Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа бакалавриата

ОТЧЕТ

По проекту
«Разработка пайплайна для автоматизации тестовых стендов в k8s под Linux»

по дисциплине «Проектный практикум»

Заказчик: Фамилия И.О. Присяжный А. В.

Куратор: Фамилия И.О. Присяжный А. В.
ученая степень, ученое звание, должность

Студенты команды Проклятые работяги

Фамилия И.О. Дрюк И. В.

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
1 Аналитика 5
1.1 GitLab CI/CD 5
1.2 ArgoCD 6
1.3 Существующие скриптовые решения 7
1.4 Итоговый анализ решений 7
2 Архитектура и инфраструктура проекта 9
2.1 Общая структура решения 9
2.2 Репозитории и CI/CD 9
2.3 Инфраструктура и окружение 10
2.4 K3s и выбор Kubernetes 13
2.5 Работа с базами данных 14
3 Ход разработки и трудности 15
4 Отчет участников 18
4.1 Отчет Дрюк Ивана 18
ЗАКЛЮЧЕНИЕ 20
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21
ПРИЛОЖЕНИЕ А (обязательное) Техническое задание (ТЗ) на разработку системы управления освещением 22
1 Стек используемых технологий и компонентов 23
2 Схема архитектуры развертывания 24

ВВЕДЕНИЕ

Целью данного проекта является разработка пайплайна для автоматизации создания тестовых стендов в среде Kubernetes под операционной системой Linux. Этот пайплайн предназначен для упрощения и ускорения процессов развертывания и управления изолированными окружениями, используемыми для тестирования разработки программного продукта Directum RX.

Актуальность проекта обусловлена необходимостью повышения эффективности DevOps-процессов в компании. На текущий момент уже реализована частичная автоматизация с использованием shell-скриптов, Python, Docker и вспомогательных сервисов (PostgreSQL, RabbitMQ и др.). Однако в существующей системе отсутствует возможность полного автоматизированного создания стенда по заданным параметрам, а также актуализации уже запущенных стендов. Поддержка Kubernetes со стороны вендора заявлена, но находится на ранней стадии и требует значительной доработки.

Современные инструменты CI/CD, такие как GitLab CI/CD, Jenkins, ArgoCD, обеспечивают высокий уровень автоматизации и удобную интеграцию с Kubernetes, но они не учитывают специфику работы с Directum RX — наличие специализированной схемы стендов, внутренней логики сборки пакетов и особых требований к инфраструктуре. Поэтому в рамках проекта создаётся кастомное решение, которое объединяет в себе гибкость GitLab CI/CD, использование Helm-чартов, генерацию образов, систему управления стендами и средства наблюдаемости (Prometheus, Grafana и др.).

Проект ориентирован на инженеров DevOps, разработчиков, тестировщиков и системных администраторов, работающих с Directum RX и заинтересованных в ускорении процессов тестирования, уменьшении количества ручных операций, а также повышении прозрачности и управляемости инфраструктуры.

В результате выполнения проекта будет создана система, способная:

– автоматически поднимать тестовый стенд в Kubernetes за менее чем 10 минут;

– актуализировать уже существующие стенды без их полного пересоздания;

– отображать текущее состояние всех стендов, связанных с репозиториями и ветками;

– предоставлять информацию о статусах, ошибках и логах работы пайплайнов;

– сводить к минимуму необходимость в ручной документации благодаря высокой степени автоматизации.

Реализация данной системы позволит существенно сократить время на развёртывание тестовой среды, обеспечить её стабильную работу и упростить контроль над тестовой инфраструктурой в условиях ограниченной документации и нестандартных требований.

1 Аналитика

Для реализации автоматизированного пайплайна развертывания тестовых стендов Directum RX в Kubernetes было проведено исследование существующих решений и подходов в области CI/CD, DevOps и управления кластерной инфраструктурой. Анализ охватывает наиболее популярные инструменты и технологии, применяемые в сфере автоматизации развёртывания программных продуктов, с акцентом на интеграцию с Kubernetes, масштабируемость, адаптируемость под бизнес-логику и удобство сопровождения.

В качестве объектов анализа выбраны три направления:

– стандартные CI/CD пайплайны на базе GitLab CI/CD;

– специализированные инструменты управления приложениями в Kubernetes, такие как ArgoCD;

– собственные решения на базе скриптов и Helm-чартов, кастомизированные под конкретную предметную область.

Анализ проводился по следующим критериям:

1) Уровень автоматизации: возможность развертывания, обновления и удаления стендов без ручных действий;

2) Интеграция с Kubernetes: использование Kubernetes-нативных подходов (Helm, Custom Resources, GitOps);

3) Гибкость настройки: возможность адаптации под нестандартные сценарии сборки и специфические требования Directum RX;

4) Удобство использования и визуализация: наличие интерфейсов, трекинга статуса, логов, уведомлений об ошибках;

5) Сложность поддержки и масштабирования: потребность в ручной доработке, стабильность, наличие документации.

1.1 GitLab CI/CD

GitLab CI/CD — одно из наиболее широко применяемых решений для автоматизации процессов сборки, тестирования и развертывания. Благодаря тесной интеграции с Git-репозиториями, GitLab обеспечивает высокую степень автоматизации и удобства, особенно в средах, где активно используется git-flow.

GitLab CI/CD позволяет создавать пайплайны для автоматической сборки и доставки Docker-образов, применения Helm-чартов и управления Kubernetes-объектами. Однако реализация сложной логики (например, проверка состояния существующего стенда, выбор версии пакета, кастомная генерация конфигураций) требует значительного объёма кастомных скриптов. GitLab не предоставляет встроенных средств наблюдаемости за стендами и не обеспечивает «живой» мониторинг текущего состояния среды. Кроме того, требуется внешняя интеграция с инструментами визуализации (например, Prometheus, Grafana).

GitLab CI/CD остаётся мощным решением, но его использование в контексте Directum RX требует создания дополнительной логики и сервисов для полноценной автоматизации.

1.2 ArgoCD

ArgoCD — инструмент GitOps-подхода, позволяющий управлять развертыванием приложений в Kubernetes через декларативные манифесты, хранимые в Git. Это решение обладает высокой степенью автоматизации, обеспечивает непрерывную синхронизацию состояния кластера с Git-репозиториями, предоставляет визуальный интерфейс и уведомления об отклонениях.

Однако ArgoCD предполагает, что все развертывания должны быть описаны заранее в виде манифестов и не адаптировано под частые изменения логики установки, характерные для стендов Directum RX. Кроме того, для динамической генерации значений (например, для конфигурации сервисов,

баз данных и сетей) требуется создание сложных пайплайнов подготовки и управления Helm-значениями, что усложняет сопровождение. ArgoCD эффективно работает в стабильной продуктивной среде, но не так гибок для быстрой генерации временных окружений.

1.3 Существующие скриптовые решения

На момент начала проекта существовала система, основанная на shell- и Python-скриптах, которая частично автоматизировала процесс создания тестовых стендов Directum RX. Она включала в себя этапы генерации Helm-значений, запуск Helm-чартов, настройку конфигураций и сервисов. Однако такая архитектура не обеспечивала повторяемости процессов, была слабо масштабируемой, не имела централизованного мониторинга и затрудняла отладку.

Скрипты разрабатывались по мере необходимости, не имели чёткой структуры и документации, что снижало надёжность системы и делало её уязвимой к ошибкам при ручном вмешательстве. Отсутствие визуального контроля над состоянием стендов также усложняло сопровождение, особенно при большом числе параллельных развёртываний.

1.4 Итоговый анализ решений

В таблице 1 представлено сравнительное описание рассмотренных подходов.

Таблица 1 Сравнительная таблица решений

Характеристика	GitLab CI/CD	ArgoCD	Скриптовое решение	Разрабатываемая система
Уровень автоматизации	Средний	Высокий	Низкий	Высокий
Интеграция с Kubernetes	Да	Да	Частично	Да
Гибкость под Directum RX	Средняя	Низкая	Высокая	Высокая
Визуализация/мониторинг	Ограниченно	Есть	Нет	Да
Удобство масштабирования	Среднее	Высокое	Низкое	Высокое
Уровень сопровождения	Средний	Сложный	Высокий	Упрощённый

Результаты анализа показали, что ни одно из существующих решений не удовлетворяет всем требованиям проекта без значительных доработок. Поэтому в рамках проекта разрабатывается собственный пайплайн, сочетающий удобство GitLab CI/CD, управляемость Helm-чартов, визуализацию через Prometheus и Grafana, а также адаптированную под Directum RX логику. Это позволит добиться высокой гибкости, масштабируемости и надёжности в развёртывании тестовых стендов в Kubernetes.

2 Архитектура и инфраструктура проекта

2.1 Общая структура решения

Проект представляет собой инфраструктурную платформу для автоматизированного развёртывания и управления стендами системы DirectumRX с использованием контейнеризации, оркестрации и CI/CD. Архитектура построена по принципу инфраструктура как код (IaC) и обеспечивает полную автоматизацию развёртывания, конфигурирования и сопровождения окружений.

Решение реализовано с использованием следующих компонентов:

– CI/CD пайплайны (GitLab CI)

– Контейнеризация (Docker)

– Оркестрация (K3s)

– Хранилище образов (Harbor)

– Виртуализация (Proxmox)

– Облачное хранилище (MinIO)

– Внутренняя DNS и реклама доменов (Pi-hole)

– RabbitMQ, PostgreSQL, Longhorn и др.

2.2 Репозитории и CI/CD

В проекте используется многоуровневая структура репозиториев, каждый из которых выполняет определённую роль:

2.2.1 Основной репозиторий

Содержит основной gitlab-ci.yml со следующими джобами:

– автоматический деплой стендов;

– инициализация базы данных (do.sh db up);

– копирование БД между стендами;

а также скрипты и конфигурации инфраструктуры.

2.2.2 Репозиторий конфигураций

В данном репозитории каждому стенду соответствует отдельная ветка с файлом config.yml;

Используется в процессе CI/CD — основной репозиторий подтягивает config.yml из соответствующей ветки;

Описывает параметры подключения к БД, переменные развертывания, настройки компонентов.

2.2.3 Репозиторий DirectumLauncher Docker

Предназначен для сборки кастомного образа с DirectumLauncher — инструментом развёртывания DirectumRX;

На текущем этапе функциональность частично внедрена, потенциально может использоваться для более гибкой настройки сборок и автоматизации версионирования.

2.3 Инфраструктура и окружение

Развёртывание осуществляется в изолированной инфраструктуре, построенной на базе Proxmox VE (рисунок 1). Архитектура включает:

– Мастер-нода Kubernetes (k3s), две воркер-ноды для балансировки нагрузки, storage-нода с Longhorn (рисунок 2);

– Сервер Harbor — локальное Docker-registry (рисунок 3);

– MinIO-сервер (рисунок 4);

– PostgreSQL — основная СУБД (рисунок 5);

– RabbitMQ, установленный через do.sh;

– Pi-hole — для локального DNS и контроля доменных имён (рисунок 6).

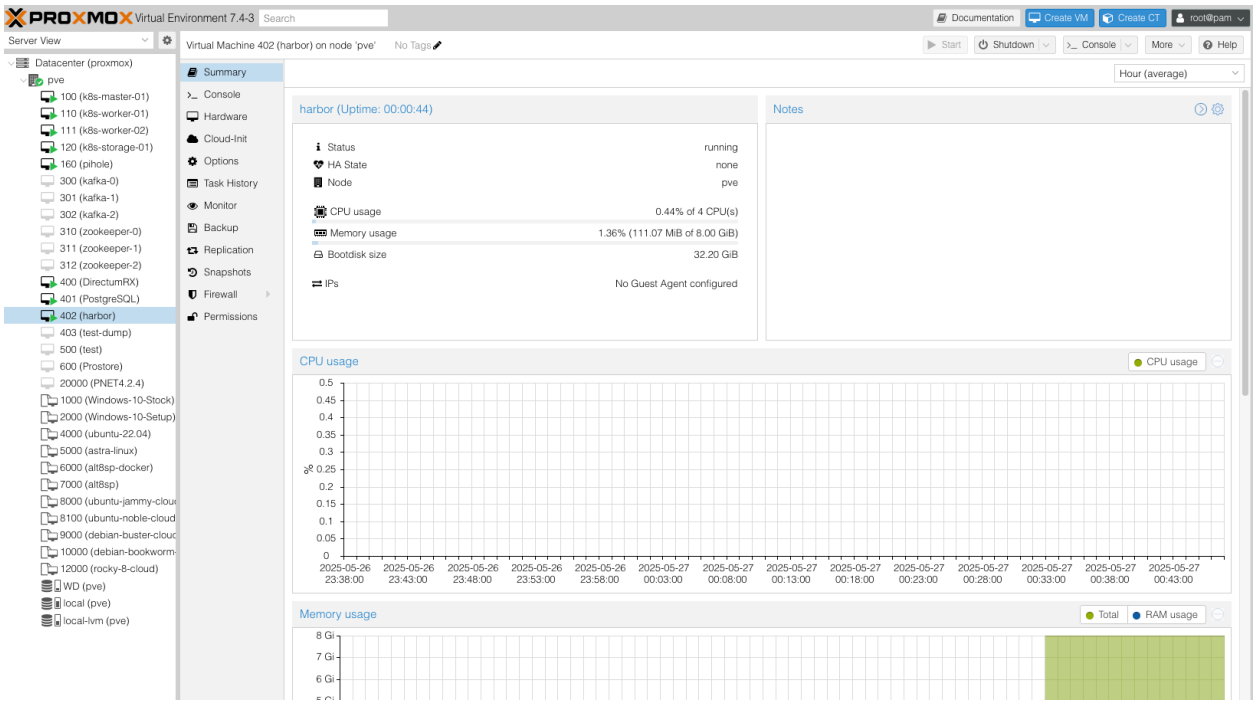

Рисунок 1 – Инфраструктура на Proxmox VE

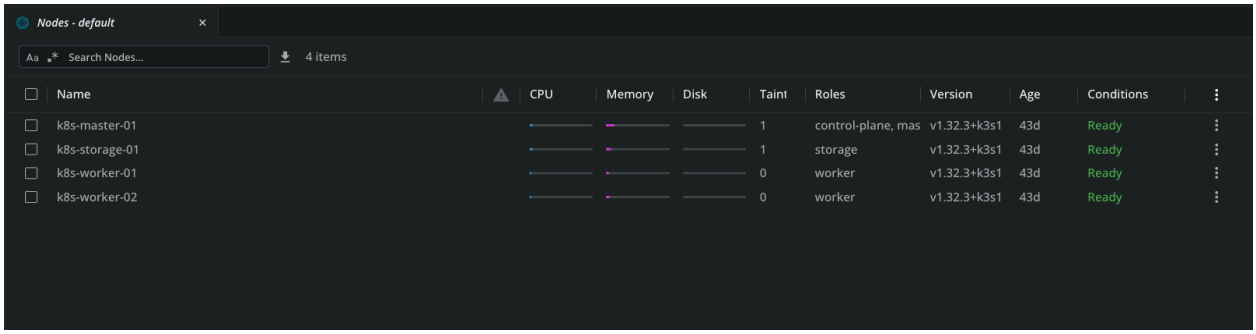

Рисунок 2 – Kubernetes-кластер

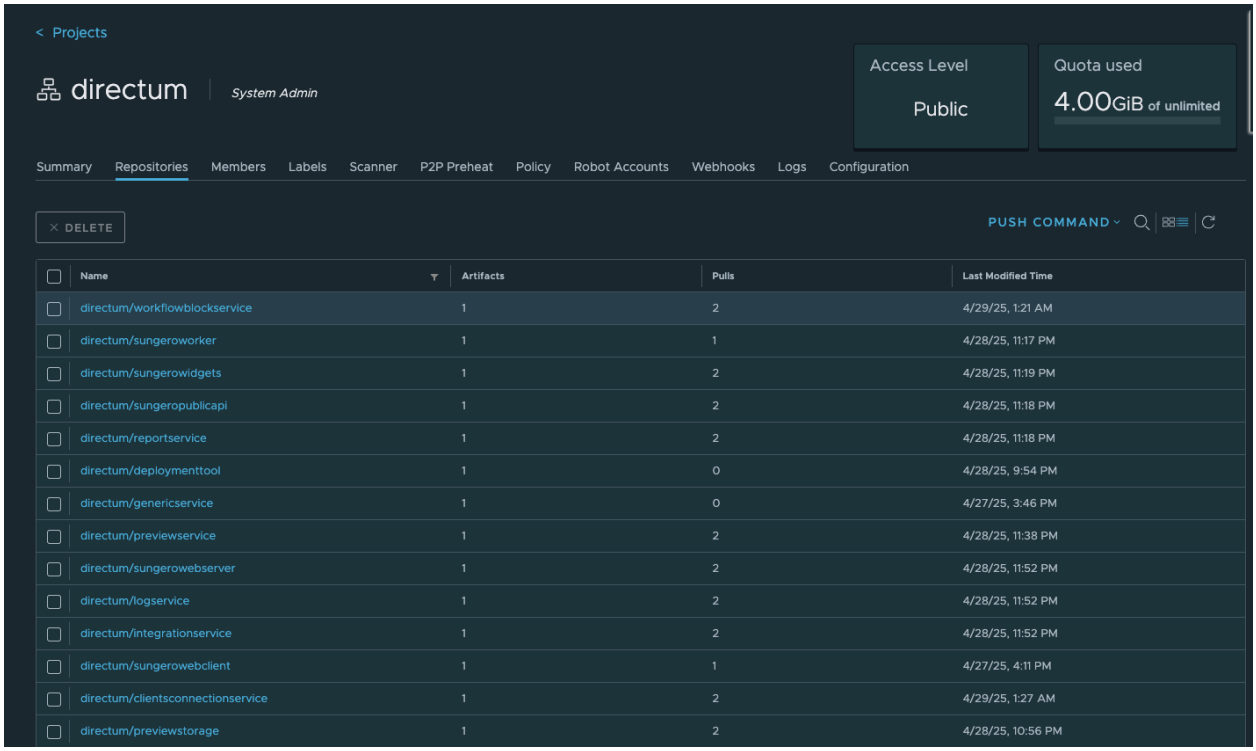

Рисунок 3 – Локальный Docker-registry Harbor

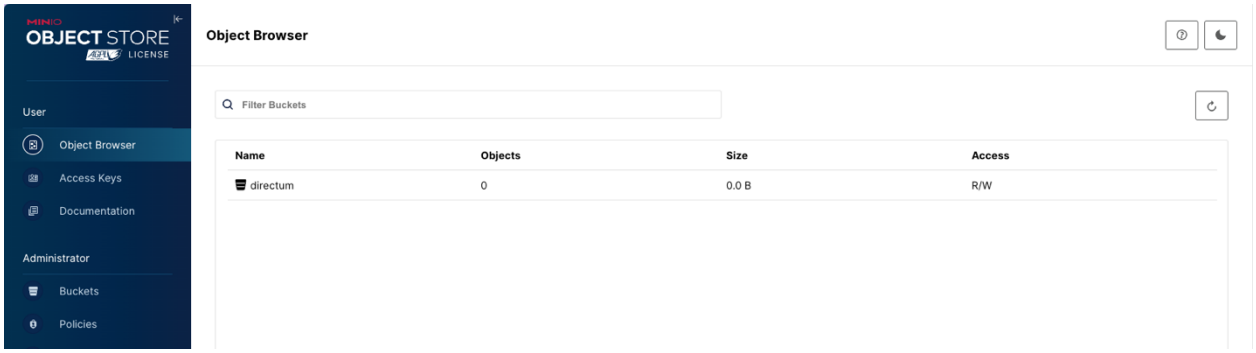

Рисунок 4 – MinIO

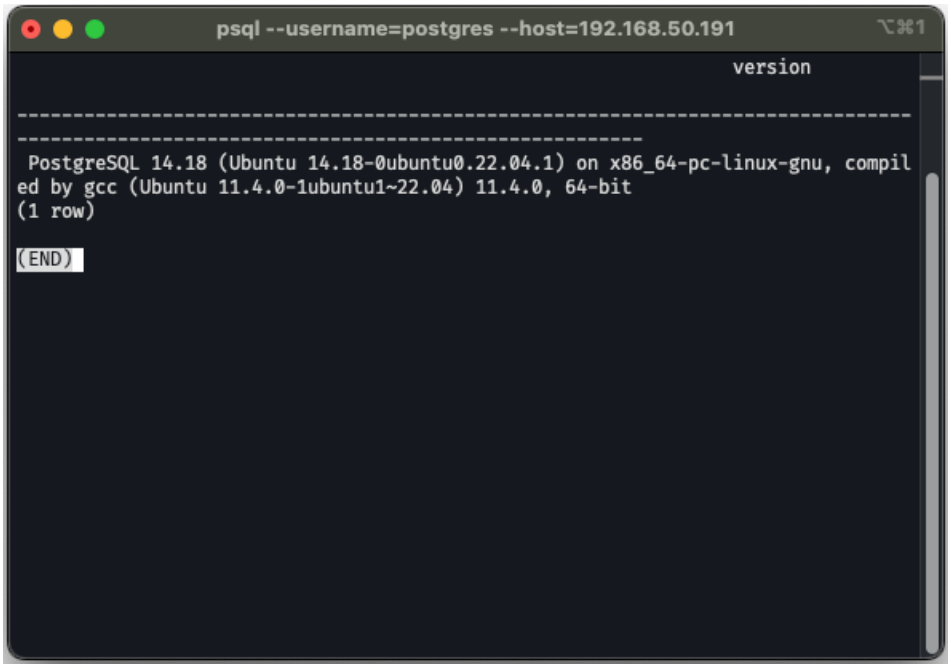

Рисунок 5 – PostgreSQL

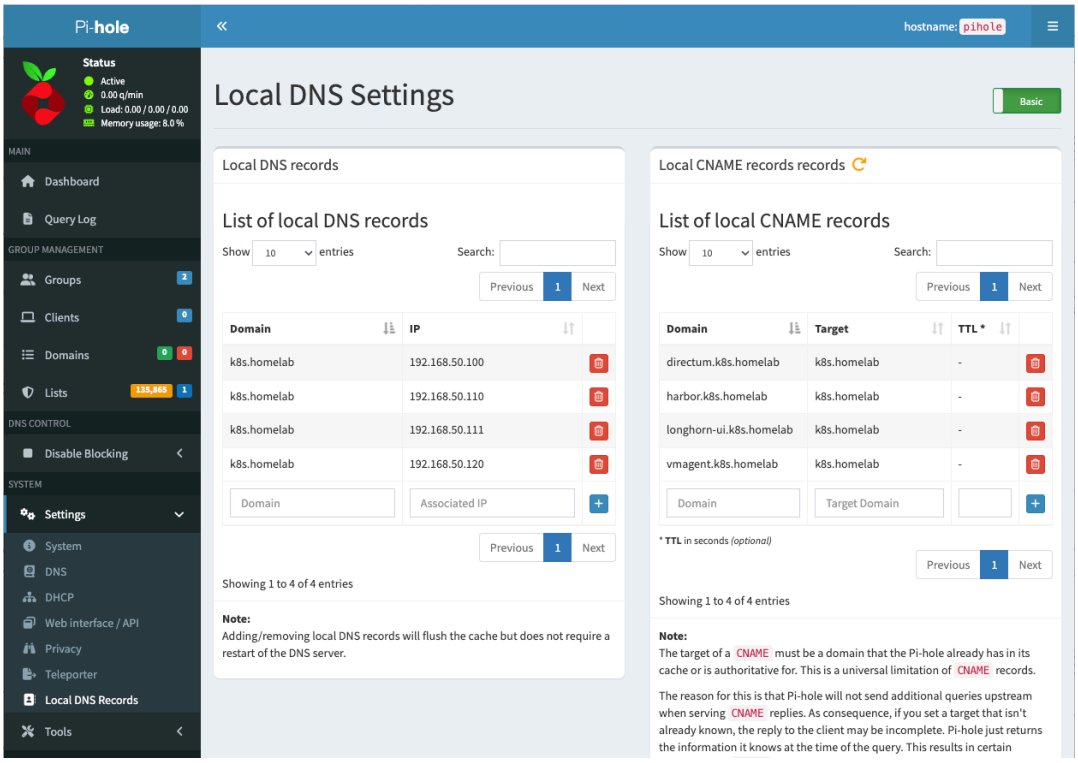

Рисунок 6 – Pi-hole

2.4 K3s и выбор Kubernetes

В качестве платформы для оркестрации контейнеров выбран k3s — лёгковесная, но полностью совместимая с Kubernetes дистрибуция, идеально подходящая для частных инфраструктур и edge-решений. Причины выбора:

1) Простота установки и поддержки;

2) Поддержка Helm, Traefik, CRD и всех необходимых компонентов;

3) Совместимость с CI/CD процессами;

4) Низкие системные требования.

2.5 Работа с базами данных

Для управления PostgreSQL в проекте предусмотрены два подхода:

– Инициализация с нуля: используется do.sh, принимающий параметры подключения из config.yml (репозиторий конфигураций).

– Копирование БД между стендами: специальный скрипт копирует базу данных из одного стенда в другой, с параметрами через ENV или переменные проекта GitLab.

3 Ход разработки и трудности

Разработка инфраструктуры для автоматизированного развёртывания DirectumRX началась с анализа требований к системе, а также ознакомления с особенностями платформы и сопутствующих инструментов. На этом этапе были изучены ключевые компоненты: механизмы деплоя DirectumRX, взаимодействие с Harbor, настройка k3s, структура образов и работа скриптов do.sh.

В качестве архитектурной основы была выбрана Kubernetes-совместимая платформа k3s благодаря её лёгкости и простоте развёртывания в частных инфраструктурах. На базе Proxmox были созданы виртуальные машины — мастер-нода, два воркера и отдельная storage-нода для Longhorn, что позволило обеспечить устойчивое и масштабируемое окружение.

Важным этапом стало проектирование структуры репозиториев. Основной репозиторий содержал:

– .gitlab-ci.yml, обеспечивающий автоматический деплой с использованием GitLab CI/CD;

– скрипт миграции БД, позволяющий создавать копии текущих стендов;

– инструмент настройки новой базы данных do.sh, параметризуемый через ENV.

Дополнительно был создан отдельный репозиторий для конфигураций — с отдельной веткой на каждый стенд, содержащей файл config.yml, откуда основной пайплайн получал настройки.

Также было предусмотрено:

– развертывание Harbor в качестве локального реестра Docker-образов;

– установка MinIO (используется DirectumRX для хранения файлов);

– развертывание RabbitMQ с помощью утилиты do.sh;

– поднятие сервера PostgreSQL как основной СУБД системы;

– настройка локального DNS через Pi-hole для управления доменными именами в тестовом окружении.

В итоге с помощью созданной инфраструктуры и репозиториев был развернут тестовый стенд DirectumRX доступный по домену, настроенному в Pi-hole (рисунок 7 и рисунок 8).

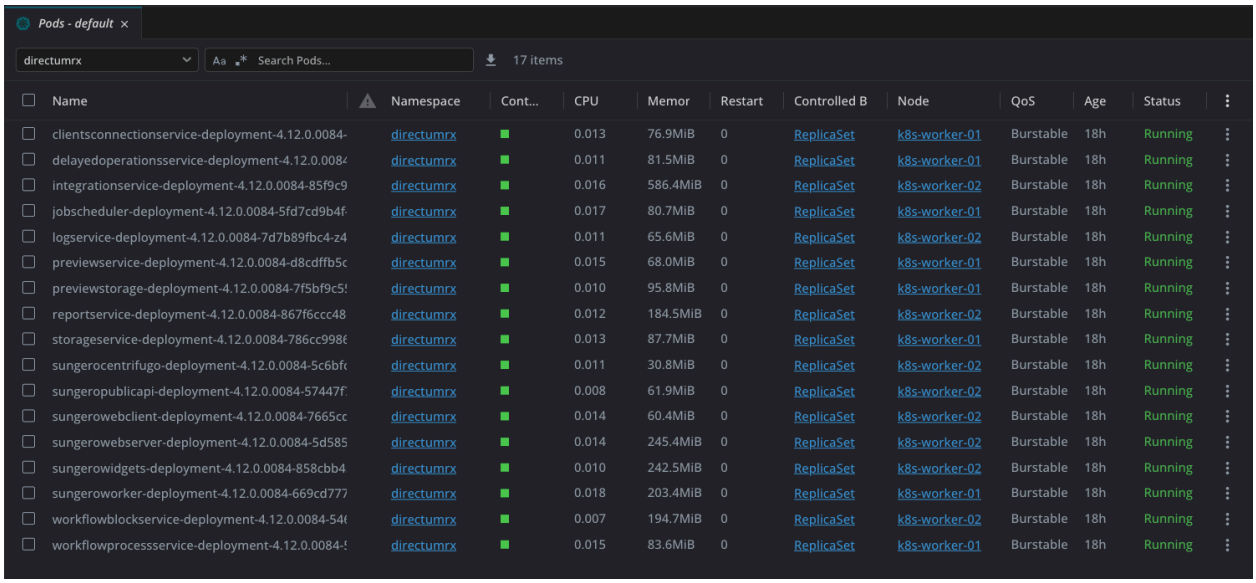

Рисунок 7 – Стенд в неймспейсе Kubernetes

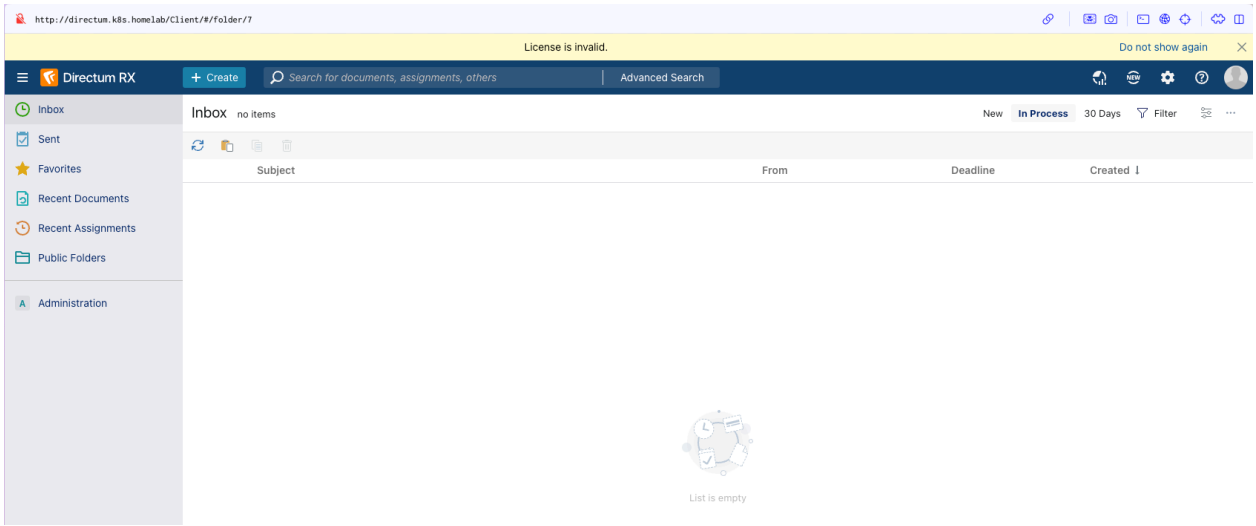

Рисунок 8 – Web-UI Directum RX

На практике в ходе работы было выявлено несколько узких мест и сложностей:

– Из-за отсутствия полной технической документации часть возможностей и требований DirectumRX приходилось исследовать вручную

— как по логам, так и через эмпирические эксперименты с параметрами в config.yml и поведением в среде.

– Одной из технически сложных задач стала настройка k3s на работу с небезопасным (http) Harbor-реестром. Пришлось вручную добавлять доверенные реестры на каждом узле, настраивать сертификаты и вносить правки в systemd unit-файлы.

Тем не менее, все задачи были успешно решены, и по результатам тестов система показала стабильную и воспроизводимую работу. Итоговая инфраструктура получилась модульной, расширяемой и легко масштабируемой на новые стенды.

4 Отчет участников

4.1 Отчет Дрюк Ивана

В рамках проекта по автоматизации развёртывания инфраструктуры DirectumRX мной были реализованы ключевые задачи, связанные с проектированием, настройкой и оптимизацией CI/CD-процесса, а также автоматизацией развёртывания среды на базе k3s и сопутствующих сервисов.

На первом этапе работы я провёл анализ требований к будущей системе, включая принципы функционирования DirectumRX и состав её компонентов. На основании полученной информации была составлена предварительная архитектура проекта, включающая кластер из виртуальных машин на базе Proxmox, разделённых на мастер-ноду, воркеры и отдельный узел хранения данных с Longhorn.

Основной вклад был сосредоточен в следующих направлениях:

– Проектирование и реализация CI/CD-пайплайна. Мной был разработан файл .gitlab-ci.yml, обеспечивающий автоматическое развертывание среды с учётом переменных окружения и индивидуальных конфигураций. Была реализована поддержка развёртывания разных стендов на основе отдельных веток конфигурационного репозитория.

– Организация работы с конфигурациями. Я предложил и внедрил структуру второго репозитория, в котором каждая ветка соответствовала отдельному стенду. Это позволило гибко управлять настройками, а также повысить повторяемость и устойчивость развёртывания.

– Настройка и интеграция сервисов. Я осуществил настройку локального реестра Docker-образов Harbor и обеспечил его интеграцию с k3s. Одной из сложностей стало заставить k3s доверять HTTP-реестру Harbor — для решения этой задачи потребовалось вручную внести настройки на уровне всех нод и внести изменения в конфигурацию systemd.

– Работа с системой хранения и БД. Был произведён деплой MinIO, RabbitMQ и PostgreSQL, необходимых для работы DirectumRX. Также мной была адаптирована утилита do.sh под параметры текущей инфраструктуры и реализована поддержка автоматической настройки переменных окружения.

Дополнительно мной проводилось тестирование пайплайна, отладка процессов миграции баз данных, а также настройка локального DNS через Pi-hole, обеспечивающего маршрутизацию внутри внутренней сети.

Благодаря планомерному подходу, самодисциплине и поддержке со стороны кураторов проекта все трудности с проектом были успешно преодолены. В результате проделанной работы был создан надёжный и гибкий механизм развёртывания среды, адаптированный под нужды DirectumRX и способный масштабироваться под различные задачи.

Проект стал ценным опытом как в плане технической реализации, так и в плане организации собственной деятельности в рамках команды.

ЗАКЛЮЧЕНИЕ

В рамках выполнения проекта по автоматизации развёртывания среды DirectumRX с использованием современных DevOps-подходов была достигнута основная цель — создание рабочей инфраструктуры, обеспечивающей быстрое, воспроизводимое и масштабируемое развёртывание программного комплекса в изолированной среде. Разработанная система позволяет оперативно запускать новые стенды, управлять конфигурациями и обновлениями, а также отслеживать состояние компонентов через автоматизированный пайплайн CI/CD.

Проект был выполнен в соответствии с поставленными задачами и сроками, охватывая полный цикл — от проектирования архитектуры и выбора инструментов до настройки кластера k3s, локального Docker-реестра Harbor, внедрения системы управления конфигурациями и интеграции пайплайна GitLab CI. В процессе разработки было реализовано решение, учитывающее особенности работы DirectumRX, включая использование служебных скриптов, специализированных переменных и требований к внешним сервисам, таким как RabbitMQ, PostgreSQL и MinIO.

Отдельным вызовом стала настройка доверенного доступа к небезопасному (HTTP) Docker-реестру в контексте k3s и автоматизация конфигураций всех узлов. Тем не менее, благодаря проведённому исследованию и экспериментам, все технические сложности были успешно преодолены.

Проект стал не только практической реализацией полученных теоретических знаний в области DevOps, Kubernetes и CI/CD, но и ценным опытом взаимодействия с комплексными корпоративными решениями. Полученные результаты демонстрируют высокую степень готовности системы к дальнейшему использованию, а также возможность её масштабирования и адаптации под различные сценарии эксплуатации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальная страница документации Gitlab в интернете [Электронный ресурс] – URL: https://docs.gitlab.com/ (дата обращения: 20.03.2025).

2. Официальная страница документации Harbor [Электронный ресурс] – URL: https://goharbor.io/docs/2.13.0/ (дата обращения: 21.03.2025).

3. Официальная страница документации MinIO для Docker [Электронный ресурс] – URL: https://min.io/docs/minio/container/index.html (дата обращения: 22.03.2025).

4. Официальная страница документации k3s [Электронный ресурс] – URL: https://docs.k3s.io/ (дата обращения: 23.03.2025).

5. Официальная страница документации Pi-hole [Электронный ресурс] – URL: https://docs.pi-hole.net/ (дата обращения: 24.03.2025).

6. Установка системы DirectumRX в Kubernetes [Электронный ресурс] – URL: https://club.directum.ru/webhelp/directumrx/web/index.html?admin_inst_kubernetes.htm (дата обращения: 08.05.2025).

7. Администрирование DirectumRX (Linux) [Электронный ресурс] – URL: https://club.directum.ru/webhelp/directumrx/web/index.html?admin_linux.htm (дата обращения: 9.05.2025).

8. Официальная страница Helm [Электронный ресурс]– URL: https://helm.sh/docs/ (дата обращения: 18.05.2025).

ПРИЛОЖЕНИЕ А

(обязательное)

Техническое задание (ТЗ) на разработку системы управления освещением

1 Стек используемых технологий и компонентов

Таблица 2 Таблица технологий и компонентов

Характеристика	GitLab CI/CD
Kubernetes (k3s)	Оркестрация контейнеров, управление развертыванием микросервисов
Harbor	Приватный Docker-реестр образов, используется для хранения и доставки
Directum RX	Корпоративная платформа, интеграция с внутренней документацией
Helm	Упрощённое управление конфигурацией приложений в Kubernetes
Docker	Контейнеризация
Nginx Ingress Controller	Управление входящими HTTP-запросами, маршрутизация трафика

2 Схема архитектуры развертывания

Система развёрнута в виде набора микросервисов в Kubernetes-кластере. Каждый компонент — отдельно масштабируемый Deployment или StatefulSet. Входящий трафик направляется через Ingress, обрабатывается Nginx и перенаправляется на нужный сервис (UI, API и др.). Образы всех компонентов хранятся в локальном Harbor-репозитории. Безопасная доставка образов обеспечивается путём добавления доверенного сертификата Harbor в k3s.