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

УТВЕРЖДЕНО
Т0235/011-2025-ЛУ

№ Т0235/011-2025

РУКОВОДСТВО ПО НАСТРОЙКЕ И СОПРОВОЖДЕНИЮ ПОДСИСТЕМ ТС УДП ЯОК ТОМ 1. РУКОВОДСТВО ПО НАСТРОЙКЕ И СОПРОВОЖДЕНИЮ АСУДП «ПРИЗМА» ЧАСТЬ 1. РУКОВОДСТВО ПО НАСТРОЙКЕ И СОПРОВОЖДЕНИЮ МОДУЛЯ «ПЛАТФОРМА»

В РАМКАХ ПРОЕКТА Р-МЕ1-22

«СОЗДАНИЕ ТИПОВОЙ СИСТЕМЫ УПРАВЛЕНИЯ ПРОИЗВОДСТВОМ ЯОК»

Код документа: Р-МЕ1-22-1.02.Рук.12.1

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»
БД	База данных
ПО	Программное обеспечение
СУБД	Система управления базами данных
КТС	Комплекс технический средств
ОС	Операционная система
УД	Уровень доверия
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

Сокращение	Полное наименование
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
XML	(«eXtensible Markup Language») – «расширяемый» язык разметки
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, входящего в Автоматизированную систему управления дискретным производством «Призма».

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

3. Общие сведения

Данное руководство описывает требования к программному обеспечению, персоналу, а также порядок действий для настройки 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»

3.3.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

3.4.Состав и содержание дистрибутивов.

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

- docker-compose.test.yml (настроечный файл docker)
- install.test.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. Описание операций

4.1.Обновление Astra Linux

Обновление Astra Linux согласно руководству пользователя Astra Linux.

4.2.Миграция и инициализация приложения

Т.к. у приложения нет прав на работу с объектами БД, то за выполнение миграции и инициализации БД приложения отвечает платформа. При старте, приложение информирует платформу о готовности к миграции с помощью запроса:
POST{platform_url}/applications/{application}/events/ready
(Authorization: токен HTTP/prism2.domain.net@DOMAIN.NET)

После получения данного запроса, платформа читает строку подключения к БД из параметра {application}/default/spring.datasource.url. Схема БД определяется по параметру подключения -currentSchema. Платформа подключается к БД и инициализирует схему приложения:

- создается таблица prism_schema_info

- роль ROLE_prism_executor@{application}

- группу GROUP_prism_executor@{application}

- включит в группу GROUP_prism_executor@{application} пользователя prism2-

executor@domain.net

prism2-executor@domain.net специальный пользователь, от имени которого платформа будет вызывать приложения. Имя пользователя можно переопределить параметром kerberos.executor.principal.

Далее платформа запрашивает информацию для проведения миграции приложения:

```
GET {application_url}/prism/migrations
(Authorization: токен спец. пользователя с ролью ROLE_prism_executor@{application})
```

В ответ ожидается информация в формате multipart/form-data:

"db": строка подключения из параметра {application}/default/spring.datasource.url (платформа сравнивает полученное значение со значением из параметров из платформы и выдаст ошибку миграции при несоответствии)

"migrations": sql файлы с миграциями схемы БД

"migrations-dict": sql файлы с миграциями схемы БД общих справочников приложений

"config": json файлы группами, ролями и политиками

"client-config": yaml файлы с параметрами приложений

"extensions": строка с sql скриптом создания настраиваемых полей в таблицах

Обязательным является только значение "db".

Результат миграции положительный, если нет новых изменений или миграция обработана автоматически (миграции обрабатывается автоматически, если параметр prism.migration.auto=true).

После обработки миграции платформа вызывает инициализацию приложения:

```
POST {application_url}/prism/init
(Authorization: токен спец. пользователя с ролью ROLE_prism_executor@{application})
Тело запроса: true | false (результат миграции)
```

Если приложение получило отрицательный результат миграции, то оно должно начать отдавать на все приходящие запросы http code = 503 Service unavailable и ожидать выполнения ручной миграции администратором платформы.

Если приложение получило положительный результат миграции, то оно запускается в стандартном режиме.

После выполнения администратором миграции через интерфейс платформы, платформа повторит все пункты, начиная с получения миграций.

При выполнении миграции платформа создаст:

- процедуры для работы с группами, ролями и политиками
- таблицу для очередей задач prism_queue_tasks

4.2.1. Формат файлов миграции схемы

Для работы с миграциями схемы БД платформа использует библиотеку flyway. Пример файлов:

Имя файла: V1__init_schema.sql

Содержание:

```
create table table1 (
    id int8 primary key,
    val text
);
```

Имя файла: V2__add_column_archive_to_table1.sql

Содержание:

```
alter table table1 add archive bool;
```

Имя файла: R__procedure1.sql

Содержание:

```
create or replace procedure procedure1
language plpgsql
as $
begin
    null;
end
$;
```

4.2.2. Формат файлов миграции групп, ролей, политик

Имя файла: role.user.json

Содержание:

```
{
 "name": "user",
 "schemaPermissions": [
  "usage"
 ],
 "description": "Пользователь",
 "grants": [
  {
   "type": "table",
   "relation": " table1",
   "permissions": [
    "select",
    "update",
    "usage"
```

```
  ]
 }
}
```

Имя файла: group.user.json

Содержание:

```
{
 "key": "user",
 "name": "Пользователи",
 "roles": [
  "user"
 ]
}
```

Имя файла: rls.json

Содержание:

```
[
 {
  "key": "table1_user",
  "tableName": "table1",
  "predicate": "archive = false",
  "readRoles": [],
  "readWriteRoles": ["user"]
 }
]
```

4.2.3. Формат файлов стандартных параметров конфигурации

Имя файла: {application}.yaml

Содержание:

```
custom:
  app:
    config: configuration value
    config$: Параметр, необходимый для настройки приложения
```

Имя файла: {application}-business-vals.yaml

Содержание:

```
business:
  task:
    value: business rule
    config$: Параметр, необходимый для выполнения задачи бизнеса
```

4.3. Интеграция с приложениями

4.3.1. Интеграция с приложениями Призмы через REST API

Для обращения к REST API приложений Призмы 2.0 с применением Basic-авторизации, необходимо использовать API, предоставляемое модулем «Платформа». Для этого требуется пользователь, созданный средствами централизованного управления службой каталогов (ALD PRO или любой другой аналогичной службой. Подробнее см. подраздел 3.1 раздела «Управление доступом»), либо уже существующий в каталоге. http-запрос, отправляемый в модуль «Платформа», должен быть сформирован следующим образом:

http(s):// host_name /api/admin/basic-auth/application_name/url ,

где:

- host_name – имя host,
- application_name – имя приложения,
- url – url целевого приложения
- Также должен присутствовать http-заголовок Authorization: Basic (key),
- где key - (Base64(логин пользователя:пароль))

После отправки запроса, модуль «Платформа» осуществит попытку выпустить kerberos-ticket, и с ним выполнить запрос у целевого приложения:

http(s)://domain_name/api/application_name/url

4.3.2. Интеграция посредством загрузки файлов

В модуле реализовала возможность интеграции посредством загрузки файлов. Для этого необходимо создать коннектор, имеющий определенные параметры (Рисунок 1).

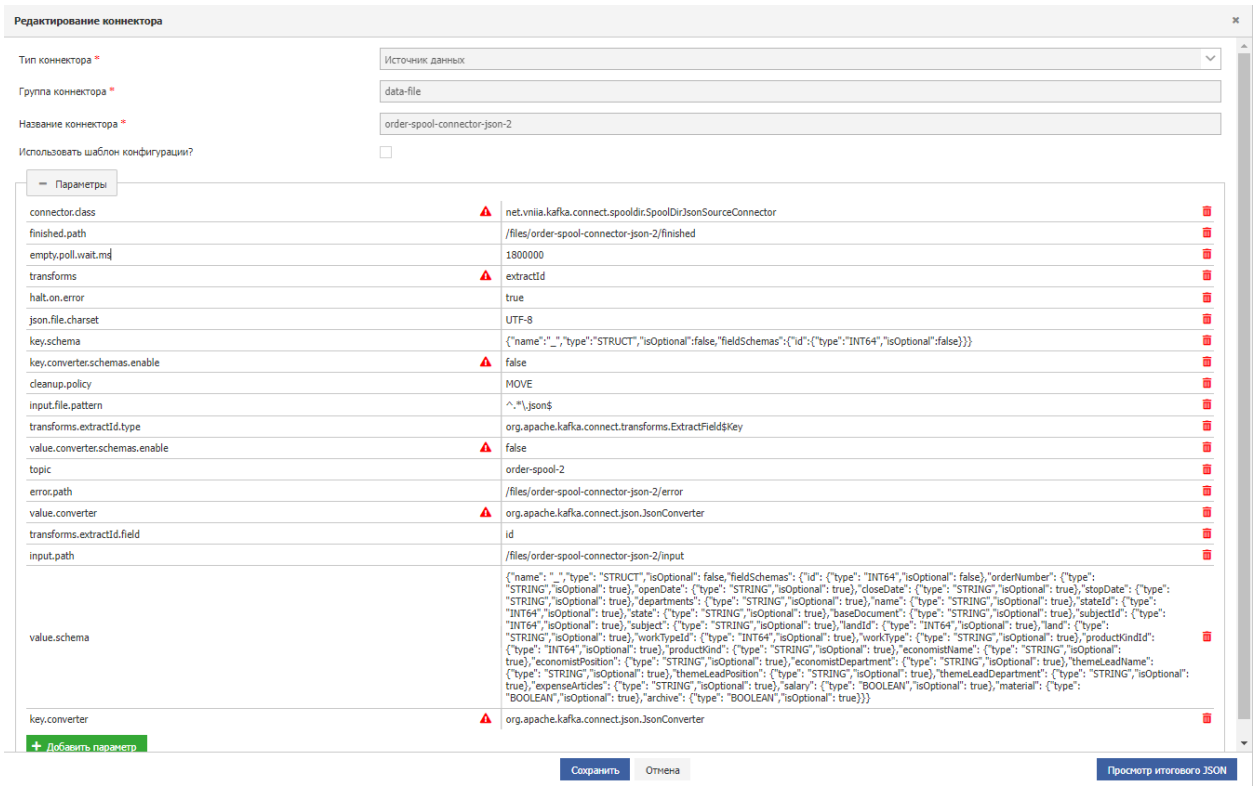

Рисунок 1 – Интеграция посредством загрузки файлов

В коннекторе реализован функционал редактирования json-формата (Рисунок 2).

Рисунок 2 – Интеграция посредством json-файла

Где:

- connector.class – определяет класс коннектора
- finished.path – путь директории успешно загруженного файла
- empty.poll.wait.ms – время обновления коннектора(в милисекундах)
- transforms – определяет список преобразований
- halt.on.error – определяет, должна ли задача остановиться при обнаружении ошибки или перейли к следующему файлу (true/false)
- json.file.charset – стандарт кодирования символов для чтения файла
- key.schema – схема ключа, записанного в Kafka
- key.converter.schemas.enable – определяет необходимость добавления описания схемы данных в ключ
- cleanup.policy – определяет, как соединитель должен очищать файлы, которые успешно обработаны
- input.file.pattern - регламентирует расширение файлов, с которыми осуществляется работа данного коннектора.
- transforms.extractId.type – тип преобразования для извлечения строки или числа
- value.converter.schemas.enable – определяет необходимость добавления описания схемы данных в тело сообщения
- topic – содержит имя топика, в который будет записано сообщение с информацией из файла
- error.path – папка для перемещения файлов, при загрузке которых произошла ошибка
- value.converter – ковертер для преобразования тела сообщения
- transforms.extractId.field – имя преобразования и поле, к которому оно относится
- input.path – путь директории для размещения файлов и дальнейшей загрузки
- value.schema – схема для значения, записанного в Kafka
- key.converter - конвертер для преобразования ключа сообщения

4.4.Отправка почтовых уведомлений

Для отправки почты из приложений, необходимо использовать API, предоставляемый модулем «Платформа». При этом пользователю приложения, использующего такой API, должна быть назначена группа, содержащая роль «prism_mail_queue». Отправка сообщений в модуле «Платформа» реализована по протоколу SMTP с использованием механизма очередей.

Для возможности использования почтового сервиса необходимо настроить следующие параметры в модуле «Платформа» (Рисунок 3):

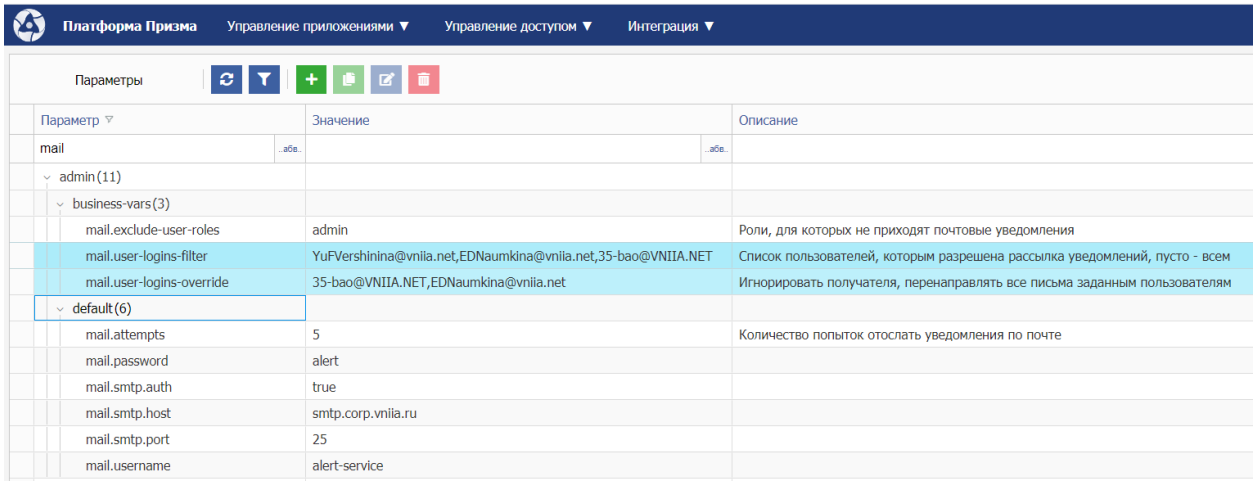

Рисунок 3 – Параметры настройки почтового сервиса

Параметры группы «business-vars» по умолчанию не должны иметь значений, используются для тестирования

mail.exclude-user-roles – список ролей, для которых не приходят почтовые уведомления

mail.user-logins-filter - список пользователей, которым разрешена рассылка уведомлений. Если параметр не заполнен, то рассылка уведомлений разрешена всем пользователям

mail.user-logins-override – список пользователей, которым будут перенаправлены все письма, отправленные адресатам

Параметры группы «default» необходимо настроить для корректной работы с почтовым сервисом.

mail.attempts - количество попыток отослать уведомления по почте

mail.password – пароль пользователя учетной записи, которой разрешено делать рассылку на стороне почтового сервера

mail.smtp.auth – авторизация на SMTP-сервере (true/false)

mail.smtp.host – хост SMTP-сервера

mail.smtp.port – порт SMTP-сервера

mail.username – имя пользователя учетной записи, которой разрешено делать рассылку на стороне почтового сервера.

Для отправки почтового сообщения, приложение должно вставить строку в таблицу prism_queue_tasks и информировать платформу о новой задаче:

```
POST {platform_url}/applications/{application}/queues/{name}/wake
(Authorization: токен пользователя)
```

Пример payload:

```
{
   "subject": "Заголовок письма",
   "message": "Содержание письма",
   "recipientPersonIds": [],
   "recipientRoles": ["user"]
}
```

Письма можно рассылать либо списку определенный пользователей (recipientPersonIds), либо пользователям с любой из заданных ролей (recipientRoles).

4.5.Внешний механизм корректировки и замены шаблонов документов

Для переопределения отчетных форм, достаточно положить папку reports рядом с нужным приложением.
Так как АСУДП «Призма» разворачивается преимущество в среде docker swarm, необходимо подкладывать файлы отчетов как отдельные volume с подключенным сетевым диском
Для этого в папке инсталятора призмы выполните команду:

```
./install.{environment}.sh extract-reports
```

Для приложений определенных в переменной PRISM2_APPLICATIONS, будет произведен поиск отчетов в их образах (папка ./images),
после чего рядом с настройками для конкретного приложения (compose files) появится папка reports:
./setup/{application}/reports

Пример подключения сетевого диска с использованием пакета программ Samba и протокола CIFS из пакета установки Astra Linux
В ОС Astra Linux установите пакет fly-admin-samba следующей командой:

```
sudo apt install fly-admin-samba
```

Создайте директорию

```
sudo mkdir /prism2_files
```

Отредактируйте конфигурационный файл
sudo nano /etc/samba/smb.conf

Добавив в его конец следующую конфигурацию:

```
[prism2_files]
  available = yes
  comment = prism2 files
  browseable = yes
  case sensitive = yes
  ea support = yes
  fstype = Samba
  path = /prism2_files
  writable = no
  smb encrypt = no
  read only = yes
```

```
#Доступно всем
guest ok = yes
#guest account = nobody
```

Запустите проверку настроек:

```
testparm
```

после чего перезагрузите сервис samba:

```
sudo systemctl restart smbd
```

Скопируйте файлы в созданную сетевую папку командой:

```
mkdir -p /prism2_files/{application}/reports $folder && cp -r ./setup/{application}/reports/*
/prism2_files/{application}/reports
```

Добавьте настройки для нужно приложения в ваш основной compose файл и также смантируйте созданынй диск с помощью драйвера cifs

```
{application}-rest:
  volumes:
    - {application}_reports:/reports


volumes:
  {application}_reports:
    driver: local
    driver_opts:
      type: cifs
      device: //{HOST_IP}/prism2_files/{application}/reports
          o: "guest,sec=none"
          #или
          #o: "username=admin,password="*******" "
```

Для извлечения, корректировки и подмены файлов в приложениях АСУДП «Призма», требуется осуществить вышестоящие шаги, при этом заменить в командах «reports» на «files».

5. Аварийные ситуации

5.1.Действия при отказах программных или технических средств

Проверка журналов операционной системы на наличие ошибок

- dmesg
- journalctl
- cat /var/log/syslog
- cat /var/log/messages

Анализ системных журналов на предмет выявления ошибок или нестандартного поведения операционной системы.

Проверка наличия свободного места и корректности точек монтирования

cat /etc/fstab

df -h

Потребуется сравнить файл /etc/fstab с текущим состоянием примонтированных устройств. Возможно возникновение ситуации, когда какое-то устройство не было примонтировано и система перешла в read only режим.

Проверка целостности файловой системы

fsck

Утилита для восстановления поврежденной файловой системы, например, при резком обрыве питания.

Для более подробной информации об ошибках ОС, диагностике и методах решения следует обратиться к документации AstraLinux: https://wiki.astralinux.ru

5.2.Действия по восстановлению программного обеспечения системы.

В случае возникновения аварийных ситуаций с Docker.

Проверка статуса службы docker

sudo systemctl status docker

Проверим, запущена ли служба docker. Если она запущена и корректно работает, при выводе команды должен быть статус службы Active: active (running).

Вывод списка контейнеров.

docker ps -a

Проверим список контейнеров, запущенных на хосте. Если status контейнера Up – он запущен и работает. Также можно обратить внимание на порты, которые он использует (столбец PORTS) и CONTAINER ID

Просмотр лог-файлов выбранного контейнера

docker logs <container id>

Выведем лог-файлы определенного контейнера, используя CONTAINER ID, полученный из предыдущей команды.

Использование утилиты просмотра системного журнала (для выбранного сервиса docker):

sudo journalctl -u docker

Анализ системного журнала сервиса docker на предмет выявления ошибок.

Перезапуск docker

sudo service docker restart

или *sudo systemctl restart docker*

Требуется:

Убедиться, что достаточно ресурсов для работы docker

Проверить правильность конфигурации docker

Для более подробной информации об ошибках, диагностике и методах решения следует обратиться к документации Docker: https://docs.docker.com

5.3.Действия по восстановлению данных.

Восстановление базы данных зависит от выбранного метода резервного копирования и степени повреждения данных.

5.3.1. Логическое восстановление

Полное логическое восстановление всех баз данных, сделанных утилитой pg_dumpall:

psql -f db.out postgres

Восстановление базы данных из текстового скрипта, сделанного утилитой pg_dump, в чистую базу newdb:

psql -d newdb -f db.sql

Восстановление из архива в чистую новую базу данных newdb (утилита pg_restore применяется для восстановления резервной копии, сделанной pg_dump в любом из нетекстовых форматов):

pg_restore -d newdb db.dump

5.3.2. Физическое восстановление

В общем случае, восстановление из бинарной копии выглядит следующим образом:

- Восстановление из резервной копии файлов кластера базы данных в новый каталог $PGDATA
- Установка параметров восстановления в конфигурационных файлах
- Запуск сервера. Сервер запустится в режиме восстановления и начнёт считывать необходимые ему архивные WAL файлы.
- При необходимости восстановить базу на какой-то момент времени, необходимо будет указать соответствующую точку остановки.

Для более подробной информации о восстановлении из резеврных копий следует обратиться к документации Postgresql: https://www.postgresql.org/docs/ > Backup and Restore.

5.4.Действия в случаях обнаружения несанкционированного доступа к данным

В случае обнаружения несанкционированного доступа к АСУДП «Призма». Платформа необходимо предпринимать действия в соответствии с политикой безопасности предприятия.