Программа «МИНОС»

**Руководство системного программиста**

**Листов \_19**

Инв. № подл.

Подп. и дата

Взам. инв. №

Инв. № дубл.

Подп. и дата

2023

**АННОТАЦИЯ**

Документ содержит описание структуры, процедуры установки, и проверки программы «МИНОС» и предназначен для системных программистов, осуществляющих подготовку программы к эксплуатации.

Настоящий документ предназначен для персонала, имеющего квалификацию системного администратора, обладающего знаниями и навыками в области системного администрирования, в том числе установки ОС Linux Debian, работы с СУБД MySQL, PostgreSQL, создания и настройки кластеров Kubernetes.

Документ не является руководством пользователя по установке и настройке, операционной системы, и прочего системного и технологического программного обеспечения.

**Содержание**

[1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ 4](#_Toc78797652)

[2. СТРУКТУРА ПРОГРАММЫ 4](#_Toc78797653)

[3. УСТАНОВКА ПРОГРАММЫ 6](#_Toc78797654)

[3.1. Установка СУБД 6](#_Toc78797655)

[3.2. Установка сервисов в кластере Kubernetes. 10](#_Toc78797656)

[3.2.1. Настройка Kubernetes. 10](#_Toc78797657)

[3.2.2. Размещение сервисов в кластере Kubernetes. 13](#_Toc78797658)

[3.3. Локальное развёртывание компонентов программы. 16](#_Toc78797659)

[4. ПРОВЕРКА ПРОГРАММЫ 17](#_Toc78797660)

[5. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ 17](#_Toc78797661)

[6. ПЕРЕЧЕНЬ СОКРАЩЕНИЙ 18](#_Toc78797662)

# 1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

* 1. Программа «МИНОС» обеспечивает выполнение функций, описанных в документе «МИНОС Руководство оператора»

Состав программы приведен в п. 2 настоящего документа.

* 1. Для обеспечения выполнения программы необходимы следующие технические средства:

Аппаратно-программные ресурсы выделенного сервера:

Операционная система: Linux Debian 9.

Используемая СУБД: MySQL, PostgreSQL 11, redis sentinel.

Развёрнутый Kubernetes-кластер.

 Сервис очередей NSQ.

# 2. СТРУКТУРА ПРОГРАММЫ

Структура Программы «МИНОС» представлена в таблице 1.

Таблица 1

|  |  |
| --- | --- |
| Название  | Наименование  |
| **Perl-fsin** | Программный сервис |
| **php-perl-proxy** | Программный сервис |
| **kv** | Программный сервис |
| **jwt-store** | Программный сервис |
| **go-cdr** | Программный сервис |
| **go-eavesdroper** | Программный сервис |
| **go-api** | Программный сервис |
| **go-person** | Программный сервис |
| **go-audioconverter** | Программный сервис |
| **go-queue-broker** | Программный сервис |
| **go-person-import** | Программный сервис |
| **notifier** | Программный сервис |
| **callrecords** | Программный сервис |
| **ReportMaker** | Программный сервис |
| **a2billingapi** | Программный сервис |
| **a2billing-grpc-v1** | Программный сервис |
| **json-export** | Программный сервис |
| **cdr-archive** | Программный сервис |
| **cashbox** | Программный сервис |
| -**ovc** | Программный сервис |

Вышеуказанные сервисы имеют следующее назначение:

**-Perl-fsin** –Frontend**-**часть программы. Отвечает за визуальное представление информации пользователям, авторизацию и аутентификацию (видимая часть сайта в браузере компьютера).

- **php-perl-proxy-** Backend-часть программы. отвечает за основную логику функционирования системы. Принимает данные от Frontend и взаимодействует с прочими вышеуказанными сервисами.

**-kv** – реализует хранилище данных о пользовательских правах.

-**jwt-store** – реализует хранилище для jwt-токенов, обновляет и выдает права пользователю.

**-go-cdr** – обеспечивает управление журналами аудио/видео звонков (запись данных в журнал, выбор необходимых данных по запросу пользователя).

-**go-eavesdroper** – оповещает аудио/видео терминалы о наличии звонка.

-**go-api** – реализует обработку данных в журнале и получение данных аудио/видео терминалов.

-**go-person** – предоставляет пользователю данные по спецконтингенту. Обеспечивает функции блокировок и перевода лиц спецконтингента между ИУ. Визуализирует информацию раздела «Учёт спецконтингента».

-**go-audioconverter** – обеспечивает конвертирование аудио звонков из «gsm» в «wav» и «mp3»-форматы.

-**go-queue-broker** – реализует обработку очередей для работы с журналами и данными спецконтингента.

-**go-person-import** – сервис для импорта спецконтингента из еxсel.

-**notifier** – сервис формирования и получения уведомлений в кабинете.

**-callrecords** – обеспечивает прослушивание wav-файлов с записями звонков и сохранение информации о факте прослушивания.

-**ReportMaker** – сервис формирования отчетов.

-**a2billingapi** – АPI для работы с картами в рамках биллинга a2billing.

-**a2billing-grpc-v1** – обеспечивает получение данных по регионам.

-**json-export** – обеспечивает экспорт данных в exсel и pdf для выгрузки журналов.

-**cdr-archive** – формирует zip архив с wav-файлами.

-**cashbox** – сервис кассы для ИУ. Предоставляет интерфейс внесения платежей на карты пользователя и получения отчетов по платежам.

-**ovc** – сервис для оповещения видео терминалов о звонках.

# УСТАНОВКА ПРОГРАММЫ

Все действия для установки компонентов программы выполняются от пользователя с правами «root».

## 3.1. Установка СУБД

Для функционирования определённых сервисов из состава программы необходимо установить СУБД MySQL.

Для установки СУБД MySQL, для сервисов **go-eavesdroper**, **go-api, a2billingapi, a2billing-grpc-v1** необходимо выполнить установку пакета:

**# apt-get install mariadb-server.**

Далее необходимо создать БД и пользователя с паролем, для этого нужно выполнить:

**# mysql -u root**

**# mysql>**

**CREATE DATABASE fsincabinet $**

**CREATE USER ' имя '@'%' IDENTIFIED BY 'пароль';**

**GRANT ALL PRIVILEGES ON \* . \* TO ' fsincabinet '@'%';**

**FLUSH PRIVILEGES.**

В полях «имя» и «пароль» прописать имя пользователя и пароль.

Для сервиса **jwt-store** необходимо установить СУБД PostgreSQL 11:

**# apt upgate**

**# apt install postgresql-11 postgresql-client-11**

Далее необходимо создать БД и пользователя с паролем, для этого нужно выполнить:

**# createdb passport;**

**# CREATE USER "vbuser" WITH ENCRYPTED PASSWORD 'password';**

**# GRANT ALL PRIVILEGES ON DATABASE passport** **TO "vbuser";**

В полях **"vbuser" и 'password'** прописать имя пользователя и пароль соответственно.

Для сервиса **go-cdr** необходимо выполнить:

**# apt upgate**

**# apt install postgresql-11 postgresql-client-11**

Далее необходимо создать БД и пользователя с паролем, для этого нужно выполнить:

**# createdb cdr\_dest;**

**# CREATE USER "vbuser" WITH ENCRYPTED PASSWORD 'password';**

**# GRANT ALL PRIVILEGES ON DATABASE cdr\_dest TO "vbuser";**

Для сервиса **go-person** необходимо выполнить:

**# apt upgate**

**# apt install postgresql-11 postgresql-client-11**

Далее необходимо создать БД и пользователя с паролем, для этого нужно выполнить:

**# createdb person;**

**# CREATE USER "vbuser" WITH ENCRYPTED PASSWORD 'password';**

**# GRANT ALL PRIVILEGES ON DATABASE go-person TO "vbuser";**

Для данного сервиса так же используется БД redis sentinel и сервис NSQ. Для установки redis sentinel в Kubernetes необходимо скачать конфигурационный файл values-production.yaml:

**# curl -Lo values-production.yaml** [**https://raw.githubusercontent.com/bitnami/**](https://raw.githubusercontent.com/bitnami/) **charts/master/bitnami/redis/values-production.yaml**

Открыть файл values-production.yaml, отредактировать секцию "sentinel":

sentinel:

 enabled: true

Далее добавить репозиторий и установить чарт:

**# helm repo add bitnami https://charts.bitnami.com/bitnami**

**# helm install redis bitnami/redis --values values-production.yaml**

После установки в кластере должны появиться следующие поды в статусе "running":



Для установки NSQ необходимо скачать конфигурационный файл:

**# git clone** [**https://github.com/adrianchifor/k8s-nsq.git**](https://github.com/adrianchifor/k8s-nsq.git)

Далее выполнить:

**# cd k8s-nsq**

**# kubectl apply -f deploy.yaml**

Для сервиса **go-audioconverter** необходимо развернуть redis sentinel.

Для сервиса **go-queue-broker** необходимо развернуть NSQ.

Для сервиса **go-person-import** необходимо развернутьredis sentinel и NSQ

Для сервиса **notifier** необходимо выполнить:

**# apt upgate**

**# apt install postgresql notifier -11 postgresql-client-11**

Далее необходимо создать БД и пользователя с паролем, для этого нужно выполнить:

**# createdb notifications;**

**# CREATE USER "vbuser" WITH ENCRYPTED PASSWORD 'password';**

**# GRANT ALL PRIVILEGES ON DATABASE notifications TO "vbuser";**

Для сервиса **callrecords** необходимо развернутьredis sentinel.

Для сервиса **ReportMaker** необходимо выполнить:

**# apt upgate**

**# apt install postgresql-11 postgresql-client-11**

Далее необходимо создать БД и пользователя с паролем, для этого нужно выполнить:

**# createdb cdr\_dest;**

**# CREATE USER "vbuser" WITH ENCRYPTED PASSWORD 'password';**

**# GRANT ALL PRIVILEGES ON DATABASE cdr\_dest TO "vbuser";**

Для сервиса **cashbox** необходимо выполнить:

**# apt upgate**

**# apt install postgresql-11 postgresql-client-11**

Далее необходимо создать БД и пользователя с паролем, для этого нужно выполнить:

**# createdb cashbox;**

**# CREATE USER "vbuser" WITH ENCRYPTED PASSWORD 'password';**

**# GRANT ALL PRIVILEGES ON DATABASE cashbox TO "vbuser";**

## 3.2. Установка сервисов в кластере Kubernetes.

### 3.2.1. Настройка Kubernetes.

На начальном этапе необходимо создать «мастер»-виртуальную машину и ноды. Далее необходимо установить ОС Linux Debian 9, sudo, docker и kubernetes, используя официальную документацию(<https://kubernetes.io/docs/setup/>).

Далее необходимо произвести настройку, для этого на мастере инициализируем kubeadm панель:

**# sudo kubeadm init --pod-network-cidr=10.244.0.0/16.**

cidr выставляется для конкретного плагина, в нашем случае это WeaveNet (взять его можно из официальной документации).

Затем необходимо выполнить:

**# mkdir -p $HOME/.kube**

**# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config**

**# sudo chown $(id -u):$(id -g) $HOME/.kube/config**

После этого нам нужно установить сетевой плагин:

**#kubectl apply -f "**[**https://cloud.weave.works/k8s/net?k8s-version=$**](https://cloud.weave.works/k8s/net?k8s-version=$)**(kubectl version | base64 | tr -d '\n')"**

Далее **каждую ноду** необходимо добавить в кластер. Пример команды (Для каждого кластера она разная, в целом Kubernetes при выполнении предыдущего пункта пишет ее в конце):

**# sudo kubeadm join 172.20.70.210:6443 --token ssot6s.vy0sld2u7kck5v1c --discovery-token-ca-cert-hash sha256:6b2ae90cb485e9f2d89c4b9a62e3e17fe57fc926652e681ce5aa917b68dffa3f**

На следующем шаге необходимо создать токен для Kubernetes dashboard, для этого нужно создать YAML-файл по подобию:

apiVersion: [rbac.authorization.k8s.io/v1](http://rbac.authorization.k8s.io/v1)
kind:ClusterRoleBinding
metadata:
name:admin-user
roleRef:
apiGroup: [rbac.authorization.k8s.io](http://rbac.authorization.k8s.io/)
kind:ClusterRole
name:cluster-admin
subjects:

kind:ServiceAccount
name:admin-user
namespace: kube-system

В трех местах, где прописано «admin-user» изменить запись на имя пользователя.

После того как скорректировали, выполнить:

**# kubectl apply -f 'имя\_файла'**

Далее выполнить:

**# kubectl -n kube-system get secret | grep admin-user**

(admin-user заменить на имя пользователя, которое указывали в YAML-файле).

**# kubectl -n kube-system describe secret admin-user-token-<id displayed by previous command>** (здесь указываем результат предыдущей команды). После этого в консоли отобразится токен и можно пробовать с ним заходить на dashboard.

Запустить kubectl proxy:

**# kubectl proxy.**

Далее необходимо перейти по ссылке [http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/](http://localhost:8001/api/v1/namespaces/kube-system/services/https%3Akubernetes-dashboard%3A/proxy/).

В появившемся окне загрузить YAML-файл, как показано на рисунке 1.



Рисунок 1 Внешний вид окна Dashboard

После этого выбрать вкладку «Token», подгрузить полученный токен и нажать «SIGN IN», после чего откроется окно Kubernetes Dashboard с правами суперадмина.

Далее необходимо установить пакетный менеджер Helm для Kubernetes.

Для этого следует скачать с официального сайта <https://github.com/helm/helm/releases> нужную версию и распаковать её, выполнив команду :

**# tar -zxvf helm-v3.0.0-linux-amd64.tar.gz**

Затем переместить бинарный файл в локацию, указанную в $PATH операционной системы:

**# mv linux-amd64/helm /usr/local/bin/helm**

### 3.2.2. Размещение сервисов в кластере Kubernetes.

Для размещения сервиса «**kv»** в кластереKubernetes необходимо руководствоваться инструкцией, расположенной: <https://www.consul.io/docs/k8s/installation/install>.

Ссылка на описание consul: <https://www.consul.io>

**$ curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -**

**sudo apt-add-repository "deb [arch=amd64]**

**https://apt.releases.hashicorp.com $(lsb\_release -cs) main"**

**$ sudo apt-get update && sudo apt-get install consul**

**# helm repo add hashicorp https://helm.releases.hashicorp.com**

**# helm search repo hashicorp/consul**

**# helm install consul hashicorp/consul --set global.name=consul**

 При первоначальном размещении в Kubernetes сервиса «**jwt-store**» необходимо в файле k8s/prod.yaml поменять значение TAG на актуальную версию приложения. Далее выполнить:

**# kubectl --context=fsin apply -f k8s/prod.yaml –record**

Репозиторий проекта: <https://git.zonatelecom.ru/fsin/jwtmanager>

Размещение в Kubernetes нижеуказанных сервисов, происходит аналогично с заменой в файле k8s/prod.yaml значения TAG. Далее также выполнить:

**# kubectl --context=fsin apply -f k8s/prod.yaml –record**

Для сервисов:

-**go-cdr** (репозиторий с проектом: <https://git.zonatelecom.ru/MicroArch/go-cdr>);

-**go-eavesdroper** (репозиторий с проектом: <https://git.zonatelecom.ru/MicroArch/go-eavesdroper>);

**-go-api** (репозиторий с проектом: <https://git.zonatelecom.ru/MicroArch/go-api>);

**-go-person** (репозиторий с проектом: <https://git.zonatelecom.ru/MicroArch/go-person>);

**-go-audioconverter** (репозиторий с проектом: <https://git.zonatelecom.ru/MicroArch/go-person>);

**-go-audioconverter** (репозиторий с проектом: <https://git.zonatelecom.ru/MicroArch/go-audioconverter>);

**-go-queue-broker** (репозиторий с проектом: [https://git.zonatelecom.ru/MicroArch/ go-queue-broker](https://git.zonatelecom.ru/MicroArch/%20%20%20%20%20%20%20%20go-queue-broker));

**-callrecords** (репозиторий с проектом: <https://git.zonatelecom.ru/MicrArch/callrecords>);

-**ReportMaker** (репозиторий с проектом: [https://git.zonatelecom.ru/MicroArch/ ReportMaker](https://git.zonatelecom.ru/MicroArch/%20%20%20%20%20%20%20%20%20%20%20%20%20ReportMaker));

-**cashbox** (репозиторий с проектом: <https://git.zonatelecom.ru/MicroArch/cashbox>);

**-оvc** (репозиторий с проектом: <https://git.zonatelecom.ru/fsin/ovc>);

-**php-perl-proxy** (репозиторий с проектом: <https://git.zonatelecom.ru/MicroArch/php-perl-proxy>);

**-a2billing-grpc-v1** (репозиторий с проектом: <https://git.zonatelecom.ru/fsin/a2billing>).

Для обновления вышеуказанных сервисов на сервере, образ с приложением при помощи CI/CD собирается, помещается в хранилище образов и далее разворачивается в Kubernetes автоматически, путем применения YAML-файлов, расположенных в директории /k8s/ с заменой указанной в них версии приложения на актуальную, при этом выполняется:

 **# sed -i 's/{TAG}/'dev'/g' k8s/devel.yaml**

 **# sed -i 's/{BUILD}/'$CI\_JOB\_ID'/g' k8s/devel.yaml**

 **# kubectl apply -f k8s/devel.yaml –record**

Для первоначального размещения сервиса **«go-person-import»** необходимо выполнить:

**# git clone** [**https://git.zonatelecom.ru/admin/projects/charts/go-person-import**](https://git.zonatelecom.ru/admin/projects/charts/go-person-import)

**# cd go-person-import**

**# helm install go-person-import--set image.tag=тег\_желаемой\_версии ./**

Репозиторий с проектом: <https://git.zonatelecom.ru/fsin/go-person-import.>

Для сервиса «**notifier**» выполнить:

**# git clone** [**https://git.zonatelecom.ru/charts/notifier**](https://git.zonatelecom.ru/charts/notifier)

**# cd notifier**

**# helm install notifier --set image.tag=тег\_желаемой\_версии ./**

Репозиторий с проектом: <https://git.zonatelecom.ru/MicroArch/go-notifier>.

Для сервиса «**a2billingapi**» выполнить:

**# git clone** [**https://git.zonatelecom.ru/charts/a2billingapi**](https://git.zonatelecom.ru/charts/a2billingapi)

**# cd a2billingapi**

**# helm install a2billingapi ./**

Репозиторий с проектом: <https://git.zonatelecom.ru/fsin/a2billing>.

Для сервиса «**json-export**» выполнить:

**# git clone https://git.zonatelecom.ru/charts/fsin-json-export**

**# cd fsin-json-export**

**# helm install fsin-json-export --set image.tag=тег\_желаемой\_версии ./**

Репозиторий с проектом: <https://git.zonatelecom.ru/fsin/json-export>

Для сервиса «**cdr-archive**» выполнить:

**# git clone** [**https://git.zonatelecom.ru/charts/cdr-archive**](https://git.zonatelecom.ru/charts/cdr-archive)

**# cd cdr-archive**

**# helm install cdr-archive --set image.tag=тег\_желаемой\_версии ./**

Репозиторий с проектом: <https://git.zonatelecom.ru/fsin/cdr-archive>

 После успешного размещения сервисов в Kubernetes, сервисы должны отображаться с зелёным значком. Пример показан на рисунке 2.



Рисунок 2. Окно отображения Pods

## 3.3. Локальное развёртывание компонентов программы.

Для локального развёртывания сервиса «**perl-fsin**» необходимо выполнить:

**# git clone** [**https://git.zonatelecom.ru/general/perl-fsin**](https://git.zonatelecom.ru/general/perl-fsin)

**# cd perl-fsin**

**# cp /opt/splatform/image/debug/SPlatform.pm /opt/splatform/lib/SPlatform.pm**

**# sudo chmod -R 777 /opt/splatform**

Репозиторий с проектом: <https://git.zonatelecom.ru/general/perl-fsin>

Процедура установки компонентов программы «Система оповещения личного состава СОЛС» описана в документе «Система оповещения личного состава СОЛС Руководство системного программиста».

# ПРОВЕРКА ПРОГРАММЫ

Для проверки программы необходимо выполнить действия, описанные в документах: «Программа МИНОС Руководство оператора», «Система оповещения личного состава СОЛС. Руководство оператора».

В случае корректного выполнения всех действий, установка программы считается завершённой успешно.

# 5. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ

Дополнительные возможности, отличные от приведённых в документах «Программа МИНОС Руководство оператора», «Программа МИНОС Описание программы» не предусмотрены и могут быть реализованы разработчиком на основании анализа потребностей эксплуатирующих учреждений.

# 6. ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

ОС-операционная система

СУБД-система управления базами данных

СОЛС-система оповещения личного состава

ИУ-исправительное учреждение

|  |
| --- |
| Лист регистрации изменений |
| Номера листов (страниц) | Всего листов (страниц)в докум. | № документа | Входящий № сопроводительного документа и дата | Подп. | Дата |
| Изм. | изменен-ных | заменен-ных | новых | аннули-рован-ных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |