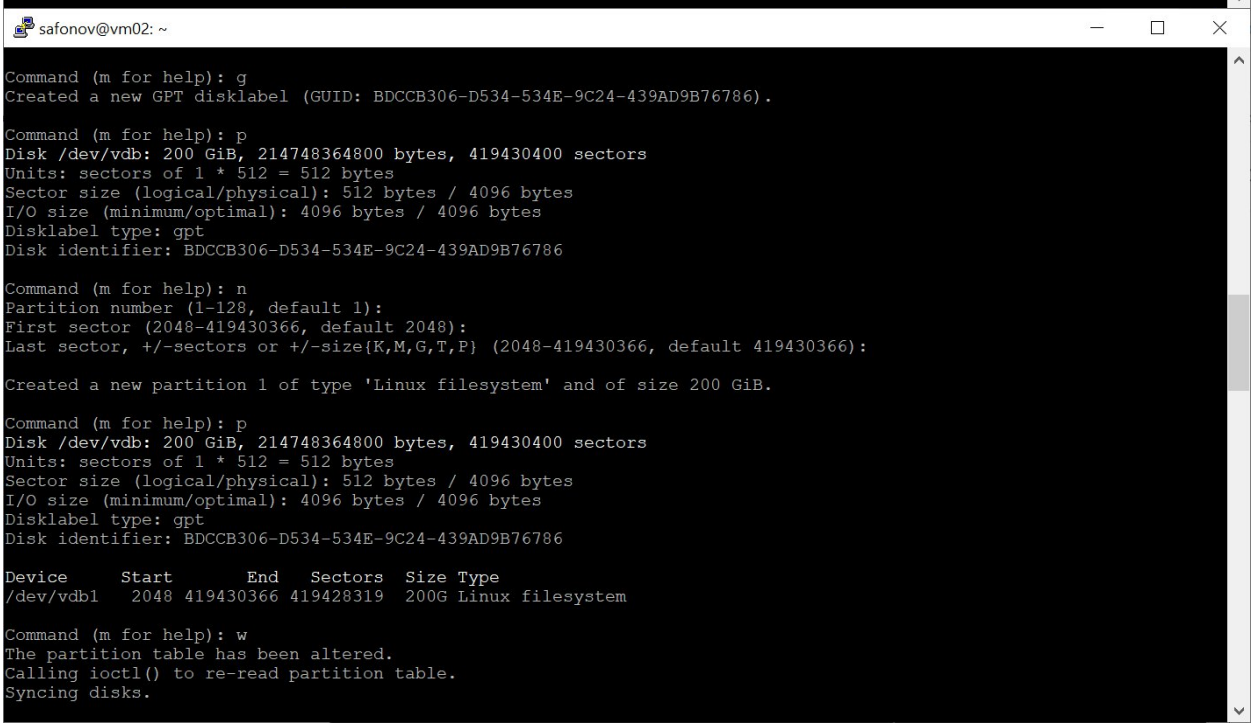
Подключение нового диска к системе

```
# ищем утилиту для создания файловой системы btrfs
$ apt-file search mkfs.btrfs
# устанавливаем утилиту
$ sudo apt install btrfs-progs
# просмотр нужного диска
$ sudo fdisk -l
# создание основного раздела с помощью утилиты fdisk
$ sudo fdisk /dev/vdb
# формирование файловой системы btrfs
```

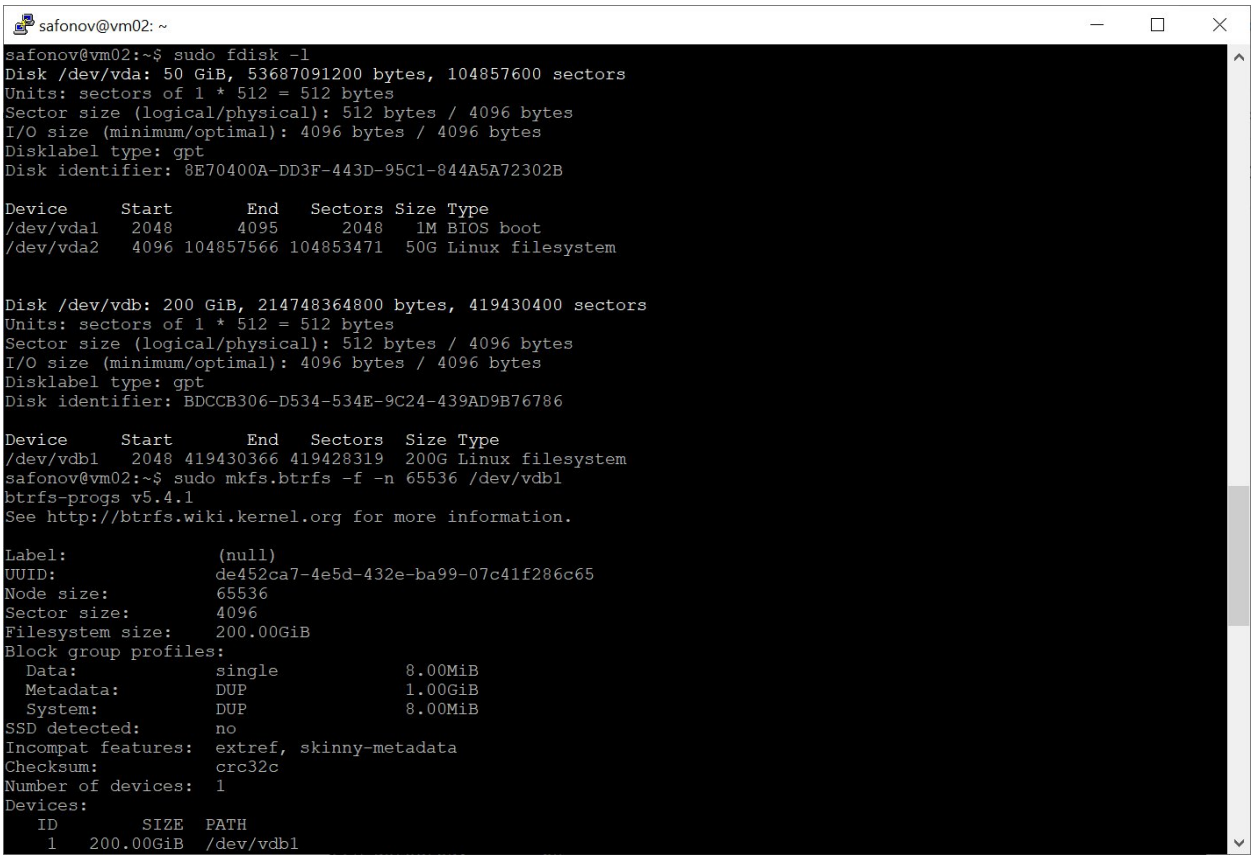
```
$ sudo mkfs.btrfs -f -n 65536 /dev/vdb1
# монтирование диска в /opt
$ sudo mount /dev/vdb1 /opt
# просмотр смонтированных дисков
$ df -h
# для генерации записи в /etc/fstab использовать утилиту genfstab
$ sudo apt install arch-install-scripts
$ genfstab -U /
# добавление автоматического монтирования диска в /etc/fstab
# /dev/vdb1
UUID=caa25284-b5ab-4f11-99e0-844ed9e73773       /opt            btrfs
rw,relatime,space_cache,subvolid=5,subvol=/       0 0
```

Создание раздела с помощью утилиты fdisk:





Формирование файловой системы:



Установка и настройка Docker Engine

Установка Docker Engine на ОС Ubuntu:

https://docs.docker.com/engine/install/ubuntu/

```
# обновляем список пакетов и устанавливаем пакеты для использования репозитория через протокол https
$ sudo apt-get update
$ sudo apt-get install ca-certificates curl gnupg lsb-release
# добавляем ключ
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
# устанавливаем репозиторий
$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
# обновляем список пакетов и устанавливаем последние версии Docker Engine, containerd и Docker Compose
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Перенос дискового пространства, используемого docker, на отдельный раздел:

https://docs.docker.com/config/daemon/systemd/#runtime-directory-and-storage-driver

```
# создаем папку для root-директории docker на отдельном разделе
$ sudo mkdir -p /opt/docker
# создаем конфигурационный файл демона docker
$ sudo touch /etc/docker/daemon.json
# устанавливаем папку для root-директории docker
$ sudo echo $'{\n "data-root": "/opt/docker/"\n}' | sudo tee /etc/docker/daemon.json
# перестартовываем демон docker
$ sudo systemctl restart docker
```

Пост-инсталляционные настройки docker:

https://docs.docker.com/engine/install/linux-postinstall/#manage-docker-as-a-non-root-user

```
# добавление пользователя в группу docker
$ sudo usermod -aG docker psadmin
```

Если пользователь присутствует в группе docker, то выполнение команд docker можно производить без sudo.

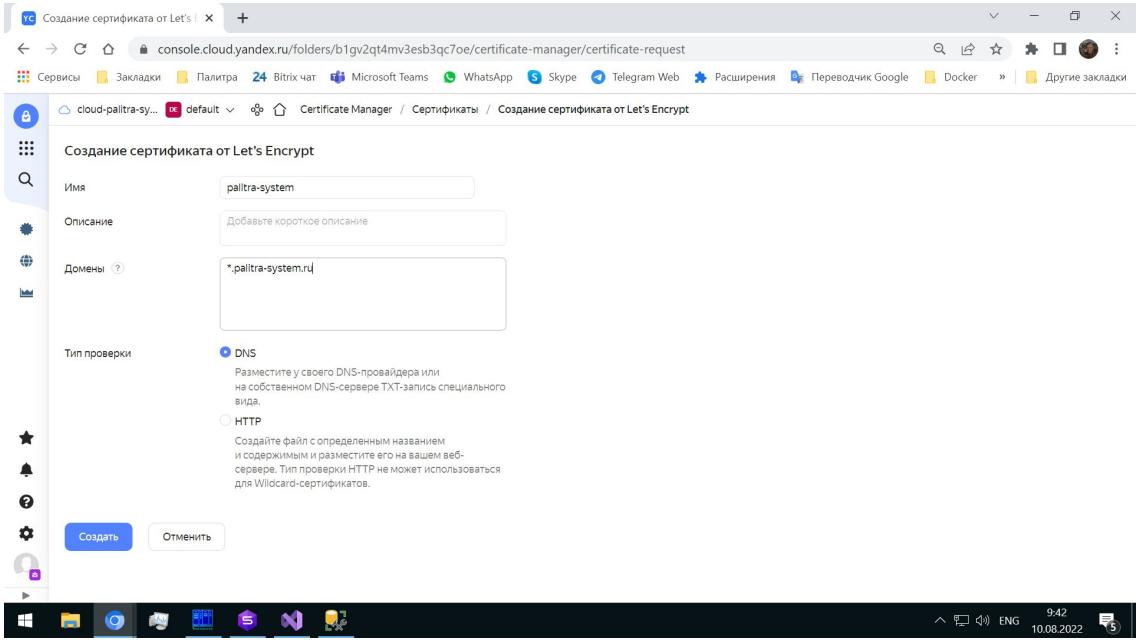
Установка и настройка сертификатов

Для работы пользователя с web-приложением через браузер и взаимодействия микро-сервисов между собой необходимо сгенерировать и использовать SSL-сертификат(ы).

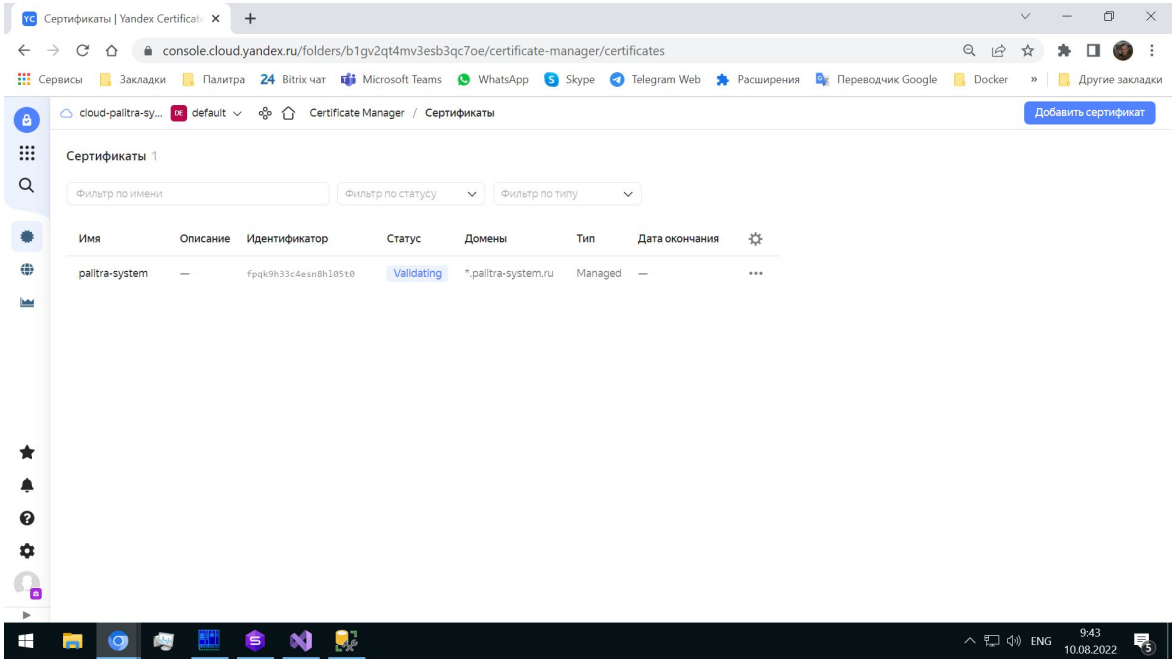
Для доступа к сервисам используется сертификат для домена 3-го уровня, напр.: web.palitra-system.ru.

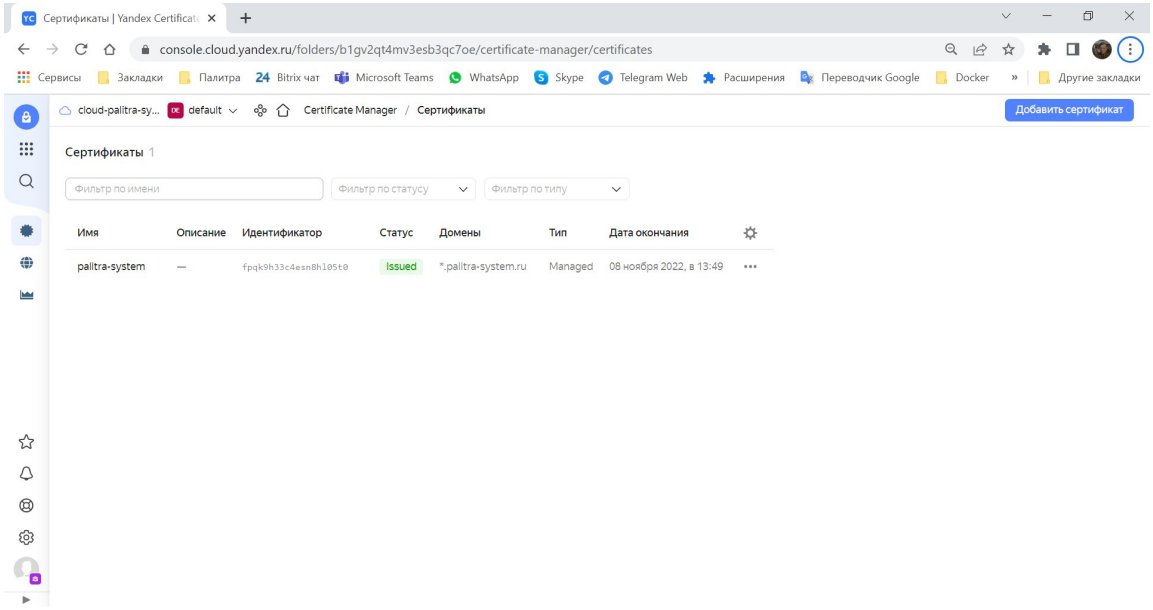
Для взаимодействия микро-сервисов между собой используется сертификат для домена 3-го уровня, либо 4-го уровней, напр.: *.web.palitra-system.ru

Создать сертификат в соответствии с описанием: https://cloud.yandex.ru/docs/certificate-manager/operations/managed/cert-create



Пройти процедуру проверки прав на домен в соответствии с описанием: https://cloud.yandex.ru/docs/certificate-manager/operations/managed/cert-validate





Установить интерфейс командной строки Yandex Cloud CLI и создать профиль в соответствии с описанием: https://cloud.yandex.ru/docs/cli/quickstart#install

```
$ curl -sSL https://storage.yandexcloud.net/yandexcloud-yc/install.sh | bash
```

Получить цепочку сертификатов и закрытый ключ:

```
$ yc certificate-manager certificate content --name 'palitra-system' --chain palsys-cert.cer --key palsys-cert.key
```

Экспортировать сертификат и ключ в pfx-формат для подключения к сервисам .net:

```
$ openssl pkcs12 -export -out ./palsys-cert.pfx -inkey palsys-cert.key -in palsys-cert.cer -name "Palitra System"
# просмотр содержимого
$ openssl x509 -in palsys-cert.cer -text -noout
```

Создание DNS-зоны

- Создать A-запись для публичной DNS-зоны в регистраторе

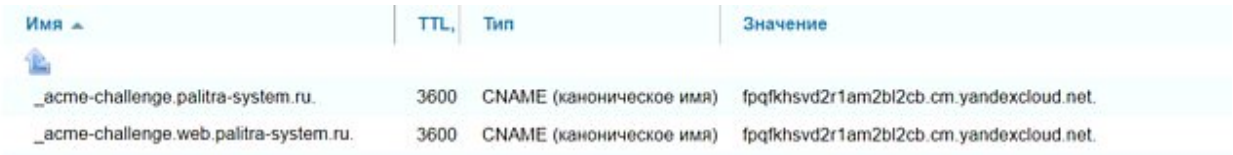
Требуется создать домен 3-го уровня, например: web.palitra-system.ru.

Для сопоставления доменного имени и IPv4-адреса требуется вручную добавить A-запись в публичную зону после создания ВМ.

Необходимо время чтобы изменения вступили в силу.

- Создать CNAME-запись для получения и обновления сертификата от Let’s Encrypt

Для получения и обновления сертификата от Let’s Encrypt необходимо пройти процедуру проверки прав на каждый домен, указанный в сертификате. Для этого для каждой DNS-записи в сертификате требуется создать CNAME-запись в регистраторе.



Установка и настройка кластера Docker Swarm

- Инициализация manager - узла:

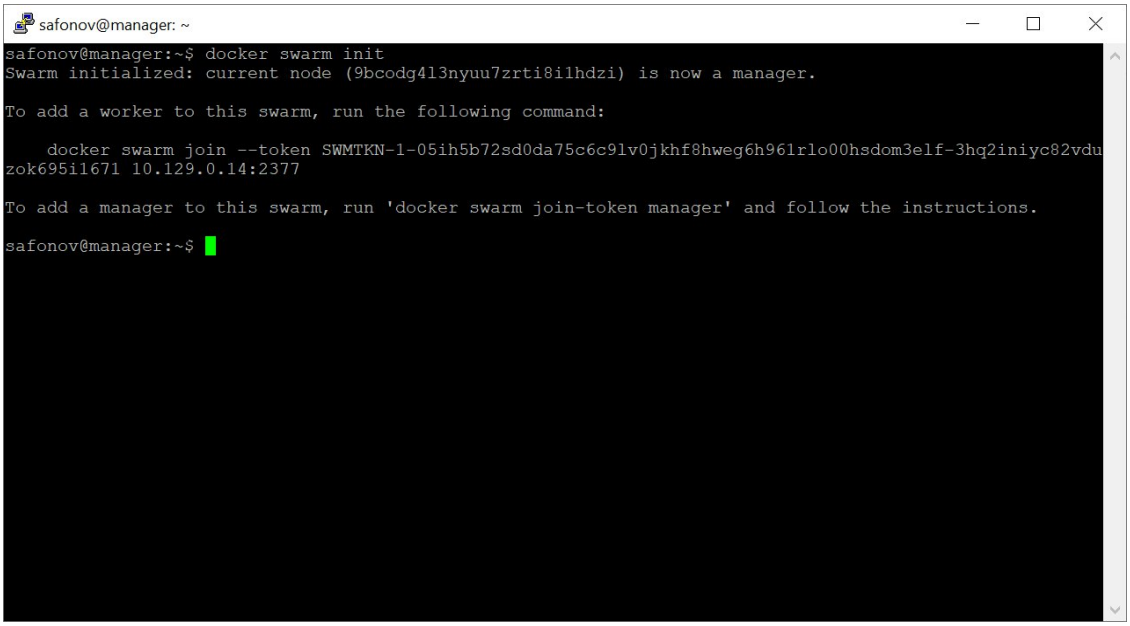
```
$ docker swarm init
```

Swarm initialized: current node (9bcodg4l3nyuu7zrti8i1hdzi) is now a manager.

To add a worker to this swarm, run the following command:

docker swarm join --token SWMTKN-1-05ih5b72sd0da75c6c9lv0jkhf8hweg6h961rlo00hsdom3elf-3hq2iniyc82vduzok695i167110.129.0.14:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.



Вывод команды для добавления manager - узла в кластер swarm:

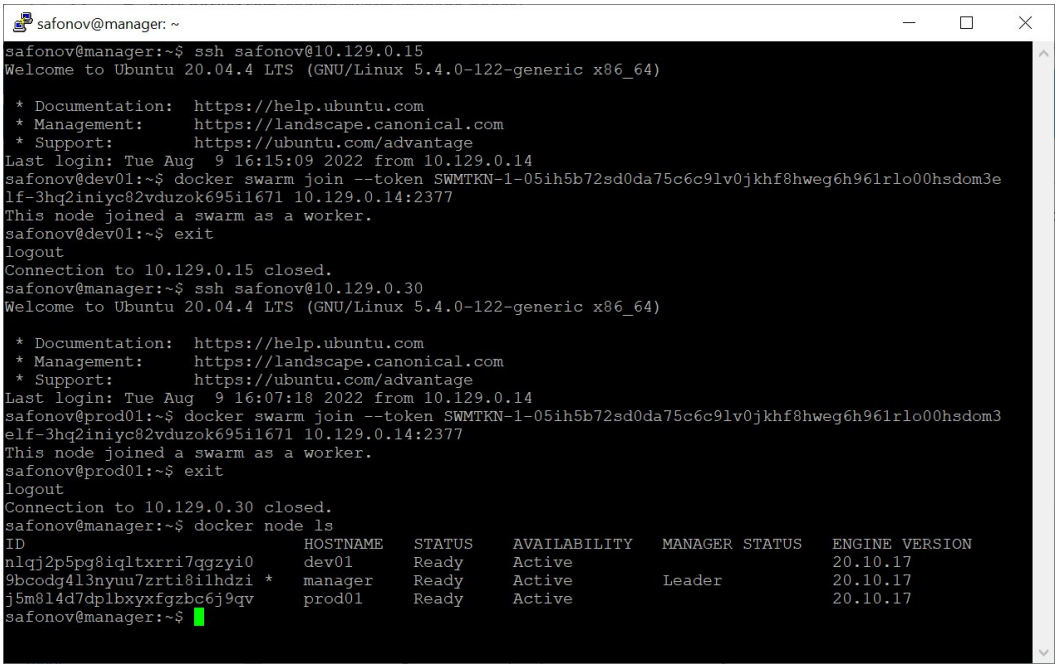
```
$ docker swarm join-token manager
```

Вывод команды для добавления worker - узла в кластер swarm:

```
$ docker swarm join-token worker
```

- Подключение worker - узла в кластер swarm:

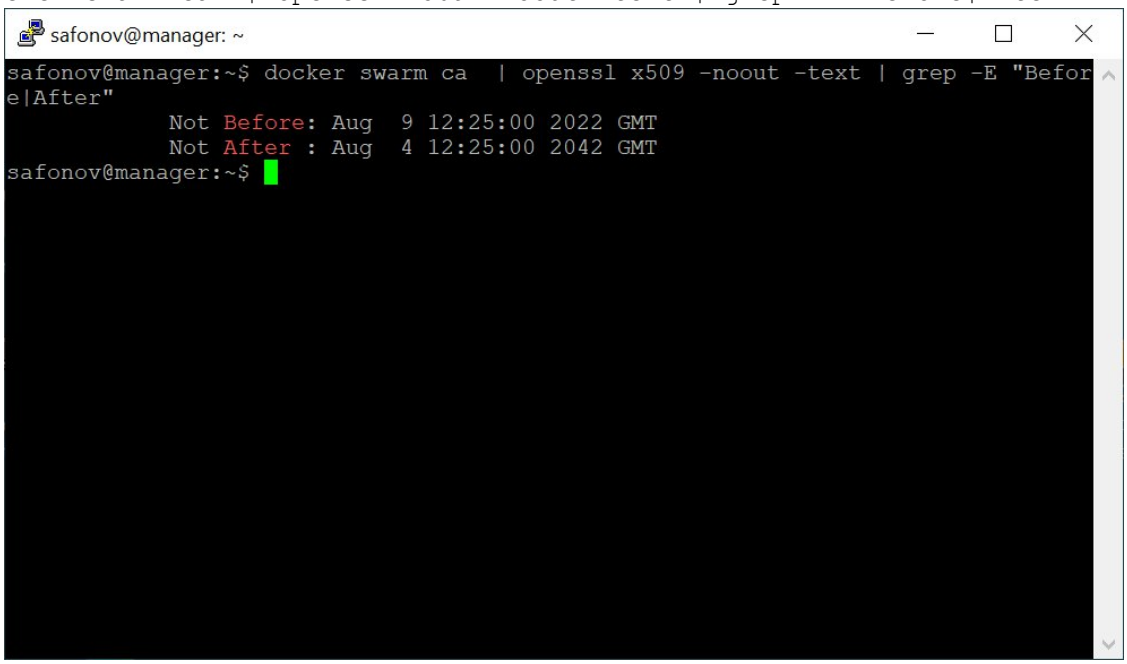
```
$ docker swarm join --token SWMTKN-1-
05ih5b72sd0da75c6c9lv0jkhf8hweg6h961rlo00hsdom3elf-3hq2iniyc82vduzok695i1671
10.129.0.14:2377
```



Управление узлами в кластере см. в https://docs.docker.com/engine/swarm/manage-nodes/

- Просмотр срока действия ca-сертификата в кластере Swarm:

```
$ docker swarm ca  | openssl x509 -noout -text | grep -E "Before|After"
```



Развертывание контейнерных приложений

Стратегия развертывания приложения

При развертывании приложения необходимо предусматривать разделение инфраструктуры на кластеры и узлы, которые будут относиться к одной из определенных сред:
dev (Development) - Среда разработки (при наличии процесса разработки)
test (Testing) - Тестовая среда
prod (Production) - Производственная среда

Соответственно узлы управления (manager) и рабочие узлы (worker) кластера производственной среды должны быть полностью изолированы от других сред.

Все сервисы для тестовой среды, включая реестр образов, должны быть настроены и развернуты аналогично как в производственной среде.

Развертывание Portainer CE для управления кластером Swarm

Стек Portainer состоит из нескольких микро-сервисов:

- агент для управления кластером, размещается на всех узлах кластера Swarm;

- приложение Portainer размещается на узлах управления (manager) кластера Swarm.

https://docs.portainer.io/start/install/server/swarm/linux

На узле, где разворачивается сервис должна быть следующая структура папок:

/opt/projects/portainer/certs/ - сертификат palsys-cert.cer и закрытый ключ palsys-cert.key;

Файл `portainer-agent-stack.yml` для настройки среды микро-сервиса Portainer:

```
version: '3.9'

services:
  agent:
    image: portainer/agent:2.14.2
    environment:
      # REQUIRED: Should be equal to the service name prefixed by "tasks."
when
      # deployed inside an overlay network
      AGENT_CLUSTER_ADDR: tasks.agent
      # AGENT_PORT: 9001
      # LOG_LEVEL: debug
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /var/lib/docker/volumes:/var/lib/docker/volumes
    networks:
      - portainer_network
    deploy:
      restart_policy:
        condition: any
        delay: 5s
```

```
        max_attempts: 3
        window: 30s
      mode: global
      placement:
        constraints: [node.platform.os == linux]

  portainer:
    image: portainer/portainer-ce:2.14.2
    command: -H tcp://tasks.agent:9001 --tlsskipverify --ssl --sslcert
/run/secrets/palsys-cert.cer --sslkey /run/secrets/palsys-cert.key
    ports:
      - "9443:9443"
      - "9000:9000"
    volumes:
      - portainer_data:/data
    networks:
      - portainer_network
    secrets:
      - palsys-cert.cer
      - palsys-cert.key
    deploy:
      restart_policy:
        condition: any
        delay: 5s
        max_attempts: 3
        window: 30s
      mode: replicated
      replicas: 1
      placement:
        constraints: [node.role == manager]

networks:
  portainer_network:
    driver: overlay
    attachable: true
    external: true

secrets:
  palsys-cert.cer:
    file: ./certs/palsys-cert.cer
  palsys-cert.key:
    file: ./certs/palsys-cert.key

volumes:
  portainer_data:
```

Разворачиваем стек Portainer на кластере:

```
$ docker stack deploy -c portainer-agent-stack.yml portainer
```

Выгружаем стек Portainer с кластера:

```
$ docker stack rm portainer
```

Установка и настройка реестра образов Registry

Размещаем сервис для реестра образов на узле prod01 кластера для продуктовой среды.
Как один из вариантов определения расположения сервиса на узлах могут использоваться метки:

```
# для узла prod01 ставим метку что на нем будет располагаться реестр образов:
$ docker node update --label-add registry=true prod01
# затем добавляем в compose-файл условие для развертывания сервиса:
# deploy:
#   placement:
#     constraints: [node.labels.registry == true]
```

Используем базовую аутентификацию (Basic Authentication) и использованием утилиты htpasswd от Apache.

```
# ищем пакет с утилитой
$ apt-file search htpasswd
# устанавливаем пакет
$ sudo apt install apache2-utils
# формируем логин и пароль для входа в реестр
$ htpasswd -nb -B **login** **password** > ./htpasswd
```

На узле, где разворачивается сервис должна быть следующая структура папок:

/opt/projects/registry/auth/ - папка с файлом `htpasswd` с логином и паролем;

/opt/projects/registry/certs/ - сертификат `palsys-cert.cer` и закрытый ключ `palsys-cert.key`;

На самом узле prod01, где развернут сервис, должна быть следующая структура папок:

/opt/data/registry - папка для хранения образов.

Файл registry-stack.yml для настройки среды микро-сервиса Registry:

```
version: '3.3'
services:
  registry:
    image: registry:2
    ports:
      - 5005:443
    environment:
      REGISTRY_AUTH: htpasswd
      REGISTRY_AUTH_HTPASSWD_REALM: Registry Realm
      REGISTRY_AUTH_HTPASSWD_PATH: /run/secrets/htpasswd
      REGISTRY_HTTP_ADDR: 0.0.0.0:443
      REGISTRY_HTTP_TLS_CERTIFICATE: /run/secrets/palsys-cert.cer
      REGISTRY_HTTP_TLS_KEY: /run/secrets/palsys-cert.key
```

```
    volumes:
      - /opt/data/registry:/var/lib/registry
    networks:
        registry-network:
          aliases:
            - registry.local
    secrets:
      - htpasswd
      - palsys-cert.cer
      - palsys-cert.key
    deploy:
          #mode: replicated
          replicas: 1
          #endpoint mode: dnsrr
          # service resource management
          resources:
            # Hard limit - Docker does not allow to allocate more
            limits:
              cpus: '0.25'
              memory: 256M
            # Soft limit - Docker makes best effort to return to it
            reservations:
              cpus: '0.25'
              memory: 512M
          # service restart policy
          restart_policy:
            condition: on-failure
            delay: 5s
            max_attempts: 3
            window: 120s
          # service update configuration
          update_config:
            parallelism: 1
            delay: 10s
            failure_action: continue
            monitor: 60s
            max_failure_ratio: 0.3
          # placement constraint - in this case on node 'prod01' only
          placement:
            constraints: [node.hostname == prod01]
networks:
  registry-network:
    driver: overlay

secrets:
  htpasswd:
    file: ./auth/htpasswd
  palsys-cert.cer:
    file: ./certs/palsys-cert.cer
  palsys-cert.key:
    file: ./certs/palsys-cert.key
```

Разворачиваем стек реестра на кластере:

```
 $ docker stack deploy -c registry-stack.yml registry
```

Выгружаем стек реестра с кластера:

```
$ docker stack rm registry
```

Установка СУБД Postgres 14

Стек postgres состоит из нескольких микро-сервисов:

- СУБД Postgres 14;
- GUI pgAdmin4 для управления СУБД.

Размещаем стек postgres на узле prod01 кластера в продуктовой среде.

На узле, где разворачивается сервис должна быть следующая структура папок и файлов:
/opt/projects/postgres/auth/ - папка с файлами `pgpasswd` и `pgadminpasswd` с паролями;
/opt/projects/postgres/certs/ - сертификат palsys-cert.cer и закрытый ключ palsys-cert.key;

На самом узле prod01, где развернут сервис, должна быть следующая структура папок:
/opt/data/postgres - папка для хранения БД;
/opt/data/postgres/pg-sql14 - папка для хранения дампов;
/opt/data/pgadmin/servers.json - файл импорта/экспорта настроек серверов.

Файл postgres-stack.yml для настройки среды микро-сервисов Postgres 14 и pgAdmin 4:

```
version: "3.9"
services:
  postgres:
    image: postgres:14
    command:
      - "postgres"
      - "-c"
      - "max_connections=50"
      - "-c"
      - "shared_buffers=1GB"
      - "-c"
      - "effective_cache_size=4GB"
      - "-c"
      - "work_mem=16MB"
      - "-c"
      - "maintenance_work_mem=512MB"
      - "-c"
      - "random_page_cost=1.1"
      - "-c"
      - "temp_file_limit=10GB"
      - "-c"
```

```
    - "log_min_duration_statement=200ms"
    - "-c"
    - "idle_in_transaction_session_timeout=10s"
    - "-c"
    - "lock_timeout=1s"
    - "-c"
    - "statement_timeout=60s"
    - "-c"
    - "shared_preload_libraries=pg_stat_statements"
    - "-c"
    - "pg_stat_statements.max=10000"
    - "-c"
    - "pg_stat_statements.track=all"
  environment:
    POSTGRES_USER: "postgres"
    POSTGRES_PASSWORD: /run/secrets/pgpasswd
    PGDATA: "/var/lib/postgresql/data/pgdata"
  volumes:
    - /opt/data/postgres:/var/lib/postgresql/data
  ports:
    - "5432:5432"
  deploy:
    resources:
      limits:
        cpus: '1'
        memory: 4G
    placement:
      constraints: [node.hostname == prod01]
  networks:
    - postgres_network
  secrets:
    - pgpasswd

pgadmin:
  image: dpage/pgadmin4:latest
  environment:
    PGADMIN_DEFAULT_EMAIL: "user@example.ru"
    PGADMIN_DEFAULT_PASSWORD: /run/secrets/pgadminpasswd
    PGADMIN_CONFIG_SERVER_MODE: "False"
    PGADMIN_ENABLE_TLS: "True"
  volumes:
    - ./certs/palsys-cert.cer:/certs/server.cert
    - ./certs/palsys-cert.key:/certs/server.key
    - /opt/data/postgres/pgsql-14:/usr/local/pgsql-14
    - /opt/data/pgadmin:/var/lib/pgadmin
    - /opt/data/pgadmin/servers.json:/pgadmin4/servers.json
  ports:
    - "5050:443"
  deploy:
    resources:
      limits:
        cpus: '0.5'
        memory: 1G
    placement:
      constraints: [node.hostname == prod01]
  networks:
    - postgres_network
```

```
    secrets:
      - pgadminpasswd

networks:
  postgres_network:
    driver: overlay

secrets:
  pgpasswd:
    file: ./auth/pgpasswd
  pgadminpasswd:
    file: ./auth/pgadminpasswd
```

Создаем оверлейную сеть postgres_network для стека postgres:

```
$ docker network create --driver overlay postgres_network
```

Разворачиваем стек postgres на кластере:

```
$ docker stack deploy -c postgres-stack.yml postgres
```

Выгружаем стек postgres с кластера:

```
$ docker stack rm postgres
```

Установка приложения АСУ МС 7

Стек приложения АСУ МС 7 состоит из нескольких микро-сервисов:

- WebHost - сервис REST API и интерфейсом OpenApi (Swagger);
- IdentityServer - сервис аутентификации и авторизации;
- SPA-Metr7 - одностраничное (SPA) web-приложение с http-сервером Nginx.

Размещаем стек приложения АСУ МС 7 на узле prod01 кластера в продуктовой среде.
На узле, где разворачивается сервис должна быть следующая структура папок и файлов:
/opt/projects/metr7/conf/ - папка с конфигурационными настройками;
/opt/projects/metr7/conf/Module.Core/connection.json - настройки для соединения с БД;
/opt/projects/metr7/conf/WebHost/appsettings.json - настройки приложения;
/opt/projects/metr7/conf/WebHost/appsettings.Development.json - настройки приложения для среды разработки;
/opt/projects/metr7/conf/IdentityServer/appsettings.Production.json - настройки приложения для производственной среды;
/opt/projects/metr7/conf/IdentityServer/appsettings.json - настройки сервиса аутентификации и авторизации;
/opt/projects/metr7/conf/IdentityServer/appsettings.Development.json - настройки сервиса аутентификации и авторизации для среды разработки;
/opt/projects/metr7/conf/IdentityServer/appsettings.Production.json - настройки сервиса аутентификации и авторизации для производственной среды;
/opt/projects/metr7/conf/conf/SPA-Metr7/nginx/nginx.conf - настройки для http-сервера

Nginx;
/opt/projects/registry/certs/ - сертификат `palsys-cert.pfx`.

На самом узле prod01, где развернуты сервисы, должна быть следующая структура папок:
/opt/projects/metr7/logs - папка для логов приложений;
/opt/projects/metr7/logs/nginx - папка для логов http-сервера Nginx;

Файл metr7-stack.yml для настройки среды АСУ МС 7:

```
services:
  webhost:
    image: web.palitra-system.ru:5005/palsysrestapiwebhost
    depends_on:
      - identityserver.web.palitra-system.ru
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
      - ASPNETCORE_URLS=http://+:5000;https://+:5001
      - ASPNETCORE_Kestrel__Certificates__Default__Path=/certs/palsys-
cert.pfx
      - ASPNETCORE_Kestrel__Certificates__Default__Password=**password**
      - IDENTITY_AUTHORITY=https://identityserver.web.palitra-system.ru:5002
      - OpenIdConnectAuthority=https://identityserver.palitra-system.ru:5002
    ports:
      - "5000:5000"
      - "5001:5001"
    volumes:
      - ./conf/Module.Core/connection.json:/app/connection.json
      - ./conf/WebHost/appsettings.json:/app/appsettings.json

- ./conf/WebHost/appsettings.Development.json:/app/appsettings.Development.js
on

- ./conf/WebHost/appsettings.Production.json:/app/appsettings.Production.json
      - ./certs:/certs/
      - ./logs:/app/Logs/
    networks:
      frontend-network:
      backend-network:
        aliases:
          - webhost.web.palitra-system.ru
    deploy:
      placement:
        constraints: [node.hostname == prod01]

  identityserver:
    image: web.palitra-system.ru:5005/palsysrestapiidentityserver
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
      - ASPNETCORE_URLS=https://+:5002
      - ASPNETCORE_Kestrel__Certificates__Default__Path=/certs/palsys-
cert.pfx
      - ASPNETCORE_Kestrel__Certificates__Default__Password=**password**
      - IDENTITY_ISSUER=https://identityserver.web.palitra-system.ru:5002
    ports:
      - "5002:5002"
```

```
    volumes:
      - ./conf/Module.Core/connection.json:/app/connection.json
      - ./conf/IdentityServer/appsettings.json:/app/appsettings.json

- ./conf/IdentityServer/appsettings.Development.json:/app/appsettings.Develop
ment.json

- ./conf/IdentityServer/appsettings.Production.json:/app/appsettings.Producti
on.json
      - ./certs:/certs/
      - ./logs:/app/Logs/
    networks:
      frontend-network:
      backend-network:
        aliases:
          - identityserver.web.palitra-system.ru
    deploy:
      placement:
        constraints: [node.hostname == prod01]

  spa-metr7:
    image: web.palitra-system.ru:5005/palsysrestapispametr7
    depends_on:
      - webhost.web.palitra-system.ru
    ports:
      - "5003:443"
    volumes:
      - ./conf/SPA-Metr7/nginx/nginx.conf:/etc/nginx/conf.d/default.conf
      - ./certs:/etc/nginx/certs/
      - ./logs/nginx:/var/log/nginx/
    networks:
      backend-network:
        aliases:
          - spa-metr7.web.palitra-system.ru
    deploy:
      placement:
        constraints: [node.hostname == prod01]

networks:
  frontend-network:
    name: postgres_network
    external: true
  backend-network:
    driver: overlay
    attachable: true
```

Для развертывания стека АСУ МС 7 на кластере выполнить команду:

```
$ docker stack deploy -c metr7-stack.yml --with-registry-auth metr7
```

Для выгрузки стека АСУ МС 7 с кластера выполнить команду:

```
$ docker stack rm metr7
```

Техническая поддержка

По всем вопросам, связанным с эксплуатацией АСУ МС пользователи могут обратиться в службу сопровождения и технической поддержки.

Услуги оказываются ежедневно с 8.00 до 18.00 по московскому времени кроме выходных и праздничных дней. Обращения пользователей принимаются по следующим каналам:

- электронная почта support@palitra-system.ru
- портал технической поддержки ПАЛИТРА СИСТЕМ. Ссылка на портал https://palitra-system.atlassian.net/servicedesk/customer/portal/1
- официальный сайт ПАЛИТРА СИСТЕМ раздел «Поддержка». Ссылка https://palitra-system.ru/tekhpodderzhka/
- Интерфейс программного обеспечения – раздел «Справка=> Техническая поддержка»
- телефон +7(499)754-10-04