



**Руководство по  
установке МоиОтчеты  
Корпоративный сервер  
и МоиОтчеты  
Публикатор**

# Разворачивание кластера Kubernetes

Продукты: [МоиОтчеты Корпоративный Сервер](#)

Для установки и настройки сервера отчётов необходимо развернуть облачную экосистему Kubernetes.

Для корректной работы МоиОтчеты Корпоративный Сервер необходимо минимум три узла - один master узел и два worker узла. В процессе работ количество узлов может быть увеличено при росте нагрузки. Так же количество узлов может быть уменьшено при снижении нагрузки. Динамическое управление количеством узлов в настоящее время не реализовано.

[Видео установки Корпоративного Сервера](#)

## Подготовка и установка компонентов Kubernetes

Установка Kubernetes осуществляется последовательным выполнением следующих bash скриптов. Эти примеры подразумевают операционную систему Debian Linux. В случае использования другого дистрибутива некоторые пункты будут отличаться. Например, для разворачивания кластера в системе Alt Linux Server, использовалась документация со страницы [www.altlinux.org/Kubernetes](http://www.altlinux.org/Kubernetes), которой будет достаточно для установки Kubernetes, описанной в этой разделе.

1. Отключение раздела swap.

```
#
# Permanently disable swap
#
sed -e 's/^/#/' -i /etc/fstab
swapoff -a
```

2. Загрузка необходимых модулей ядра.

```
#
# Enable kernel modules
#
MODULES=/etc/modules-load.d/k8s.conf
if [ ! -f $MODULES ]; then
    echo "Create $MODULES"
    cat<<EOF | tee $MODULES
overlay
br_netfilter
EOF
fi
```

3. Настройка сетевых свойств ядра системы.

```
#
# Configure kernel
#
SYSCTL=/etc/sysctl.d/k8s.conf
if [ ! -f $SYSCTL ]; then
    echo "Prepare kernel options"
    cat<<EOF | tee $SYSCTL
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
fi
sysctl --system
```

4. Установка дополнительного программного обеспечения, которое потребуется для установки оркестратора и сервера отчётов.

```
apt-get update

#
# Install pre-requested packages
#
prerequisite=( curl sudo gnupg2 apt-transport-https ca-certificates software-properties-common )

for package in "${prerequisite[@]}"
do
    echo -n "Checking $package: "
    dpkg -s $package > /dev/null 2> /dev/null
    if [ $? -ne 0 ]; then
        echo "Installing"
        apt-get install -y $package
    else
        echo "OK"
    fi
done
```

5. Установка Container Runtime Interface (CRI).

Следующий пример использует CRI containerd. Различные версии Kubernetes и Linux могут потребовать установки CRI-O. Kubernetes использует CRI для загрузки контейнеров, управления контейнерами и для запуска процессов в этих контейнерах.

```
## Add Docker's official GPG key
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key --keyring /etc/apt/trusted.gpg.d/docker.gpg add -

## Add Docker apt repository.
add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/debian \
    $(lsb_release -cs) \
    stable"

## Install containerd
apt-get update && apt-get install -y containerd.io

# Configure containerd
if [ ! -d /etc/containerd ]; then
    mkdir -p /etc/containerd
fi

# Remove default config to avoid errors
if [ -f /etc/containerd/config.toml ]; then
    rm /etc/containerd/config.toml
fi

# Restart containerd
systemctl restart containerd
```

6. Установка компонентов Kubernetes.

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
cat <<EOF | tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
apt-get update
apt-get install -y kubelet kubeadm kubectl
apt-mark hold kubelet kubeadm kubectl
```

# Старт Kubernetes

Для старта главного узла кластера отредактируйте 3 переменные, отвечающие за соответствующие им пути к конфигурационному файлу (он будет создан автоматически), IP адресу (будет доступен извне) и маску внутренней подсети.

Затем выполните команду инициализации кластера.

```
export KUBECONFIG=/etc/kubernetes/admin.conf1
export MAIN_IF=192.168.1.191
export POD_NETWORK=10.244.0.0/16 # В случае использования flannel это значение менять нельзя!

kubeadm init --pod-network-cidr=$POD_NETWORK --apiserver-advertise-address=$MAIN_IF
```

Адрес MAIN\_IF это IP-адрес главного узла кластера (master node). Адрес может быть реальным IP- адресом или адресом подсети 192.168.

В случае успешного выполнения инициализации главного узла кластера (master node), на экран будет выведена строка, с помощью которой инициализируются рабочие узлы кластера. С помощью мыши скопируйте эту строку и сохраните её в файл. В этой строке содержится команда для инициализации рабочих узлов. Команда включает в себя секретный ключ. При попадании этого ключа злоумышленнику, он может "скомпрометировать" кластер.

Пример строки:

```
kubeadm join 192.168.1.191:6443 --token lw0lgz.d5zy9fb4jkc89yv \
--discovery-token-ca-cert-hash sha256:ba83ed75e9fd5f430007000000000000039e1d05c7915a54435faaa7fe62b77
```

Для добавления узла в кластер выполните действия, описанные в предыдущем разделе (за исключением установки helm chart), скопируйте на каждый узел эту строку и выполните её от имени администратора (root). В результате выполнения этой команды в кластер будет добавлен новый узел.

Настройка kubectl.

```
mkdir ~/.kube
rm ~/.kube/config
cp /etc/kubernetes/admin.conf ~/.kube
```

Теперь необходимо установить сервис flannel.

```
kubectl apply -f https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/kube-flannel.yml
```

В случае успешной установки вывод команды `kubectl get pods --all-namespaces` должен выглядеть примерно так:

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-78fcd69978-brp76	1/1	Running	0	5m13s
kube-system	coredns-78fcd69978-shdv5	1/1	Running	0	5m13s
kube-system	etcd-altlinux-10-1	1/1	Running	0	5m27s
kube-system	kube-apiserver-altlinux-10-1	1/1	Running	0	5m22s
kube-system	kube-controller-manager-altlinux-10-1	1/1	Running	0	5m31s
kube-system	kube-flannel-ds-7nn2c	1/1	Running	0	28s
kube-system	kube-flannel-ds-9phgs	1/1	Running	0	28s
kube-system	kube-flannel-ds-ddcw6	1/1	Running	0	28s
kube-system	kube-proxy-74pll	1/1	Running	0	3m36s
kube-system	kube-proxy-82nld	1/1	Running	0	3m15s
kube-system	kube-proxy-njxwv	1/1	Running	0	5m14s
kube-system	kube-scheduler-altlinux-10-1	1/1	Running	0	5m32s

В случае, если все шаги по установке были сделаны верно, то состояние из примера выше, будет достигнуто в течение минуты. Кластеру необходимо время чтобы запустить и сконфигурировать coredns после установки flannel.

# МоиОтчеты Корпоративный Сервер и импортозамещение

Продукты: [МоиОтчеты Корпоративный Сервер](#)

## Версии компонентов Kubernetes

Изначально продукт разрабатывался для кластера, развёрнутого на основе дистрибутива Debian GNU/Linux 10 (buster). Для поддержки импортозамещения был протестирован ряд дистрибутивов Linux российских производителей. По итогам тестирования зависимости от версии Kubernetes не выявлено - минимальные номера версий, на которых производилось тестирование показаны в следующей таблице:

КОМПОНЕНТ	ВЕРСИЯ
docker	19.3.8
kubelet	1.18.1
kube-proxy	1.18.1

При тестировании проверялись также более свежие версии Kubernetes. Обновление версии Kubernetes не нарушает работу сервера отчётов. Таким образом тестирование отечественных дистрибутивов свелось к проверке возможности установки Kubernetes.

## Провайдеры CNI

Сервер отчётов был протестирован со CNI провайдерами Flannel и Calico. Оба провайдера показали удовлетворительную работу. Конфигурация с Flannel была протестирована на самостоятельно развёрнутом кластере, конфигурация с Calico была протестирована на облачном провайдере Selectel (Россия).

## Российские дистрибутивы Linux

Результаты тестирования представлены в следующей таблице:

ПРОИЗВОДИТЕЛЬ	СОВМЕСТИМОСТЬ	ПРИМЕЧАНИЯ
Alt Linux Server 9	Полная	<a href="https://www.altlinux.org/Kubernetes">https://www.altlinux.org/Kubernetes</a>
ROSA "Хром" Linux	Заявлена	<a href="http://wiki.rosalab.com/ru/index.php/Установка_docker,_kubernetes_кластера">http://wiki.rosalab.com/ru/index.php/Установка_docker,_kubernetes_кластера</a>
ASTRA Linux 1.7	Deckhouse	<a href="https://deckhouse.io/ru/">https://deckhouse.io/ru/</a>
RED OS	Заявлена	<a href="https://redos.red-soft.ru/base/server-configuring/container/kubernetes/">https://redos.red-soft.ru/base/server-configuring/container/kubernetes/</a>
AlterOS 7	Deckhouse	<a href="https://deckhouse.io/ru/">https://deckhouse.io/ru/</a>

Примечание: [Deckhouse](#) - продукт российских разработчиков для разворачивания кластера.

# Минимальные системные требования

Продукты: [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

## Аппаратное обеспечение

Для минимального запуска рекомендуется следующее аппаратное обеспечение:

- **Процессор:** 2x1,80 GHz (x86\_64)
- **Оперативная память (RAM):** 8 ГБ
- **Хранилище:** 20 ГБ диска

**Примечание:** рекомендуется заранее предусмотреть дополнительное место под логи, кэш и временные файлы отчётов.

## Программное обеспечение

Для установки без Kubernetes требуются следующие компоненты:

- **Docker Engine:** 19.03.0 или новее
- **docker-compose:** v2.12.2 или новее

## Поддерживаемые операционные системы

Поддерживаются все операционные системы, для которых доступен Docker, включая:

- Windows
- Linux (Debian, Ubuntu)
- российские операционные системы: РЕД ОС, Astra Linux, Alt Linux
- macOS
- другие операционные системы с поддержкой Docker

В наличии есть docker-образы как на базе debian, так и на базе РЕД ОС.

## Установка в Windows

Для Windows также доступен удобный **Мастер установки**, с помощью которого даже неподготовленный пользователь сможет установить МоиОтчеты Публикатор или МоиОтчеты Корпоративный Сервер.

# Установка сервиса nginx-ingress

Продукты: [МоиОтчеты](#) [Корпоративный Сервер](#)

Все описанные команды выполняется только на master узле!

Добавление в Helm репозитория `ingress-nginx`:

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update
```

## Установка nginx в Kubernetes

Внимание! Переменная `MAIN_IF` должна указывать IP адрес внешнего интерфейса для доступа к серверу отчётов.

```
export MAIN_IF=192.168.1.191
kubectl create namespace nginx

helm upgrade --install nginx-ingress ingress-nginx/ingress-nginx \
  --create-namespace --namespace nginx \
  --set controller.replicaCount=2 \
  --set controller.service.externalIPs[0]=$MAIN_IF \
  --set controller.extraArgs.v=2
```

Следующая команда конфигурирует `nginx` в кластере. `nginx` будет принимать все входящие запросы и перенаправлять их в `gateway`, который распределяет входящие запросы между компонентами сервера отчётов.

```
HOST=my.server-server.ru
NAMESPACE=fr-corporate-server

cat <<EOF | kubectl apply -n $NAMESPACE -f -
kind: Ingress
apiVersion: networking.k8s.io/v1
metadata:
  name: $HOST-gateway
  namespace: $NAMESPACE
  annotations:
    ingress.kubernetes.io/ssl-redirect: 'true'
    nginx.ingress.kubernetes.io/configuration-snippet: |
      add_header X-Robots-Tag "noindex, nofollow, nosnippet, noarchive";
    nginx.ingress.kubernetes.io/limit-rps: '50'
    nginx.ingress.kubernetes.io/proxy-body-size: '0'
    nginx.ingress.kubernetes.io/proxy-buffering: 'off'
    nginx.ingress.kubernetes.io/proxy-request-buffering: 'off'
spec:
  ingressClassName: nginx
  tls:
    - hosts:
      - $HOST
      secretName: corporate-tls-secret
  rules:
    - host: $HOST
      http:
        paths:
          - path: /
            pathType: ImplementationSpecific
            backend:
              service:
                name: fr-gateway
                port:
                  number: 80
EOF
```

Далее понадобится SSL сертификат для настройки защищённого соединения. Обычно его можно купить/получить у регистратора доменного имени или купить у удостоверяющего центра. При использовании сервера отчётов в интранете можно создать самостоятельно подписанный сертификат с помощью следующей команды:

```
export CERT_NAME=my.server-server.ru
openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -keyout del_me_file.key -out del_me_file.cer -subj
"/CN=$CERT_NAME/O=$CERT_NAME"
```

Регистрация сертификата в Kubernetes с именем `fr-corporate-tls`. Далее этот сертификат используется различными компонентами отчёта, в том числе `nginx`.

```
kubectl create secret tls fr-corporate-tls --key del_me_file.key --cert del_me_file.cer
```

Регистрация `nginx-ingress` в кластере Kubernetes.

Необходимо установить переменную `HOST`, соответствующую доменному имени сервера отчётов.

```
cat <<EOF | kubectl apply -n $NAMESPACE -f -
kind: Ingress
apiVersion: networking.k8s.io/v1
metadata:
  name: $HOST-gateway
  namespace: $NAMESPACE
  annotations:
    ingress.kubernetes.io/ssl-redirect: 'true'
    nginx.ingress.kubernetes.io/configuration-snippet: |
      add_header X-Robots-Tag "noindex, nofollow, nosnippet, noarchive";
    nginx.ingress.kubernetes.io/limit-rps: '50'
    nginx.ingress.kubernetes.io/proxy-body-size: '0'
    nginx.ingress.kubernetes.io/proxy-buffering: 'off'
    nginx.ingress.kubernetes.io/proxy-request-buffering: 'off'
spec:
  ingressClassName: nginx
  tls:
    - hosts:
      - $HOST
      secretName: corporate-tls-secret
  rules:
    - host: $HOST
      http:
        paths:
          - path: /
            pathType: ImplementationSpecific
            backend:
              service:
                name: fr-gateway
                port:
                  number: 80
EOF
```

# Лог установки helm и пакетов №2

Продукты: [МоиОтчеты](#) [Корпоративный Сервер](#)

Kubernetes-Dashboard это WEB-интерфейс для управления кластером. Он позволяет с помощью WEB-браузера управлять Kubernetes. Dashboard не является необходимым компонентом для работы сервера отчётов, однако он позволяет упростить настройку и конфигурирование сервера, а также мониторить состояние узлов и контейнеров.

## Установка Dashboard

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-rc7/aio/deploy/recommended.yaml
```

## Конфигурирование Dashboard

При конфигурации Dashboard используется переменная `SERVER_DOMAIN_NAME`. Обратите внимание, нижеприведённая конфигурация использует Dashboard на том же самом IP адресе, что и сервер отчётов. Для захода на страницу Dashboard используется правило, определяющее путь к Dashboard.

```
export SERVER_DOMAIN_NAME="my.server-server.ru"
```

```
cat <<EOF | kubectl apply -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: fastreport.cloud-dashboard
  namespace: kubernetes-dashboard
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/backend-protocol: HTTPS
    nginx.ingress.kubernetes.io/use-regex: "true"
    nginx.ingress.kubernetes.io/rewrite-target: "^$2"
    nginx.ingress.kubernetes.io/configuration-snippet: |
      rewrite ^/(dashboard)$ \ $1/ redirect;

spec:
  tls:
  - hosts:
    - $SERVER_DOMAIN_NAME
    secretName: fr-corporate-tls
  rules:
  - host: $SERVER_DOMAIN_NAME
    http:
      paths:
      - path: /dashboard(/|$)(.*)
        backend:
          serviceName: kubernetes-dashboard
          servicePort: 443
EOF
```

Проверить созданную конфигурацию nginx сервера можно с помощью следующих команд, используя вместо

```
nginx-ingress-controller-6674ff5868-t47xk
```

 имя созданного nginx контейнера:

```
kubectl exec -it nginx-ingress-controller-6674ff5868-t47xk -n nginx -- ls /etc/nginx/
kubectl exec -it nginx-ingress-controller-6674ff5868-t47xk -n nginx -- cat /etc/nginx/nginx.conf
```

Далее необходимо получить token, который будет использоваться для административного доступа к Dashboard. Создание административной учётной записи:

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: ServiceAccount
metadata:
  name: dashboard-admin-user
  namespace: kube-system
EOF
```

Назначение прав доступа административной учётной записи:

```
cat <<EOF | kubectl apply -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: dashboard-admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: dashboard-admin-user
  namespace: kube-system
EOF
```

Создание токена:

```
kubectl -n kubernetes-dashboard create token dashboard-admin-user -n kube-system
```

Распаковка и вывод на экран токена, используемого для авторизации на Dashboard:

```
kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep dashboard-admin-user | awk '{print $1}')
```

Сохраните токен для последующего использования в недоступном для посторонних месте и используйте его для авторизации в Dashboard.

Пример URL для доступа к dashboard: <https://my.server-server.ru/dashboard/>

# Лог установки helm и пакетов №3

Продукты: [МоиОтчеты](#) [Корпоративный Сервер](#)

Grafana это компонент аналитики и интерактивной визуализации для мониторинга состояния кластера. Это необязательный компонент для разворачивания сервера отчётов, но он может быть полезен для контроля состояния сервера.

Loki это агрегатор логов, который используется совместно с Grafana.

## Создание пространства имён для компонентов мониторинга

```
kubectl create namespace monitoring
```

## Обновление чартов для установки компонентов мониторинга

```
helm repo add loki https://grafana.github.io/loki/charts
helm repo update
```

## Установка Loki

```
helm upgrade \
--install loki loki/loki \
--namespace=monitoring \
--set persistence.enabled=true \
--set persistence.storageClassName=hcloud-volumes
```

```
helm upgrade \
--install promtail loki/promtail \
--namespace=monitoring \
--set loki.serviceName=loki.monitoring
```

## Установка Grafana

Не забудьте установить переменную `SERVER_DOMAIN_NAME`.

```
export SERVER_DOMAIN_NAME="<доменное имя сервера>"

helm upgrade \
--install grafana stable/grafana \
--namespace=monitoring \
--set persistence.enabled=true \
--set persistence.storageClassName=hcloud-volumes \
--set 'grafana.ini'.server.serve_from_sub_path=true \
--set 'grafana.ini'.server.root_url=https://$SERVER_DOMAIN_NAME/grafana/ \
--set 'grafana.ini'.server.domain=$SERVER_DOMAIN_NAME \
--set readinessProbe.httpGet.path=/grafana/api/health \
--set livenessProbe.httpGet.path=/grafana/api/health \
```

## Получение пароля от Grafana

```
kubectl get secret --namespace monitoring grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo
```

```
admin
*****
```

## Добавление ingress

Этот блок конфигурации позволяет расположить Grafana на том же самом сетевом интерфейсе, что и сервер отчётов и обращаться к ней по ссылке вида: <https://my.server-server.ru/grafana>

```
cat <<EOF | kubectl apply -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: fastreport.cloud-grafana
  namespace: monitoring
  annotations:
    kubernetes.io/ingress.class: "nginx"
    ingress.kubernetes.io/ssl-redirect: "true"
spec:
  tls:
  - hosts:
    - $SERVER_DOMAIN_NAME
    secretName: fr-corporate-tls
  rules:
  - host: $SERVER_DOMAIN_NAME
    http:
      paths:
      - path: /grafana
        backend:
          serviceName: grafana
          servicePort: 80
EOF
```

Для завершения конфигурации Grafana необходимо зайти на её страницу, авторизоваться и добавить ссылку на Loki

<http://loki.monitoring:3100>

# Установка брокера сообщений RabbitMQ

Продукты: [МоиОтчеты](#) [Корпоративный Сервер](#)

Для установки RabbitMQ необходимо определить переменную `RABBITMQ_PASSWORD`, содержащую пароль. Также можно настроить размер тома для очереди брокера сообщений.

Добавление чарта и обновление helm репозитория.

```
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo update
```

Создать пространство имён для RabbitMQ.

```
kubectl create namespace rabbitmq
```

Установка RabbitMQ может варьироваться от типа установки. Ниже приведены два примера - для облачного провайдера и для установки на тестовой конфигурации.

## Установка на облачном хостинге

Нижеприведённый скрипт использует размер очереди 10 Гб.

```
export RABBITMQ_PASSWORD="<пароль для RabbitMQ>"

cat <<EOF > ./rabbitmq.yaml
rabbitmq:
  extraConfiguration: |-
    #disk_free_limit.absolute = 10GB
    management.path_prefix = /rabbit
    #management.load_definitions = /app/load_definition.json
  persistence:
    enabled: true
    storageClass: hcloud-volumes
    size: 10Gi
  livenessProbe:
    commandOverride:
      - sh
      - -c
      - rabbitmq-api-check "http://user:RABBITMQ_PASSWORD@127.0.0.1:15672/rabbit/api/healthchecks/node" '{"status":"ok"}'
  readinessProbe:
    commandOverride:
      - sh
      - -c
      - rabbitmq-api-check "http://user:RABBITMQ_PASSWORD@127.0.0.1:15672/rabbit/api/healthchecks/node" '{"status":"ok"}'
EOF

helm upgrade \
  --install rabbitmq bitnami/rabbitmq \
  --namespace=rabbitmq \
  --values ./rabbitmq.yaml

rm ./rabbitmq.yaml
```

## Пример минимальной установки RabbitMQ для разворачивания на тестовом кластере

Нижеприведённая конфигурация использует размер очереди 4 Гб.

Обратите внимание - данный пример является минимальным для запуска МоиОтчеты Корпоративный Сервер и описанная конфигурация томов не является отказоустойчивой.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  namespace: "rabbitmq"
  name: pv-for-rmq
  labels:
    name: mongo-volume-1-0-0
spec:
  storageClassName: manual
  capacity:
    storage: 4Gi
  accessModes:
  - ReadWriteOnce
  hostPath:
    path: /devkube/rabbitmq
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: "rabbitmq-pvc"
  namespace: "rabbitmq"
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: manual
  resources:
    requests:
      storage: 4Gi
```

Выполните `kubectl apply -f <имя вышеприведённого yaml>` и собственно скрипт установки RabbitMQ. Обратите внимание, что используется ранее созданный PersistentVolumeClaim.

```
helm upgrade \
--namespace=rabbitmq \
--install rabbitmq bitnami/rabbitmq \
--set persistence.existingClaim=rabbitmq-pvc \
--set volumePermissions.enabled=true \
--set ingress.tlsSecret="fr-corporate-tls" \
| tee RabbitMQ.txt
```

## Добавление конфигурации ingress

Этот пункт необходим если вы хотите получить доступ к контрольной панели сервиса RabbitMQ извне кластера.

```
export SERVER_DOMAIN_NAME="<доменное имя сервера>"
```

```
cat <<EOF | kubectl apply -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: fastreport.cloud-rabbitmq
  namespace: rabbitmq
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/rewrite-target: "^/$2"
    nginx.ingress.kubernetes.io/configuration-snippet: |
      rewrite ^(/rabbit)\$ \$1/ redirect;

spec:
  tls:
    - hosts:
      - $SERVER_DOMAIN_NAME
      secretName: fr-corporate-tls
  rules:
    - host: $SERVER_DOMAIN_NAME
      http:
        paths:
          - path: /rabbit
            backend:
              serviceName: rabbitmq
              servicePort: 15672
EOF
```

Доступ по  .

# Установка MongoDB в кластер

Продукты: [МоиОтчеты](#) [Корпоративный Сервер](#)

## Хранилище данных с открытым исходным кодом

На данной странице рассказывается о варианте установки с использованием MongoDB. Альтернативой выступает FerretDB - аналог с открытым исходным кодом, который использует PostgreSQL как движок базы данных. Инструкцию по установке с использованием FerretDB можно прочитать [по ссылке](#).

## Начало установки

Добавление репозитория для helm:

```
helm repo add bitnami bitnami/mongodb
helm repo update
```

Сервер отчётов использует MongoDB для хранения шаблонов отчётов, подготовленных отчётов и различных документов, экспортированных из подготовленных отчётов.

Подготовка пространства имён:

```
NAMESPACE=mongo
MONGO_USER=any_name
MONGO_PASS=mongo_password
MONGO_DB_SIZE=10Gi
MONGO_SERVICE_NAME=fr-mongo
kubectl create namespace $NAMESPACE
```

Обратите внимание на переменную MONGO\_SERVICE\_NAME - это имя в дальнейшем будет использоваться для обращения к MongoDB из сервера отчётов.

Дальнейшая настройка будет отличаться от того, используется ли хранилище облачного провайдера или локальное хранилище. Ниже приведено два примера. Используйте только один из них, или модифицируйте настройки для своего облачного провайдера.

## Настройка хранилища для облачного провайдера

```
helm install $MONGO_SERVICE_NAME bitnami/mongodb \
--namespace $NAMESPACE \
--set persistence.enabled=true \
--set persistence.storageClass=hcloud-volumes \
--set persistence.size=$MONGO_DB_SIZE \
--set auth.username=$MONGO_USER \
--set auth.password=$MONGO_PASS \
| tee MongoDB.txt
```

## Настройка локального хранилища

StorageClass предоставляет средства для администраторов для описания "классов" хранилищ, которые они предоставляют.

```
MONGO_DATA_DIR=/devkube/mongodb
MONGO_DB=reports_db # Можно использовать любое имя

##
## Create StorageClass for MongoDB
##
cat <<EOF | kubectl apply -n $NAMESPACE -f -
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: local-sc-mongodb
provisioner: kubernetes.io/no-provisioner
volumeBindingMode: WaitForFirstConsumer
EOF

##
## Create Persistent volume for MongoDB
##

cat <<EOF | kubectl apply -n $NAMESPACE -f -
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mongo-storage
  namespace: $NAMESPACE
  labels:
    type: local
spec:
  capacity:
    storage: $MONGO_DB_SIZE
  accessModes:
    - ReadWriteOnce
  storageClassName: local-sc-mongodb
  local:
    path: $MONGO_DATA_DIR
  nodeAffinity:
    required:
      nodeSelectorTerms:
        - matchExpressions:
            - key: kubernetes.io/hostname
              operator: In
              values:
                - altlinux-10-2
EOF
```

Для установки MongoDB используйте следующий скрипт:

```
helm install $MONGO_SERVICE_NAME bitnami/mongodb \
  --namespace $NAMESPACE \
  --set persistence.storageClass=local-sc-mongodb \
  --set auth.username=$MONGO_USER \
  --set auth.password=$MONGO_PASS \
  --set auth.database=$MONGO_DB | tee MongoDB.txt
```

Результатом выполнения этого скрипта, помимо установки MongoDB в кластер, будет создан файл, содержащий информацию о результате установки базы данных. Обратите внимание, что параметры, передаваемые скрипту, были описаны в самом начале этого документа.

Проверить корректность установки MongoDB проще всего при помощи Kubernetes Dashboard, установка которого была описана ранее.

# Установка FerretDB в кластер

Продукты: [МоиОтчеты](#) [Корпоративный Сервер](#)

## MongoDB как хранилище данных

На данной странице рассказывается о варианте установки с использованием FerretDB - хранилище с открытым исходным кодом, которое использует PostgreSQL как движок базы данных. Альтернативно вы можете установить MongoDB. Инструкцию по установке с использованием MongoDB можно прочитать [по ссылке](#).

## Разница с MongoDB

Вы не почувствуете значительных изменений и сможете общаться с базой аналогично MongoDB (в том числе используя MongoDB Compass и утилиту `mongosh`). Единственное значительное различие - FerretDB не поддерживает роли, поэтому создание новых пользователей базы данных должно происходить следующим образом:

```
db.createUser({
  user: 'newuser',
  pwd: 'newpassword',
  roles: []
})
```

## Начало установки

Подготовка пространства имён:

```
FERRET_USER=any_name
FERRET_PASS=ferret_password
FERRET_SERVICE_NAME=fr-ferret
NAMESPACE=ferret
kubectl create namespace $NAMESPACE
```

Обратите внимание на переменную `FERRET_SERVICE_NAME` - это имя в дальнейшем будет использоваться для обращения к FerretDB из сервера отчётов.

## Создание Deployment

```
cat <<EOF | kubectl apply -n $NAMESPACE -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: $FERRET_SERVICE_NAME
spec:
  replicas: 1
  selector:
    matchLabels:
      app: $FERRET_SERVICE_NAME
  template:
    metadata:
      labels:
        app: $FERRET_SERVICE_NAME
    spec:
      containers:
        - name: $FERRET_SERVICE_NAME
          image: ghcr.io/ferretdb/ferretdb-eval:2
          ports:
            - containerPort: 27017
          env:
```

```

- name: POSTGRES_USER
  value: $FERRET_USER
- name: POSTGRES_PASSWORD
  value: $FERRET_PASS
volumeMounts:
- mountPath: /var/lib/postgresql/data
  name: volume
volumes:
- name: volume
  persistentVolumeClaim:
    claimName: fr-ferret-pvc
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ferret-storageclass
provisioner: kubernetes.io/no-provisioner
volumeBindingMode: WaitForFirstConsumer
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: ferret-storage
  namespace: $NAMESPACE
labels:
  type: local
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  storageClassName: ferret-storageclass
  local:
    path: /devkube/ferretdb
  nodeAffinity:
    required:
      nodeSelectorTerms:
        - matchExpressions:
            - key: kubernetes.io/hostname
              operator: In
              values:
                - node1
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: fr-ferret-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: ferret-storageclass
  resources:
    requests:
      storage: 2Gi
---
apiVersion: v1
kind: Service
metadata:
  name: $FERRET_SERVICE_NAME
labels:
  app: $FERRET_SERVICE_NAME
spec:
  selector:
    app: $FERRET_SERVICE_NAME
  ports:

```

```
- port: 27017  
  targetPort: 27017  
EOF
```

Проверить корректность установки FerretDB проще всего при помощи Kubernetes Dashboard, установка которого была описана ранее.

## Внимание

Проверьте создалась ли на ноде директория `/devkube/ferretdb`, если нет - создайте её самостоятельно.

# Установка компонентов сервера отчётов МоиОтчеты Корпоративный Сервер на узлы кластера Kubernetes

Продукты: [МоиОтчеты Корпоративный Сервер](#)

## Получение образов Docker

Образы компонентов доступны в реестре по следующим URL. Вы можете загрузить их с помощью команд `docker pull`:

```
docker pull xn--80akiaokt3b4b.xn--90aia9aifhdb2cxbdg.xn--p1ai/repository/docker-registry/moiotchety-corporate-server-app:latest
docker pull xn--80akiaokt3b4b.xn--90aia9aifhdb2cxbdg.xn--p1ai/repository/docker-registry/moiotchety-corporate-server-backend:latest
docker pull xn--80akiaokt3b4b.xn--90aia9aifhdb2cxbdg.xn--p1ai/repository/docker-registry/moiotchety-corporate-server-designer:latest
docker pull xn--80akiaokt3b4b.xn--90aia9aifhdb2cxbdg.xn--p1ai/repository/docker-registry/moiotchety-corporate-server-fonts:latest
docker pull xn--80akiaokt3b4b.xn--90aia9aifhdb2cxbdg.xn--p1ai/repository/docker-registry/moiotchety-corporate-server-gateway:latest
docker pull xn--80akiaokt3b4b.xn--90aia9aifhdb2cxbdg.xn--p1ai/repository/docker-registry/moiotchety-corporate-server-homer:latest
docker pull xn--80akiaokt3b4b.xn--90aia9aifhdb2cxbdg.xn--p1ai/repository/docker-registry/moiotchety-corporate-server-scheduler:latest
docker pull xn--80akiaokt3b4b.xn--90aia9aifhdb2cxbdg.xn--p1ai/repository/docker-registry/moiotchety-corporate-server-static-preview:latest
docker pull xn--80akiaokt3b4b.xn--90aia9aifhdb2cxbdg.xn--p1ai/repository/docker-registry/moiotchety-corporate-server-wasm-preview:latest
docker pull xn--80akiaokt3b4b.xn--90aia9aifhdb2cxbdg.xn--p1ai/repository/docker-registry/moiotchety-corporate-server-workercore:latest
```

**Примечание:** Загружать образы на каждую ноду вручную не обязательно. Достаточно указать нужные образы и теги в манифестах Kubernetes (`Deployment`, `StatefulSet` и т.д.). Kubernetes сам скачает необходимые образы при запуске подов.

## Описание микросервисов

НАЗВАНИЕ СЕРВИСА	НАЗНАЧЕНИЕ
<code>app</code>	Панель пользователя — интерфейс для работы с отчётами.
<code>backend</code>	Основное ядро системы — обрабатывает запросы, управляет доступом к БД.
<code>designer</code>	Дизайнер отчётов — инструмент для создания и редактирования отчётов.
<code>fonts</code>	Сервис обработки шрифтов, необходим для корректного отображения отчётов.
<code>gateway</code>	Шлюз — центральная точка входа для всех HTTP-запросов.
<code>homer</code>	Панель администрирования — управление пользователями, настройками и т.д.
<code>scheduler</code>	Менеджер задач — управляет расписанием выполнения отчётов.
<code>static-preview</code>	Фронтенд для просмотра отчётов в статичном виде.
<code>wasm-preview</code>	Фронтенд для интерактивного просмотра отчётов (только для C# отчётов).
<code>workercore</code>	Ядро обработки отчётов — выполняет генерацию и построение отчётов.

## Поддерживаемые теги образов

ТЕГ	ОПИСАНИЕ
<code>:latest</code>	Последняя версия на основе контейнеров Debian.
<code>:debian-latest</code>	Последняя версия на основе контейнеров Debian.
<code>:redos-latest</code>	Последняя версия на основе контейнеров RedOS (ubi).
<code>:debian-{version}</code>	Конкретная версия на основе Debian, например: <code>:debian-2026.2.2</code> .
<code>:redos-{version}</code>	Конкретная версия на основе RedOS, например: <code>:redos-2026.2.2</code> .

**Важно:**

- Используйте **конкретные версии** образов (например, `:debian-2026.2.2`), а не `:latest`. Это обеспечит стабильность и предсказуемость при развертывании.
- Все микросервисы должны использовать **одинаковую версию** образов для обеспечения совместимости.

## Дополнительные замечания

- Обновления могут не затрагивать все микросервисы сразу. Однако для стабильной работы рекомендуется обновлять все компоненты одновременно.
- В случае проблем с совместимостью — обращайтесь к технической поддержке.

# Настройка сертификатов

Продукты: [МоиОтчеты](#) [Корпоративный Сервер](#)

Для работы сервера отчётов необходимо создать сертификат.

Чтобы создать самостоятельно подписанный сертификат, используйте следующие команды:

```
#!/bin/sh

openssl req -x509 -nodes -new -sha256 -days 1024 -newkey rsa:2048 -keyout RootCA.key -out RootCA.pem -subj "/C=US/CN=debian-master.fast-report.com"
openssl x509 -outform pem -in RootCA.pem -out RootCA.crt
openssl pkcs12 -export -out ./certificate.pfx -inkey RootCA.key -in RootCA.crt
```

Чтобы создать секрет, используйте следующие команды:

```
NAMESPACE=fr-corporate-server
SECRET_VOLUME_NAME=corporate-volume-secret
kubectl create namespace $NAMESPACE
kubectl create secret generic $SECRET_VOLUME_NAME -n $NAMESPACE --from-file=certificate.pfx
```

## Альтернативный способ генерации сертификата, предложенный пользователем продукта

1. Создайте и отредактируйте файл `san.cnf`:

```
[ req ]
default_bits = 2048
default_md = sha256
distinguished_name = req_distinguished_name
req_extensions = v3_req
[ req_distinguished_name ]
countryName = CN # C=
stateOrProvinceName = Shanghai # ST=
localityName = MyCity # L=
#postalCode = 200000 # L/postalcode=
#streetAddress = "My Address" # L/street=
organizationName = My Corporation # O=
organizationalUnitName = My Department # OU=
commonName = myname.mysoftware.mycorporation.com # CN=
emailAddress = myname@example.com # CN/emailAddress=
[ v3_req ]
subjectAltName = @alt_names
[ alt_names ]
DNS.1 = myname.mysoftware.mycorporation.com
#DNS.2 = other2.com
#DNS.3 = other3.coM
```

2. Сгенерируйте сертификат:

```
openssl req -x509 -nodes -days 365 -subj "/C=CN/ST=Shanghai/L=Shanghai/O=My Corporation/OU=My Department/CN=myname.mysoftware.mycorporation.com/emailAddress=myname@example.com" -keyout privateKey.pem -out public.crt -config san.cnf -extensions v3_req
openssl pkcs12 -export -out ./certificate.pfx -inkey privateKey.key -in public.crt
```

3. Создайте секрет в облачном сервере:

```
NAMESPACE=fr-corporate-server
```

```
SECRET_VOLUME_NAME=corporate-volume-secret
```

```
kubectl create namespace $NAMESPACE
```

```
kubectl create secret generic $SECRET_VOLUME_NAME -n $NAMESPACE --from-file=certificate.pfx
```

# Настройка параметров МоиОтчеты Корпоративный Сервер

Продукты: [МоиОтчеты Корпоративный Сервер](#)

Данный раздел описывает процесс конфигурирования сервера отчётов. Здесь описана базовая информация. Для тонкой настройки сервера обратитесь к руководству `admi_config`. Конфигурация сохраняется в защищенном хранилище Kubernetes. Эту операцию оптимально выполнить до старта контейнеров, описанных в секции `fr-corporate-server`.

Базовые параметры сервера отчётов определены ниже. Для удобства они показаны в отдельной секции. Возможные варианты: поместить этот фрагмент в отдельный файл и с помощью `операции точка` внедрить их в конфигурационный скрипт. Или же скопировать этот фрагмент непосредственно в конфигурационный скрипт. Так возможно дописать `export` перед каждой переменной и экспортировать в переменные среды.

Значение переменной `MONGO_HOST` (или `FERRET_HOST`) можно проверить в Kubernetes Dashboard - оно соответствует имени сервиса и его пространству имён, разделённых точкой. Для успешного старта сервера отчётов необходима лицензия - специальная строка, описывающая права использования сервера отчётов. Получите её у компании ООО "Быстрые отчеты".

Значения параметров `MONGO_USER` и `MONGO_PASS` (или `FERRET_USER` и `FERRET_PASS`) должны соответствовать тем же самым параметрам, что указаны при установке MongoDB (FerretDB).

Вариант для MongoDB:

```

NAMESPACE=fr-corporate-server
CLOUD_ENV=prod

## Пароль и имя пользователя могут быть установлены любые, главное, чтобы совпадали с теми, что были заданы при установке
MongoDB.
## MONGO_DB должна быть admin.
## Однако, есть правила установки имени хоста - оно образуется из имени сервиса и пространства имён, в котором
зарегистрирована MongoDB.

MONGO_HOST=fr-mongo-mongodb.mongo
MONGO_USER=root
MONGO_PASS=mxzgrkGvvk
MONGO_DB=admin

## Следующие параметры лучше ЗДЕСЬ не изменять. Хост и имя пользователя устанавливаются автоматически, а
## пароль для RabbitMQ генерируется автоматически на момент установки и сохраняется в хранилище секретов Kubernetes.
RABBITMQ_HOST=rabbitmq.rabbitmq.svc.cluster.local
# RABBITMQ_HOST=rabbitmq-0.rabbitmq-headless.rabbitmq
RABBITMQ_USER=user
RABBITMQ_PASS=$(kubectl get secret --namespace rabbitmq rabbitmq -o jsonpath="{.data.rabbitmq-password}" | base64 --decode)

## Разработчики утверждают, что здесь может быть любая последовательность символов. И длина не определена.
## Для теста были использованы следующие значения, и установка прошла успешно

DESIGNER_SECRET=yXgQDpNzohhN3dAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHrXiepieQtOaLwdqfRijMaZwy5p27tho7XK4C
WORKER_SECRET=9bYuYu4cvt73oWD3AAAAAAAAAAAAAAAAAAAAAAAAAAAAA5Z6i6pRzkR8mKGL1pTRnns3P0clxzTuVu
GATEWAY_SECRET=R0jEINg6OA4TssyAAAAAAAAAAAAAAAAAAAAAAAAAAAAAY46ZtsYowl7gB9yteZNMsvPoL6sGgiWC4
SCHEDULER_SECRET=fcfFZNWC3zyBmAAAAAAAAAAAAAAAAAAAAAAAAAAAAA5gD2MeexV3svLdy87qC6bX3bM2W6mZjRC

## В эту переменную должна быть вписана лицензия, полученная у компании ООО "Быстрые отчеты". Увы, без лицензии сервер
отчётов работать не будет.
## Не используйте кавычки.

LICENSE=

## Строка подключения к MongoDB формируется автоматически на основе ранее определённых переменных. Тут лучше ничего не
менять.

MONGO_ACCESS="mongodb://$MONGO_USER:$MONGO_PASS@$MONGO_HOST:27017/"
CONNECTION_STRING="$MONGO_ACCESS?
authSource=$MONGO_DB&readPreference=primary&appName=MongoDB%20Compass&ssl=false&maxPoolSize=100&waitQueueMultiple=
100"

```

Вариант для FerretDB:

```

NAMESPACE=fr-corporate-server
CLOUD_ENV=prod

## Пароль и имя пользователя могут быть установлены любые, главное, чтобы совпадали с теми, что были заданы при установке FerretDB.
## FERRET_DB должна быть admin.
## Однако, есть правила установки имени хоста - оно образуется из имени сервиса и пространства имён, в котором зарегистрирована FerretDB.

FERRET_HOST=fr-ferret.ferret
FERRET_USER=any_name
FERRET_PASS=ferret_password
FERRET_DB=admin

## Следующие параметры лучше ЗДЕСЬ не изменять. Хост и имя пользователя устанавливаются автоматически, а
## пароль для RabbitMQ генерируется автоматически на момент установки и сохраняется в хранилище секретов Kubernetes.
RABBITMQ_HOST=rabbitmq.rabbitmq.svc.cluster.local
# RABBITMQ_HOST=rabbitmq-0.rabbitmq-headless.rabbitmq
RABBITMQ_USER=user
RABBITMQ_PASS=$(kubectl get secret --namespace rabbitmq rabbitmq -o jsonpath="{.data.rabbitmq-password}" | base64 --decode)

## Разработчики утверждают, что здесь может быть любая последовательность символов. И длина не определена.
## Для теста были использованы следующие значения, и установка прошла успешно

DESIGNER_SECRET=yXgQDpNzohhN3dAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHrXiepieQtOaLwdqfRijMaZwy5p27tho7XK4C
WORKER_SECRET=9bYuYu4cvt73oWD3AAAAAAAAAAAAAAAAAAAAAAAAAAAAA5Z6i6pRzkR8mKGL1pTRnns3P0clxzTuVu
GATEWAY_SECRET=R0jEINg6OA4TssyAAAAAAAAAAAAAAAAAAAAAAAAAAAAAY46ZtsYowl7gB9yteZNMsvPoL6sGgiWC4
SCHEDULER_SECRET=fcfFZNWC3zyBmAAAAAAAAAAAAAAAAAAAAAAAAAAAAA5gD2MeexV3svLdy87qC6bX3bM2W6mZjRC

## В эту переменную должна быть вписана лицензия, полученная у компании ООО "Быстрые отчёты". Увы, без лицензии сервер отчётов работать не будет.
## Не используйте кавычки.

LICENSE=

## Строка подключения к FerretDB формируется автоматически на основе ранее определённых переменных. Тут лучше ничего не менять.

FERRET_ACCESS="mongodb://$FERRET_USER:$FERRET_PASS@$FERRET_HOST:27017/"
CONNECTION_STRING="$FERRET_ACCESS?
authSource=$FERRET_DB&readPreference=primary&appName=MongoDB%20Compass&ssl=false&maxPoolSize=100&waitQueueMultiple=100"

```

Регистрация пространства имён сервера отчётов в кластере:

```
kubectl create namespace $NAMESPACE
```

Создание минимальной конфигурации сервера отчётов:

```

cat <<EOF | kubectl apply -n $NAMESPACE -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: fast-report-config
data:
  appsettings.Production.json: |
    {
      "Auth": {
        "UseOpenId": false,
        "UseLocal": true
      },

```



```

"Namespace": "$NAMESPACE",
"HostType": "WebApp",
"PathBase": "/staticpreview",
"Type": "K8s",
"PingPath": "/staticpreview/",
"IsSignInRequired": false,
"Priority": 21
},
"Default": {
  "Namespace": "$NAMESPACE"
},
"HomerApp": {
  "Namespace": "$NAMESPACE",
  "WhiteListClaims": {
    "cloud_service_access": "super_user"
  }
}
},
"Designer": {
  "BackendUrl": "http://fr-backend.$NAMESPACE:80",
  "InternalKey": "$DESIGNER_SECRET"
},
"WorkerCore": {
  "BackendUrl": "http://fr-backend.$NAMESPACE:80",
  "InternalKey": "$WORKER_SECRET"
},
"Scheduler": {
  "BackendUrl": "http://fr-backend.$NAMESPACE:80",
  "InternalKey": "$SCHEDULER_SECRET"
}
}
EOF

```

Используя FerretDB замените

```

"Database": {
  "ConnectionString": "$CONNECTION_STRING",
  "DatabaseName": "$MONGO_DB"
}

```

на

```

"Database": {
  "ConnectionString": "$CONNECTION_STRING",
  "DatabaseName": "$FERRET_DB"
}

```

Обратите внимание, вышеприведённый фрагмент shell-скрипта содержит базовые настройки, проверенные разработчиками. При первоначальном разворачивании сервера отчётов рекомендуется использовать их, чтобы убедиться в работоспособности сервера. Перед изменением настроек сохраните эту версию, таким образом вы сэкономите значительно время при тонкой настройке сервера, используя параметры, описанные в секции `admin_config`.

# Установка МоиОтчеты Корпоративный Сервер

Продукты: [МоиОтчеты Корпоративный Сервер](#)

Минимальная конфигурация корпоративного сервера отчётов состоит из девяти компонентов, описанных ниже. Перед установкой необходимо определить переменные, например, в файле `config.sh`. Во избежание конфликтов имён лучше иметь глобальный файл конфигурации для всех скриптов отчёта. Ниже показан пример инициализации переменных, используемых скриптами установки сервера отчётов:

```
IMAGE_STORAGE=xn--80akiaokt3b4b.xn--90aia9aifhdb2cxbdg.xn--p1ai/repository/docker-registry
IMAGE_TAG=debian-2026.2.2
NAMESPACE=fr-corporate-server
SECRET_VOLUME_NAME=corporate-volume-secret
IMAGE_REGISTRY_SECRET_NAME=storage_secret
SECRET_VOLUME_NAME=corporate-volume-secret
PULL_POLICY=Never
```

Описание переменных:

`IMAGE_STORAGE` - адрес хранилища образов сервисов сервера отчётов, Корпоративный Сервер использует локальное хранилище, наполнение образами которого описано в разделе [Подготовка образов](#).

`IMAGE_TAG` - версия сервера отчётов.

`NAMESPACE` - пространство имён Kubernetes, в котором будет выполняться сервер отчётов. `SECRET_VOLUME_NAME` - имя тома, на котором будет храниться конфигурация и информация для авторизации.

`IMAGE_REGISTRY_SECRET_NAME` - имя записи, сохраняющей информацию для авторизации на сервере, хранящем образы сервисов сервера отчётов. `PULL_POLICY` - определяет как Kubernetes будет скачивать образы сервисов.

Возможные значения:

- Always - всегда скачивать образы при старте сервиса;
- IfNotPresent - скачивать образ только если он не присутствует локально;
- Never - использовать заранее импортированные образы, никогда не скачивать их с внешнего сервера.

Обратите внимание: описанная конфигурация не скачивает образы из внешнего хранилища (`PULL_POLICY=Never`), эти переменные должны быть заданы во избежание ошибок при применении конфигурации. Перед установкой значения `PULL_POLICY` обратитесь к секции `fr-images` для ознакомления с настройками.

Настройка параметров сервера отчётов описана в документе [Конфигурация сервера отчётов](#). В этом документе описаны базовые параметры, определяющие режимы работы сервера отчётов.

## Gateway - сервис обработки входящих запросов

Gateway это сервис, обрабатывающий все соединения к серверу отчётов. Он анализирует запросы и распределяет их между другими сервисами. Если сервис Gateway не работает или работает неправильно, то ни один из компонентов сервера отчётов работать не будет. Ingress конфигурация по умолчанию отправляет все внешние запросы именно этому сервису. В случае каких-либо проблем с сервисом начните анализ проблемы с просмотром логов сервиса `fr-gateway`.

```
#!/bin/sh

HTTPS_NODE_PORT=32222
IMAGE="${IMAGE_STORAGE}/moiotchety-corporate-server-gateway:${IMAGE_TAG}"

cat <<EOF | kubectl apply -n $NAMESPACE -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: fr-gateway
  namespace: $NAMESPACE
```

```
labels:
  app.kubernetes.io/name: fr-gateway
rules:
- apiGroups: [""]
  resources:
    - services
    - endpoints
    - pods
  verbs: ["get", "list", "watch"]
```

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: fr-gateway
  namespace: $NAMESPACE
  labels:
    app.kubernetes.io/name: fr-gateway
```

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: fr-gateway
  namespace: $NAMESPACE
  labels:
    app.kubernetes.io/name: fr-gateway
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: fr-gateway
subjects:
- kind: ServiceAccount
  name: fr-gateway
  namespace: $NAMESPACE
```

```
---
apiVersion: v1
kind: Service
metadata:
  name: fr-gateway
  namespace: $NAMESPACE
spec:
  type: NodePort
  selector:
    app: fr-gateway
  ports:
    - name: http
      protocol: TCP
      port: 80
      nodePort: 32223
    - name: https
      protocol: TCP
      port: 5005
      nodePort: $HTTPS_NODE_PORT
```

```
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: fr-gateway
  namespace: $NAMESPACE
spec:
  replicas: 1
  selector:
```

```
matchLabels:
  app: fr-gateway
strategy:
  type: Recreate
template:
  metadata:
    namespace: $NAMESPACE
  labels:
    app: fr-gateway
spec:
  serviceAccountName: fr-gateway
  containers:
  - image: $IMAGE
    imagePullPolicy: $PULL_POLICY
    name: fr-gateway
    #env:
    # # Use secret in real usage
    #- name: MYSQL_ROOT_PASSWORD
    # value: password
  ports:
  - containerPort: 80
    name: fr-gateway
  volumeMounts:
  - name: config-volume
    mountPath: /app/appsettings.Production.json
    subPath: appsettings.Production.json
  - name: secret-volume
    mountPath: /etc/cert
  imagePullSecrets:
  - name: $IMAGE_REGISTRY_SECRET_NAME
  volumes:
  - name: config-volume
    configMap:
      name: fast-report-config
      items:
      - key: appsettings.Production.json
        path: appsettings.Production.json
  - name: secret-volume
    secret:
      secretName: $SECRET_VOLUME_NAME
EOF
```

## Установка планировщика сервера отчётов

```
IMAGE="{IMAGE_STORAGE}/moiotchety-corporate-server-scheduler:{IMAGE_TAG}"
```

```
cat <<EOF | kubectl apply -n $NAMESPACE -f -  
apiVersion: v1  
kind: Service  
metadata:  
  name: fr-scheduler  
  namespace: $NAMESPACE  
spec:  
  type: ClusterIP  
  ports:  
  - port: 80  
  selector:  
    app: fr-scheduler  
EOF
```

```
cat <<EOF | kubectl apply -n $NAMESPACE -f -  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: fr-scheduler  
  namespace: $NAMESPACE  
spec:  
  replicas: 1  
  selector:  
    matchLabels:  
      app: fr-scheduler  
  strategy:  
    type: Recreate  
  template:  
    metadata:  
      namespace: $NAMESPACE  
      labels:  
        app: fr-scheduler  
    spec:  
      containers:  
      - image: $IMAGE  
        imagePullPolicy: $PULL_POLICY  
        name: fr-scheduler  
        #env:  
        # # Use secret in real usage  
        # name: MYSQL_ROOT_PASSWORD  
        # value: password  
        ports:  
        - containerPort: 80  
          name: fr-scheduler  
        volumeMounts:  
        - name: config-volume  
          mountPath: /app/appsettings.Production.json  
        - name: secret-volume  
          mountPath: /etc/cert  
      imagePullSecrets:  
      - name: $IMAGE_REGISTRY_SECRET_NAME  
      volumes:  
      - name: config-volume  
        configMap:  
          name: fast-report-config  
      - name: secret-volume  
        secret:  
          secretName: $SECRET_VOLUME_NAME  
EOF
```

## Static Preview - сервис просмотра отчётов

```
#!/bin/sh

IMAGE="{IMAGE_STORAGE}/moiotchety-corporate-server-static-preview:{IMAGE_TAG}"

cat <<EOF | kubectl apply -n $NAMESPACE -f -
apiVersion: v1
kind: Service
metadata:
  name: fr-s-preview
  namespace: $NAMESPACE
spec:
  type: ClusterIP
  ports:
    - port: 80
  selector:
    app: fr-s-preview
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: fr-s-preview
  namespace: $NAMESPACE
spec:
  replicas: 1
  selector:
    matchLabels:
      app: fr-s-preview
  strategy:
    type: Recreate
  template:
    metadata:
      namespace: $NAMESPACE
    labels:
      app: fr-s-preview
    spec:
      containers:
        - image: $IMAGE
          resources:
            limits:
              memory: 200Mi
            requests:
              memory: 200Mi
          imagePullPolicy: $PULL_POLICY
          name: fr-s-preview
      #env:
      # # Use secret in real usage
      # - name: MYSQL_ROOT_PASSWORD
      #   value: password
      ports:
        - containerPort: 80
          name: fr-s-preview
      imagePullSecrets:
        - name: $IMAGE_REGISTRY_SECRET_NAME
EOF
```

## Сервис Homer

```
#!/bin/sh

IMAGE="${IMAGE_STORAGE}/moiotchety-corporate-server-homer:${IMAGE_TAG}"

SECRET_VOLUME_NAME=corporate-volume-secret

cat <<EOF | kubectl apply -n $NAMESPACE -f -
apiVersion: v1
kind: Service
metadata:
  name: fr-homer
  namespace: $NAMESPACE
spec:
  type: ClusterIP
  ports:
  - port: 80
  selector:
    app: fr-homer
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: fr-homer
  namespace: $NAMESPACE
spec:
  replicas: 1
  selector:
    matchLabels:
      app: fr-homer
  strategy:
    type: Recreate
  template:
    metadata:
      namespace: $NAMESPACE
    labels:
      app: fr-homer
    spec:
      containers:
      - image: $IMAGE
        imagePullPolicy: $PULL_POLICY
        name: fr-homer
        #env:
        # # Use secret in real usage
        # name: MYSQL_ROOT_PASSWORD
        # value: password
        ports:
        - containerPort: 80
          name: fr-homer
      imagePullSecrets:
      - name: $IMAGE_REGISTRY_SECRET_NAME
EOF
```

## Сервис Backend

```
#!/bin/sh

IMAGE="${IMAGE_STORAGE}/moiotchety-corporate-server-backend:${IMAGE_TAG}"

cat <<EOF | kubectl apply -n $NAMESPACE -f -
apiVersion: v1
kind: Service
metadata:
  name: fr-backend
```

```

namespace: $NAMESPACE
spec:
  type: ClusterIP
  ports:
  - port: 80
  selector:
    app: fr-backend
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: fr-backend
  namespace: $NAMESPACE
spec:
  replicas: 3
  selector:
    matchLabels:
      app: fr-backend
  strategy:
    type: Recreate
  template:
    metadata:
      namespace: $NAMESPACE
    labels:
      app: fr-backend
    spec:
      containers:
      - image: $IMAGE
        resources:
          limits:
            memory: 400Mi
          requests:
            memory: 400Mi
        imagePullPolicy: $PULL_POLICY
        name: fr-backend
        env:
        - name: Serilog__Using__0
          value: FastReport.Cloud.Base.Mvc
        ports:
        - containerPort: 80
          name: fr-backend
        volumeMounts:
        - name: config-volume
          mountPath: /app/appsettings.Production.json
          subPath: appsettings.Production.json
        imagePullSecrets:
        - name: $IMAGE_REGISTRY_SECRET_NAME
      volumes:
      - name: config-volume
        configMap:
          name: fast-report-config
          items:
          - key: appsettings.Production.json
            path: appsettings.Production.json
EOF

```

## Сервис Designer - онлайн дизайнер отчётов

```

#!/bin/sh

IMAGE="${IMAGE_STORAGE}/moiotchety-corporate-server-designer:${IMAGE_TAG}"

cat <<EOF | kubectl apply -n $NAMESPACE -f -
apiVersion: v1

```

```
kind: Service
metadata:
  name: fr-designer
  namespace: $NAMESPACE
spec:
  type: ClusterIP
  ports:
    - port: 80
  selector:
    app: fr-designer
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: fr-designer
  namespace: $NAMESPACE
spec:
  replicas: 1
  selector:
    matchLabels:
      app: fr-designer
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: fr-designer
        namespace: $NAMESPACE
    spec:
      containers:
        - image: $IMAGE
          imagePullPolicy: $PULL_POLICY
          name: fr-designer
          #env:
          # # Use secret in real usage
          #- name: MYSQL_ROOT_PASSWORD
          # value: password
          ports:
            - containerPort: 80
              name: fr-designer
          volumeMounts:
            - name: config-volume
              mountPath: /app/appsettings.Production.json
              subPath: appsettings.Production.json
            - name: secret-volume
              mountPath: /etc/cert
      imagePullSecrets:
        - name: $IMAGE_REGISTRY_SECRET_NAME
      volumes:
        - name: config-volume
          configMap:
            name: fast-report-config
            items:
              - key: appsettings.Production.json
                path: appsettings.Production.json
        - name: secret-volume
          secret:
            secretName: $SECRET_VOLUME_NAME
EOF
```

## Сервис шрифтов

```
#!/bin/sh
```

```
IMAGE="${IMAGE_STORAGE}/moiotchety-corporate-server-fonts:${IMAGE_TAG}"
```

```
cat <<EOF | kubectl apply -n $NAMESPACE -f -
```

```
apiVersion: v1
```

```
kind: Service
```

```
metadata:
```

```
  name: fr-fonts
```

```
  namespace: $NAMESPACE
```

```
spec:
```

```
  type: ClusterIP
```

```
  ports:
```

```
  - port: 80
```

```
  selector:
```

```
    app: fr-fonts
```

```
---
```

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
```

```
kind: Deployment
```

```
metadata:
```

```
  name: fr-fonts
```

```
  namespace: $NAMESPACE
```

```
spec:
```

```
  replicas: 1
```

```
  selector:
```

```
    matchLabels:
```

```
      app: fr-fonts
```

```
  strategy:
```

```
    type: Recreate
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        app: fr-fonts
```

```
        namespace: $NAMESPACE
```

```
    spec:
```

```
      containers:
```

```
      - image: $IMAGE
```

```
        resources:
```

```
          limits:
```

```
            memory: 200Mi
```

```
          requests:
```

```
            memory: 200Mi
```

```
        imagePullPolicy: $PULL_POLICY
```

```
        name: fr-fonts
```

```
        env:
```

```
        - name: Serilog__Using__0
```

```
          value: FastReport.Cloud.MvcExtentions
```

```
        ports:
```

```
        - containerPort: 80
```

```
          name: fr-fonts
```

```
        volumeMounts:
```

```
        - name: config-volume
```

```
          mountPath: /app/appsettings.Production.json
```

```
          subPath: appsettings.Production.json
```

```
        imagePullSecrets:
```

```
        - name: $IMAGE_REGISTRY_SECRET_NAME
```

```
        volumes:
```

```
        - name: config-volume
```

```
          configMap:
```

```
            name: fast-report-config
```

```
            items:
```

```
            - key: appsettings.Production.json
```

```
              path: appsettings.Production.json
```

```
EOF
```

## Сервис облачного приложения

```
#!/bin/sh

IMAGE="{IMAGE_STORAGE}/moiotchety-corporate-server-app:{IMAGE_TAG}"

cat <<EOF | kubectl apply -n $NAMESPACE -f -
apiVersion: v1
kind: Service
metadata:
  name: fr-app
  namespace: $NAMESPACE
spec:
  type: ClusterIP
  ports:
    - port: 80
  selector:
    app: fr-app
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: fr-app
  namespace: $NAMESPACE
spec:
  replicas: 1
  selector:
    matchLabels:
      app: fr-app
  strategy:
    type: Recreate
  template:
    metadata:
      namespace: $NAMESPACE
    labels:
      app: fr-app
    spec:
      containers:
        - image: $IMAGE
          imagePullPolicy: $PULL_POLICY
          name: fr-app
          #env:
          # # Use secret in real usage
          #- name: MYSQL_ROOT_PASSWORD
          # value: password
          ports:
            - containerPort: 80
              name: fr-app
          imagePullSecrets:
            - name: $IMAGE_REGISTRY_SECRET_NAME
EOF
```

## Сервис построения отчётов

```
#!/bin/sh

IMAGE="{IMAGE_STORAGE}/moiotchety-corporate-server-workercore:{IMAGE_TAG}"

cat <<EOF | kubectl apply -n $NAMESPACE -f -
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: fr-workercore
  namespace: $NAMESPACE
spec:
  replicas: 4
  selector:
    matchLabels:
      app: fr-workercore
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: fr-workercore
        namespace: $NAMESPACE
    spec:
      containers:
        - image: $IMAGE
          imagePullPolicy: $PULL_POLICY
          name: fr-workercore
          volumeMounts:
            - name: config-volume
              mountPath: /app/appsettings.Production.json
              subPath: appsettings.Production.json
            - name: secret-volume
              mountPath: /etc/cert
      imagePullSecrets:
        - name: $IMAGE_REGISTRY_SECRET_NAME
      volumes:
        - name: config-volume
          configMap:
            name: fast-report-config
            items:
              - key: appsettings.Production.json
                path: appsettings.Production.json
        - name: secret-volume
          secret:
            secretName: $SECRET_VOLUME_NAME
EOF
```

После выполнения всех вышеописанных скриптов установка сервера отчётов закончена. Для того чтобы проверить успешность установки и начать работу, зайдите с помощью браузера на сервер отчётов, используя доменное имя, указанное в переменной HOST при настройке nginx-ingress и в появившемся окне введите следующие данные:

Имя пользователя:

Пароль:

Приятной работы!

# Конфигурация

Продукты: [МоиОтчеты](#) [Корпоративный Сервер](#)

Конфигурация МоиОтчеты Корпоративный Сервер осуществляется через файлы `appsettings.json`, которые лежат в директории приложения. Эти файлы по умолчанию уже имеют ряд свойств, которые можно переопределить одним из трех способов:

1. Через `appsettings.{Environment}.json`, по умолчанию переменная `Environment` имеет значение `Production`, поэтому достаточно изменить содержание файла `appsettings.Production.json`. Регистр имеет значение!

Для смены значения используйте переменные среды `ASPNETCORE_ENVIRONMENT` и `DOTNET_ENVIRONMENT`.

```
export ASPNETCORE_ENVIRONMENT=Production
export DOTNET_ENVIRONMENT=Production
```

2. Через переменные среды. Подробности описаны ниже.

Приоритезация загрузки конфига `appsettings.json` -> `appsettings.{Environment}.json` -> переменные среды. Это означает, что конфигурация будет загружена слева направо, иначе говоря, те файлы что находятся правее перезаписывают конфиг предыдущих.

## Описание формата `appsettings.json`

В каждом из разделов будет приведён пример конфигурации и описательная часть: ключ -> значение.

```
{
  "Logging":{
    {
      "Name":"ToEmail"
    }
  ]
}
```

Этот ключ можно использовать как переменную среды:

```
export Logging__0__Name=ToEmail
```

Здесь запись `Logging__0__Name` означает обращение к разделу `Logging`, элементу под индексом `0` и свойство `Name`.

## Раздел `Kestrel`

Позволяет конфигурировать http сервер для прослушивания определённого порта или использования сертификата. Пример конфига:

```

{
  "Kestrel":{
    "Endpoints":{
      "Http":{
        "Url":"http://localhost:5000"
      },
      "Https":{
        "Url":"https://localhost:5001",
        "Certificate":{
          "Path":"<путь к файлу .pfx>",
          "Password":"<пароль к сертификату>"
        }
      }
    }
  }
}

```

КЛЮЧ	ТИП	ОПИСАНИЕ
Kestrel__Endpoints__Http__Url	string (uri)	Точка доступа для прослушивания http.
Kestrel__Endpoints__Https__Url	string (uri)	Точка доступа для прослушивания https.
Kestrel__Endpoints__Https__Certificate__Path	string (local path)	Путь к файлу <code>.pfx</code> сертификата для https трафика.
Kestrel__Endpoints__Https__Certificate__Password	string	Пароль для доступа к <code>.pfx</code> файлу.

## Раздел `Auth`

Позволяет конфигурировать процесс аутентификации и авторизации.

```

{
  "Auth":{
    "ClientId":"<клиент OpenID>",
    "Scopes":"OpenID и другие области видимости",
    "Authority":"https://id.быстрыеотчеты.рф",
    "Audience":"https://облако.моиотчеты.рф",
    "Secret":"<секрет для клиента OpenID>",
    "UseApiKeys":true,
    "UseOpenId":true,
    "UseLocal":false,
    "AllowLocalSignUp":false,
    "RsaXml":"<XML-кодированный объект>",
    "UserInfoEndpoint":"https://example.com/userinfo",
    "TokenEndpoint":"https://example.com/token",
    "MetadataEndpoint":"https://example.com/.well-known/openid-configuration"
  }
}

```

КЛЮЧ	ТИП	ОПИСАНИЕ
Auth__ClientId	string	Уникальный идентификатор на сервере аутентификации по протоколу <code>openid</code> .
Auth__Scopes	string	Список областей клиента по протоколу <code>openid</code> .
Auth__Authority	string (uri)	Точка доступа <code>Authority</code> у которой будет запрашиваться токен доступа по протоколу <code>openid</code> .

КЛЮЧ	ТИП	ОПИСАНИЕ
Auth__Audience	string (uri)	Точка доступа <code>Audience</code> для которой будет запрашиваться токен доступа по протоколу <code>openid</code> .
Auth__Secret	string	Секретный токен для подтверждения авторизации по протоколу <code>openid</code> .
Auth__UseApiKeys	boolean	Включает или отключает возможность авторизации через ключи доступа.
Auth__UseOpenId	boolean	Включает или отключает возможность авторизации по протоколу <code>openid</code> , обратите внимание, клиент должен быть настроен для авторизации через <a href="#">code flow</a> .
Auth__UseLocal	boolean	Включает или отключает возможность авторизации по почте и паролю. Данные пользователей в этом случае хранятся локально в базе сервера.
Auth__AllowLocalSignUp	boolean	Включает или отключает регистрацию со стороны пользователя. Если отключено, то регистрация доступна только администраторам.
Auth__RsaXml	string (xml)	RSA сериализованный в XML. Например, <pre>&lt;RSAKeyValue&gt;&lt;Modulus&gt;{base64}&lt;/Modulus&gt;&lt;Exponent&gt;{base64}&lt;/Exponent&gt;&lt;P&gt;{base64}&lt;/P&gt;&lt;Q&gt;{base64}&lt;/Q&gt;&lt;DP&gt;{base64}&lt;/DP&gt;&lt;DQ&gt;{base64}&lt;/DQ&gt;&lt;InverseQ&gt;{base64}&lt;/InverseQ&gt;&lt;D&gt;{base64}&lt;/D&gt;&lt;/RSAKeyValue&gt;</pre>
Auth__UserInfoEndpoint	string (uri)	Точка доступа, через которую будут запрашиваться данные пользователя по протоколу <code>openid</code> .
Auth__TokenEndpoint	string (uri)	Точка доступа, через которую будет запрашиваться токен пользователя по протоколу <code>openid</code> .
Auth__MetadataEndpoint	string (uri)	Точка доступа <code>well-known</code> с метаданными для <code>openid</code> .

## Раздел `MainConfig`

Позволяет конфигурировать основную конфигурацию МоиОтчеты Корпоративный Сервер.

```
{
  "MainConfig":{
    "Database":{
      "ConnectionString":"<строка подключения>",
      "DatabaseName":"ReportstoreDb",
      "ExportsCollectionName":"Exports",
      "ReportsCollectionName":"Reports",
      "TemplatesCollectionName":"Templates",
      "TemplateFoldersCollectionName":"TemplateFolders",
      "ReportFoldersCollectionName":"ReportFolders",
      "ExportFoldersCollectionName":"ExportFolders",
      "UsersCollectionName":"Users",
      "SubscriptionPlansCollectionName":"SubscriptionPlans",
      "SubscriptionsCollectionName":"Subscriptions",
      "SubscriptionInvitesCollectionName":"SubscriptionInvites",
      "GroupsCollectionName":"Groups",
      "DataSourceCollectionName":"DataSources",
      "GridFSFilesCollectionName":"fs.files",
      "MigrationsCollectionName":"Migrations",
      "CurrentTasksCollectionName":"CurrentTasks",
      "TasksCollectionName":"Tasks",
      "AuditCollectionName":"Audit",
    }
  }
}
```

```

"ContactsCollectionName": "Contacts",
"ContactGroupsCollectionName": "ContactGroups",
"FontsCollectionName": "Fonts",
"UserFontsCollectionName": "UserFonts"
};
"Server": {
  "Title": "<Заголовок в приложении>",
  "Copyright": "<Краткое описание авторского права>",
  "LogoLink": "<путь к логотипу, отображаемому в панели пользователя>",
  "FaviconLink": "<путь к иконке в формате ico>",
  "LastSlaUpdate": "2022-11-28",
  "CorporateServerMode": true,
  "SlaLink": "<Ссылка на пользовательское соглашение>",
  "FirstStepsVideoLink": "https://rutube.ru/play/embed/f64e80d2fa8e2d73b30c47483fa2b59e",
  "AboutLink": "https://быстрыеотчеты.рф/products/corporate-server",
  "HomePageLink": "быстрыеотчеты.рф",
  "AuthServerName": "ID Быстрые отчеты",
  "UsersPerWorkSpace": null,
  "DataSourcesPerWorkSpace": null,
  "GroupsPerWorkSpace": null
};
"Designer" : {
  "IntellisenseEnabled" : false,
  "DataSources" : {
    "Edit" : true,
    "Create" : true,
    "Remove" : true
  },
  "CodeRestricted" : false
};
"Pagination":{
  "MaxEntries":120
};
"Rabbit":{
  "Host":"my-rabbit-server.com",
  "Port":5672,
  "UserName":"<имя пользователя>",
  "Password":"<пароль пользователя>",
  "QueueName":"ReportProcessQueue",
  "DirectExchangeName":"DirectEx",
  "AlternateExchangeName":"AExchange",
  "UnroutedQueueName":"Default",
  "Capacity":1
};
"SecurityAdvisor":{
  "RestrictUnsafe":true,
  "RestrictUnmanaged":true,
  "RestrictExtern":true,
  "RestrictAsync":true,
  "RestrictTypeOf":true,
  "Whitelist":[
    "^\\w+:\\s+FastReport.*$",
    "^\\w+\\.\\. *System\\.\\.Math.*$",
    "^\\w+\\.\\. *System\\.\\.DateTime.*$",
    "^\\w+\\.\\. *System\\.\\.Environment.*$"
  ],
  "Blacklist":[
    "^Method\\.\\. *\\.\\.GetType\\(\\)\\)$",
    "^\\w+\\.\\. *System\\.\\.IO.*$",
    "^\\w+\\.\\. *FastReport\\.\\.Utils\\.\\.Config.*$",
    "^\\w+\\.\\. *System\\.\\.Environment.*$",
    "^\\w+\\.\\. *System\\.\\.Diagnostics.*$",
    "^\\w+\\.\\. *System\\.\\.Reflection.*$",
    "^\\w+\\.\\. *System\\.\\.Net.*$",
    "^\\w+\\.\\. *System\\.\\.Threading.*$"
  ]
};

```

```

},
"License": "",
"SmtпServer": {
  "EnableSsl": true,
  "Server": "smtp.s.com",
  "Port": 587,
  "Username": "--",
  "From": "--",
  "Password": "--"
},
"Tasks": {
  "Attempts": 3
},
"Frontend": {
  "Mixins": {
    "Head": "",
    "Body": ""
  }
},
"Rfc2898CryptorSettings": {
  "Salt": "",
  "Password": "",
  "IV": ""
},
"DataSourcesConfig": {
  "XmlConfig": { "CommandTimeout": 30 },
  "JsonConfig": { "CommandTimeout": 30 },
  "CsvConfig": { "CommandTimeout": 30 },
  "MySQLConfig": { "CommandTimeout": 30 },
  "PostgreSqlConfig": { "CommandTimeout": 30 },
  "MsSqlConfig": { "CommandTimeout": 30 },
  "OracleDbConfig": { "CommandTimeout": 30 },
  "FirebirdConfig": { "CommandTimeout": 30 },
  "MongoDbConfig": { "CommandTimeout": 30 },
  "ClickHouseConfig": { "CommandTimeout": 30 }
},
"FontServerAddress": "http://fr-fonts.fr-cloud:80"
}
}

```

КЛЮЧ	ТИП	ОПИСАНИЕ
MainConfig__Database__ConnectionString	string	Строка для подключения к базе данных MongoDB (FerretDB).
MainConfig__Database__DatabaseName	string	Название базы данных сервера MongoDB (FerretDB).
MainConfig__Database__ExportsCollectionName	string	Название коллекции для хранения списка экспортов.
MainConfig__Database__ReportsCollectionName	string	Название коллекции для хранения списка отчётов.
MainConfig__Database__TemplatesCollectionName	string	Название коллекции для хранения списка шаблонов.
MainConfig__Database__TemplateFoldersCollectionName	string	Название коллекции для хранения древовидной структуры шаблонов.

КЛЮЧ	ТИП	ОПИСАНИЕ
MainConfig__Database__ReportFoldersCollectionName	string	Название коллекции для хранения древовидной структуры отчётов.
MainConfig__Database__ExportFoldersCollectionName	string	Название коллекции для хранения древовидной структуры экспортов.
MainConfig__Database__UsersCollectionName	string	Название коллекции для хранения списка пользователей.
MainConfig__Database__SubscriptionPlansCollectionName	string	Название коллекции для хранения списка планов подписки.
MainConfig__Database__SubscriptionsCollectionName	string	Название коллекции для хранения списка подписок.
MainConfig__Database__SubscriptionInvitesCollectionName	string	Название коллекции для хранения списка приглашений.
MainConfig__Database__GroupsCollectionName	string	Название коллекции для хранения списка групп.
MainConfig__Database__DataSourceCollectionName	string	Название коллекции для хранения списка источников данных.
MainConfig__Database__GridFSFilesCollectionName	string	Название коллекции для хранения файлов.
MainConfig__Database__MigrationsCollectionName	string	Название коллекции для хранения списка применённых миграций.
MainConfig__Database__CurrentTasksCollectionName	string	Название коллекции для хранения задач. Вспомогательная коллекция для текущих задач.
MainConfig__Database__TasksCollectionName	string	Название коллекции для хранения задач.
MainConfig__Database__AuditCollectionName	string	Название коллекции для хранения аудитов.
MainConfig__Database__ContactsCollectionName	string	Название коллекции для хранения контактов.
MainConfig__Database__ContactGroupsCollectionName	string	Название коллекции для хранения групп контактов.
MainConfig__Database__FontsCollectionName	string	Название коллекции для хранения шрифтов.
MainConfig__Database__UserFontsCollectionName	string	Название коллекции для хранения пользовательских шрифтов.
MainConfig__Server__Title	string	Имя для сервера, которое будет использовать пользовательское приложение в заголовке страницы.

КЛЮЧ	ТИП	ОПИСАНИЕ
MainConfig__Server__Copyright	string	Текст для информации об авторском праве (или любой другой информации, которая будет отображена внизу страницы пользовательского приложения).
MainConfig__Server__LogoLink	string	Ссылка на файл с логотипом.
MainConfig__Server__FaviconLink	string	Ссылка на файл с favicon.
MainConfig__Server__LastSLAUpdate	string	Дата последнего изменения пользовательского соглашения, если изменить эту дату на более новую, при открытии пользовательского приложения откроется диалог, в котором будет ссылка на обновлённые условия (из поля <code>SlaLink</code> ).
MainConfig__Server__CorporateServerMode	boolean	Это поле изменяет поведение части API, например, если указано false, то в админской панели нельзя будет удалять рабочие пространства. Также если оно активно, то включается показ Swagger-документации панели администратора в Swagger UI.
MainConfig__Server__FirstStepsVideoLink	string	Ссылка на видео руководство к корпоративному серверу, которое появится при первом входе в пользовательское приложение или при нажатии на <code>?</code> внизу страницы.
MainConfig__Server__AboutLink	string	Ссылка на информацию о продукте.
MainConfig__Server__HomePageLink	string	Ссылка на домашнюю страницу сайта продукта.
MainConfig__Server__AuthServerName	string	Имя для сервера авторизации, указанного в разделе <code>AuthConfig</code> , используемое в пользовательском приложении.
MainConfig__Server__UsersPerWorkSpace	integer?	Максимальное количество пользователей в рабочем пространстве, задаваемое администратором.
MainConfig__Server__DataSourcesPerWorkSpace	integer?	Максимальное количество источников данных в рабочем пространстве, задаваемое администратором.
MainConfig__Server__GroupsPerWorkSpace	integer?	Максимальное количество групп в рабочем пространстве, задаваемое администратором.
MainConfig__Rabbit__Host	string	Адрес хоста для доступа к RabbitMQ.
MainConfig__Rabbit__Port	string	Порт для доступа к RabbitMQ.

КЛЮЧ	ТИП	ОПИСАНИЕ
MainConfig__Rabbit__UserName	string	Имя пользователя для доступа к RabbitMQ.
MainConfig__Rabbit__Password	string	Пароль для доступа к RabbitMQ.
MainConfig__Rabbit__QueueName	string	Название очереди, которая используется для базового имени конкретных очередей подписок пользователей.
MainConfig__Rabbit__DirectExchangeName	string	Название обменника RabbitMQ, который используется для направления сообщений на очереди подписок пользователей.
MainConfig__Rabbit__AlternateExchangeName	string	Название обменника RabbitMQ, который будет использоваться по умолчанию, когда для пользователя ещё не создана подписка.
MainConfig__Rabbit__UnroutedQueueName	string	Название очереди RabbitMQ, которая будет обрабатываться по умолчанию, когда для пользователя ещё не создана подписка.
MainConfig__Rabbit__Capacity	1	Не меняйте это значение! Задаёт максимальное количество параллельных построений отчётов на одном рабочем строителе.
MainConfig__SecurityAdvisor__RestrictUnsafe	string	Включает или отключает ключевое слово <code>unsafe</code> в скрипте.
MainConfig__SecurityAdvisor__RestrictUnmanaged	string	Включает или отключает ключевое слово <code>unmanaged</code> в скрипте.
MainConfig__SecurityAdvisor__RestrictExtern	string (uri)	Включает или отключает ключевое слово <code>extern</code> в скрипте.
MainConfig__SecurityAdvisor__RestrictAsync	string (uri)	Включает или отключает ключевое слово <code>async</code> в скрипте.
MainConfig__SecurityAdvisor__RestrictTypeOf	string	Включает или отключает ключевое слово <code>typeof</code> в скрипте.
MainConfig__SecurityAdvisor__Whitelist__0	boolean	Задаёт список API, которые можно использовать без предупреждений в скрипте отчёта. Число указывает порядковый номер в списке.
MainConfig__SecurityAdvisor__Blacklist__0	boolean	Задаёт список API, которые нельзя использовать в скрипте отчёта. Число указывает порядковый номер в списке.
MainConfig__License	string	Лицензионный ключ. Предоставляется вместе с продуктом.
MainConfig__SmtpServer__Server	string	Адрес почтового сервера

КЛЮЧ	ТИП	ОПИСАНИЕ
MainConfig__SmtpServer__Port	integer	Порт SMTP сервера
MainConfig__SmtpServer__Username	string	Имя пользователя SMTP сервера
MainConfig__SmtpServer__Password	string	Пароль пользователя SMTP сервера
MainConfig__SmtpServer__From	string	Почтовый адрес отправителя (отображается в письме)
MainConfig__Tasks__Attempts	integer	Количество попыток запуска задачи воркером
MainConfig__Frontend__Mixins__Head	string	Примесь, встраиваемая в header пользовательской панели (например, код аналитики)
MainConfig__Frontend__Mixins__Body	string	Примесь, встраиваемая в body пользовательской панели (например, код аналитики)
MainConfig__Rfc2898CryptorSettings__Salt	string	Соль криптографического алгоритма (используется для шифрования паролей)
MainConfig__Rfc2898CryptorSettings__Password	string	Пароль криптографического алгоритма
MainConfig__Rfc2898CryptorSettings__IV	string	Вектор криптографического алгоритма
MainConfig__InvariantLocale	string	Постоянная локализация. Работает независимо от языка браузера
MainConfig__DataSourcesConfig__XmlConfig__CommandTimeout	integer	Время ожидания ответа от источника данных XML в секундах. Если не указать значение, будет использоваться стандартный таймаут для этого источника данных.
MainConfig__DataSourcesConfig__JsonConfig__CommandTimeout	integer	Время ожидания ответа от источника данных JSON в секундах. Если не указать значение, будет использоваться стандартный таймаут для этого источника данных.
MainConfig__DataSourcesConfig__CsvConfig__CommandTimeout	integer	Время ожидания ответа от источника данных CSV в секундах. Если не указать значение, будет использоваться стандартный таймаут для этого источника данных.
MainConfig__DataSourcesConfig__MySqlConfig__CommandTimeout	integer	Время ожидания ответа от источника данных MySQL в секундах. Если не указать значение, будет использоваться стандартный таймаут для этого источника данных.

КЛЮЧ	ТИП	ОПИСАНИЕ
MainConfig__DataSourcesConfig__PostgreSqlConfig__CommandTimeout	integer	Время ожидания ответа от источника данных PostgreSQL в секундах. Если не указать значение, будет использоваться стандартный таймаут для этого источника данных.
MainConfig__DataSourcesConfig__MsSqlConfig__CommandTimeout	integer	Время ожидания ответа от источника данных MS SQL в секундах. Если не указать значение, будет использоваться стандартный таймаут для этого источника данных.
MainConfig__DataSourcesConfig__OracleDbConfig__CommandTimeout	integer	Время ожидания ответа от источника данных Oracle DB в секундах. Если не указать значение, будет использоваться стандартный таймаут для этого источника данных.
MainConfig__DataSourcesConfig__FirebirdConfig__CommandTimeout	integer	Время ожидания ответа от источника данных Firebird в секундах. Если не указать значение, будет использоваться стандартный таймаут для этого источника данных.
MainConfig__DataSourcesConfig__MongoDbConfig__CommandTimeout	integer	Время ожидания ответа от источника данных Mongo DB в секундах. Если не указать значение, будет использоваться стандартный таймаут для этого источника данных.
MainConfig__DataSourcesConfig__ClickHouseConfig__CommandTimeout	integer	Время ожидания ответа от источника данных ClickHouse в секундах. Если не указать значение, будет использоваться стандартный таймаут для этого источника данных.
FontServerAddress	string (uri)	Абсолютная ссылка для доступа к серверу шрифтов.

## Раздел Gateway

Позволяет конфигурировать шлюз доступа МоиОтчеты Корпоративный Сервер.

```
{
  "Gateway":{
    "WhiteListForDisabled":{
      "<any_claim_name>":"<claim_value>"
    },
    "IsDisabled":false,
    "ExcludePaths":[
      "/account",
      "/disabled"
    ],
    "IsSignInRequired":false
  }
}
```

КЛЮЧ	ТИП	ОПИСАНИЕ
Gateway__WhiteListForDisabled__<any_claim_name>	string	Список утверждений пользователя в токене для доступа к выключенному МоиОтчеты Облако.
Gateway__IsDisabled	boolean	Включает или отключает доступ к МоиОтчеты Облако.
Gateway__ExcludePaths__0	string	Список путей, к которым можно получить доступ даже при выключенном доступе к МоиОтчеты Облако. Число указывает порядковый номер в списке.
Gateway__IsSignInRequired	boolean	Включает или отключает необходимость входа для пользователя для доступа к МоиОтчеты Облако.

## Раздел `Services`

Позволяет конфигурировать список сервисов для маршрутизации шлюза.

```
{
  "Services":{
    "HealthCheckInterval":30,
    "Items":{
      "<name>":{
        "Urls":[
          "http://localhost:5555"
        ],
        "Scheme":"http",
        "Port":5555,
        "K8sServiceName":"fr-rp",
        "HostType":"WebApp",
        "PathBase":"/api/rp/swagger",
        "Namespace":"fr-cloud",
        "Type":"K8s",
        "PingPath":"/api/rp/v1/healthcheck",
        "IsSignInRequired":true,
        "Priority":10,
        "PingResponseCode":200,
        "LoadBalanceMode":"Random",
        "HealthCheckAttemptsNumber":3,
        "WhiteListClaims":{
          "<claim_name>":"<claim_value>"
        }
      }
    }
  }
}
```

КЛЮЧ	ТИП	ОПИСАНИЕ
Services__HealthCheckInterval	integer	Интервал для проверки работоспособности сервисов, задаётся в секундах.
Services__Items__<name>__Urls__0	string (url)	Список url адресов для доступа к статичным сервисам. Число указывает порядковый номер в списке.
Services__Items__<name>__Scheme	string	Схема доступа к сервису: http или https.

КЛЮЧ	ТИП	ОПИСАНИЕ
<code>Services__Items__&lt;name&gt;__Port</code>	integer	Порт доступа к сервису.
<code>Services__Items__&lt;name&gt;__K8sServiceName</code>	string	Название сервиса в Kubernetes.
<code>Services__Items__&lt;name&gt;__HostType</code>	string	Тип сервиса для обработки перенаправления шлюзом, может принимать значения: <code>WebApp</code> , <code>API</code> , <code>External</code> , <code>Websocket</code> .
<code>Services__Items__&lt;name&gt;__PathBase</code>	string	Базовый путь для сервиса.
<code>Services__Items__&lt;name&gt;__Namespace</code>	string	Пространство имён сервиса в Kubernetes.
<code>Services__Items__&lt;name&gt;__Type</code>	string	Тип сервиса для обработки доступа шлюзом; может принимать значения: <code>Static</code> , <code>K8s</code> .
<code>Services__Items__&lt;name&gt;__PingPath</code>	string	Часть строки запроса для получения информации о состоянии сервиса.
<code>Services__Items__&lt;name&gt;__IsSignInRequired</code>	string	Указывает необходимость авторизации перед получением доступа к сервису пользователям.
<code>Services__Items__&lt;name&gt;__Priority</code>	number	Приоритет сервиса перед другими; значение меньше значит сильнее.
<code>Services__Items__&lt;name&gt;__PingResponseCode</code>	integer	Статус код ответа, который будет ожидаться от healthcheck.
<code>Services__Items__&lt;name&gt;__LoadBalanceMode</code>	string	Указывает тип балансировки нагрузки для этого сервиса; может принимать одно из значений: <code>Random</code> , <code>AverageMetric</code> .
<code>Services__Items__&lt;name&gt;__HealthCheckAttemptsNumber</code>	integer	Количество попыток проверить состояния сервиса.
<code>Services__Items__&lt;name&gt;__WhiteListClaims__&lt;claim_name&gt;</code>	string	Указывает утверждение у пользователя для получения доступа к сервису.

Вместо `<name>` следует использовать имя сервиса; список сервисов можно посмотреть в файле `appsettings.json`.

## Раздел `Constants`

Позволяет устанавливать ограничения на размер тела запроса в приложении.

```
{
  "Constants": {
    "LimitsMaxRequestBodySize": 2097152000
  }
}
```

КЛЮЧ	ТИП	ОПИСАНИЕ
<code>Constants__LimitsMaxRequestBodySize</code>	long	Максимальный размер тела запроса. При превышении данного ограничения запрос будет отклонен.

## Раздел `Designer`

Позволяет настроить некоторые компоненты онлайн дизайнера.

```
"Designer": {
  "IntellisenseEnabled" : true,
  "DataSources" : {
    "Edit" : true,
    "Create" : true,
    "Remove" : true
  },
  "CodeRestricted" : false
},
```

| `Designer__IntellisenseEnabled` | boolean | Это поле позволяет включить или выключить подсказки при написании кода внутри шаблона в онлайн дизайнера. | | `Designer__CodeRestricted` | boolean | Это поле позволяет включить или выключить вкладку для редактирования кода внутри шаблона в онлайн дизайнера. | | `Designer__DataSources__Create` | boolean | Включает или отключает возможность подключать новые источники данных к шаблонам через онлайн дизайнер. | | `Designer__DataSources__Edit` | boolean | Включает или отключает возможность редактировать источники данных в шаблонах через онлайн дизайнер. | | `Designer__DataSources__Remove` | boolean | Включает или отключает возможность отключать источники данных от шаблонов через онлайн дизайнер. |

Этот раздел конфигурации также содержит поля "BackendUrl и InternalHeaders", которые описаны ниже

## Дополнительный раздел `Migrations` для `MainConfig`

Позволяет настраивать конфигурацию применения миграций.

```
{
  "MainConfig": {
    "Migrations": {
      "DataBaseCheckInterval": 1,
      "LockDoubleCheckInterval": 5,
      "MigrationTimeOut": 60
    }
  }
}
```

КЛЮЧ	ТИП	ОПИСАНИЕ
<code>MainConfig__Migrations__DataBaseCheckInterval</code>	double	Интервал (в секундах) проверки наличия доступа к базе данных перед запуском миграций. По умолчанию: 1.
<code>MainConfig__Migrations__LockDoubleCheckInterval</code>	double	Интервал (в секундах) для повторной проверки блокировки миграции. Может потребоваться увеличить, если используется репликасет с задержкой. По умолчанию: 5.
<code>MainConfig__Migrations__MigrationTimeOut</code>	double	Время ожидания выполнения миграции (в секундах), после которого происходит таймаут. По умолчанию: 60.

## Дополнительный раздел `InternalHeaders` для `MainConfig`

Раздел `InternalHeaders` используется для настройки внутренней авторизации между сервисами системы. Каждый сервис должен использовать уникальный ключ для идентификации при внутреннем взаимодействии. Эти ключи **должны быть сгенерированы пользователем самостоятельно** — использование значений из примера недопустимо в целях безопасности.

```

{
  "Designer": {
    "BackendUrl": "http://fr-backend.fr-cloud:80",
    "InternalKey": "fc51a5d6-95c3-4679-8c03-c9ac6536bf5d"
  },
  "Fonts": {
    "BackendUrl": "http://fr-backend.fr-cloud:80",
    "InternalKey": "22788120-f7cd-43f2-938b-327d6c3ceed4"
  },
  "Scheduler": {
    "BackendUrl": "http://fr-backend.fr-cloud:80",
    "InternalKey": "9da6cb50-8796-42cc-b3a7-53c1761f0ac9"
  },
  "WorkerCore": {
    "BackendUrl": "http://fr-backend.fr-cloud:80",
    "InternalKey": "ee444483-e0f1-4ec2-9b4e-5cbacdfaa07f"
  },
  "Gateway": {
    "BackendUrl": "http://fr-backend.fr-cloud:80",
    "InternalKey": "4632ab6a-3e20-4430-9f13-2fe1851809db"
  },
  "MainConfig": {
    "InternalHeaders": {
      "ee444483-e0f1-4ec2-9b4e-5cbacdfaa07f": "000000000000000000000001",
      "fc51a5d6-95c3-4679-8c03-c9ac6536bf5d": "000000000000000000000002",
      "4632ab6a-3e20-4430-9f13-2fe1851809db": "000000000000000000000003",
      "9da6cb50-8796-42cc-b3a7-53c1761f0ac9": "000000000000000000000004",
      "22788120-f7cd-43f2-938b-327d6c3ceed4": "000000000000000000000005"
    }
  }
}

```

КЛЮЧ	ТИП	ОПИСАНИЕ
Designer__BackendUrl	string	URL бэкенда сервиса Designer.
Designer__InternalKey	string	Уникальный ключ для внутренней авторизации Designer.
Fonts__BackendUrl	string	URL бэкенда сервиса Fonts.
Fonts__InternalKey	string	Уникальный ключ для внутренней авторизации Fonts.
Scheduler__BackendUrl	string	URL бэкенда сервиса Scheduler.
Scheduler__InternalKey	string	Уникальный ключ для внутренней авторизации Scheduler.
WorkerCore__BackendUrl	string	URL бэкенда сервиса WorkerCore.
WorkerCore__InternalKey	string	Уникальный ключ для внутренней авторизации WorkerCore.
Gateway__BackendUrl	string	URL бэкенда сервиса Gateway.
Gateway__InternalKey	string	Уникальный ключ для внутренней авторизации Gateway.
MainConfig__InternalHeaders__<ключ>	string	Сопоставление InternalKey — внутреннего идентификатора сервиса.

**Важно:** ключи InternalKey и соответствующие им значения в InternalHeaders должны быть уникальными и

### **Рекомендации по генерации ключей**

- Ключи могут быть представлены в формате GUID или произвольной сложной строкой.
- Рекомендуется использовать криптографически безопасные генераторы случайных строк.
- Ключи должны быть достаточно длинными и уникальными, чтобы исключить возможность коллизий или подбора.

# Ограничения

Продукты: [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

В демо-версии действуют следующие ограничения:

- Можно экспортировать не более пяти страниц.
- На страницах экспортируемых документов отображается водяной знак «Демо версия».
- Бесконечные страницы отключены.

# Удаление МоиОтчеты Корпоративный Сервер из кластера

Продукты: [МоиОтчеты Корпоративный Сервер](#)

## 1. Удаление компонентов сервера отчётов из кластера:

```
kubectl delete -n fr-corporate-server deployment fr-designer
kubectl delete -n fr-corporate-server deployment fr-scheduler
kubectl delete -n fr-corporate-server deployment fr-workercore
kubectl delete -n fr-corporate-server deployment fr-gateway
kubectl delete -n fr-corporate-server deployment fr-app
kubectl delete -n fr-corporate-server deployment fr-backend
kubectl delete -n fr-corporate-server deployment fr-s-preview
kubectl delete -n fr-corporate-server deployment fr-fonts
kubectl delete -n fr-corporate-server deployment fr-homer
kubectl delete -n fr-corporate-server secret corporate-volume-secret
```

## 2. Удаление MongoDB или FerretDB.

Удаление MongoDB (FerretDB) приведёт к удалению пользователей, групп, шаблонов отчётов, подготовленных отчётов и экспортированных отчётов.

```
kubectl delete -n mongo deployment fr-mongo-mongodb
```

или

```
kubectl delete -n ferret deployment fr-ferret
```

Обратите внимание - в зависимости от настроек, вышеприведённая команда может не удалять дисковое пространство, арендованное у провайдеров облачных сервисов. Если вы арендуете кластер у облачного провайдера, воспользуйтесь web-интерфейсом провайдера для удаления Persistent Volumes сервера отчётов - mongo-storage.

## 3. Удаление остальных компонентов кластера необязательно, а в случае, если сервер был использован в рабочем кластере совместно со сторонними сервисами, то удаление этих компонентов может повредить инфраструктуру. Например, RabbitMQ может быть использован сторонними сервисами кластера.

## 4. Для полного удаления кластера и его компонентов используйте команду:

```
kubeadm reset
```

# Разворачивание Корпоративного Сервера или Публикатора без Kubernetes (docker-compose) - MongoDB

Продукты: [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

## Общее описание

Данная инструкция представляет собой **пример** развёртывания сервера отчётов (Корпоративный Сервер / Публикатор) с использованием **MongoDB** в качестве базы данных и **RabbitMQ** в качестве брокера сообщений.

Все конфигурационные файлы в данном каталоге являются **примерами** и предназначены для локального тестирования и ознакомления. Для рабочего контура необходимо адаптировать настройки под ваше окружение.

## Важная информация о поддержке

Компания «Быстрые отчеты» оказывает поддержку **только на продукты МоиОтчеты** (Корпоративный Сервер, Публикатор).

Мы **не несём ответственности** за работу, настройку и поддержку следующих компонентов:

- **MongoDB** - база данных
- **RabbitMQ** - брокер сообщений
- Любые другие сторонние сервисы и контейнеры, представленные в данном примере

Если у вас возникнут проблемы с базой данных или сторонними сервисами, обратитесь к их документации или соответствующим сообществам.

## Предварительные требования

Перед началом установки убедитесь, что на вашей системе установлены:

1. **Docker** (версия 20.10+ рекомендуется) - [инструкция по установке](#)
2. **Docker Compose** (плагин для Docker, версия v2+) - обычно входит в Docker Desktop. Проверить можно командой:

```
docker compose version
```

3. Достаточное количество оперативной памяти (рекомендуется **не менее 4 ГБ** свободной RAM для всех контейнеров).
4. Порт **8080** (веб-интерфейс) должен быть свободен на хост-машине.

## Содержимое каталога

ФАЙЛ	НАЗНАЧЕНИЕ
<code>docker-compose.yml</code>	Описание всех сервисов (контейнеров), их связей и настроек.
<code>.env</code>	Переменные окружения - имя и версия docker-образов продукта.

ФАЙЛ	НАЗНАЧЕНИЕ
appsettings.Production.json	Конфигурация приложения: подключение к БД, брокеру, внутренние ключи безопасности, адреса сервисов.
mongo-init.js	Скрипт инициализации базы данных MongoDB (создание пользователей и настройка прав доступа).

## Переменные окружения (.env)

Файл `.env` определяет параметры, которые подставляются в `docker-compose.yml` через синтаксис `${ПЕРЕМЕННАЯ}`.

```
FRC_SOURCE=xn--80akiaokt3b4b.xn--90aia9aifhdb2cxbdg.xn--p1ai/repository/docker-registry/moiotchety-corporate-server
FRC_VERSION=2026.2.2
```

### Пояснения:

- `FRC_SOURCE` - адрес docker-регистра (registry), откуда будут скачиваться docker-образы продукта МоиОтчеты.
- `FRC_VERSION` - версия продукта. Убедитесь, что указанная версия существует в реестре и у вас есть доступ к ней.

Важно: Не изменяйте `FRC_SOURCE`, если вы не уверены в корректности адреса реестра. Убедитесь, что ваш docker-демон имеет доступ к этому реестру.

## Конфигурация приложения (appsettings.Production.json)

Этот файл содержит основные настройки приложения. Рассмотрим ключевые секции:

```
{
  "Auth": {
    "UseOpenId": false,
    "UseLocal": true
  },
  "MainConfig": {
    "InternalHeaders": {
      "S56nHMSzjQzXYx5KJJsU3cjU": "000000000000000000000001",
      "x2aHtuSsxFeYqE8xPTaxAnbH": "00000000000000000000000002",
      "QNpq2nyzvDNSCLtVhMJ9e8m": "0000000000000000000000000003",
      "9MXgeFwLNjeUrJvw7N2aQv9F": "0000000000000000000000000004",
      "227881200f7cd43f238b327d": "00000000000000000000000005"
    },
    "Frontend": {
      "Mixins": {
        "Head": "",
        "Body": ""
      },
      "InvariantLocale": ""
    },
    "License": "",
    "Server": {
      "Title": "МоиОтчеты Сервисные решения",
      "CorporateServerMode": true
    },
    "Rabbit": {
      "Host": "rabbitmq",
      "Port": 5672,
      "UserName": "fastreports",
      "Password": "Qwerty!23456",
      "QueueName": "ReportProcessQueue",
      "DirectExchangeName": "DirectEx",
      "AlternateExchangeName": "AltExchange"
    }
  }
}
```

```

    "AlternateExchangeName": "AExchange",
    "UnroutedQueueName": "Default",
    "Capacity": 1
  },
  "Database": {
    "ConnectionString": "mongodb://fastreport:Qwerty!23456@mongo:27017/?
authSource=ReportStore&readPreference=primary&maxPoolSize=100&waitQueueMultiple=100",
    "DatabaseName": "ReportStore"
  }
},
"Gateway": {
  "BackendUrl": "http://fr-backend:80",
  "InternalKey": "QNpq2nyzvDNSCBLtVhMJ9e8m",
  "SignInPagePath": "/account/signin?r={0}",
  "MaxConcurrentRequests": 200,
  "RequestQueueLimit": 5000
},
"Serilog": {
  "MinimumLevel": {
    "Default": "Debug"
  }
},
"Services": {
  "Items": {
    "OnlineDesigner": {
      "Type": "Static",
      "Urls": [
        "http://fr-onlinedesigner:80"
      ]
    },
    "Backend": {
      "Type": "Static",
      "Urls": [
        "http://fr-backend:80"
      ]
    },
    "FrontendApp": {
      "Type": "Static",
      "Urls": [
        "http://fr-app:80"
      ]
    },
    "Fonts": {
      "Type": "Static",
      "Urls": [
        "http://fr-fonts:80"
      ]
    },
    "StaticPreviewApp": {
      "Type": "Static",
      "Urls": [
        "http://fr-staticpreview:80"
      ]
    },
    "HomerApp": {
      "Type": "Static",
      "Urls": [
        "http://fr-homer:80"
      ]
    }
  }
},
"Designer": {
  "BackendUrl": "http://fr-backend:80",
  "InternalKey": "x2aHtuSsxFeYqE8xPTaxAnbH"
},

```

```
"WorkerCore": {
  "BackendUrl": "http://fr-backend:80",
  "InternalKey": "S56nHMSzjQzXYx5KJJsU3cjU"
},
"Scheduler": {
  "BackendUrl": "http://fr-backend:80",
  "InternalKey": "9MXgeFwLNjeUrJvw7N2aQv9F"
},
"Fonts": {
  "BackendUrl": "http://fr-backend:80",
  "InternalKey": "227881200f7cd43f238b327d"
}
}
```

## Аутентификация

- `UseLocal: true` - используется встроенная локальная авторизация (логин/пароль).
- `UseOpenId: false` - внешний OpenID Connect-провайдер не используется. Для рабочего контура рекомендуется настроить OIDC.

## Внутренние заголовки безопасности (InternalHeaders)

```
"InternalHeaders": {
  "S56nHMSzjQzXYx5KJJsU3cjU": "00000000000000000000000000000000",
  "x2aHtuSsxFeYqE8xPTaxAnbH": "00000000000000000000000000000002",
  "QNpq2nyzvDNSCLtVhMJ9e8m": "00000000000000000000000000000003",
  "9MXgeFwLNjeUrJvw7N2aQv9F": "00000000000000000000000000000004",
  "227881200f7cd43f238b327d": "00000000000000000000000000000005"
}
```

Каждая пара ключ:значение - это уникальный идентификатор и секрет для внутренней коммуникации между сервисами продукта.

Предупреждение безопасности:

- Эти ключи **должны быть уникальными и не должны совпадать** с какими-либо другими секретами в системе.
- Они используются для авторизации внутренних запросов между микросервисами (Gateway, Backend, Designer, WorkerCore, Scheduler, Fonts).
- **Никогда** не публикуйте эти ключи в открытом доступе.
- Для рабочего контура **обязательно замените** примерные значения на ваши собственные случайные строки.

## Подключение к RabbitMQ (брокер сообщений)

```
"Rabbit": {
  "Host": "rabbitmq",
  "Port": 5672,
  "UserName": "fastreports",
  "Password": "Qwerty!23456",
  ...
}
```

- `Host` - имя сервиса RabbitMQ из `docker-compose.yml` (в данном примере - `rabbitmq`).
- `Port` - порт AMQP (по умолчанию 5672).
- `UserName` / `Password` - учётные данные для подключения к RabbitMQ.

Предупреждение безопасности: Пароль `Qwerty!23456` и логин `fastreports` указаны **только для примера**. В рабочем контуре необходимо использовать **надёжные, случайно сгенерированные учётные данные**. Не используйте этот пароль в реальных развертываниях.

## Подключение к базе данных (MongoDB)

```
"Database": {
  "ConnectionString": "mongodb://fastreport:Qwerty!23456@mongo:27017/?
authSource=ReportStore&readPreference=primary&maxPoolSize=100&waitQueueMultiple=100",
  "DatabaseName": "ReportStore"
}
```

- `ConnectionString` - адрес подключения к MongoDB. Сервис указан как `mongo` (соответствует имени в `docker-compose.yml`), порт `27017`.
- `authSource=ReportStore` - база данных, используемая для аутентификации.
- `fastreport:Qwerty!23456` - логин и пароль пользователя базы данных.
- `ReportStore` - имя целевой базы данных.

Предупреждение безопасности: Пароль базы данных `Qwerty!23456` указан **только для примера**. Для рабочего контура **обязательно замените** его на надёжный пароль. Убедитесь, что пароли в `appsettings.Production.json` совпадают с настройками контейнера MongoDB в `docker-compose.yml`.

## Gateway, Designer, WorkerCore, Scheduler, Fonts

Каждый из этих сервисов содержит поле `InternalKey`, которое должно **совпадать** с соответствующим ключом из секции `InternalHeaders`. Это обеспечивает безопасную внутреннюю коммуникацию.

## Скрипт инициализации MongoDB (`mongo-init.js`)

```
db = db.getSiblingDB('admin')

db.auth('admin', 'Qwerty!23456')

db = db.getSiblingDB('ReportStore')

db.createUser({
  user: "fastreport",
  pwd: "Qwerty!23456",
  roles: [ { role: "readWrite", db: "ReportStore" } ]
});
```

Файл `mongo-init.js` выполняется автоматически при первом запуске контейнера MongoDB. Он выполняет следующие действия:

1. Переключается на базу `admin` и выполняет аутентификацию администратора.
2. Переключается на базу `ReportStore`.
3. Создаёт пользователя `fastreport` с паролем и правами `readWrite` на базу `ReportStore`.

Примечание: Этот скрипт выполняется **только при первичном создании базы данных**. Если база данных уже существует (например, при повторном запуске с сохранённым томом данных), скрипт не будет выполнен повторно.

## Файл `docker-compose.yml`

```
services:
  mongo:
    image: mongo:5.0
    volumes:
      - ./mongo:/data/db
      - ./mongo-init.js:/docker-entrypoint-initdb.d/mongo-init.js:ro
    restart: always
    environment:
      - MONGO_INITDB_ROOT_USERNAME=admin
      - MONGO_INITDB_ROOT_PASSWORD=Qwerty!23456
```

```
- MONGO_INITDB_ROOT_PASSWORD=Qwerty!23456
- MONGO_INITDB_DATABASE=ReportStore
networks:
- fr-cs
rabbitmq:
image: bitnamilegacy/rabbitmq:3.9.27-debian-11-r9
volumes:
- ./rabbitmq/bitnami
restart: always
networks:
- fr-cs
environment:
- RABBITMQ_USERNAME=fastreports
- RABBITMQ_PASSWORD=Qwerty!23456
fr-backend:
image: ${FRC_SOURCE}-backend:debian-${FRC_VERSION}
restart: always
volumes:
- ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
- fr-cs
fr-gateway:
image: ${FRC_SOURCE}-gateway:debian-${FRC_VERSION}
ports:
- 8080:80
restart: always
volumes:
- ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
- fr-cs
fr-fonts:
image: ${FRC_SOURCE}-fonts:debian-${FRC_VERSION}
restart: always
volumes:
- ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
- fr-cs
fr-app:
image: ${FRC_SOURCE}-app:debian-${FRC_VERSION}
restart: always
volumes:
- ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
- fr-cs
fr-staticpreview:
image: ${FRC_SOURCE}-static-preview:debian-${FRC_VERSION}
restart: always
volumes:
- ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
- fr-cs
fr-onlinedesigner:
image: ${FRC_SOURCE}-designer:debian-${FRC_VERSION}
restart: always
volumes:
- ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
- fr-cs
fr-homer:
image: ${FRC_SOURCE}-homer:debian-${FRC_VERSION}
restart: always
volumes:
- ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
- fr-cs
fr-workercore:
```

```
image: ${FRC_SOURCE}-workercore:debian-${FRC_VERSION}
restart: always
volumes:
  - ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
  - fr-cs
deploy:
  mode: replicated
  replicas: 1
  endpoint_mode: vip
fr-scheduler:
image: ${FRC_SOURCE}-scheduler:debian-${FRC_VERSION}
restart: always
volumes:
  - ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
  - fr-cs
networks:
fr-cs:
  driver: bridge
```

Файл `docker-compose.yml` описывает все контейнеры, которые будут запущены.

Примечание: Все сервисы продукта (fr-\*) используют один и тот же конфигурационный файл `appsettings.Production.json`, который монтируется в контейнер как read-only том (`:ro`). Благодаря этому при обновлении образов конфигурация сохраняется.

Предупреждение безопасности:

- Образы контейнеров (`image`) берутся из переменных `.env`. Убедитесь, что вы доверяете источнику образов.
- Установленные в примере пароли (`Qwerty!23456`) используются **только для тестирования**. Замените их перед использованием в рабочем контуре.

## Установка и запуск

### Шаг 1. Создайте файлы конфигурации

В данном каталоге должны находиться файлы:

- `.env`
- `appsettings.Production.json`
- `docker-compose.yml`
- `mongo-init.js`

Отредактируйте файлы, заменив пароли на свои.

### Шаг 2. Запустите контейнеры

```
docker compose up -d
```

Данная команда создаст и запустит все контейнеры в фоновом режиме (`-d`).

### Шаг 3. Проверьте, что все контейнеры запущены

```
docker compose ps
```

Убедитесь, что все контейнеры имеют статус **running**. Первый запуск может занять несколько минут - Docker будет скачивать образы из реестра. А так же будут применены миграции. Отслеживайте логи что миграции применились у сервиса `backend`.

## Шаг 4. Откройте веб-интерфейс

Перейдите в браузере по адресу:

```
http://localhost:8080
```

## Шаг 5. Войдите в систему

Используйте учётные данные по умолчанию (**только для тестирования!**):

- **Логин:** `admin@example.com`
- **Пароль:** `admin`

## Шаг 6. Панель администрирования

Для доступа к панели администрирования перейдите по ссылке:

```
http://localhost:8080/homer
```

## Остановка сервера

Для остановки и удаления всех контейнеров выполните:

```
docker compose down
```

Для остановки с сохранением данных (томов) просто выполните `docker compose stop`.

### Полное удаление данных (включая базу)

Предупреждение: Следующая команда **удалит все данные** из MongoDB, RabbitMQ и загруженные отчёты. Используйте только если хотите начать с чистого состояния.

```
docker compose down -v
```

## Полезные команды

КОМАНДА	ОПИСАНИЕ
<code>docker compose logs -f</code>	Просмотр логов всех сервисов в реальном времени
<code>docker compose logs fr-backend</code>	Лог только сервиса <code>fr-backend</code>
<code>docker compose restart</code>	Перезапуск всех сервисов
<code>docker compose ps</code>	Статус всех контейнеров
<code>docker compose stop</code>	Остановка контейнеров без удаления

## Устранение неполадок

- **Контейнер не запускается** - проверьте логи: `docker compose logs <имя_сервиса>`. Убедитесь, что образы доступны в реестре и параметры `.env` корректны.
- **Не удаётся подключиться к веб-интерфейсу** - убедитесь, что порт 8080 не занят другим приложением. Проверьте: `docker compose ps`. Проверьте что применены все миграции через логи сервиса backend.
- **Ошибки подключения к базе данных** - убедитесь, что MongoDB полностью инициализировался и скрипт

`mongo-init.js` выполнен успешно. Подождите 1–2 минуты после запуска.

- **Ошибки подключения к RabbitMQ** - проверьте, что учётные данные в `appsettings.Production.json` совпадают с настройками контейнера RabbitMQ в `docker-compose.yml`.

# Разворачивание Корпоративного Сервера или Публикатора без Kubernetes (docker-compose) - FerretDB

Продукты: [МоиОтчеты Корпоративный Сервер](#), [МоиОтчеты Публикатор](#)

## Общее описание

Данная инструкция представляет собой **пример** развёртывания сервера отчётов (Корпоративный Сервер / Публикатор) с использованием **FerretDB** в качестве базы данных и **RabbitMQ** в качестве брокера сообщений. FerretDB обеспечивает совместимость с протоколом MongoDB, при этом используя PostgreSQL в качестве хранилища.

Все конфигурационные файлы в данном каталоге являются **примерами** и предназначены для локального тестирования и ознакомления. Для рабочего контура необходимо адаптировать настройки под ваше окружение.

## Важная информация о поддержке

Компания «Быстрые отчеты» оказывает поддержку **только на продукты МоиОтчеты** (Корпоративный Сервер, Публикатор).

Мы **не несём ответственности** за работу, настройку и поддержку следующих компонентов:

- **FerretDB** (PostgreSQL, база данных)
- **RabbitMQ** (брокер сообщений)
- **PostgreSQL** (внутренняя СУБД FerretDB)
- Любые другие сторонние сервисы и контейнеры, представленные в данном примере

Если у вас возникнут проблемы с базой данных или сторонними сервисами, обратитесь к их документации или соответствующим сообществам.

## Предварительные требования

Перед началом установки убедитесь, что на вашей системе установлены:

1. **Docker** (версия 20.10+ рекомендуется) - [инструкция по установке](#)
2. **Docker Compose** (плагин для Docker, версия v2+) - обычно входит в Docker Desktop. Проверить можно командой:

```
docker compose version
```

3. Достаточное количество оперативной памяти (рекомендуется **не менее 4 ГБ** свободной RAM для всех контейнеров).
4. Порт **8080** (веб-интерфейс) должен быть свободен на хост-машине.

## Содержимое каталога

ФАЙЛ	НАЗНАЧЕНИЕ
<code>docker-compose.yml</code>	Описание всех сервисов (контейнеров), их связей и настроек
<code>.env</code>	Переменные окружения - имя и версия docker-образов продукта

ФАЙЛ	НАЗНАЧЕНИЕ
appsettings.Production.json	Конфигурация приложения: подключение к БД, брокеру, внутренние ключи безопасности, адреса сервисов

## Переменные окружения (.env)

Файл `.env` определяет параметры, которые подставляются в `docker-compose.yml` через синтаксис `${ПЕРЕМЕННАЯ}`.

```
FRC_SOURCE=xn--80akiaokt3b4b.xn--90aia9aifhdb2cxbdg.xn--p1ai/repository/docker-registry/moiotchety-corporate-server
FRC_VERSION=2026.2.2
```

### Пояснения:

- `FRC_SOURCE` - адрес docker-регистри (registry), откуда будут скачиваться docker-образы продукта МоиОтчеты.
- `FRC_VERSION` - версия продукта. Убедитесь, что указанная версия существует в реестре и у вас есть доступ к ней.

Важно: Не изменяйте `FRC_SOURCE`, если вы не уверены в корректности адреса реестра. Убедитесь, что ваш docker-демон имеет доступ к этому реестру.

## Конфигурация приложения (appsettings.Production.json)

Этот файл содержит основные настройки приложения. Рассмотрим ключевые секции:

```
{
  "Auth": {
    "UseOpenId": false,
    "UseLocal": true
  },
  "MainConfig": {
    "InternalHeaders": {
      "S56nHMSzjQzXYx5KJJsU3cjU": "000000000000000000000001",
      "x2aHtuSsxFeYqE8xPTaxAnbH": "000000000000000000000002",
      "QNpq2nyzvDNSCBLtVhMJ9e8m": "000000000000000000000003",
      "9MXgeFwLNjeUrJvw7N2aQv9F": "000000000000000000000004",
      "227881200f7cd43f238b327d": "000000000000000000000005"
    },
    "Frontend": {
      "Mixins": {
        "Head": "",
        "Body": ""
      },
      "InvariantLocale": ""
    },
    "License": "",
    "Server": {
      "Title": "МоиОтчеты Сервисные решения",
      "CorporateServerMode": true
    },
    "Rabbit": {
      "Host": "rabbitmq",
      "Port": 5672,
      "UserName": "fastreports",
      "Password": "Qwerty!23456",
      "QueueName": "ReportProcessQueue",
      "DirectExchangeName": "DirectEx",
      "AlternateExchangeName": "AExchange",
      "UnroutedQueueName": "Default",
      "Capacity": 1
    }
  }
}
```

```

},
"Database": {
  "ConnectionString": "mongodb://fastreport:Qwerty!23456@ferret:27017/ReportStore?maxPoolSize=100&waitQueueMultiple=100",
  "DatabaseName": "ReportStore"
}
},
"Gateway": {
  "BackendUrl": "http://fr-backend:80",
  "InternalKey": "QNpq2nyzvDNSCLtVhMJ9e8m",
  "SignInPagePath": "/account/signin?r={0}",
  "MaxConcurrentRequests": 200,
  "RequestQueueLimit": 5000
},
"Serilog": {
  "MinimumLevel": {
    "Default": "Debug"
  }
},
"Services": {
  "Items": {
    "OnlineDesigner": {
      "Type": "Static",
      "Urls": [
        "http://fr-onlinedesigner:80"
      ]
    },
    "Backend": {
      "Type": "Static",
      "Urls": [
        "http://fr-backend:80"
      ]
    },
    "FrontendApp": {
      "Type": "Static",
      "Urls": [
        "http://fr-app:80"
      ]
    },
    "Fonts": {
      "Type": "Static",
      "Urls": [
        "http://fr-fonts:80"
      ]
    },
    "StaticPreviewApp": {
      "Type": "Static",
      "Urls": [
        "http://fr-staticpreview:80"
      ]
    },
    "HomerApp": {
      "Type": "Static",
      "Urls": [
        "http://fr-homer:80"
      ]
    }
  }
},
"Designer": {
  "BackendUrl": "http://fr-backend:80",
  "InternalKey": "x2aHtuSsxFeYqE8xPTaxAnbH"
},
"WorkerCore": {
  "BackendUrl": "http://fr-backend:80",
  "InternalKey": "S56nHMSzjQzXYx5KJJsU3cjU"
}

```



```
"Database": {
  "ConnectionString": "mongodb://fastreport:Qwerty!23456@ferret:27017/ReportStore&maxPoolSize=100&waitQueueMultiple=100",
  "DatabaseName": "ReportStore"
}
```

- `ConnectionString` указывает на сервис `ferret` (FerretDB), который эмулирует MongoDB-совместимый интерфейс поверх PostgreSQL.
- `fastreport:Qwerty!23456` - логин и пароль для подключения к базе данных.
- `ReportStore` - имя базы данных.

Предупреждение безопасности: Пароль базы данных `Qwerty!23456` указан **только для примера**. Для рабочего контура **обязательно замените** его на надёжный пароль. Убедитесь, что пароли в `appsettings.Production.json`, `docker-compose.yml` и в настройках PostgreSQL/FerretDB **совпадают**.

## Gateway, Designer, WorkerCore, Scheduler, Fonts

Каждый из этих сервисов содержит поле `InternalKey`, которое должно **совпадать** с соответствующим ключом из секции `InternalHeaders`. Это обеспечивает безопасную внутреннюю коммуникацию.

## Файл docker-compose.yml

```
services:
  postgres:
    image: ghcr.io/ferretdb/postgres-documentdb:17-0.107.0-ferretdb-2.7.0
    restart: always
    environment:
      - POSTGRES_USER=fastreport
      - POSTGRES_PASSWORD=Qwerty!23456
      - POSTGRES_DB=postgres
    volumes:
      - ./postgres:/var/lib/postgresql/data
    networks:
      - fr-cs
  ferret:
    image: ghcr.io/ferretdb/ferretdb:2.7.0
    restart: always
    environment:
      - FERRETDB_AUTH=true
      - FERRETDB_SETUP_DATABASE=ReportStore
      - FERRETDB_POSTGRESQL_URL=postgres://fastreport:Qwerty!23456@postgres:5432/postgres
    networks:
      - fr-cs
  rabbitmq:
    image: bitnamilegacy/rabbitmq:3.9.27-debian-11-r9
    volumes:
      - ./rabbitmq:/bitnami
    restart: always
    networks:
      - fr-cs
    environment:
      - RABBITMQ_USERNAME=fastreports
      - RABBITMQ_PASSWORD=Qwerty!23456
  fr-backend:
    image: ${FRC_SOURCE}-backend:debian-${FRC_VERSION}
    restart: always
    volumes:
      - ./appsettings.Production.json:/app/appsettings.Production.json:ro
    networks:
      - fr-cs
  fr-gateway:
    image: ${FRC_SOURCE}-gateway:debian-${FRC_VERSION}
    ports:
```

```

ports:
  - 8080:80
restart: always
volumes:
  - ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
  - fr-cs
fr-fonts:
image: ${FRC_SOURCE}-fonts:debian-${FRC_VERSION}
restart: always
volumes:
  - ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
  - fr-cs
fr-app:
image: ${FRC_SOURCE}-app:debian-${FRC_VERSION}
restart: always
volumes:
  - ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
  - fr-cs
fr-staticpreview:
image: ${FRC_SOURCE}-static-preview:debian-${FRC_VERSION}
restart: always
volumes:
  - ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
  - fr-cs
fr-onlinedesigner:
image: ${FRC_SOURCE}-designer:debian-${FRC_VERSION}
restart: always
volumes:
  - ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
  - fr-cs
fr-homer:
image: ${FRC_SOURCE}-homer:debian-${FRC_VERSION}
restart: always
volumes:
  - ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
  - fr-cs
fr-workercore:
image: ${FRC_SOURCE}-workercore:debian-${FRC_VERSION}
restart: always
volumes:
  - ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
  - fr-cs
deploy:
  mode: replicated
  replicas: 1
  endpoint_mode: vip
fr-scheduler:
image: ${FRC_SOURCE}-scheduler:debian-${FRC_VERSION}
restart: always
volumes:
  - ./appsettings.Production.json:/app/appsettings.Production.json:ro
networks:
  - fr-cs
networks:
fr-cs:
  driver: bridge

```

Файл `docker-compose.yml` описывает все контейнеры, которые будут запущены.

Примечание: Все сервисы продукта (fr-\*) используют один и тот же конфигурационный файл `appsettings.Production.json`, который монтируется в контейнер как `read-only` том (`:ro`). Благодаря этому при обновлении образов конфигурация сохраняется.

Предупреждение безопасности:

- Образы контейнеров (`image`) берутся из переменных `.env`. Убедитесь, что вы доверяете источнику образов.
- Установленные в примере пароли (`Qwerty!23456`) используются **только для тестирования**. Замените их перед использованием в рабочем контуре.

## Установка и запуск

### Шаг 1. Создайте файлы конфигурации

В данном каталоге должны находиться файлы:

- `.env`
- `appsettings.Production.json`
- `docker-compose.yml`

Отредактируйте файлы, заменив пароли на свои.

### Шаг 2. Запустите контейнеры

```
docker compose up -d
```

Данная команда создаст и запустит все контейнеры в фоновом режиме (`-d`).

### Шаг 3. Проверьте, что все контейнеры запущены

```
docker compose ps
```

Убедитесь, что все контейнеры имеют статус **running**. Первый запуск может занять несколько минут - Docker будет скачивать образы из реестра. А так же будут применены миграции. Отслеживайте логи что миграции применились у сервиса `backend`.

### Шаг 4. Откройте веб-интерфейс

Перейдите в браузере по адресу:

```
http://localhost:8080
```

### Шаг 5. Войдите в систему

Используйте учётные данные по умолчанию (**только для тестирования!**):

- **Логин:** `admin@example.com`
- **Пароль:** `admin`

### Шаг 6. Панель администрирования

Для доступа к панели администрирования перейдите по ссылке:

```
http://localhost:8080/homer
```

## Остановка сервера

Для остановки и удаления всех контейнеров выполните:

```
docker compose down
```

Для остановки с сохранением данных (томов) просто выполните `docker compose stop`.

### Полное удаление данных (включая базу)

Предупреждение: Следующая команда **удалит все данные** из PostgreSQL, FerretDB, RabbitMQ и загруженные отчёты. Используйте только если хотите начать с чистого состояния.

```
docker compose down -v
```

## Полезные команды

КОМАНДА	ОПИСАНИЕ
<code>docker compose logs -f</code>	Просмотр логов всех сервисов в реальном времени
<code>docker compose logs fr-backend</code>	Лог только сервиса <code>fr-backend</code>
<code>docker compose restart</code>	Перезапуск всех сервисов
<code>docker compose ps</code>	Статус всех контейнеров
<code>docker compose stop</code>	Остановка контейнеров без удаления

## Устранение неполадок

- **Контейнер не запускается** - проверьте логи: `docker compose logs <имя_сервиса>`. Убедитесь, что образ доступен в реестре и параметры `.env` корректны.
- **Не удаётся подключиться к веб-интерфейсу** - убедитесь, что порт 8080 не занят другим приложением. Проверьте: `docker compose ps`. Проверьте что применены все миграции через логи сервиса backend.
- **Ошибки подключения к базе данных** - убедитесь, что PostgreSQL и FerretDB полностью инициализировались перед запуском сервисов продукта. Подождите 1–2 минуты после запуска.
- **Ошибки подключения к RabbitMQ** - проверьте, что учётные данные в `appsettings.Production.json` совпадают с настройками контейнера RabbitMQ в `docker-compose.yml`.

# Диагностика и устранение неполадок

Продукты: [МоиОтчеты](#) [Корпоративный Сервер](#)

Если при установке или запуске приложения возникли проблемы, данный раздел поможет вам в их устранении.

## Миграции

Корпоративный сервер предусматривает автоматическое применение миграций при запуске приложения. Ручное управление миграциями, как правило, не предусмотрено для конечного пользователя.

Хотя сервер имеет встроенный механизм восстановления после сбоев, определённые проблемы могут быть вызваны внешними факторами — например, отключением питания или потерей соединения с базой данных.

Если во время выполнения миграций произойдёт разрыв соединения с базой данных, система активирует режим блокировки и предотвратит запуск бэкенда для защиты целостности базы данных. В таком случае потребуется ручное восстановление состояния миграций.

**Важно:** Рекомендуется всегда создавать резервную копию базы данных перед обновлением версии корпоративного сервера. Это позволит оперативно восстановить работоспособность системы в случае возникновения ошибок.

Если резервная копия отсутствует, можно воспользоваться инструментами разработчика. Следует учитывать, что использование этих инструментов может привести к потере данных, однако они позволяют восстановить целостность базы данных и обеспечить стабильную работу сервера.

Для ознакомления с инструментами разработчика перейдите в соответствующий раздел.

## Инструменты разработчика

Стандартная поставка корпоративного сервера поддерживает развертывание с использованием `docker-compose` и `Kubernetes`. В данной документации рассматриваются только эти два способа развертывания.

Если у вас возникли сложности или требуется дополнительная помощь, вы можете обратиться в нашу службу технической поддержки.

### Docker

Чтобы использовать инструменты разработчика, необходимо временно остановить работающий экземпляр бэкенда. Выполните следующие команды:

1. Проверка состояния контейнеров

Получите список запущенных контейнеров:

```
docker-compose -f /путь/к/папке/docker-compose.yml ps
```

Обратите внимание: для выполнения всех команд необходим файл `docker-compose.yml`. Убедитесь, что текущая директория содержит этот файл, либо указывайте путь к нему с помощью параметра `-f`.

2. Остановка контейнера бэкенда

Остановите нужный контейнер (например, `fr-backend`):

```
docker-compose -f /путь/к/папке/docker-compose.yml stop fr-backend
```

3. Запуск контейнера с инструментами разработчика

Для работы с инструментами разработчика создайте новый контейнер и подключите его к той же сети, которая используется в `docker-compose.yml`.

Сначала проверьте доступные сети:

```
docker network ls
```

Затем выполните команду запуска контейнера. Не забудьте заменить `debian-2026.2.2` на актуальный тег, который указан в вашем файле `docker-compose.yml`:

```
docker run -it --network corporate_fr-cs -v /путь/к/папке/appsettings.Production.json:/app/appsettings.Production.json fastreport-corporate-server-backend:debian-2026.2.2 /bin/bash
```

После входа в контейнер запустите утилиту:

```
dotnet FastReport.Cloud.Backend.dll -- --corporate-dev-tools
```

Следуйте инструкциям, отображаемым в терминале. После завершения работы выйдите из контейнера:

```
exit
```

#### 4. Перезапуск основного сервиса

После использования инструментов разработчика перезапустите бэкэнд:

```
docker-compose -f /путь/к/папке/docker-compose.yml start fr-backend
```

## Kubernetes

Для использования инструментов разработчика необходимо временно остановить работающий экземпляр бэкэнд-сервиса. Выполните следующие шаги:

#### 1. Проверка состояния deployment

Получите список запущенных deployment-объектов. Обратите внимание на количество активных реплик — это значение потребуется на следующем этапе:

```
kubectl get deployments --namespace fr-corporate
```

Важно: укажите корректное пространство имён. В данном примере используется `fr-corporate`.

#### 2. Остановка подов путём масштабирования до нуля реплик

Установите количество реплик равным нулю, чтобы остановить соответствующие поды:

```
kubectl scale --replicas=0 deployment/fr-backend --namespace fr-corporate
```

#### 3. Запуск контейнера с инструментами разработчика

Создайте файл `my-debug-pod.yml` со следующим содержимым:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-debug-pod
  namespace: fr-corporate
spec:
  containers:
    - name: app
      image: fastreport-corporate-server-backend:debian-2026.2.2
      command: ["/bin/bash"]
      args: ["-c", "while true; do sleep 10; done"]
      tty: true
      volumeMounts:
        - mountPath: /app/appsettings.Production.json
          name: config-volume
          subPath: appsettings.Production.json
  volumes:
    - name: config-volume
      configMap:
        name: fast-report-config
        items:
          - key: appsettings.Production.json
            path: appsettings.Production.json
          defaultMode: 420
```

Примените манифест:

```
kubectl apply -f my-debug-pod.yml --namespace fr-corporate
```

#### 4. Подключение к поду

Подключитесь к созданному поду:

```
kubectl exec -it my-debug-pod --namespace fr-corporate -- /bin/bash
```

Запустите утилиту разработчика:

```
dotnet FastReport.Cloud.Backend.dll -- --corporate-dev-tools
```

Следуйте инструкциям в терминале. По завершении работы выйдите из контейнера:

```
exit
```

#### 5. Удаление отладочного пода

После выполнения всех операций удалите временный под:

```
kubectl delete pod my-debug-pod --namespace fr-corporate
```

#### 6. Восстановление количества реплик

Верните deployment к исходному количеству реплик (например, **1**):

```
kubectl scale --replicas=1 deployment/fr-backend --namespace fr-corporate
```