Version 2025.2.1

Manual for FastReport Corporate Server installation

Deploy a Kubernetes cluster

To install and configure the report server, you need to deploy the Kubernetes cloud ecosystem.

FastReport Corporate Server requires at least three nodes for correct operation: one master node and two worker nodes. The number of nodes can be increased when the load grows. And vice versa, the number of nodes can be decreased when the load drops. Dynamic control of the node quantity is currently not implemented.

Prepare and install Kubernetes components

Kubernetes is installed by executing the following bash scripts in sequence. These examples assume the Debian Linux operating system. In case of using another distribution, some steps will be different.

1. Disable swap partition.

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

2. Load the required kernel modules.

```
#
# 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. Configure the network properties of the system kernel.

```
#
# Confugre 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. Install additional software that will be required to install the orchestrator and report server.

```
apt-get update

#
# Install prerequested packages
#
prerequest=( curl sudo gnupg2 apt-transport-https ca-certificates software-properties-common )

for package in "${prerequest[@]}"
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. Install Container Runtime Interface (CRI).

In the following example, CRI container is used. Different versions of Kubernetes and Linux may require CRI-O to be installed. Kubernetes uses CRI to load and manage containers, and to run processes in those containers.

```
## 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. Install Kubernetes components.

```
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
```

Start Kubernetes

To start the cluster master node, edit 3 variables responsible for the corresponding paths to the configuration

file (it will be created automatically), IP address (it will be accessible externally), and internal subnet mask.

Then run the cluster initialization command.

```
export KUBECONFIG=/etc/kubernetes/admin.conf1
export MAIN_IF=192.168.1.191
export POD_NETWORK=10.244.0.0/16  # If flannel is used, this value cannot be changed!

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

The MAIN_IF address is the IP address of the cluster master node. The address can be an actual IP address or 192.168 subnet address.

If the initialization of the cluster master node is successful, a line will be displayed to initialize the cluster worker nodes. Use the mouse to copy this line and save it to a file. It contains a command to initialize the worker nodes. The command includes a secret key. If this key is exposed to an attacker, they can compromise the cluster.

Example line:

```
kubeadm join 192.168.1.191:6443 --token lw0lgz.d5zy9fb4jikc89yv \
	--discovery-token-ca-cert-hash sha256:ba83ed75e9fd5f430007000000000000039e1d05c7915a54435faaa7fe62b77
```

To add a node to the cluster, follow the steps described in the previous section (except for installing the helm chart), copy this command line to each node, and run it as administrator (root). This will add a new node to the cluster.

Configure kubectl.

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

Now you need to install the flannel service.x

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

If the installation is successful, the output of the `kubectl get pods --all-namespaces` should look approximately like this:

```
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-debian-10-1                    1/1     Running   0          5m27s
kube-system   kube-apiserver-debian-10-1          1/1     Running   0          5m22s
kube-system   kube-controller-manager-debian-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-debian-10-1          1/1     Running   0          5m32s
```

If all the installation steps were done correctly, the state from the example above will be reached within a minute. The cluster needs time to start and configure the coredns after the flannel is installed.

Install the nginx-ingress service

All described commands are executed only on the master node!

Add the ingress-nginx repository to Helm:

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

Install nginx in Kubernetes

Attention! The variable `MAIN_IF` must specify the IP address of the external interface to access the report server.

```
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
```

The following command configures `nginx` in cluster. nginx will accept all incoming requests and forward them to the gateway, which distributes incoming requests to the report server components.

```
HOST=my.server-server.com
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
```

Next, you will need an SSL certificate to set up a secure connection. Usually it can be purchased or obtained from a domain name registrar or purchased from a certification center. When using a report server on the intranet, you can create a self-signed certificate using the following command:

```
export CERT_NAME=my.server-server.com
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"
```

Register a certificate in Kubernetes named `fr-corporate-tls`. This certificate is then used by various report components, including `nginx`.

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

Register nginx-ingress in a Kubernetes cluster.
You need to set the `HOST`, variable corresponding to the domain name of the report 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
```

Log of helm and package installation No. 2

Kebrenetes-Dashboard is a WEB interface for cluster management. It allows you to manage Kubernetes using a WEB browser. Dashboard is not a necessary component for running the report server, but it helps to simplify server setup and configuration, as well as to monitor the state of nodes and containers.

Install Dashboard

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

Configure Dashboard

When configuring Dashboard, the variable `SERVER_DOMAIN_NAME` is used. Note, that the same IP address is used for Dashboard as for the report server in the configuration below. To access the Dashboard page, a rule is used to define the path to the Dashboard.

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

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
```

You can verify the created nginx server configuration using the following commands, replacing `nginx-ingress-controller-6674ff5868-t47xk` with the name of the created nginx container:

```
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
```

Next, you need to create a token that will be used for administrative access to the Dashboard. Create an administrative account:

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

Assign access rights to the administrative account:

```
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
```

Create a token:

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

Unpack and display the token used for authorization on the Dashboard:

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

Save the token for later use in an inaccessible place and use it for authorization in Dashboard. Example URL to access the dashboard:

```
https://my.server-server.com/dashboard/
```

Log of helm and package installation No. 3

Grafana is an analytics and interactive visualization component for monitoring cluster state. It is an optional component for deploying a report server, but it can be useful for tracking server state.

Loki is a log aggregator that is used in conjunction with Grafana.

Create a namespace for monitoring components

```
kubectl create namespace monitoring
```

Update charts for installing monitoring components

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

Install 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
```

Install Grafana

Don't forget to set the `SERVER_DOMAIN_NAME` variable.

```
export SERVER_DOMAIN_NAME="<server domain name>"

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 \
```

Obtain a Grafana password

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

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

Add ingress

This configuration block allows Grafana to be placed on the same network interface as the report server and

accessed via a link of the form: `https://my.server-server.com/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
```

To complete Grafana’s configuration, you need to go to its page, authorize, and add a link to Loki `http://loki.monitoring:3100`

Install the RabbitMQ message broker

To install RabbitMQ, the variable `RABBITMQ_PASSWORD`, containing the password needs to be defined. You can also configure the volume size for the message broker queue.

Add chart and update helm repository.

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

Create a namespace for RabbitMQ.

```
kubectl create namespace rabbitmq
```

The installation of RabbitMQ can vary depending on the type of installation. Below are two examples: for the Hetzner provider and for the installation on a test configuration.

Install on Hetzner cloud hosting

The script below uses a queue size of 10 GB.

```
export RABBITMQ_PASSWORD="<password for 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
```

Example of a minimal RabbitMQ installation to deploy on a test cluster

The configuration below uses a queue size of 4 GB. Please note, this example is the minimum for running FastReportort Corporate Server and the described volume configuration is not fail-safe.

```
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
```

Run `kubectl apply -f <name of yaml>` and the RabbitMQ installation script. Note, that the previously created PersistentVolumetClaim is used.

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

Add ingress configuration

This step is required if you want to access the RabbitMQ service control panel from outside the cluster.

```
export SERVER_DOMAIN_NAME="<server domain name>"

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
```

Access by `user` `********`.

Install MongoDB in a cluster

Add a repository for helm:

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

The report server uses MongoDB to store report templates, prepared reports, and various documents exported from prepared reports.

Prepare the namespaces:

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

Note of the MONGO_SERVICE_NAME variable¾this name will be used to access MongoDB from the report server in the future.

Further configuration will differ depending on whether cloud provider storage or local storage is used. Below are two examples. Use only one of them, or modify the settings for your cloud provider.

Configure storage for the Hetzner cloud provider

```
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
```

Configure local storage

StorageClass provides a means for administrators to describe the storage “classes” they provide.

```
MONGO_DATA_DIR=/devkube/mongodb
MONGO_DB=reports_db # Any name can be used

##
## 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:
          - debian-10-2
EOF
```

To install MongoDB, use the following script:

```
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
```

The result of running this script, in addition to installing MongoDB in the cluster, will be a file containing information about the result of the database installation. Note, that the parameters passed to the script were described at the very beginning of this document.

The easiest way to verify the correctness of MongoDB installation is to use Kubernetes Dashboard, the installation of which was described earlier.

Install the FastReport Corporate Server report server components on Kubernetes cluster nodes

The archive with images of report server components is called `frcs.tar.gz`. The link to download the archive is provided by Fast Reports Inc. via email.

To deploy the report server, perform the following steps on **each** node in the cluster:

1. Check if the ctr utility is installed. If not, install the podman utility (https://podman.io/) it is better to apply the package manager of the distribution you are using.
2. Copy the archive with the product to each cluster node and decompress the archive using the `gunzip frcs.tar.gz` command.
3. Download images from the tar archive to the local image repository. In case ctr is available, command to download the image is `ctr -n=k8s.io image import frcs.tar`, otherwise use the podman utility: `podman load -i frcs.tar`.
4. Verify that the images are available in local storage using the following command: `crictl img`.

In case everything went without errors, the output of `crictl img` will be similar to the following example:

```
IMAGE                                                     TAG                 IMAGE ID            SIZE
docker.io/bitnami/bitnami-shell                           10-debian-10-r394   45faa1e302205       83.1MB
docker.io/bitnami/mongodb                                 4.4.13-debian-10-r46 1fc49bc75137e       437MB
docker.io/bitnami/rabbitmq                                3.9.14-debian-10-r15 6c871950ea79a       240MB
docker.io/kubernetesui/dashboard                          v2.5.0              57446aa2002e1       226MB
docker.io/kubernetesui/metrics-scraper                    v1.0.7              7801cfc6d5c07       34.5MB
docker.io/library/nginx                                   1.14.2              295c7be079025       113MB
docker.io/rancher/mirrored-flannelcni-flannel-cni-plugin  v1.0.1              ac40ce6257406       8.38MB
docker.io/rancher/mirrored-flannelcni-flannel             v0.17.0             9247abf086779       60.8MB
k8s.gcr.io/coredns/coredns                                v1.8.4              8d147537fb7d1       47.7MB
k8s.gcr.io/defaultbackend-amd64                           1.5                 b5af743e59849       5.14MB
k8s.gcr.io/e2e-test-images/jessie-dnsutils                1.3                 4db8afc88fa88       261MB
k8s.gcr.io/etcd                                           3.5.0-0             0048118155842       296MB
k8s.gcr.io/ingress-nginx/controller                       <none>              c1695499dda39       288MB
k8s.gcr.io/ingress-nginx/kube-webhook-certgen             <none>              c41e9fcadf5a2       49.1MB
k8s.gcr.io/kube-apiserver                                 v1.22.5             059e6cd8cf78e       130MB
k8s.gcr.io/kube-controller-manager                        v1.22.5             04185bc88e08d       123MB
k8s.gcr.io/kube-proxy                                     v1.22.8             c1cfbd59f7747       105MB
k8s.gcr.io/kube-scheduler                                 v1.22.5             935d8fdc2d521       53.9MB
k8s.gcr.io/pause                                          3.5                 ed210e3e4a5ba       690kB
localhost/fastreport-corporate-server-app                 2022.2.3            2fc71408d7440       266MB
localhost/fastreport-corporate-server-backend             2022.2.3            154ab29853315       225MB
localhost/fastreport-corporate-server-designer            2022.2.3            fee8cebc69ebb       220MB
localhost/fastreport-corporate-server-gateway             2022.2.3            3ba7f871d665d       223MB
localhost/fastreport-corporate-server-homer               2022.2.3            da03d26d32fc6       264MB
localhost/fastreport-corporate-server-scheduler           2022.2.3            f39731aa92549       217MB
localhost/fastreport-corporate-server-static-preview      2022.2.3            0d3d42c9e1289       235MB
localhost/fastreport-corporate-server-workercore          2022.2.3            7bd7a33418995       898MB
localhost/fastreport-corporate-server-fonts               2022.2.3            b0f5e585a7563       216MB
us.gcr.io/k8s-artifacts-prod/ingress-nginx/controller     v0.34.1             6fb0739a741f4       332MB
```

At this stage the preparation for deployment of the report server is finished, further procedures are described in Install FastReport Corporate Server

Configure certificates

You need to create a certificate for the report server to work.

To create a self-signed certificate, use the following commands:

```
#!/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
```

To create a secret, run the following commands:

```
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
```

Alternative method of certificate generation suggested by the product user

1. Create and edit the `san.cnf` file:

```
[ 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. Generate a certificate:

```
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. Create a secret in the cloud server:

```
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
```

Configure FastReport Corporate Server settings

This section describes how to configure the report server. It covers general information. For fine-tuning the server, please refer to `admi_config` guide. The configuration is saved in a secure Kubernetes repository. It is optimal to perform this operation before running the containers described in the `fr-corporate-server` section.

The basic parameters of the report server are defined below. For convenience, they are shown in a separate section. Possible options: put this fragment in a separate file and use `dot operation` to embed them in the configuration script. Or copy this fragment directly into the configuration script. It is possible to add `export` before each variable and export to environment variables.

The value of the `MONGO_HOST` variable can be checked in Kubernetes Dashboard - it corresponds to the service name and its namespace separated by a dot. To successfully start the report server, you need a license - a special string describing the rights to use the report server. You can obtain it from the Fast Reports Inc. company.

The values of MONGO_USER and MONGO_PASS parameters must correspond to the same parameters specified when installing MongoDB.

```
NAMESPACE=fr-corporate-server
CLOUD_ENV=prod

## You can set any password and username you want, as long as they match the ones you set when installing MongoDB.
## MONGO_DB must be admin.
## However, there are rules for setting the hostname¾it is formed from the service name and the namespace in which MongoDB is registered.

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


## The following parameters are best left unchanged HERE. The host and username are set automatically, and the
## password for RabbitMQ is generated automatically during installation and stored in the Kubernetes secrets repository.
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)


## The developers claim that any sequence of characters can be here. And the length is not defined.
## The following values were used for the test and the installation was successful

DESIGNER_SECRET=yXgQDpNzohhN3dAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHrXiepieQtOaLwdqfRijMaZwy5p27tho7XK4C
WORKER_SECRET=9bYuYu4cvt73oWD3AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA5Z6i6pRzkR8mKGL1pTRnns3P0clxzTuVu
GATEWAY_SECRET=R0jEINg6OA4TssyAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAY46ZtsYowI7gB9yteZNMSvPoL6sGgiWC4
SCHEDULER_SECRET=fcfFZNWC3zyBmAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA5gD2MeexV3svLdy87qC6bX3bM2W6m
ZjRC


## The license obtained from Fast Reports Inc. must be entered in this variable. Unfortunately, the report server will not work without a license.
## Do not use quotation marks.

LICENSE=

## The MongoDB connection string is automatically generated based on the previously defined variables. It is better not to change anything here.

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"
```

Register the report server namespace in the cluster:

```
kubectl create namespace $NAMESPACE
```

Create the minimum configuration of the report server:

```
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
```

```
  },

  "MainConfig": {
   "InternalHeaders": {
    "$WORKER_SECRET": "00000000000000000000001",
    "$DESIGNER_SECRET": "00000000000000000000002",
    "$GATEWAY_SECRET": "00000000000000000000003",
    "$SCHEDULER_SECRET": "00000000000000000000004"
   },
  "License": "$LICENSE",
   "Server": {
    "Title": "FastReport Corporate Server",
    "CorporateServerMode" : true
   },
   "Rabbit": {
    "Host": "$RABBITMQ_HOST",
    "Port": 5672,
    "UserName": "$RABBITMQ_USER",
    "Password": "$RABBITMQ_PASS",
    "QueueName": "ReportProcessQueue",
    "DirectExchangeName": "DirectEx",
    "AlternateExchangeName": "AExchange",
    "UnroutedQueueName": "Default",
    "Capacity": 1
   },
   "Database": {
    "ConnectionString": "$CONNECTION_STRING",
    "DatabaseName": "$MONGO_DB"
   }
  },
  "Gateway": {
   "BackendUrl": "http://fr-backend.$NAMESPACE:80",
   "InternalKey": "$GATEWAY_SECRET",
   "SignInPagePath": "/account/signin?r={0}",
   "MaxConcurrentRequests": 200,
   "RequestQueueLimit": 5000
  },
  "Serilog": {
   "MinimumLevel": {
    "Default": "Debug"
   }
  },
  "Services": {
   "Items": {
    "OnlineDesigner": {
     "Namespace": "$NAMESPACE"
    },
    "Backend": {
     "Namespace": "$NAMESPACE"
    },
    "FrontendApp": {
     "Namespace": "$NAMESPACE"
    },
    "Fonts": {
     "Namespace": "$NAMESPACE"
    },
    "StaticPreviewApp": {
     "K8sServiceName": "fr-s-preview",
     "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
```

Please note, the above shell script contains basic settings that have been tested by the developers. When deploying the report server for the first time, it is recommended to use them to make sure that the server works. Before changing the settings, save this version, thus you will save a lot of time when fine-tuning the server using the parameters described in the `admin_config` section.

Install FastReport Corporate Server

The minimum configuration of the corporate report server consists of the nine components described below. Before installation, variables must be defined, for example, in the `config.sh` file. To avoid name conflicts, it is better to have a global configuration file for all report scripts. The following example shows how to initialize the variables used by the report server installation scripts:

```
IMAGE_STORAGE=docker.io/library
IMAGE_TAG=2022.2.22
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
```

Variable description:
`IMAGE_STORAGE` - address of the report server service images storage, the Corporate Server uses local storage; filling it with images is described in the Prepare images section. `IMAGE_TAG` - report server version. `NAMESPACE` - the Kubernetes namespace in which the report server will run. `SECRET_VOLUME_NAME` - the name of the volume on which the configuration and authorization information will be stored. `IMAGE_REGISTRY_SECRET_NAME` - the name of the record that stores authorization information on the server, holding service images on the report server. `PULL_POLICY` - determines how Kubernetes will download service images. Possible values:

- Always - always download images at service startup;
- IfNotPresent - download an image only if it is not present locally;
- Never - use pre-imported images, never download them from an external server.

Note: the described configuration does not download images from external storage (PULL_POLICY=Never), these variables must be set to avoid errors when applying the configuration. Before setting the PULL_POLICY value, please refer to the `fr-images` section to learn the settings.

Configuring the report server settings is described in Report server configuration. This document features the basic parameters that define the report server operating modes.

Gateway is a service for processing incoming requests

Gateway is a service that handles all connections to the report server. It analyzes requests and distributes them among other services. If the Gateway service does not work or does not work properly, none of the report server components will work. Ingess configuration by default sends all external requests to this service. In case of any problems with the service, start analyzing the problem by looking at the logs of the fr-gateway service.

```
#!/bin/sh

HTTPS_NODE_PORT=32222
IMAGE="${IMAGE_STORAGE}/fastreport-corporate-server-gateway:${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
```

```
      - 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
```

Install the report server scheduler

```
IMAGE="${IMAGE_STORAGE}/fastreport-corporate-server-scheduler:${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 is a service for viewing reports

```
#!/bin/sh

IMAGE="${IMAGE_STORAGE}/fastreport-corporate-server-static-preview:${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
```

Homer service

```
#!/bin/sh

IMAGE="${IMAGE_STORAGE}/fastreport-corporate-server-homer:${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
```

Backend service

```
#!/bin/sh


IMAGE="${IMAGE_STORAGE}/fastreport-corporate-server-backend:${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
```

Service Designer is an online report designer

```
#!/bin/sh

IMAGE="${IMAGE_STORAGE}/fastreport-corporate-server-designer:${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
```

Font service

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

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

Cloud application service

```
#!/bin/sh

IMAGE="${IMAGE_STORAGE}/fastreport-corporate-server-app:${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
```

Reporting service

```
#!/bin/sh

IMAGE="${IMAGE_STORAGE}/fastreport-corporate-server-workercore:${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
```

After performing all the above scripts, the installation of the report server is complete. To check the success of the installation and start working, log to the report server in a browser, using the domain name specified in the HOST variable when configuring nginx-ingress and in the window that appears enter the following data:

Username: `admin@example.com`
Password: `admin`

Have a great job!

Configuration

FastReport Corporate Server is configured through the `appsettings.json` files, which are located in the application directory. These files by default already have a number of properties that can be overridden in one of three ways:

1. Via `appsettings.{Enviroment}.json`, by default, the `Enviroment` variable is set to the `Production` value, so just change the content of the `appsettings.Production.json` file. It is case-sensitive!

   To change the value, use the environment variables: `ASPNETCORE_ENVIRONMENT` and `DOTNET_ENVIRONMENT`.

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

2. Via environment variables. The details are described below.

Prioritize configuration loading `appsettings.json` -> `appsettings.{Enviroment}.json` -> environment variables. This means that the configuration will be loaded from left to right, in other words, the files on the right will overwrite the configuration of the previous ones.

Description of `appsettings.json`

Each section will have an example configuration and a descriptive part: key -> value.

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

This key can be used as an environment variable:

```
export Logging__0__Name=ToEmail
```

Here the entry `Logging__0__Name` means accessing the `Logging` section, the element with `0` index, and the `Name` property.

`Kestrel` section

It allows you to configure http server to listen on a specific port or use a certificate. Example configuration:

```
{
  "Kestrel":{
    "Endpoints":{
      "Http":{
        "Url":"http://localhost:5000"
      },
      "Https":{
        "Url":"https://localhost:5001",
        "Certificate":{
          "Path":"<path to .pfx files>",
          "Password":"<certificate password>"
        }
      }
    }
  }
}
```

KEY	TYPE	DESCRIPTION
`Kestrel__Endpoints__Http_Url`	string (uri)	Access point for listening to http.
`Kestrel__Endpoints__Https_Url`	string (uri)	Access point for listening to https.
`Kestrel__Endpoints__Https_Certificate_Path`	string (local path)	Path to the `.pfx` certificate file for https traffic.
`Kestrel__Endpoints__Https_Certificate_Password`	string	Password to access the `.pfx` file.

`Auth` section

It allows you to configure the authentication and authorization process.

```
{
  "Auth":{
    "ClientId":"<openid client>",
    "Scopes":"openid and other scopes",
    "Authority":"https://id.fast-report.com",
    "Audience":"https://fastreport.cloud",
    "Secret":"<secret for for openid client>",
    "UseApiKeys":true,
    "UseOpenId":true,
    "UseLocal":false,
    "RsaXml":"<xml encoded object>",
    "UserInfoEndpoint":"https://example.com/userinfo",
    "TokenEndpoint":"https://example.com/token",
    "MetadataEndpoint":"https://example.com/.well-known/openid-configuration"

  }
}
```

KEY	TYPE	DESCRIPTION
`Auth__ClientId`	string	Unique identifier on the authentication server using the protocol `openid`.
`Auth__Scopes`	string	List of client areas by protocol `openid`.
`Auth__Authority`	string (uri)	Access point `Authority` from which the access token will be requested via the `openid` protocol.

KEY	TYPE	DESCRIPTION
`Auth__Audience`	string (uri)	Access point `Audience` for which the access token will be requested via the `openid` protocol.
`Auth__Secret`	string	Secret token to confirm authorization via the `openid` protocol.
`Auth__UseApiKeys`	boolean	It enables or disables the ability to authorize via access keys.
`Auth__UseOpenId`	boolean	It enables or disables authorization via `openid` protocol. Note, that the client must be configured for authorization via code flow.
`Auth__UseLocal`	boolean	[Coming Soon]
`Auth_RsaXml`	string (xml)	RSA serialized to XML. For example, `<RSAKeyValue><Modulus>{base64}</Modulus><Exponent>{base64}</Exponent><P>{base64}</P><Q>{base64}</Q><DP>{base64}</DP><DQ>{base64}</DQ><InverseQ>{base64}</InverseQ><D>{base64}</D></RSAKeyValue>`
`Auth__UserInfoEndpoint`	string (uri)	Access point used to request user data via `openid` protocol.
`Auth__TokenEndpoint`	string (uri)	Access point used to request user token via `openid` protocol.
`Auth__MetadataEndpoint`	string (uri)	Access point `well-known` with metadata for `openid`.

`MainConfig` section

It allows you to configure the basic configuration of FastReport Corporate Server.

```
{
  "MainConfig":{
    "Database":{
      "ConnectionString":"<connection string>",
      "DatabaseName":"<name of database>",
      "ExportsCollectionName":"Exports",
      "ReportsCollectionName":"Reports",
      "TemplatesCollectionName":"Templates",
      "TemplateFoldersCollectionName":"TemplateFolders",
      "ReportFoldersCollectionName":"ReportFolders",
      "ExportFoldersCollectionName":"ExportFolders",
      "UsersCollectionName":"Users",
      "SubscriptionPlansCollectionName":"SubscriptionPlans",
      "SubscriptionsCollectionName":"Subscriptions",
      "GroupsCollectionName":"Groups",
      "DataSourceCollectionName":"DataSources",
      "GridFSFilesCollectionName":"fs.files",
      "MigrationsCollectionName":"Migrations",
       "SubscriptionInvitesCollectionName":"SubscriptionInvites"
    },
    "Server": {
      "Title": "<Application title>",
      "Copyright": "<Short description of copyright>",
      "LogoLink" : "<path to logo, displayed in the user panel>",
      "LastSlaUpdate": "2022-11-28",
      "CorporateServerMode": false,
```

```
    "SlaLink": "<Link to the user agreement>",
    "FirstStepsVideoLink": "<Link to tutorial video>",
    "AboutLink": "<Link to page about product>",
    "HomePageLink": "<Link to product home page>",
    "AuthServerName": "<Name of ID Server>",
    "UsersPerWorkSpace": 0,
    "DataSourcesPerWorkSpace": 0
  },
  "Pagination":{
    "MaxEntries":120
  },
  "Rabbit":{
    "Host":"my-rabbit-server.com",
    "Port":5672,
    "UserName":"<user name>",
    "Password":"<user password>",
    "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": "",
  "SmtpServer": {
  "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 }
   }
  }
}
```

KEY	TYPE	DESCRIPTION
`MainConfig__Database__ConnectionString`	string	String to connect to the MongoDB database.
`MainConfig__Database__DatabaseName`	string	MongoDB server database name.
`MainConfig__Database__ExportsCollectionName`	string	Collection name for storing the list of exports.
`MainConfig__Database__ReportsCollectionName`	string	Collection name for storing the list of reports.
`MainConfig__Database__TemplatesCollectionName`	string	Collection name for storing the list of templates.
`MainConfig__Database__TemplateFoldersCollectionName`	string	Collection name for storing a tree structure of templates.
`MainConfig__Database__ReportFoldersCollectionName`	string	Collection name for storing a tree structure of reports.
`MainConfig__Database__ExportFoldersCollectionName`	string	Collection name for storing a tree structure of exports.
`MainConfig__Database__UsersCollectionName`	string	Collection name for storing a list of users.
`MainConfig__Database__SubscriptionPlansCollectionName`	string	Collection name for storing a list of subscription plans.
`MainConfig__Database__SubscriptionsCollectionName`	string	Collection name for storing a list of subscriptions.
`MainConfig__Database__GroupsCollectionName`	string	Collection name for storing a list of groups
`MainConfig__Database__DataSourceCollectionName`	string	Collection name for storing a list of data sources.

KEY	TYPE	DESCRIPTION
`MainConfig__Database__GridFSFilesCollectionName`	string	Collection name for storing a list of files.
`MainConfig__Database__MigrationsCollectionName`	string	Collection name for storing the list of applied migrations.
`MainConfig__Database__SubscriptionInvitesCollectionName`	string	Collection name for storing the list of invitations.
`MainConfig__Server__Title`	string	Name for the server that the application will use in the header.
`MainConfig__Server__Copyright`	string	Text for copyright information (or any other information that will be displayed at the bottom of the user application page).
`MainConfig__Server__LastSLAUpdate`	string	The date of the last modification of the user agreement, if you change this date to a newer one, when you open the user application, a dialog will open in which there will be a link to the updated conditions (from the field `SlaLink`).
`MainConfig__Server__CorporateServerMode`	boolean	This field changes the behavior of a part of the API, for example, if false is specified, then workspaces cannot be deleted in the admin panel.
`MainConfig__Server__FirstStepsVideoLink`	string	A link to the video tutorial to the corporate server that will appear when you first log in to the custom application or when you click on the `?` at the bottom of the page.
`MainConfig__Server__AboutLink`	string	Link to product information.
`MainConfig__Server__HomePageLink`	string	Link to the home page of the product.
`MainConfig__Server__AuthServerName`	string	The name for the authorization server specified in the `AuthConfig` section used in the user application.
`MainConfig__Server__UsersPerWorkSpace`	integer	The maximum number of users in a workspace set by an administrator.

KEY	TYPE	DESCRIPTION
`MainConfig__Server__DataSourcesPerWorkSpace`	integer	The maximum number of data sources in a workspace set by an administrator.
`MainConfig__Pagination__MaxEntries`	integer	Maximum number of items to paginate when requesting the list via API.
`MainConfig__Rabbit__Host`	string	Host address to access RabbitMQ.
`MainConfig__Rabbit__Port`	string	Port to access RabbitMQ.
`MainConfig__Rabbit__UserName`	string	Username to access RabbitMQ.
`MainConfig__Rabbit__Password`	string	Password to access RabbitMQ.
`MainConfig__Rabbit__QueueName`	string	HThe name of the queue that is used for the base name of specific user subscription queues.
`MainConfig__Rabbit__DirectExchangeName`	string	The name of the RabbitMQ exchange that is used to direct messages to users' subscription queues.
`MainConfig__Rabbit__AlternateExchangeName`	string	The name of the RabbitMQ exchange that will be processed by default when a subscription has not yet been created for the user.
`MainConfig__Rabbit__UnroutedQueueName`	string	The name of the RabbitMQ queue that will be processed by default when a subscription has not yet been created for the user.
`MainConfig__Rabbit__Capacity`	1	Do not change this value! It specifies the maximum number of parallel report generations on one working builder.
`MainConfig__SecurityAdvisor__RestrictUnsafe`	string	It enables or disables the `unsafe` keyword in the script.
`MainConfig__SecurityAdvisor__RestrictUnmanaged`	string	It enables or disables the `unmanaged` keyword in the script.
`MainConfig__SecurityAdvisor__RestrictExtern`	string (uri)	It enables or disables the `extern` keyword in the script.
`MainConfig__SecurityAdvisor__RestrictAsync`	string (uri)	It enables or disables the `async` keyword in the script.

KEY	TYPE	DESCRIPTION
MainConfig__SecurityAdvisor__RestrictTypeOf	string	It enables or disables the typeof keyword in the script.
MainConfig__SecurityAdvisor__Whitelist__0	boolean	It specifies the list of APIs that can be used without warnings in the report script. The number indicates the sequence number in the list.
MainConfig__SecurityAdvisor__Blacklist__0	boolean	It specifies the list of APIs that cannot be used in the report script. The number indicates the sequence number in the list.
MainConfig__License	string	License key. It is provided with the product.
MainConfig__SmtpServer__Server	string	Mail server address.
MainConfig__SmtpServer__Port	integer	SMTP server port.
MainConfig__SmtpServer__Username	string	SMTP server user name.
MainConfig__SmtpServer__Password	string	SMTP server user password
MainConfig__SmtpServer__From	string	Sender’s mail address (displayed in the email).
MainConfig__Tasks__Attempts	integer	Number of attempts to start the task by the worker.
MainConfig__Frontend__Mixins__Head	string	Mixes embedded in the header of the user panel (e.g., analytics code).
MainConfig__Frontend__Mixins__Body	string	Mixes embedded in the body of the user panel (e.g., analytics code).
MainConfig__Rfc2898CryptorSettings__Salt	string	Cryptographic algorithm salt (used to encrypt passwords).
MainConfig__Rfc2898CryptorSettings__Password	string	Cryptographic algorithm password.
MainConfig__Rfc2898CryptorSettings__IV	string	Cryptographic algorithm vector.
MainConfig__InvariantLocale	string	Permanent localization. It works independently of the browser language.
MainConfig__ErrorHooks_RocketChat_BaseAddress	string	Base address of all RocketChat webhooks.

KEY	TYPE	DESCRIPTION
MainConfig__ErrorHooks_RocketChat_Path	string	Path used to send a message.
MainConfig__DataSourcesConfig__XmlConfig__CommandTimeout	integer	Queue time for a response from the XML data source in seconds. If you do not specify a value, the default timeout for that data source will be used.
MainConfig__DataSourcesConfig__JsonConfig__CommandTimeout	integer	Queue time for a response from the JSON data source in seconds. If you do not specify a value, the default timeout for that data source will be used.
MainConfig__DataSourcesConfig__CsvConfig__CommandTimeout	integer	Queue time for a response from the CSV data source in seconds. If you do not specify a value, the default timeout for that data source will be used.
MainConfig__DataSourcesConfig__MySqlConfig__CommandTimeout	integer	Queue time for a response from the MySQL data source in seconds. If you do not specify a value, the default timeout for that data source will be used.
MainConfig__DataSourcesConfig__PostgreSqlConfig__CommandTimeout	integer	Queue time for a response from the PostreSQL data source in seconds. If you do not specify a value, the default timeout for that data source will be used.
MainConfig__DataSourcesConfig__MsSqlConfig__CommandTimeout	integer	Queue time for a response from the MS SQL data source in seconds. If you do not specify a value, the default timeout for that data source will be used.
MainConfig__DataSourcesConfig__OracleDbConfig__CommandTimeout	integer	Queue time for a response from the Oracle DB data source in seconds. If you do not specify a value, the default timeout for that data source will be used.
MainConfig__DataSourcesConfig__FirebirdConfig__CommandTimeout	integer	Queue time for a response from the Firebird data source in seconds. If you do not specify a value, the default timeout for that data source will be used.
MainConfig__DataSourcesConfig__MongoDbConfig__CommandTimeout	integer	Queue time for a response from the Mongo DB data source in seconds. If you do not specify a value, the default timeout for that data source will be used.

KEY	TYPE	DESCRIPTION
MainConfig__DataSourcesConfig__ClickHouseConfig__CommandTimeout	integer	Queue time for a response from the ClickHouse data source in seconds. If you do not specify a value, the default timeout for that data source will be used.

`Gateway` section

It allows you to configure the FastReport Corporate Server access gateway.

```
{
  "Gateway":{
    "WhiteListForDisabled":{
      "<any_claim_name>":"<claim_value>"
    },
    "IsDisabled":false,
    "ExcludePaths":[
      "/account",
      "/disabled"
    ],
    "SignInPagePath":"/account/signin?r={0}",
    "DisabledPath":"/disabled?r={0}",
    "IsSignInRequired":false
  }
}
```

KEY	TYPE	DESCRIPTION
Gateway__WhiteListForDisabled__<any_claim_name>	string	List of user assertions in the token to access the disabled FastReport Cloud.
Gateway__IsDisabled	boolean	It enables or disables access to FastReport Cloud.
Gateway__ExcludePaths__0	string	List of paths that can be accessed even when access to FastReport Cloud is disabled. The number indicates the sequence number in the list.
Gateway__SignInPagePath	string	String to redirect to the login form page.
Gateway__DisabledPath	string	String to redirect to a technical maintenance or access shutdown message.
Gateway__IsSignInRequired	boolean	It enables or disables the need to log in for the user to access FastReport Cloud.

`Services` section

It allows you to configure a list of services for gateway routing.

```
{
  "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>"
        }
      }
    }
  }
}
```

KEY	TYPE	DESCRIPTION
`Services__HealthCheckInterval`	integer	Interval for service health check, it is set in seconds.
`Services__Items__<name>__Urls__0`	string (url)	List of url URLs for accessing static services. The number indicates the sequence number in the list.
`Services__Items__<name>__Scheme`	string	Service access scheme: http or https.
`Services__Items__<name>__Port`	integer	Service access port.
`Services__Items__<name>__K8sServiceName`	string	Service name in Kubernetes.
`Services__Items__<name>__HostType`	string	The type of service to process the redirection by the gateway, it can take the values: `WebApp`, `API`, `External`, `Websocket`.
`Services__Items__<name>__PathBase`	string	Basic path for the service.
`Services__Items__<name>__Namespace`	string	Service namespace in Kubernetes.
`Services__Items__<name>__Type`	string	Type of service for access processing by the gateway; it can take the values: `Static`, `K8s`.

KEY	TYPE	DESCRIPTION
`Services__Items__<name>__PingPath`	string	Part of a query string to retrieve service health information.
`Services__Items__<name>__IsSignInRequired`	string	It specifies the need for authorization before users can access the service.
`Services__Items__<name>__Priority`	number	Service priority over others; the less the value is, the stronger it is.
`Services__Items__<name>__PingResponseCode`	integer	Status response code to be expected from health check.
`Services__Items__<name>__LoadBalanceMode`	string	t specifies the type of load balancing for this service; it can take one of the following values: `Random`, `AverageMetric`.
`Services__Items__<name>__HealthCheckAttemptsNumber`	integer	Number of attempts to check service health.
`Services__Items__<name>__WhiteListClaims__<claim_name>`	string	It indicates an assertion to the user to gain access to the service.

Instead `<name>` service name should be used; the list of services can be found in the following file `appsettings.json`.

`Constants` section

It allows you to limit the size of the request body in the application.

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

КЛЮЧ	ТИП	ОПИСАНИЕ
`Constants__LimitsMaxRequestBodySize`	long	The maximum size of the request body. If this limit is exceeded, the request will be rejected.

Additional `Migrations` Section for `MainConfig`

Allows configuring migration execution settings.

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

KEY	TYPE	DESCRIPTION
DataBaseCheckInterval	double	Interval (in seconds) to check database availability before starting migrations. Default: 1.
LockDoubleCheckInterval	double	Interval (in seconds) for re-checking migration lock. May need to be increased (~20 + ping time) if a replica set is used. Default: 5.
MigrationTimeOut	double	Time (in seconds) to wait for migration to complete before timing out. Default: 60.

Remove FastReport Corporate Server from the cluster

1. Remove report server components from the cluster:

```
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. Delete MongoDB.

Removing MongoDB will delete users, groups, report templates, prepared reports, and exported reports.

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

Please note, that depending on the settings, the above command may not delete disk space rented from cloud providers. If you rent a cluster from a cloud provider, use the provider's web interface to delete the Persistent Volumes of the report server - mongo-storage.

3. Removing the rest of the cluster components is optional, and if the server was used in a production cluster in conjunction with external services, removing these components may damage the infrastructure. For example, RabitMQ can be used by external services in the cluster.

4. To completely remove the cluster and its components, run the command:

```
kubeadm reset
```

Deploy the Corporate Server without Kubernetes environment

In addition to being deployed in Kubernetes, the Corporate Server can be run using docker compose.

To do this, the following steps should be performed.

Download the archive with Corporate Server docker images

The demo version of the Corporate Server can be requested in your account in the support section. If you already have the full version of the Corporate Server, you will have access to the license file in your account.

Upload FastReport Corporate Server images to local storage.

```sh
docker load -i frcs.tar.gz
```

Specify the license in the `appsettings.Production.json` file

Create a file called appsettings.Production.json and specify the license key in it.

```
{
  "Auth": {
    "UseOpenId": false,
    "UseLocal": true
  },
  "MainConfig": {
    "InternalHeaders": {
      "S56nHMSzjQzXYx5KJJsU3cjU": "000000000000000000000001",
      "x2aHtuSsxFeYqE8xPTaxAnbH": "000000000000000000000002",
      "QNpq2nyzvDNSCBLtVhMJ9e8m": "000000000000000000000003",
      "9MXgeFwLNjeUrJvw7N2aQv9F": "000000000000000000000004"
    },
    "Frontend": {
      "Mixins": {
        "Head": "",
        "Body": ""
      },
      "InvariantLocale": ""
    },
    "License": "",
    "Server": {
      "Title": "FastReport Corporate Server",
      "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@mongo:27017/?
authSource=ReportStore&readPreference=primary&appname=MongoDB%20Compass&ssl=false&maxPoolSize=100&waitQ
ueueMultiple=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"
  }
```

```
    }
  }
```

Find the License field and specify the key:

```
...
"License": "{key from license file}"
...
```

Configure the database

Create a mongo-init.js file with the following contents:

```
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" } ]
});
```

Docker compose file

Create a file named docker-compose.yml with the following contents:

```
version: "3.9"
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_DATABASE=ReportStore
    networks:
      - fr-cs
  rabbitmq:
    image: bitnami/rabbitmq:3.9
    volumes:
      - rabbitmq:/bitnami
    restart: always
    networks:
      - fr-cs
    environment:
      - RABBITMQ_USERNAME=fastreports
      - RABBITMQ_PASSWORD=Qwerty!23456
  fr-backend:
    image: fastreport-corporate-server-backend:2022.2.22
    restart: always
    volumes:
      - ./appsettings.Production.json:/app/appsettings.Production.json:ro
    networks:
      - fr-cs
  fr-gateway:
    image: fastreport-corporate-server-gateway:2022.2.22
    ports:
```

```
    ports:
      - 8080:80
    restart: always
#    environment:
#      - SomeEnviromentSettings=value
    volumes:
      - ./appsettings.Production.json:/app/appsettings.Production.json:ro
    networks:
      - fr-cs
  fr-fonts:
    image: fastreport-corporate-server-fonts:2022.2.22
    restart: always
    volumes:
      - ./appsettings.Production.json:/app/appsettings.Production.json:ro
    networks:
      - fr-cs
  fr-app:
    image: fastreport-corporate-server-app:2022.2.22
    restart: always
    volumes:
      - ./appsettings.Production.json:/app/appsettings.Production.json:ro
    networks:
      - fr-cs
  fr-staticpreview:
    image: fastreport-corporate-server-static-preview:2022.2.22
    restart: always
    volumes:
      - ./appsettings.Production.json:/app/appsettings.Production.json:ro
    networks:
      - fr-cs
  fr-onlinedesigner:
    image: fastreport-corporate-server-designer:2022.2.22
    restart: always
    volumes:
      - ./appsettings.Production.json:/app/appsettings.Production.json:ro
    networks:
      - fr-cs
  fr-homer:
    image: fastreport-corporate-server-homer:2022.2.22
    restart: always
    volumes:
      - ./appsettings.Production.json:/app/appsettings.Production.json:ro
    networks:
      - fr-cs
  fr-workercore:
    image: fastreport-corporate-server-workercore:2022.2.22
    restart: always
    volumes:
      - ./appsettings.Production.json:/app/appsettings.Production.json:ro
    networks:
      - fr-cs
    deploy:
      mode: replicated
      replicas: 2
      endpoint_mode: vip
  fr-scheduler:
    image: fastreport-corporate-server-scheduler:2022.2.22
    restart: always
    volumes:
      - ./appsettings.Production.json:/app/appsettings.Production.json:ro
    networks:
      - fr-cs
networks:
  fr-cs:
    driver: bridge
volumes:
```

```
mongo:
rabbitmq:
```

Start the Corporate Server

As a result, the downloaded images should be loaded in the docker, and the files appsettings.Production.json, docker-compose.yml, and mongo-init.js should be in the folder.

Start the server:

```
docker-compose up --build -d
```

Login to system

After executing all the above scripts, the installation of the report server is complete. To check the success of the installation and get started, use your browser to go to the report server at localhost:8080 and enter the following data in the window that appears:

Username: `admin@example.com`
Password: `admin`

To get to the admin panel, go to localhost:8080/homer

Shut down the Corporate Server

```
docker-compose down
```

Troubleshooting

If you encounter any issues during installation or startup, this section will help you resolve them.

Migrations

The corporate server is designed to automatically apply database migrations on startup. Manual migration control is generally not available to end users.
Although the server includes a built-in recovery mechanism, certain issues may arise from external factors unrelated to the server itself — for example, power outages or network interruptions.

If the connection to the database is lost during a migration process, the system will activate a lock mode and prevent the backend from starting, in order to avoid database corruption. In such cases, manual migration recovery will be required.

> **Important:** It is strongly recommended to create a backup of the database before upgrading the corporate server version. This ensures that you can restore the database and restart the server if anything goes wrong.

If no backups are available, developer tools are available for use. Please note that using these tools may result in data loss; however, they can help recover database integrity and allow the corporate server to run again.

To access the developer tools, proceed to the corresponding section.

Developer Tools

The standard distribution of the corporate server supports deployment via `docker-compose` and `Kubernetes`. This documentation covers only these two deployment methods.
If you encounter any issues or require additional assistance, please contact our technical support team.

Docker

To use the developer tools, you must first stop the running backend instance to avoid conflicts. Follow the steps below:

1. Check Running Containers

List all active containers:

```
docker-compose -f /path/to/folder/docker-compose.yml ps
```

> Note: All commands require the `docker-compose.yml` file. Ensure that the current directory contains this file, or specify its location using the `-f` parameter.

2. Stop the Backend Container

Stop the relevant container (e.g., `fr-backend`):

```
docker-compose -f /path/to/folder/docker-compose.yml stop fr-backend
```

3. Run a New Container with Developer Tools

Create a new container and attach it to the same network used in your `docker-compose.yml` file.

First, check available networks:

```
docker network ls
```

Then, run the container. Replace `debian-2025.2.0` with the appropriate tag from your `docker-compose.yml` file:

```
docker run -it --network corporate_fr-cs -v /path/to/folder/appsettings.Production.json:/app/appsettings.Production.json
fastreport-corporate-server-backend:debian-2025.2.0 /bin/bash
```

Once inside the container, launch the utility:

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

Follow the on-screen instructions. When done, exit the terminal:

```
exit
```

4. Restart the Main Service

After completing the tool operations, restart the backend service:

```
docker-compose -f /path/to/folder/docker-compose.yml start fr-backend
```

Kubernetes

To use the developer tools, you need to temporarily stop the running backend instance. Follow these steps:

1. Check Deployment Status

Retrieve a list of running deployments. Note the current number of replicas — you will need this value later:

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

> Important: Replace `fr-corporate` with your actual namespace if different.

2. Stop Pods by Scaling Replicas to Zero

Scale the deployment to zero replicas to stop all associated pods:

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

3. Launch a Container with Developer Tools

Create a file named `my-debug-pod.yml` with the following content:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-debug-pod
  namespace: fr-corporate
spec:
  containers:
    - name: app
      image: fastreport-corporate-server-backend:debian-2025.2.0
      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
```

Apply the manifest:

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

4. Connect to the Pod

Access the container:

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

Run the developer utility:

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

Follow the on-screen instructions. When finished, exit the shell:

```
exit
```

5. Delete the Debug Pod

After completing the tasks, remove the temporary pod:

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

6. Restore Replica Count

Return the deployment to its original replica count (e.g., `1`):

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