

**ПРОГРАММНЫЙ ПРОДУКТ
ЦИФРОВОЙ ШТАБ**

Руководство по развертыванию

2024

АННОТАЦИЯ

Данный документ предназначен для специалистов, выполняющих администрирование программного продукта «Цифровой штаб» (далее – **Программный продукт**), и включает описание действий по установке и настройке **Программного продукта** в операционных системах, поддерживающих систему управления контейнерами Docker (Astra Linux, РЕД ОС, Ubuntu, Debian и т. д.).

СОДЕРЖАНИЕ

1	Требования к программно-аппаратной части и персоналу	5
1.1	Требования к серверной части.....	5
1.2	Требования к квалификации персонала	6
2	Установка и Развертывание программного продукта	7
2.1	Установка кластера kubernetes.....	7
2.1.1	Настройка мастер нод.....	7
2.1.2	Настройка рабочих нод	10
2.1.3	Подключение рабочей ноды к кластеру	12
2.2	Установка Kafka	12
2.2.1	Установка zookeeper	12
2.2.2	Установка kafka	14
2.3	Установка Schema Registry	15
2.4	Установка Postgresql	17
2.5	Установка MongoDB	18
2.6	Установка Keycloak.....	20
2.7	Установка Minio	21
2.8	Установка mediamtx.....	23
2.9	Установка сервисов Цифрового Штаба	27
2.9.1	Установка реестра образов и загрузка образов	27
2.9.2	Сервис dhq-backend-for-frontend-ms	31
2.9.3	Сервис dhq-bastion-adapter-ms.....	35
2.9.4	Сервис dhq-bpm-ms.....	37
2.9.5	Сервис dhq-data-gateway-ms.....	39
2.9.6	Сервис dhq-dictionaries-update-ms	41
2.9.7	Сервис dhq-enrichment-ms	43
2.9.8	Сервис dhq-event-processing-ms	45

Программный продукт
Цифровой штаб
Руководство по развертыванию

2.9.9	Сервис dhq-frontend	47
2.9.10	Сервис dhq-integration-ms.....	49
2.9.11	Сервис dhq-messages-ms.....	52
2.9.12	Сервис dhq-metadata	54
2.9.13	Сервис dhq-notification-ms.....	56
2.9.14	Сервис dhq-rdm-ms	59
2.9.15	Сервис dhq-video-processing-ms	62

1 ТРЕБОВАНИЯ К ПРОГРАММНО-АППАРАТНОЙ ЧАСТИ И ПЕРСОНАЛУ

1.1 Требования к серверной части

Все компоненты **Программного продукта** устанавливаются на один виртуальный либо физический сервер под управлением ОС, поддерживающей систему управления контейнерами Docker.

Минимальные требования к аппаратной части:

- Процессор: не менее 2 ГГц, 22 ядра.
- Оперативная память: не менее 44 Гб.
- Дисковое пространство: не менее 6544 Гб свободного дискового пространства.

Требования к программной части:

- ОС, поддерживающие систему управления контейнерами Docker (Astra Linux, РЕД ОС, Ubuntu, Debian и т. д.).
- СУБД: PostgreSQL версии 15.2.

Список программных комплексов, доступ к которым **Программный продукт** должен иметь монополярный доступ:

- инфраструктурный домен **Программного продукта**;
- платформа виртуализации;
- система учета сетевых конфигураций;
- система управления конфигурациями;
- хранилище секретов.

Под монополярным доступом подразумевается, что никакие другие системы или пользователи не используют указанные системы.

1.2 Требования к квалификации персонала

Администратор **Программного продукта** должен обладать квалификацией, обеспечивающей:

- базовые навыки администрирования ОС семейства Linux (настройка репозитория, системные настройки);
- базовые навыки работы с Docker, Docker Compose;
- базовые навыки работы с СУБД PostgreSQL.
- базовые навыки работы со средствами автоматизации Ansible;
- базовые навыки работы со средствами виртуализации;
- базовые навыки работы с сетевой инфраструктурой;
- базовые навыки работы со средствами мониторинга ИБ;
- базовые навыки работы со средствами мониторинга ИТ;
- базовые навыки работы со службами каталогов.

2 УСТАНОВКА И РАЗВЕРТЫВАНИЕ ПРОГРАММНОГО ПРОДУКТА

2.1 Установка кластера kubernetes

2.1.1 Настройка мастер нод

Переключаемся на root

```
1: su -
```

отключаем swap и selinux

```
2: swapoff -a && sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab  
3: setenforce && sed -I 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
```

создайте файл для автозагрузки модулей ядра, необходимых для работы сервиса cri-o:

```
1: cat <<EOF> /etc/modules-load.d/crio.conf  
overlay  
br_netfilter  
EOF
```

Загрузите модули в ядро

```
1: cat <<EOF> /etc/modules-load.d/crio.conf  
overlay  
br_netfilter  
EOF
```

Проверьте, что данные модули работают:

```
1: lsmod | egrep "br_netfilter|overlay"
```

Создайте конфигурационный файл для работы сети внутри kubernetes:

```
1: cat <<EOF> /etc/sysctl.d/99-kubernetes-cri.conf  
2: net.bridge.bridge-nf-call-iptables = 1  
3: net.ipv4.ip_forward = 1  
4: net.bridge.bridge-nf-call-ip6tables = 1  
5: EOF
```

Примените параметры командой:

```
1: sysctl --system
```

Установите необходимые пакеты:

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: dnf install kubernetes kubernetes-kubeadm cri-o cri-tools -y
```

Настройте проброс портов в iptables:

```
1: iptables -P FORWARD ACCEPT
```

Установите настройки по умолчанию для конфигурации контейнера:

```
1: sed -i '/^[crio\.runtime\]/a seccomp_profile = "\/etc\/containers\/policy.json"'\n    /etc/crio/crio.conf
```

Запустите службу cri-o и добавьте ее в автозагрузку:

```
1: systemctl enable --now cri-o
```

Добавьте службу kubelet в автозагрузку:

```
1: systemctl enable kubelet.service
```

Загрузите образы контейнеров, необходимых kubeadm для инициализации ноды кластера:

```
1: kubeadm config images pull
```

Запустите инициализацию мастер-ноды в одноранговом кластере. Данная команда выполнит начальную настройку и подготовку основного узла кластера. Ключ `--pod-network-cidr` задает адрес внутренней подсети для вашего кластера.

```
1: kubeadm init --pod-network-cidr=10.244.0.0/16
```

В случае успешной инициализации, в конце вывода команды будет отображаться примерно следующее:

Программный продукт
Цифровой штаб
Руководство по развертыванию

```

1: Your Kubernetes control-plane has initialized successfully!
2:
3: To start using your cluster, you need to run the following as a regular user:
4:
5:   mkdir -p $HOME/.kube
6:   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
7:   sudo chown $(id -u):$(id -g) $HOME/.kube/config
8:
9: Alternatively, if you are the root user, you can run:
10:
11:   export KUBECONFIG=/etc/kubernetes/admin.conf
12:
13: You should now deploy a pod network to the cluster.
14: Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
15:   https://kubernetes.io/docs/concepts/cluster-administration/addons/
16:
17: Then you can join any number of worker nodes by running the following on each as root:
18:
19: kubectl join 10.81.186.86:6443 --token sj5u6t.m8w118o26fhj4gml \
20: --discovery-token-ca-cert-hash
   sha256:53cbbbf46036dadd55d348eea7b7585d7e7e048a554f7b684a5e5322f9ba498

```

Настройте параметры управления кластером. Настройку можно выполнить как для локального пользователя, так и для суперпользователя root.

Для управления кластером от имени локального пользователя выполните команды:

```

1: mkdir /home/$USER/.kube
2: cp -i /etc/kubernetes/admin.conf /home/$USER/.kube/config
3: chown $USER. /home/$USER/.kube /home/$USER/.kube/config

```

Для управления кластером от имени суперпользователя root выполните команды:

```

1: echo "export KUBECONFIG=/etc/kubernetes/admin.conf" >> /root/.bashrc
2: source .bashrc
3: export KUBECONFIG=/etc/kubernetes/admin.conf

```

Установите внутреннюю конфигурацию сети в кластере (в примере используется calico).

```

1: kubectl                                apply                                -f
   https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml

```

Проверьте список и статус всех подов в кластере:

```

1: kubectl get pod -n kube-system

```

Выполните дополнительную настройку:

```

1: sudo hostnamectl set-hostname masternode

```

Проверьте изменения:

```
1: hostname && hostname -I
2: masternode10.81.186.86 10.81.186.67 10.85.0.1 1100:200::1
```

В файл /etc/hosts внесите данные о master и worker:

```
1:sudo nano /etc/hosts
```

```
2: 10.81.186.86 masternode
3: 10.81.186.106 worker
```

Для вывода команды присоединения worker к кластеру выполните:

```
1: kubeadm token create --print-join-command
2:
3: kubeadm join 10.81.186.86:6443 --token nxuiy0.xt5ts9u4ynhdoowr --discovery-token-ca-cert-hash sha256:53cbbbf46036dadd55d348eea7b7585d7e7e048a554f7b684a5e5322f9ba498
```

2.1.2 Настройка рабочих нод

Переключаемся на root

```
4: su -
```

отключаем swap и selinux

```
5: swapoff -a && sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
6: setenforce && sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
```

создайте файл для автозагрузки модулей ядра, необходимых для работы сервиса cri-o:

```
7: cat <<EOF> /etc/modules-load.d/crio.conf
  overlay
  br_netfilter
  EOF
```

Загрузите модули в ядро

```
8: cat <<EOF> /etc/modules-load.d/crio.conf
  overlay
  br_netfilter
  EOF
```

Проверьте, что данные модули работают:

```
9: lsmod | egrep "br_netfilter|overlay"
```

Создайте конфигурационный файл для работы сети внутри kubernetes:

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
10: cat <<EOF> /etc/sysctl.d/99-kubernetes-cri.conf
11: net.bridge.bridge-nf-call-iptables = 1
12: net.ipv4.ip_forward = 1
13: net.bridge.bridge-nf-call-ip6tables = 1
14: EOF
```

Примените параметры командой:

```
15: sysctl --system
```

Установите необходимые пакеты:

```
16: dnf install kubernetes kubernetes-kubeadm cri-o cri-tools -y
```

Настройте проброс портов в iptables:

```
17: iptables -P FORWARD ACCEPT
```

Установите настройки по умолчанию для конфигурации контейнера:

```
18: sed -i '/^\[crio\.runtime\]/a seccomp_profile = "\/etc\/containers\/policy.json"'
    /etc/crio/crio.conf
```

Запустите службу cri-o и добавьте ее в автозагрузку:

```
19: systemctl enable --now cri-o
```

Добавьте службу kubelet в автозагрузку:

```
20: systemctl enable kubelet.service
```

Выполните дополнительную настройку:

```
21: sudo hostnamectl set-hostname worker
```

Проверьте изменения:

```
22: hostname && hostname -I
23: worker10.81.186.106 10.81.186.142
```

В файл /etc/hosts внесите данные о master и worker:

```
24: sudo nano /etc/hosts
```

```
25: 10.81.186.86 masternode
26: 10.81.186.106 worker
```

2.1.3 Подключение рабочей ноды к кластеру

На master-ноде получите команду присоединения worker:

```
1: kubectl token create --print-join-command
2:
3: kubectl join 10.81.186.86:6443 --token nxuiy0.xt5ts9u4ynhdoowr --discovery-token-ca-cert-hash sha256:53cbbbf46036dadd55d348eea7b7585d7e7e048a554f7b684a5e5322f9ba498
```

На worker выполните полученную команду:

```
1: kubectl join 10.81.186.86:6443 --token nxuiy0.xt5ts9u4ynhdoowr --discovery-token-ca-cert-hash sha256:53cbbbf46036dadd55d348eea7b7585d7e7e048a554f7b684a5e5322f9ba498
2:
3: [preflight] Running pre-flight checks
4: [preflight] Reading configuration from the cluster...
5: [preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
6: [kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
7: [kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
8: [kubelet-start] Starting the kubelet
9: [kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...
10:
11: This node has joined the cluster:
12: * Certificate signing request was sent to apiserver and a response was received.
13: * The Kubelet was informed of the new secure connection details.
14:
15: Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

2.2 Установка Kafka

2.2.1 Установка zookeeper

Установите Java в системе:

```
1: dnf install java-11-openjdk -y
```

Скачайте архив zookeeper и распакуйте

```
1: sudo wget https://www.apache.org/dist/zookeeper/zookeeper-3.6.2/apache-zookeeper-3.6.2-  
bin.tar.gz  
2: sudo tar -xvzf apache-zookeeper-3.6.2-bin.tar.gz -C /opt  
3: sudo mv /opt/apache-zookeeper-3.6.2-bin /opt/zookeeper
```

Создайте конфигурационный файл Zookeeper:

```
1: sudo cp /opt/zookeeper/conf/zoo_sample.cfg /opt/zookeeper/conf/zoo.cfg
```

Добавьте переменные окружения в файл /etc/environment:

```
1: echo "export ZOOKEEPER_HOME=/opt/zookeeper" | sudo tee -a /etc/environment  
2: echo "export PATH=\$ZOOKEEPER_HOME/bin:\$PATH" | sudo tee -a /etc/environment
```

Откройте порты:

```
1: sudo firewall-cmd --zone=public --add-port=2181/tcp --add-port=2888/tcp --add-port=3888/tcp  
-permanent  
2: sudo firewall-cmd --reload
```

Создайте пользователя и смените владельца

```
1: sudo useradd zookeeper --system --no-create-home  
2: sudo chown -R zookeeper:zookeeper /opt/zookeeper
```

Создайте юнит для запуска сервиса

```
1: cat <<EOF> /etc/systemd/system/zookeeper.service
2: [Unit]
3: Description=Apache Zookeeper
4: Documentation=http://zookeeper.apache.org
5: Requires=network.target
6: After=network.target
7:
8: [Service]
9: Type=forking
10: EnvironmentFile=/etc/default/zookeeper
11: ExecStart=/usr/share/zookeeper/bin/zkServer.sh start
12: ExecStop=/usr/share/zookeeper/bin/zkServer.sh stop
13: SuccessExitStatus=143
14: Restart=on-failure
15: User=zookeeper
16: Group=zookeeper
17:
18: [Install]
19: WantedBy=multi-user.target
20: Alias=zookeeper.service
21: EOF
```

Запустите сервис

```
1: sudo systemctl enable --now zookeeper
```

2.2.2 Установка kafka

Установите Java в системе:

```
2: dnf install java-11-openjdk -y
```

Скачайте архив zookeeper и распакуйте

```
3: sudo wget https://archive.apache.org/dist/kafka/3.2.1/kafka_2.13-3.2.1.tgz
4: sudo tar -xvzf kafka_2.13-3.2.1.tgz -C /opt
5: sudo mv /opt/kafka_2.13-3.2.1 /opt/kafka
```

Отредактируйте файл конфигурации Kafka config/server.properties, укажите свой broker.id, listeners и advertised.listeners.

```
1: nano config/server.properties
2: broker.id=0
3: listeners=PLAINTEXT://localhost:9092
4: advertised.listeners=PLAINTEXT://your-hostname:9092
```

Создайте юнит для запуска сервиса

```
1: cat <<EOF> /etc/systemd/system/kafka.service
2: [Unit]
3: Description=Apache Kafka
4: Documentation=http://kafka.apache.org/documentation.html
5: Requires=network.target
6: After=network.target
7:
8: [Service]
9: Type=simple
10: StandardOutput=null
11: Environment="KAFKA_HEAP_OPTS=-Xms1G -Xmx1G"
12: Environment="LOG_DIR=/var/log/kafka"
13: ExecStart=/opt/kafka/bin/kafka-server-start.sh /etc/kafka/server.properties
14: ExecStop=/opt/kafka/bin/kafka-server-stop.sh
15: User=kafka
16: Group=kafka
17: Restart=on-failure
18: LimitNOFILE=infinity
19: SuccessExitStatus=143
20:
21: [Install]
22: WantedBy=multi-user.target
23: Alias=kafka.service
24: EOF
```

Откройте порты:

```
25: sudo firewall-cmd --zone=public --add-port=9092/tcp --permanent
26: sudo firewall-cmd --reload
```

Создайте пользователя и смените владельца

```
27: sudo useradd kafka --system --no-create-home
28: sudo chown -R kafka:kafka /opt/kafka
```

Запустите сервис

```
1: sudo systemctl enable --now kafka
```

2.3 Установка Schema Registry

Установите Java в системе:

```
2:dnf install java-11-openjdk -y
```

Скачайте архив schema_registry и распакуйте

```
3: sudo wget https://packages.confluent.io/archive/6.0/confluent-6.0.1.tar.gz
4: sudo tar -xvzf confluent-6.0.1.tar.gz -C /opt
```

Отредактируйте `/etc/schema-registry/log4j.properties`

```
1: log4j.rootLogger=INFO, stdout, file
2:
3: log4j.appender.stdout=org.apache.log4j.ConsoleAppender
4: log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
5: log4j.appender.stdout.layout.ConversionPattern=[%d] %p %m (%c:%L)%n
6:
7: log4j.logger.kafka=ERROR, stdout
8: log4j.logger.org.apache.zookeeper=ERROR, stdout
9: log4j.logger.org.apache.kafka=ERROR, stdout
10: log4j.additivity.kafka.server=false
11:
12: log4j.appender.file=org.apache.log4j.RollingFileAppender
13: log4j.appender.file.maxBackupIndex=10
14: log4j.appender.file.maxFileSize=100MB
15: log4j.appender.file.File=${schema-registry.log.dir}/schema-registry.log
16: log4j.appender.file.layout=org.apache.log4j.PatternLayout
17: log4j.appender.file.layout.ConversionPattern=[%d] %p %m (%c)%n
```

Отредактируйте `/etc/schema-registry/schema-registry.properties`

```
18: listeners= http://0.0.0.0:8081
19: kafkastore.bootstrap.servers=PLAINTEXT://<HOST-WITH-KAFKA>:9092
20: kafkastore.topic=_schemas
21: debug=False
22: schema.compatibility.level=full_transitive
```

Создайте юнит для запуска сервиса

Программный продукт
Цифровой штаб
Руководство по развертыванию

```

23: cat <<EOF> /etc/systemd/system/schema_registry.service
24: [Unit]
25: Description=schema registry
26: After=network.target
27:
28: [Service]
29: PrivateTmp=True
30: User=schema_registry
31: Group=schema_registry
32: LimitNOFILE=32768
33: Environment='SCHEMA_REGISTRY_LOG4J_OPTS=-Dlog4j.configuration=file:/etc/schema-
registry/log4j.properties -Dschema-registry.log.dir=/var/log/schema_registry'
34: Environment='SCHEMA_REGISTRY_HEAP_OPTS=-Xmx256M'
35: Environment='SCHEMA_REGISTRY_JVM_PERFORMANCE_OPTS=-server -XX:+UseG1GC -
XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=35 -
XX:+ExplicitGCInvokesConcurrent -Djava.awt.headless=true'
36: Environment='SCHEMA_REGISTRY_JMX_OPTS=-Dcom.sun.management.jmxremote -
Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false'
37: Environment='JMX_PORT=5555'
38:
39:
40: ExecStart=/opt/confluent-6.0.1/bin/schema-registry-start /etc/schema-registry/schema-
registry.properties
41: ExecStop=/opt/confluent-6.0.1/bin/schema-registry-stop
42:
43: ExecReload=/bin/kill -HUP $MAINPID
44: KillSignal=SIGTERM
45: Restart=on-failure
46:
47: [Install]
48: WantedBy=multi-user.target
49: EOF

```

Откройте порты:

```

50: sudo firewall-cmd --zone=public --add-port=8081/tcp --permanent
51: sudo firewall-cmd --reload

```

Создайте пользователя и смените владельца

```

52: sudo useradd schema_registry --system --no-create-home
53: sudo chown -R schema_registry:schema_registry /opt/confluent*

```

Запустите сервис

```

1: sudo systemctl enable --now schema_registry

```

2.4 Установка Postgresql

Переключаемся на root

```
1: su -
```

Для установки postgresql выполните команду:

```
1: dnf install postgresql15-server
```

Далее необходимо произвести инициализацию базы данных postgresql:

```
1: postgresql-15-setup initdb
```

После успешной инициализации запустите службу postgresql и добавьте ее в автозагрузку:

```
1: systemctl enable postgresql-15.service --now
```

Убедитесь, что служба запущена:

```
1: systemctl status postgresql-15.service
```

В статусе должно отображаться active (running).

2.5 Установка MongoDB

Установите mongodb:

```
1: sudo dnf install mongodb-org-server-6.0* -y
```

Создайте каталог для данных и установите владельцем mongod:

```
1: sudo mkdir -p /data/db
2: sudo chown -R mongod:mongod /data/db
```

Конфигурационный файл mongodb:

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: sudo cat <<EOF> /etc/mongod.conf
2: net:
3:   bindIp: 0.0.0.0
4:   ipv6: false
5:   maxIncomingConnections: 65536
6:   port: 27017
7:
8: processManagement:
9:   timeZoneInfo: /usr/share/zoneinfo
10:
11: systemLog:
12:   destination: file
13:   logAppend: true
14:   path: /var/log/mongodb/mongod.log
15:
16: storage:
17:   dbPath: /data/db
18:   journal:
19:     enabled: true
20:
21: security:
22:   authorization: enabled
23: EOF
```

Создайте пользователей СУБД.

Переключитесь на пользователя mongod

```
1: su - mongod
```

Запустите mongod без контроля доступа

```
1: mongod --port 27017 --dbpath /data/db
```

Подключаемся к экземпляру и создаём пользователя

```
1: mongosh --port 27017
2: use admin
3: db.createUser(
4:   {
5:     user: "myUserAdmin",
6:     pwd: passwordPrompt(), // or cleartext password
7:     roles: [
8:       { role: "userAdminAnyDatabase", db: "admin" },
9:       { role: "readWriteAnyDatabase", db: "admin" }
10:    ]
11:  }
12: )
```

Завершите сеанс, остановите процесс mongod, а затем добавьте mongod в автозагрузку

```
1: sudo systemctl enable --now mongod
```

2.6 Установка Keycloak

Установите Java в системе:

```
2: sudo dnf install java-11-openjdk -y
```

Создайте системного пользователя keycloak

```
1: sudo useradd keycloak --system --no-create-home
```

Скачайте и распакуйте архив с keycloak

```
1: curl -L https://github.com/keycloak/keycloak/releases/download/19.0.1/keycloak-19.0.1.tar.gz -o /opt/keycloak
```

Необходимо убедиться, что создан каталог для данных, логов и ssl сертификата

```
1: sudo mkdir -p /opt/keycloak/keycloak-10.0.1/{data,log}
```

Заполните необходимые параметры в конфигурационном файле

```
1: sudo cat <<EOF> /opt/keycloak/keycloak-19.0.1/conf/keycloak.conf
2: db-username=<db_user>
3: db-password=<db_passr>
4: db=postgres
5: health-enabled=True
6: metrics-enabled=True
7: proxy=edge
8: db-url=jdbc:postgresql://<db_ip>:<db_port>/<db_name>
9: EOF
```

Смените владельца на keycloak

```
10: sudo chown -R keycloak:keycloak /opt/keycloak
```

Создайте системный юнит для запуска keycloak

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: sudo cat <<EOF> /etc/systemd/system/keycloak.service
2: [Unit]
3: Description=Keycloak server
4: After=network-online.target
5: Wants=network-online.target systemd-networkd-wait-online.service
6:
7: [Service]
8: User=keycloak
9: Group=keycloak
10: Environment="KEYCLOAK_ADMIN=<username>"
11: Environment="KEYCLOAK_ADMIN_PASSWORD=<password_for_username>"
12: Environment="KC_HTTP_RELATIVE_PATH=/auth"
13: Environment="PROXY_ADDRESS_FORWARDING=true"
14: KC_HOSTNAME="somedomain.ru"
15: ExecStart=/opt/keycloak/keycloak-19.0.1/bin/kc.sh start --http-enabled=true \
16:           --http-port=8080 --hostname-strict=false --hostname-strict-https=false \
17:           --log=console,file --log-file=/opt/keycloak/log/server.log
18: WorkingDirectory=/opt/keycloak/keycloak-19.0.1
19: ReadWritePaths=/opt/keycloak/keycloak-19.0.1/conf /opt/keycloak/keycloak-19.0.1/data
20:           /opt/keycloak/keycloak-19.0.1/lib/quarkus
21: SuccessExitStatus=0 143
22:
23: # Hardening options
24: CapabilityBoundingSet=
25: AmbientCapabilities=
26: NoNewPrivileges=true
27: ProtectHome=true
28: ProtectSystem=strict
29: ProtectKernelTunables=true
30: ProtectKernelModules=true
31: ProtectControlGroups=true
32: PrivateTmp=true
33: PrivateDevices=true
34: LockPersonality=true
35: [Install]
36: WantedBy=multi-user.target
37: EOF
```

Добавьте в автозагрузку сервис

```
1: Sudo systemctl daemon-reload
2: sudo systemctl enable --now keycloak
```

2.7 Установка Minio

Создайте пользователя minio

```
3: sudo useradd minio --system --no-create-home
```

Скачайте и установите пакет minio

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
4: go_arch=amd64
5: # RELEASE.2023-03-24T21-41-23Z
6: release=20230324214123.0.0.x86_64
7: curl -L https://dl.minio.io/server/minio/release/linux-amd64/archive/minio-
  \${minio_server_release}.rpm \
8: -o /tmp/minio-\${minio_server_release}.rpm
9: rpm -iU /tmp/minio-\${minio_server_release}.rpm
```

Создайте каталог для данных

```
1: mkdir -p /var/lib/minio
2: chown minio:minio -R /var/lib/minio
```

Отредактируйте файл /etc/default/minio

```
1: cat <<EOF> /etc/default/minio
2: # Minio local/remote volumes.
3: MINIO_VOLUMES="/var/lib/minio"
4: # Minio cli options.
5: MINIO_OPTS="--address :9000 --console-address :9001 "
6:
7: # Access Key of the server.
8: MINIO_ACCESS_KEY="root"
9: # Secret key of the server.
10: MINIO_SECRET_KEY="password"
11:
12: MINIO_DOMAIN=minio.domain.ru
13: MINIO_SERVER_URL=https://minio.domain.ru
14: MINIO_BROWSER_REDIRECT_URL=https://minio.domain.ru/console
15: CONSOLE_SUBPATH=/console
16: MINIO_ROOT_USER=root
17: MINIO_ROOT_PASSWORD=P@ssw0rd
18: EOF
```

Создайте системный юнит minio

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: cat <<EOF> /etc/systemd/system/minio.service
2: [Unit]
3: Description=MinIO
4: Documentation=https://docs.min.io
5: Wants=network-online.target
6: After=network-online.target
7: AssertFileIsExecutable=/usr/local/bin/minio
8:
9: [Service]
10: WorkingDirectory=/usr/local
11:
12: User=minio
13: Group=minio
14: ProtectProc=invisible
15:
16: EnvironmentFile=-/etc/default/minio
17: ExecStartPre=/bin/bash -c "if [ -z \"\${MINIO_VOLUMES}\" ]; then echo \"Variable
MINIO_VOLUMES not set in /etc/default/minio\"; exit 1; fi"
18: ExecStart=/usr/local/bin/minio server \${MINIO_OPTS} \${MINIO_VOLUMES}
19:
20: # Let systemd restart this service always
21: Restart=always
22:
23: # Specifies the maximum file descriptor number that can be opened by this process
24: LimitNOFILE=1048576
25:
26: # Specifies the maximum number of threads this process can create
27: TasksMax=infinity
28:
29: # Disable timeout logic and wait until process is stopped
30: TimeoutStopSec=infinity
31: SendSIGKILL=no
32:
33: [Install]
34: WantedBy=multi-user.target
35: EOF
```

Запустите сервис

```
1: systemctl daemon-reload
2: systemctl enable --now minio
```

2.8 Установка mediamtx

Создайте пользователя mediamtx

```
3: sudo useradd mediamtx --system --no-create-home
```

Скачайте архив с mediamtx и распакуйте

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: cd /tmp
2:
3: mkdir -p /etc/mediamtx/
4: #useradd mediamtx --system --no-create-home -g mediamtx
5: useradd mediamtx --system --no-create-home
6: curl -L
   https://github.com/bluenvion/mediamtx/releases/download/v0.22.0/mediamtx_v0.22.0_linux_am
   d64.tar.gz \
7: -o /tmp/mediamtx_v0.22.0_linux_amd64.tar.gz
8: tar -xzvf mediamtx_v0.22.0_linux_amd64.tar.gz
9: mv mediamtx /usr/local/bin/
10: chown -R mediamtx:mediamtx /usr/local/bin/mediamtx
```

Отредактируйте файл `/etc/mediamtx/mediamtx.yml`

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: cat <<EOF> /etc/mediamtx/mediamtx.yml
2:  logLevel: info
3:  logDestinations: [syslog]
4:  logFile: mediamtx.log
5:
6:  readTimeout: 10s
7:  writeTimeout: 10s
8:  readBufferCount: 512
9:  udpMaxPayloadSize: 1472
10:
11: externalAuthenticationURL:
12:
13: api: yes
14: apiAddress: 0.0.0.0:9997
15:
16: metrics: yes
17: metricsAddress: 0.0.0.0:9998
18:
19: pprof: no
20: pprofAddress: 127.0.0.1:9999
21:
22: runOnConnect:
23: runOnConnectRestart: no
24:
25:
26: rtspDisable: no
27: protocols: [udp, multicast, tcp]
28: encryption: "no"
29: rtspAddress: :8554
30: rtspAddress: :8322
31: rtpAddress: :8000
32: rtcpAddress: :8001
33: multicastIPRange: 224.1.0.0/16
34: multicastRTPPort: 8002
35: multicastRTCPPort: 8003
36: serverKey: server.key
37: serverCert: server.crt
38: authMethods: [basic, digest]
39:
40:
41: rtmpDisable: no
42: rtmpAddress: :1935
43: rtmpEncryption: "no"
44: rtmpsAddress: :1936
45: rtmpServerKey: server.key
46: rtmpServerCert: server.crt
47:
48:
49: hlsDisable: no
50: hlsAddress: :8888
51: hlsEncryption: no
52: hlsServerKey: server.key
53: hlsServerCert: server.crt
54: hlsAlwaysRemux: no
55: hlsVariant: lowLatency
56: hlsSegmentCount: 7
57: hlsSegmentDuration: 1s
58: hlsPartDuration: 200ms
59: hlsSegmentMaxSize: 50M
60: hlsAllowOrigin: '*'
61: hlsTrustedProxies: []
62: hlsDirectory: ''
63:
64:
65: webrtcDisable: no
66: webrtcAddress: :8889
```

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
67: webrtcEncryption: no
68: webrtcServerKey: server.key
69: webrtcServerCert: server.crt
70: webrtcAllowOrigin: '*'
71: webrtcTrustedProxies: []
72: webrtcICEServers: [stun:stun.l.google.com:19302]
73: webrtcICEHostNAT1To1IPs: []
74: webrtcICEUDPMuxAddress:
75: webrtcICETCPMuxAddress:
76:
77:
78: paths:
79:   all:
80:     source: publisher
81:
82:     sourceProtocol: automatic
83:
84:     sourceAnyPortEnable: no
85:
86:     sourceFingerprint:
87:
88:     sourceOnDemand: no
89:     sourceOnDemandStartTimeout: 10s
90:     sourceOnDemandCloseAfter: 10s
91:
92:     sourceRedirect:
93:
94:     disablePublisherOverride: no
95:
96:     fallback:
97:
98:     rpiCameraCamID: 0
99:     rpiCameraWidth: 1920
100:    rpiCameraHeight: 1080
101:    rpiCameraHFlip: false
102:    rpiCameraVFlip: false
103:    rpiCameraBrightness: 0
104:    rpiCameraContrast: 1
105:    rpiCameraSaturation: 1
106:    rpiCameraSharpness: 1
107:    rpiCameraExposure: normal
108:    rpiCameraAWB: auto
109:    rpiCameraDenoise: "off"
110:    rpiCameraShutter: 0
111:    rpiCameraMetering: centre
112:    rpiCameraGain: 0
113:    rpiCameraEV: 0
114:    rpiCameraROI:
115:    rpiCameraTuningFile:
116:    rpiCameraMode:
117:    rpiCameraFPS: 30
118:    rpiCameraIDRPeriod: 60
119:    rpiCameraBitrate: 1000000
120:    rpiCameraProfile: main
121:    rpiCameraLevel: '4.1'
122:    rpiCameraAfMode: auto
123:    rpiCameraAfRange: normal
124:    rpiCameraAfSpeed: normal
125:    rpiCameraLensPosition: 0.0
126:    rpiCameraAfWindow:
127:    rpiCameraTextOverlayEnable: false
128:    rpiCameraTextOverlay: '%Y-%m-%d %H:%M:%S - MediaMTX'
129:
130:    publishUser: video-processing-ms
131:    publishPass: PASSWORD
132:    publishIPs: []
```

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
133:
134:     readUser: video-processing-ms
135:     readPass: PASSWORD
136:     readIPs: []
137:
138:     runOnInit:
139:     runOnInitRestart: no
140:
141:     runOnDemand:
142:     runOnDemandRestart: no
143:     runOnDemandStartTimeout: 10s
144:     runOnDemandCloseAfter: 10s
145:
146:     runOnReady:
147:     runOnReadyRestart: no
148:
149:     runOnRead:
150:     runOnReadRestart: no
151: EOF
```

Укажем владельца

```
1: sudo chown -R mediamtx:mediamtx /etc/mediamtx/
```

Создайте системный юнит mediamtx

```
1: cat <<EOF> /etc/systemd/system/mediamtx.service
2: [Unit]
3: Description=mediamtx
4: After=network.target
5:
6: [Service]
7: User=mediamtx
8: Group=mediamtx
9: Type=simple
10: ExecStart=/usr/local/bin/mediamtx /etc/mediamtx/mediamtx.yml
11:
12: [Install]
13: WantedBy=multi-user.target
14: EOFЗапустите сервис
15: systemctl daemon-reload
16: systemctl enable --now minio
```

Запустите mediamtx

```
1: sudo systemctl daemon-reload
2: sudo systemctl enable mediamtx --now
```

2.9 Установка сервисов Цифрового Штаба

2.9.1 Установка реестра образов и загрузка образов

Создайте пользователя registry

```
1: sudo useradd registry --system --no-create-home
```

Создайте каталог для данных /var/lib/registry

```
1: sudo mkdir -p /var/lib/registry
2: sudo chown -R registry:registry /var/lib/registry
```

Создайте каталог для конфигурационных файлов

```
1: sudo mkdir -p /etc/registry/
2: sudo cat <<EOF> /etc/registry/config.yml
3: version: 0.1
4: log:
5:   fields:
6:     service: registry
7: storage:
8:   cache:
9:     blobdescriptor: inmemory
10:  filesystem:
11:    rootdirectory: /var/lib/registry
12: http:
13:   addr: :5000
14:   debug:
15:     addr: localhost:5001
16:   prometheus:
17:     enabled: true
18:     path: /metrics headers:
19:     X-Content-Type-Options: [nosniff]
20: health:
21:   storagedriver:
22:     enabled: true
23:     interval: 10s
24:     threshold: 3
25: EOF
26: sudo chown -R registry:registry /etc/registry/
```

Создайте системный юнит для сервиса registry

```
1: sudo cat << EOF > /etc/systemd/system/registry.service
2: [Unit]
3: Description=registry.service
4:
5: [Service]
6: User=registry
7: Group=registry
8: Restart=on-failure
9: ExecStartPre=-/usr/bin/docker rm -f registry
10: ExecStart=/usr/bin/docker run --name registry \
11:         -p 5000:5000 -v /var/lib/registry:/var/lib/registry \
12:         --restart=always registry:2
13: ExecStop=/usr/bin/docker stop -t 10 registry
14: ExecStopPost=-/usr/bin/docker rm -f registry
15: ExecReload=/usr/bin/docker restart registry
16:
17: [Install]
18: WantedBy=multi-user.target
19: EOF
```

Добавьте в автозагрузку

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: sudo systemctl daemon-reload
2: sudo systemctl enable --now registry
```

Установите nginx

```
1: sudo dnf install nginx -y
```

Создайте каталог для сертификата и скопируйте обе части

```
1: sudo mkdir -p /etc/nginx/ssl
2: sudo chown -R nginx:nginx /etc/nginx
```

Добавьте конфигурационный файл

Программный продукт
Цифровой штаб
Руководство по развертыванию

```

1: sudo cat <<EOF> /etc/nginx/conf.d/registry.conf
2: server {
3:     listen 443 ssl;
4:     server_name registry.somedomain.ru;
5:
6:     # SSL
7:     ssl_certificate /etc/nginx/ssl/fullchain.pem;
8:     ssl_certificate_key /etc/nginx/ssl/privkey.pem;
9:
10:    # Recommendations from
    https://raymii.org/s/tutorials/Strong_SSL_Security_On_nginx.html
11:    ssl_protocols TLSv1.1 TLSv1.2;
12:    ssl_ciphers 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH';
13:    ssl_prefer_server_ciphers on;
14:    ssl_session_cache shared:SSL:10m;
15:
16:    # disable any limits to avoid HTTP 413 for large image uploads
17:    client_max_body_size 0;
18:
19:    # required to avoid HTTP 411: see Issue #1486
    (https://github.com/moby/moby/issues/1486)
20:    chunked_transfer_encoding on;
21:
22:    location /v2/ {
23:        # Do not allow connections from docker 1.5 and earlier
24:        # docker pre-1.6.0 did not properly set the user agent on ping, catch "Go *" user
    agents
25:        if (\$http_user_agent ~ "^(docker\/1\.(3|4|5(?:?!\.[0-9]-dev))|Go ).*\$" ) {
26:            return 404;
27:        }
28:
29:        # To add basic authentication to v2 use auth_basic setting.
30:        # auth_basic "Registry realm";
31:        # auth_basic_user_file /etc/nginx/conf.d/nginx.htpasswd;
32:
33:        ## If \$docker_distribution_api_version is empty, the header is not added.
34:        ## See the map directive above where this variable is defined.
35:        add_header 'Docker-Distribution-Api-Version' \$docker_distribution_api_version
    always;
36:        proxy_pass http://localhost:5000;
37:        proxy_set_header Host \$http_host; # required for docker client's
    sake
38:        proxy_set_header X-Real-IP \$remote_addr; # pass on real client's IP
39:        proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
40:        proxy_set_header X-Forwarded-Proto \$scheme;
41:        proxy_read_timeout 900;
42:    }
43: }
44: EOF
45: sudo chown nginx:nginx -R /etc/nginx

```

Добавьте nginx в автозагрузку

```

1: sudo systemctl daemon-reload
2: sudo systemctl enable --now nginx

```

Скопируйте предоставленные архивы с образами в каталог /var/tmp и загрузите в реестр образов

```
1: cd /var/tmp
2: ls | xargs -I {} docker load -i {}
```

2.9.2 Сервис dhq-backend-for-frontend-ms

Configmap

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: data:
4:   APPLICATION_PORT: "8080"
5:   BPM_URL: http://dhq-bpm-ms:8080
6:   CORS_ALLOWED_ORIGINS: 'https://somedomain.ru'
7:   DATA_GATEWAY_URL: http://dhq-data-gateway-ms:8080
8:   EVENT_PROCESSING_URL: http://dhq-event-processing-ms:8080
9:   KEYCLOAK_CLIENT_SECRET: <KEYCLOAK_CLIENT_SECRET>
10:  KEYCLOAK_REALM: DHQ
11:  KEYCLOAK_RESOURCE: my-client
12:  KEYCLOAK_SERVER: https://somedomain.ru/auth
13:  LOGGING_LEVEL: INFO
14:  ORGANIZATION_ID: <ORGANIZATION_ID>
15:  TZ: Europe/Moscow
16:  VIDEO_PROCESSING_URL: http://dhq-video-processing-ms:8080
17: kind: ConfigMap
18: metadata:
19:   labels:
20:     project: dhq
21:   name: dhq-backend-for-frontend-ms-configmap-env
22: EOF
```

Ingress

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: kubectl apply -f - <<EOF
2: apiVersion: networking.k8s.io/v1
3: kind: Ingress
4: metadata:
5:   annotations:
6:     meta.helm.sh/release-name: dhq-dev
7:     meta.helm.sh/release-namespace: dhq-dev
8:     nginx.ingress.kubernetes.io/cors-allow-credentials: "true"
9:     nginx.ingress.kubernetes.io/cors-allow-headers: Cache-Control, Content-Disposition,
10:    Content-Length, Content-Type, Date, Expires, Pragma, Strict-Transport-Security,
11:    Vary, X-Content-Type-Options, X-Frame-Options, X-Xss-Protection, Accept, Accept-
Encoding,
12:    Accept-Language, Authorization, Cache-Control, Cookie, Referer, Sec-Ch-Ua, Sec-Ch-
Ua-Mobile,
13:    Sec-Ch-Ua-Platform, Sec-Fetch-Dest, Sec-Fetch-Mode, Sec-Fetch-Site, User-Agent,
14:    X-Timezone
15:     nginx.ingress.kubernetes.io/cors-allow-methods: PATCH, PUT, GET, POST, OPTIONS,
16:    DELETE
17:     nginx.ingress.kubernetes.io/cors-allow-origin: https://somedomain.ru
18:     nginx.ingress.kubernetes.io/enable-cors: "true"
19:     nginx.ingress.kubernetes.io/proxy-body-size: 16m
20:     nginx.ingress.kubernetes.io/proxy-read-timeout: "1800"
21:     nginx.ingress.kubernetes.io/proxy-send-timeout: "1800"
22:     nginx.org/websocket-services: dhq-backend-for-frontend-ms
23:   labels:
24:     app: dhq-backend-for-frontend-ms
25:     project: dhq
26:     name: dhq-backend-for-frontend-ms
27: spec:
28:   ingressClassName: nginx
29:   rules:
30:   - host: somedomain.ru
31:     http:
32:       paths:
33:       - backend:
34:         service:
35:           name: dhq-backend-for-frontend-ms
36:           port:
37:             number: 8080
38:         path: /api/(?!pass-requests)
39:         pathType: Prefix
40:       - backend:
41:         service:
42:           name: dhq-backend-for-frontend-ms
43:           port:
44:             number: 8080
45:         path: /[\d]/.*
46:         pathType: Prefix
47:       - backend:
48:         service:
49:           name: dhq-backend-for-frontend-ms
50:           port:
51:             number: 8080
52:         path: /info.*
53:         pathType: Prefix
54:     tls:
55:     - hosts:
56:       - somedomain.ru
57:       secretName: somedomain
58: EOF
```

Serviceaccount

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: ServiceAccount
4: metadata:
5:   labels:
6:     app: dhq-backend-for-frontend-ms
7:     project: dhq
8:   name: dhq-backend-for-frontend-ms
9: EOF
```

Deployment

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: kind: Deployment
4: metadata:
5:   labels:
6:     app: dhq-backend-for-frontend-ms
7:     project: dhq
8:   name: dhq-backend-for-frontend-ms
9: spec:
10:  replicas: 1
11:  selector:
12:    matchLabels:
13:      app: dhq-backend-for-frontend-ms
14:  strategy:
15:    rollingUpdate:
16:      maxSurge: 25%
17:      maxUnavailable: 25%
18:    type: RollingUpdate
19:  template:
20:    metadata:
21:      labels:
22:        app: dhq-backend-for-frontend-ms
23:        project: dhq
24:    spec:
25:      containers:
26:      - envFrom:
27:        - configMapRef:
28:            name: dhq-backend-for-frontend-ms-configmap-env
29:          image: dhq-backend-for-frontend-ms:v1.0
30:          imagePullPolicy: IfNotPresent
31:          name: dhq-backend-for-frontend-ms
32:          ports:
33:          - containerPort: 8080
34:            protocol: TCP
35:          resources:
36:            limits:
37:              cpu: 150m
38:              memory: 500Mi
39:            requests:
40:              cpu: 50m
41:              memory: 400Mi
42:          terminationMessagePath: /dev/termination-log
43:          terminationMessagePolicy: File
44:        dnsPolicy: ClusterFirst
45:        imagePullSecrets:
46:        - name: regcred
47:        restartPolicy: Always
48:        schedulerName: default-scheduler
49:        serviceAccount: dhq-backend-for-frontend-ms
50:        serviceAccountName: dhq-backend-for-frontend-ms
51:        terminationGracePeriodSeconds: 30
52:        volumes:
53:        - configMap:
54:            defaultMode: 420
55:            name: dhq-backend-for-frontend-ms-configmap
56:            name: dhq-backend-for-frontend-ms-configmap
57: EOF
```

Service

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: Service
4: metadata:
5:   labels:
6:     app: dhq-backend-for-frontend-ms
7:     project: dhq
8:   name: dhq-backend-for-frontend-ms
9:   namespace: dhq-dev
10: spec:
11:   ports:
12:   - name: http
13:     targetPort: 8080
14:     port: 8080
15:     protocol: TCP
16:   selector:
17:     app: dhq-backend-for-frontend-ms
18:   type: ClusterIP
19: EOF
```

2.9.3 Сервис dhq-bastion-adapter-ms

Serviceaccount dhq-bastion-adapter-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: ServiceAccount
4: metadata:
5:   labels:
6:     app: dhq-bastion-adapter-ms
7:     project: dhq
8:   name: dhq-bastion-adapter-ms
9: EOF
```

Configmap dhq-bastion-adapter-ms-configmap-env

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: data:
4:   APP_PORT: "8080"
5:   BASTION_OPC_SERVER_URL: <BASTION_OPC_SERVER_URL>
6:   KAFKA_BOOTSTRAP_SERVERS: <kafka-host>:9092
7:   KAFKA_SCHEMA_REGISTRY_URL: http://<schema-registry-host>:8081
8:   SPLIT_SIZE: "20"
9:   TZ: Europe/Moscow
10: kind: ConfigMap
11: metadata:
12:   labels:
13:     app: dhq-bastion-adapter-ms
14:     project: dhq
15:   name: dhq-bastion-adapter-ms-configmap-env
16: EOF
```

Deploy dhq-bastion-adapter-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: kind: Deployment
4: metadata:
5:   labels:
6:     app: dhq-bastion-adapter-ms
7:     project: dhq
8:   name: dhq-bastion-adapter-ms
9: spec:
10:   replicas: 1
11:   selector:
12:     matchLabels:
13:       app: dhq-bastion-adapter-ms
14:   strategy:
15:     rollingUpdate:
16:       maxSurge: 25%
17:       maxUnavailable: 25%
18:     type: RollingUpdate
19:   template:
20:     labels:
21:       app: dhq-bastion-adapter-ms
22:       project: dhq
23:     spec:
24:       containers:
25:         - envFrom:
26:           - configMapRef:
27:               name: dhq-bastion-adapter-ms-configmap-env
28:             image: dhq-bastion-adapter-ms:v1.0
29:             imagePullPolicy: IfNotPresent
30:             name: dhq-bastion-adapter-ms
31:             ports:
32:               - containerPort: 8080
33:                 protocol: TCP
34:             terminationMessagePath: /dev/termination-log
35:             terminationMessagePolicy: File
36:           dnsPolicy: ClusterFirst
37:           restartPolicy: Always
38:           schedulerName: default-scheduler
39:           serviceAccount: dhq-bastion-adapter-ms
40:           serviceAccountName: dhq-bastion-adapter-ms
41: EOF
```

Service dhq-bastion-adapter-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: Service
4: metadata:
5:   labels:
6:     app: dhq-bastion-adapter-ms
7:     project: dhq
8:   name: dhq-bastion-adapter-ms
9: spec:
10:   ports:
11:     - name: http
12:       targetPort: 8080
13:       port: 8080
14:   selector:
15:     app: dhq-bastion-adapter-ms
16:   type: ClusterIP
17: EOF
```

2.9.4 Сервис dhq-bpm-ms

Serviceaccount dhq-bpm-ms

```
18: kubectl apply -f - <<EOF
19: apiVersion: v1
20: kind: ServiceAccount
21: metadata:
22:   labels:
23:     app: dhq-bpm-ms
24:     project: dhq
25:   name: dhq-bpm-ms
26: EOF
```

Configmap dhq-bpm-ms-configmap-env

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: data:
4:   ACTIVITI_HISTORY_LEVEL: full
5:   APPLICATION_PORT: "8080"
6:   DATA_GATEWAY_URL: http://dhq-data-gateway-ms:8080
7:   DB_CONNECTION: postgres
8:   DB_HOST: <DB_HOST>
9:   DB_NAME: <DB_NAME>
10:  DB_PASSWORD: <DB_PASSWORD>
11:  DB_PORT: <DB_PORT>
12:  DB_USER: <DB_USER>
13:  EVENT_PROCESSING_URL: http://dhq-event-processing-ms:8080
14:  LOGGING_LEVEL: INFO
15:  PROCESSES_FOLDER_NAME: classpath:processes/**/**/*.bpmn
16:  TZ: Europe/Moscow
17: kind: ConfigMap
18: metadata:
19:   labels:
20:     app: dhq-bpm-ms
21:     project: dhq
22:   name: dhq-bpm-ms-configmap-env
23: EOF
```

Deployment dhq-bpm-ms

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: kind: Deployment
4: metadata:
5:   labels:
6:     app: dhq-bpm-ms
7:     project: dhq
8:     name: dhq-bpm-ms
9: spec:
10:   progressDeadlineSeconds: 600
11:   replicas: 2
12:   revisionHistoryLimit: 10
13:   selector:
14:     matchLabels:
15:       app: dhq-bpm-ms
16:   strategy:
17:     rollingUpdate:
18:       maxSurge: 25%
19:       maxUnavailable: 25%
20:     type: RollingUpdate
21:   template:
22:     metadata:
23:       labels:
24:         app: dhq-bpm-ms
25:         project: dhq
26:     spec:
27:       containers:
28:         - envFrom:
29:           - configMapRef:
30:               name: dhq-bpm-ms-configmap-env
31:             image: dhq-bpm-ms:v1.0
32:             imagePullPolicy: IfNotPresent
33:             name: dhq-bpm-ms
34:             ports:
35:               - containerPort: 8080
36:                 protocol: TCP
37:             resources:
38:               limits:
39:                 cpu: 150m
40:                 memory: 500Mi
41:               requests:
42:                 cpu: 50m
43:                 memory: 400Mi
44:             terminationMessagePath: /dev/termination-log
45:             terminationMessagePolicy: File
46:           dnsPolicy: ClusterFirst
47:           initContainers:
48:             - args:
49:                 - -c
50:                 - until pg_isready -h $DB_HOST -p $DB_PORT; do echo waiting for database;
51:                   sleep 2; done;
52:               command:
53:                 - /bin/sh
54:               envFrom:
55:                 - configMapRef:
56:                     name: dhq-bpm-ms-configmap-env
57:                   image: postgres:11.15
58:                   imagePullPolicy: IfNotPresent
59:                   name: check-db-ready
60:                   terminationMessagePath: /dev/termination-log
61:                   terminationMessagePolicy: File
62:             restartPolicy: Always
63:           schedulerName: default-scheduler
64:           serviceAccount: dhq-bpm-ms
65:           serviceAccountName: dhq-bpm-ms
```

```
66: EOF
```

Service dhq-bpm-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: Service
4: metadata:
5:   labels:
6:     app: dhq-bpm-ms
7:     project: dhq
8:   name: dhq-bpm-ms
9: spec:
10:  ports:
11:    - name: http
12:      targetPort: 8080
13:      port: 8080
14:  selector:
15:    app: dhq-bpm-ms
16:  type: ClusterIP
17: EOF
```

2.9.5 Сервис dhq-data-gateway-ms

Serviceaccount dhq-data-gateway-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: ServiceAccount
4: metadata:
5:   labels:
6:     app: dhq-data-gateway-ms
7:     project: dhq
8:   name: dhq-data-gateway-ms
9: EOF
```

Configmap dhq-data-gateway-ms-configmap-env

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: data:
4:   TZ: Europe/Moscow
5: kind: ConfigMap
6: metadata:
7:   labels:
8:     app: dhq-data-gateway-ms
9:     project: dhq
10:  name: dhq-data-gateway-ms-configmap-env
11: EOF
```

Deployment dhq-data-gateway-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: kind: Deployment
4: metadata:
5:   labels:
6:     app: dhq-data-gateway-ms
7:     project: dhq
8:   name: dhq-data-gateway-ms
9: spec:
10:  replicas: 1
11:  selector:
12:    matchLabels:
13:      app: dhq-data-gateway-ms
14:  strategy:
15:    rollingUpdate:
16:      maxSurge: 25%
17:      maxUnavailable: 25%
18:    type: RollingUpdate
19:  template:
20:    metadata:
21:      labels:
22:        app: dhq-data-gateway-ms
23:        project: dhq
24:    spec:
25:      containers:
26:      - envFrom:
27:        - configMapRef:
28:            name: dhq-data-gateway-ms-configmap-env
29:          image: dhq-data-gateway-ms:v1.0
30:          imagePullPolicy: IfNotPresent
31:          name: dhq-data-gateway-ms
32:          ports:
33:          - containerPort: 8080
34:            protocol: TCP
35:          terminationMessagePath: /dev/termination-log
36:          terminationMessagePolicy: File
37:        dnsPolicy: ClusterFirst
38:        restartPolicy: Always
39:        schedulerName: default-scheduler
40:        serviceAccount: dhq-data-gateway-ms
41:        serviceAccountName: dhq-data-gateway-ms
42: EOF
```

Service dhq-data-gateway-ms


```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: Service
4: metadata:
5:   labels:
6:     app: dhq-data-gateway-ms
7:     project: dhq
8:     name: dhq-data-gateway-ms
9: spec:
10:  ports:
11:    - name: http
12:      targetPort: 8080
13:      port: 8080
14:  selector:
15:    app: dhq-data-gateway-ms
16:  type: ClusterIP
17: EOF
```

2.9.6 Сервис dhq-dictionaries-update-ms

Serviceaccount dhq-dictionaries-update-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: ServiceAccount
4: metadata:
5:   labels:
6:     app: dhq-dictionaries-update-ms
7:     project: dhq
8:     name: dhq-dictionaries-update-ms
9: EOF
```

Configmap dhq-dictionaries-update-ms-configmap-env

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: data:
4:   APPLICATION_PORT: "8080"
5:   DATA_GATEWAY_URL: http://dhq-data-gateway-ms:8080
6:   KAFKA_BOOTSTRAP_SERVERS: <kafka-host>:9092
7:   KAFKA_SCHEMA_REGISTRY_URL: http://<schema-registry-host>:8081
8:   TZ: Europe/Moscow
9: kind: ConfigMap
10: metadata:
11:   labels:
12:     app: dhq-dictionaries-update-ms
13:     project: dhq
14:     name: dhq-dictionaries-update-ms-configmap-env
15: EOF
```

Deployment dhq-dictionaries-update-ms

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: kind: Deployment
4: metadata:
5:   labels:
6:     app: dhq-dictionaries-update-ms
7:     project: dhq
8:   name: dhq-dictionaries-update-ms
9: spec:
10:  replicas: 1
11:  selector:
12:    matchLabels:
13:      app: dhq-dictionaries-update-ms
14:  strategy:
15:    rollingUpdate:
16:      maxSurge: 25%
17:      maxUnavailable: 25%
18:    type: RollingUpdate
19:  template:
20:    metadata:
21:      labels:
22:        app: dhq-dictionaries-update-ms
23:        project: dhq
24:    spec:
25:      containers:
26:      - envFrom:
27:        - configMapRef:
28:            name: dhq-dictionaries-update-ms-configmap-env
29:          image: dhq-dictionaries-update-ms:v1.0
30:          imagePullPolicy: IfNotPresent
31:          name: dhq-dictionaries-update-ms
32:          ports:
33:          - containerPort: 8080
34:            protocol: TCP
35:          terminationMessagePath: /dev/termination-log
36:          terminationMessagePolicy: File
37:        dnsPolicy: ClusterFirst
38:        restartPolicy: Always
39:        schedulerName: default-scheduler
40:        serviceAccount: dhq-dictionaries-update-ms
41:        serviceAccountName: dhq-dictionaries-update-ms
42:        terminationGracePeriodSeconds: 30
43: EOF
```

Service dhq-dictionaries-update-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: Service
4: metadata:
5:   labels:
6:     app: dhq-dictionaries-update-ms
7:     project: dhq
8:     name: dhq-dictionaries-update-ms
9: spec:
10:  ports:
11:    - name: http
12:      targetPort: 8080
13:      port: 8080
14:  selector:
15:    app: dhq-dictionaries-update-ms
16:  type: ClusterIP
17: EOF
```

2.9.7 Сервис dhq-enrichment-ms

Serviceaccount dhq-enrichment-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: ServiceAccount
4: metadata:
5:   labels:
6:     app: dhq-enrichmentg-ms
7:     project: dhq
8:     name: dhq-enrichment-ms
9: EOF
```

Configmap dhq-enrichment-ms-configmap-env

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: data:
4:   APP_LOG_LEVEL: INFO
5:   DATA_GATEWAY_URL: http://dhq-data-gateway-ms:8080
6:   KAFKA_BOOTSTRAP_SERVERS: <kafka-host>:9092
7:   KAFKA_SCHEMA_REGISTRY_URL: http://<schema-registry-host>:8081
8:   TZ: Europe/Moscow
9: kind: ConfigMap
10: metadata:
11:   labels:
12:     app: dhq-enrichment-ms
13:     project: dhq
14:     name: dhq-enrichment-ms-configmap-env
15: EOF
```

Deployment dhq-enrichment-ms

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: kind: Deployment
4: metadata:
5:   labels:
6:     app: dhq-enrichment-ms
7:     project: dhq
8:   name: dhq-enrichment-ms
9: spec:
10:   progressDeadlineSeconds: 600
11:   replicas: 1
12:   revisionHistoryLimit: 10
13:   selector:
14:     matchLabels:
15:       app: dhq-enrichment-ms
16:   strategy:
17:     rollingUpdate:
18:       maxSurge: 25%
19:       maxUnavailable: 25%
20:     type: RollingUpdate
21:   template:
22:     metadata:
23:       labels:
24:         app: dhq-enrichment-ms
25:         project: dhq
26:     spec:
27:       containers:
28:         - env:
29:           - name: JAVA_TOOL_OPTIONS
30:             value: -Xdebug -
31:             agentlib:jwdp=transport=dt_socket,address=*:5005,server=y,suspend=n
32:             envFrom:
33:               - configMapRef:
34:                 name: dhq-enrichment-ms-configmap-env
35:             image: dhq-enrichment-ms:v1.0
36:             imagePullPolicy: IfNotPresent
37:             name: dhq-enrichment-ms
38:             ports:
39:               - containerPort: 8080
40:                 name: enrichment
41:                 protocol: TCP
42:               - containerPort: 5005
43:                 name: debug
44:                 protocol: TCP
45:             terminationMessagePath: /dev/termination-log
46:             terminationMessagePolicy: File
47:             dnsPolicy: ClusterFirst
48:             restartPolicy: Always
49:             schedulerName: default-scheduler
50:             serviceAccount: dhq-enrichment-ms
51:             serviceAccountName: dhq-enrichment-ms
52:             terminationGracePeriodSeconds: 30
53: EOF
```

Service dhq-enrichment-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: Service
4: metadata:
5:   labels:
6:     app: dhq-enrichment-ms
7:     project: dhq
8:     name: dhq-enrichment-ms
9: spec:
10:  ports:
11:    - name: http
12:      targetPort: 8080
13:      port: 8080
14:  selector:
15:    app: dhq-enrichment-ms
16:  type: ClusterIP
17: EOF
```

2.9.8 Сервис dhq-event-processing-ms

Serviceaccount

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: ServiceAccount
4: metadata:
5:   labels:
6:     app: dhq-event-processing-ms
7:     project: dhq
8:     name: dhq-event-processing-ms
9: EOF
```

Configmap dhq-event-processing-ms-configmap-env

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: data:
4:   BPM_URL: http://dhq-bpm-ms:8080
5:   DATA_GATEWAY_URL: http://dhq-data-gateway-ms:8080
6:   INCIDENT_WORKFLOW_URL: http://dhq-incident-workflow-ms:8080
7:   KAFKA_BOOTSTRAP_SERVERS: kafka:9092
8:   KAFKA_SCHEMA_REGISTRY_URL: http://<schema-registry-host>:8081
9:   KAFKA_SERVER_HOST: kafka-headless
10:  KAFKA_SERVER_PORT: "9092"
11:  TZ: Europe/Moscow
12: kind: ConfigMap
13: metadata:
14:   labels:
15:     app: dhq-event-processing-ms
16:     project: dhq
17:     name: dhq-event-processing-ms-configmap-env
18: EOF
```

Deployment dhq-event-processing-ms

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: kind: Deployment
4: metadata:
5:   labels:
6:     app: dhq-event-processing-ms
7:     project: dhq
8:   name: dhq-event-processing-ms
9: spec:
10:  replicas: 1
11:  selector:
12:    matchLabels:
13:      app: dhq-event-processing-ms
14:  strategy:
15:    rollingUpdate:
16:      maxSurge: 25%
17:      maxUnavailable: 25%
18:    type: RollingUpdate
19:  template:
20:    metadata:
21:      labels:
22:        app: dhq-event-processing-ms
23:        project: dhq
24:    spec:
25:      containers:
26:      - envFrom:
27:        - configMapRef:
28:            name: dhq-event-processing-ms-configmap-env
29:          image: dhq-event-processing-ms:v1.0
30:          imagePullPolicy: IfNotPresent
31:          name: dhq-event-processing-ms
32:          ports:
33:          - containerPort: 8080
34:            protocol: TCP
35:          terminationMessagePath: /dev/termination-log
36:          terminationMessagePolicy: File
37:        dnsPolicy: ClusterFirst
38:        restartPolicy: Always
39:        schedulerName: default-scheduler
40:        serviceAccount: dhq-event-processing-ms
41:        serviceAccountName: dhq-event-processing-ms
42: EOF
```

Service dhq-event-processing-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: Service
4: metadata:
5:   labels:
6:     app: dhq-event-processing-ms
7:     project: dhq
8:   name: dhq-event-processing-ms
9: spec:
10:  ports:
11:    - name: http
12:      targetPort: 8080
13:      port: 8080
14:  selector:
15:    app: dhq-event-processing-ms
16:  type: ClusterIP
17: EOF
```

2.9.9 Сервис dhq-frontend

Serviceaccount dhq-frontend

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: ServiceAccount
4: metadata:
5:   labels:
6:     app: dhq-frontend
7:     project: dhq
8:   name: dhq-frontend
9: EOF
```

Configmap dhq-frontend-configmap-env

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: data:
4:   TZ: Europe/Moscow
5: kind: ConfigMap
6: metadata:
7:   labels:
8:     app: dhq-frontend
9:     project: dhq
10:  name: dhq-frontend-configmap-env
11: EOF
```

Deployment dhq-frontend

```
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: kind: Deployment
4: metadata:
5:   labels:
6:     app: dhq-frontend
7:     project: dhq
8:   name: dhq-frontend
9:   namespace: dhq-dev
10: spec:
11:   progressDeadlineSeconds: 600
12:   replicas: 1
13:   revisionHistoryLimit: 10
14:   selector:
15:     matchLabels:
16:       app: dhq-frontend
17:   strategy:
18:     rollingUpdate:
19:       maxSurge: 25%
20:       maxUnavailable: 25%
21:     type: RollingUpdate
22:   template:
23:     metadata:
24:       labels:
25:         app: dhq-frontend
26:         project: dhq
27:     spec:
28:       containers:
29:         - envFrom:
30:           - configMapRef:
31:               name: dhq-frontend-configmap-env
32:             image: dhq-frontend:v1.0
33:             imagePullPolicy: IfNotPresent
34:             name: dhq-frontend
35:             ports:
36:               - containerPort: 80
37:                 protocol: TCP
38:             terminationMessagePath: /dev/termination-log
39:             terminationMessagePolicy: File
40:           dnsPolicy: ClusterFirst
41:           restartPolicy: Always
42:           schedulerName: default-scheduler
43:           serviceAccount: dhq-frontend
44:           serviceAccountName: dhq-frontend
45: EOF
```

Service dhq-frontend


```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: Service
4: metadata:
5:   labels:
6:     app: dhq-frontend
7:     project: dhq
8:     name: dhq-frontend
9: spec:
10:  ports:
11:    - name: http
12:      targetPort: 80
13:      port: 80
14:  selector:
15:    app: dhq-frontend
16:  type: ClusterIP
17: EOF
```

Ingress dhq-frontend

```
1: kubectl apply -f - <<EOF
2: apiVersion: networking.k8s.io/v1
3: kind: Ingress
4: metadata:
5:   labels:
6:     app: dhq-frontend
7:     project: dhq
8:     name: dhq-frontend
9:     namespace: dhq-dev
10: spec:
11:   ingressClassName: nginx
12:   rules:
13:     - host: somedomain.ru
14:       http:
15:         paths:
16:           - backend:
17:               service:
18:                 name: dhq-frontend
19:                 port:
20:                   number: 80
21:             path: /
22:             pathType: Prefix
23:   tls:
24:     - hosts:
25:       - somedomain.ru
26:       secretName: somedomain
27: EOF
```

2.9.10 Сервис dhq-integration-ms

Serviceaccount

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: ServiceAccount
4: metadata:
5:   labels:
6:     app: dhq-integration-ms
7:     project: dhq
8:   name: dhq-integration-ms
9: EOF
```

Deployment dhq-integration-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: kind: Deployment
4: metadata:
5:   labels:
6:     app: dhq-integration-ms
7:     project: dhq
8:   name: dhq-integration-ms
9: spec:
10: replicas: 1
11: selector:
12:   matchLabels:
13:     app: dhq-integration-ms
14: strategy:
15:   rollingUpdate:
16:     maxSurge: 25%
17:     maxUnavailable: 25%
18:   type: RollingUpdate
19: template:
20:   metadata:
21:     labels:
22:       app: dhq-integration-ms
23:       project: dhq
24:   spec:
25:     containers:
26:     - envFrom:
27:       - configMapRef:
28:         name: dhq-integration-ms-configmap-env
29:       - secretRef:
30:         name: dhq-integration-ms-secrets
31:       image: dhq-integration-ms:v1.0
32:       imagePullPolicy: IfNotPresent
33:       name: dhq-integration-ms
34:       ports:
35:       - containerPort: 8080
36:         protocol: TCP
37:       terminationMessagePath: /dev/termination-log
38:       terminationMessagePolicy: File
39:     dnsPolicy: ClusterFirst
40:     restartPolicy: Always
41:     schedulerName: default-scheduler
42:     serviceAccount: dhq-integration-ms
43:     serviceAccountName: dhq-integration-ms
44: EOF
```

Secret dhq-integration-ms-secrets

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: data:
4:   KEYCLOAK_PASSWORD: <KEYCLOAK_PASSWORD_base64>
5: kind: Secret
6: metadata:
7:   labels:
8:     app: dhq-integration-ms
9:     project: dhq
10:  name: dhq-integration-ms-secrets
11: type: Opaque
12: EOF
```

Configmap dhq-integration-ms-configmap-env

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: data:
4:   APPLICATION_PORT: "8080"
5:   BASTION_ADAPTER_URL: dhq-bastion-adapter-ms:8080
6:   BPM_URL: http://dhq-bpm-ms:8080
7:   COMPANY_FLAG: "false"
8:   DATA_GATEWAY_URL: http://dhq-data-gateway-ms:8080
9:   EDMS_PASSWORD: ""
10:  EDMS_USERNAME: ""
11:  IS_COMPANY: "false"
12:  KEYCLOAK_CLIENT_ID: <KEYCLOAK_CLIENT_ID>
13:  KEYCLOAK_REALM: DHQ
14:  KEYCLOAK_URL: https://somedomain.ru/auth
15:  KEYCLOAK_USER: <KEYCLOAK_USER>
16:  ORGANIZATION_ID: <ORGANIZATION_ID>
17:  SYSTEM_USER_ID: <SYSTEM_USER_ID>
18:  TOKEN_GRANT_TYPE: password
19:  TZ: Europe/Moscow
20: kind: ConfigMap
21: metadata:
22:   labels:
23:     app: dhq-integration-ms
24:     project: dhq
25:  name: dhq-integration-ms-configmap-env
26: EOF
```

Service dhq-integration-ms

```
27: kubectl apply -f - <<EOF
28: apiVersion: v1
29: kind: Service
30: metadata:
31:   labels:
32:     app: dhq-integration-ms
33:     project: dhq
34:     name: dhq-integration-ms
35: spec:
36:   ports:
37:   - name: http
38:     targetPort: 8080
39:     port: 8080
40:   selector:
41:     app: dhq-integration-ms
42:   type: ClusterIP
43: EOF
```

2.9.11 Сервис dhq-messages-ms

Serviceaccount dhq-messages-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: ServiceAccount
4: metadata:
5:   labels:
6:     app: dhq-messages-ms
7:     project: dhq
8:     name: dhq-messages-ms
9: EOF
```

Configmap dhq-messages-ms-configmap-env

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: data:
4:   APPLICATION_PORT: "8080"
5:   DATA_GATEWAY_URL: http://dhq-data-gateway-ms:8080
6:   DB_HOST: <mongodb_host>
7:   DB_PASSWORD: <mongodb_user_pass>
8:   DB_PORT: "27017"
9:   DB_USER: <mongodb_user>
10:  RDM_URL: dhq-rdm-ms:8080
11:  REDIS_SERVER_HOST: redis
12:  REDIS_SERVER_PORT: "6379"
13:  TZ: Europe/Moscow
14:  WEBSOCKET_ENABLED: "true"
15:  WEBSOCKET_URL: ws://dhq-backend-for-frontend-ms:8080/api/ws
16: kind: ConfigMap
17: metadata:
18:   labels:
19:     app: dhq-messages-ms
20:     project: dhq
21:     name: dhq-messages-ms-configmap-env
22: EOF
```

Deployment dhq-messages-ms

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: kind: Deployment
4: metadata:
5:   labels:
6:     app: dhq-messages-ms
7:     project: dhq
8:   name: dhq-messages-ms
9: spec:
10:  replicas: 1
11:  selector:
12:    matchLabels:
13:      app: dhq-messages-ms
14:  strategy:
15:    rollingUpdate:
16:      maxSurge: 25%
17:      maxUnavailable: 25%
18:    type: RollingUpdate
19:  template:
20:    metadata:
21:      labels:
22:        app: dhq-messages-ms
23:        project: dhq
24:    spec:
25:      containers:
26:        - envFrom:
27:          - configMapRef:
28:              name: dhq-messages-ms-configmap-env
29:            image: dhq-messages-ms:v1.0
30:            imagePullPolicy: IfNotPresent
31:            name: dhq-messages-ms
32:            ports:
33:              - containerPort: 8080
34:                protocol: TCP
35:            terminationMessagePath: /dev/termination-log
36:            terminationMessagePolicy: File
37:          dnsPolicy: ClusterFirst
38:          restartPolicy: Always
39:          schedulerName: default-scheduler
40:          serviceAccount: dhq-messages-ms
41:          serviceAccountName: dhq-messages-ms
42: EOF
```

Service dhq-messages-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: Service
4: metadata:
5:   labels:
6:     app: dhq-messages-ms
7:     project: dhq
8:     name: dhq-messages-ms
9: spec:
10:  ports:
11:    - name: http
12:      targetPort: 8080
13:      port: 8080
14:  selector:
15:    app: dhq-messages-ms
16:  type: ClusterIP
17: EOF
```

2.9.12 Сервис dhq-metadata

Serviceaccount dhq-metadata

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: ServiceAccount
4: metadata:
5:   labels:
6:     app: dhq-metadata
7:     project: dhq
8:     name: dhq-metadata
9: EOF
```

Configmap dhq-metadata-configmap-env

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: apiVersion: v1
4: data:
5:   TZ: Europe/Moscow
6: kind: ConfigMap
7: metadata:
8:   labels:
9:     app: dhq-metadata
10:    project: dhq
11:   name: dhq-metadata-configmap-env
12: EOF Deployment dhq-metadata
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: kind: Deployment
4: metadata:
5:   labels:
6:     app: dhq-metadata
7:     project: dhq
8:   name: dhq-metadata
9: spec:
10:  replicas: 1
11:  selector:
12:    matchLabels:
13:      app: dhq-metadata
14:  strategy:
15:    rollingUpdate:
16:      maxSurge: 25%
17:      maxUnavailable: 25%
18:    type: RollingUpdate
19:  template:
20:    metadata:
21:      labels:
22:        app: dhq-metadata
23:        project: dhq
24:    spec:
25:      containers:
26:        - envFrom:
27:          - configMapRef:
28:              name: dhq-metadata-configmap-env
29:            image: dhq-metadata:v1.0
30:            imagePullPolicy: IfNotPresent
31:            name: dhq-metadata
32:            ports:
33:              - containerPort: 80
34:                protocol: TCP
35:            terminationMessagePath: /dev/termination-log
36:            terminationMessagePolicy: File
37:          dnsPolicy: ClusterFirst
38:          restartPolicy: Always
39:          schedulerName: default-scheduler
40:          serviceAccount: dhq-metadata
41:          serviceAccountName: dhq-metadata
42: EOF
```

Service dhq-metadata

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: Service
4: metadata:
5:   labels:
6:     app: dhq-metadata
7:     project: dhq
8:     name: dhq-metadata
9: spec:
10:  ports:
11:    - name: http
12:      targetPort: 80
13:      port: 80
14:  selector:
15:    app: dhq-metadata
16:  type: ClusterIP
17: EOF
```

Ingress dhq-metadata

```
1: kubectl apply -f - <<EOF
2: apiVersion: networking.k8s.io/v1
3: kind: Ingress
4: metadata:
5:   annotations:
6:     nginx.ingress.kubernetes.io/configuration-snippet: |
7:       more_set_headers "Access-Control-Allow-Origin: \${http_origin}";
8:     nginx.ingress.kubernetes.io/enable-cors: "true"
9:     nginx.ingress.kubernetes.io/rewrite-target: /\$1
10:  labels:
11:    app: dhq-metadata
12:    project: dhq
13:    name: dhq-metadata
14: spec:
15:   ingressClassName: nginx
16:   rules:
17:     - host: somedomain.ru
18:       http:
19:         paths:
20:           - backend:
21:               service:
22:                 name: dhq-metadata
23:                 port:
24:                   number: 80
25:               path: /templates/(.*)
26:               pathType: Prefix
27:   tls:
28:     - hosts:
29:       - somedomain.ru
30:       secretName: somedomain
31: EOF
```

2.9.13 Сервис dhq-notification-ms

Serviceaccount


```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: ServiceAccount
4: metadata:
5:   labels:
6:     app: dhq-notification-ms
7:     project: dhq
8:   name: dhq-notification-ms
9: EOF
```

Configmap dhq-notification-ms-configmap-env

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: data:
4:   APP.EMAIL.BATCH_SIZE: "1"
5:   APP.EMAIL.CRON: "0 * * * * *"
6:   APP.EMAIL.FROM: <user@somedomain.ru>
7:   APP.USERS.URL: http://dhq-rdm-ms:8080
8:   DATA_GATEWAY_URL: http://dhq-data-gateway-ms:8080
9:   DATASOURCE.PASSWORD: <pgsql_user_pass>
10:  DATASOURCE.URL: jdbc:postgresql://<pgsql_db_host>:5432/<pgsql_db_name>
11:  DATASOURCE.USERNAME: <pgsql_db_username>
12:  DB_HOST: mongodb
13:  DB_PASSWORD: <mongodb_db_pass>
14:  DB_PORT: "27017"
15:  DB_USER: <mongodb_user_pass>
16:  JPA.DATABASE.PLATFORM: org.hibernate.dialect.PostgreSQLDialect
17:  JPA.SHOW.SQL: "false"
18:  MAIL.AUTH: "true"
19:  MAIL.CON.TIMEOUT: "5000"
20:  MAIL.DEBUG: "true"
21:  MAIL.HOST: email.somedomain.ru
22:  MAIL.PASSWORD: <mail_pass>
23:  MAIL.PORT: "587"
24:  MAIL.PROTOCOL: smtp
25:  MAIL.READ.TIMEOUT: "5000"
26:  MAIL.TEST_CONNECTION: "false"
27:  MAIL.TLS.ENABLED: "true"
28:  MAIL.TLS.REQUIRED: "true"
29:  MAIL.USERNAME: <user@somedomain.ru>
30:  MAIL.WRITE.TIMEOUT: "5000"
31:  REDIS_SERVER_HOST: redis
32:  REDIS_SERVER_PORT: "6379"
33:  SERVER.PORT: "8080"
34: kind: ConfigMap
35: metadata:
36:   labels:
37:     app: dhq-notification-ms
38:     project: dhq
39:   name: dhq-notification-ms-configmap-env
40: EOF
```

Deployment dhq-notification-ms

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: kind: Deployment
4: metadata:
5:   labels:
6:     app: dhq-notification-ms
7:     project: dhq
8:   name: dhq-notification-ms
9: spec:
10:  replicas: 1
11:  selector:
12:    matchLabels:
13:      app: dhq-notification-ms
14:  strategy:
15:    rollingUpdate:
16:      maxSurge: 25%
17:      maxUnavailable: 25%
18:    type: RollingUpdate
19:  template:
20:    metadata:
21:      labels:
22:        app: dhq-notification-ms
23:        project: dhq
24:    spec:
25:      containers:
26:        - envFrom:
27:          - configMapRef:
28:              name: dhq-notification-ms-configmap-env
29:            image: dhq-notification-ms:v1.0
30:            imagePullPolicy: IfNotPresent
31:            name: dhq-notification-ms
32:            ports:
33:              - containerPort: 8080
34:                protocol: TCP
35:            terminationMessagePath: /dev/termination-log
36:            terminationMessagePolicy: File
37:          dnsPolicy: ClusterFirst
38:          restartPolicy: Always
39:          schedulerName: default-scheduler
40:          serviceAccount: dhq-notification-ms
41:          serviceAccountName: dhq-notification-ms
42:          terminationGracePeriodSeconds: 30
43:          volumes:
44:            - configMap:
45:                defaultMode: 420
46:                name: dhq-notification-ms-configmap
47:                name: dhq-notification-ms-configmap
48: EOF
```

Service

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: Service
4: metadata:
5:   labels:
6:     app: dhq-notification-ms
7:     project: dhq
8:     name: dhq-notification-ms
9: spec:
10:  ports:
11:    - name: http
12:      targetPort: 8080
13:      port: 8080
14:  selector:
15:    app: dhq-notification-ms
16:  type: ClusterIP
17: EOF
```

2.9.14 Сервис dhq-rdm-ms

Serviceaccount

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: ServiceAccount
4: metadata:
5:   labels:
6:     app: dhq-rdm-ms
7:     project: dhq
8:     name: dhq-rdm-ms
9: EOF
```

Configmap dhq-rdm-ms-configmap-env

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: data:
4:   APPLICATION_PORT: "8080"
5:   DB_CONNECTION: postgres
6:   DB_HOST: <DB_HOST>
7:   DB_NAME: <DB_NAME>
8:   DB_PASSWORD: <DB_PASSWORD>
9:   DB_PORT: "5432"
10:  DB_USER: <DB_USER>
11:  EMAIL_NOTIFICATION_ENABLED: "true"
12:  EMAIL_NOTIFICATION_URL: http://dhq-notification-ms:8080/notification
13:  IS_COMPANY: "false"
14:  LOGGING_LEVEL: INFO
15:  NOTIFICATION_URL: http://dhq-notification-ms:8080/notification
16:  ORGANIZATION_ID: <ORGANIZATION_ID>
17:  REDIS_SERVER_HOST: redis
18:  REDIS_SERVER_PORT: "6379"
19:  SPRING_PROFILES_ACTIVE: organization
20:  TZ: Europe/Moscow
21:  UI_NOTIFICATION_ENABLED: "true"
22:  WEBSOCKET_ENABLED: "true"
23:  WEBSOCKET_URL: ws://dhq-backend-for-frontend-ms:8080/api/ws
24: kind: ConfigMap
25: metadata:
26:   labels:
27:     app: dhq-rdm-ms
28:     project: dhq
29:     name: dhq-rdm-ms-configmap-env
30: EOFDeployment
1: kubectl apply -f - <<EOF
```

Service

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: Service
4: metadata:
5:   labels:
6:     app: dhq-rdm-ms
7:     project: dhq
8:     name: dhq-rdm-ms
9: spec:
10:  ports:
11:    - name: http
12:      targetPort: 8080
13:      port: 8080
14:  selector:
15:    app: dhq-rdm-ms
16:  type: ClusterIP
17: EOF
```

Deployment dhq-rdm-ms

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: kind: Deployment
4: metadata:
5:   labels:
6:     app: dhq-rdm-ms
7:     project: dhq
8:     name: dhq-rdm-ms
9: spec:
10:   progressDeadlineSeconds: 600
11:   replicas: 1
12:   revisionHistoryLimit: 10
13:   selector:
14:     matchLabels:
15:       app: dhq-rdm-ms
16:   strategy:
17:     rollingUpdate:
18:       maxSurge: 25%
19:       maxUnavailable: 25%
20:     type: RollingUpdate
21:   template:
22:     metadata:
23:       labels:
24:         app: dhq-rdm-ms
25:         project: dhq
26:     spec:
27:       containers:
28:         - env:
29:           - name: JAVA_TOOL_OPTIONS
30:             value: -Xdebug -
31:             agentlib:jdwp=transport=dt_socket,address=*:5005,server=y,suspend=n
32:             envFrom:
33:               - configMapRef:
34:                 name: dhq-rdm-ms-configmap-env
35:             image: dhq-rdm-ms:v1.0
36:             imagePullPolicy: IfNotPresent
37:             name: dhq-rdm-ms
38:             ports:
39:               - containerPort: 8080
40:                 name: http-rdm
41:                 protocol: TCP
42:             terminationMessagePath: /dev/termination-log
43:             terminationMessagePolicy: File
44:             dnsPolicy: ClusterFirst
45:             initContainers:
46:               - args:
47:                 - -c
48:                 - until pg_isready -h $DB_HOST -p $DB_PORT; do echo waiting for database;
49:                   sleep 2; done;
50:                 command:
51:                 - /bin/sh
52:             envFrom:
53:               - configMapRef:
54:                 name: dhq-rdm-ms-configmap-env
55:             image: postgres:11.15
56:             imagePullPolicy: IfNotPresent
57:             name: check-db-ready
58:             terminationMessagePath: /dev/termination-log
59:             terminationMessagePolicy: File
60:             restartPolicy: Always
61:             schedulerName: default-scheduler
62:             serviceAccount: dhq-rdm-ms
63:             serviceAccountName: dhq-rdm-ms
64:             terminationGracePeriodSeconds: 30
65:             volumes:
66:               - configMap:
```

```
66:         defaultMode: 420
67:         name: dhq-rdm-ms-configmap
68:     name: dhq-rdm-ms-configmap
69: EOF
```

2.9.15 Сервис dhq-video-processing-ms

Serviceaccount

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: ServiceAccount
4: metadata:
5:   labels:
6:     app: dhq-video-processing-ms
7:     project: dhq
8:   name: dhq-video-processing-ms
9: EOF
```

Configmap configmap-nginx

```
1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: data:
4:   nginx.conf: |-
5:     worker_processes auto;
6:     rtmp_auto_push on;
7:     events {}
8:     rtmp {
9:       server {
10:        listen 1935;
11:        application live {
12:          live on;
13:          #record off;
14:          #interleave on;
15:
16:          hls on;
17:          hls_path /tmp/hls/;
18:          hls_nested on;
19:          hls_fragment 4s; # default is 5s
20:          # hls_playlist_length 60; # default is 30s
21:        }
22:      }
23:    }
24:   http {
25:     default_type application/octet-stream;
26:
27:     server {
28:       listen 80;
29:       location /video {
30:         root /tmp/hls;
31:       }
32:     }
33:
34:     types {
35:       application/vnd.apple.mpegurl m3u8;
36:       video/mp2t ts;
37:       text/html html;
38:     }
39:   }
40: kind: ConfigMap
41: metadata:
42:   labels:
43:     app: dhq-video-processing-ms
44:     project: dhq
45:   name: configmap-nginx
46: EOF
```

Configmap dhq-video-processing-ms-configmap-env

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
47: kubectl apply -f - <<EOF
48: apiVersion: v1
49: data:
50:   APPLICATION_PORT: "8080"
51:   COMPANY_STREAM_LIMIT: "4"
52:   DATA_GATEWAY_URL: dhq-data-gateway-ms:8080
53:   DB_HOST: pgsq1
54:   DB_NAME: <db_name>
55:   DB_PASSWORD: <db_pass>
56:   DB_PORT: "5432"
57:   DB_USER: <db_user>
58:   INTELLECT_ADAPTER_URL: http://<intellect_ip>:8080
59:   IS_ORGANIZATION: "true"
60:   MEDIA_SERVER_IP: <mediamtx_ip>:8554
61:   MEDIA_SERVER_LOGIN: <MEDIA_SERVER_LOGIN>
62:   MEDIA_SERVER_PASSWORD: <MEDIA_SERVER_PASSWORD>
63:   MEDIA_SERVER_URL: ${MEDIA_SERVER_LOGIN}:${MEDIA_SERVER_PASSWORD}@<mediamtx_ip>:8554
64:   MINIO_ACCESS_KEY: <MINIO_ACCESS_KEY>
65:   MINIO_SECRET_KEY: <MINIO_SECRET_KEY>
66:   MINIO_URL: https://minio.somedomain.ru
67:   ORGANIZATION_STREAM_LIMIT: "4"
68:   TZ: Europe/Moscow
69:   VIDEO_PROVIDER_PROCESSING_RATE: "30"
70:   VIDEO_PROVIDER_RECOVERY_RATE: "120"
71: kind: ConfigMap
72: metadata:
73:   labels:
74:     app: dhq-video-processing-ms
75:     project: dhq
76:     name: dhq-video-processing-ms-configmap-env
77: EOF
```

Deployment

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: kubectl apply -f - <<EOF
2: apiVersion: apps/v1
3: kind: Deployment
4: metadata:
5:   labels:
6:     app: dhq-video-processing-ms
7:     project: dhq
8:   name: dhq-video-processing-ms
9: spec:
10:  progressDeadlineSeconds: 600
11:  replicas: 1
12:  revisionHistoryLimit: 10
13:  selector:
14:    matchLabels:
15:      app: dhq-video-processing-ms
16:  strategy:
17:    rollingUpdate:
18:      maxSurge: 25%
19:      maxUnavailable: 25%
20:    type: RollingUpdate
21:  template:
22:    metadata:
23:      labels:
24:        app: dhq-video-processing-ms
25:        project: dhq
26:    spec:
27:      containers:
28:        - image: tiangolo/nginx-rtmp:latest-2023-04-03
29:          imagePullPolicy: IfNotPresent
30:          name: nginx
31:          ports:
32:            - containerPort: 80
33:              name: http-nginx
34:              protocol: TCP
35:          securityContext:
36:            runAsUser: 0
37:          terminationMessagePath: /dev/termination-log
38:          terminationMessagePolicy: File
39:          volumeMounts:
40:            - mountPath: /etc/nginx/nginx.conf
41:              name: configmap-nginx
42:              subPath: nginx.conf
43:            - mountPath: /tmp/hls
44:              name: shared-data
45:            - mountPath: /etc/localtime
46:              name: tz
47:        - envFrom:
48:          - configMapRef:
49:              name: dhq-video-processing-ms-configmap-env
50:            image: dhq-video-processing-ms:v1.0
51:            imagePullPolicy: IfNotPresent
52:            name: dhq-video-processing-ms
53:            ports:
54:              - containerPort: 8080
55:                name: http
56:                protocol: TCP
57:            securityContext:
58:              runAsUser: 0
59:            terminationMessagePath: /dev/termination-log
60:            terminationMessagePolicy: File
61:            volumeMounts:
62:              - mountPath: /dhq
63:                name: shared-data
64:            dnsPolicy: ClusterFirst
65:            initContainers:
66:              - args:
```

Программный продукт
Цифровой штаб
Руководство по развертыванию

```

67:     - cp -a /dhq/. /dhq2/; chown -R dhq:dhq /dhq2/;
68:     command:
69:     - /bin/sh
70:     - -c
71:     envFrom:
72:     - configMapRef:
73:       name: dhq-video-processing-ms-configmap-env
74:     image: dhq-video-processing-ms:v1.0
75:     imagePullPolicy: IfNotPresent
76:     name: migrate
77:     securityContext:
78:       runAsUser: 0
79:     terminationMessagePath: /dev/termination-log
80:     terminationMessagePolicy: File
81:     volumeMounts:
82:     - mountPath: /dhq2
83:       name: shared-data
84:     restartPolicy: Always
85:     schedulerName: default-scheduler
86:     serviceAccount: dhq-dev-dhq-video-processing-ms
87:     serviceAccountName: dhq-dev-dhq-video-processing-ms
88:     terminationGracePeriodSeconds: 30
89:     volumes:
90:     - name: shared-data
91:     - configMap:
92:       defaultMode: 420
93:       name: dhq-video-processing-ms-configmap
94:     name: dhq-video-processing-ms-configmap
95:     - configMap:
96:       defaultMode: 420
97:       name: configmap-nginx
98:     name: configmap-nginx
99:     - hostPath:
100:      path: /usr/share/zoneinfo/Europe/Moscow
101:      type: ""
102:     name: tz
103: EOF

```

Service

```

1: kubectl apply -f - <<EOF
2: apiVersion: v1
3: kind: Service
4: metadata:
5:   labels:
6:     app: dhq-video-processing-ms
7:     project: dhq
8:     name: dhq-video-processing-ms
9: spec:
10:   ports:
11:   - name: http
12:     targetPort: 80
13:     port: 80
14:   - name: http-jar
15:     targetPort: 8080
16:     port: 8080
17:   selector:
18:     app: dhq-video-processing-ms
19:   type: ClusterIP
20: EOF

```

Ingress dhq-video-processing-ms

```
1: kubectl apply -f - <<EOF
2: apiVersion: networking.k8s.io/v1
3: kind: Ingress
4: metadata:
5:   annotations:
6:     nginx.ingress.kubernetes.io/configuration-snippet : |
7:       if ($request_uri ~* \.(m3u8|ts)) {
8:         add_header Cache-Control "no-cache,no-store";
9:       }
10:    nginx.ingress.kubernetes.io/cors-allow-origin: https://somedomain.ru
11:    nginx.ingress.kubernetes.io/enable-cors: "true"
12:   labels:
13:     app: dhq-video-processing-ms
14:     project: dhq
15:     name: dhq-video-processing-ms
16: spec:
17:   ingressClassName: nginx
18:   rules:
19:   - host: somedomain.ru
20:     http:
21:       paths:
22:       - backend:
23:         service:
24:           name: dhq-video-processing-ms
25:           port:
26:             number: 80
27:           path: /video/*
28:           pathType: Prefix
29:     tls:
30:     - hosts:
31:       - somedomain.ru
32:       secretName: somedomain
33: EOF
```

Ingress dhq-video-processing-ms-video

Программный продукт
Цифровой штаб
Руководство по развертыванию

```
1: kubectl apply -f - <<EOF
2: apiVersion: networking.k8s.io/v1
3: kind: Ingress
4: metadata:
5:   annotations:
6:     nginx.ingress.kubernetes.io/cors-allow-credentials: "true"
7:     nginx.ingress.kubernetes.io/cors-allow-methods: PATCH, PUT, GET, POST, OPTIONS
8:     nginx.ingress.kubernetes.io/cors-allow-origin: https://somedomain.ru
9:     nginx.ingress.kubernetes.io/enable-cors: "true"
10:    nginx.ingress.kubernetes.io/proxy-read-timeout: "1800"
11:    nginx.ingress.kubernetes.io/proxy-send-timeout: "1800"
12:   labels:
13:     app: dhq-video-processing-ms
14:     project: dhq
15:     name: dhq-video-processing-ms-video
16: spec:
17:   ingressClassName: nginx
18:   rules:
19:     - host: somedomain.ru
20:       http:
21:         paths:
22:           - backend:
23:               service:
24:                 name: dhq-video-processing-ms
25:                 port:
26:                   number: 8080
27:               path: /videostream.*
28:               pathType: Prefix
29:           - backend:
30:               service:
31:                 name: dhq-video-processing-ms
32:                 port:
33:                   number: 8080
34:               path: /videofiles/*
35:               pathType: Prefix
36:         tls:
37:           - hosts:
38:             - somedomain.ru
39:             secretName: somedomain
40: EOF
```