ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ УНИТАРНОЕ ПРЕДПРИЯТИЕ
«ВСЕРОССИЙСКИЙ НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ИНСТИТУТ АВТОМАТИКИ ИМ. Н.Л. ДУХОВА»
(ФГУП «ВНИИА»)

УТВЕРЖДЕНО
Т0235/012-2025-ЛУ

№ Т0235/012-2025

РУКОВОДСТВО ПО УСТАНОВКЕ ТС УДП ЯОК ТОМ 1. РУКОВОДСТВО ПО УСТАНОВКЕ АСУДП «ПРИЗМА» ЧАСТЬ 1. РУКОВОДСТВО ПО УСТАНОВКЕ МОДУЛЯ «ПЛАТФОРМА»

В РАМКАХ ПРОЕКТА Р-МЕ1-22

«СОЗДАНИЕ ТИПОВОЙ СИСТЕМЫ УПРАВЛЕНИЯ ПРОИЗВОДСТВОМ ЯОК»

Код документа: Р-МЕ1-22-1.02.Рук.05.1

Редакция: 1

Лист регистрации изменений:

Дата	Автор	Редакция	Описание

Содержание

1. Перечень терминов, определений и сокращений

В настоящем документе используются термины, определения и сокращения в соответствии со словарем терминов и сокращений проекта.

Специфичные для настоящего документа термины, определения и сокращения представлены ниже (Таблица 1, Таблица 2).

Таблица 1 – Термины и определения

Термин	Определение
Kerberos	Сетевой протокол аутентификации взаимной аутентификации клиента и сервера перед установлением связи между ними
PostgreSQL	Объектно-реляционная система управления базами данных
Модуль	Структурная единица, входящая в группу «Поддерживающие системы» АСУДП «Призма»

Таблица 2 – Сокращения

Сокращение	Полное наименование
АРМ	Автоматизированное рабочее место
АСУДП, АСУДП «ПРИЗМА»	Автоматизированная система управления дискретным производством
АО «Гринатом»	Акционерное общество «Гринатом»
ПО	Программное обеспечение
СУБД	Система управления базами данных
БД	База данных
ФГУП ВНИИА им. Духова	Федеральное государственное унитарное предприятие «Всероссийский научно-исследовательский институт автоматики имени Н.Л. Духова»
ЯОК	Ядерный оружейный комплекс
API	Описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой
JDK	(«Java Development Kit») – комплект разработчика приложений на языке Java
JSON	(«JavaScript Object Notation») – текстовый формат обмена данными, основанный на JavaScript
REST	(«Representational State Transfer») – передача состояния представления
SQL	(«Structured Query Language») – язык структурированных запросов к реляционным базам данных
SSO	(«Single Sign-On») – технология единого входа, при которой пользователь переходит из одной системы в другую без повторной аутентификации

Сокращение	Полное наименование
XML	(«eXtensible Markup Language») – «расширяемый» язык разметки
АСУДП	Автоматизированная система управления дискретным производством
БД	База данных
КТС	Комплекс технический средств
ОС	Операционная система
ПО	Программное обеспечение
СУБД	Система управления базами данных
УД	Уровень доверия
RPM	Red Hat Package Manager
RPM-based	Дистрибутивы Linux, распространяемые в формате .rpm и использующий одноименный менеджер пакетов (например ОС RHEL, Centos, Fedora)

2. Введение

Настоящий документ является руководством по настройке и сопровождению АСУДП «Призма». Платформа.

Руководство предназначено для работников, обеспечивающих развертывание АСУДП «Призма». Платформа в организации.

3. Общие сведения

Данное руководство описывает требования к программному обеспечению, персоналу, а также порядок действий для установки АСУДП «Призма». Платформа на предприятии.

3.1.Требуемый уровень подготовки обслуживающего персонала

Компетенции персонала представлены ниже:

- Знание Astra Linux и его компонентов, опыт конфигурирования серверного и пользовательского ПО
- Знание работы веб сервисов и веб приложений
- Опыт работы с Docker и Docker swarm, конфигурирование Compose файлов
- Опыт работы с ALD Pro, протоколом Kerberos. Понимание работы контроллера домена. Механизмы ввода серверных и клиентских машин в домен. Настройка доверия между доменами
- Понимание модели OSI, стек протоколов tcp/ip, протоколов прикладного уровня (http, dns, smtp)
- Умение разбираться в логах и пользоваться соответствующим программным обеспечением для анализа трафика в режиме реального времени
- Администрирование Postgres, развертка кластеров и настройки резервного копирования, а также знание cli psql, а также языки SQL и PL/pgsql
- Работа с файловыми протоколами CIFS, SMB
- Bash и написание shell скриптов
- Опыт работы с maven, настройки java
- Концептуальное понимание работы брокера сообщений Kafka, а также KafkaConnect и его настроек

3.2.Перечень документации, с которой необходимо ознакомиться обслуживающему персоналу

Данные документы входят в состав дистрибутива или прилагаются с данным документом:

- «Руководство по установке АСУДП «Призма». Платформа»
- «Руководство по настройке и сопровождению модуля АСУДП «Призма». Платформа»
- «Руководство пользователя модуля АСУДП «Призма». Платформа»
- «Общее описание модуля АСУДП «Призма». Платформа»

4. Установка АСУДП

4.1. Состав программного обеспечения

Программное обеспечение, необходимое для работы АСУДП «Призма» в составе автоматизированной системы, к которой предъявляются требования к средствам технической защиты информации по 4 УД:

- OS Astra Linux 1.7.3, уровень безопасности: максимальный (Смоленск)
- Docker 20.10.2 и выше
- Postgres 11.17 и выше
- Ald Pro 1.4.0 или 1.4.1
- BellSoft Axiom Certified JDK Версия 17.0.6

Программное обеспечение, необходимое для работы АСУДП «Призма» в составе автоматизированной системы, к которой не предъявляются требования к средствам технической защиты информации:

- Debian-based дистрибутив (Debian 9 и выше) или RPM-based решения с открытым исходным кодом Docker 20.10.2 и выше
- Postgres 11.17 и выше
- Freeipa 4.6.4 и выше
- OpenJDK 17

4.2.Состав и содержание дистрибутивов.

Дистрибутив представляет из себя набор bash скриптов, docker images, jar и файлов конфигурации имеющий следующую структуру:

- docker-compose.{STACK}.yml (настроечный файл docker)
- install.{STACK}.sh (установочный файл)
- config (настройки внешних интеграций)
- distr (набор необходимого ПО)
- doc (набор документации)
- images (набор docker image’s)
- keytabs (директория в которой хранятся keytab’s)
- setup (установочные bash скрипты)
- sql (SQl файлы инициализации БД и тестовые данные)

4.3.Порядок установки.

4.3.1. Основные компоненты АСУДП «Призма»

4.3.1.1.Шлюз данных (gateway)

Так как все приложения АСУДП «Призма» находятся в изолированной среде, к ним нельзя обратиться “извне” напрямую. Gateway является основной точкой входа – перенаправляет все входящие http-запросы соответствующим образом.

Через данный сервис будут происходить подключения к приложениям АСУДП «Призма». В случае добавления нового сервиса – необходимо будет пересобрать образ, соответствующим образом отредактировав файл конфигурации ./source/gateway/site.conf

4.3.1.2.Сервисы приложений

Каждое приложение в АСУДП «Призма» состоит из серверной части и вэб-интерфейса, которые, в свою очередь, расположены в Docker контейнерах/сервисах:

– <Название приложения>:rest – контейнер с бэкэндом приложения. Серверная часть.
– <Название приложения>:webapp – контейнер с фронтэндом приложения. Внешний интерфейс.

Конфигурационные файлы по сборке данных контейнеров находятся в директории:

./setup/<Название приложения>.

4.3.1.3.Платформа в режиме брокера сообщений

Платформа в режиме брокера сообщений предоставляет возможность сбора, обработки, хранения и гарантированной доставки данных между СУБД и приложениями АСУДП «Призма».

Файлы конфигурации находятся в директории:

– /config - настроечные файлы стандартной интеграции (для внешних интеграций)
– /config/kafka-connectors/sink – для синхронизации данных между приложениями
– /config/kafka-connectors/source – внешние источники данных

Платформа в режиме брокера сообщений является своеобразной “шиной данных” между приложениями АСУДП «Призма», СУБД и источниками внешних данных.

4.3.2. Проверка настройки окружения

Для корректной установки АСУДП «Призма» необходимо, чтобы предварительно было установлено соответствующее ПО: контроллер домена freeipa/aldpro, кластер PostgreSQL версии 9.6 и старше, Docker 20.10 и старше в режиме swarm и docker registry.

Здесь и далее в качестве домена используется {DOMAIN.RU}

4.3.2.1.Проверка docker swarm, registry

На сервере с установленным docker`ом проверим его версию:

```
$ sudo docker –v

Docker version 20.10.2+dfsg1, build 2291f61 kadmin Service: RUNNING
```

А также посмотрим, проинициализирован ли Swarm:

```
$ sudo docker node ls

ID                            HOSTNAME         STATUS    AVAILABILITY
MANAGER STATUS   ENGINE VERSION

wtvq9r1owizjdy9msx1a9vxhe *   prism2.{DOMAIN.RU} Ready     Active
Leader           20.10.2+dfsg1 ipa-otpd Service: RUNNING
```

Если docker не в режиме swarm, необходимо его инициализировать:

```
$ sudo docker swarm init --default-addr-pool <выбранная подсеть *.*.0.0/16>
```

Выбранная подсеть не должна пересекаться с существующими. Для корректного указания подсети обратитесь к своему сетевому администратору.

На сервере, где развёрнут registry, убедимся, что он работает, например, если он запущен как отдельная служба, следующей командой:

```
$ sudo systemctl status docker-registry
```

```
docker-registry.service - the Docker toolset to pack, ship, store, and
deliver content

   Loaded: loaded (/lib/systemd/system/docker-registry.service; enabled;
vendor preset: enabled)

   Active: active (running) …
```

4.3.2.2.Проверка freeipa/aldpro

На сервере контроллера домена, проверим, что службы freeipa запущены и работают в штатном режиме.

```
$ sudo ipactl status

Directory Service: RUNNING

krb5kdc Service: RUNNING

kadmin Service: RUNNING

named Service: RUNNING

httpd Service: RUNNING

ipa-custodia Service: RUNNING

smb Service: RUNNING

winbind Service: RUNNING

ipa-otpd Service: RUNNING

ipa-dnskeysyncd Service: RUNNING
```

Проверить, что был создан соответствующий домен и он отображается в графическом интерфейсе AldPro.

4.3.2.3.Проверка Postgresql

На сервере с установленным кластером Postgresql необходимо прописать соответствующие переменные окружения (путь к исполняемым файлам и данным кластера) и проверить доступность последнего, например, с помощью одной из команд (зависит от установленного Postgresql):

```
$ sudo pg_lsclusters
```

```
$ sudo pg_ctl status
```

```
$ sudo systemctl status <название службы postgresql>
```

После этого, на хосте контроллера домена необходимо создать службу в ALDPRO для PostgreSQL. Например, командой:

```
$ ipa service-add postgres/$POSTGRES_FQDN
```

Затем, нужно выписать keytab и положить их в соответствующую директорию на сервере Postgres. При самостоятельном переносе keytab необходимо назначить владельцем файла keytab пользователя postgres.

```
ipa service-allow-retrieve-keytab postgres/$POSTGRES_FQDN --
users=$SUDO_USER
```

```
ipa-getkeytab -r -q -p postgres/$POSTGRES_FQDN -k $keytabFile

ipa service-disallow-retrieve-keytab postgres/$POSTGRES_FQDN --
users=$SUDO_USER
```

Т.к. АСУДП «Призма» для подключений использует Kerberos, кроме подключений внешней интеграции (например, БД integration), необходимо убедиться, что конфигурационные файлы БД настроены верно. Например, в нашем случае для БД integration будет парольная авторизация (md5 протокол), для всех остальных – через Kerberos (gss протокол).

Отредактируем конфигурационный файл <PG_HOME>/pg_hba.conf соответствующим образом

```
host    integration    all    <подсеть докера>    md5

host    all            all    <подсеть докера>    gss
```

Убедимся, что в <PG_HOME>/postgresql.conf указан верный путь на созданный ранее keytab

```
krb_server_keyfile = <PG_HOME>/prism2.postgres.keytab'
```

В случае, если необходимо создавать подключения к БД сторонними средствами пользователей с ролью администратор, то надо добавить в $PGDATA/pg_hba.conf следующую строку:

```
host all +ROLE_admin@admin all gss
```

После настройки $PGDATA/pg_hba.conf нужно перечитать конфигурации PostgreSQL.

Далее необходимо создать БД и назначить роли пользователю. Это можно сделать следующими командами в psql:

```
create database prism2db;

create role "GROUP_admin_service@admin";

create role "ROLE_admin@admin" createrole;

grant "ROLE_admin@admin" to "GROUP_admin_service@admin";

grant create on database prism2db to "ROLE_admin@admin";

create user "prism2-admin/prism2.{domain.ru}@{DOMAIN.RU}";

grant "GROUP_admin_service@admin" to "{prism2-
admin/prism2.{domain.ru}@{DOMAIN.RU}";
```

Обратите внимание, что имя пользователя "prism2-admin/prism2.{domain.ru}@{DOMAIN.RU} " определяется следующим образом:

prism2-admin/${PRISM2_HOST}.${domain}@${DOMAIN}

На основе скрипта ./sql/V0__init_schema_integration.sql можно подготовить схему и структуру баз данных в кластере, необходимых для работы платформы. (См. документацию “Входные данные”).

Все вышеперечисленное можно воспроизвести предоставленными скриптами

```
# на сервере контроллера домена (админом конроллера)

$ sudo ./install.{STACK}.sh generate-postgres-keytabs

# на сервере с postgres
```

```
$ sudo ./install.{STACK}.sh deploy-postgres
```

4.3.2.4.Настройка переменных окружения АСУДП «Призма»

env.sh – содержит переменные окружения используемые установщиком по умолчанию, если они не переопределены в основном скрипте

./install.{STACK}.sh.

Скрипт env.sh модифицировать не рекомендуется.

```
#!/bin/bash


#condition

export CONDITION=${CONDITION:="base"} #среда выполнения PRISM_2.0


#deploy settings

export GENERATE_KEYTABS=${GENERATE_KEYTABS:="false"} #генерировать keytabs
при наличии администского доступа к ald-Pro

export GENERATE_INTEGRATION_SCHEMA=${GENERATE_INTEGRATION_SCHEMA:="false"}
#генерировать схему integration в postgres для проверки интеграций


#global settings

export DOMAIN=${DOMAIN:="DEMO.RU"} # имя домена, в котором будет
установлена Призма

export DOMAIN_DN=${DOMAIN_DN:="dc=demo,dc=ru"} # ldap директория домена

export ENVIRONMENT=${ENVIRONMENT:="test"} # имя среды установки Призмы
(можно запустить несколько сред на одном сервере, например test1, test2)

export PRISM2_HOST=${PRISM2_HOST:="prism2"} # host имя основного сервера
Призмы

export PRISM2_PORT=${PRISM2_PORT:="80"} # порт, по которому будет доступен
http клиент Призмы

export PRISM2_FQDN=${PRISM2_HOST}.${DOMAIN,,} # полное имя сервера Призмы

export PRISM2_CORS="http://"${PRISM2_FQDN}$([[ $PRISM2_PORT = 80 ]] && echo
"" || echo ":"${PRISM2_PORT}) #CORS адрес сервера приложений

export KDC=${KDC:=${PRISM2_FQDN}} # адрес контроллера kerberos


#kafka settings

export KAFKA_HOST=${KAFKA_HOST:=${PRISM2_HOST}} # host имя сервера, на
котором запускается kafka

export KAFKA_FQDN=${KAFKA_HOST}.${DOMAIN,,} # полное имя сервера, на
котором запускается kafka

export KAFKA_STACK=${KAFKA_STACK:="kafka_"${ENVIRONMENT}} # название стека
docker swarm для сервисов kafka
```

```
export KAFKA_EXTENSION_COMPOSE_NAME=${KAFKA_EXTENSION_COMPOSE_NAME:=""} #
расширяющий базовый компоус файл для брокера сообщений


#postgres settings

export POSTGRES_HOST=${POSTGRES_HOST:=${PRISM2_HOST}} # host имя сервера,
на котором запускается кластер postgresql

export POSTGRES_PORT=${POSTGRES_PORT:="5432"} # порт кластера postgresql

export POSTGRES_FQDN=${POSTGRES_FQDN:=${POSTGRES_HOST}.${DOMAIN,,}}

export POSTGRES_VERSION=${POSTGRES_VERSION:="11"} # версия кластера
postgresql

export
POSTGRES_GENERATE_EXTERNAL_SUPERUSER=${POSTGRES_GENERATE_EXTERNAL_SUPERUSER
:="false"} #создавать superuser-а для внешних подключений

export
POSTGRES_EXTERNAL_SUPERUSER_NAME=${POSTGRES_EXTERNAL_SUPERUSER_NAME:="prism
2_superuser"} #логин superuser для внешних подключений

export
POSTGRES_EXTERNAL_SUPERUSER_PASSWORD=${POSTGRES_EXTERNAL_SUPERUSER_PASSWORD
:="admin"} #пароль superuser для внешних подключений

export POSTGRES_PRISM2_SUBNET=${POSTGRES_PRISM2_SUBNET:="0.0.0.0/0"}
#подсеть сервера приложений для настройки gss

export POSTGRES_KAFKA_SUBNET=${POSTGRES_KAFKA_SUBNET:="0.0.0.0/0"} #подсеть
брокера сообщений для схемы интеграции


#apache webserver gateway

export SERVER_STATUS_PORT=${SERVER_STATUS_PORT:="8080"} #порт статистики
apache webserver

export SERVER_STATUS_USER=${SERVER_STATUS_USER:="admin"} #логин apache
webserver

export SERVER_STATUS_PASSWORD=${SERVER_STATUS_PASSWORD:="admin"} #пароль
apache webserver


#docker settings

export
BASE_ASTRA_IMAGE=${BASE_ASTRA_IMAGE:="registry.astralinux.ru/library/alse:1
.7.4-mg11.2.0"} # имя базового docker образа для сборки JDK и gateway

export BASE_ASTRA_JDK_VERSION=${BASE_ASTRA_JDK_VERSION:="jdk-17.0.6"} #
версия JDK

export BUILD_ASTRA_IMAGE=${BUILD_ASTRA_IMAGE:="false"} # нужно ли собирать
docker образ astra linux + java (если false,

    # то образ должен существовать в docker regerstry или на сервере, на
котором осуществляется установка, до начала установки)

export HTTP_PROXY=${HTTP_PROXY:-} # адрес http прокси сервера, необходимый
для доступа в интернет при сборке образа astra
```

```
export REMOTE_MACHINE=${REMOTE_MACHINE:=${PRISM2_FQDN}} #сервер приложений в режиме MASTER для установки по ssh

export REMOTE_SWARM_MACHINES=${REMOTE_SWARM_MACHINES:=${REMOTE_MACHINE}} #сервер приложений в режиме NODE для установки по ssh

export REMOTE_KAFKA_MACHINE=${REMOTE_KAFKA_MACHINE:=${REMOTE_MACHINE}} #сервер брокера сообщений для установки по ssh

export REMOTE_USER_NAME=${REMOTE_USER_NAME:="admin"} #пользователь для подключения по ssh к серверу приложений

export REMOTE_KAFKA_USER_NAME=${REMOTE_KAFKA_USER_NAME:=${REMOTE_USER_NAME}} #пользователь для подключения по ssh к серверу брокера сообщений

export VIA_SSH=${VIA_SSH:="false"} # разворачивать по ssh(запускать без sudo) или прямо на локальной машине

export VERSION=${ENVIRONMENT} # версия docker образов, используемых при установке



#docker registry settings

export DOCKER_REGISTRY=${DOCKER_REGISTRY-${PRISM2_FQDN}":5000/"} # адрес docker registry (должен оканчиваться на /)

export DOCKER_REGISTRY_AS_DOCKER_CONTAINER=${DOCKER_REGISTRY_AS_DOCKER_CONTAINER:="false"} # docker registry развернут в докере, а не локально

export DOCKER_REGISTRY_CONTAINER_NAME=${DOCKER_REGISTRY_CONTAINER_NAME:="registry"} # docker registry service name - lkz grep поиска по имени

export DOCKER_REGISTRY_CONFIG_PATH=${DOCKER_REGISTRY_CONFIG_PATH:="/etc/docker/registry/config.yml"} #путь к кофигурации ocker registry

export PUSH_TO_DOCKER_REGISTRY=${PUSH_TO_DOCKER_REGISTRY:="true"} # нужно ли отправлять загруженнные и собранные docker образы в docker registry

export CLEAR_DOCKER_REGISTRY=${CLEAR_DOCKER_REGISTRY:="false"} #очистить docker registry от лишних слоев, на машине должен быть установлен jq(command-line processor for JSON)



#kafka kerberos configs

export KAFKA_HTTP_KEYTAB=${KAFKA_HTTP_KEYTAB:="http-kafka-"${KAFKA_FQDN}".keytab"} # имя файла keytab для HTTP сервиса kafka

export KAFKA_KEYTAB=${KAFKA_KEYTAB:="kafka-"${KAFKA_FQDN}".keytab"} # имя файла keytab для сервиса kafka



#prism2 deploy configs

export PRISM2_STACK=${PRISM2_STACK:="prism2_"${ENVIRONMENT}} # название стека docker swarm для сервисов Призмы

export PRISM2_APPLICATIONS=${PRISM2_APPLICATIONS} # список приложений Призмы для установки
```

```
export PRISM2_SCHEMALESS_APPLICATIONS=${PRISM2_SCHEMALESS_APPLICATIONS:-} #
список интеграционных приложений для установки


#prism2 kerberos configs

export
KERBEROS_CONFIG_SERVER=${KERBEROS_CONFIG_SERVER:="krbconfigserver:http://${
PRISM2_FQDN}:${PRISM2_PORT}/api/admin/config"}"?fail-fast=true&max-
attempts=99999999&max-interval=10001&multiplier=2&initial-interval=10000"

export
PRISM2_HTTP_PRINCIPAL=${PRISM2_HTTP_PRINCIPAL:="HTTP/"${PRISM2_FQDN}"@"${DO
MAIN}}

export PRISM2_HTTP_KEYTAB=${PRISM2_HTTP_KEYTAB:="http-
"${PRISM2_FQDN}".keytab"} # имя файла keytab для HTTP сервиса Призма

export PRISM2_ADMIN_PRINCIPAL=${PRISM2_ADMIN_PRINCIPAL:="prism2-
admin/"${PRISM2_FQDN}"@"${DOMAIN}}

export PRISM2_ADMIN_KEYTAB=${PRISM2_ADMIN_KEYTAB:="prism2-admin-
"${PRISM2_FQDN}".keytab"} # имя файла keytab для сервиса Призма
prism2_admin


#prism2 ldap configs

export LDAP_URL=${LDAP_URL:="ldap://"${KDC}} # url для синхронизации
пользователей из LDAP

export LDAP_USER_NAME=${LDAP_USER_NAME:-} #логин подключения к LDAP, если
не задан подключение осуществялется по GSS

export LDAP_PASSWORD=${LDAP_PASSWORD:-} #пароль подключения к LDAP

export LDAP_PERSON_ID_PROPERTY_NAME=${LDAP_PERSON_ID_PROPERTY_NAME:-} #
параметр пользователя в ad, хранящий personId пользователя в dict_person

export
LDAP_SEARCH_BASE_FILTER=${LDAP_SEARCH_BASE_FILTER:="CN=USERS,CN=ACCOUNTS,"$
{DOMAIN_DN}} #LDAP Base filter

export LDAP_SEARCH_FILTER=${LDAP_SEARCH_FILTER:="(&(OBJECTCLASS=PERSON))"}
#LDAP filter

export LDAP_LAST_CHANGE_KEY=${LDAP_LAST_CHANGE_KEY:="krbLastPwdChange"}
#LDAP ключ пользователя в AD с датой изменения пароля

export LDAP_EXPIRATION_KEY=${LDAP_EXPIRATION_KEY:="krbPasswordExpiration"}
#LDAP ключ пользователя в AD с датой


#Дополнительный фильтра LDAP для поиска политики паролей (только для
windows AD), для корреткного отображения срока действия пароля

#Включается через правку admin-rest/docker-compose.ext.yml

export
LDAP_PWD_SEARCH_BASE_FILTER=${LDAP_PWD_SEARCH_BASE_FILTER:=${DOMAIN_DN}}
#LDAP PWD Base filter
```

```
export
LDAP_PWD_SEARCH_FILTER=${LDAP_PWD_SEARCH_FILTER:="(&(objectCategory=domain)
)"} #LDAP PWD filter

export LDAP_PWD_MAX_AGE_KEY=${LDAP_PWD_MAX_AGE_KEY:="42"} # Задать срок
действия пароля в днях, чтобы не искать фильтром



#prism2 platform configs

export KAFKA_PORT=${KAFKA_PORT:="9095"} # порт kafka

export KAFKA_CONTROLLER_PORT=${KAFKA_CONTROLLER_PORT:="9093"} # порт kafka
контроллера

export KAFKA_CONNECT_PORT=${KAFKA_CONNECT_PORT:="9089"} # порт kafka
connect

export BOOTSRAP_SERVERS=${BOOTSRAP_SERVERS:=${KAFKA_FQDN}":"${KAFKA_PORT}}
# адреса bootstrap серверов kafka

export KAFKA_CONNECT_URL=${KAFKA_CONNECT_URL:="http://${KAFKA_FQDN}:9089"}
# url для подключения к kafka connect

export DATASOURCE_SCHEMA=${DATASOURCE_SCHEMA:="prism2_platform"} # схема БД
для платформы

export
DATASOURCE_URL=${DATASOURCE_URL:="jdbc:postgresql://${POSTGRES_FQDN}:${POST
GRES_PORT}/prism2db?currentSchema=${DATASOURCE_SCHEMA}&applicationName=${DA
TASOURCE_SCHEMA}[${ENVIRONMENT}]"} # строка подключения платформы к
postgres
```

4.3.3. Установка АСУДП «Призма»

Если все предыдущие шаги были выполнены верно, все необходимое ПО установлено и настроено соответствующим образом и все переменные окружения определены корректно, подготавливаем скрипты командой <Путь до папки с установкой>/setup

```
chmod a+x ./prepair-files.sh

./prepair-files.sh
```

и устанавливаем Призму + Брокер сообщений

```
sudo./install.{STACK}.sh
```

что равносильно запуску команд:

- sudo ./install.{STACK}.sh generate-postgres-keytabs (генерация сервиса и keytab-а для postgres), при выставленной переменной GENERATE_KEYTABS=”true”
- sudo ./install.{STACK}.sh generate-users-keytabs (генерация сервисов и keytab), при выставленной переменной GENERATE_KEYTABS=”true”
- sudo ./install.{STACK}.sh load-images.sh (загрузка docker images из папки ./images)
- sudo ./install.{STACK}.sh build-images.sh (сборка образа из сертифицированной платформы и загрузки ее в докер)
- sudo ./install.{STACK}.sh deploy-kafka.sh (разворачивание Платформы в режиме брокера сообщений)
- sudo ./install.{STACK}.sh deploy-prism2.sh (разворачивание Платформы в режиме админки + функциональные модули)

Если необходимо развернуть компоненты АСУДП «Призма» на нескольких серверах, можно использовать установщик, предварительно скопировав его на нужный сервер и запустив с соответствующими аргументами.

Рассмотрим выполнение установки более подробно.

4.3.3.1.Создание служб и пользователя в контроллере домена

Для корректной работы необходимо создать определённые службы, указать keytab`ы и создать пользователя для выполнения отложенных заданий. Данные шаги выполнит команда

```
sudo ./install.{STACK}.sh generate-users-keytabs
```

Установщик на сервере контроллера домена создаст службы HTTP, prism2-admin, kafka и укажет расположение сгенерированных keytab`ов командой (здесь и далее приведена часть команд из скрипта установщика generate-users-keytabs.sh):

```
ipa service-add HTTP/$PRISM2_FQDN

ipa service-mod HTTP/$PRISM2_FQDN --ok-as-delegate=true

keytabFile="../keytabs/${ENVIRONMENT}/${PRISM2_HTTP_KEYTAB}”

ipa-getkeytab -q -p HTTP/$PRISM2_FQDN -k $keytabFile
```

```
ipa service-add prism2-admin/$PRISM2_FQDN

ipa service-mod prism2-admin/$PRISM2_FQDN --ok-as-delegate=true --ok-to-
auth-as-delegate=true

keytabFile="../keytabs/${ENVIRONMENT}/${PRISM2_ADMIN_KEYTAB}"

ipa-getkeytab -q -p prism2-admin/$PRISM2_FQDN -k $keytabFile
```

```
ipa service-add kafka/$KAFKA_FQDN

keytabFile="../keytabs/${ENVIRONMENT}/${KAFKA_KEYTAB}"

ipa-getkeytab -q -p kafka/$KAFKA_FQDN -k $keytabFile
```

```
ipa service-add HTTP/$KAFKA_FQDN

keytabFile="../keytabs/${ENVIRONMENT}/${KAFKA_HTTP_KEYTAB}"

ipa-getkeytab -q -p HTTP/$KAFKA_FQDN -k $keytabFile
```

где переменные $KAFKA_FQDN и $PRISM2_FQDN – полные имена серверов, где установлены kafka и АСУДП «Призма» соответственно. Определены в файле env.sh

Далее создается пользователь prism2-executor командой

```
ipa user-add prism2-executor --first="Сервис выполнения отложенных задач" -
-last="Платформа Призма 2"
```

4.3.3.2.Загрузка и сборка образов

Все образы, расположенные в директории ./images, загружаются (пушатся) в registry.

Для этого используется следующая команда:

```
sudo ./install.{STACK}.sh load-images
```

Для установки АСУДП «Призма» необходимо собрать образ AstraLinux с сертифицированным jdk. Обратите внимание, что по умолчанию переменная BUILD_ASTRA_IMAGE = “false”, т.е. образ собираться не будет. Для сборки образа необходимо присвоить этой переменной значение “true”. Сертифицированный jdk должен находиться в директории ./distr/base-java. В таком случае, команда

```
sudo ./install.{STACK}.sh build-images
```

соберёт образы, необходимые при дальнейшей установке АСУДП «Призма», из образа AstraLinux и сертифицированного jdk (должен быть приобретён заказчиком), а также образ, используемый в дальнейшем для создания контейнера платформы и запушим последний в registry (если проставлен соответствующий флаг ${PUSH_TO_REGISTRY} = "true").

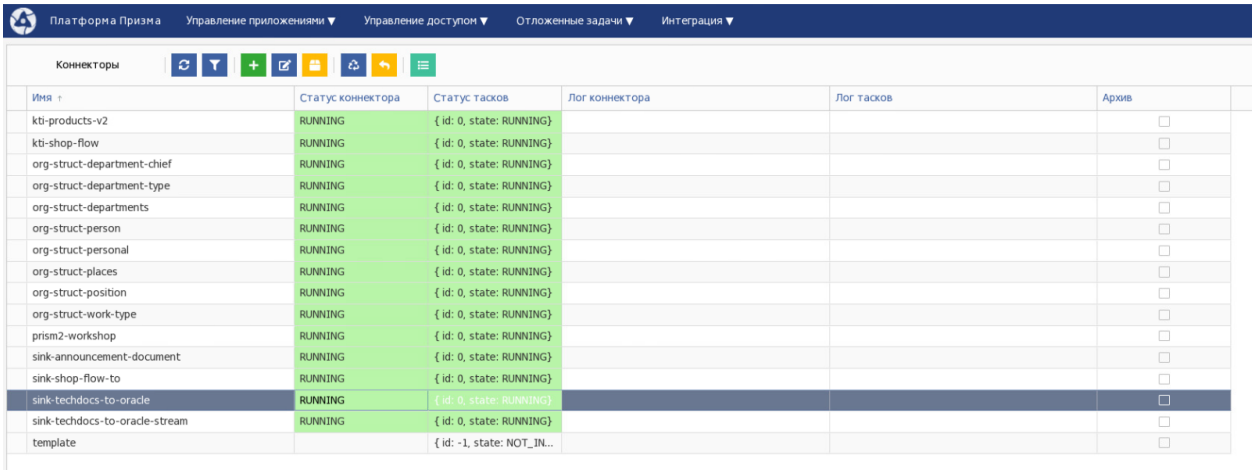
4.3.3.3.Развертка платформы в режиме брокера сообщений

Для того, чтобы развернуть платформу в режиме брокера сообщений используется следующая команда:

```
sudo ./install.{STACK}.sh deploy-kafka
```

Данный скрипт на основе compose-файла /setup/kafka/docker-compose.yml подготовит стэк для работы платформы в режиме брокера сообщений.

Для функционирования системы АСУДП «Призма» необходимо наличие следующих коннекторов (Рисунок 1):



Имя	Статус коннектора	Статус тасков	Лог коннектора	Лог тасков	Архив
kti-products-v2	RUNNING	{ id: 0, state: RUNNING}			
kti-shop-flow	RUNNING	{ id: 0, state: RUNNING}			
org-struct-department-chief	RUNNING	{ id: 0, state: RUNNING}			
org-struct-department-type	RUNNING	{ id: 0, state: RUNNING}			
org-struct-departments	RUNNING	{ id: 0, state: RUNNING}			
org-struct-person	RUNNING	{ id: 0, state: RUNNING}			
org-struct-personal	RUNNING	{ id: 0, state: RUNNING}			
org-struct-places	RUNNING	{ id: 0, state: RUNNING}			
org-struct-position	RUNNING	{ id: 0, state: RUNNING}			
org-struct-work-type	RUNNING	{ id: 0, state: RUNNING}			
prism2-workshop	RUNNING	{ id: 0, state: RUNNING}			
sink-announcement-document	RUNNING	{ id: 0, state: RUNNING}			
sink-shop-flow-to	RUNNING	{ id: 0, state: RUNNING}			
sink-techdocs-to-oracle	RUNNING	{ id: 0, state: RUNNING}			
sink-techdocs-to-oracle-stream	RUNNING	{ id: 0, state: RUNNING}			
template		{ id: -1, state: NOT_IN...			

Рисунок 1 – Коннекторы

Стоит обратить внимание, что коннекторы преднастроены определенным образом (Рисунок 2). В общем случае, для внешних источников в коннекторе потребуется изменить имя схемы, пароль и URL подключения к данным.

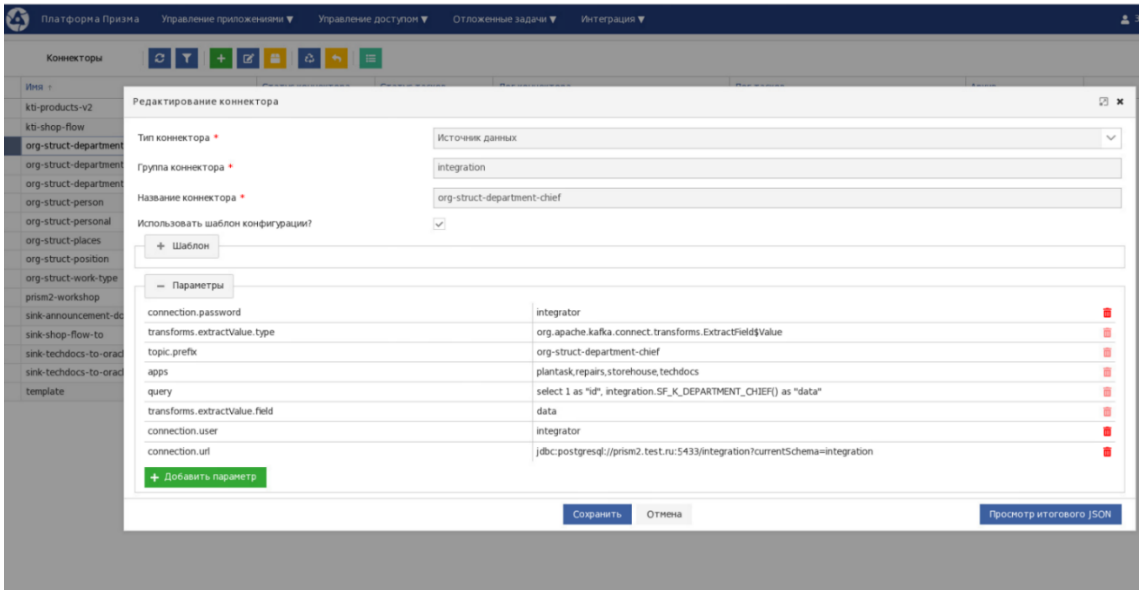


Рисунок 2 – Настройка коннектора

Список коннекторов к внешним источникам данных может посмотреть по пути – ./config/kafka-connectors/source/integration

4.3.3.4.Развертка АСУДП «Призма»

Чтобы развернуть платформу АСУДП «Призма» используется следующая команда:

```
sudo ./install.{STACK}.sh deploy-prism2
```

После установки, для получения доступа к веб приложению платформы, необходимо выдать права администратора одному из пользователей

```
$ sudo -u postgres psql --port $(./install.{STACK}.sh pg-port) prism2db -c
"grant \"GROUP_admin@admin\" to \"admin@{DOMAIN}.RU\""
```

В дальнейшем, выдача прав может осуществляться через веб интерфейс платформы.

4.3.4. Структура дистрибутива

Структура дистрибутива состоит из:

- /config - настроечные файлы стандартной интеграции (для внешних интеграций)
- /config/kafka-connectors/sink – для синхронизации данных между приложениями
- /config/kafka-connectors/source – внешние источники данных
- /distr – файлы дистрибутивов
- distr/platfrom – бинарник kafka + файлы конфигурации
- distr/base-java – файл jdk.tar.gz (архив с Java JDK) для сборки базового docker образа astra-java.
- /images – tar-файлы, с docker-образами приложений Призмы
- /keytabs - в данном каталоге находятся keytab файлы сервисов Призмы
- /source – конфигурационные файлы для сборки образов Docker (gateway и base-java)
- /sql - sql скрипты, необходимые для запуска платформы (создание БД в кластере PostgreSQL) и развертывания стандартной (преднастроенной) интеграции
- V0__init_schema_integration_prism2.sql – шаблон скрипта по созданию необходимой структуры в БД для подготовленных данных из внешних источников

- gen_data_prism2.sql - скрипт по наполнению предварительными данными БД
- install.sql.template – скрипт по созданию БД, пользователей и выдаче грантов
- install.{STACK}.sh, docker-compose.{STACK}.yml – файлы настройки окружения
- ./setup/<имя приложения> - файл конфигурации для создания контейнера соответствующего приложения. Представлены два compose-файла:
- docker-compose.yml – базовый конфигурационный файл. Менять не следует.
- docker-compose-ext.yml – конфигурационный файл для внесения изменений.
- install.{STACK}.sh – скрипт для настройки окружения и запуска установки. В нем можно переопределить любые переменные окружения, которые, по умолчанию, если они не заданы, определяются в скрипте ./setup/env.sh
- install.{STACK}.sh.5s.template – шаблон для пользователя для создания своего скрипта установки. Данный шаблон инициализирует переменные окружения, необходимые для установки АСУДП «Призма» на 5 разных ЭВМ.
- install.{STACK}.sh.template – шаблон для пользователя для создания своего скрипта установки. Данный шаблон инициализирует переменные окружения, необходимые для установки АСУДП «Призма» на одну ЭВМ.
- *.yml и *.yml.template файлы содержат информацию о сборке Docker-контейнера основной системы администрирования АСУДП «Призма».
- ./setup/env.sh - содержит переменные окружения, используемые установщиком по умолчанию, если они не переопределены в основном скрипте /home/admin/prism2/install.xxx.sh. Скрипт env.sh модифицировать не рекомендуется.
- ./setup/build-gateway.sh – скрипт для сборки Gateway приложения АСУДП «Призма». Им можно воспользоваться для сборки нового образа gateway.
- ./setup/build-images.sh – скрипт сборки основного контейнера с бэкэндом приложения. Для корректной сборки контейнера в директории ./prism2/distr должна существовать директория с сертифицированным JDK, который должен быть приобретен заказчиком отдельно.
- ./setup/deploy-kafka.sh – скрипт развёртывания платформы в режиме брокера сообщений.
- ./setup/deploy-postgres.sh – скрипт для создания кластера PostgreSQL
- ./setup/deploy-prism2.sh – скрипт развертывания приложения АСУДП «Призма».
- ./setup/download-images.sh – скрипт загрузки docker images сервисов АСУДП «Призма».
- ./setup/generate-users-keytabs.sh – скрипт, создающий службы prism2-admin, kafka, http в FREEIPA, также генерирующий keytabs для этих служб (если их нет) и создающий пользователя prism2-executor – от его имени будут выполняться отложенные задания приложения.
- install.sh – основной скрипт установки, вызываемый из install.{STACK}.sh
- ./setup/load-images.sh - скрипт, в котором поставленным images из директории ./images присваиваются тэги и где images пушатся в локальный registry.
- ./setup/palette.sh – содержит кодировки цветов, используемых при установке приложения, а также цвета иконок продукта
- ./setup/pg-cluster.sh – выводит название Postgres кластера с которым будет работать/работает призма с текущими переменными окружения. Является заменой стандартных команд PostgreSQL
- ./setup/pg-port.sh – выводит порт на котором работает БД приложения (только для установленного PostgreSQL из поставки AstraLinux). Является заменой стандартных команд PostgreSQL
- ./setup/pg-version.sh – выводит версию БД, используемую приложением (работает при уже установленном кластере для АСУДП «Призма». Является заменой стандартных команд PostgreSQL

- ./setup/generate-postgres-keytabs.sh – создает Kerberos службу postgres, а также создает в корне приложения ./prism2/ директорию postgres, где будет хранится keytab службы во FREEIPA.
- ./setup/deploy-postgres.sh – создает в корне приложения ./prism2/ директорию postgres, где сгенерируется sql скрипт создания базы данных, а также пользователей и ролей необходимых для работы АСУДП «Призма»
- ./setup/prepair-files.sh – скрипт для подготовки файлов к исполнению – выдача необходимых грантов на запуск скриптов.
- ./docker-subnet.sh – скрипт получения подсети Docker bridge

4.4.Порядок проверки работоспособности.

Работоспособность определяется способностью программы функционировать и обрабатывать информацию в соответствии с программными документами при отсутствии сбоев технических средств.

Проверка функционирования docker-контейнеров описано в п.6 документа «Общее описание модуля АСУДП «Призма». Платформа».