

Retail Ocean
ASSORTMENT
BY GLOWBYTE

Инструкция по установке экземпляра программного обеспечения «Инструмент по управлению квотированием полочного пространства и ассортимента», предоставленного для проведения экспертной проверки

Москва, 2026

Оглавление

Введение **4**
Дистрибутив 4
Доступ к репозиторию через web интерфейс 4
Дистрибутив для проведения экспертной проверки 5
Демонстрационный стенд с развернутым экземпляром ПО 5
Контакты технического специалиста 5
Содержание поставки **5**
Системные требования к ПО «Инструмент по управлению квотированием полочного пространства и ассортимента» **7**
Архитектура и компоненты ПО **7**
Компоненты ПО 7
Архитектура 8
Пререквизиты **9**
Kubernetes 10
PostgreSQL 11
Keycloak 11
Установка **12**
Доступ к репозиторию образов 12
Проверка наличия образов в registry 12
Доставка образов на стенд 13
Порядок установки 14
Подготовка PostgreSQL 14
Настройка Keycloak 16
Конфигурация стенда 17
Первый запуск 24
Настройка путей Ingress 25
Проверка корректности установки **25**
Чек-лист приёмки стенда 27
Обновление системы **28**
Порядок обновления 28
Когда можно перезапускать calc и ex 29
Откат 29
Изменение настроек 30

Регламент калькулятора 30
Масштабирование 30
Возможные проблемы **30**
Логи 32
Приложение. Листинги скриптов и конфигураций **33**

Введение

Данная версия документа подготовлена для проведения экспертной проверки установки экземпляра ПО.

Документ предназначен для системных администраторов, которые разворачивают экземпляр ПО (стенд) в существующем кластере Kubernetes. В нём описано, из каких образов состоит ПО и где они находятся, как доставить образы на стенд, что подготовить в окружении (PostgreSQL, Keycloak, хранилище обмена, TLS), какие конфигурационные файлы заполнить, как запустить, проверить и обновить ПО. Сборка образов из исходных текстов в документе не рассматривается: образы считаются готовыми и опубликованными в репозитории.

Инструкция актуальна для версий модулей: api `3.0.1.7`, calculator `3.0.0.18`, exchange `3.0.0.1`, ui `3.0.1.4`.

Дистрибутив

Установка ПО «Инструмент по управлению квотированием полочного пространства и ассортимента» осуществляется методом развертывания на кластере Kubernetes с использованием манифестов kustomize (встроен в kubectl, команда `kubectl apply -k`). Образы для установки находятся в репозитории https://retailocean.datasapience.ru/repo/

Доступ к репозиторию через web интерфейс

Учетная запись для доступа к репозиторию из web интерфейса по запросу.

Образы для установки экземпляра ПО расположены в проекте:

- https://retailocean.datasapience.ru/repo/

Дистрибутив для проведения экспертной проверки

Демонстрационный стенд с развернутым экземпляром ПО

Адрес демонстрационного стенда и учетная запись для входа: https://retailocean.datasapience.ru

Доступы по запросу.

Контакты технического специалиста

По запросу.

Содержание поставки

В поставку входят:

- Образы четырех модулей ПО: api, calculator, exchange, ui
- Манифесты Kubernetes (kustomize) для развертывания стенда
- SQL-скрипты подготовки базы данных
- Скрипт копирования образов в registry стенда или в tar-архив

Образы находятся в репозитории https://retailocean.datasapience.ru/repo/:

Модуль	**Образ в registry**	**Размер**
api	`https://retailocean.datasapience.ru/repo/retailocean-assortment-api-20260928-2661eab.tar.gz`	~70 МБ
calculator	`https://retailocean.datasapience.ru/repo/retailocean-assortment-calculator-20260928-490920c.tar.gz`	~150 МБ
exchange	`https://retailocean.datasapience.ru/repo/retailocean-assortment-exchange-20260928-9d0541e.tar.gz`	~55 МБ
ui	`https://retailocean.datasapience.ru/repo/retailocean-assortment-ui-20260928-31cb6b5.tar.gz`	~10 МБ

Образы публикуются в registry при выпуске релиза. Registry — единственный источник образов для стендов: все варианты доставки образов (раздел «Доставка образов на стенд») берут образы из него.

После установки контейнеров рекомендуется залить в ваш локальный docker registry - далее по текста your.local.registry

Номер релиза системы общий (`3.0.1.7`), но **в релиз входят только изменённые модули**. Поэтому тега `3.0.1.7` у calc, ex и ui нет. Стенд описывается набором из четырёх тегов, по одному на модуль. Актуальный набор (последний выпущенный тег каждого модуля):

Модуль	**Тег**	**Дата релиза**
api	`3.0.1.7`	2026-09-08

Модуль	Тег	Дата релиза
calc	`3.0.0.18`	2026-08-18
ex	`3.0.0.1`	2026-08-08
ui	`3.0.1.4`	2026-08-28

Этот набор уже записан в `k8s/stands/example/config/kustomization.yaml` (блок `images`).

Состав архива **retailocean-k8s-deploy-guide.zip**:

```
k8s/
  base/                     общие манифесты, одинаковые для всех стендов
    api.yaml                Deployment api + Service assort-api-service:8081
    calc.yaml               Deployment calc (1 реплика, Recreate)
    ex.yaml                 Deployment ex (1 реплика, Recreate) + initContainer с
каталогами
    ui.yaml                 Deployment ui + Service assort-ui-service:8080
  jobs/
    kustomization.yaml
    db-migrate.yaml         Job миграций БД (alembic upgrade head)
  stands/example/           ОБРАЗЕЦ стенда: копируется в stands/<стенд>/
    kustomization.yaml      namespace, состав стенда
    ingress.yaml            вход https://<хост>/ -> ui
    storage-exchange.yaml   PV/PVC каталогов обмена (NFS или hostPath)
    config/                 набор версий образов и все конфиги модулей
      kustomization.yaml    теги образов + генерация ConfigMap
      common.env api.env calc.env ex.env ui-config.js schedule.json
    migrate/
      kustomization.yaml    Job миграций с тегом api и ConfigMap-ами этого стенда
sql/
  01-roles-and-database.sql роли и база стенда
  02-seed-first-admin.sql   первый администратор системы
scripts/
  mirror-images.sh          копирование набора образов в registry стенда или в tar
```

Системные требования к ПО «Инструмент по управлению квотированием полочного пространства и ассортимента»

Требования к ресурсам одного стенда (сумма стартовых ресурсов модулей, заданных в `k8s/base/*.yaml`; уточняются по мониторингу):

Параметр	Значение
Процессор (requests)	Не менее 1,6 ядра

Параметр	Значение
Оперативная память	requests — не менее 3,1 ГиБ; limits — до 8,3 ГиБ
Место на узлах кластера под образы	Около 1 ГБ
Хранилище обмена	50 ГиБ в образце (`storage-exchange.yaml`), по объёму обмена
Kubernetes	Версия 1.27 и выше
PostgreSQL	Версия 13 и выше, отдельная база под стенд
Соединения с PostgreSQL	Около 115 на стенд с одной репликой api
Браузер пользователя	Доступ к стенду только по HTTPS с доверенным сертификатом

Архитектура и компоненты ПО

Компоненты ПО

- **api**. HTTP API `/api/v1`, проверка JWT, владелец схемы БД. Deployment `api`, Service `assort-api-service` (порт контейнера 8000, порт Service 8081), 1 и более реплик.
- **calculator**. Берёт задачи из `data.process_log`, выполняет расчёты, ведёт регламент фоновых задач. Deployment `calc`, строго 1 реплика.
- **exchange**. Импорт и экспорт CSV через каталоги обмена, импорт остатков и продаж по DB-link. Deployment `ex` и PVC каталогов обмена, строго 1 реплика.
- **ui**. Отдаёт веб-приложение (SPA); nginx внутри контейнера проксирует `/api/` на api. Deployment `ui`, Service `assort-ui-service` (порт 8080), Ingress, 1 и более реплик.
- **Миграции БД**. Разовый Job `assort-db-migrate`: `alembic upgrade head` образом api.

Архитектура

Архитектура решения представлена на схеме ниже.

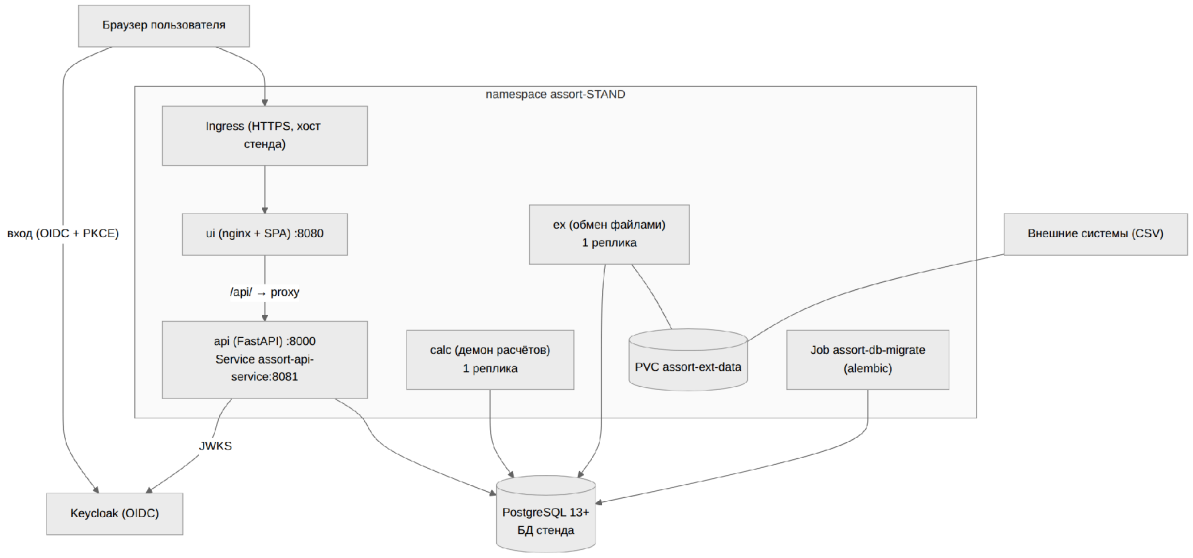


Рисунок 1. Состав системы в Kubernetes

Ключевые свойства, из которых вытекают решения в манифестах:

- **Модули не вызывают друг друга по сети.** Вся координация идёт через общую БД (очередь `data.process_log`, outbox `"int".export_data`). UI ходит только в api, и только через nginx собственного контейнера.
- **Один стенд — один namespace — одна база.** Два calc или два ex на одной базе ломают данные (дубли регламентных задач, гонки за файлы и staging-таблицы).
- **Вход один — Ingress на ui.** У api нет своего публичного адреса: снаружи к нему ходят как `https://<хост>/api/v1/...`.
- **calc и ex — фоновые демоны без HTTP.** Порт 8000 в их образах (`EXPOSE`) ничего не слушает, Service им не нужен.

Пререквизиты

Описанный в настоящем документе процесс установки предполагает выполнение следующих пререквизитов:

Что	Требование	Примечание
Кластер Kubernetes	1.27+, выделенный namespace под стенд	Ресурсы на стенд — раздел «Kubernetes»
kubectl	1.27+ (встроенный kustomize с поддержкой `components`)	На машине администратора

Что	Требование	Примечание
Ingress-контроллер	Любой, с TLS-терминацией	Класс указывается в `ingress.yaml`
DNS-имя стенда	Например `assort-qa.example.org`	На него же настраивается redirect URI в Keycloak
TLS-сертификат	**Доверенный браузерами** пользователей (корпоративный УЦ или публичный)	Без HTTPS вход не работает вовсе, с недоверенным — белый экран (раздел «Keycloak»)
PostgreSQL	13+, вне кластера или в нём	Отдельная база под стенд, раздел «Подготовка PostgreSQL»
Keycloak	Realm и клиент для UI стенда	Раздел «Настройка Keycloak». Может быть общим для нескольких стендов
Хранилище обмена	Каталог, доступный и поду `ex`, и внешним системам (NFS и т.п.)	Раздел «exchange — config/ex.env и хранилище»
Доступ к образам	Учётка на чтение в `https://retailocean.datasapience.ru/repo/` или локальное зеркало	Раздел «Доставка образов на стенд»
Эталонная БД	Дамп эталонной БД (стенд dev или test)	Схема стенда создаётся только из дампа, раздел «Наполнение базы из дампа»

Для начала работы на машине администратора необходимо установить стороннее ПО:

- kubectl версии 1.27 и выше — применение манифестов;
- клиент PostgreSQL (`psql`, `pg_dump`, `pg_restore`) — подготовка базы данных;
- skopeo или docker — проверка наличия образов и копирование их в registry стенда;
- curl и python3 — проверка работоспособности и формата конфигурационных файлов.

Условия использования стороннего ПО: kubectl, skopeo, docker и Keycloak распространяются на основании открытой лицензии Apache License, версия

2.0; PostgreSQL и его клиентские утилиты — на основании открытой лицензии PostgreSQL License.

Для установки «Инструмент по управлению квотированием полочного пространства и ассортимента» необходимо подготовить следующие компоненты:

Kubernetes

Манифесты собираются встроенным в kubectl kustomize (`kubectl apply -k`). Под стенд выделяется отдельный namespace.

Стартовые ресурсы подов (заданы в `k8s/base/*.yaml`, уточняются по мониторингу):

Под	**requests**	**limits**	**Комментарий**
api	250m / 512Mi	1Gi	На реплику до 20 соединений с БД
calc	1 CPU / 2Gi	6Gi	До 10 параллельных расчётов (pandas/sklearn), каждый — отдельный процесс
ex	250m / 512Mi	1Gi	
ui	50m / 64Mi	256Mi	

Прим. 1000m = 1 CPU

Сетевые доступы:

Откуда	**Куда**	**Порт**	**Зачем**
Браузер пользователя	Ingress стенда	443	UI и `/api/`
Браузер пользователя	Keycloak	443	Вход (браузер ходит в Keycloak напрямую, не через ui)
Под api	Keycloak	443 или 8080	JWKS (ключи проверки токенов)
Поды api, calc, ex, Job миграций	PostgreSQL	5432	Работа с БД
Узлы кластера	Registry	443	Скачивание образов

Откуда	Куда	Порт	Зачем
Внешние системы	Хранилище обмена	по типу шары	Выкладка и забор CSV

PostgreSQL

PostgreSQL версии 13 и выше используется всеми модулями как общая база данных стенда: через неё идёт вся координация модулей. Под каждый стенд создаётся отдельная база. Сервер может располагаться как вне кластера, так и в нём. Подготовка базы описана в разделе «Подготовка PostgreSQL».

Keycloak

Keycloak используется для входа пользователей по протоколу OIDC.

- **ui** (браузер) проводит вход по OIDC Authorization Code + PKCE (библиотека `oidc-client-ts`). Адрес Keycloak, realm и client id берутся из `config.js` (раздел «ui — config/ui-config.js»). Браузер ходит в Keycloak **напрямую**.
- **api** проверяет подпись токена по JWKS `<KEYCLOAK_SERVER_URL>/realms/<KEYCLOAK_REALM>/protocol/openid-connect/certs`. Проверяются только подпись (RS256) и срок действия. Issuer и audience не проверяются, client id и secret api не нужны.
- **Роли Keycloak не используются.** Права задаются в БД системы (`data.user_role_assignment`). Keycloak отвечает только на вопрос «кто это». Имя пользователя (`preferred_username`) должно совпадать с `data.user_list.user_login`, иначе api отвечает 403 «Your account is not registered in the application».

Требования к сертификатам и адресам:

- **Браузер** должен доверять сертификату Keycloak. Если сертификат не принят, UI показывает пустой белый экран без сообщения.
- **api**: если `KEYCLOAK_SERVER_URL` начинается с `https://`, при старте пода скрипт `install_keycloak_cert.py` подключается к Keycloak, скачивает предъявленную цепочку сертификатов и добавляет её в доверенные. Это сделано под самоподписанные сертификаты. Если Keycloak недоступен, после 5 попыток под уходит в CrashLoopBackOff. Отсюда

требование: **Keycloak должен быть доступен раньше, чем стартует api**.

- Если Keycloak работает в том же кластере, api можно направить на внутренний адрес (`KEYCLOAK_SERVER_URL=http://keycloak.keycloak.svc:8080`). Браузер при этом ходит на внешний `https://`. Это работает, потому что api не сверяет issuer. Заодно пропадает зависимость старта от сертификата.
- `KEYCLOAK_VERIFY_SSL=false` отключает проверку TLS при запросе JWKS. Использовать только как временную диагностику.

Установка

Доступ к репозиторию образов

Проверка наличия образов в registry

Какие теги каждого модуля есть в registry:

```
for m in api calc ex ui; do
  echo "== $m"
  skopeo list-tags --creds "$REG_USER"
docker://your.local.registry/assortiment/assort/$m \
    | python3 -c 'import json, sys; print(*json.load(sys.stdin)["Tags"])'
done
```

Проверка, что теги набора стенда есть в registry:

```
# Вариант 1: skopeo (не требует docker)
for img in api:3.0.1.7 calc:3.0.0.18 ex:3.0.0.1 ui:3.0.1.4; do
  skopeo inspect --creds "$REG_USER" --format "$img {{.Digest}}" \
    docker://your.local.registry/assortiment/assort/$img
done

# Вариант 2: docker (после docker login your.local.registry)
docker manifest inspect your.local.registry/assortiment/assort/api:3.0.1.7
>/dev/null && echo ok
```

Теги в registry перезаписываемы. Для стенда, который должен точно повторять другой, запишите digest каждого образа (вывод skopeo выше) и сверяйте его, а не только тег.

Доставка образов на стенд

Выберите один вариант. Манифесты от выбора почти не зависят.

Вариант A: узлы кластера ходят в your.local.registry напрямую

Нужна учётка на чтение (желательно robot-аккаунт). Секрет с доступом создаётся в namespace стенда под именем `registry-retailocean`, на него уже ссылаются все манифесты (`imagePullSecrets`):

```
NS=assort-qa
# Отдельный конфиг docker, чтобы в секрет не попали чужие учётки и credsStore
docker --config /tmp/regcfg login your.local.registry
kubectl -n "$NS" create secret generic registry-retailocean \
  --type=kubernetes.io/dockerconfigjson \
  --from-file=.dockerconfigjson=/tmp/regcfg/config.json
rm -rf /tmp/regcfg
```

Вариант B: зеркало в registry стенда

С машины, у которой есть доступ к обоим registry:

```
docker login your.local.registry
docker login registry.stand.local
scripts/mirror-images.sh \
  --tags api=3.0.1.7,calc=3.0.0.18,ex=3.0.0.1,ui=3.0.1.4 \
  --to registry.stand.local/assort
```

Скрипт печатает digest каждого образа. В `config/kustomization.yaml` стенда к каждому образу добавляется `newName`:

```
images:
  - name: your.local.registry/assortiment/assort/api
    newName: registry.stand.local/assort/api
    newTag: "3.0.1.7"
  # ... так же для calc, ex, ui
```

Секрет `registry-retailocean` создаётся так же, как в варианте A, но для registry стенда. Если registry стенда анонимный, секрет не нужен (ссылка на отсутствующий секрет не мешает скачиванию).

Вариант C: закрытый контур без доступа к registry

Образы переносятся файлом и импортируются в containerd **каждого узла**, на котором могут запуститься поды стенда:

```
# 1. Там, где есть доступ к your.local.registry
scripts/mirror-images.sh \
  --tags api=3.0.1.7,calc=3.0.0.18,ex=3.0.0.1,ui=3.0.1.4 \
  --save assort-images.tar

# 2. На каждом узле кластера (после копирования файла)
sudo ctr -n k8s.io images import assort-images.tar     # containerd (kubeadm и
т.п.)
sudo k3s ctr images import assort-images.tar           # k3s
microk8s images import < assort-images.tar             # microk8s

# 3. Проверка на узле
sudo ctr -n k8s.io images ls | grep assortiment/assort
```

Имена образов в tar-файле остаются `your.local.registry/...`, поэтому манифесты не меняются. `imagePullPolicy: IfNotPresent` не даст kubelet ходить в registry, пока образ есть на узле.

Порядок установки

Порядок важен: схема БД идёт первой, остальные модули рассчитывают на готовую схему.

1. Подготовка PostgreSQL: роли, база, восстановление из дампа.
2. Настройка Keycloak: realm, клиент стенда, пользователь-администратор.
3. Конфигурация стенда: оверлей, секреты, конфигурационные файлы модулей.
4. Первый запуск: миграции, первый администратор, выкат модулей.
5. Проверка — раздел «Проверка корректности установки».

Подготовка PostgreSQL

Роли и база

Схема владения: одна учётка на все модули. Роль `assortment` владеет всеми объектами, учётка `retailocean_user` входит в неё. Роль `assortment` обязательна: на неё ссылается одна из миграций api (`ALTER FUNCTION ... OWNER TO assortment`). Права владельца модулям действительно нужны: calc пересоздаёт таблицы в схеме `calc`, exchange пересоздаёт триггер и функции в схеме `"int"`, и оба делают `TRUNCATE` staging-таблиц.

```
psql -h <db-host> -U postgres -v stand_db=retailocean_assortment_qa \
     -f sql/01-roles-and-database.sql      # спросит пароль для retailocean_user
```

Пароль БД — без символов `@ : / # %`. api, calc и ex собирают строку подключения без URL-кодирования. С такими символами модули не подключатся к БД, а healthcheck calc и ex при этом может оставаться зелёным.

Бюджет соединений (`max_connections` сервера) на один стенд:

Потребитель	Максимум соединений
api	`(DB_POOL_SIZE + DB_MAX_OVERFLOW) × реплики` = 20 на реплику
calc	`(DB_POOL_SIZE + DB_MAX_OVERFLOW) × (MAX_CONCURRENT_PROCESSES + 1)` = 88
ex	~5

Итого около 115 на стенд с одной репликой api. Добавьте запас на администрирование и другие базы того же сервера.

Наполнение базы из дампа

Штатные миграции не создают схему БД с нуля: на пустой базе `alembic upgrade head` завершается ошибкой. Схема нового стенда восстанавливается из дампа эталонной БД, после чего миграции доводят её до версии api стенда (раздел «Первый запуск», шаг 4).

Источник дампа — эталонная БД (стенд dev или test), в которой `alembic current` показывает известную ревизию. Продуктивная БД в качестве источника не используется.

Схема `assort` (источник остатков и продаж для DB-link, раздел «exchange — config/ex.env и хранилище») в дамп не включается. Если на стенде нужен импорт по DB-link, её настраивает DBA отдельно.

Вариант 1. Полная копия (данные как на исходном стенде). Надёжный вариант, но `data.sale_history` и `data.stock` занимают сотни миллионов строк:

```
pg_dump -h <src-host> -U <src-user> -d retailocean_assortment_test \
        -Fc --no-owner --no-privileges --exclude-schema=assort \
        -f assort-full.dump

pg_restore -h <db-host> -U postgres -d retailocean_assortment_qa \
           --no-owner --no-privileges --role=assortment -j 4 \
           assort-full.dump
```

Вариант 2. Схема и справочники без боевых данных (пустой стенд, наполняется импортом через exchange):

```
pg_dump -h <src-host> -U <src-user> -d retailocean_assortment_test \
        --schema-only --no-owner --no-privileges --exclude-schema=assort \
        -f assort-schema.sql

pg_dump -h <src-host> -U <src-user> -d retailocean_assortment_test \
        --data-only --no-owner --no-privileges \
        -t data.alembic_version \
        -t data.tref_role -t data.tref_capability -t data.tref_page -t
data.tref_code \
        -t data.role_capability_assignment -t data.page_role_permission \
        -t data.page_capability_assignment -t data.status_capability_assignment \
        -t data.parameter_list \
        -f assort-reference.sql

psql -h <db-host> -U retailocean_user -d retailocean_assortment_qa -v
ON_ERROR_STOP=1 \
     -c 'SET ROLE assortment' -f assort-schema.sql -f assort-reference.sql
```

Таблица `data.alembic_version` обязательна: без неё миграции попытаются начать с нуля. Остальные данные (магазины, товары, иерархии) загружаются импортом.

Общие замечания:

- `pg_dump` должен быть не старше версии исходного сервера, целевой сервер — не старше исходного.
- Ошибки восстановления про расширения, внешние серверы или user mapping означают, что в исходной БД есть объекты DB-link. Их настраивает DBA, в стенд они не переносятся.

Миграции накатывает Job из образа api (раздел «Первый запуск», шаг 4). Первого пользователя заводят после миграций (раздел «Первый запуск», шаг 5), потому что пользователи в системе не создаются автоматически.

Настройка Keycloak

Keycloak может быть общим для нескольких стендов: достаточно отдельного клиента на каждый стенд (у каждого свой redirect URI). Пример на `kcadm.sh` (realm `ASSORT`, клиент `assort-ui-qa`, хост `assort-qa.example.org`):

```
KC=/opt/keycloak/bin/kcadm.sh
UI=https://assort-qa.example.org

$KC config credentials --server https://sso.example.org --realm master --user admin

# realm (если его ещё нет)
$KC create realms -s realm=ASSORT -s enabled=true -s sslRequired=external

# публичный клиент для UI стенда: Standard Flow + PKCE S256
$KC create clients -r ASSORT \
  -s clientId=assort-ui-qa \
```

```
  -s publicClient=true \
  -s standardFlowEnabled=true \
  -s directAccessGrantsEnabled=false \
  -s implicitFlowEnabled=false \
  -s "redirectUris=[\"$UI/\"]" \
  -s "webOrigins=[\"$UI\"]" \
  -s 'attributes."pkce.code.challenge.method"=S256' \
  -s "attributes.\"post.logout.redirect.uris\"=$UI/"

# пользователь (логин — в нижнем регистре, он же user_login в БД)
$KC create users -r ASSORT -s username=ivanov.i -s enabled=true \
  -s firstName=Иван -s lastName=Иванов -s email=ivanov.i@example.org
$KC set-password -r ASSORT --username ivanov.i --temporary
```

Требования к клиенту, если его настраивают через консоль: тип Public, Standard Flow, PKCE S256; Valid Redirect URIs и Valid Post Logout Redirect URIs — `https://<хост>/` (с завершающим `/`); Web Origins — `https://<хост>`. UI сам вычисляет redirect URI как `window.location.origin + '/'`, поэтому адрес в Keycloak должен совпадать с адресом стенда посимвольно.

Конфигурация стенда

Создание оверлея

```
cp -r k8s/stands/example k8s/stands/qa
grep -rn CHANGE_ME k8s/stands/qa        # всё найденное заменить
```

Что заполняется в каталоге стенда:

Файл	**Что задаёт**	**Меняется между стендами**
`kustomization.yaml`	namespace, метка стенда	да
`migrate/kustomization.yaml`	тот же namespace	да
`ingress.yaml`	хост, класс Ingress, TLS-секрет	да
`storage-exchange.yaml`	где лежат каталоги обмена	да
`config/kustomization.yaml`	**набор версий образов** (и `newName` для зеркала)	по ситуации
`config/common.env`	адрес БД, TZ	да
`config/api.env`	режим, Keycloak, пул	Keycloak — да
`config/calc.env`	таймауты и параллелизм расчётов	обычно нет
`config/ex.env`	каталоги и расписание обмена, DB-link	DB-link — по ситуации

Файл	Что задаёт	Меняется между стендами
`config/ui-config.js`	адрес Keycloak, realm, client id для браузера	да
`config/schedule.json`	регламент калькулятора	обычно нет

Секреты в файлах оверлея не хранятся. Они создаются командами (раздел «Секреты»).

Всё, что лежит в `config/`, превращается в ConfigMap с хеш-суффиксом в имени. Правка любого `.env` или `ui-config.js` и `kubectl apply -k` автоматически пересоздают поды затронутых модулей. Исключение — `schedule.json`: он без суффикса и подхватывается калькулятором на лету (раздел «Регламент калькулятора»).

Секреты

Секрет	Тип	Содержимое	Кто использует
`registry-retailocean`	dockerconfigjson	доступ к registry	все поды (раздел «Доставка образов на стенд»)
`assort-db`	Opaque	`DB_USER`, `DB_PASSWORD`	api, calc, ex, Job миграций
`assort-tls`	kubernetes.io/tls	сертификат и ключ хоста стенда	Ingress

```
NS=assort-qa
kubectl create namespace "$NS"

read -rsp 'Пароль retailocean_user: ' DB_PASSWORD; echo
kubectl -n "$NS" create secret generic assort-db \
  --from-literal=DB_USER=retailocean_user \
  --from-literal=DB_PASSWORD="$DB_PASSWORD"
unset DB_PASSWORD

kubectl -n "$NS" create secret tls assort-tls \
  --cert=assort-qa.fullchain.pem --key=assort-qa.key.pem
```

Отдельная учётка для миграций. Если политика БД запрещает модулям права владельца, создайте Secret `assort-db-migrator` с учёткой владельца схем. В `k8s/jobs/db-migrate.yaml` замените `secretRef` на него. Учётке модулей тогда выдайте права:

Модуль	Права
api	CRUD в схемах `data`, `calc`, `"int"`; использование sequences; TEMP
calc	владение схемой `calc` (DROP/CREATE таблиц); TEMP
ex	TRUNCATE staging-таблиц; DDL триггера и функций в схеме `"int"`; TEMP

api — config/api.env (+ common.env, assort-db)

Переменная	Значение в образце	Обяз.	Стенд	Смысл
`ENVIRONMENT`	`production`	**да**	нет	`local` | `staging` | `production`; другое значение — api не стартует. **Без переменной — `local`: проверка токенов и прав выключена.** `staging` — авторизация есть, но без пула соединений
`DB_HOST`, `DB_PORT`, `DB_NAME`	из `common.env`	да	да	Адрес БД стенда
`DB_USER`, `DB_PASSWORD`	из Secret `assort-db`	да	да	Учётка БД
`KEYCLOAK_SERVER_URL`	`https://sso.example.org`	да	да	Корень Keycloak, доступный из пода (раздел «Keycloak»). Умолчание `http://keycloak:8080`
`KEYCLOAK_REALM`	`ASSORT`	да	да	Realm. Умолчание `DEV`
`KEYCLOAK_VERIFY_SSL`	`true`	нет	нет	Проверка TLS при запросе JWKS
`KEYCLOAK_TIMEOUT`, `KEYCLOAK_CACHE_TTL`	5 / 300	нет	нет	Таймаут JWKS и время жизни кэша, с
`KEYCLOAK_CERT_FETCH_RETRIES` / `_RETRY_DELAY` / `_TIMEOUT`	5 / 3 / 10	нет	нет	Скачивание сертификата Keycloak при старте
`LOG_LEVEL`	`INFO`	нет	нет	`DEBUG` пишет весь SQL с параметрами — не на общих стендах

Переменная	Значение в образце	Обяз.	Стенд	Смысл
API_V1_STR	/api/v1	нет	нет	Префикс API. UI ожидает /api/v1, не менять
DB_POOL_SIZE, DB_MAX_OVERFLOW	10 / 10	нет	нет	Пул на реплику (только при production)
THREADPOOL_SIZE	20	нет	нет	Должно быть ≤ DB_POOL_SIZE + DB_MAX_OVERFLOW
DB_POOL_TIMEOUT	30	нет	нет	Ожидание соединения из пула, с
AUDIT_ENABLED	true	нет	нет	Журнал действий пользователей в data.user_audit_log

В шаблонах модуля встречаются переменные, которые в k8s не нужны, потому что код их не читает: KEYCLOAK_CLIENT_ID, API_CONTAINER_NAME, API_HOST_PORT, APP_ROOT, VENV_PATH, SSH_*, REMOTE_PROJECT_PATH, DATABASE_URL.

calculator — config/calc.env и config/schedule.json

Переменная	Значение в образце	Стенд	Смысл
DB_*	common.env + assort-db	да	Без любого из них демон работает вхолостую: DB parameters not fully configured
PROCESS_TIMEOUT	72000	нет	Жёсткий таймаут задачи, с. Рекомендуемое значение 72000 (20 ч). Умолчание 3600 прерывает долгие расчёты
MAX_CONCURRENT_PROCESSES	10	по ресурсам	Параллельных расчётов. Меньше — меньше памяти и соединений
MAIN_LOOP_INTERVAL	5	нет	Период опроса очереди, с
SCHEDULE_FILE_PATH	/etc/assort-calc/schedule.json	нет	Путь к регламенту из ConfigMap. Не менять без правки calc.yaml

Переменная	Значение в образце	Стенд	Смысл
`AUDIT_RETENTION_DAYS`	180	нет	Срок хранения журнала аудита (чистит calc). Применяется после рестарта
`TZ`	`common.env`	да	Регламент считается по часам контейнера, без TZ — UTC
`TASK_SHUTDOWN_TIMEOUT_SECONDS`	30 (умолч.)	нет	Дедлайн остановки. Увеличили — увеличьте `terminationGracePeriodSeconds`
`DB_POOL_SIZE`, `DB_MAX_OVERFLOW`	3 / 5 (умолч.)	нет	Пул **на каждый процесс** расчёта
`SALE_HISTORY_THRESHOLD_DAYS`, `STOCK_THRESHOLD_DAYS`	90 (умолч.)	нет	Глубина хранения продаж и остатков (чистят регламентные задачи)

`schedule.json` — регламент фоновых задач. В образце лежит копия из образа calc `3.0.0.18`:

Задача	Когда
`prc_process_timeout_guard`	каждые 5 минут
`prc_substitute_review`	ежедневно 00:01
`prc_user_audit_cleanup`	ежедневно 01:00
`prc_segments`	суббота 09:00
`prc_sale_history_cleanup`, `prc_stock_cleanup`	суббота 14:00
`prc_submatrices_cleanup`	воскресенье 01:00
`prc_calc_rating_preparation`	воскресенье 15:00

При смене версии calc регламент в образе может измениться. Эталон для версии берётся из образа:

```
docker run --rm --entrypoint cat \
  your.local.registry/assortiment/assort/calc:<версия> config/schedule.json
```

Битый JSON или одно невалидное правило отключает **весь** регламент. Перед применением проверяйте файл хотя бы `python3 -m json.tool schedule.json`.

exchange — config/ex.env и хранилище

Переменная	Значение в образце	Стенд	Смысл
`DB_*`	`common.env` + `assort-db`	да	
`IMPORT_DIR`, `SUCCESS_IMPORT_DIR`, `ERROR_IMPORT_DIR`, `EXPORT_DIR`	`/data/assort-ext-data/{IMPORT,SUCCESS,ERROR,EXPORT}/`	нет	Каталоги внутри пода (точка монтирования PVC). Умолчания кода (`.`) не годятся
`TIME_NIGHT_IMPORT`	`01:00`	нет	Ночная группа импорта (география, оборудование, иерархия, ответственные), `чч:ММ` по TZ
`TIME_MORNING_IMPORT`	`03:00`	нет	Утренняя группа (товары, товар-город, оборудование магазинов, взаимозаменяемость, DB-link)
`TASK_IMPORT_INTERVAL`	60	нет	Период онлайн-группы (подматрицы, квоты), с
`BATCH_SIZE`	500	нет	Размер пачки вставки и чтения
`ENABLE_DB_LINK`	`false`	да	Импорт `stock/sale_history` из схемы `assort` той же БД. `true` только если DBA настроил схему `assort` (проверка: `select count(*) from assort.stock`)
`FULL_LOAD`	`false`	нет	Полная перезагрузка по DB-link. Включать осознанно и однократно
`DISABLE_MDMON`	`true`	нет	Импорт MDMon выключен
`TZ`	`common.env`	да	Расписание по часам контейнера

Хранилище обмена. Это тот же каталог, куда внешние системы кладут CSV и откуда забирают выгрузки. Поэтому он должен быть общим для пода и для внешних систем. В образце — NFS (`storage-exchange.yaml`, вариант A). Для однонодового кластера (microk8s) — hostPath (вариант B в том же файле).

Структура создаётся initContainer'ом `prepare-dirs` при каждом старте пода (`mkdir -p` существующее не трогает):

```
/data/assort-ext-data/
├── IMPORT/  ┐  Store_update_Ext  Assort_Stack_Type_Ext  Item_group_update_Ext
├── SUCCESS/ ├─ Supplier_Category_Manager  Item_update_Ext  Item_District
├── ERROR/   ┘  Assort_Store_Stack_Type  Assort_Stack_Zone  Assort_Stack_Equipment
│            Submatrix_Store  Submatrix_Class_Limit  Matrix_Manual  Store_Item_Hier_Limit
└── EXPORT/     Assort_Submatrix  Assort_Submatrix_Item  Assort_Submatrix_Store
```

- Имена папок фиксированы и **регистрозависимы**. Список в `base/ex.yaml` соответствует exchange `3.0.0.1`; при обновлении exchange сверяйте его с описанием новой версии.
- Файл импорта ищется как `IMPORT/<папка>/<маска>*.csv`: разделитель `;`, UTF-8, первая строка — заголовок. После обработки файл переносится в `SUCCESS/<папка>/` или `ERROR/<папка>/`, рядом кладётся `*_answer.csv` с результатом.
- Выгрузки пишутся в `EXPORT/<папка>/<сущность>_<YYYYmmddHHMMSSfff>.csv` с правами 0777.
- Процесс в поде работает от root. На NFS с `root_squash` он станет `nobody`, и записи не будет. Тогда задайте в `ex.yaml` `securityContext.runAsUser/fsGroup` под владельца шары или экспортируйте шару с `no_root_squash`.
- Запись файлов не атомарная с обеих сторон. Договоритесь с внешними системами, что они пишут во временное имя (не `*.csv`) и переименовывают по окончании.
- `SUCCESS/` сам не чистится, ротация — на стороне администратора.

ui — config/ui-config.js

Файл `config.js` в образе ui содержит значения по умолчанию, непригодные для стенда. Он всегда подменяется файлом из ConfigMap `assort-ui-config`. Остальное содержимое образа от стенда не зависит, поэтому один образ ui годится для всех стендов.

Ключ	Значение в образце	Стенд	Смысл
`API_BASE_URL`	`/api/v1`	нет	Путь API относительно хоста UI. Не менять
`REPORTS_BASE_URL`	`/reports`	нет	Статические шаблоны отчётов
`USE_MOCKS`	`false`	нет	Заглушки страницы «Процессы». Только `false`

Ключ	Значение в образце	Стенд	Смысл
`OIDC_BASE_URL`	`https://sso.example.org`	да	Корень Keycloak **для браузера**
`OIDC_REALM`	`ASSORT`	да	Realm
`OIDC_CLIENT_ID`	`assort-ui-qa`	да	Клиент стенда из раздела «Настройка Keycloak»
`OIDC_SCOPE`	`openid profile email`	нет	

Если любого из `OIDC_*` нет, приложение не запускается (белый экран).

nginx отдаёт `.js`-файлы, включая `config.js`, с заголовком `Cache-Control: immutable` на год. Заполните `ui-config.js` правильно до первого входа пользователей. После смены значений пользователям потребуется жёсткое обновление страницы (Ctrl+F5).

Первый запуск

```
NS=assort-qa
ST=k8s/stands/qa
```

Шаг 1. Namespace и секреты — раздел «Секреты» (секрет доступа к registry — раздел «Доставка образов на стенд»).

Шаг 2. БД — раздел «Подготовка PostgreSQL»: роли, база, восстановление из дампа.

Шаг 3. Просмотр того, что будет создано:

```
kubectl kustomize "$ST" | less
kubectl diff -k "$ST"            # при первом запуске покажет всё как новое
```

Шаг 4. Миграции. Job берёт тег api из `config/` стенда:

```
kubectl -n "$NS" delete job assort-db-migrate --ignore-not-found
kubectl apply -k "$ST/migrate"
kubectl -n "$NS" wait --for=condition=complete job/assort-db-migrate --timeout=60m
kubectl -n "$NS" logs job/assort-db-migrate
```

Исправный результат — последняя строка лога вида `<ревизия> (head)`. Если Job упал:

```
kubectl -n "$NS" describe job assort-db-migrate
kubectl -n "$NS" logs job/assort-db-migrate
```

Типовые причины: неверный адрес или пароль БД, у учётки нет прав владельца, дамп с неизвестной ревизией (`Can't locate revision ...`, раздел «Наполнение базы из дампа»).

Шаг 5. Первый администратор (логин совпадает с пользователем в Keycloak, раздел «Настройка Keycloak»):

```
psql -h <db-host> -U retailocean_user -d retailocean_assortment_qa \
     -v login=ivanov.i -v surname=Иванов -v name=Иван -v initial='Иванов И.' \
     -f sql/02-seed-first-admin.sql
```

Остальных пользователей с нужными ролями заводит администратор. Роли `data.tref_role`: 1 `admin`, 2 `km`, 3 `lkm`, 4 `merchandising`, 5 `imd`, 6 `cd`, 7 `dcd`. Права применяются в течение минуты (кэш api).

Шаг 6. Выкат модулей:

```
kubectl apply -k "$ST"
kubectl -n "$NS" rollout status deploy/api  --timeout=5m
kubectl -n "$NS" rollout status deploy/calc --timeout=5m
kubectl -n "$NS" rollout status deploy/ex   --timeout=5m
kubectl -n "$NS" rollout status deploy/ui   --timeout=5m
kubectl -n "$NS" get pods -o wide
```

Шаг 7. Проверка — раздел «Проверка корректности установки».

Настройка путей Ingress

Вход на стенд один — Ingress на ui (`k8s/stands/example/ingress.yaml`): `https://<хост>/` → Service `assort-ui-service:8080`. В `ingress.yaml` стенда задаются хост, класс Ingress и TLS-секрет `assort-tls`. Отдельного пути для api в Ingress не требуется.

nginx внутри ui проксирует `/api/` на `http://assort-api-service:8081`. Поэтому Service api называется `assort-api-service` и слушает порт 8081. **Не переименовывайте его.**

UI работает только из корня домена (`https://<хост>/`). Публикация под подпутём (`https://host/assort/`) требует пересборки образа.

Проверка корректности установки

```
HOST=https://assort-qa.example.org
SSO=https://sso.example.org
```

#	Проверка	Исправно	Неисправно и что это значит
1	`kubectl -n $NS get pods`	4 пода `Running`, `READY 1/1`	`ImagePullBackOff` — нет образа или доступа (раздел «Доставка образов на стенд»). api в `CrashLoopBackOff` — чаще всего недоступен Keycloak по https (раздел «Keycloak») или неверный `ENVIRONMENT`. calc или ex не Ready — нет соединения с БД
2	`curl -s -o /dev/null -w '%{http_code}\n' $HOST/`	`200`	Ошибка TLS или 404 — Ingress, DNS, сертификат
3	`curl -s $HOST/config.js`	Значения стенда (`OIDC_BASE_URL`, `OIDC_CLIENT_ID`)	Значения из образа — ConfigMap не смонтирован
4	`curl -s -w '\n%{http_code}\n' $HOST/api/v1/`	`401` и `"Authorization header is missing"` — связка ingress → ui → api работает	`502` — nginx ui не достучался до `assort-api-service:8081` (имя или порт Service, api не Ready). `404` или `200` с JSON без 401 — **авторизация выключена**, `ENVIRONMENT` не задан или равен `local`. `200` с HTML — `/api/` не проксируется
5	`curl -s $SSO/realms/ASSORT/.well-known/openid-configuration`	JSON с `issuer`	Keycloak или realm недоступен
6	`kubectl -n $NS logs deploy/api`	`Uvicorn running on http://0.0.0.0:8000`, нет `Failed to fetch JWKS`	`No Authorization header provided, using default user_rk=1` — **api в режиме local**, исправить немедленно
7	`kubectl -n $NS logs deploy/calc`	🎯 `Parallel manager started`; в течение 5 минут — `Created scheduled task … prc_process_timeout_guard`	`DB parameters not fully configured`; 📈 `DB failure #N` — нет БД; ошибки разбора `schedule.json`
8	`kubectl -n $NS logs deploy/ex`	`Set import directory: /data/assort-ext-data/IMPORT/`, `Initialize schedule for importing` и `… for exporting`, раз в минуту `Looking for data for exporting`	`FileNotFoundError` на `os.listdir` — нет папки сущности (раздел «exchange — config/ex.env и хранилище»)

#	Проверка	Исправно	Неисправно и что это значит
9	`kubectl -n $NS exec deploy/calc -- date`	Местное время по TZ	UTC — TZ не применён
10	Браузер: открыть `$HOST/`, войти	Редирект в Keycloak, вход, интерфейс с меню	Белый экран — открыть DevTools → Console: `Ошибка запуска приложения` (Keycloak недоступен или сертификат не доверен). `Invalid parameter: redirect_uri` — redirect URI клиента. `/forbidden` или 403 «not registered» — нет записи в `data.user_list`
11	SQL: `SELECT process_type_cd, process_status_cd, created_dttm FROM data.process_log ORDER BY created_dttm DESC LIMIT 5;`	Через 5–10 минут после старта calc есть свежие `prc_process_timeout_guard` в статусе `success`	Строк нет — calc не ставит или не берёт задачи (лог calc); висят в `pending` — calc не видит очередь

Проверка api изнутри кластера, минуя Ingress и ui:

```
kubectl -n "$NS" port-forward svc/assort-api-service 8081:8081 &
curl -s -o /dev/null -w '%{http_code}\n' http://127.0.0.1:8081/openapi.json   # 200
kill %1
```

`/health` api **требует токен** (вне режима `local`), поэтому без токена отвечает `401`. Это нормально, для проверки без токена используйте `/openapi.json`.

Чек-лист приёмки стенда

№	Проверка	Отметка
1	Набор тегов стенда записан в `config/kustomization.yaml`, все теги есть в registry (раздел «Проверка наличия образов в registry»), digest-ы зафиксированы	
2	Namespace создан, секреты `registry-retailocean`, `assort-db`, `assort-tls` на месте	
3	БД создана по разделу «Роли и база», восстановлена из дампа по разделу «Наполнение базы из дампа», пароль без спецсимволов	
4	`max_connections` сервера БД учитывает бюджет стенда (раздел «Роли и база»)	
5	Job миграций завершился, `alembic current` = `(head)`	

№	Проверка	Отметка
6	Клиент Keycloak стенда создан, redirect URI = `https://<хост>/`, PKCE S256	
7	Первый администратор заведён в Keycloak и в `data.user_list` с одинаковым логином	
8	`ui-config.js` указывает на Keycloak и клиент стенда	
9	`ENVIRONMENT=production`, `TZ` задан	
10	Хранилище обмена смонтировано, структура папок создана, внешние системы видят тот же каталог	
11	`ENABLE_DB_LINK` выставлен осознанно	
12	Все проверки раздела «Проверка корректности установки» пройдены	
13	Сбор логов подов стенда настроен	
14	Резервное копирование БД стенда настроено	

Обновление системы

Обновление системы происходит путём смены тегов образов в конфигурации стенда. Меняются только модули, вошедшие в релиз.

Порядок обновления

резервная копия БД → миграции (Job, новый тег api) → api → calc → ex → ui

1. **Резервная копия БД.** Миграции идут одной транзакцией, но откат ревизий поддерживается не везде. Надёжный откат схемы — только из копии.
2. Новые теги записать в `config/kustomization.yaml` стенда. Меняются только модули, вошедшие в релиз.
3. Если обновляется api — миграции (раздел «Первый запуск», шаг 4). Job возьмёт новый тег.
4. Проверить момент для calc и ex (раздел «Когда можно перезапускать calc и ex»).

5. `kubectl diff -k "$ST"`, затем `kubectl apply -k "$ST"` и `rollout status` (раздел «Первый запуск», шаг 6).
6. Проверка — раздел «Проверка корректности установки».

Миграции с изменением типов колонок `data.sale_history` и `data.stock` (сотни миллионов строк) — только в окно обслуживания, при остановленных calc и ex:

```
kubectl -n "$NS" scale deploy/calc deploy/ex --replicas=0
# ... миграции ...
kubectl -n "$NS" scale deploy/calc deploy/ex --replicas=1
```

Когда можно перезапускать calc и ex

- **calc: рестарт убивает идущие расчёты**, режима «доработать и выйти» нет. Перед выкатом проверьте, что ничего не выполняется:

  ```
  SELECT process_rk, process_type_cd, created_dttm
  FROM data.process_log
  WHERE process_status_cd = 'running';
  ```

  Прерванная `prc_cluster_saving` не восстанавливается сама: пользователь должен сохранить кластеризацию заново.
- **ex**: процесс не обрабатывает SIGTERM и убивается через 30 с. Не перезапускайте его около `TIME_NIGHT_IMPORT` и `TIME_MORNING_IMPORT`. Пропущенное окно импорта не навёрстывается, а прерванный импорт остатков и продаж может задвоить данные.
- Обоим модулям нужна стратегия `Recreate` (уже задана): старый под останавливается до старта нового, двух экземпляров одновременно не бывает.

Откат

- **Образы:** вернуть прежние теги в `config/kustomization.yaml` и выполнить `kubectl apply -k`. Для быстрого отката одного модуля: `kubectl -n $NS rollout undo deploy/<модуль>`. Учтите, что следующий `apply -k` снова выставит теги из файла.
- **api после миграций:** откатывать только образ нельзя, если новая схема с ним несовместима. Схема возвращается восстановлением из резервной копии.

Изменение настроек

Правка `config/*.env` или `ui-config.js` применяется командой `kubectl apply -k "$ST"`. Поды затронутых модулей пересоздаются сами. Учтите раздел «Когда можно перезапускать calc и ex»: пересоздание calc прерывает расчёты.

Правка секрета не пересоздаёт поды, их перезапускают вручную:

```
kubectl -n "$NS" rollout restart deploy/api deploy/calc deploy/ex
```

Регламент калькулятора

```
python3 -m json.tool "$ST/config/schedule.json" >/dev/null && kubectl apply -k "$ST"
```

ConfigMap доходит до пода примерно за минуту, calc перечитывает файл сам, рестарт не нужен. Проверка: `kubectl -n $NS exec deploy/calc -- cat /etc/assort-calc/schedule.json`.

Масштабирование

- api и ui масштабируются репликами: `kubectl -n $NS scale deploy/api --replicas=2`. Кэш прав у каждой реплики свой, изменения прав расходятся между репликами до минуты. Каждая реплика api добавляет до 20 соединений с БД.
- calc и ex **не масштабируются**. Только `replicas: 1`.

Возможные проблемы

Симптом	**Причина и решение**
Под в `ImagePullBackOff`	Нет образа или доступа к registry: проверить теги и секрет `registry-retailocean` (раздел «Доставка образов на стенд»)
api в `CrashLoopBackOff`	Чаще всего недоступен Keycloak по https или не принят его сертификат (раздел «Keycloak»); либо неверное значение `ENVIRONMENT`
calc или ex не Ready, в логе calc `DB parameters not fully configured` или `DB failure #N`	Нет соединения с БД: проверить `common.env` и секрет `assort-db`

Симптом	Причина и решение
Модули не подключаются к БД, хотя healthcheck calc и ex зелёный	В пароле БД есть символы `@ : / # %`; сменить пароль (раздел «Роли и база»)
Job миграций завершился ошибкой	Неверный адрес или пароль БД, у учётки нет прав владельца, дамп с неизвестной ревизией (`Can't locate revision ...`, раздел «Наполнение базы из дампа»)
`$HOST/api/v1/` отвечает `502`	nginx ui не достучался до `assort-api-service:8081`: имя или порт Service изменены либо api не Ready
`$HOST/api/v1/` отвечает `404` или `200` без `401`; в логе api `using default user_rk=1`	Авторизация выключена: `ENVIRONMENT` не задан или равен `local`. Исправить немедленно
Белый экран в браузере	DevTools → Console: `Ошибка запуска приложения` — Keycloak недоступен или сертификат не доверен; нет одного из ключей `OIDC_*` в `ui-config.js`
`$HOST/config.js` возвращает значения из образа	ConfigMap `assort-ui-config` не смонтирован
После правки `ui-config.js` у пользователей старые значения	`config.js` кэшируется браузером на год: нужно жёсткое обновление страницы (Ctrl+F5)
`Invalid parameter: redirect_uri` при входе	Redirect URI клиента в Keycloak не совпадает с адресом стенда посимвольно (раздел «Настройка Keycloak»)
`/forbidden` или 403 «Your account is not registered in the application»	Нет записи пользователя в `data.user_list` или логин не совпадает с `preferred_username` в Keycloak
В логе ex `FileNotFoundError` на `os.listdir`	Нет папки сущности в хранилище обмена (раздел «exchange — config/ex.env и хранилище»)
ex не может записать файлы на NFS	Шара экспортирована с `root_squash`: задать `securityContext.runAsUser/fsGroup` в `ex.yaml` или экспортировать с `no_root_squash`
Регламент calc не выполняется	Битый JSON или невалидное правило в `schedule.json` отключает весь регламент; проверить `python3 -m json.tool`
Задачи висят в `pending` или не появляются в `data.process_log`	calc не ставит или не берёт задачи: смотреть лог calc
Время в контейнере calc — UTC	Не применён `TZ` в `common.env`

Логи

Все модули пишут только в stdout/stderr, файлов логов нет. История выполнения расчётов и импортов живёт **только в логах**. Без внешнего сбора логов кластера она теряется вместе с подом.

```
kubectl -n "$NS" logs deploy/api  --since=1h
kubectl -n "$NS" logs deploy/calc --since=1h | grep -E 'Task|failed|timeout'
kubectl -n "$NS" logs deploy/ex   --since=1h | grep -E 'Importing|exporting|error'
kubectl -n "$NS" logs deploy/ui   --since=1h      # access-log nginx, в т.ч. 502 на
/api/
```

В логе calc есть ANSI-цвета и эмодзи. Для чтения в файле пропускайте его через `sed 's/\x1b\[[0-9;]*m//g'`.

Приложение. Листинги скриптов и конфигураций

Ниже приведены полные тексты всех файлов архива **retailocean-k8s-deploy-guide.zip**. Структура каталогов показана в разделе «Содержание поставки», пути файлов указаны в заголовках. Значения, помеченные `CHANGE_ME`, заменяются значениями стенда (раздел «Создание оверлея»).

k8s/base/kustomization.yaml

Общая часть манифестов, одинаковая для всех стендов.

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
# Общая для всех стендов часть. Ничего стендозависимого здесь нет:
# теги образов, адреса, конфиги и хранилище задаются в stands/<стенд>/.
labels:
  - pairs:
      app.kubernetes.io/part-of: retailocean-assortment
    includeSelectors: false
    includeTemplates: true
resources:
  - api.yaml
  - calc.yaml
  - ex.yaml
  - ui.yaml
```

k8s/base/api.yaml

Deployment и Service модуля api.

```
# api — backend HTTP API (FastAPI, uvicorn). Масштабируется репликами.
# Миграции БД этот Deployment НЕ выполняет — см. k8s/jobs/db-migrate.yaml.
apiVersion: apps/v1
kind: Deployment
metadata:
  name: api
  labels:
    app.kubernetes.io/name: api
    app.kubernetes.io/component: backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: api
  template:
    metadata:
      labels:
        app.kubernetes.io/name: api
        app.kubernetes.io/component: backend
    spec:
      imagePullSecrets:
        - name: registry-retailocean
      terminationGracePeriodSeconds: 30
```

```
      containers:
        - name: api
          # Тег подставляется из stands/<стенд>/kustomization.yaml (блок images).
          image: your.local.registry/assortiment/assort/api
          imagePullPolicy: IfNotPresent
          # Та же команда, что CMD образа, но в exec-форме: иначе PID 1 — /bin/sh,
          # и SIGTERM не доходит до uvicorn.
          command:
            - python
            - scripts/install_keycloak_cert.py
            - uvicorn
            - retailocean_assortment_api.main:app
            - --host
            - 0.0.0.0
            - --port
            - "8000"
          ports:
            - name: http
              containerPort: 8000
          envFrom:
            - configMapRef:
                name: assort-common
            - configMapRef:
                name: assort-api-env
            - secretRef:
                name: assort-db
          # /health требует Bearer-токен при ENVIRONMENT != local, поэтому для проб
          # используется публичный /openapi.json (в БД не ходит).
          readinessProbe:
            httpGet:
              path: /openapi.json
              port: http
            initialDelaySeconds: 5
            periodSeconds: 10
            timeoutSeconds: 5
          livenessProbe:
            tcpSocket:
              port: http
            initialDelaySeconds: 30
            periodSeconds: 20
            failureThreshold: 3
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              memory: 1Gi
          volumeMounts:
            - name: tmp
              mountPath: /tmp
      volumes:
        - name: tmp
          emptyDir: {}
---
# Имя и порт Service НЕ менять: nginx образа ui проксирует /api/
# на http://assort-api-service:8081.
apiVersion: v1
kind: Service
```

```
metadata:
  name: assort-api-service
  labels:
    app.kubernetes.io/name: api
spec:
  selector:
    app.kubernetes.io/name: api
  ports:
    - name: http
      port: 8081
      targetPort: http
```

k8s/base/calc.yaml

Deployment модуля calculator (одна реплика, Recreate).

```
# calc — демон расчётов. HTTP не обслуживает, Service не нужен.
# Строго одна реплика на базу и стратегия Recreate: второй экземпляр на той же
# БД задваивает регламентные задачи (планировщик без блокировок).
apiVersion: apps/v1
kind: Deployment
metadata:
  name: calc
  labels:
    app.kubernetes.io/name: calc
    app.kubernetes.io/component: worker
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app.kubernetes.io/name: calc
  template:
    metadata:
      labels:
        app.kubernetes.io/name: calc
        app.kubernetes.io/component: worker
    spec:
      imagePullSecrets:
        - name: registry-retailocean
      # Остановка: запрет новых задач -> cancel SQL -> terminate/kill -> запись
статусов.
      # Укладывается в MAIN_LOOP_INTERVAL + TASK_SHUTDOWN_TIMEOUT_SECONDS; 30 с k8s
мало.
      terminationGracePeriodSeconds: 90
      containers:
        - name: calc
          image: your.local.registry/assortiment/assort/calc
          imagePullPolicy: IfNotPresent
          envFrom:
            - configMapRef:
                name: assort-common
            - configMapRef:
                name: assort-calc-env
            - secretRef:
                name: assort-db
          # Только readiness: проверяет лишь соединение с БД. Liveness не ставить —
```

```
          # рестарт пода убивает многочасовые расчёты, а при недоступной БД демон
          # сам ждёт в backoff.
          readinessProbe:
            exec:
              command:
                - python
                - /retailocean-assortment/calculator-service/src/healthcheck.py
            initialDelaySeconds: 30
            periodSeconds: 60
            timeoutSeconds: 15
            failureThreshold: 3
          resources:
            requests:
              cpu: "1"
              memory: 2Gi
            limits:
              memory: 6Gi
          volumeMounts:
            # Каталог, а не subPath: так правка ConfigMap доходит до пода без
            # рестарта, и калькулятор перечитывает файл по mtime.
            - name: schedule
              mountPath: /etc/assort-calc
              readOnly: true
      volumes:
        - name: schedule
          configMap:
            name: assort-calc-schedule
```

k8s/base/ex.yaml

Deployment модуля exchange с initContainer, создающим каталоги обмена.

```
# ex — файловый обмен (импорт/экспорт CSV) и импорт по DB-link.
# HTTP не обслуживает, Service не нужен. Строго одна реплика на базу, Recreate:
# два экземпляра делят одни файлы и staging-таблицы.
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ex
  labels:
    app.kubernetes.io/name: ex
    app.kubernetes.io/component: worker
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app.kubernetes.io/name: ex
  template:
    metadata:
      labels:
        app.kubernetes.io/name: ex
        app.kubernetes.io/component: worker
    spec:
      imagePullSecrets:
        - name: registry-retailocean
      initContainers:
```

```
        # Сервис сам папки не создаёт, а при отсутствии подпапки импорт/экспорт
        # сущности падает. Список соответствует exchange 3.0.0.1 (folder_name в
        # src/processes/imports/dictionary.py и exports/dictionary.py);
        # при обновлении exchange сверять.
        - name: prepare-dirs
          image: your.local.registry/assortiment/assort/ex
          imagePullPolicy: IfNotPresent
          command:
            - sh
            - -c
            - |
              set -e
              ROOT=/data/assort-ext-data
              for d in IMPORT SUCCESS ERROR; do
                for f in Store_update_Ext Assort_Stack_Type_Ext
Item_group_update_Ext \
                         Supplier_Category_Manager Item_update_Ext Item_District \
                         Assort_Store_Stack_Type Assort_Stack_Zone
Assort_Stack_Equipment \
                         Submatrix_Store Submatrix_Class_Limit Matrix_Manual
Store_Item_Hier_Limit; do
                  mkdir -p "$ROOT/$d/$f"
                done
              done
              for f in Assort_Submatrix Assort_Submatrix_Item
Assort_Submatrix_Store; do
                mkdir -p "$ROOT/EXPORT/$f"
              done
              echo "exchange dirs ready under $ROOT"
          volumeMounts:
            - name: ext-data
              mountPath: /data/assort-ext-data
      # SIGTERM процесс не обрабатывает (python — PID 1) и будет убит по истечении
      # этого срока. Перезапускать между окнами расписания.
      terminationGracePeriodSeconds: 30
      containers:
        - name: ex
          image: your.local.registry/assortiment/assort/ex
          imagePullPolicy: IfNotPresent
          envFrom:
            - configMapRef:
                name: assort-common
            - configMapRef:
                name: assort-ex-env
            - secretRef:
                name: assort-db
          # Только readiness (SELECT 1). Мёртвый воркер она не видит; liveness не
ставить.
          readinessProbe:
            exec:
              command:
                - python
                - /retailocean-assortment/exchange-service/src/healthcheck.py
            initialDelaySeconds: 30
            periodSeconds: 60
            timeoutSeconds: 15
            failureThreshold: 3
          resources:
```

```
            requests:
              cpu: 250m
              memory: 512Mi
            limits:
              memory: 1Gi
          volumeMounts:
            - name: ext-data
              mountPath: /data/assort-ext-data
      volumes:
        - name: ext-data
          persistentVolumeClaim:
            # Создаётся в stands/<стенд>/storage-exchange.yaml.
            claimName: assort-ext-data
```

k8s/base/ui.yaml

Deployment и Service модуля ui.

```
# ui — SPA на nginx (образ nginx-unprivileged, uid 101, порт 8080).
# nginx внутри контейнера отдаёт статику и проксирует /api/ на
assort-api-service:8081.
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ui
  labels:
    app.kubernetes.io/name: ui
    app.kubernetes.io/component: frontend
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: ui
  template:
    metadata:
      labels:
        app.kubernetes.io/name: ui
        app.kubernetes.io/component: frontend
    spec:
      imagePullSecrets:
        - name: registry-retailocean
      containers:
        - name: ui
          image: your.local.registry/assortiment/assort/ui
          imagePullPolicy: IfNotPresent
          ports:
            - name: http
              containerPort: 8080
          readinessProbe:
            httpGet:
              path: /
              port: http
            periodSeconds: 10
          livenessProbe:
            httpGet:
              path: /
              port: http
            initialDelaySeconds: 10
```

```
            periodSeconds: 20
          resources:
            requests:
              cpu: 50m
              memory: 64Mi
            limits:
              memory: 256Mi
          volumeMounts:
            # В образе лежит config.js со значениями стенда dev — всегда подменять.
            - name: ui-config
              mountPath: /usr/share/nginx/html/config.js
              subPath: config.js
              readOnly: true
      volumes:
        - name: ui-config
          configMap:
            name: assort-ui-config
---
apiVersion: v1
kind: Service
metadata:
  name: assort-ui-service
  labels:
    app.kubernetes.io/name: ui
spec:
  selector:
    app.kubernetes.io/name: ui
  ports:
    - name: http
      port: 8080
      targetPort: http
```

k8s/jobs/kustomization.yaml

Подключение разовых задач.

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
# Разовые задачи. Подключаются оверлеями stands/<стенд>/migrate.
resources:
  - db-migrate.yaml
```

k8s/jobs/db-migrate.yaml

Job миграций схемы БД (alembic upgrade head).

```
# Накат миграций схемы БД (Alembic) образом api той версии, что будет выкатываться.
# Применяется только через stands/<стенд>/migrate (тег и ConfigMap-ы — из config/
стенда).
# Job неизменяем: перед повторным запуском удалить предыдущий (раздел 8
инструкции).
apiVersion: batch/v1
kind: Job
metadata:
  name: assort-db-migrate
  labels:
    app.kubernetes.io/name: db-migrate
    app.kubernetes.io/part-of: retailocean-assortment
```

```
spec:
  backoffLimit: 0
  ttlSecondsAfterFinished: 604800
  template:
    metadata:
      labels:
        app.kubernetes.io/name: db-migrate
    spec:
      restartPolicy: Never
      imagePullSecrets:
        - name: registry-retailocean
      containers:
        - name: alembic
          image: your.local.registry/assortiment/assort/api
          imagePullPolicy: IfNotPresent
          workingDir: /retailocean-assortment/api
          command: ["sh", "-c", "alembic current && alembic upgrade head && alembic
current"]
          envFrom:
            - configMapRef:
                name: assort-common
            # Для миграций нужны права владельца схем (DDL, TRUNCATE, членство
            # в роли assortment). При раздельных учётках — Secret
assort-db-migrator.
            - secretRef:
                name: assort-db
          resources:
            requests:
              cpu: 100m
              memory: 256Mi
            limits:
              memory: 1Gi
```

k8s/stands/example/kustomization.yaml

Оверлей стенда: namespace и состав.

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
# Оверлей одного стенда. Скопировать каталог в stands/<стенд>/ и заменить
# значения, помеченные CHANGE_ME, во всех файлах каталога (включая config/ и
migrate/).
# Версии образов и конфиги модулей — в config/.

namespace: assort-qa            # CHANGE_ME: один namespace = один стенд = одна БД

resources:
  - ../../base
  - ingress.yaml
  - storage-exchange.yaml

components:
  - config

labels:
  - pairs:
      app.kubernetes.io/instance: assort-qa   # CHANGE_ME
    includeSelectors: false
```

```
    includeTemplates: true
```

k8s/stands/example/ingress.yaml

Вход на стенд по HTTPS.

```
# Единственная точка входа стенда: весь трафик, включая /api/, идёт в ui,
# а nginx ui сам проксирует /api/ на api. TLS обязателен: без HTTPS браузер
# не даёт crypto.subtle, и вход через Keycloak (PKCE) не работает.
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: assort-ui
spec:
  ingressClassName: nginx            # CHANGE_ME: класс Ingress-контроллера
кластера
  tls:
    - hosts:
        - assort-qa.example.org      # CHANGE_ME
      secretName: assort-tls         # kubectl create secret tls assort-tls ...
(раздел 7.2)
  rules:
    - host: assort-qa.example.org    # CHANGE_ME
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: assort-ui-service
                port:
                  name: http
```

k8s/stands/example/storage-exchange.yaml

PV и PVC каталогов обмена (NFS или hostPath).

```
# Хранилище каталогов обмена exchange (IMPORT, SUCCESS, ERROR, EXPORT).
# Это тот же каталог, в который внешние системы кладут и из которого забирают CSV,
# поэтому он должен быть доступен и им, и поду. Выбрать ОДИН вариант.
#
# Вариант A (по умолчанию): общая NFS-шара.
# PV кластерный (без namespace) — имя должно быть уникально на кластер.
apiVersion: v1
kind: PersistentVolume
metadata:
  name: assort-qa-ext-data           # CHANGE_ME
spec:
  capacity:
    storage: 50Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: ""
  nfs:
    server: nfs.example.local        # CHANGE_ME
    path: /export/assort-qa-ext-data # CHANGE_ME
  claimRef:
```

```
    namespace: assort-qa             # CHANGE_ME: = namespace стенда
    name: assort-ext-data
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: assort-ext-data
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  volumeName: assort-qa-ext-data     # CHANGE_ME: = имя PV выше
  resources:
    requests:
      storage: 50Gi
#
# Вариант B: однонодовый кластер (microk8s и т.п.), каталог на самом узле.
# Заменить блок nfs: в PV на:
#   hostPath:
#     path: /opt/assort-ext-data
#     type: DirectoryOrCreate
# и accessModes на ReadWriteOnce в PV и PVC.
```

k8s/stands/example/config/kustomization.yaml

Набор версий образов и генерация ConfigMap.

```
apiVersion: kustomize.config.k8s.io/v1alpha1
kind: Component
# Конфигурация стенда: набор версий образов и ConfigMap-ы модулей.
# Подключается и оверлеем стенда (../), и оверлеем миграций (../migrate/),
# поэтому теги и имена ConfigMap у них всегда совпадают.

# Набор версий стенда. Теги модулей независимы: в релиз входят только задетые
# модули (раздел 3 инструкции). Для зеркала registry добавить newName.
images:
  - name: your.local.registry/assortiment/assort/api
    newTag: "3.0.1.7"
  - name: your.local.registry/assortiment/assort/calc
    newTag: "3.0.0.18"
  - name: your.local.registry/assortiment/assort/ex
    newTag: "3.0.0.1"
  - name: your.local.registry/assortiment/assort/ui
    newTag: "3.0.1.4"

configMapGenerator:
  # Общие для всех модулей: адрес БД и часовой пояс.
  - name: assort-common
    envs:
      - common.env
  - name: assort-api-env
    envs:
      - api.env
  - name: assort-calc-env
    envs:
      - calc.env
  - name: assort-ex-env
    envs:
```

```
      - ex.env
  - name: assort-ui-config
    files:
      - config.js=ui-config.js
  # Без хеш-суффикса: правка регламента подхватывается калькулятором на лету,
  # без пересоздания пода (иначе рестарт убьёт идущие расчёты).
  - name: assort-calc-schedule
    files:
      - schedule.json
    options:
      disableNameSuffixHash: true
```

k8s/stands/example/config/common.env

Общие переменные всех модулей.

```
# Общие переменные всех модулей стенда (ConfigMap assort-common).
# Логин и пароль БД сюда НЕ класть — они в Secret assort-db (раздел 7.2
инструкции).
DB_HOST=pg-assort-qa.example.local
DB_PORT=5432
DB_NAME=retailocean_assortment_qa
# Регламенты calc и exchange считаются по часам контейнера; без TZ это UTC.
TZ=Asia/Yekaterinburg
```

k8s/stands/example/config/api.env

Переменные модуля api.

```
# api (ConfigMap assort-api-env).
# ENVIRONMENT задавать всегда: без неё api стартует как local и ОТКЛЮЧАЕТ
# проверку токенов и прав. Допустимо только local | staging | production.
ENVIRONMENT=production
LOG_LEVEL=INFO
API_V1_STR=/api/v1

# Keycloak, в который api ходит за JWKS. Адрес должен быть доступен ИЗ ПОДА.
# Issuer api не сверяет, поэтому можно указать внутренний адрес Keycloak,
# даже если браузер ходит на внешний. Схема https => при старте пода скрипт
# скачивает цепочку сертификатов Keycloak (до 5 попыток, затем CrashLoop).
KEYCLOAK_SERVER_URL=https://sso.example.org
KEYCLOAK_REALM=ASSORT
KEYCLOAK_VERIFY_SSL=true

# Пул соединений (действует при production): на реплику до POOL+OVERFLOW
соединений.
DB_POOL_SIZE=10
DB_MAX_OVERFLOW=10
THREADPOOL_SIZE=20
AUDIT_ENABLED=true
```

k8s/stands/example/config/calc.env

Переменные модуля calculator.

```
# calculator (ConfigMap assort-calc-env).
# Жёсткий таймаут задачи, с. Рекомендуется 72000 (20 ч); меньшие значения прерывают
# долгие расчёты. Сторож осиротевших задач срабатывает при 2 x PROCESS_TIMEOUT.
PROCESS_TIMEOUT=72000
MAX_CONCURRENT_PROCESSES=10
MAIN_LOOP_INTERVAL=5
# Путь к регламенту, смонтированному из ConfigMap assort-calc-schedule.
SCHEDULE_FILE_PATH=/etc/assort-calc/schedule.json
AUDIT_RETENTION_DAYS=180
LOG_LEVEL=INFO
```

k8s/stands/example/config/ex.env

Переменные модуля exchange.

```
# exchange (ConfigMap assort-ex-env).
# Корень каталогов обмена смонтирован в /data/assort-ext-data (см. base/ex.yaml).
IMPORT_DIR=/data/assort-ext-data/IMPORT/
SUCCESS_IMPORT_DIR=/data/assort-ext-data/SUCCESS/
ERROR_IMPORT_DIR=/data/assort-ext-data/ERROR/
EXPORT_DIR=/data/assort-ext-data/EXPORT/

# Расписание импорта: время по TZ контейнера (ЧЧ:ММ), онлайн-группа — период в с.
TIME_NIGHT_IMPORT=01:00
TIME_MORNING_IMPORT=03:00
TASK_IMPORT_INTERVAL=60
BATCH_SIZE=500

# Импорт остатков и продаж из схемы assort (DB-link). Если на стенде схемы
# assort нет — false, иначе утренний импорт будет падать.
ENABLE_DB_LINK=false
FULL_LOAD=false
DISABLE_MDMON=true
```

k8s/stands/example/config/ui-config.js

Конфигурация ui для браузера (подменяет config.js образа).

```
// Монтируется в /usr/share/nginx/html/config.js (ConfigMap assort-ui-config).
// Файл публичный: его скачивает браузер. Секретов здесь быть не должно.
window.env = {
  API_BASE_URL: '/api/v1',
  REPORTS_BASE_URL: '/reports',
  USE_MOCKS: false,
  // Адрес Keycloak, открываемый из БРАУЗЕРА пользователя, с доверенным
сертификатом.
  OIDC_BASE_URL: 'https://sso.example.org',
  OIDC_REALM: 'ASSORT',
  OIDC_CLIENT_ID: 'assort-ui-qa',
  OIDC_SCOPE: 'openid profile email',
};
```

k8s/stands/example/config/schedule.json

Регламент калькулятора (копия из образа calc 3.0.0.18).

```
[
  {
    "script_name": "prc_substitute_review",
    "enabled": true,
    "type": "time",
    "time": "00:01"
  },
  {
    "script_name": "prc_user_audit_cleanup",
    "enabled": true,
    "type": "time",
    "time": "01:00"
  },
  {
    "script_name": "prc_process_timeout_guard",
    "enabled": true,
    "type": "interval",
    "minutes": 5,
    "start_time": "00:00"
  },
  {
    "script_name": "prc_calc_rating_preparation",
    "enabled": true,
    "type": "time",
    "time": "15:00",
    "days": [
      "sunday"
    ]
  },
  {
    "script_name": "prc_segments",
    "enabled": true,
    "type": "time",
    "time": "09:00",
    "days": [
      "saturday"
    ]
  },
  {
    "script_name": "prc_sale_history_cleanup",
    "enabled": true,
    "type": "time",
    "time": "14:00",
    "days": [
      "saturday"
    ]
  },
  {
    "script_name": "prc_stock_cleanup",
    "enabled": true,
    "type": "time",
    "time": "14:00",
    "days": [
      "saturday"
```

```
    ]
  },
  {
    "script_name": "prc_submatrices_cleanup",
    "enabled": true,
    "type": "time",
    "time": "01:00",
    "days": [
      "sunday"
    ]
  }
]
```

k8s/stands/example/migrate/kustomization.yaml

Оверлей Job миграций стенда.

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
# Job миграций БД для этого стенда. Берёт тег api и ConfigMap-ы из ../config,
# то есть накатывает схему ровно той версии api, что записана в стенде.
# Применять ДО выката модулей (раздел 8 инструкции).

namespace: assort-qa            # CHANGE_ME: = namespace стенда

resources:
  - ../../../jobs

components:
  - ../config
```

sql/01-roles-and-database.sql

Роли и база данных стенда.

```
-- Подготовка PostgreSQL нового стенда. Выполнять суперпользователем на ЦЕЛЕВОМ
сервере:
--   psql -h <db-host> -U postgres -v stand_db=retailocean_assortment_qa -f
01-roles-and-database.sql
--
-- Модель владения (как на существующих контурах: одна учётка на все модули):
--   assortment    – NOLOGIN-роль, владелец БД и всех объектов (на неё же ссылается
--                   миграция api: ALTER FUNCTION ... OWNER TO assortment);
--   retailocean_user  – учётка модулей, член assortment => права владельца
--                   (DDL calc.*, TRUNCATE staging, триггеры exchange, TEMP).

SELECT 'CREATE ROLE assortment NOLOGIN'
WHERE NOT EXISTS (SELECT 1 FROM pg_roles WHERE rolname = 'assortment') \gexec

SELECT 'CREATE ROLE retailocean_user LOGIN IN ROLE assortment'
WHERE NOT EXISTS (SELECT 1 FROM pg_roles WHERE rolname = 'retailocean_user') \gexec

-- Пароль задать интерактивно, чтобы он не попал в историю команд и в логи сервера:
\password retailocean_user

SELECT format('CREATE DATABASE %I OWNER assortment', :'stand_db')
WHERE NOT EXISTS (SELECT 1 FROM pg_database WHERE datname = :'stand_db') \gexec
```

```
SELECT format('REVOKE ALL ON DATABASE %I FROM PUBLIC', :'stand_db') \gexec
SELECT format('GRANT CONNECT, TEMPORARY ON DATABASE %I TO retailocean_user',
:'stand_db') \gexec
```

sql/02-seed-first-admin.sql

Первый администратор системы.

```
-- Первый пользователь-администратор стенда. Выполнять в БД стенда после миграций:
--   psql -h <db-host> -U retailocean_user -d retailocean_assortment_qa \
--        -v login=ivanov.i -v surname=Иванов -v name=Иван -v initial='Иванов И.' \
--        -f 02-seed-first-admin.sql
--
-- login ОБЯЗАН совпадать с preferred_username в токене Keycloak. Keycloak хранит
-- имена пользователей в нижнем регистре, поэтому и здесь — только нижний регистр.
-- role_rk = 1 — роль admin (data.tref_role), проходит любые проверки прав.

BEGIN;

INSERT INTO data.user_list (user_rk, user_login, user_surname, user_name,
user_initial, approver_flg)
SELECT next_rk.user_rk, :'login', :'surname', :'name', :'initial', false
FROM (SELECT COALESCE(MAX(user_rk), 0) + 1 AS user_rk FROM data.user_list) AS
next_rk
WHERE NOT EXISTS (SELECT 1 FROM data.user_list WHERE user_login = :'login');

INSERT INTO data.user_role_assignment (user_rk, role_rk)
SELECT ul.user_rk, 1
FROM data.user_list ul
WHERE ul.user_login = :'login'
  AND NOT EXISTS (
      SELECT 1 FROM data.user_role_assignment ura
      WHERE ura.user_rk = ul.user_rk AND ura.role_rk = 1
  );

SELECT ul.user_rk, ul.user_login, tr.role_cd
FROM data.user_list ul
JOIN data.user_role_assignment ura ON ura.user_rk = ul.user_rk
JOIN data.tref_role tr ON tr.role_rk = ura.role_rk
WHERE ul.user_login = :'login';

COMMIT;
```

scripts/mirror-images.sh

Копирование набора образов в registry стенда или в tar-архив.

```
#!/usr/bin/env bash
# Копирует набор образов стенда из your.local.registry в registry стенда
# (или сохраняет их в tar для закрытого контура).
#
#   ./mirror-images.sh --tags api=3.0.1.7,calc=3.0.0.18,ex=3.0.0.1,ui=3.0.1.4 \
#                      --to registry.stand.local/assort
#   ./mirror-images.sh --tags api=3.0.1.7,calc=3.0.0.18,ex=3.0.0.1,ui=3.0.1.4 \
#                      --save assort-images.tar
#
# Нужны docker и выполненный docker login в источник (и в приёмник для --to).
```

```
set -Eeuo pipefail

SRC="${SRC_REGISTRY:-your.local.registry/assortiment/assort}"
TAGS=""
DEST=""
SAVE=""

while (($# > 0)); do
  case "$1" in
    --tags) TAGS="$2"; shift ;;
    --to)   DEST="$2"; shift ;;
    --save) SAVE="$2"; shift ;;
    -h|--help) sed -n '2,11p' "$0"; exit 0 ;;
    *) echo "Неизвестный аргумент: $1" >&2; exit 2 ;;
  esac
  shift
done

if [[ -z "$TAGS" || ( -z "$DEST" && -z "$SAVE" ) ]]; then
  echo "Нужны --tags и --to или --save (см. --help)." >&2
  exit 2
fi

declare -a IMAGES=()
IFS=',' read -r -a PAIRS <<<"$TAGS"
for pair in "${PAIRS[@]}"; do
  module="${pair%%=*}"
  tag="${pair#*=}"
  case "$module" in api|calc|ex|ui) ;; *) echo "Неизвестный модуль: $module" >&2;
exit 2 ;; esac
  src="$SRC/$module:$tag"
  echo "==> pull $src"
  docker pull "$src"
  echo "    digest: $(docker image inspect --format '{{index .RepoDigests 0}}'
"$src")"
  if [[ -n "$DEST" ]]; then
    dst="$DEST/$module:$tag"
    docker tag "$src" "$dst"
    echo "==> push $dst"
    docker push "$dst"
  fi
  IMAGES+=("$src")
done

if [[ -n "$SAVE" ]]; then
  echo "==> save ${#IMAGES[@]} образа(ов) в $SAVE"
  docker save -o "$SAVE" "${IMAGES[@]}"
  ls -lh "$SAVE"
fi

echo "Готово."
```