

KSE Platform Neo

Руководство администратора

Содержание

1. Введение.....	5
2. О программном обеспечении.....	6
3. Общие концепции.....	7
3.1. Архитектура ПО.....	7
3.2. Межсервисное взаимодействие.....	7
3.2.1. Брокер сообщений.....	8
3.2.2. Сервис объектной модели.....	9
3.2.3. Драйвер Modbus.....	9
3.2.4. БД конфигураций.....	9
3.2.5. Веб-приложение.....	10
3.2.6. Веб-сервер.....	10
3.2.7. БД временных рядов.....	10
3.2.8. Тренды.....	10
3.2.9. Сервис событий.....	10
3.2.10. Отчеты.....	10
3.2.11. Скрипты.....	10
4. Установка и запуск.....	11
4.1. Технические требования.....	11
4.1.1. Требования к серверу.....	11
4.1.2. Требования для клиента (рабочие станции, гаджеты).....	11
4.2. Установка.....	11
4.3. Работа с прикладным проектом.....	14
4.4. Включение автоматического входа локального пользователя.....	15
4.5. Режим киоска.....	17
5. Приложение 1. Текст файла docker-compose.yml.....	23
6. Приложение 2. Текст файла install.sh.....	32

1. Соглашения и условные обозначения

Меню, названия диалоговых окон и их свойства, названия документов, ключевые слова.	Жирный шрифт
Команды, примеры программ.	<code>Runtime.exe</code>
Имена файлов и пути.	<i>Курсив</i>
Ссылка на документ. Если в скобках указан номер страницы - ссылка внутри документа. Если указан графический символ и наименование документа, документ следует искать по названию на жестком диске (по умолчанию Документация устанавливается сюда): C:\ProgramData\KSoft\Documentation.	ссылка
Информация обязательная для прочтения/выполнения.	

2. Список терминов и сокращений

Аларм	Звуковое или визуальное средство оповещения оператора о неполадках оборудования, отклонениях в ходе технологического процесса или нештатной ситуации, требующей вмешательства
АСУ ТП	Автоматизированная система управления технологическими процессами
БД	База данных
Микросервисная архитектура	Тип организации ПО, при котором функции большого приложения разделяются на маленькие независимые программные модули
ОС	Операционная система
Прикладной проект	Прикладное программное обеспечение, файлы конфигурации и графические элементы, созданные с использованием ПО
ПО	Программное обеспечение "KSE Platform Neo"
Тренд	Графическое представление изменения переменных значений технологических параметров
Доскег-контейнер	Стандартизированный, изолированный и портативный пакет программного обеспечения, который включает в себя все необходимое для запуска приложения, включая код, среду выполнения, системные инструменты, библиотеки и настройки

3. Введение

1. Настоящий документ предназначен для квалифицированных специалистов, отвечающих за внедрение, ввод в эксплуатацию и обслуживание программного обеспечения “KSE Platform Neo” (далее ПО).
2. Документ содержит сведения для установки/конфигурирования/настройки ПО.
3. ООО «К-СОФТ ИНЖИНИРИНГ» оставляет за собой право на внесение изменений в настоящий документ в любое время.
4. Вопросы по настоящему документу, а также запросы на техническую поддержку следует отправлять на электронный адрес: support@k-soft-spb.ru.

4. О программном обеспечении

ПО "KSE Platform Neo" предназначено для реализации решений в области управления технологическими процессами (АСУ ТП).

ПО предоставляет следующие возможности:

- Автоматический сбор данных;
- Визуализация процессов и состояния объектов контроля в графическом режиме;
- Отображение событий и алармов;
- Отображение трендов изменения параметров по объектам контроля;
- Долгосрочное хранение и агрегирование данных;
- Формирование и печать отчетов;
- Ролевая модель доступа к различным функциям прикладного проекта, определенных полномочиями каждого пользователя;
- API.

Основным средством представления является веб-приложение, которое реализуется в рамках разработки прикладного проекта.

5. Общие концепции

5.1. Архитектура ПО

ПО реализовано на принципах микросервисной архитектуры. Каждый сервис выполняет конкретную функцию и упакован в Docker-контейнер. В таблице ниже ([Таблица 1. Сервисы](#)) перечислено наименование всех сервисов (компонентов ПО), описание их назначения и доступные порты.

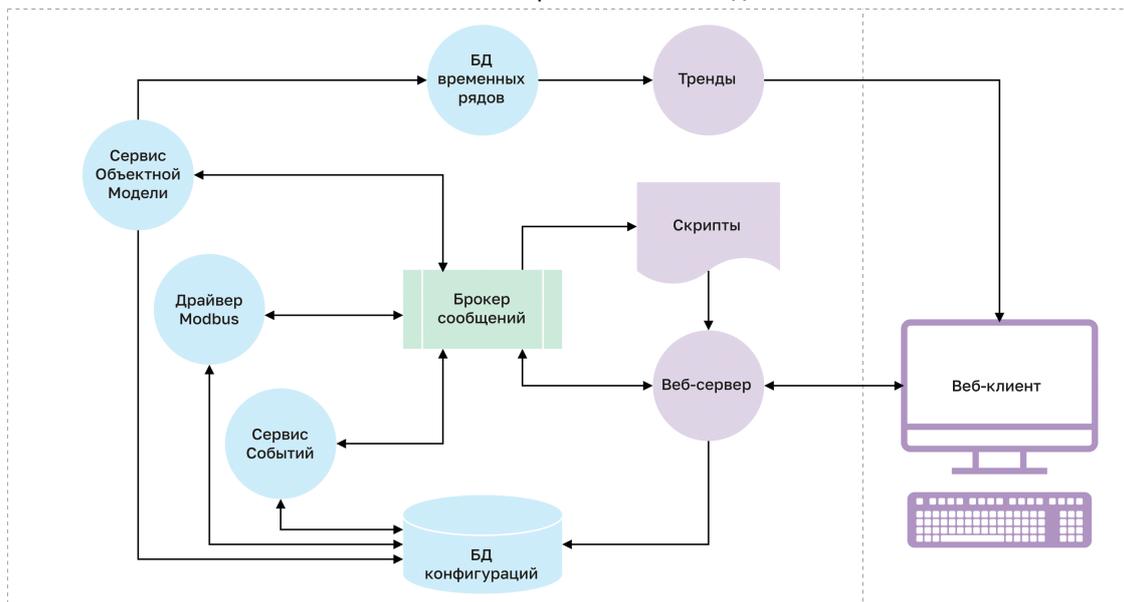
Табл. 1. Сервисы

Наименование	Описание	Порт подключения
ckline4frontend	Веб-приложение	80
object_model	Сервис хранения конфигурации прикладного проекта	5140, 7043
ksereports	Сервис формирования отчетов	5235
ksescheduler	Управление расписанием запуска сервисов	5234
agent-modbus-tcp	Сервис клиента Modbus TCP	5271-5272
socketgateway	Веб-сервер	4200
heatingeventservice	Сервис хранения и предоставления сообщений для веб-приложения	5200
postgres	Базы данных PostgreSQL	5432
rabbitmq	Программный брокер сообщений	15672, 5672
grafana	Платформа мониторинга и анализа данных, предназначена для сбора данных из различных источников и их дальнейшей обработки, например, отображения в виде графиков и диаграмм.	18080
keydb	Сервис распределенного кеша	6379
victoria_metrics	БД временных рядов	8428
vmagent	Агент для сбора метрик из различных источников	8429

5.2. Межсервисное взаимодействие

На рисунке ниже представлена схема межсервисного взаимодействия.

Рис. 1. Схема межсервисного взаимодействия



Ниже приведены описания основных компонентов.

5.2.1. Брокер сообщений

Связующее звено во взаимодействии сервисов между собой. Реализуется сервисом rabbitmq. Брокер сообщений отвечает за связь между отдельными сервисами и обеспечивает гарантированную доставку данных от одного сервиса другому. Каждый сервис может передать/получить данные, которые брокер помещает в определенную очередь. В таблице ниже приведено описание используемых очередей и их названия:

Табл. 2. Наименование очередей RabbitMq

Наименование	Описание
tag_value_updated	Агент “пушит” сообщения при изменении значения тега.
backend_frontend	Объектная модель “пушит” сообщения при изменении значения атрибута у свойства объекта типа ValueAttribute.
agent_tag_value_updated	Объектная модель “пушит” сообщения при изменении значения атрибута свойства провайдера типа TagAttribute.
system_alarm	Объектная модель “пушит” сообщения (системные уведомления). Например, при возникновении ошибки в процессе обработки сообщения из очереди tag_value_updated, объектная модель отправит туда сообщение.
dlx_queue	Сюда попадают сообщения, которые не могут быть обработаны, по какой-либо причине.
cache_warmup_stream_address	В эту очередь агент “пушит” сообщения для прогрева кеша.

5.2.2. Сервис объектной модели

Ядро разрабатываемых прикладных проектов, содержащее структуру объектов, составляющих проект, их свойства, атрибуты, операции, взаимосвязи между собой. Реализуется сервисом `object_model`, который обеспечивает доступ к объектной модели прикладного проекта.

Сущности объектной модели:

- Объект - цифровое отображение реального объекта, может состоять из других объектов.
- Свойства - описывают объект.
- Атрибут - конкретизирует или дополняют описание свойств объекта.
- Провайдер - связывает внешние данные, получаемые через агента, с объектной моделью.

Объектная модель предоставляет доступ к актуальным значениям свойств объекта.

5.2.3. Драйвер Modbus

Отвечает за обмен данными с различными устройствами по промышленным протоколам. Реализуется сервисом `agent-modbus-tcp`, который выступает в качестве клиента протокола Modbus TCP.

Сущности агента:

- Узел - позволяет указать адрес и порт сервера (хоста), а также дополнительные настройки подключения.
- Устройство - позволяет указать настройки подключения к физическому устройству, с которого считываются/записываются данные.
- Тег - описывает единичный объем данных, а также настройки по его считыванию и обработке.

5.2.4. БД конфигураций

Совокупность БД, в которых хранятся конфигурации для работы сервисов. Реализуется сервисом `postgres`, который представляет собой БД PostgreSQL. В таблице ниже приведено описание БД.

Табл. 3. Описание БД

Наименование	Описание
ObjectModel	Хранение конфигурации проекта
agent	Хранение конфигурации агента
heatingDB	Хранение конфигурации виджетов и выражений для веб-приложения
messages	Хранение сообщений
reports	Хранение конфигурации отчетов
schedules	Хранение конфигурации (расписания) запуска сервисов
zabbix	Хранение конфигурации для мониторинга за сервисами, ресурсами и т.д.

5.2.5. Веб-приложение

Веб-приложение, которое является частью разрабатываемого прикладного проекта. Реализуется сервисом `skline4frontend`, который представляет собой визуальную часть для систем управления электрообогревом. Описание работы с веб-приложением приведено в Руководстве пользователя.

5.2.6. Веб-сервер

Веб-сервер, который отвечает за взаимодействие веб-приложения с другими сервисами. Реализуется сервисом `socketgateway`.

5.2.7. БД временных рядов

База данных временных рядов реализуется сервисом `victoria_metrics`, который получает данные из объектной модели по мере их обновления и сохраняет их во временные ряды. Осуществляет долгосрочное хранение и агрегирование данных. Для сбора метрик используется сервис `vmagent`.

5.2.8. Тренды

Обеспечивает доступ и графическое представление изменения данных контролируемого технологического процесса. Реализуется сервисом `grafana`. Не участвует в межсервисном обмене, получает данные из БД временных рядов и отображает их в веб-приложении через встраиваемый компонент.

5.2.9. Сервис событий

Обеспечивает обработку поступающих сообщений и доступ к журнала сообщений. Реализуется сервисом `heatingeventservice`.

5.2.10. Отчеты

Формирование отчетов реализуется сервисом `ksereports`. Не участвует в межсервисном обмене, получает агрегированные данные из БД временных рядов и журнала событий.

5.2.11. Скрипты

Скрипты позволяют расширить функционал прикладного проекта, в том числе для изменения состояния графических элементов. Исполнение скриптов происходит на уровне сервиса `socketgateway`.

6. Установка и запуск

Данное руководство не включает в себя установку ОС, создание пользователей ОС, настройку их прав и т.д. Ниже приводятся инструкции непосредственно относящиеся к установке\конфигурированию\настройке ПО "KSE Platform Neo".

6.1. Технические требования



Прим.: При выборе оборудования следует придерживаться рекомендуемых в документе требований и учитывать особенности реализуемого проекта.

6.1.1. Требования к серверу

Требования к аппаратному обеспечению для сервера:

- Процессор: 8 ядер;
- Оперативная память: 16 Гб;
- Дисковый накопитель: SSD, не менее 1 Тб.

Требования к программному окружению для сервера:

- ОС: Astra Linux 1.7.x;
- Поддержка контейнеризации.

6.1.2. Требования для клиента (рабочие станции, гаджеты)

Требования к аппаратному обеспечению для клиента:

- Процессор: 2 ядра;
- Оперативная память: 8 Гб.

Требования к программному окружению для клиента:

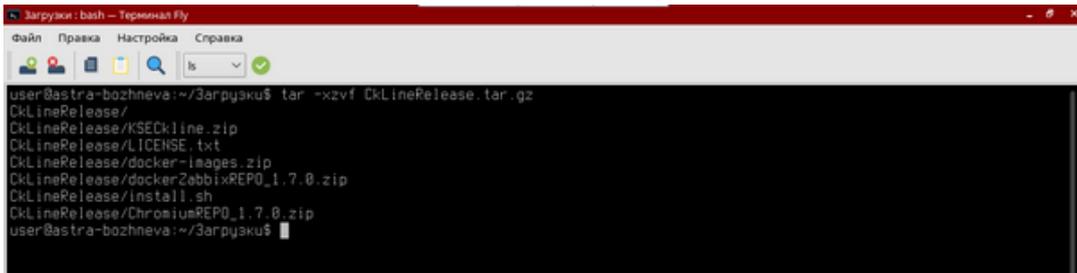
- ОС: Astra Linux 1.7.x;
- Наличие браузера с поддержкой JS и русской локализацией.

6.2. Установка

Для автоматической установки всех необходимых компонентов и сервисов через терминал выполните следующее:

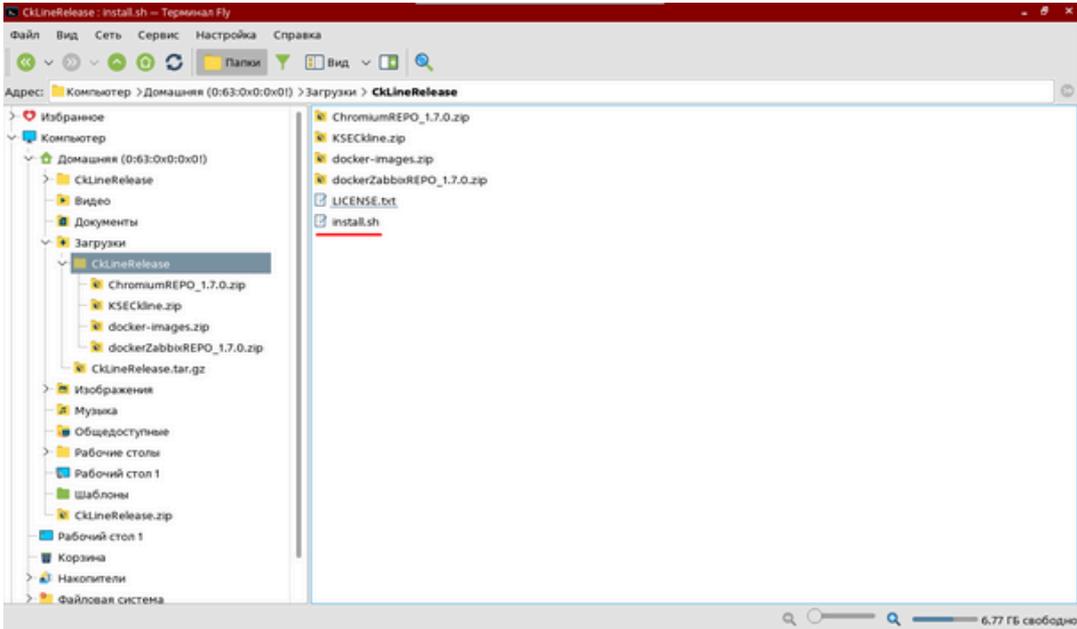
1. Скачайте инсталлятор `CkLineRelease.tar.gz`.
2. Для вызова терминала выберите: "Пуск" → "Системные" → "Терминал", после чего запустится Терминал Fly / "Пуск" → "Менеджер файлов" → вкладка "Сервис" → "Открыть терминал".
3. Распакуйте инсталлятор командой `tar -xzf CkLineRelease.tar.gz` (обычным пользователем, не root, без sudo):

Рис. 2. Распаковка инсталлятора



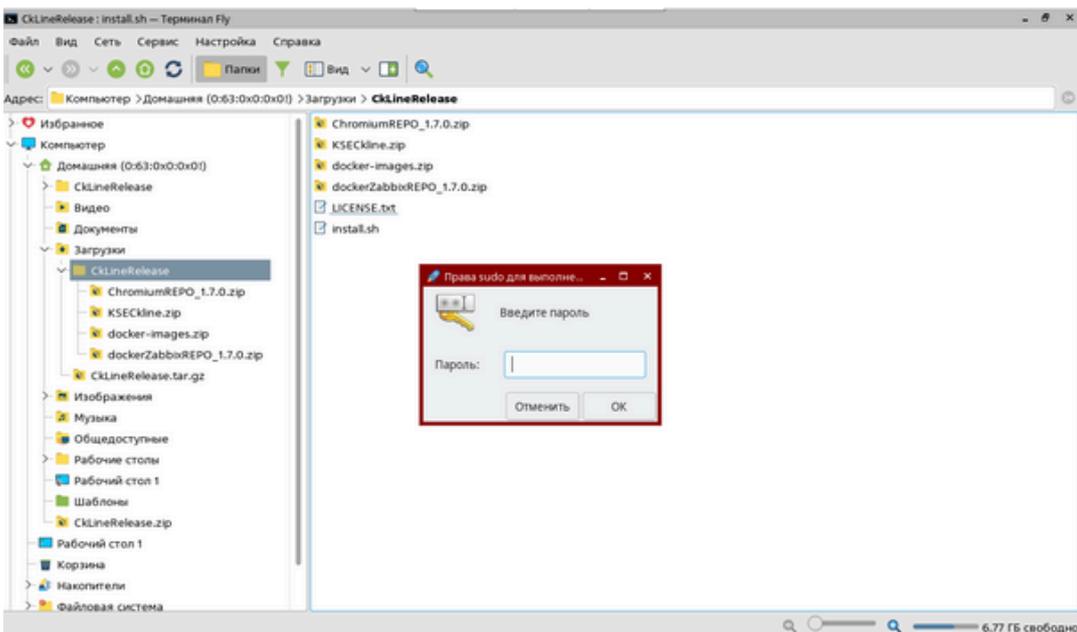
4. Для запуска процесса установки в Менеджере файлов дважды кликните на файл `install.sh`:

Рис. 3. Запуск процесса установки



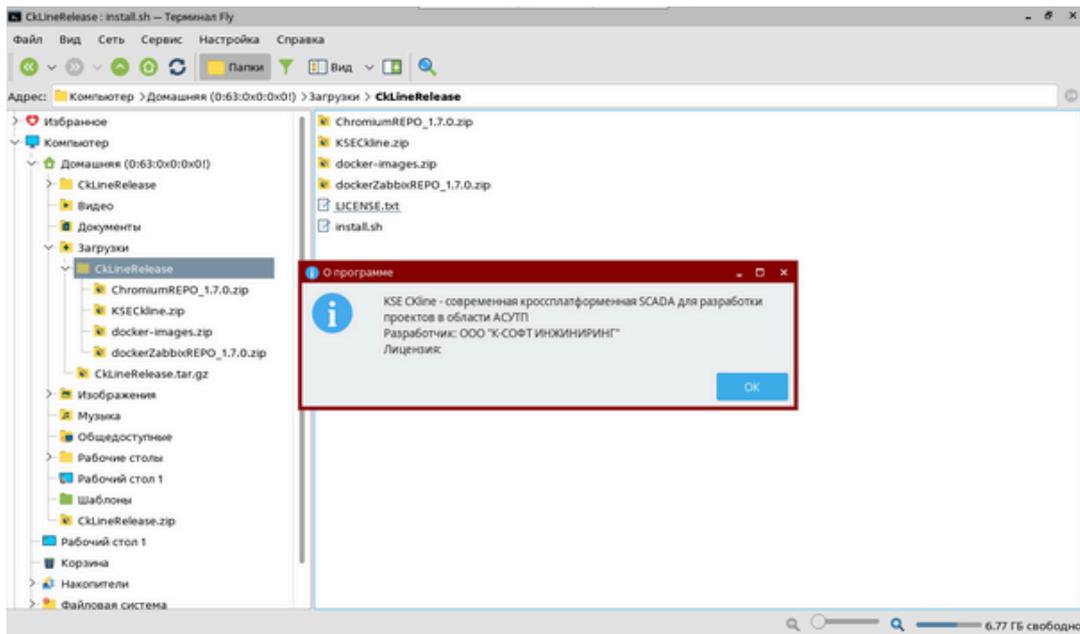
5. Введите пароль:

Рис. 4. Окно ввода пароля



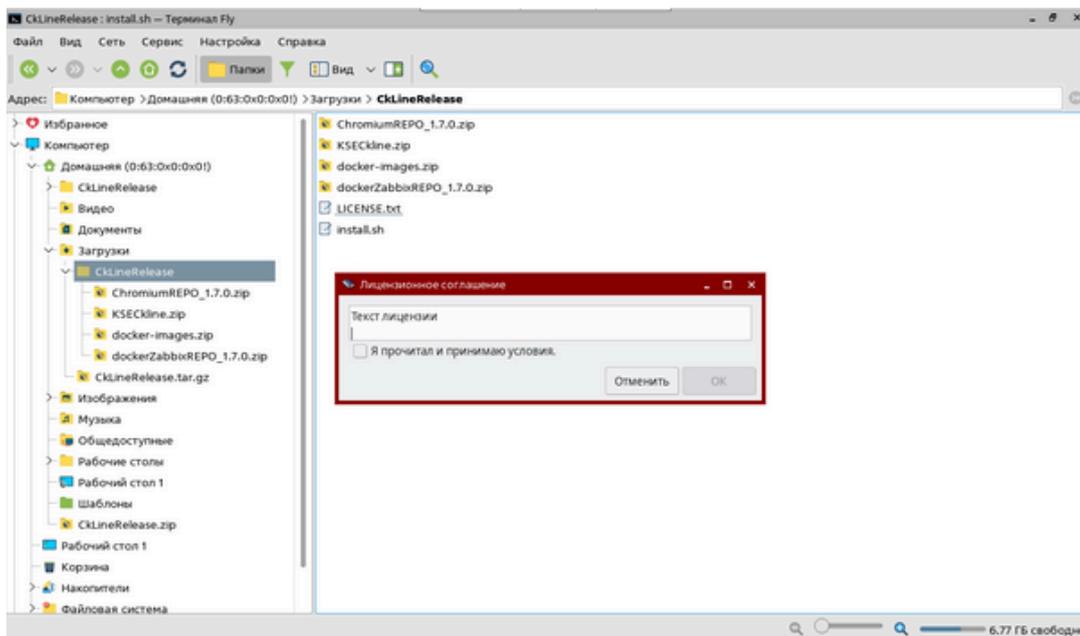
6. В окне “О программе” нажмите ОК, для продолжения установки:

Рис. 5. Окно информации о программе



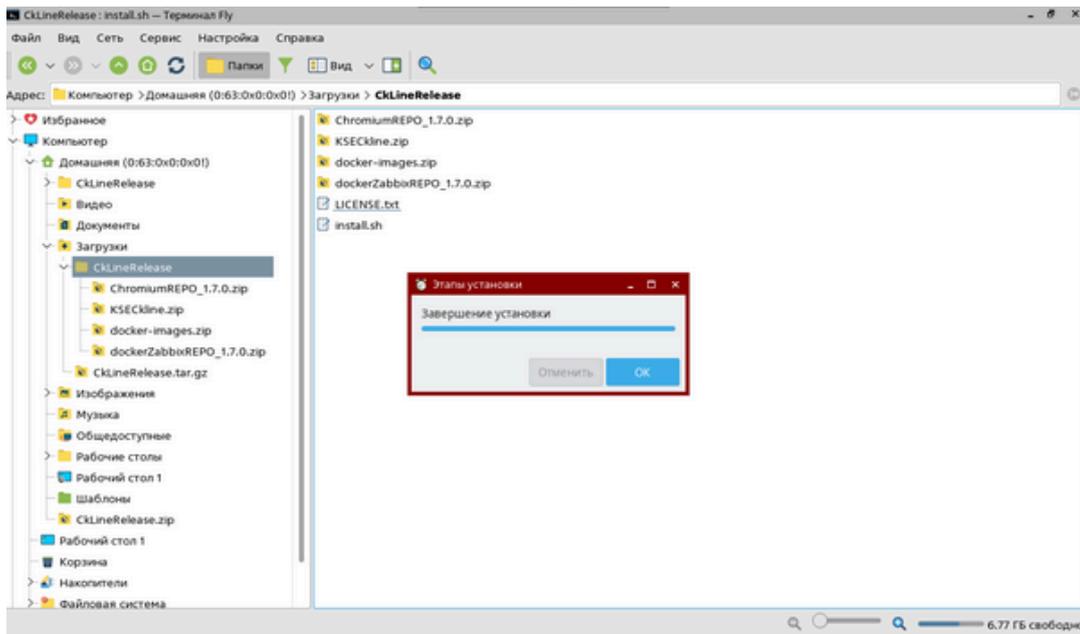
7. В окне “Лицензионное соглашение” ознакомьтесь с лицензионным соглашением и примите его условия, установив флаг в соответствующем поле, нажмите ОК, для продолжения процесса установки:

Рис. 6. Окно лицензионного соглашения



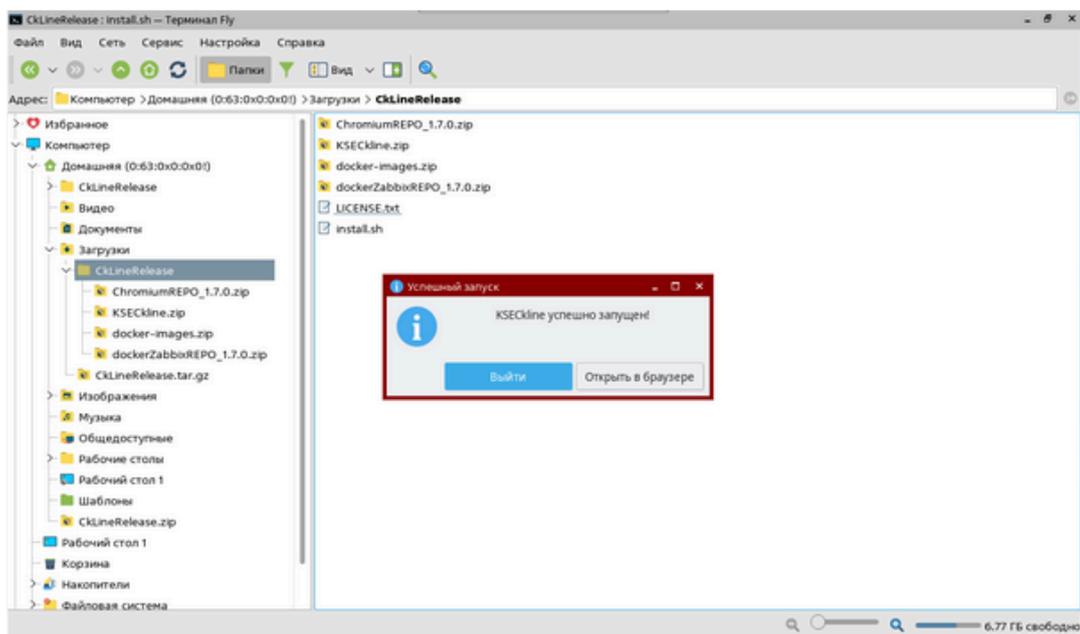
8. В окне “Этапы установки” нажмите ОК для завершения процесса установки:

Рис. 7. Этапы установки



9. В результате ранее выполненных пунктов и успешной установки, должно отобразиться соответствующее сообщение:

Рис. 8. Сообщение об успешной установке



10. Для входа из установки - нажмите кнопку "Выйти".

6.3. Работа с прикладным проектом

Разработка и внедрение прикладного проекта осуществляется компанией-интегратором. При необходимости внести изменения в текущий проект по каким-либо причинам, следует обратиться к компании-интегратору, ранее выполнявшим работы по внедрению данного проекта.

Данное руководство не включает в себя действия необходимые по разработке\внедрению\изменению\запуску прикладного проекта.

Веб-приложение прикладного проекта доступно по следующему сетевому адресу <http://xxx.xxx.xxx.xxx:port>, где:

- xxx.xxx.xxx.xxx - IP адрес сервера в вашей локальной сети;
- port - номер порта, по умолчанию 8088.

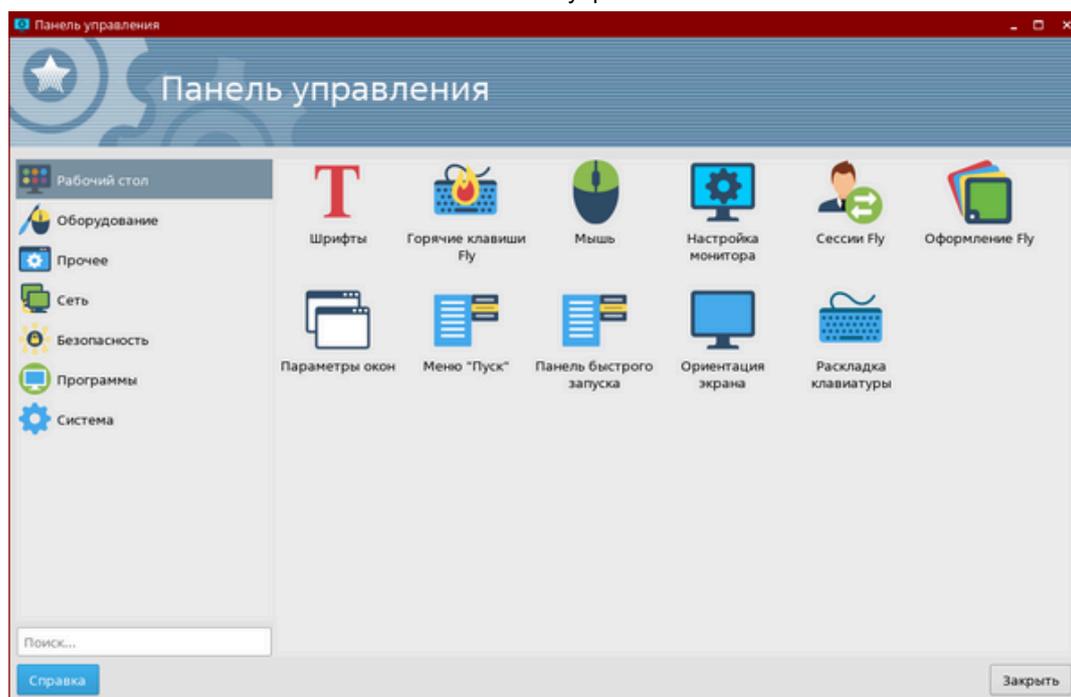
6.4. Включение автоматического входа локального пользователя

Данная настройка позволяет настроить автоматический вход, например, если нужно чтобы вход в ОС был осуществлен под локальным пользователем "Оператор". Далее локальному пользователю "Оператор" можно настроить режим киоска (см. [Режим киоска](#)).

Выполните следующие настройки:

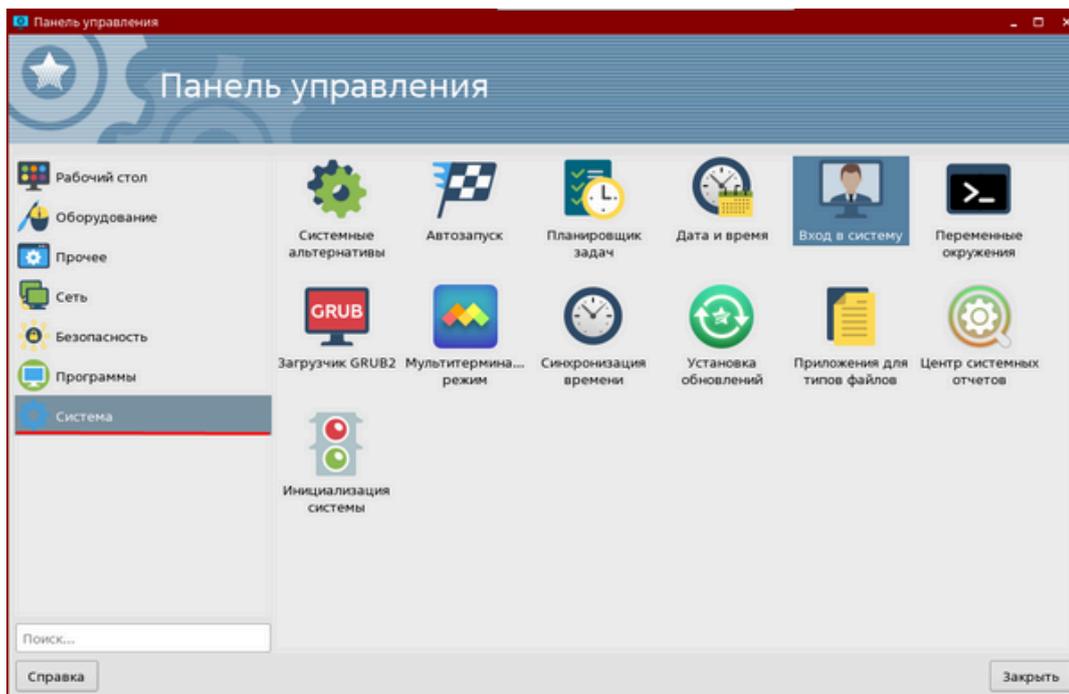
1. Выберите "Меню Пуск" → "Панель управления":

Рис. 9. Панель управления



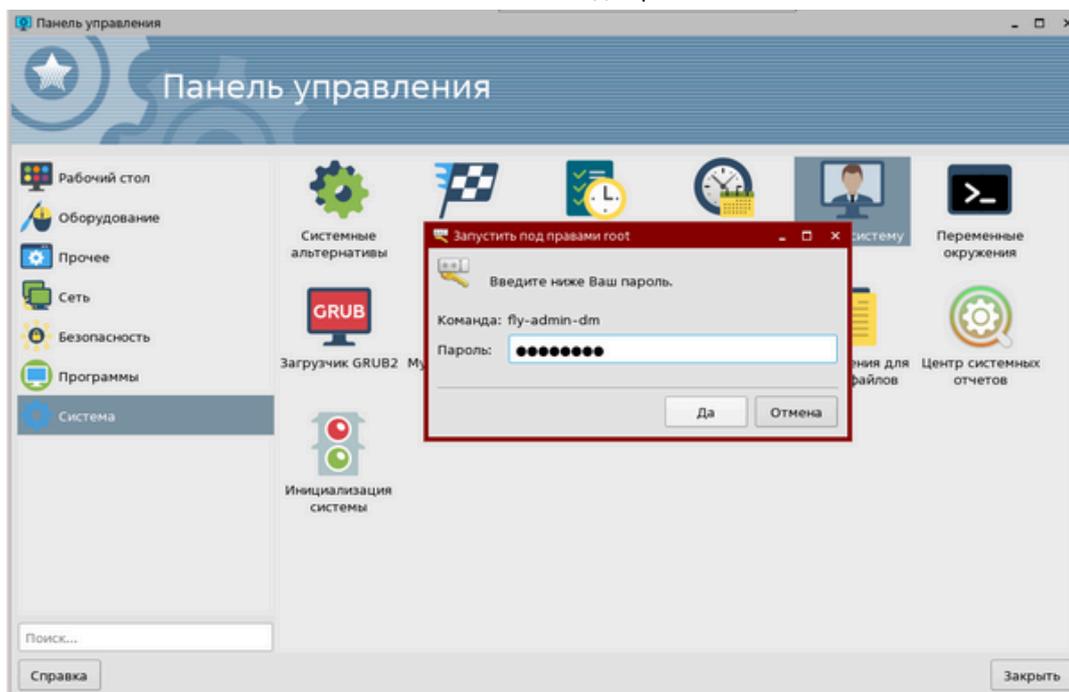
2. В панели слева выберите "Система":

Рис. 10. Вкладка "Система"



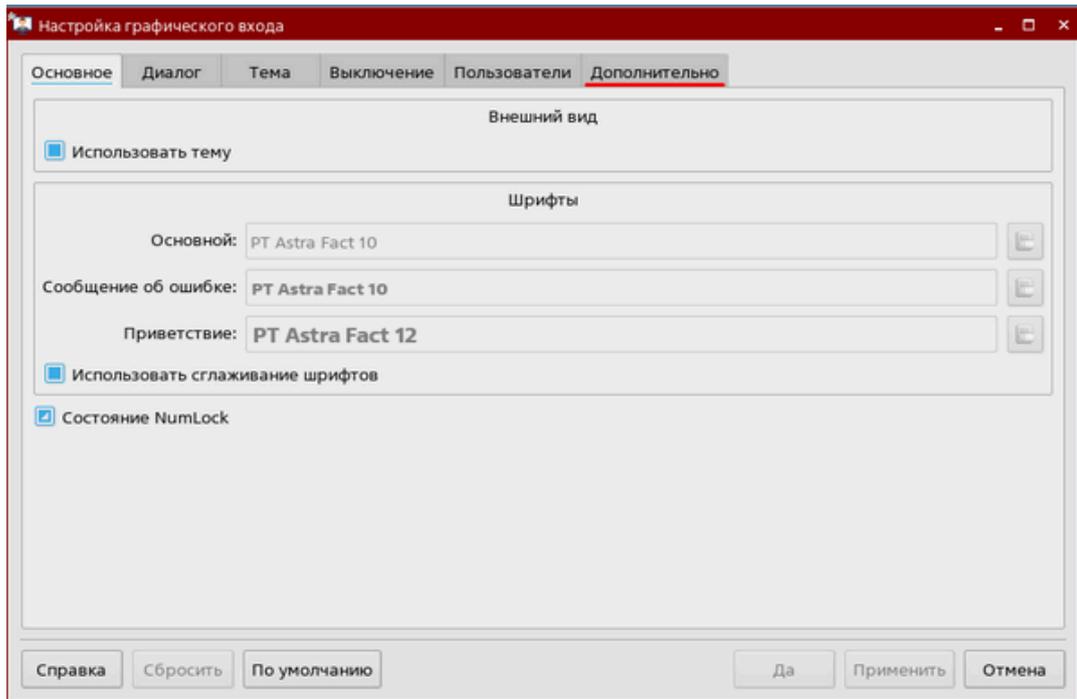
3. Выберите "Вход в систему" (см. [Рисунок 10. Вкладка "Система"](#)).
4. Введите пароль администратора ОС:

Рис. 11. Окно "Ввод пароля"



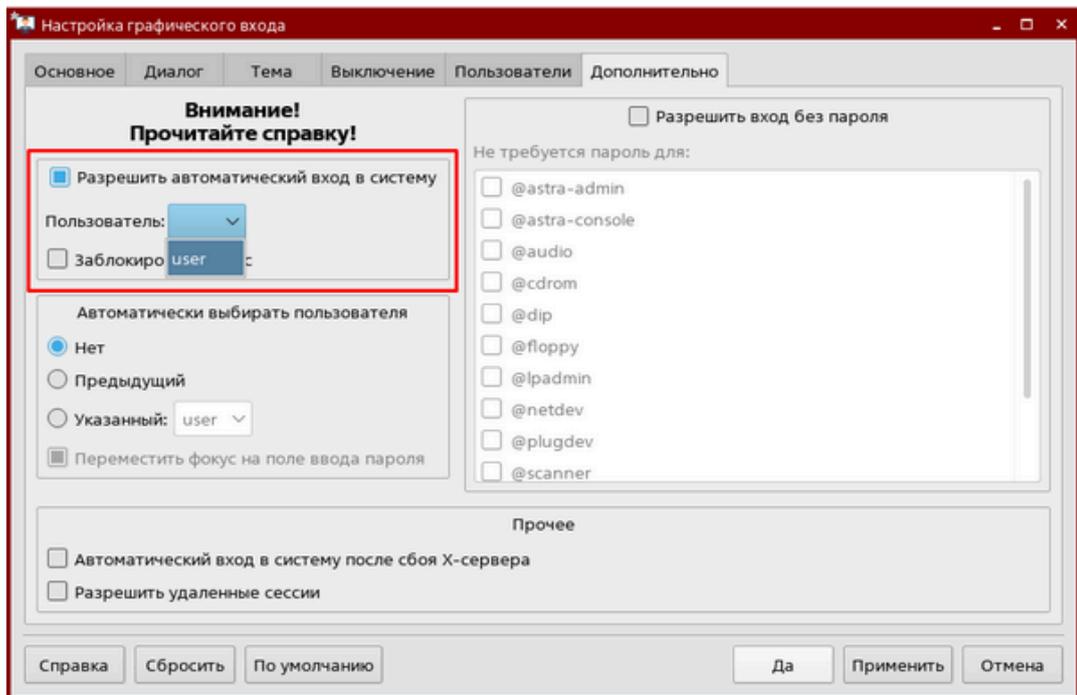
5. В открывшемся окне "Настройка графического входа" выберите вкладку "Дополнительно":

Рис. 12. Окно "Настройка графического входа"



6. Установите флаг "Разрешить автоматический вход в систему" и выберите локального пользователя:

Рис. 13. Окно "Настройка графического входа - Разрешить автоматический вход в систему"



7. Нажмите кнопку "Применить".

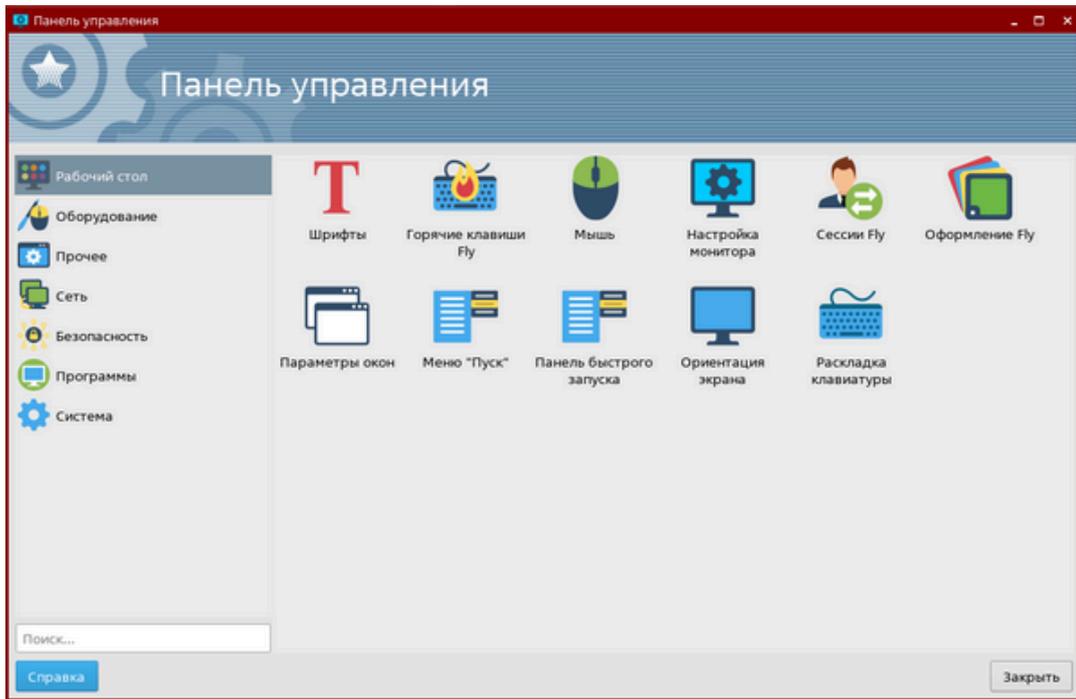
6.5. Режим киоска

При использовании графического киоска пользователю или группе пользователей разрешается запускать только приложения, явно указанные в их профиле. Графический киоск ограничивает доступ на уровне графической среды.

Настроить режим графического киоска можно с помощью графической утилиты "Политика безопасности". Для этого выполните следующие:

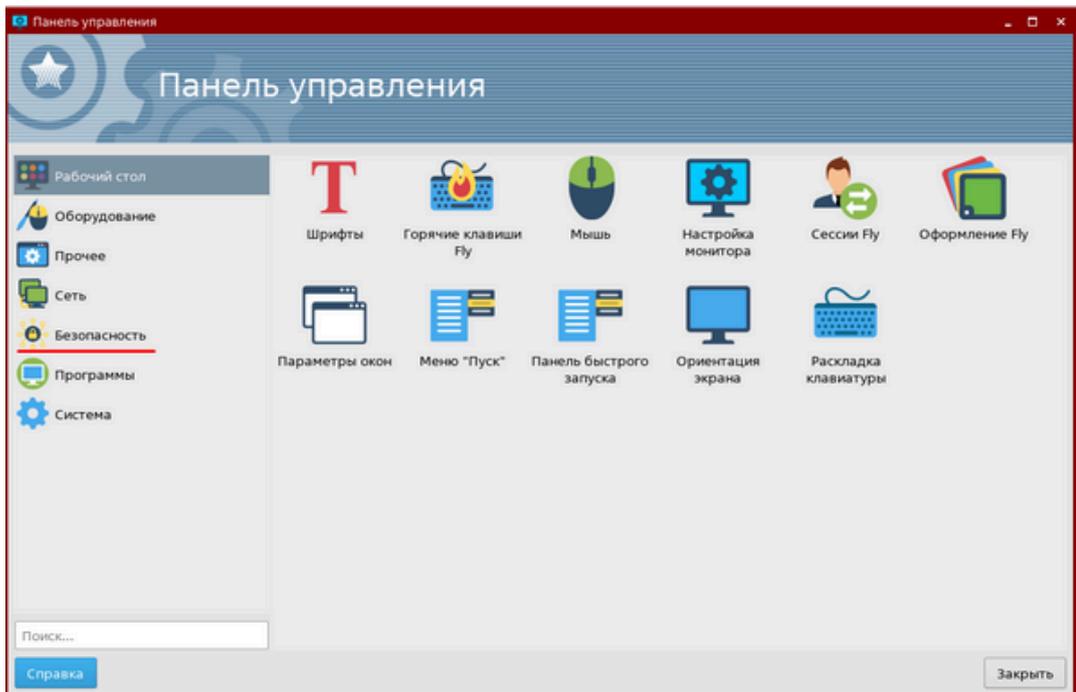
1. Выберите "Меню Пуск" → "Панель управления":

Рис. 14. Панель управления



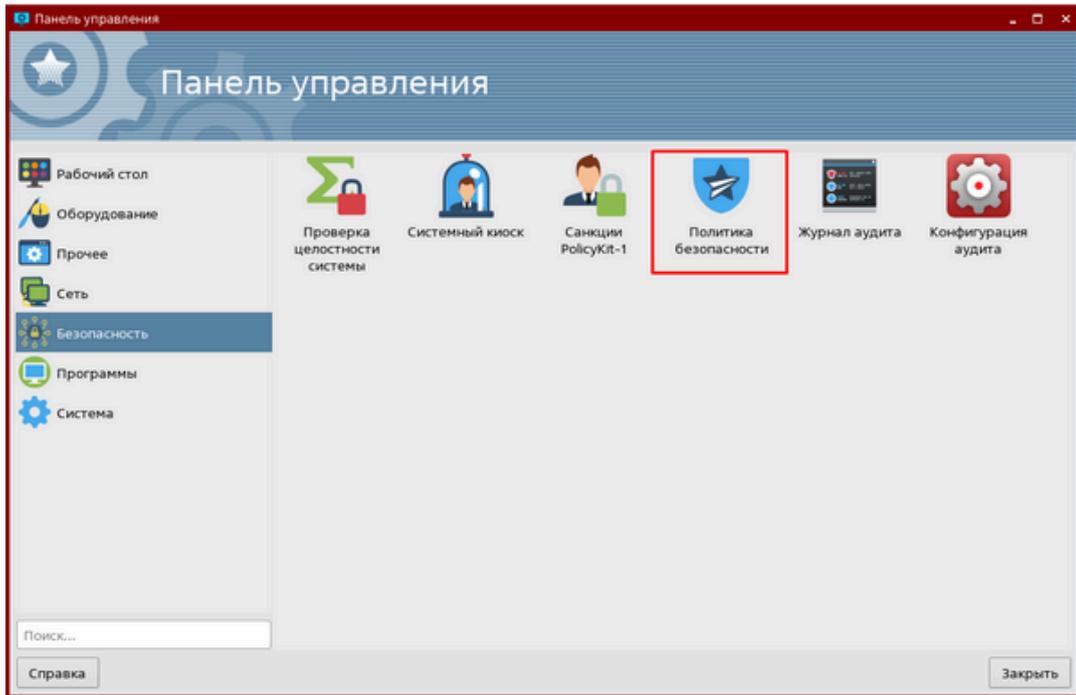
2. В панели слева выберите "Безопасность":

Рис. 15. Вкладка "Безопасность"



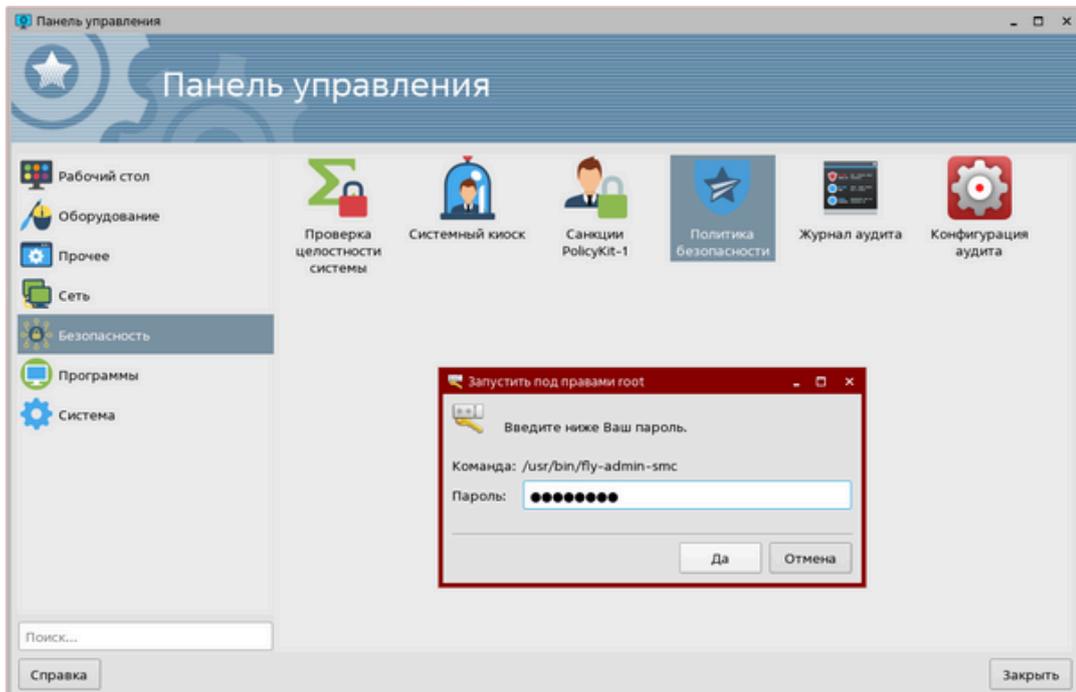
3. Выберите "Политика безопасности":

Рис. 16. Политика безопасности



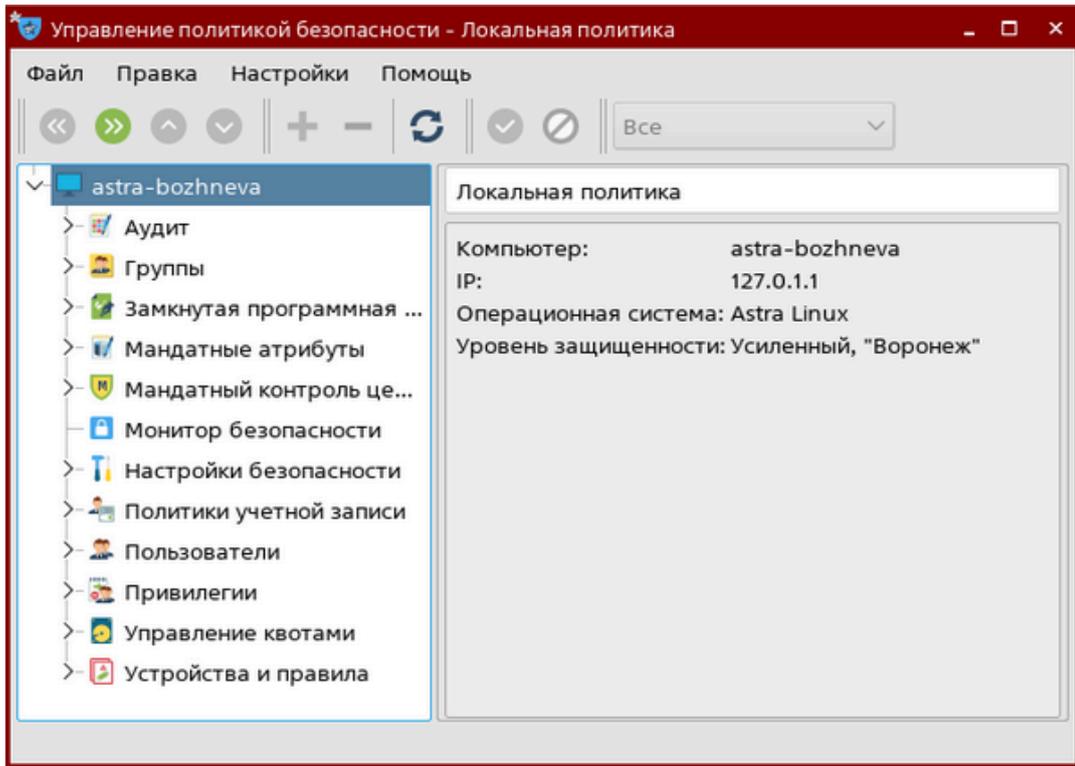
4. Введите пароль администратора ОС:

Рис. 17. Окно ввода пароля



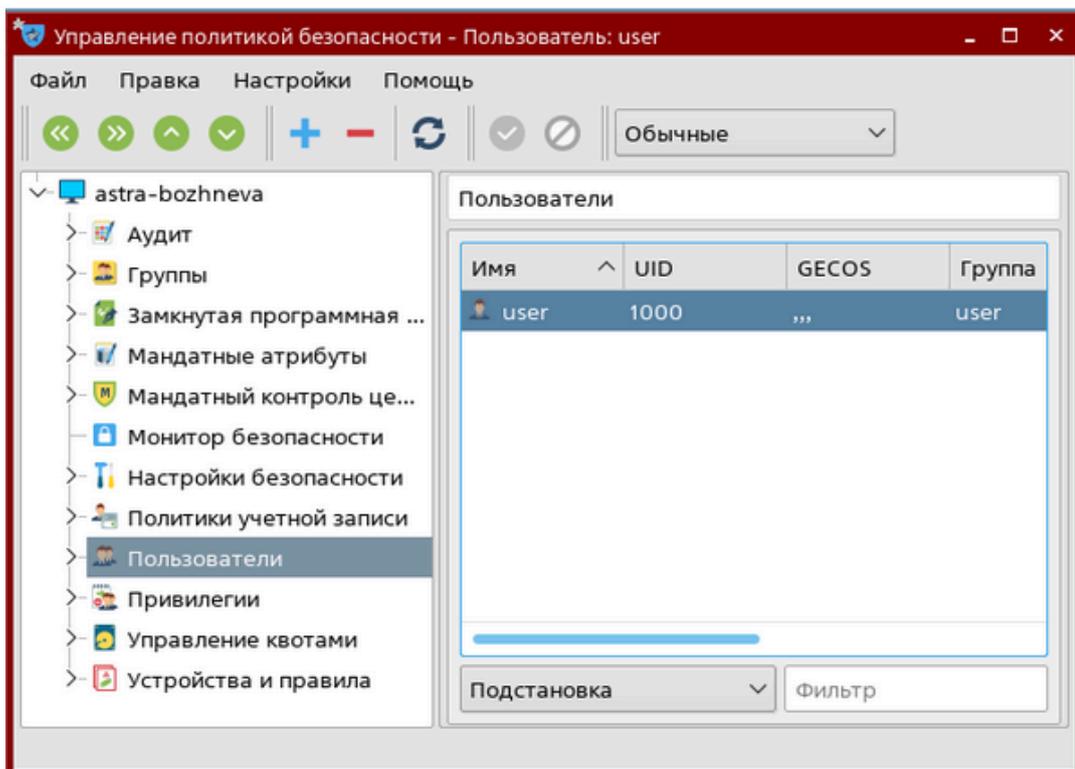
5. В результате должна завестись утилита "Политика безопасности":

Рис. 18. Окно утилиты Политика безопасности



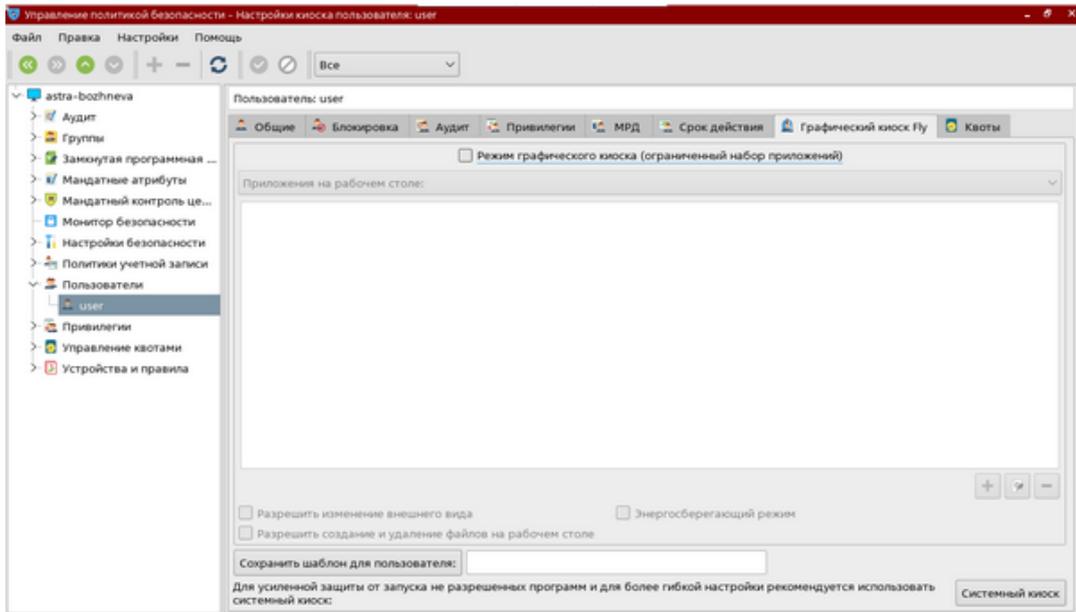
6. В дереве объектов выберите ветку "Пользователи":

Рис. 19. Ветка "Пользователи"



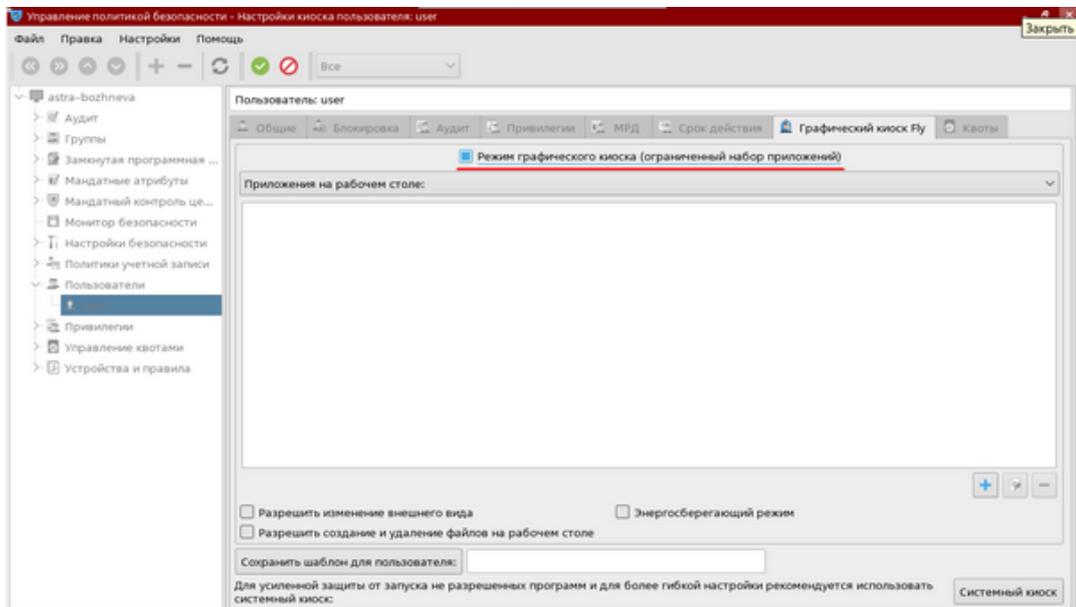
7. У выбранного пользователя перейдите на вкладку "Графический киоск Fly":

Рис. 20. Вкладка "Графический киоск Fly"



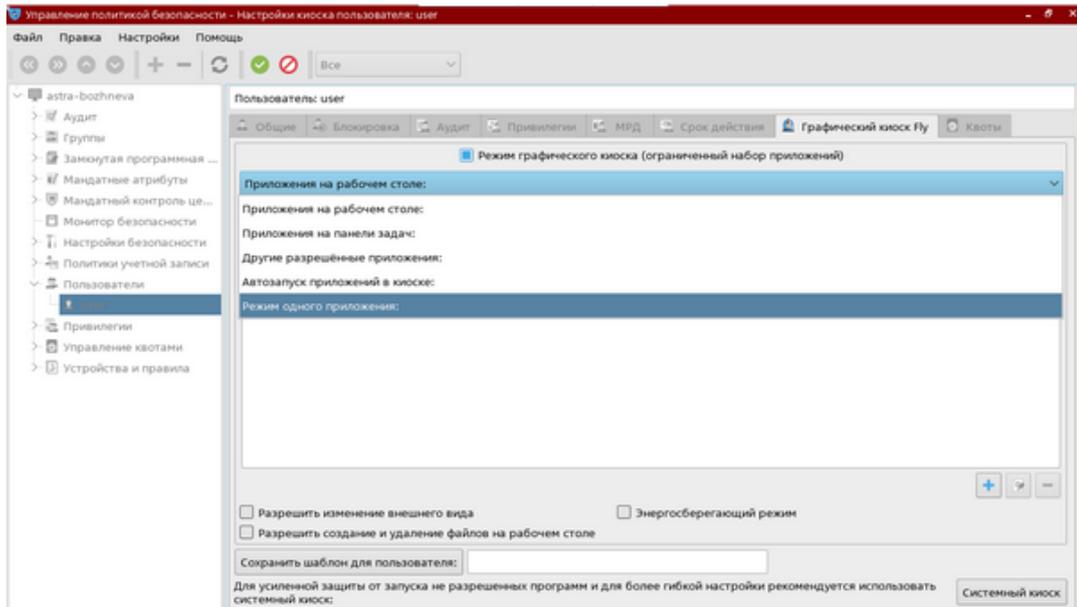
8. Поставьте флаг в поле "Режим графического киоска (ограниченный набор приложений):"

Рис. 21. Настройка режима графического киоска



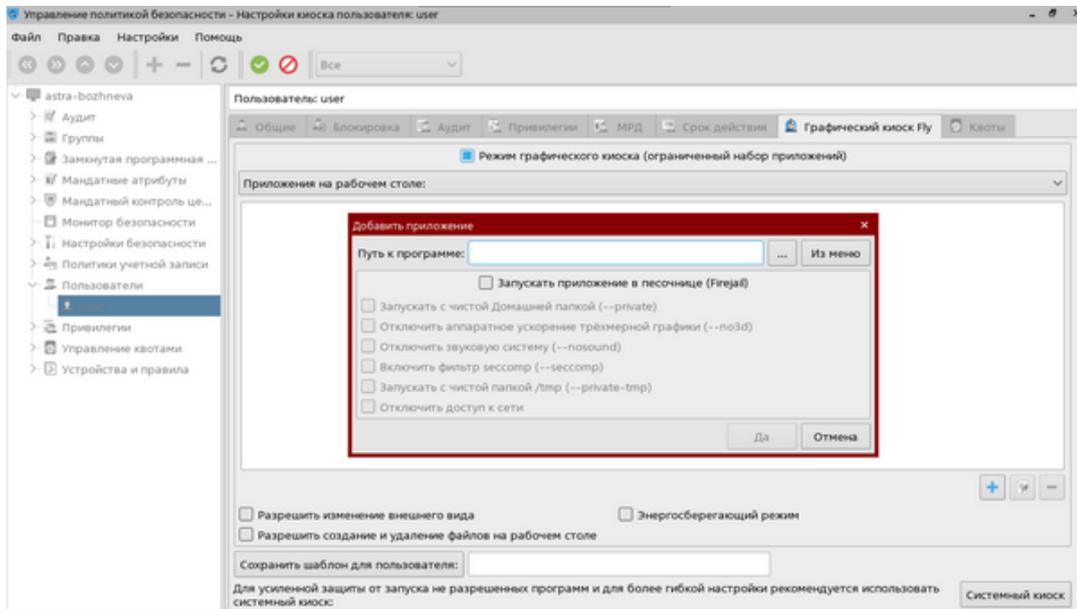
9. В выпадающем списке выберите "Режим одного приложения" и нажмите кнопку "+":

Рис. 22. Настройка режима графического киоска - Режим одного приложения



10. В открывшемся окне "Добавить приложение" укажите путь к приложению, которое надо запустить в режиме киоска:

Рис. 23. Окно "Добавить приложение"



11. В результате выполненных настроек, указанный локальный пользователь при загрузке ОС сразу будет видеть указанное приложение в режиме графического киоска.

7. Приложение 1. Текст файла docker-compose.yml

```
services:
  postgres:
    healthcheck:
      test: [ "CMD-SHELL", "pg_isready -U postgres" ]
      interval: 10s
      timeout: 5s
      retries: 5
    build:
      context: .
      dockerfile: Postgres_Dockerfile
    container_name: postgres
    pull_policy: always
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: heatingDB
      PG_DATA: /var/lib/postgresql/data
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./postgresql.conf:/etc/postgresql/postgresql.conf
    command:
      - postgres
      - -c
      - 'config_file=/etc/postgresql/postgresql.conf'
    networks:
      - my_network
    restart: always

  rabbitmq:
    image: rabbitmq:4.0.4-management
    container_name: rabbitmq
    pull_policy: always
    healthcheck:
      test: [ "CMD", "rabbitmq-diagnostics", "ping" ] # Проверяет, запущен ли RabbitMQ
      interval: 10s # Интервал между проверками
      timeout: 5s # Время ожидания ответа
      retries: 5 # Количество попыток
      start_period: 30s # Время ожидания до первой проверки (добавьте для долгих запусков)
    environment:
      RABBITMQ_DEFAULT_USER: user
      RABBITMQ_DEFAULT_PASS: user
    networks:
      - my_network
```

```
ports:
  - "15672:15672" # порт для веб-интерфейса управления
  - "5672:5672" # порт для стандартных AMQP соединений
volumes:
  - rabbit_data:/var/lib/rabbitmq/data
restart: always

socketgateway:
image: dergard/socketgateway:kozharov_2443
container_name: socketgateway
pull_policy: always
depends_on:
  postgres:
    condition: service_healthy
  rabbitmq:
    condition: service_healthy
networks:
  - my_network
environment:
  PORT: 4200
  FRONTEND_HOST: http://ckline4frontend
  FRONTEND_PORT: 5173
  DB_HOST: postgres
  DB_USER: postgres
  DB_PASSWORD: postgres
  DB_NAME: heatingDB
  OM_HOST: http://object_model
  OM_PORT: 5140
  RM_HOST: amqp://rabbitmq
  RM_PORT: 5672
  RM_DURABLE: "true"
  RM_AUTODELETE: "false"
  RM_USER: user
  RM_PWD: user
  RM_EXCHANGENAME: object_attribute_value_updated
  RM_QUEUEUENAME: backend_frontend
  BACKEND_HOST: http://socketgateway
  BACKEND_PORT: 4200
  JWT_SECRET: SuPeR_SeCrEt
  EXPIRATION: 1h
  PROJECT_NAME: УГБДФ. ШС1,3 #Подставить имя нужного проекта
command: >
  sh -c "while ! nc -z rabbitmq 5672; do
  echo 'Waiting for RabbitMQ...'; sleep 5;
  done;
  npm start"
ports:
```

```
- 4200:4200
restart: always

  heatingeventservice:
image: dergard/heatingeventservice:latest
container_name: heatingeventservice
pull_policy: always
depends_on:
  postgres:
    condition: service_healthy
  rabbitmq:
    condition: service_healthy
networks:
  - my_network
environment:
  PORT: 5200
  DB_HOST: postgres
  DB_USER: postgres
  DB_PASSWORD: postgres
  DB_NAME: messages
ports:
  - 5200:5200
restart: always

  prometheus:
image: prom/prometheus:v2.40.7
container_name: prometheus
pull_policy: always
ports:
  - "9090:9090"
volumes:
  - ./prometheus.yml:/etc/prometheus/prometheus.yml
  - prometheus_data:/prometheus
networks:
  - my_network
restart: always

  object_model:
image: dergard/objectmodel:latest
container_name: object_model
pull_policy: always
ports:
  - "5140:5140"
  - "7043:7043"
environment:
  - ASPNETCORE_ENVIRONMENT=Docker
```

```
- ConnectionStrings__EntityContext=Server=postgres;Port=5432;Database=ObjectModel;User
Id=postgres;Password=postgres;Pooling=true;Maximum Pool Size=1000
- Logging__LogLevel__Default=Information
- Logging__LogLevel__Microsoft_AspNetCore=Warning
- Kestrel__Endpoints__Http__Url=http://*:5140
- RabbitMQ__RabbitMqConnection__Server=rabbitmq
- RabbitMQ__RabbitMqConnection__Username=user
- RabbitMQ__RabbitMqConnection__Password=user
- RabbitMQ__RabbitMqConnection__Port=5672
- RabbitMQ__RabbitMqConnection__ClientProvidedName=ObjectModel
- RabbitMQ__RabbitMqConnection__Instance=Instance1
- RabbitMQ__RabbitMqConnection__AutomaticRecoveryEnabled=true
- RabbitMQ__RabbitMqConnection__NetworkRecoveryIntervalInSeconds=5
- RabbitMQ__RabbitMqConnection__TopologyRecoveryEnabled=true
- RabbitMQ__Producers__ObjectAttributeValueUpdatedProducer__UseBatch=true
- RabbitMQ__RabbitMqConnection__ReconnectDelayInSeconds=3
- Cache__KeyDb__ConnectionString=keydb:6379
- Cache__KeyDb__InstanceName=ObjectModel_
- Grpc__TagCacheInitClient__Settings__RetryDelayInSeconds=3
- Grpc__TagCacheInitClient__Settings__NumberOfAttemptsForRetry=3
- SqlSeedingScriptPath=/app/Sql/SeedObjectModel.sql
- AllowedHosts=*
depends_on:
  postgres:
    condition: service_healthy
  rabbitmq:
    condition: service_healthy
  keydb:
    condition: service_healthy
  prometheus:
    condition: service_started
networks:
  - my_network
restart: always

  keydb:
image: eqalpha/keydb:x86_64_v6.3.3
container_name: keydb
pull_policy: always
healthcheck:
  test: ["CMD", "keydb-cli", "ping"]
  interval: 10s
  timeout: 5s
  retries: 3
ports:
  - "6379:6379"
volumes:
```

```
- keydb_data:/data
networks:
  - my_network
restart: always

agent:
image: dergard/agent:latest
container_name: agent
pull_policy: always
environment:
  ConnectionStrings__EntityContext: Server=postgres;Port=5432;Database=agent;User
  Id=postgres;Password=postgres;Pooling=true;Maximum Pool Size=20;
  RabbitMqConnection__Server: rabbitmq
  RabbitMqConnection__Username: user
  RabbitMqConnection__Password: user
  Kestrel__Endpoints__gRPC__Url: "http://agent:5272"
  Cache__CacheService__DefaultExpirationTimeInMinutes: 9999999
  HangfireSettings__WorkerCount: 255
  gRPC__BatchSize: 50000
ports:
  - 5271:5271
  - 5272:5272
depends_on:
  postgres:
    condition: service_healthy
  rabbitmq:
    condition: service_healthy
networks:
  - my_network
restart: always

ckline4frontend:
image: dergard/ckline4frontend:main_vgo #контейнер под водоблок и факел, для м340
использовать "ckline4frontend:main_vgo"
container_name: ckline4frontend
pull_policy: always
depends_on:
  socketgateway:
    condition: service_started
networks:
  - my_network
ports:
  - 80:80
restart: always

ksescheduler:
container_name: ksescheduler
pull_policy: always
```

```
environment:
  ConnectionStrings__DbContext: Server=postgres;Port=5432;Database=schedules;User
  Id=postgres;Password=postgres;
image: dergard/kscheduler:latest
ports:
  - 5234:5234
depends_on:
  postgres:
    condition: service_healthy
networks:
  - my_network
restart: always

ksereports:
container_name: ksereports
image: dergard/ksereports:master
pull_policy: always
environment:
  ConnectionStrings__DbContext: Server=postgres;Port=5432;Database=reports;User
  Id=postgres;Password=postgres;
  SaveSettings__ShiftReportDestinationPath: /home/reports/shift
  SaveSettings__DailyReportDestinationPath: /home/reports/daily
  HttpConnection__HeatingAddress: http://socketgateway:4200/
  HttpConnection__PrometheusAddress: http://prometheus:9090/
  RoundDigits: 5
ports:
  - 5235:5235
volumes:
  - /etc/localtime:/etc/localtime:ro
  - /etc/timezone:/etc/timezone:ro
depends_on:
  postgres:
    condition: service_healthy
networks:
  - my_network
restart: always

grafana:
container_name: grafana
build:
  context: .
  dockerfile: Grafana_Dockerfile
pull_policy: always
environment:
  - GF_SECURITY_ADMIN_USER=admin
  - GF_SECURITY_ADMIN_PASSWORD=admin
  - GF_SECURITY_ALLOW_EMBEDDING=true
  #- GF_PATHS_PROVISIONING=/etc/grafana/provisioning
```

```
ports:
  - 18080:3000
volumes:
  - grafana_data:/var/lib/grafana
networks:
  - my_network
restart: always

messagescriptmodbus:
container_name: messagescriptmodbus
image: dergard/messagescriptmodbus:latest
environment:
  AGENT_URL: http://agent.api:5271
  HEATING_EVENT_SERVICE_URL: http://heatingeventservice:5200
  MODBUS_HOST: 127.0.0.1
  MODBUS_PORT: 502
  MODBUS_PLC_ADDRESS: 246
networks:
  - my_network
restart: always

zabbix-server:
image: zabbix/zabbix-server-pgsql:alpine-6.0-latest
container_name: zabbix-server
pull_policy: always
environment:
  ZBX_SERVER_HOST: zabbix-server
  DB_SERVER_HOST: postgres
  POSTGRES_USER: postgres
  POSTGRES_PASSWORD: postgres
  POSTGRES_DB: zabbix
  DB_SERVER_PORT: 5432
depends_on:
  postgres:
    condition: service_healthy
networks:
  - my_network
ports:
  - "10051:10051"
volumes:
  - zabbix-server-data:/var/lib/zabbix
restart: always

zabbix-web-nginx:
image: zabbix/zabbix-web-nginx-pgsql:alpine-6.0-latest
container_name: zabbix-web-nginx
pull_policy: always
environment:
```

```
ZBX_SERVER_HOST: zabbix-server
DB_SERVER_HOST: postgres
POSTGRES_USER: postgres
POSTGRES_PASSWORD: postgres
POSTGRES_DB: zabbix
DB_SERVER_PORT: 5432
depends_on:
  zabbix-server:
    condition: service_started
  postgres:
    condition: service_healthy
networks:
  - my_network
ports:
  - "8080:8080"
  - "8443:8443"
restart: always

zabbix-agent:
image: zabbix/zabbix-agent2:alpine-6.0-latest
container_name: zabbix-agent
pull_policy: always
environment:
  ZBX_HOSTNAME: docker-host
  ZBX_SERVER_HOST: zabbix-server
  DOCKER_GID: 130 # Укажите GID группы Docker. Чтобы узнать, в терминале ввести команду getent
group docker
depends_on:
  zabbix-server:
    condition: service_started
networks:
  - my_network
ports:
  - "10050:10050"
volumes:
  - ./zabbix_agent2.conf:/etc/zabbix/zabbix_agent2.conf
  - /var/run/docker.sock:/var/run/docker.sock
user: "1000:130" # UID пользователя и GID группы Docker. А чтобы узнать UID, ввести в терминале команду id -u имя_пользователя
restart: always

volumes:
  keydb_data:
    driver: local
  postgres_data:
    driver: local
  prometheus_data:
    driver: local
```

```
  rabbit_data:
driver: local
  grafana_data:
driver: local
  zabbix-postgres-data:
driver: local
  zabbix-server-data:
driver: local
networks:
  my_network:
driver: bridge
```

8. Приложение 2. Текст файла install.sh

```
#!/bin/bash

(
  handle_exit_code() {

    exit_code=$1
    title=$2
    text=$3

    case $exit_code in
      0)
        ;;
      1)
        zenity --warning --width=300 --title="$title" --text="$text"
        exit 1
        ;;
    esac
  }

  about_text() {
    echo -e "KSE SKline - современная кроссплатформенная SCADA для разработки проектов в области АСУТП"
    echo -e "Разработчик: ООО \"К-СОФТ ИНЖИНИРИНГ\""
    echo -e "Лицензия: "
  }

  about_page() {
    zenity --info \
    --title="О программе" \
    --width=500 \
    --text="$(about_text)"
  }

  handle_exit_code $? "Выход" "Закрытие окон мастера установки.."
}

license_page() {
  zenity --text-info \
    --title="Лицензионное соглашение" \
    --width=500 \
    --filename="LICENSE.txt" \
    --checkbox="Я прочитал и принимаю условия."
  return $?
}
}
```

```
password_page() {
TITLE='Права sudo для выполнения команд'

PASSWORD=$(zenity --password \
                --width=500 \
                --title="$TITLE")

case $? in
    0)
        sudo -k
        if ! echo $PASSWORD | sudo -Sv; then
            zenity --error --width=300 --text="Неверный пароль, попробуйте еще раз"
            password_page
        fi
        ;;
    1) # Нажатие на кнопку "Отменить" или крестик
        zenity --warning --width=300 --title="Выход" --text="Установка невозможна без ввода
        пароля sudo. Закрытие окон мастера установки.."
        exit 1
        ;;
esac
}

# Вызов окна для ввода пароля sudo
password_page

# Вызов стартового окна "О программе"
about_page

# Вызов окна с лицензией
license_page
handle_exit_code $? "Выход" "Установка невозможна без принятия лицензии. Закрытие окон мастера
установки.."

(
# =====
# Обновление браузера Chromium
echo "# Обновление браузера Chromium"
function version { echo "$@" | awk -F. '{ printf("%d%04d%04d%04d\n", $1,$2,$3,$4); }'; }

CURRENT_VERSION=$(chromium --version | awk '{print $2}')
REQUIRED_VERSION=128.0.6613.119

if [ $(version $CURRENT_VERSION) -lt $(version $REQUIRED_VERSION) ]
then
    echo "Update browser"
```

```
unzip -o ChromiumREPO.zip
echo "deb [trusted=yes] file:/home/$USER/CkLineRelease/ChromiumREPO/ ./" | sudo tee
-a /etc/apt/sources.list.d/rspoREPO.list
sudo apt update
sudo apt install chromium=1:128.0.6613.119-0astragost0+ci202409101631+astra8
chromium-l10n=1:128.0.6613.119-0astragost0+ci202409101631+astra8 -y
fi

# =====
# Установка Docker
echo "10"
echo "# Установка Docker"
if ! command -v docker &> /dev/null;
then
unzip dockerZabbixREPO.zip
echo "deb [trusted=yes] file:/home/$USER/CkLineRelease/dockerZabbixREPO/ ./" | sudo tee
-a /etc/apt/sources.list.d/rspoREPO.list
sudo apt update
sudo apt install docker-ce=5:26.1.4-1~debian.10~buster docker-ce-cli=5:26.1.4-1~debian.10~buster
docker-buildx-plugin=0.14.1-1~debian.10~buster docker-compose-plugin=2.27.1-1~debian.10~buster -y
sudo usermod -aG docker $USER

fi

# =====
# Загрузка образов из архива
echo "25"
echo "# Загрузка образов из архива"
unzip -o docker-images.zip -d docker-images
for TAR_FILE in "docker-images"/*.tar; do
docker load -i "$TAR_FILE"
done

# =====
# Извлечение файлов из архива KSEckline
echo "40"
echo "# Извлечение файлов из архива KSEckline"
unzip -o KSEckline.zip -d KSEckline

cd KSEckline

# =====
# Изменение docker-compose.yml для Zabbix
echo "45"
echo "# Изменение docker-compose.yml"
GROUP_ID=$(getent group docker | awk -F: '{print $3}')
```

```
RESULT1=" DOCKER_GID: $(echo "$GROUP_ID")"
sed -i '309c\'"$RESULT1"' docker-compose.yml

USER_ID=$(id -u $USER | awk -F: '{print $1}')

RESULT2=" user: \"$(echo "$USER_ID"):$(echo "$GROUP_ID")\""
sed -i '320c\'"$RESULT2"' docker-compose.yml

# =====
# Установка проекта из docker-compose.yml
echo "50"
echo "# Установка проекта"

echo $PASSWORD | sudo -S docker compose up -d

# =====
# Установка portainer
echo "90"
echo "# Установка portainer"
sudo docker run -d \
  --name portainer \
  -p 9000:9000 \
  -p 8000:8000 \
  --restart=always \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v portainer_data:/data \

# =====
echo "# Завершение установки"
echo "100"

) |
zenity --progress \
  --width=300 \
  --title="Этапы установки" \
  --text="Логи" \
  --percentage=0

handle_exit_code $? "Выход" "Закрытие окон мастера установки.."

# Проверка доступности 80го порта

URL="http://localhost"

if curl -s --head --request GET "$URL" | grep -q "200 OK"; then
  ans=$(zenity --info --width=300 \
```

```
--title="Успешный запуск" \  
--text="KSEckline успешно запущен!" \  
--ok-label "Выйти" \  
--extra-button "Открыть в браузере" \  
)  
  
if [[ $ans == "Открыть в браузере" ]];  
then  
    /usr/bin/chromium --disable-features=Translate "$URL"  
else  
    zenity --warning --width=300 --title="Выход" --text="Заккрытие окон мастера установки.."  
    exit 1  
fi  
else  
    zenity --error --width=300 --text="Ошибка: KSEckline не запущен или URL недоступен."  
fi  
  
#) 2>&1 | tee output.log > /dev/null  
) 2>&1 | tee output.log
```