

ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ
«Смарт-Ком»

**«Opti-Track TMS – система управления транспортом»
Инструкция по установке**

Версия 1.0

Содержание

1. ВВЕДЕНИЕ.....	4
2. ОБЩИЕ СВЕДЕНИЯ.....	5
2.1. Назначение и область применения	5
2.2. Эксплуатационное назначение программы.....	5
2.3. Задачи, решаемые программой	5
2.4. Состав программы и архитектурное окружение	6
3. ПОДГОТОВКА ИНФРАСТРУКТУРЫ.....	7
3.1. Назначение и ограничения	7
3.2. Предварительные требования	7
3.3. Подготовка узлов и сервисных машин	8
3.4. Первичная настройка кластера Kubernetes	9
3.5. Загрузка секретов в Vault.....	10
4. РАЗВЁРТЫВАНИЕ СЕРВЕРНОЙ ЧАСТИ ПРОГРАММЫ.....	12
4.1. Назначение и ограничения	12
4.2. Предварительные требования	12
4.3. Настройка параметров развёртывания	12
4.4. Остановка и удаление текущего набора сервисов программы	15
4.5. Создание и запуск нового набора сервисов программы	15
4.5.1. Общая последовательность	15
4.5.2. Подготовка к запуску конвейера	16
4.5.3. Этап подготовки задания.....	16
4.5.4. Задание validate-app	16
4.5.5. Задание build-app.....	17
4.5.6. Задание build-cdcmonitor	18
4.5.7. Задание deploy-app	18
4.6. Проверка работоспособности.....	18
5.1. Назначение и ограничения	20
5.2. Предварительные требования	20
5.3. Настройка параметров развёртывания	20
6. ОПЦИОНАЛЬНО. РАЗВЁРТЫВАНИЕ СЕРВИСОВ НАБЛЮДАЕМОСТИ	21
6.1. Назначение и ограничения	21
6.2. Предварительные требования	21
6.3. Развёртывание инфраструктурного стека наблюдаемости	21

6.4. Развёртывание мониторинга потоков Kafka Connect.....	22
6.5. Адреса служб	22
7. ОПЦИОНАЛЬНО. РАЗВЁРТЫВАНИЕ КОНТРОЛЛЕРА ПЕРЕКЛЮЧЕНИЯ ЗОН.....	24
7.1. Назначение и ограничения	24
7.3. Развёртывание.....	24
7.4. Безопасность контроллера	26
8. ОПЦИОНАЛЬНО. РАЗВЁРТЫВАНИЕ ИНТЕГРАЦИОННОГО ПРОГРАММНОГО ИНТЕРФЕЙСА.....	27
8.1. Назначение и ограничения	27
8.2. Развёртывание.....	27
9. ОПЦИОНАЛЬНО. РАЗВЁРТЫВАНИЕ СЕРВИСА КОНВЕРТАЦИИ ДОКУМЕНТОВ ...	28
9.1. Назначение и ограничения	28
9.2. Развёртывание.....	28
10. ОТКАТ ВЕРСИИ.....	29
10.1. Назначение и ограничения	29
10.2. Порядок отката.....	29
Приложение А. Перечень сервисов прикладной части	30
Приложение Б. Перечень служебных контейнеров	33
Лист регистрации изменений.....	34

1. ВВЕДЕНИЕ

Данный документ содержит порядок установки программного обеспечения «OptiTrack TMS» (далее — «Программа»).

В разделе «Общие сведения» указаны сведения о назначении Программы и информация, достаточная для понимания её функций и эксплуатации.

В разделе «Подготовка инфраструктуры» представлен порядок первичной подготовки кластера Kubernetes и сервисных виртуальных машин (мониторинг, картографические сервисы), без которого развёртывание Программы невозможно.

В разделах «Развёртывание серверной части Программы» и «Развёртывание клиентской части Программы» описан порядок установки прикладных микросервисов и точек входа, в процессе которой выполняется автоматизированное развёртывание контейнеров в кластере Kubernetes через инструменты непрерывной поставки.

Разделы, содержащие в наименовании пометку «ОПЦИОНАЛЬНО», выполняются в случае необходимости установки указанных сервисов в инфраструктурном контуре.

Раздел «Откат версии» содержит порядок возврата прикладной части к предыдущей рабочей сборке.

Эксплуатация Программы после установки осуществляется в соответствии с эксплуатационной документацией на неё.

2. ОБЩИЕ СВЕДЕНИЯ

2.1. Назначение и область применения

Программа предназначена для управления транспортной логистикой и цепочками поставок промышленных предприятий. Программа поддерживает планирование рейсов, проведение аукционов перевозчиков, бронирование таймслотов, ведение контрактов, мастер-данных, инцидентов, печатных и электронных транспортных документов, а также интеграцию со смежными системами заказчика.

Область применения — промышленные предприятия, эксплуатирующие собственный или подрядный транспорт и нуждающиеся в централизованном управлении транспортными операциями.

2.2. Эксплуатационное назначение программы

Программа эксплуатируется в составе действующего промышленного контура заказчика. Серверная часть размещается в управляемом кластере Kubernetes Яндекс Облака, клиентская часть доступна пользователям и внешним системам через входной балансировщик нагрузки.

Программа поддерживает два изолированных контура эксплуатации:

- Предпродуктивный контур (Stage) — для проверки релизов и интеграций.
- Промышленный контур (Prod) — для пользователей и внешних систем.

2.3. Задачи, решаемые программой

Программа решает следующие задачи:

- Управление транспортными операциями (рейсы, маршруты, поставки).
- Проведение аукционов между перевозчиками.
- Бронирование таймслотов разгрузки и погрузки.
- Ведение мастер-данных логистики (адреса, контрагенты, ставки).
- Ведение контрактов с перевозчиками.
- Учёт инцидентов и сервисных заявок.
- Формирование, хранение и обмен электронными транспортными документами.
- Формирование пользовательских и регламентных отчётов, ведение информационных панелей.

- Уведомление пользователей о событиях, отклонениях и регламентных операциях.
- Интеграция со смежными системами заказчика по линиям SAP, ADFS, SMTP, картографических сервисов и брокера сообщений.
- Обеспечение ролевой модели доступа и защита от несанкционированного использования.

2.4. Состав программы и архитектурное окружение

Программа состоит из следующих логических групп компонентов:

- Прикладные микросервисы серверной части (платформа .NET 9) — справочники, мастер-данные, контракты, транспортные операции, аукционы, таймслоты, eTRN, отчёты, уведомления, обработка, импорт/экспорт, файловое хранилище, история, базовые типы, администрирование безопасности, аутентификация, планировщик и иные сервисы согласно манифесту сборки Build/services.yaml.
- Точки входа: пользовательский веб-интерфейс на платформе Vue 3, общий шлюз пользовательского интерфейса (DefaultGateway), интеграционный программный интерфейс (IntegrationApi).
- Платформенные сервисы кластера: входной контроллер балансировщика нагрузки Яндекс Облака, хранилище секретов Vault/OpenBao, брокер интеграционных потоков Kafka Connect, мониторинг состояния потоков cdc-monitor.
- Внешние управляемые сервисы заказчика: управляемый PostgreSQL, управляемый MongoDB, Redis, почтовый сервер SMTP, служба единого входа ADFS, конечные точки Microsoft, конвертер документов Gutenberg (в промышленном контуре), картографические сервисы OSM/Valhalla.
- Сервисные инструменты: инструмент импорта конфигурации ConfigurationImportTool, мигратор схемы истории ProSpace.History.Migrator, средство синхронизации редактируемых наборов OptiTrack.EditablePresetSync, контроллер переключения зон failover-controller.

Архитектурное взаимодействие компонентов приведено в эксплуатационной документации, разделы «Инфраструктура» и «Внешние интеграции».

3. ПОДГОТОВКА ИНФРАСТРУКТУРЫ

3.1. Назначение и ограничения

Данный раздел описывает порядок первичной подготовки инфраструктурного слоя, после которой возможна установка прикладной части Программы:

- Создание именованных областей кластера Kubernetes.
- Установку платформенных пакетов Helm: входного контроллера балансировщика нагрузки, агентов наблюдаемости, хранилища секретов, брокера интеграции, при необходимости — конвертера документов и иных пакетов.
- Создание и загрузку прикладных секретов в Vault.
- Развёртывание мониторингового и картографического стека на выделенных виртуальных машинах.

Раздел не описывает создание самого кластера Kubernetes, управляемых баз данных и почтового сервера — эти ресурсы предоставляются заказчиком или платформой Яндекс Облака.

3.2. Предварительные требования

В общем случае требуется наличие:

1. Учётной записи сервиса Яндекс Облака с правами на управляемый Kubernetes, Container Registry и балансировщик нагрузки.
2. Созданного управляемого кластера Kubernetes; идентификаторы кластера и каталога фиксируются в окруженческой конфигурации:
 - Предпродуктивный контур: идентификатор кластера `catele2r8njpcrd424ep`, идентификатор каталога `blg7msmf8ch01qduu2jb`.
 - Промышленный контур: идентификатор кластера `catn88bhmu2m95bmmdhu`, идентификатор каталога `blg57454rvbbe9sea315`.
 - Созданных управляемых экземпляров PostgreSQL (порт 6432) и MongoDB (порт 27018).
 - Сетевой связности от инфраструктурных и пользовательских узлов до конечных точек:
 - Реестра контейнеров `cr.yandex`.
 - Службы единого входа `adfs.brew4ru.net:443`.
 - Конечных точек Microsoft `*.microsoftonline.com`, `*.microsoft.com`.
 - Почтового сервера `mail.unitedbrew.com:587`.
 - Доступа к репозиторию исходного кода и пакетов GitLab.

- Установленных на рабочем месте инженера инструментов: Ansible, Helm, kubectl, Яндекс Облако CLI (yc), PowerShell 7.
- Конфигурации kubesconfig целевого кластера.

Перечень виртуальных машин мониторинга и картографических сервисов:

Контур	Виртуальная машина мониторинга	Виртуальные машины картографических сервисов
Stage	opti-track-stage-mon-vm / 10.130.10.30	opti-track-stage-osm-vm / 10.130.10.24
Prod	opti-track-prod-mon-vm-b / 10.130.12.31	opti-track-prod-osm-vm-a / 10.130.13.5; opti-track-prod-osm-vm-b / 10.130.12.19

3.3. Подготовка узлов и сервисных машин

Подготовка выполняется средствами Ansible из репозитория optitrack-infra/.

1. Зафиксировать выбор контура в переменной ENVIRONMENT: stage либо prod.
2. При первичной настройке узлов Kubernetes выполнить общий сценарий optitrack-infra/k8s-playbook.yaml. Сценарий применяет роли:
 - Containerd.
 - Helm.
 - Kubernetes.
3. Развернуть мониторинговый стек на выделенной виртуальной машине командой запуска сценария optitrack-infra/monitoring-playbook.yaml. Сценарий применяет роли:
 - Loki.
 - Tempo.
 - Otel.
 - Prometheus.
 - Grafana.
4. Развернуть картографические сервисы командой запуска сценария optitrack-infra/osm-playbook.yaml.

5. Учётные данные доступа по протоколу безопасной оболочки передаются в Ansible через переменные окружения: `ANSIBLE_SSH_MON_USER`, `ANSIBLE_SSH_MON_PRIVATEKEY` (мониторинг), `ANSIBLE_SSH_OSM_USER`, `ANSIBLE_SSH_OSM_PRIVATEKEY` (картография).

Список целевых узлов выбирается из перечня хостов:

- Предпродуктивный контур — `optitrack-infra/configs/stage/inventory.yaml`.
- Промышленный контур — `optitrack-infra/configs/prod/inventory.yaml`.

3.4. Первичная настройка кластера Kubernetes

Первичная настройка выполняется сценарием Ansible `optitrack-clusterinit/ci/ansible/init-cluster.yaml`.

Сценарий последовательно выполняет:

1. Проверку обязательных переменных окружения.
2. Загрузку окруженческих параметров и секретов.
3. Создание именованных областей `optitrack-app`, `optitrack-web`, `optitrack-config`, `optitrack-monitoring`, `optitrack-kafka`, `ingress-alb`.
4. Установку и настройку платформенных пакетов Helm в зависимости от флагов `install_charts`:
 - Входного контроллера балансировщика нагрузки Яндекс Облака.
 - Агента наблюдаемости и экспортёра состояния объектов кластера.
 - Экспортёра метрик PostgreSQL.
 - Хранилища секретов Vault и службы автоматического снятия защитной печати `vault-unseal`.
 - Брокера интеграционных потоков Kafka Connect и пользовательского интерфейса Kafka UI.
 - Конвертера документов Gotenberg (в промышленном контуре).
 - Инициализацию роли приложения в Vault (`AppRole` с именем `app`).
 - Создание ключа сервисной учётной записи входного контроллера.
 - Создание секретов на основе сертификатов: `ingress-tls`, `tls-pfx` в именованных областях прикладных сервисов и веб-слоя.

При необходимости — создание секрета удостоверяющего центра Яндекс для Kafka.

Обязательные переменные окружения сценария:

- CI_PROJECT_DIR — корень репозитория.
- K8S_CONFIGFILE — путь к файлу подключения kubeconfig.
- ENVIRONMENT — выбор контура (stage или prod).

Файлы окруженческих параметров:

- Предпродуктивный контур: optitrack-clusterinit/configs/stage/vars-ci.yaml;
- Промышленный контур: optitrack-clusterinit/configs/prod/vars-ci.yaml.

Особенности контуров:

- В предпродуктивном контуре сценарий устанавливает Vault и выполняет его первичную инициализацию.
- В промышленном контуре Vault уже развёрнут заранее — сценарий выполняет только инициализацию роли приложения для существующего хранилища.
- Конвертер документов Gotenberg по умолчанию выключен в предпродуктивном контуре и включён в промышленном.

3.5. Загрузка секретов в Vault

После инициализации хранилища секретов сценарий выполняет:

- Создание секрета vault-secret в именованных областях optitrack-app, optitrack-web, optitrack-kafka. Секрет содержит поля VAULT_ADDR, VAULT_MOUNT_POINT, VAULT_ROLE_ID, VAULT_SECRET_ID.
- Загрузку наборов прикладных секретов в Vault из шаблонов optitrack-clusterinit/configs/vault-secrets/*.j2.

Загружаемые наборы покрывают:

- Строки подключения к базам данных (ConnectionStrings).
- Параметры планировщика заданий Quartz.
- Сертификаты.
- Настройки источника и приёмника Kafka Connect.
- Входящие и исходящие потоки SAP.
- Параметры наблюдаемости (OpenTelemetry, Loki, Zipkin).
- Параметры конвертера документов Gotenberg.
- Параметры почтового сервера.

- Параметры провайдеров аутентификации Microsoft и OpenIdConnect.
- Параметры соединителя eTRN.
- Параметры картографических сервисов OSM/Valhalla.
- Параметры внешней системы Sbis.

Получение прикладными сервисами значений секретов выполняется автоматически: каждый под получает адрес Vault, точку монтирования, идентификатор и секрет роли из `vault-secret`, а значения читает напрямую из соответствующих путей хранилища ключей.

ВНИМАНИЕ. Значения секретов в простом виде не должны храниться в Markdown-файлах, чатах, переписке. Передача учётных данных, файлов сертификатов в формате PFX, маркеров доступа допускается только через защищённые каналы и хранилища.

4. РАЗВЁРТЫВАНИЕ СЕРВЕРНОЙ ЧАСТИ ПРОГРАММЫ

4.1. Назначение и ограничения

Данная инструкция описывает порядок развёртывания серверной части Программы:

- Настройку параметров развёртывания.
- Проверку файлов развёртывания.
- Сборку и публикацию образов контейнеров.
- Остановку и обновление текущего набора сервисов Программы.
- Запуск нового набора сервисов Программы.
- Проверку работоспособности.

Развёртывание выполняется средствами непрерывной поставки GitLab. Прямые ручные действия с инфраструктурой допускаются только в порядке отладки и при невозможности применить регулярный конвейер.

4.2. Предварительные требования

На момент запуска развёртывания требуется наличие:

1. Завершённой первичной настройки кластера Kubernetes (раздел 3 настоящего документа).
2. Доступа сервисной учётной записи Яндекс Облака к управляемому кластеру и реестру контейнеров.
3. Защищённых переменных GitLab CI:
 - Ключа сервисной учётной записи Яндекс Облака (YC_SERVICE_ACCOUNT_KEY).
 - Идентификаторов облака и каталога (YC_CLOUD_ID, YC_FOLDER_ID).
 - Реквизитов реестра контейнеров.
 - Инструментов в среде запуска заданий: PowerShell 7, Docker, Helm, kubectl, набор средств разработчика .NET 9, Node.js и npm для сборки клиентского интерфейса.
 - Доступа к реестру пакетов GitLab для зависимостей серверной (NuGet) и клиентской (npm) частей.

4.3. Настройка параметров развёртывания

Настройка выполняется через переменные конвейера GitLab CI.

Ключевые переменные конвейера:

Переменная	Назначение
ENVIRONMENT	Выбор контура: stage или prod
BUILD_APP	Включение этапов проверки и сборки прикладной части
DEPLOY_APP	Включение этапа развёртывания прикладной части
DEPLOY_APP_MANUAL	Перевод этапа развёртывания в режим ручного запуска
RELEASE_CRITICALITY	Критичность выпуска: minor, major, critical; передаётся в сборку клиентской части
APP_SERVICES	Список сервисов для сборки и развёртывания
APP_VERSION	Версия для развёртывания (хэш фиксации или релизная метка)
DEPLOY_INTEGRATIONAPI	Включение интеграционного программного интерфейса в состав развёртывания
DEPLOY_EDITABLEPRESETSYNC	Включение средства синхронизации редактируемых наборов
BUILD_CDCMONITOR	Сборка и публикация образа мониторинга потоков Kafka Connect

Окруженческие файлы переменных и наложений значений Helm:

Параметр	Stage	Prod
Файл переменных окружения сборки	Build/.env.stage	Build/.env.prod
Базовые значения Helm для первичной настройки	Deployment/helm-init/values-stage.yaml	Deployment/helm-init/values-prod.yaml
Наложения значений Helm сервисов	Deployment/helm/env/stage/values-*.yaml	Deployment/helm/env/prod/values-*.yaml
Публичный адрес пользовательского интерфейса	tms-stage.brew4ru.net	tms.brew4ru.net
Адрес интеграционного программного интерфейса	tms-integration-stage.brew4ru.net	уточнить у владельца системы
Окружение прикладной платформы (.NET)	Stage	Prod
Идентификатор репозитория в реестре контейнеров	crpra0lmbeqc5pbikmfm	crp1hkvnur2m8jvmarir
Адрес мониторинга	tms-monitoring.brew4ru.net	tms-mon.brew4ru.net
Управляемый PostgreSQL	rc1b-hn37r31ktm4jg4b7.mdb.yandexcloud.net:6432	rc1b-hn37r31ktm4jg4b7.mdb.yandexcloud.net:6432
Управляемый MongoDB	rc1b-an2kcglpumu2uhat.mdb.yandexcloud.net:27018	rc1b-an2kcglpumu2uhat.mdb.yandexcloud.net:27018
Кеш-сервер Redis	redis-service.optitrack-app.svc:6379/6380	redis-service.optitrack-app.svc:6379/6380
Почтовый сервер SMTP	mail.unitedbrew.com:587	mail.unitedbrew.com:587

Состав сервисов и режим их сборки фиксируется в манифесте `Build/services.yaml` и в файле общих параметров сборки `Build/config.yaml`. Прикладной сценарий сборки и развёртывания — `Build/Build.ps1`.

Сценарий `Build/Build.ps1` поддерживает следующие команды:

- `validate` — проверка файлов развёртывания.
- `build` — сборка сервисов.
- `deploy` — развёртывание сервисов.
- `publish` — последовательная сборка и развёртывание.
- `init` — проверка окружения.

Ключевые параметры сценария: `-EnvFile`, `-Services`, `-IncludeServices`, `-Output`, `-Helm`, `-Push`, `-Force`, `-Parallel`, `-NoFrontends`, `-OnlyFrontends`.

4.4. Остановка и удаление текущего набора сервисов программы

Для регламентной остановки и обновления текущего развёртывания необходимо:

1. Зафиксировать текущую версию приложения (значение `APP_VERSION` действующего развёртывания, хэш фиксации или релизная метка) — для возможной операции отката.
2. При необходимости временно вывести входной трафик из обслуживания путём корректировки правил во входном балансировщике нагрузки.
3. В рамках штатного обновления отдельные сервисы не удаляются — этап развёртывания (`deploy-app`) применяет обновлённые манифесты и значения Helm, после чего управляемый Kubernetes автоматически выполняет последовательную замену подов.

Полное удаление прикладного развёртывания выполняется только в порядке вывода контура из эксплуатации:

- Удалить релизы Helm прикладных сервисов в именованных областях `optitrack-app`, `optitrack-web`.
- Убедиться, что платформенные пакеты (входной контроллер, хранилище секретов, брокер интеграции) остаются неизменными.

4.5. Создание и запуск нового набора сервисов программы

4.5.1. Общая последовательность



4.5.2. Подготовка к запуску конвейера

1. Подготовить релизную метку или хэш фиксации для значения переменной APP_VERSION.
2. Перейти к параметризованному запуску конвейера в проекте GitLab.
3. Задать переменные:
 - ENVIRONMENT — stage либо prod;
 - BUILD_APP=true — при необходимости новой сборки;
 - DEPLOY_APP=true — для развёртывания;
 - при необходимости — DEPLOY_INTEGRATIONAPI, DEPLOY_EDITABLEPRESETSYNC, BUILD_CDCMONITOR, APP_SERVICES.

4.5.3. Этап подготовки задания

Общий блок инициализации задания выполняет:

1. Создание профиля интерфейса командной строки Яндекс Облака с именем sa.
2. Раскодирование ключа сервисной учётной записи YC_SERVICE_ACCOUNT_KEY.
3. Настройку идентификаторов облака и каталога (YC_CLOUD_ID, YC_FOLDER_ID).
4. Получение файла подключения kubeconfig командой `yc managed-kubernetes cluster get-credentials --internal`.
5. Создание маркера доступа в систему управления идентификацией и доступом.
6. Авторизацию в реестре контейнеров `cr.yandex`.

4.5.4. Задание validate-app

Задание запускается при `BUILD_APP=true` и проверяет наличие файлов развёртывания и значений Helm.

Команда задания:

```
pwsh Build/Build.ps1 validate `
-EnvFile Build/.env.${ENVIRONMENT} `
-Services "${APP_SERVICES}" `
-IncludeServices "${APP_INCLUDE_SERVICES}" `
-Helm
```

4.5.5. Задание build-app

Задание запускается при `BUILD_APP=true` и выполняет:

- Настройку источника пакетов npm для пространства имён @prospace через реестр пакетов GitLab.
- Добавление источника пакетов NuGet `${CI_API_V4_URL}/projects/${CI_PROJECT_ID}/packages/nuget/index.json`.
- Задание префикса репозитория образов в реестре контейнеров `REGISTRY_REPO="${REGISTRY_HOST/smartcom/optitrack}"`.
- Запуск сборки командой:

```
pwsh Build/Build.ps1 build `
-EnvFile Build/.env.${ENVIRONMENT} `
-Services "${APP_SERVICES}" `
-IncludeServices "${APP_INCLUDE_SERVICES}" `
-Helm -Push -Force
```

Сохранение артефактов сборки:

- Каталога `Build/_work`.
- Шаблонов первичной настройки `Deployment/helm-init/templates`.
- Очистку временных образов контейнеров.

Метки образов:

- Образы прикладных сервисов — `${Build.CommitHash}` (хэш фиксации).
- Образ `cdc-monitor` — `${CI_COMMIT_SHORT_SHA}` и дополнительная метка `latest`.

- Образ контроллера переключения зон — метка задаётся при сборке и передаётся в значения Helm.

4.5.6. Задание build-cdcmmonitor

Задание запускается при `BUILD_CDCMONITOR=true` и публикует образ мониторинга потоков по адресам:

- `${REGISTRY_HOST}/smartcom/optitrack/cdc-monitor:${CI_COMMIT_SHORT_SHA}`.
- `${REGISTRY_HOST}/smartcom/optitrack/cdc-monitor:latest`.

4.5.7. Задание deploy-app

Задание запускается при `DEPLOY_APP=true`. При `DEPLOY_APP_MANUAL=true` задание переводится в режим ручного запуска.

Команда задания:

```
pwsh Build/Build.ps1 deploy `
-EnvFile Build/.env.${ENVIRONMENT} `
-Services "${APP_SERVICES}" `
-IncludeServices "${APP_INCLUDE_SERVICES}" `
-Output Build/_work -Helm
```

Задание применяет подготовленные на этапе сборки манифесты и значения Helm к управляемому кластеру Kubernetes. Повторная сборка контейнеров на этом этапе не выполняется.

Под прикладного сервиса при запуске получает:

- Значение окружения прикладной платформы `ASPNETCORE_ENVIRONMENT` из словаря настроек `env-config`.
- Параметры Vault из секрета `vault-secret`.
- Сертификаты из словаря настроек `infra-certs` и секрета `certs`.
- Учётные данные для получения образов из реестра контейнеров — секрет `gitlab-regcred`.

4.6. Проверка работоспособности

После завершения этапа развёртывания необходимо последовательно проверить:

1. Состояние подов в именованных областях optitrack-app и optitrack-web. Все поды прикладных сервисов и точек входа должны находиться в состоянии «работает» (Running) и иметь признак готовности.

2. Состояние правил входного балансировщика нагрузки:

- Запрос пути /api маршрутизируется на gateway-service:443.
- Запрос пути / маршрутизируется на frontend-service:443.
- При необходимости — путь к разделу документации маршрутизируется на docs-service:9443.
- Правила интеграционного программного интерфейса применяются отдельным шаблоном правил.

3. Доступность публичного адреса контура:

- Предпродуктивный контур: <https://tms-stage.brew4ru.net>.
- Промышленный контур: <https://tms.brew4ru.net>.

4. Состояние интеграционных потоков Kafka Connect (наличие активных соединителей источника и приёмника).

5. Поступление телеметрии в систему наблюдаемости Grafana (журналы Loki, трассировки Tempo, метрики).

5. РАЗВЁРТЫВАНИЕ КЛИЕНТСКОЙ ЧАСТИ ПРОГРАММЫ

5.1. Назначение и ограничения

Клиентская часть Программы представлена пользовательским веб-интерфейсом, выполняющимся в браузере пользователя. Образ контейнера клиентской части собирается и развёртывается в кластере Kubernetes в составе серверной части (раздел 4 настоящего документа) — отдельное развёртывание на стороне пользователя не выполняется.

Данный раздел описывает порядок предоставления пользовательского доступа к развёрнутой клиентской части.

5.2. Предварительные требования

На рабочем месте пользователя требуется:

- Современный браузер с поддержкой стандартов веб-приложений.
- Сетевая связность от рабочего места до публичного адреса контура.
- Учётная запись пользователя в службе единого входа ADFS заказчика и назначенные роли в Программе.

5.3. Настройка параметров развёртывания

Сборка контейнера клиентской части управляется в составе общего сценария сборки. Значение критичности выпуска передаётся в качестве аргумента сборки:

- Значение задаётся переменной конвейера `RELEASE_CRITICALITY` (minor, major, critical).
- Значение прокидывается во внутренний процесс сборки клиентской части и может использоваться в логике, зависящей от уровня выпуска.

Базовый образ для сборки клиентской части использует образ среды веб-сервера из реестра контейнеров: `${Env:REGISTRY_REPO}/nginx:1.21-alpine`.

Пакеты пространства имён `@prospace` загружаются из реестра пакетов GitLab по маркеру `CI_JOB_TOKEN`.

После успешного развёртывания пользовательский доступ предоставляется по публичному адресу контура (см. раздел 4.3 настоящего документа). Аутентификация выполняется через службу единого входа ADFS заказчика.

6. ОПЦИОНАЛЬНО. РАЗВЁРТЫВАНИЕ СЕРВИСОВ НАБЛЮДАЕМОСТИ

6.1. Назначение и ограничения

Данный раздел описывает развёртывание стека наблюдаемости Программы:

- Инструментов стека Grafana: Grafana, Grafana Loki, Grafana Tempo, Prometheus.
- Сборщика телеметрии OpenTelemetry Collector.
- Агента сбора телеметрии Grafana Agent внутри кластера.
- Экспортёра состояния объектов кластера kube-state-metrics.
- Экспортёра метрик управляемого PostgreSQL prometheus-postgres-exporter.
- Мониторинга потоков Kafka Connect cdc-monitor.

Не рассматриваются:

- Настройка пороговых значений и оповещений после установки.
- Настройка прикладных сервисов на отправку телеметрии (выполняется через секреты Vault).

6.2. Предварительные требования

- Завершённая первичная настройка кластера (раздел 3 настоящего документа).
- Доступ инженера по протоколу безопасной оболочки к виртуальной машине мониторинга соответствующего контура.
- Доступ от виртуальной машины мониторинга до службы единого входа ADFS заказчика для авторизации в Grafana по протоколу OAuth.

6.3. Развёртывание инфраструктурного стека наблюдаемости

Развёртывание выполняется сценарием Ansible `optitrack-infra/monitoring-playbook.yaml`. Сценарий применяет роли `loki`, `tempo`, `otel`, `prometheus`, `grafana`.

Образы Grafana и связанных компонентов задаются в переменной `vars-grafana.yaml`. Базовый образ Grafana: `docker.io/grafana/grafana:12.3.0`.

Доступ к Grafana:

- Предпродуктивный контур: `https://10.130.10.30:3000` (рабочий адрес виртуальной машины) и `tms-monitoring.brew4ru.net` (публичный адрес).

- Промышленный контур: <https://tms-mon.brew4ru.net:3000> (публичный адрес).

Доступ Grafana интегрирован со службой единого входа ADFS по протоколу авторизации OAuth; хост службы единого входа — `adfs.brew4ru.net`. Конкретные адреса конечных точек авторизации, выдачи маркера и сведений о пользователе задаются в переменных файла `optitrack-infra/configs/<контур>/vars-grafana.yaml`.

Подготовка наборов источников данных, информационных панелей и сертификатов выполняется ролями каталога `optitrack-infra/roles/grafana/templates/provisioning/` и `optitrack-infra/monitoring/`.

6.4. Развёртывание мониторинга потоков Kafka Connect

Мониторинг потоков `cdc-monitor` устанавливается в составе сценария первичной настройки кластера. Назначение — отслеживание состояния соединителей Kafka Connect и автоматическое восстановление после сбоев.

Базовые параметры предпродуктивного контура:

- Именованная область — `optitrack-kafka`.
- Адрес Kafka Connect — `http://kafka-connect.optitrack-kafka.svc:8083`.
- Настройка соединителей — словарь настроек `kafka-connect-config`.
- Интервал опроса — 30 секунд.
- Максимальное число автоматических перезапусков соединителя — 5.
- Интервал между перезапусками — 60 секунд.
- Автоматическая регистрация соединителей — включена.

Параметры промышленного контура задаются в окруженческих значениях Helm.

6.5. Адреса служб

Назначение	Контур	Адрес или порт
Журналы Loki	Stage / Prod	порт 3100
Трассировки Tempo (совместимый с Zipkin)	Stage / Prod	порт 9411
Сборщик OpenTelemetry	Stage / Prod	порт 9495

Назначение	Контур	Адрес или порт
Grafana	Stage	tms-monitoring.brew4ru.net
Grafana	Prod	tms-mon.brew4ru.net

7. ОПЦИОНАЛЬНО. РАЗВЁРТЫВАНИЕ КОНТРОЛЛЕРА ПЕРЕКЛЮЧЕНИЯ ЗОН

7.1. Назначение и ограничения

Контроллер переключения зон (failover-controller) — служебный компонент, выполняющий управляемое переключение управляемого PostgreSQL и прикладной нагрузки между зонами доступности Яндекс Облака.

Контроллер выполняет последовательную цепочку шагов:

1. Иницирует переключение управляемого PostgreSQL средствами интерфейса командной строки и программного интерфейса Яндекс Облака.
2. Ожидает готовности кластера управляемого PostgreSQL.
3. Применяет обновлённые значения в Vault.
4. Переносит прикладную нагрузку в целевую зону (метка узлов `topology.kubernetes.io/zone`).

В контур контроллера не входит создание кластера PostgreSQL, восстановление данных из резервных копий, сборка прикладных образов и миграции схемы данных.

7.2. Предварительные требования

- Авторизация Яндекс Облака внутри контейнера контроллера (учётные данные доступа к программному интерфейсу управляемого PostgreSQL).
- Секрет с маркером Vault при включённом шаге обновления Vault.
- Секрет с учётными данными почтового сервера при включённых почтовых уведомлениях.
- Роль управления учётными записями кластера, выдаваемая пакетом Helm контроллера; для управления нагрузкой в нескольких именованных областях используется параметр `rbac.clusterWide=true`.

7.3. Развёртывание

Контроллер устанавливается в именованную область `optitrack-failover` пакетом Helm `failover/helm/failover-controller/`.

Ключевые параметры пакета:

- `rbac.clusterWide=true` — управление нагрузкой в именованных областях `optitrack-app` и `optitrack-web`.
- `controller.stateBackend=configmap` — сохранение состояния между перезапусками пода.
- `controller.intervalSeconds=30` — интервал опроса источника решения.
- `controller.cooldownSeconds=900` — окно подавления повторных решений.
- `controller.rolloutTimeoutSeconds=600` — допустимое время ожидания завершения замены подов.
- `controller.zones.labelKey=topology.kubernetes.io/zone` — ключ метки зоны.
- `controller.zones.available` — перечень зон Яндекс Облака.
- `controller.workloads` — перечень контролируемых развёртываний.

Для эталонной конфигурации высокой доступности (`values-dev2ha.yaml`) применяются зоны `ru-central1-a`, `ru-central1-b`, `ru-central1-d`, а перенос нагрузки распространяется на развёртывания в именованных областях `optitrack-app` и `optitrack-web`.

Источник решения задаётся в режимах:

- `static` — ручное решение через значения Helm.
- `manual-file` — решение из файла, смонтированного в контейнер.
- `http` — решение из внешнего программного интерфейса.

Полезная нагрузка решения:

```
{
  "shouldFailover": true,
  "targetZone": "ru-central1-b",
  "reason": "error-rate-threshold",
  "decisionId": "failover-2026-02-17T11:55:00Z"
}
```

Идентификатор решения `decisionId` предотвращает повторное применение одного и того же решения. Значение `targetZone` со значением `auto` или `next` включает автоматический выбор следующей зоны из перечня.

7.4. Безопасность контроллера

Параметры запуска контейнера:

- allowPrivilegeEscalation: false.
- readOnlyRootFilesystem: true.
- runAsNonRoot: true.
- runAsUser: 10001.

8. ОПЦИОНАЛЬНО. РАЗВЁРТЫВАНИЕ ИНТЕГРАЦИОННОГО ПРОГРАММНОГО ИНТЕРФЕЙСА

8.1. Назначение и ограничения

Интеграционный программный интерфейс (IntegrationApi) — точка входа для обмена данными со смежными системами заказчика. Сервис размещается в кластере Kubernetes и доступен по отдельному правилу входного балансировщика нагрузки.

Сервис развёртывается по требованию: ключ `onDemand` в манифесте сборки и переменная конвейера `DEPLOY_INTEGRATIONAPI` управляют его включением в состав развёртывания.

8.2. Развёртывание

Включение сервиса в развёртывание:

1. В параметрах конвейера задать `DEPLOY_INTEGRATIONAPI=true`.
2. Запустить задания `build-app` и `deploy-app` (раздел 4.5 настоящего документа).

Шаблон правил входного балансировщика нагрузки для интеграционного программного интерфейса — `Deployment/helm-init/templates/integrationapiurl-ingress-rules.yaml`.

Адрес сервиса:

- Предпродуктивный контур: `tms-integration-stage.brew4ru.net`.
- Промышленный контур: значение в конфигурации репозитория не зафиксировано; требуется уточнение у владельца системы.

9. ОПЦИОНАЛЬНО. РАЗВЁРТЫВАНИЕ СЕРВИСА КОНВЕРТАЦИИ ДОКУМЕНТОВ

9.1. Назначение и ограничения

Сервис конвертации документов (Gotenberg) преобразует входные документы в требуемые форматы (в частности, формирует файлы PDF). Сервис используется прикладными сервисами через отдельную точку доступа внутри кластера.

В предпродуктивном контуре сервис по умолчанию выключен, в промышленном — включён в составе первичной настройки кластера.

9.2. Развёртывание

Развёртывание выполняется в составе сценария первичной настройки кластера `optitrack-clusterinit/ci/ansible/init-cluster.yaml` при установленном флаге `install_charts.gotenberg` в файле `optitrack-clusterinit/configs/<контур>/vars-ci.yaml`. Для промышленного контура установка включена по умолчанию, для предпродуктивного — выключена.

Значения пакета Helm для промышленного контура задаются в `optitrack-clusterinit/configs/prod/values-gotenberg.yaml`.

ПРИМЕЧАНИЕ. В текущей конфигурации Gotenberg по умолчанию включён для промышленного контура и выключен для предпродуктивного. Если требуется включение сервиса на Stage, источник значений пакета Helm определяется у владельца системы.

Прикладные сервисы получают параметры подключения к Gotenberg из соответствующего набора секретов Vault.

10. ОТКАТ ВЕРСИИ

10.1. Назначение и ограничения

Отдельное задание отката в файле конвейера непрерывной поставки не предусмотрено. Откат к предыдущей рабочей версии выполняется регулярным заданием `deploy-app` с указанием соответствующей метки версии.

10.2. Порядок отката

1. По журналу заданий GitLab или по реестру контейнеров определить метку предыдущего рабочего образа (хэш фиксации либо релизную метку).
2. Запустить параметризованный конвейер:
 - `ENVIRONMENT` — выбранный контур.
 - `APP_VERSION` — метка предыдущего рабочего образа.
 - `DEPLOY_APP=true`.
 - `APP_SERVICES` — перечень сервисов, требующих отката (либо полный состав).
3. Дождаться завершения задания `deploy-app`.
4. Проверить состояние замены подов в управляемом Kubernetes.
5. Выполнить проверку работоспособности (раздел 4.6 настоящего документа), в том числе проверку конечных точек контрольной диагностики и поступление телеметрии в Grafana и Loki.

Приложение А. Перечень сервисов прикладной части

Артефакт	Сервис	Назначение
frontend	Frontend	Пользовательский интерфейс на платформе Vue 3 / TypeScript
gateway	DefaultGateway	Общий шлюз пользовательского интерфейса
integrationapi	IntegrationApi	Интеграционный программный интерфейс (по требованию)
dictionary	ProSpace.Dictionary.Backend	Справочники
dashboard	ProSpace.Dashboard.Backend	Информационные панели и витрины
masterdata	OptiTrack.MasterDataService.Backend	Мастер-данные логистики
contract	OptiTrack.ContractService.Backend	Контракты
transportation	OptiTrack.TransportationService.Backend	Транспортные операции
auction	OptiTrack.AuctionService.Backend	Аукционы перевозчиков
timeslot	OptiTrack.TimeslotService.Backend	Таймслоты
etrn	OptiTrack.ETRN.Backend	Электронные транспортные документы
reports	ProSpace.Reports.Backend	Отчёты

Артефакт	Сервис	Назначение
usersetting	ProSpace.UserSetting.Backend	Пользовательские настройки
integration	ProSpace.Integration.Backend	Интеграционные операции
collector	ProSpace.Collector.Backend	Сбор данных
processing	ProSpace.Processing.Backend	Обработка данных
securityadmin	ProSpace.SecurityAdmin.Backend	Управление безопасностью и пользователями
authservice	AuthService	Аутентификация
quartz	ProSpace.Scheduler.Service.Quartz	Исполнитель планировщика заданий Quartz
scheduler	ProSpace.Scheduler.Backend	Программный интерфейс планировщика
passwordmanagment	ProSpace.PasswordManagement.Backend	Управление паролями
notification	ProSpace.Notification.Backend	Уведомления
notificationsender	ProSpace.Notification.Sender	Отправка уведомлений
importer	ProSpace.Importer.Backend	Импорт данных
filestore	ProSpace.FileStore.Backend	Файловые операции

Артефакт	Сервис	Назначение
exporter	ProSpace.Exporter.Backend	Экспорт данных
basetype	BaseTypeService.Backend	Базовые типы и конфигурации
history	ProSpace.History.Backend	Историзация

ПРИМЕЧАНИЕ. Сервис workflow (ProSpace.Workflow.Backend) в конфигурации репозитория отключён (enabled: false в Build/services.yaml).

Приложение Б. Перечень служебных контейнеров

Компонент	Назначение	Конфигурация
ConfigurationImportTool	Импорт конфигурации, развёртывается в составе каждого выпуска	Tools/ConfigurationImportTool /, Deployment/tools/configurationimport.yaml
ProSpace.History.Migrator	Миграции схемы хранилища истории	Application/Services/History/ProSpace.History.Migrator, Deployment/tools/historymigrator.yaml
cdc-monitor	Мониторинг и автоматическое восстановление соединителей Kafka Connect	Deployment/helm-init/templates/cdc-monitor.yaml
OptiTrack.EditablePresetSync	Синхронизация редактируемых наборов после развёртывания	Tools/editable-preset-sync/
failover-controller	Управляемое переключение зон	failover/ failover/helm/failover-controller/

Лист регистрации изменений

[illegible]