Документация Deckhouse Galleon

Дата генерации: 26.10.2026

Документация

Концепции 4
- Репозитории и форматы 4
- Управление доступом 6
- Проксирование и кеширование 8

Руководство администратора 12
- Обзор 12
- Установка 12
- Первичная настройка 16
- Конфигурация 17
- Параметры конфигурации 21
- Секреты и ключ шифрования 32
- Репозитории 34
- Управление доступом 38
- Привилегии 40
- Хранилище и сборка мусора 45
- Политики очистки 48
- Резервное копирование 52
- TLS 54
- Обратный прокси 55
- Мониторинг 58
- Логирование 59
- API 62

Руководство пользователя 63
- Обзор 63
- Веб-интерфейс 63
- Учётные данные 65
- Форматы 66
  - Docker (OCI) 67
  - Maven 69
  - Go 71
  - npm 73
  - PyPI 74
  - RubyGems 76

Cargo 77
raw 78

Концепции

В этом разделе описано устройство Deckhouse Galleon: репозитории и форматы артефактов, модель управления доступом, проксирование и кеширование внешних реестров.

Читать удобнее по порядку: репозитории и форматы объясняют, из чего состоит хранилище и как к нему обращаются клиенты; управление доступом — кто и что может делать с этим содержимым; проксирование и кеширование — как Galleon работает с внешними реестрами. Практика — в руководстве администратора и руководстве пользователя.

Репозитории и форматы

Репозиторий — основная единица хранения в Galleon. У каждого репозитория есть имя, формат и тип.

Форматы

Формат определяет, какие артефакты хранит репозиторий и по какому протоколу с ним работают клиенты.

Формат	Содержимое	Типичный клиент
Docker (OCI)	Образы контейнеров, OCI-артефакты (например, Helm-чарты)	`docker`, `crane`, `skopeo`, `helm`
Maven	Java-библиотеки и другие артефакты сборки	`mvn`, `gradle`
Go	Go-модули	`go`
npm	Пакеты npm	`npm`, `yarn`, `pnpm`
PyPI	Пакеты Python	`pip`, `uv`, `twine`
RubyGems	Гемы Ruby	`gem`, `bundle`
Cargo	Крейты Rust	`cargo`
raw	Произвольные файлы	`curl`, браузер

Типы репозиториев

Поддерживаются два типа репозиториев: hosted и proxy.

Hosted

Hosted-репозиторий хранит артефакты, которые публикуете вы: результаты сборок, внутренние библиотеки, релизные архивы. Политика записи репозитория определяет, что разрешено: перезапись опубликованных версий, только первая публикация каждой версии (опубликованное нельзя перезаписать) или полный запрет записи (репозиторий доступен только на чтение).

Proxy

Proxy-репозиторий — кеширующее зеркало внешнего реестра, например Maven Central или Docker Hub. При первом запросе артефакт скачивается из внешнего реестра и сохраняется, повторные запросы обслуживаются из кеша. Proxy-репозиторий доступен только на чтение; для публикации используйте hosted-репозиторий. Подробнее о кешировании — в разделе Проксирование и кеширование.

> Групповые репозитории (несколько репозиториев под одним адресом) пока не поддерживаются.

Адрес репозитория

Galleon работает со штатными клиентами каждого формата. В настройках клиента указывается адрес репозитория вида:

```
https://<GALLEON_HOST>/repository/<REPOSITORY>/...
```

Исключение — репозитории формата Docker (OCI): контейнерные клиенты обращаются к ним без префикса `/repository/`, адрес образа выглядит как `<GALLEON_HOST>/<REPOSITORY>/<IMAGE>:<TAG>`.

Как настроить клиент каждого формата, описано в руководстве пользователя.

Компоненты и ассеты

Содержимое репозитория складывается из компонентов и ассетов:

- **Компонент** — единица версионирования: версия пакета, образ с тегом, Go-модуль конкретной версии.
- **Ассет** — отдельный файл, принадлежащий компоненту: архив пакета, POM-файл, слой образа.

В raw-репозиториях компонентом считается каждый отдельный файл — его именем служит полный путь файла.

Образ контейнера устроен так же: компонент — образ с тегом, ассеты — его манифест и слои. Слои у образов общие: базовый слой нескольких образов хранится один раз. Кроме образов с

тегами в репозитории могут быть манифесты, на которые ссылаются только по контрольной сумме, — в дереве содержимого они помечены как `digest`.

Одинаковое содержимое хранится в одном экземпляре: файл узнаётся по контрольной сумме, и один и тот же архив, опубликованный в двух репозиториях, занимает место один раз. Место освобождается, когда на содержимое не осталось ни одной ссылки.

В веб-интерфейсе содержимое репозитория отображается в виде дерева компонентов и ассетов. Там же при наличии прав доступны удаление артефактов, а для форматов Maven, npm, RubyGems, PyPI и raw — загрузка.

Стартовый набор репозиториев

При первом запуске с пустой базой данных Galleon создаёт стартовый набор репозиториев: proxy-зеркала общедоступных реестров (Maven Central, proxy.golang.org, база контрольных сумм Go, Docker Hub) и hosted-репозитории для Maven (отдельно релизы без перезаписи и snapshot-версии), Go и образов контейнеров.

Набор создаётся только при пустом списке репозиториев: удалённый из него репозиторий при перезапуске не появится заново. У proxy-зеркал общедоступных реестров есть права анонимного чтения; они действуют после включения анонимного доступа (Управление доступом).

Состав набора и настройки каждого репозитория описаны в разделе Репозитории руководства администратора.

Управление доступом

Доступ ко всем операциям Galleon, от чтения артефактов до администрирования, регулируется единой моделью: пользователи получают роли, а роли объединяют привилегии.

Модель

Модель строится на трёх сущностях:

- **Привилегия** — право выполнять определённые действия над объектом: репозиторием или административной областью.
- **Роль** — именованный набор привилегий. Роль может включать другие роли; их привилегии тоже достаются пользователю с этой ролью.
- **Пользователь** — учётная запись с назначенными ролями.

Встроенные роли:

- `admin` — полный доступ ко всем операциям;
- `anonymous` — привилегии этой роли действуют для запросов без аутентификации (Анонимный доступ).

Привилегии

Привилегия разрешает набор действий над одним объектом: репозиторием или административной областью — пользователями, ролями, настройками, задачами.

Привилегии создаются автоматически: административные — при первом запуске, привилегии на каждое действие с репозиторием — при его создании. Есть и универсальные с подстановкой, например право чтения во всех репозиториях.

Типы привилегий, их поля, устройство имён и сборка типовых ролей — в разделе Привилегии руководства администратора.

Действия

Привилегии на репозитории оперируют пятью независимыми действиями:

Действие	Что разрешает
`browse`	Видеть репозиторий и его содержимое в веб-интерфейсе
`read`	Скачивание артефактов
`add`	Публикация новых артефактов
`edit`	Изменение уже опубликованного содержимого
`delete`	Удаление артефактов

> ⚠ Действия независимы: `read` не включает `browse`, `edit` не включает `add`. Иерархии «администрирование включает запись, запись включает чтение» нет — нужный набор действий указывается явно.

Исключение — публикация образов контейнеров: клиент запрашивает у реестра доступ и на чтение, и на запись сразу, поэтому для `docker push` нужны оба действия — `read` и `add`.

В веб-интерфейсе и его API репозиторий без прав `browse` или `read` выглядит несуществующим: сервер отвечает кодом `404`.

Анонимный доступ

Анонимный доступ (чтение без аутентификации) складывается из двух настроек:

1. Глобальный переключатель анонимного доступа — по умолчанию выключен.
2. Привилегии, назначенные встроенной роли `anonymous`, — они определяют, что именно доступно без аутентификации.

Привилегии роли `anonymous` действуют только для запросов без аутентификации: аутентифицированный пользователь получает ровно привилегии своих ролей, даже если анонимный доступ настроен шире.

Публикация артефактов всегда требует аутентификации, независимо от настроек анонимного доступа.

Учётные данные

Для аутентификации используются два вида учётных данных:

- **Пароль** — используется для входа в веб-интерфейс, подходит и для клиентов командной строки.
- **Токен пользователя** — сгенерированная пара значений, которая используется как логин и пароль в клиентах командной строки и CI, не раскрывая пароль учётной записи. У каждого пользователя может быть один действующий токен; его можно отозвать и выпустить заново в профиле.

В веб-интерфейсе после входа сеанс сохраняется; клиенты командной строки передают учётные данные при каждом запросе.

Первая учётная запись администратора создаётся автоматически при первом запуске из конфигурации сервера (Первичная настройка).

Проксирование и кеширование

Proxy-репозиторий — кеширующее зеркало внешнего реестра: клиенты всегда обращаются к Galleon, а Galleon по мере необходимости скачивает артефакты из внешнего реестра и сохраняет их локально.

Как обрабатывается запрос

Запрос к proxy-репозиторию обрабатывается так:

- артефакт есть в кеше и срок его хранения не истёк — Galleon отдаёт его из кеша, внешний реестр не участвует;
- срок истёк — Galleon спрашивает внешний реестр, изменился ли артефакт. Не изменился — клиент получает копию из кеша, а срок хранения отсчитывается заново; изменился — Galleon скачивает новую версию, заменяет ею кеш и отдаёт клиенту;
- артефакта в кеше нет — Galleon скачивает его из внешнего реестра, сохраняет и отдаёт клиенту.

Если внешний реестр недоступен

Пока в кеше есть копия, клиенты её получают — даже если срок хранения истёк и проверить актуальность не удалось. Ошибкой завершаются только запросы к тому, чего в кеше нет.

Если сбои повторяются, а в настройках репозитория включена автоматическая блокировка, Galleon перестаёт обращаться к реестру и периодически проверяет, не восстановился ли он; всё это время кеш продолжает отдаваться. Обращения можно приостановить и вручную — тогда репозиторий работает только на кеше.

Текущее состояние связи видно в колонке «Источник» списка репозиториев; значения этой колонки описаны в разделе Репозитории руководства администратора.

Сроки хранения

Кеш различает два вида данных:

- **Контент** — сами артефакты: архивы пакетов, слои образов. Содержимое таких файлов для конкретной версии обычно не меняется.
- **Метаданные** — списки версий, индексы и дескрипторы пакетов, которые внешний реестр обновляет по мере выхода новых версий.

Сроки задаются в настройках репозитория двумя параметрами: `contentMaxAge` — для контента, `metadataMaxAge` — для метаданных. Оба принимают три вида значений:

- **Положительное число** — срок хранения в минутах. По умолчанию 1440, то есть сутки: в течение этого времени запросы обслуживаются из кеша, после — Galleon проверяет у внешнего реестра, не изменился ли артефакт.
- `0` — проверять внешний реестр при каждом запросе. Содержимое берётся из кеша, если реестр отвечает «не изменилось», но каждый запрос ждёт ответа внешнего реестра.
- `-1` — кешировать бессрочно, к внешнему реестру больше не обращаться. Подходит для неизменяемого содержимого и для закрытого контура.

Какие файлы к какому классу относятся:

Формат	**Метаданные (`metadataMaxAge`)**	**Контент (`contentMaxAge`)**
Docker (OCI)	Теги: какой образ стоит за тегом, список тегов	— (манифесты и слои неизменяемы, объяснение ниже)
Maven	`maven-metadata.xml` и его контрольные суммы	Артефакты сборки (`.jar`, `.pom`) и их контрольные суммы
Go	Список версий и запрос последней версии	— (файлы версий неизменяемы, объяснение ниже)
npm	Дескриптор пакета	Архивы версий
PyPI	Страница проекта в индексе	Файлы дистрибутивов
RubyGems	Индексы Compact Index (списки версий и имён)	Файлы гемов
Cargo	Индекс крейта	Файлы крейтов
raw	—	Все файлы

Файлы, которые запрашиваются по контрольной сумме, кешируются бессрочно независимо от обеих настроек: их содержимое не может измениться. Это файлы версий Go-модулей, слои и манифесты образов контейнеров. Для образов срок `metadataMaxAge` задаёт, как часто Galleon проверяет у внешнего реестра, не указывает ли тег на другой образ.

У Maven-репозиториев с политикой версий RELEASE контент по умолчанию кешируется бессрочно: релизные версии не переиздаются.

По истечении срока артефакт не удаляется из кеша: Galleon лишь проверяет его актуальность при следующем запросе. Вытеснить закешированное можно политикой очистки — она удаляет компоненты по возрасту, давности скачивания и пути (Политики очистки).

Сбросить кеш можно и не дожидаясь истечения сроков — кнопкой «Сбросить кеш» в настройках любого proxy-репозитория. Содержимое при этом не удаляется, а помечается устаревшим: каждый следующий запрос сначала сверяется с внешним реестром и только потом отдаётся. Заодно очищается кеш отказов. Пометка действует при любом сроке хранения, включая `-1`; содержимое, запрошенное по контрольной сумме, она не затрагивает.

Кеширование отказов

Ответы «не найдено» от внешнего реестра тоже кешируются — по умолчанию на 15 минут. Это защищает внешний реестр от потока повторных запросов к несуществующим путям.

Если пакет только что опубликован во внешнем реестре, а Galleon отвечает, что его нет, вероятно, действует кеш отказов: он истекает сам, по умолчанию через 15 минут, а сбросить его сразу можно кнопкой «Сбросить кеш» в настройках репозитория.

Обращения к внешнему реестру

Временные ошибки внешнего реестра (перегрузка, сетевые сбои) обрабатываются повторными попытками с нарастающей задержкой; число попыток задаётся в настройках репозитория.

Если внешний реестр требует аутентификации, например приватный реестр контейнеров, учётные данные задаются там же, в настройках proxy-репозитория.

Руководство администратора

Раздел для администратора Galleon: развёртывание и конфигурация сервера, управление репозиториями и доступом, эксплуатация хранилища, резервное копирование и наблюдение.

Обзор

Администратор отвечает за инстанс Galleon: развёртывание и конфигурацию сервера, репозитории, доступ пользователей, хранилище и наблюдение.

Инструменты

Инструменты администрирования:

- **Раздел «Настройки» веб-интерфейса** — репозитории, пользователи, роли и привилегии, цепочка аутентификации и анонимный доступ, сборка мусора, политики очистки. Все операции над сущностями выполняются здесь.
- **Конфигурация сервера** — файл `config.yaml` и переменные окружения `GALLEON_*`: адреса, база данных, хранилище, лимиты, ключи и параметры фоновых процессов. Применяется перезапуском сервера.

Те же операции доступны программно — через встроенный API.

> Операции раздела «Настройки» доступны любому пользователю с соответствующими привилегиями. Пункты, недоступные по правам, отображаются неактивными с подсказкой «Недоступно с вашими правами».

Маршрут администратора

Типовой порядок ввода инстанса в работу:

1. Установите Galleon: PostgreSQL, каталог хранилища, TLS.
2. Выполните первичную настройку: войдите bootstrap-администратором, создайте рабочие учётные записи.
3. Создайте репозитории и настройте доступ.
4. Настройте резервное копирование и наблюдение.

Установка

Для развёртывания Galleon нужны сам сервер и PostgreSQL. Сервер — один процесс, по умолчанию он принимает запросы на порту `8080` и может отдавать HTTPS сам, если задать

сертификат и ключ (TLS). В PostgreSQL хранятся метаданные инсталляции. При необходимости перед сервером ставят обратный прокси (Обратный прокси).

Сами артефакты хранятся файлами в каталоге хранилища на диске сервера (Хранилище и сборка мусора).

> За дистрибутивом Galleon обратитесь к вендору.

Системные требования

Минимальные требования к серверу:

- Процессор и память: 4 vCPU и 4 ГБ RAM.
- Диск: самому серверу дисковое пространство почти не требуется — объём определяется хранимыми артефактами (каталог хранилища) и базой данных.
- PostgreSQL версии 16 или новее. При настройках сервера по умолчанию базе данных нужно `max_connections` не менее 110.
- База данных и роль, указанные в конфигурации, создаются заранее. Galleon подключается к базе данных под этой ролью и от неё же применяет миграции, поэтому роли нужны права на создание таблиц и расширения `pg_stat_statements`.

Первый запуск

Минимальная конфигурация состоит из четырёх блоков (полный перечень параметров приведён в разделе Параметры конфигурации):

```
database:
  host: localhost
  port: 5432
  user: galleon
  password_env: GALLEON_DATABASE_PASSWORD
  dbname: galleon

storage:
  default:
    type: local
    local:
      root: /var/lib/galleon/data

security:
  secret_key_file: /etc/galleon/secret-key.yaml

auth:
  bootstrap_admin:
    username: admin
    password_env: GALLEON_BOOTSTRAP_ADMIN_PASSWORD
```

Чтобы запустить сервер, выполните следующие действия:

1. Сохраните конфигурацию в файл, например `/etc/galleon/config.yaml`.
2. Создайте ключ шифрования и запишите его в файл ключей (подробнее — в разделе Секреты и ключ шифрования):

   ```
   openssl rand -base64 32
   ```

   ```
   # /etc/galleon/secret-key.yaml
   active: 1
   keys:
     - id: 1
       key: "<BASE64>"
   ```

3. Задайте пароли в переменных окружения, имена которых указаны в конфигурации:

   ```
   export GALLEON_DATABASE_PASSWORD='<пароль базы данных>'
   export GALLEON_BOOTSTRAP_ADMIN_PASSWORD='<пароль первого администратора>'
   ```

4. Запустите сервер:

   ```
   galleon --config /etc/galleon/config.yaml
   ```

При первом запуске создаются первый администратор и стартовый набор репозиториев.

Миграции при старте

Запущенный без флагов сервер сам приводит схему базы данных к нужной версии и продолжает обслуживание — отдельный шаг для этого не нужен. Когда серверов инстанса несколько, миграции выполняет один из них, остальные ждут своей очереди и запускаются, когда схема готова.

Миграции можно вынести в отдельный запуск — например, чтобы обновить схему до развёртывания новой версии. Для этого есть два взаимоисключающих флага:

- `--migrate-only` — применить миграции и завершиться, не начиная обслуживание;
- `--skip-migrate` — не применять миграции, а проверить, что схема не старее требуемой, и начать обслуживание. Если схема старее, сервер не стартует.

Если база данных на момент запуска ещё поднимается, сервер завершается сразу. Чтобы он её дождался, задайте бюджет ожидания:

```
database:
  migrate_wait: 5m
```

Ожидание распространяется только на признаки «база ещё не готова»: отказ в соединении, неразрешимое имя, тайм-аут подключения, запуск или восстановление сервера базы данных, отсутствующая база. Неверные учётные данные под ожидание не подпадают — сервер завершается сразу.

Код выхода отличает временную причину от той, что требует вмешательства:

Код	Значение
`0`	Миграции применены
`1`	Запуск не удался: неверные учётные данные, ошибка SQL. Нужно разобраться в причине
`75`	Запуск прерван внешней причиной: сигнал остановки или исчерпанный бюджет ожидания базы данных. Достаточно запустить сервер снова

Прерывать миграции можно в любой момент. Незавершённый шаг откатывается целиком, и следующий запуск продолжает с него.

Проверка

Убедитесь, что сервер отвечает:

```
curl http://localhost:8080/healthz
curl http://localhost:8080/readyz
```

`healthz` подтверждает, что процесс жив; `readyz` — что доступна база данных. Подробнее — в разделе Мониторинг.

Первичная настройка

При первом старте с пустой базой данных Galleon создаёт первого администратора и стартовый набор репозиториев.

Bootstrap-администратор

Первая учётная запись администратора описывается блоком `auth.bootstrap_admin` конфигурации: имя пользователя и пароль — через переменную окружения (`password_env`) либо сразу bcrypt-хешем (`password_hash` , имеет приоритет).

Учётная запись создаётся, только если роль `admin` не назначена ни одному пользователю.

> ⚠ После того как администратор создан, блок `bootstrap_admin` игнорируется: смена значения переменной с паролем не меняет пароль существующей учётной записи (в логе сервера появится предупреждение «bootstrap_admin is configured but an admin already exists»). Пароль меняется только через веб-интерфейс.
> Средств восстановления утерянного пароля администратора в Galleon нет: сбросить его может только другой пользователь с административными правами. Поэтому сразу создайте вторую административную учётную запись — иначе единственным способом вернуть доступ останется развёртывание инстанса заново.

Опция `force_password_change: true` делает bootstrap-пароль одноразовым: до смены пароля пользователь заблокирован и в веб-интерфейсе, и в клиентах форматов.

Стартовое состояние

На свежей базе создаются:

- встроенная роль `admin` с полным доступом и роль `anonymous` для анонимного чтения;
- каталог привилегий (Управление доступом);
- репозитории стартового набора (таблица в разделе Репозитории): proxy-зеркала Maven Central, proxy.golang.org, базы контрольных сумм Go и Docker Hub — с включённой автоблокировкой при недоступности внешнего реестра — и hosted-репозитории для Maven (релизы и snapshot-версии), Go и образов контейнеров.

Роль `anonymous` заранее получает привилегии чтения четырёх публичных proxy-зеркал, однако анонимный доступ не действует, пока не включён глобальный переключатель («Настройки» →

«Анонимный доступ»). Hosted-репозитории стартового набора анонимно недоступны в любом случае.

Стартовый набор создаётся, только когда список репозиториев пуст: удалённые по одному репозитории при перезапуске не возвращаются. Если же удалить все репозитории до единого, при следующем старте набор будет создан заново вместе с их привилегиями и анонимными правами чтения.

Первые шаги

После первого входа bootstrap-администратором выполните следующие действия:

1. Создайте персональные учётные записи для всех, кто администрирует инстанс, и назначьте им роль `admin` («Настройки» → «Пользователи»). Не работайте под bootstrap-учётной записью постоянно.
2. Создайте роли для команд разработки из привилегий нужных репозиториев («Настройки» → «Роли», Управление доступом).
3. Решите, нужна ли пользователям самостоятельная смена пароля: привилегия `nx-userschangepw` не входит ни в одну роль по умолчанию — выдайте её ролям пользователей, иначе пароли меняет только администратор.
4. Настройте TLS, при необходимости обратный прокси и резервное копирование.

Конфигурация

Сервер настраивается файлом `config.yaml` (путь передаётся флагом `--config`). Настройки применяются при старте — изменения требуют перезапуска сервера. Сущности (репозитории, пользователи, роли) в файле не описываются — ими управляют через веб-интерфейс.

Перечень всех параметров со значениями по умолчанию — в разделе Параметры конфигурации.

Форматы значений

Длительности и размеры записываются одинаково и в файле, и в переменных окружения:

- длительность — строка с единицей измерения: `s` — секунды, `m` — минуты, `h` — часы; допустимы составные значения (`1h30m`). Число без единицы измерения сервер отклоняет при старте;
- размер — строка с единицей измерения: `K`, `M`, `G`, `T` (или `KB`, `MB`, `GB`, `TB`); множители двоичные, поэтому `256K` — это 262 144 байта. Число без единицы измерения означает байты, `0` снимает ограничение.

Переменные окружения

Любой ключ файла переопределяется переменной `GALLEON_*`: точки заменяются подчёркиваниями, `database.host` → `GALLEON_DATABASE_HOST`, `gc.grace_period` → `GALLEON_GC_GRACE_PERIOD`.

Правила:

- переменная окружения имеет приоритет над значением из файла;
- списки задаются через запятую: `GALLEON_SERVER_MONITORING_ALLOWLIST="127.0.0.0/8,::1/128"`;
- не оставляйте в файле пустые родительские блоки (например, голый `security:` без ключей) — они могут помешать применению вложенных переменных окружения.

Обязательные параметры

Значений по умолчанию нет у следующих ключей (без них сервер не стартует):

- `database.user` и `database.dbname` — учётные данные базы данных;
- `storage.default.type` и путь хранилища для выбранного типа;
- `security.secret_key_file` — путь к файлу ключей шифрования;
- `auth.bootstrap_admin` — только при первом запуске с пустой базой данных.

Что где настраивается

Параметры сгруппированы по блокам:

Блок	Что настраивает
`server`	Адрес и порт, лимиты запросов, доверие обратному прокси, доступ к health-эндпоинтам
`database`	Подключение к PostgreSQL и размеры пулов соединений
`storage`	Тип и корневой каталог файлового хранилища артефактов
`security`	Путь к файлу ключей шифрования
`auth`	Первый администратор, кеш проверенных учётных данных, время жизни сессий
`oci`	Тайм-ауты скачивания образов контейнеров из внешних реестров
`gc`	Расписание и параметры сборки мусора в хранилище
`index`	Фоновое построение служебных индексов репозиториев
`logging`	Уровень, формат и источники записей журнала
`proxy`	Защита исходящих запросов proxy-репозиториев

Типовые задачи

Ограничить размер загружаемых артефактов

Собственного лимита у Galleon по умолчанию нет (`server.max_upload_size: 0`). Если перед сервером стоит обратный прокси, согласуйте с ним размер тела запроса (Обратный прокси).

Задать лимит на стороне Galleon:

```
server:
  max_upload_size: 10GB
```

Превышение лимита клиент видит как ошибку `413` .

Включить TLS-подключение к базе данных

Режим проверки сертификата задаёт ключ `database.sslmode` :

```
database:
  sslmode: verify-full
```

Значения соответствуют режимам PostgreSQL: `require` шифрует соединение, `verify-ca` и `verify-full` дополнительно проверяют сертификат.

Собрать подробные логи для диагностики

Подробностью логирования управляют два ключа блока `logging`:

```
logging:
  level: debug
  registry_level: info
```

Первый ключ повышает подробность логов Galleon, второй — записей о работе с образами контейнеров; он нужен при разборе проблем с ними. Возвращайте прежние значения после диагностики: на уровне `debug` объём записей значительно выше.

Разрешить проксирование во внутреннюю сеть

Обращения proxy-репозиториев к приватным IP-адресам запрещены по умолчанию: loopback, диапазоны RFC 1918 (`10/8`, `172.16/12`, `192.168/16`), link-local, CGNAT (`100.64/10`), multicast, broadcast и ULA IPv6. Запрет проверяется при сохранении репозитория (по адресам, в которые разрешается имя внешнего реестра), при каждом соединении и на редиректах: адрес из заголовка `Location` проходит ту же проверку.

Чтобы разрешить конкретный внутренний адрес:

```
proxy:
  ssrf:
    allowlist:
      ips:
        - 10.0.0.10
      domains:
        - registry.internal.example.com
```

Список исключений принимает и IP-адреса, и имена хостов; имя сверяется целиком, поддомены под исключение не попадают. На приватные адреса вне списка запрет продолжает действовать. Адреса сервисов метаданных облачных провайдеров (например, `169.254.169.254`) заблокированы всегда — внести их в список исключений нельзя.

Расширить пулы соединений под нагрузку

Расширьте пулы, если соединений к базе данных перестало хватать под нагрузкой. Соединения занимают параллельные сборки, фоновое построение индексов и особенно проксирование

образов контейнеров: пока слой скачивается из внешнего реестра, он держит соединение (до двух на один `docker pull`).

Признаки нехватки:

- запросы отвечают медленно, а сама база данных при этом не загружена: активных запросов в `pg_stat_activity` мало, но число соединений Galleon упирается в `max_open_conns` — значит, запросы стоят в очереди за свободным соединением;
- слотов не хватает уже в самой базе данных: PostgreSQL перестаёт принимать новые соединения, запросы завершаются ошибкой, а в access-логе появляется поле `sqlstate` с кодом класса `53` — исчерпание ресурсов (Логирование).

Занятость пулов видна в PostgreSQL:

```
SELECT application_name, count(*) FROM pg_stat_activity GROUP BY 1 ORDER BY 2 DESC;
```

Метка `galleon` — основной пул, `galleon-oci-meta` — пул чтения метаданных образов; соединения, обслуживающие образы контейнеров, идут без метки.

```
database:
  max_open_conns: 80
  registry_max_open_conns: 40
```

По умолчанию — 40 и 25. После изменения пересчитайте `max_connections` базы данных. Galleon подключается к ней тремя пулами: два ограничены `max_open_conns` (поэтому в формуле оно удваивается), третий — `registry_max_open_conns`. Ещё несколько соединений PostgreSQL держит в резерве для суперпользователя (`superuser_reserved_connections`, по умолчанию 3):

```
max_connections >= max_open_conns × 2 + registry_max_open_conns +
superuser_reserved_connections
```

При значениях по умолчанию это 40 + 40 + 25 + 3 = 108, поэтому для промышленной инсталляции — не меньше 110. Оставьте несколько свободных слотов для `psql` и разовых операций.

Параметры конфигурации

Полный перечень параметров файла `config.yaml`, сгруппированный по блокам. Правила переопределения переменными окружения, список обязательных параметров и типовые задачи настройки описаны в разделе Конфигурация.

У параметров без значения по умолчанию приведён пример допустимого значения: он показывает форму записи, а не рекомендуемую величину.

server

Сетевые параметры, лимиты и доверие обратному прокси.

Подробнее — в разделах Обратный прокси и Мониторинг.

Параметр	Описание
`listen`	Адрес и порт, на которых сервер принимает HTTP-запросы. **По умолчанию:** `:8080`
`public_base_url`	Внешний адрес инстанса. Подставляется в адреса, которые сервер сообщает клиентам. **По умолчанию:** нет. **Пример:** `https://galleon.example.com`
`max_upload_size`	Предельный размер файла, загружаемого в репозиторий. Значение `0` снимает ограничение, и тогда фактический предел задаёт обратный прокси. **По умолчанию:** `0`. **Пример:** `10GB`
`tls.cert_file`	Путь к файлу сертификата в формате PEM. Внутри файла сначала идёт сертификат сервера, затем промежуточные сертификаты. TLS включается, только если задан и `tls.key_file` (TLS). **По умолчанию:** нет. **Пример:** `/etc/galleon/tls/fullchain.pem`
`tls.key_file`	Путь к файлу закрытого ключа в формате PEM. Ключи с парольной фразой не принимаются. TLS включается, только если задан и `tls.cert_file`. **По умолчанию:** нет. **Пример:** `/etc/galleon/tls/privkey.pem`
`max_control_body`	Предельный размер тела запроса к API управления — создания репозиториев, пользователей, ролей. **По умолчанию:** `256K`
`max_query_value_bytes`	Предельная длина одного значения в строке запроса, после знака `?`. Значение `0` снимает ограничение. **По умолчанию:** `4096`
`max_header_bytes`	

Параметр	Описание
	Предельный суммарный размер заголовков запроса в байтах. **По умолчанию:** `1048576`
`read_timeout`	Предельное время чтения запроса целиком, вместе с телом. Значение `0s` снимает ограничение, чтобы медленные загрузки успевали завершиться. **По умолчанию:** `0s`. **Пример:** `30m`
`read_header_timeout`	Предельное время чтения заголовков. Защищает от соединений, которые открыты, но запрос не присылают. **По умолчанию:** `10s`
`idle_timeout`	Время, после которого простаивающее keep-alive-соединение закрывается. **По умолчанию:** `120s`
`trusted_proxies`	Адреса обратных прокси в нотации CIDR или отдельными IP-адресами. Заголовкам запросов от них можно доверять. **По умолчанию:** `[]`. **Пример:** `["10.0.0.0/8"]`
`proxy_auth.header`	Заголовок, в котором обратный прокси передаёт общий секрет. **По умолчанию:** `X-Origin-Token`
`proxy_auth.secret_env`	Имя переменной окружения, из которой читается этот секрет. **По умолчанию:** нет. **Пример:** `GALLEON_ORIGIN_TOKEN`
`trust_forwarded_headers`	Учитывать заголовки `X-Forwarded-Host` и `X-Forwarded-Proto` при построении адресов для клиентов. **По умолчанию:** `false`
`monitoring_allowlist`	Адреса, которым доступны health-эндпоинты. Остальным они отвечают как несуществующий путь.

Параметр	Описание
	По умолчанию: `["127.0.0.0/8", "::1/128"]`. **Пример:** `["127.0.0.0/8", "10.0.0.0/8"]`

database

Подключение к PostgreSQL и пулы соединений.

При изменении размеров пулов пересчитайте `max_connections` базы данных: нужно `max_open_conns × 2 + registry_max_open_conns + superuser_reserved_connections`. Откуда берутся слагаемые, объяснено в разделе Конфигурация.

Параметр	Описание
`host`	Адрес сервера базы данных. **По умолчанию:** `localhost`. **Пример:** `postgres.internal`
`port`	Порт сервера базы данных. **По умолчанию:** `5432`. **Пример:** `6432`
`user`	Имя роли базы данных. **По умолчанию:** нет. **Пример:** `galleon`
`dbname`	Имя базы данных. **По умолчанию:** нет. **Пример:** `galleon`
`password_env`	Имя переменной окружения с паролем. Без этого ключа подключение выполняется без пароля. **По умолчанию:** нет. **Пример:** `GALLEON_DATABASE_PASSWORD`
`sslmode`	Режим TLS-подключения к базе данных: `disable`, `require`, `verify-ca` или `verify-full`. **По умолчанию:** `disable`. **Пример:** `verify-full`
`connect_timeout`	Предельное время установления соединения с базой данных. **По умолчанию:** `5s`
`migrate_wait`	Сколько ждать базу данных перед миграциями, если она ещё поднимается. `0` — не ждать и завершиться сразу. **По умолчанию:** `0s`. **Пример:** `5m`
`max_open_conns`	Размер каждого из двух основных пулов соединений. **По умолчанию:** `40`. **Пример:** `80`

Параметр	Описание
`max_idle_conns`	Число соединений, которые эти пулы держат открытыми в простое. **По умолчанию:** `5`
`registry_max_open_conns`	Размер отдельного пула соединений для работы с образами контейнеров. **По умолчанию:** `25`. **Пример:** `40`
`statement_timeout`	Предельное время выполнения одного SQL-запроса. На миграции не распространяется. **По умолчанию:** `30s`
`lock_timeout`	Предельное время ожидания блокировки в базе данных. **По умолчанию:** `10s`
`idle_in_transaction_session_timeout`	Предельное время простоя внутри открытой транзакции. **По умолчанию:** `60s`
`max_conn_lifetime`	Время жизни соединения, после которого оно пересоздаётся. **По умолчанию:** `30m`
`max_conn_idle_time`	Время простоя соединения, после которого оно закрывается. **По умолчанию:** `5m`

storage

Файловое хранилище артефактов. Ключи задаются для именованного хранилища; backend с именем `default` обязателен.

Устройство каталога описано в разделе Хранилище и сборка мусора.

Параметр	Описание
`type`	Тип хранилища. Поддерживается только `local`. **По умолчанию:** нет. **Пример:** `local`
`local.root`	Корневой каталог хранилища. **По умолчанию:** нет. **Пример:** `/var/lib/galleon/data`

security

Шифрование хранимых секретов.

Подробнее — в разделе Секреты и ключ шифрования.

Параметр	Описание
`secret_key_file`	Путь к файлу ключей шифрования. Параметр обязателен: без него сервер не запускается. **По умолчанию:** нет. **Пример:** `/etc/galleon/secret-key.yaml`

auth

Первый администратор, кеш проверки паролей и сессии веб-интерфейса.

Подробнее — в разделе Первичная настройка.

Параметр	Описание
`bootstrap_admin.username`	Имя первого администратора. **По умолчанию:** нет. **Пример:** `admin`
`bootstrap_admin.password_env`	Имя переменной окружения с паролем первого администратора. **По умолчанию:** нет. **Пример:** `GALLEON_BOOTSTRAP_ADMIN_PASSWORD`
`bootstrap_admin.password_hash`	Bcrypt-хеш пароля. Имеет приоритет над `password_env`. **По умолчанию:** нет. **Пример:** `$2a$12$LQv3c1yqBWVHxkd0...`
`bootstrap_admin.force_password_change`	Требовать смену пароля при первом входе. **По умолчанию:** `false`
`credential_cache_ttl`	Сколько результат проверки пароля хранится в памяти. Значение `0s` отключает кеш, и пароль проверяется заново на каждый запрос клиента. **По умолчанию:** `60s`. **Пример:** `0s`
`session_max_lifetime`	Предельное время жизни сессии веб-интерфейса независимо от активности. **По умолчанию:** `168h`. **Пример:** `24h`
`session_idle_timeout`	Время бездействия, после которого сессия завершается. **По умолчанию:** `168h`. **Пример:** `8h`

oci

Параметры скачивания образов контейнеров из внешних реестров.

Параметр	Описание
`proxy.blob_idle_window`	Допустимая пауза в передаче данных при скачивании слоя из внешнего реестра. После неё соединение обрывается. **По умолчанию:** `60s`
`proxy.db_mirror_timeout`	Предельное время записи метаданных скачанного слоя в базу данных. **По умолчанию:** `60s`

gc

Сборка мусора в хранилище. Семантика описана в разделе Хранилище и сборка мусора.

Параметр	Описание
`auto`	Плановые циклы по расписанию. Ручной запуск работает независимо от этого значения. **По умолчанию:** `true`. **Пример:** `false`
`interval`	Пауза между плановыми циклами. **По умолчанию:** `5m`. **Пример:** `15m`
`grace_period`	Минимальный возраст файла, допускающий его удаление. **По умолчанию:** `24h`. **Пример:** `5m`
`batch_size`	Сколько файлов-кандидатов отбирается за один запрос к базе данных. **По умолчанию:** `500`. **Пример:** `1000`
`max_duration`	Предельная длительность одного цикла. Остаток обрабатывает следующий цикл. **По умолчанию:** `15m`. **Пример:** `5m`

index

Фоновое построение индексов репозиториев — служебных файлов, которые Galleon генерирует сам: списков версий, метаданных пакетов.

Число обработчиков подбирается по нагрузке: чем их больше, тем быстрее индексы обновляются после публикаций. **Значение `workers: 0` полностью останавливает фоновое построение индексов** и в промышленной установке задаваться не должно.

Параметр	Описание
`workers`	Число обработчиков очереди построения. **По умолчанию:** `4`. **Пример:** `8`
`poll_interval`	Пауза между опросами очереди заданий. **По умолчанию:** `1s`. **Пример:** `5s`
`max_backoff`	Предельная задержка перед повтором задания, завершившегося ошибкой. **По умолчанию:** `1h`. **Пример:** `15m`
`lease_duration`	Время, на которое обработчик забирает задание. Если он не завершил работу, задание возвращается в очередь. **По умолчанию:** `5m`. **Пример:** `10m`
`backfill_on_start`	Достраивать недостающие индексы при запуске сервера. **По умолчанию:** `true`. **Пример:** `false`

logging

Журнал сервера.

Подробнее — в разделе Логирование.

Параметр	Описание
`level`	Порог логов Galleon: `debug`, `info`, `warn` или `error`. **По умолчанию:** `info`. **Пример:** `debug`
`format`	Формат записей журнала. **По умолчанию:** `json`. **Пример:** `text`
`registry_level`	Отдельный порог для записей о работе с образами контейнеров. **По умолчанию:** `warn`. **Пример:** `info`

proxy

Защита исходящих запросов proxy-репозиториев.

Адреса сервисов метаданных облачных провайдеров заблокированы всегда, независимо от этих настроек.

Параметр	Описание
`ssrf.block_private_networks`	Запрещать обращения proxy-репозиториев к приватным сетям. **По умолчанию:** `true`. **Пример:** `false`
`ssrf.allowlist.ips`	Адреса, разрешённые в обход запрета. **По умолчанию:** `[]`. **Пример:** `["10.0.0.10"]`
`ssrf.allowlist.domains`	Домены, разрешённые в обход запрета. **По умолчанию:** `[]`. **Пример:** `["registry.internal.example.com"]`

Секреты и ключ шифрования

Значения секретов никогда не записываются в `config.yaml`. Файл конфигурации содержит только имена переменных окружения или пути к файлам:

Ключ	Назначение
`database.password_env`	Имя переменной с паролем PostgreSQL
`auth.bootstrap_admin.password_env`	Имя переменной с паролем первого администратора
`server.proxy_auth.secret_env`	Имя переменной с секретом обратного прокси
`security.secret_key_file`	Путь к файлу ключей шифрования

Если ключ `*_env` задан, а сама переменная пуста или отсутствует, сервер не стартует — сообщение об ошибке называет переменную.

Файл ключей шифрования

`security.secret_key_file` — обязательный параметр: без файла ключей сервер не запускается. Ключом шифруются секреты, которые хранятся в базе данных: учётные данные внешних реестров proxy-репозиториев и служебный ключ подписи загрузок образов контейнеров. Пароли пользователей и токены хранятся хешами и этим ключом не шифруются.

Формат файла:

```
active: 1
keys:
  - id: 1
    key: "<base64-значение 32 байт>"
```

Значение генерируется командой `openssl rand -base64 32`. Права доступа к файлу сервер не проверяет — ограничьте их средствами системы развёртывания.

> Файл ключей — часть данных инстанса. При его утере сервер не запустится, а сохранённые учётные данные внешних реестров будут безвозвратно потеряны. Не перегенерируйте файл при развёртывании: если инструмент деплоя создаёт секрет заново при каждом обновлении, все серверы перестанут стартовать. Резервируйте файл вместе с базой данных (Резервное копирование).

Ротация ключа

Ротация выполняется в три фазы, с перезапуском серверов после первых двух:

1. **Добавление.** Добавьте новый ключ в `keys` со следующим `id`, не меняя `active`. Разверните файл на все серверы и перезапустите их.

2. **Переключение.** Смените `active` на `id` нового ключа, разверните и перезапустите. Новые записи шифруются новым ключом; служебный ключ загрузки образов перешифровывается автоматически при старте.
3. **Перешифрование учётных данных.** Откройте «Изменить» каждого proxy-репозитория с сохранёнными учётными данными внешнего реестра и введите пароль или токен заново — только так они пересохраняются под новым ключом. Автоматически это не происходит.

Пока перешифрованы не все записи, сервер при старте пишет предупреждение `secrets: upstream credentials still sealed under a non-primary key` с числом таких записей в поле `stragglers`. Старый ключ удаляйте из файла только после того, как предупреждение исчезло.

> ⚠ Если удалить старый ключ, не выполнив третью фазу, репозитории с сохранёнными учётными данными перестанут работать: каждый запрос к ним будет завершаться ошибкой, а вернуть секреты можно будет только вводом заново.

Все серверы инстанса обязаны использовать один и тот же файл ключей.

Репозитории

Репозитории управляются на странице «Настройки» → «Репозитории». Типы репозиториев, форматы и кеширование описаны в разделах Репозитории и форматы и Проксирование и кеширование.

Список репозиториев

В списке видны репозитории, на администрирование которых у вас есть права. Колонки: «Имя», «Формат», «Тип», «URL», «В сети», «Источник».

- «URL» — адрес для клиентов; для репозиториев Docker (OCI) показывается pull-референс с пометкой «pull ref».
- «В сети» — обслуживается ли репозиторий клиентами. При значении `no` запросы к нему отклоняются, а содержимое остаётся на месте: так репозиторий выводят из обслуживания, не удаляя.
- «Источник» — состояние соединения с внешним реестром (только proxy):

Состояние	Значение
«Готов»	Обращений к внешнему реестру ещё не было
«Доступен»	Внешний реестр отвечает
«Автоблокировка»	Обращения приостановлены автоматически: реестр недоступен; в подсказке — время следующей попытки
«Заблокирован»	Обращения заблокированы вручную настройкой репозитория
«Недоступен»	Реестр недоступен (автоблокировка выключена)

Под состоянием печатается причина, если она известна: `unauthorized` — реестр отклонил учётные данные; `service unavailable`, `bad gateway`, `gateway timeout` — ответ самого реестра; `upstream connection failure` — запрос до реестра не дошёл; `background probe failed` — не прошла фоновая проверка доступности. Причина у состояния «Доступен» означает особый случай: реестр отвечает, но отклоняет учётные данные, а автоблокировка выключена. Состояние в этом случае подсвечено жёлтым, а незакешированные запросы отвечают `502`.

Автоблокировка снимается сама, когда внешний реестр отвечает снова. Чтобы снять её досрочно, когда реестр уже починили, а окно следующей попытки ещё не подошло, сохраните настройки репозитория кнопкой «Сохранить»: счётчик автоблокировки сбрасывается.

Создание

Нажмите «+ Новый репозиторий» и заполните форму:

1. «Имя», «Формат», «Тип» и хранилище фиксируются при создании — изменить их позже нельзя.
2. Чекбокс «в сети» оставляет репозиторий в обслуживании; сняв его, репозиторий можно создать заранее и открыть клиентам позже.
3. `writePolicy` (только hosted) — что клиентам разрешено делать с содержимым:
   - `ALLOW` — публиковать, перезаписывать и удалять;
   - `ALLOW_ONCE` — публиковать и удалять, но не перезаписывать уже опубликованное. Удаление разрешено, поэтому «удалить и опубликовать заново» остаётся способом заменить версию: политика запрещает перезапись, а не гарантирует неизменяемость;
   - `DENY` — только чтение: отклоняются и публикация, и перезапись, и удаление.

Политика действует на всех путях записи: и на клиентах форматов, включая `docker push`, и на удалении в веб-интерфейсе. Для npm, RubyGems, PyPI и Cargo форма предлагает `ALLOW_ONCE` — версии этих форматов принято делать неизменяемыми.

4. Для proxy-репозитория заполните блок кеширования:
   - «Remote URL» — адрес внешнего реестра; плейсхолдер подсказывает стандартный адрес для каждого формата;
   - `contentMaxAge` (минут) и `metadataMaxAge` (минут) — сроки хранения кеша: положительное число задаёт минуты (по умолчанию 1440), `0` — проверять внешний реестр при каждом запросе, `-1` — кешировать бессрочно. Что относится к контенту, а что к метаданным, описано в разделе Проксирование и кеширование;
   - `negativeCache.enabled` и `negativeCache.timeToLive` (минут) — кеш отказов (по умолчанию 15 минут).
5. «Блокировать исходящие соединения» — ручная заморозка обращений к внешнему реестру: отдаётся только кеш, остальное отвечает `404`.
6. «Автоблокировка исходящих соединений» — автоматическая приостановка обращений, пока внешний реестр недоступен: кеш отдаётся, некешированные запросы отвечают `502`. В форме включена по умолчанию.
7. «Аутентификация» — учётные данные внешнего реестра: тип `username` (логин и пароль) или `bearerToken` (статический токен).
8. «Параметры HTTP-запросов» — число повторных попыток (`retries`), тайм-аут и суффикс User-Agent исходящих запросов.
9. «Политики очистки» — привязка политик автоудаления, доступны политики того же формата. Что они удаляют и как запускаются, описано в разделе Политики очистки.
10. Формат-специфичные поля:
    - Maven — `maven.versionPolicy` (`RELEASE` / `SNAPSHOT` / `MIXED`);
    - raw — `raw.contentDisposition` (`INLINE` — браузер открывает файлы, `ATTACHMENT` — принудительное скачивание);
    - Cargo proxy — `cargo.requireAuthentication` (клиент `cargo` будет присылать учётные данные);
    - Docker proxy — `insecureSkipTlsVerify` (отключает проверку TLS-сертификата внешнего реестра — только для доверенных приватных реестров с самоподписанными сертификатами).

При создании репозитория автоматически появляется набор его привилегий (Управление доступом).

Редактирование и обслуживание

Кнопка «Изменить» открывает те же настройки, кроме зафиксированных при создании. Сохранённые секреты не отображаются: пустое поле пароля означает «оставить прежний» (подсказка «Оставьте пустым, чтобы сохранить текущий секрет»); при смене адреса реестра или типа аутентификации секрет нужно ввести заново.

Для любого proxy-репозитория в диалоге редактирования доступна кнопка «Сбросить кеш». Она не удаляет закешированное содержимое, а помечает его устаревшим: каждый следующий запрос сначала сверяется с внешним реестром и только потом отдаётся клиенту. Заодно сбрасывается кеш отказов, то есть запомненные ответы «не найдено». Операция требует привилегии на изменение репозитория; повторный вызов ничего не меняет.

Кнопка «Запустить очистку» выполняет привязанные политики очистки по этому репозиторию немедленно, не дожидаясь планового прохода. Запуск требует подтверждения: компоненты, подходящие под критерии этих политик, удаляются безвозвратно.

Удаление репозитория

Кнопка «Удалить» удаляет репозиторий вместе с артефактами. Репозиторий исчезает из каталога сразу: с этого момента запросы к нему получают ответ «не найдено». Место на диске освобождает последующая сборка мусора.

Если удаление docker-репозитория не довели до конца (например, сервер остановили в этот момент), создание репозитория под тем же именем отвечает ошибкой `cleanup_in_progress`. Доделывает удаление сборка мусора: запустите её кнопкой «Запустить» на странице «Настройки» → «Сборка мусора».

Стартовый набор

На пустой базе создаются восемь репозиториев:

Репозиторий	Формат	Тип	Политика записи	Назначение
`maven-central`	`maven2`	`proxy`	—	Зеркало Maven Central; контент кешируется бессрочно
`maven-releases`	`maven2`	`hosted`	`ALLOW_ONCE`	Релизные Java-артефакты (неизменяемые)
`maven-snapshots`	`maven2`	`hosted`	`ALLOW`	Snapshot-версии
`go-proxy`	`go`	`proxy`	—	Зеркало proxy.golang.org
`go-hosted`	`go`	`hosted`	`ALLOW`	Внутренние Go-модули
`go-sumdb`	`raw`	`proxy`	—	База контрольных сумм Go (sum.golang.org)
`docker-hosted`	`docker`	`hosted`	`ALLOW`	Внутренние образы контейнеров (теги перезаписываемы)
`docker-hub-proxy`	`docker`	`proxy`	—	Зеркало Docker Hub, без учётных данных

У всех proxy-репозиториев набора включена автоблокировка. Роль `anonymous` заранее имеет привилегии чтения четырёх публичных зеркал — доступ активируется глобальным переключателем анонимного доступа.

Управление доступом

Доступ управляется на страницах «Пользователи», «Роли», «Права», «Анонимный доступ» и «Реалмы» раздела «Настройки». На этой странице собраны операции: как завести и отключить

пользователя, собрать роль, включить анонимный доступ и изменить цепочку аутентификации. Как устроена сама модель доступа, описано в разделе Управление доступом.

Пользователи

Страница «Настройки» → «Пользователи». Кнопка «+ Новый пользователь» создаёт локальную учётную запись: идентификатор, имя, почта, пароль, статус и стартовая роль.

Статусы: «Активен», «Отключён» и «Требовать смену пароля» — пароль становится одноразовым: до его смены пользователь заблокирован и в веб-интерфейсе, и в клиентах форматов.

Перевод в «Отключён» — штатный способ немедленно отозвать доступ: сессии пользователя отзываются, его токен удаляется, а запросы перестают проходить и в веб-интерфейсе, и в клиентах (если серверов несколько, на остальных — с задержкой до минуты). Возврат в «Активен» доступ восстанавливает, но токен придётся выпустить заново.

Роли назначаются прямо в списке (кнопка «Добавить роль»; крестик рядом с ролью снимает её) или в диалоге «Изменить» (там список ролей задаётся целиком; минимум одна роль).

Смена чужого пароля (кнопка «Сменить пароль») доступна только с привилегией `nx-all` (роль `admin`); привилегий области управления `users` недостаточно. Существующие сессии и токен пользователя при смене пароля продолжают действовать — чтобы прекратить доступ немедленно, переведите учётную запись в «Отключён».

Роли

Страница «Настройки» → «Роли». Роль собирается из привилегий (по именам) и вложенных ролей; привилегии вложенных ролей тоже достаются пользователю. Встроенные роли помечены как «только чтение».

> Galleon не даст снять последнего администратора: операция, после которой роль `admin` не останется ни у одного активного пользователя (напрямую или через вложенные роли), отклоняется. Защита действует только на роль `admin`: собственная роль с теми же привилегиями под неё не попадает.

Привилегии

Страница «Настройки» → «Права» показывает каталог привилегий инсталляции с фильтром по типу и позволяет создать собственную кнопкой «+ Новое право».

Типы привилегий и их поля, действия над содержимым и над самим репозиторием, устройство имён, ограничения на собственные привилегии и готовые наборы под типовые задачи — в разделе Привилегии.

Анонимный доступ

Страница «Настройки» → «Анонимный доступ» — глобальный переключатель «Включить анонимный доступ» (по умолчанию выключен). Что именно доступно анонимно, определяют привилегии, выданные роли `anonymous` (через страницы «Роли»/«Пользователи»).

> ⚠ Перед включением на публичной инсталляции проверьте набор привилегий роли `anonymous`: каждый неаутентифицированный запрос выполняется с этими правами.

Отключается анонимный доступ тем же переключателем. Изменение вступает в силу сразу; если серверов несколько, на остальных — с задержкой до минуты.

> ℹ Не отключайте анонимный доступ снятием привилегий с роли `anonymous`: если у роли не останется ни привилегий, ни пользователей, при следующем перезапуске сервер выдаст ей стартовые привилегии заново. Выключайте анонимный доступ только переключателем «Включить анонимный доступ».

Цепочка аутентификации

Страница «Настройки» → «Реалмы» управляет составом и порядком реалмов: запрос проверяется по цепочке сверху вниз, побеждает первый подходящий. Реалмы с пометкой `stub` пока не работают: они отклоняют все запросы.

> ✖ Не убирайте realm `cookie-session` из активной цепочки — без него вход в веб-интерфейс станет невозможен. Сохранение такой конфигурации блокируется.

Привилегии

Привилегия — минимальная единица прав: она разрешает набор действий над одним объектом, репозиторием или административной областью. Пользователю привилегии не выдаются напрямую: их перечисляет роль, а пользователь получает роли (модель целиком — в разделе Управление доступом).

Все привилегии инсталляции видны на странице «Настройки» → «Права», фильтр по типу сужает список. Каталог наполняется автоматически: административные привилегии создаются при первом запуске, привилегии на репозиторий — вместе с самим репозиторием. Для большинства задач достаточно выбрать из каталога нужные имена.

Типы

Тип определяет, над чем привилегия работает и какие у неё поля.

Тип	Над чем работает	Поля
`repository-view`	Содержимое репозитория: артефакты	Формат, репозиторий, действия
`repository-admin`	Сам репозиторий: его настройки и существование	Формат, репозиторий, действия
`application`	Административная область: пользователи, роли, привилегии, настройки, задачи	Домен, действия

Одно и то же действие в двух семействах репозитория означает разное: `add` в `repository-view` — опубликовать артефакт, `add` в `repository-admin` — создать репозиторий.

Действия над содержимым

Привилегии `repository-view` оперируют пятью действиями:

Действие	Что разрешает
`browse`	Видеть репозиторий и его содержимое в веб-интерфейсе
`read`	Скачивать артефакты клиентом формата
`add`	Публиковать новые артефакты
`edit`	Изменять уже опубликованное содержимое
`delete`	Удалять артефакты

Что разрешают `read` и `browse`, зависит от того, куда идёт запрос, и одинаково для всех форматов.

`read` покрывает всё, что клиент формата получает при обычной работе: и сам файл, и служебные списки, из которых он собирает зависимости, — `maven-metadata.xml` с контрольными суммами, список версий Go-модуля, дескриптор пакета npm, индексы PyPI, RubyGems и Cargo, список тегов образа. Сборочному агенту, который только скачивает, достаточно одного `read`.

`browse` нужен для веб-интерфейса: без него репозиторий не появится в разделе «Обзор», даже если из него можно качать. Карточка репозитория, списки компонентов и ассетов открываются по любому из двух действий.

> ⚠ Действия независимы: `read` не включает `browse`, `edit` не включает `add`. Иерархии «администрирование включает запись, запись включает чтение» нет — нужный набор действий указывается явно.

Исключение — публикация образов контейнеров: клиент запрашивает у реестра доступ и на чтение, и на запись сразу, поэтому для `docker push` нужны оба действия — `read` и `add`.

Действия над репозиторием

Привилегии `repository-admin` управляют самим репозиторием: его настройками и существованием.

Действие	Что разрешает
`read`	Просматривать настройки репозитория
`add`	Создавать репозитории
`edit`	Изменять настройки, в том числе сбрасывать кеш proxy-репозитория
`delete`	Удалять репозиторий

Действие `add` — единственное, которое не относится к существующему репозиторию: в момент создания его ещё нет. Поэтому право создавать репозитории выдаётся привилегией, охватывающей не один репозиторий, а все сразу: `nx-repository-admin-*-*-add` разрешает создавать репозитории любого формата. Как устроены такие имена, разобрано ниже.

Административные области

Привилегии типа `application` закрывают операции, не связанные с конкретным репозиторием. Область называется доменом:

Домен	Что закрывает
`users`	Учётные записи: создание, просмотр, изменение, удаление
`roles`	Роли
`privileges`	Привилегии
`settings`	Системные настройки: анонимный доступ, цепочка реалмов, политики очистки
`tasks`	Задачи: просмотр состояния, запуск и приостановка
`userschangepw`	Самостоятельная смена собственного пароля

Глаголы у административных привилегий свои: `create`, `read`, `update`, `delete` и `all`. Привилегия на изменение включает и чтение своей области: пользователь с `nx-users-update` видит список пользователей, отдельная `nx-users-read` для этого не нужна. У домена `settings` нет `create` и `delete` — настройки не создают и не удаляют, их читают и меняют.

Имена привилегий

Имена привилегий построены по шаблону, в который входят область или формат, репозиторий и действие. Зная шаблон, имя нужной привилегии можно составить самостоятельно и найти его в каталоге поиском:

- `nx-<DOMAIN>-<VERB>` — административные: `nx-users-create`, `nx-roles-read`, `nx-settings-update`;
- `nx-repository-view-<FORMAT>-<REPOSITORY>-<ACTION>` — содержимое: `nx-repository-view-maven2-maven-releases-read`;
- `nx-repository-admin-<FORMAT>-<REPOSITORY>-<ACTION>` — сам репозиторий: `nx-repository-admin-docker-docker-hosted-edit`.

В позиции формата и репозитория допустима подстановка `*`: `nx-repository-view-*-*-read` — чтение во всех репозиториях, `nx-repository-view-docker-*-browse` — просмотр всех репозиториев формата Docker (OCI).

Из шаблона выпадают две привилегии:

- `nx-tasks-run` — запуск задач и приостановка сборки мусора. Только эта привилегия даёт такое право: `nx-tasks-read` позволяет лишь смотреть состояние;
- `nx-userschangepw` — самостоятельная смена пароля; не входит ни в одну роль по умолчанию (Первичная настройка).

Полный доступ ко всем операциям даёт встроенная привилегия `nx-all` — она входит в роль `admin`.

Откуда привилегии берутся

Каталог наполняется без участия администратора:

- при первом запуске создаются административные привилегии по каждому домену и привилегии с подстановкой: на все репозитории сразу и на все репозитории каждого формата;
- при создании репозитория появляется набор привилегий на него: на каждое действие с содержимым и на каждое действие с самим репозиторием. Удаляются они вместе с репозиторием.

Встроенные привилегии доступны только для чтения: изменить или удалить их нельзя. В роли они включаются так же, как любые другие.

Собственные привилегии

Прежде чем создавать свою привилегию, поищите подходящую в каталоге: большинство задач закрывается встроенными именами.

Встроенный каталог не покрывает один случай — **администратор всех репозиториев одного формата**. Готовые привилегии `repository-admin` с подстановкой есть только на все форматы сразу (`nx-repository-admin-*-*-<ACTION>`), а привилегия «все Maven-репозитории и никакие другие» создаётся вручную: тип `repository-admin`, формат `maven2`, репозиторий `*`, нужные действия.

Создать привилегию можно на странице «Настройки» → «Права» кнопкой «+ Новое право». Правила при создании:

- **Имя.** Имена, начинающиеся с `nx-`, зарезервированы за встроенным каталогом. Для своих привилегий выберите другой префикс, например `team-`.
- **Формат и репозиторий.** Если указан конкретный репозиторий, он должен существовать и принадлежать выбранному формату: привилегия с форматом `maven2` и репозиторием `docker-hosted` не создастся. Если в позиции репозитория стоит `*`, формат всё равно должен быть одним из поддерживаемых Galleon.
- **Домен.** Домен административной привилегии со списком доменов не сверяется. Опечатка вроде `user` вместо `users` ошибки не вызовет: привилегия сохранится, её можно будет включить в роль, но прав она не даст ни на одну операцию. Сверяйтесь с таблицей доменов выше.

> ⚠ Удаляя репозиторий, удалите и созданные вручную привилегии на него. Автоматические удаляются вместе с ним, а ваши остаются в ролях и снова начнут действовать, если репозиторий с тем же именем и форматом создадут заново.

Типовые роли

Что положить в роль под конкретную задачу:

Задача	Привилегии
Сборочный агент только скачивает	`nx-repository-view-*-*-read`
Сборочный агент публикует в один репозиторий	`nx-repository-view-<FORMAT>-<REPOSITORY>-add`; для образов контейнеров добавьте `-read`
Разработчик работает через веб-интерфейс	`browse` и `read` на нужные репозитории
Администратор репозиториев одного формата	Собственная привилегия `repository-admin` с форматом `docker` и `*` в позиции репозитория, действия `read`, `edit`, `delete` — управление всеми репозиториями Docker (OCI) и никакими другими
Дежурный по обслуживанию	`nx-tasks-read` и `nx-tasks-run` — видеть состояние сборки мусора, запускать её и ставить на паузу

Роль собирается на странице «Настройки» → «Роли». В роль можно включить другие роли: пользователь с такой ролью получает и её собственные привилегии, и привилегии включённых в неё ролей. Те тоже могут содержать роли, и привилегии собираются по всей цепочке. Так делают базовую роль с общими правами и добавляют её в роли команд. Роль нельзя включить в саму себя, в том числе через промежуточные роли: сервер отклонит такую вложенность.

Хранилище и сборка мусора

Galleon хранит данные в два слоя: метаданные (каталог артефактов, пользователи, настройки) — в PostgreSQL, содержимое артефактов — файлами на диске в каталоге хранилища (`storage.default.local.root`).

Устройство каталога хранилища

Корень хранилища содержит:

- `blobs/` — содержимое пакетных форматов. Файл именуется хешем SHA-256 своего содержимого, поэтому одинаковые данные хранятся в одном экземпляре, сколько бы репозиториев и путей на них ни ссылалось.
- `docker/` — манифесты и слои образов контейнеров, со своей раскладкой. Слои тоже адресуются по контрольной сумме и разделяются между образами; учёт ведётся в отдельных таблицах той же базы данных.

- `galleon-storage.json` — маркер привязки каталога к базе данных. Сервер создаёт его при первом запуске; копируйте его вместе с каталогом.

На чужом или непримонтированном каталоге сервер не стартует: сообщение об ошибке начинается с `storage identity:` и указывает на причину.

Запись об ассете хранится только в базе данных и указывает на файл в хранилище. На один и тот же файл может ссылаться сколько угодно ассетов из разных репозиториев, поэтому удаление ассета освобождает место только тогда, когда исчезает последняя ссылка на файл. Модель компонентов и ассетов описана в разделе Репозитории и форматы.

Внезапное отключение питания не оставляет наполовину записанных файлов, на которые уже ссылается база данных.

Сборка мусора

Сборка мусора не решает, что устарело: она забирает содержимое, на которое не осталось ссылок. Решение о том, какие компоненты удалить, принимают администратор вручную или политики очистки (Политики очистки).

Артефакт, удалённый из каталога артефактов, не удаляется с диска немедленно: файлы, на которые больше нет ссылок, находит и удаляет фоновая сборка мусора. Она обслуживает оба дерева хранилища одним циклом по одному расписанию. Останавливать загрузки на время цикла не нужно.

Параметры — блок `gc` конфигурации:

Ключ	По умолчанию	Назначение
`auto`	`true`	Плановые циклы; `false` отключает только расписание — ручной запуск работает всегда
`interval`	`5m`	Пауза между плановыми циклами
`grace_period`	`24h`	Минимальный возраст удалённого содержимого, после которого его место освобождается
`batch_size`	`500`	Кандидатов за один запрос сканирования
`max_duration`	`15m`	Потолок длительности одного цикла; остаток доберёт следующий

> Из-за grace-периода место освобождается не сразу: удалённое содержимое остаётся на диске не меньше `grace_period`. Это же значение действует и для образов контейнеров.

Наблюдение и ручной запуск

Страница «Настройки» → «Сборка мусора» показывает состояние сборки: идёт ли цикл прямо сейчас, когда был предыдущий («Последний запуск»), чем он закончился («Последний результат» — сводка вида «OK: N objects reclaimed») и когда ожидается следующий.

Кнопка «Запустить» запускает цикл немедленно. Запуск асинхронный: интерфейс подтверждает, что цикл начат, а сам он продолжается в фоне — закрытая страница его не прерывает. Одновременно выполняется только один цикл, повторный запуск отклоняется.

Просмотр страницы требует привилегии `nx-tasks-read`, запуск — `nx-tasks-run`.

Параметры сборки мусора со страницы не меняются — только в конфигурации сервера, с перезапуском.

Пауза на время обслуживания

Кнопка «Приостановить» на той же странице останавливает и плановые циклы, и ручной запуск. Пауза действует ограниченное время: по умолчанию сутки, максимум семь дней. По истечении срока она снимается автоматически, поэтому забытая пауза не остановит освобождение места

навсегда. Кнопка «Возобновить» снимает её досрочно. Состояние переживает перезапуск сервера; если серверов несколько, оно общее для всех.

Пауза нужна прежде всего для резервного копирования — подробнее в разделе Резервное копирование. На удаление артефактов пользователями она не влияет: удалить компонент или репозиторий можно и во время паузы, место освободится позже.

Если место закончилось

Переполнение диска проявляется ошибками при загрузке артефактов; в логе сервера у таких записей в поле `err` видно `no space left on device`. Чтение при этом продолжает работать.

Чтобы освободить место, выполните следующие действия:

1. Удалите ненужные артефакты в разделе «Обзор» веб-интерфейса — в первую очередь крупные и устаревшие сборки.
2. Запустите сборку мусора кнопкой «Запустить» на странице «Настройки» → «Сборка мусора».
3. Если освободилось недостаточно, уменьшите `grace_period` в конфигурации и перезапустите сервер: файлы моложе grace-периода сборка мусора не трогает.

> Не удаляйте файлы из каталога хранилища вручную. Имя файла — хеш его содержимого, поэтому по имени нельзя понять, на какой артефакт он ссылается; удалённый по ошибке файл не восстанавливается и не обнаруживается сборкой мусора, а обращения к соответствующему артефакту начнут завершаться ошибкой.

Политики очистки

Политики очистки задают срок хранения содержимого репозиториев: администратор описывает правилами, что и как долго держать, а Galleon применяет эти правила сам, без ручной уборки.

Типовые задачи, которые они закрывают: не держать snapshot-сборки дольше месяца, убирать версии, которые давно никто не скачивает, ограничивать размер кеша proxy-репозитория, оставлять только несколько последних релизов каждого пакета.

Политика — это набор критериев: возраст, давность скачивания, путь. Компонент удаляется, только если подходит под все критерии политики одновременно: политика с возрастом «больше 30 дней» и путём `.*-SNAPSHOT.*` удалит лишь то, что и старше тридцати дней, и лежит по подходящему пути.

Политика создаётся для одного формата и применяется только к репозиториям этого формата: политику формата Maven нельзя привязать к репозиторию npm. К одному репозиторию можно привязать несколько политик — компонент удаляется, если его отобрала любая из них. Порядок политик значения не имеет: результаты складываются.

Политика удаляет компоненты из каталога, а место на диске возвращает сборка мусора — она забирает содержимое, на которое не осталось ссылок, не раньше чем через `gc.grace_period` (Хранилище и сборка мусора).

Hosted и proxy

Политики работают с репозиториями обоих типов, но результат разный.

В hosted-репозитории удаление безвозвратно: артефакт публиковали в Galleon, и вернуть его можно только повторной публикацией. Поэтому для hosted-репозиториев выбирайте критерии осторожно и проверяйте их пробным запуском.

В proxy-репозитории удаляется только закешированная копия. Сам артефакт остаётся во внешнем реестре, и следующий запрос скачает его снова — для пользователей это незаметно, разве что первый запрос будет медленнее. Очистка работает здесь как вытеснение кеша: освобождает место, занятое тем, что перестали спрашивать.

> ⚠ Очистка не подчиняется политике записи репозитория. Репозиторий с политикой записи `DENY` закрыт от записи для клиентов, но привязанная политика очистки его вычистит.

Создание политики

Страница «Настройки» → «Политики очистки», кнопка «+ Новая политика».

Имя политики начинается с буквы, цифры или дефиса, содержит буквы, цифры, дефисы, подчёркивания и точки, не длиннее 255 символов. Имена сравниваются без учёта регистра, а после создания имя изменить нельзя.

Формат менять можно, пока политика не привязана ни к одному репозиторию. У привязанной политики смена формата отклоняется: сначала снимите привязки, потом меняйте формат.

> ℹ Политика без единого критерия не удаляет ничего — в списке она помечена как «критерии не заданы». Это же относится к политике, где задано только «Сохранять новейших версий»: отбирать нечего, и исключать тоже нечего.

Критерии

Компонент удаляется, только если подходит под все заданные критерии одновременно. Общие для всех форматов критерии:

«Возраст компонента (дней)» — отбирает компоненты, у которых последнее изменение содержимого по всем ассетам старше указанного числа дней. Значение — целое число от 1 до 24 855.

«Использование компонента (дней)» — отбирает компоненты, ни один ассет которых не скачивали указанное число дней. Ассет, который не скачивали ни разу, считается от последнего изменения содержимого. Диапазон значений тот же.

«Регулярное выражение пути asset» — отбирает компоненты, у которых хотя бы у одного ассета путь совпадает с выражением. Совпадение полное: выражение закреплено с обеих сторон и сверяется с полным путём, включая ведущий слэш. Синтаксис — RE2, длина до 1024 символов; конструкции, которых в RE2 нет, отклоняются при сохранении политики — например, `.*(?!-SNAPSHOT)\.jar`.

> ⚠ Выражение `antlr.*` не подойдёт ни под один путь, потому что сверяется с путём целиком. Пишите `.*antlr.*`.

Критерии, зависящие от формата

«Тип релиза» доступен для форматов Maven и npm и отбирает либо только релизы, либо только предрелизы. Предрелиз определяется по-разному: у Maven это версия с суффиксом `-SNAPSHOT`, у npm — версия с semver-суффиксом предрелиза.

«Сохранять новейших версий» оставляет указанное число новейших версий пакета: они не удаляются, даже если подходят под остальные критерии. Счёт ведётся по каждому пакету, а не по репозиторию целиком. Критерий доступен двум форматам:

- Maven — версии упорядочиваются по номеру версии, и критерий требует, чтобы был выбран тип релиза «Релизы»: исключение действует только для релизных версий;
- Docker — теги упорядочиваются по дате и учитываются только в области тегов.

Особенности Docker

Компонент формата Docker — это пара «образ и тег», поэтому у политики есть две области, и хотя бы одна должна быть включена:

- «Теги» — тегированные образы, по одному компоненту на тег. Включён по умолчанию. Только в этой области действует «Сохранять новейших версий»;
- «Manifest без тега» — отдельно стоящие манифесты, не входящие в живой индекс образа. По умолчанию выключен.

Очистка снимает теги и удаляет манифесты; сами файлы образа забирает потом сборка мусора.

Когда в одной политике включены обе области, они оцениваются по очереди: сначала манифесты без тега, затем теги. Поэтому манифест, у которого этот же проход снял последний тег, попадает под удаление только на следующем — ночном или запущенном вручную. Планируя первую большую уборку, рассчитывайте, что образ полностью исчезнет за два прохода: сначала теги, потом манифест.

Если теги и манифесты без тега разнесены по разным политикам, всё зависит от порядка привязки: политика манифестов, стоящая в списке после политики тегов, увидит освободившиеся манифесты уже в текущем проходе.

Возраст Docker-компонента считается по его тегу: создание тега и push, меняющий то, на что тег указывает, сбрасывают отсчёт, а повторный push того же содержимого — нет. Возраст манифеста без тега считается с момента, когда его загрузили в репозиторий.

Регулярное выражение отбирает Docker-компонент только по пути манифеста: `/v2/<IMAGE>/manifests/<TAG>` для тегированного компонента и путь с digest для манифеста без тега. Пути слоёв компонент не отбирают никогда.

Подписи и attestation

Подписи и attestation живут в репозитории рядом с образом, и очистка обращается с ними по-разному — в зависимости от того, как они к образу привязаны.

Подпись, привязанная к образу ссылкой `subject` из стандарта OCI 1.1, хранится ровно столько же, сколько сам образ: пока образ в репозитории, политика подпись не удалит, а вместе с образом удалит и её. Так подписи публикуют `oras attach` и `cosign`, запущенный с параметром `--registry-referrers-mode=oci-1-1`. Настраивать для таких подписей ничего не нужно — достаточно включить область манифестов без тега.

> Подпись, опубликованная отдельным тегом вида `sha256-<DIGEST>.sig` (а также `.att` и `.sbom`), с образом ничем не связана: для Galleon это отдельный образ со своим тегом, и политика оценивает его сама по себе. Такие теги создаёт `cosign` в режиме `legacy`: этот режим появился раньше стандарта OCI 1.1 и остаётся включённым, пока клиенту не задан режим `oci-1-1`. Подпись скачивают редко, только когда её проверяют, поэтому по давности скачивания она попадает под удаление, даже если сам образ скачивают каждый день. В hosted-репозитории подпись после этого не восстановить: политика её удалит, образ останется, и проверка подписи начнёт падать там, где раньше проходила. Вернуть подпись можно только повторной подписью образа. В proxy-репозитории последствий нет: удаляется закешированная копия, и при следующей проверке подпись будет скачана из внешнего реестра заново.
> Если такие подписи есть в hosted-репозитории, составьте регулярное выражение так, чтобы под политику попадали только нужные образы, задавайте сроки с запасом и перед привязкой обязательно выполните пробный запуск.

Привязка к репозиториям

Политика начинает действовать, когда её привязали к репозиторию: в форме репозитория (Репозитории) есть поле «Политики очистки», в котором выбираются политики того же формата.

Удаление политики снимает привязку у всех репозиториев, которые её использовали.

Запуск

Способы запуска очистки:

1. **Планово** — ежедневно в 01:00 UTC по всем репозиториям с привязанными политиками. Репозиторий, выведенный из обслуживания («в сети» снят), плановый проход пропускает.
2. **Вручную по всем репозиториям** — кнопка «Запустить сейчас» на карточке задачи на странице политик очистки.
3. **Вручную по одному репозиторию** — кнопка «Запустить очистку» в форме репозитория. Ручной запуск работает и для выведенного из обслуживания репозитория.

Одновременно для репозитория выполняется только один запуск; повторный отвечает, что очистка уже выполняется. Итог планового прохода виден на карточке задачи в поле «Последний запуск».

Пробный запуск

Кнопка «Пробный запуск» считает компоненты, которые политика удалила бы из выбранного репозитория, и ничего не удаляет. Политика при этом может быть и не привязана к репозиторию — это способ проверить критерии до того, как они начнут действовать.

Пробный запуск считает по каждой политике отдельно, поэтому компонент, который отобрали две привязанные политики, будет посчитан дважды; при настоящем запуске он удаляется один раз.

У образов есть ещё одно расхождение: если проход удаляет индекс без тега, входившие в него манифесты становятся кандидатами прямо в этом проходе, и настоящий запуск может удалить больше, чем показал пробный.

Права

Политики относятся к области `settings`: просмотр списка требует привилегии `nx-settings-read`, а создание, правка и удаление — привилегии `nx-settings-update` (Привилегии).

Запуск очистки — привилегия `nx-tasks-run`, состояние задачи — `nx-tasks-read`. Кнопка запуска работает и без права на чтение состояния задачи: в этом случае не показывается только карточка с итогом прошлого прохода.

Резервное копирование

Резервная копия Galleon состоит из трёх частей — все три обязательны:

- **База данных PostgreSQL** — каталог артефактов всех форматов (включая метаданные образов контейнеров), пользователи и права, настройки.
- **Каталог хранилища** (`storage.default.local.root` целиком: `blobs/`, `docker/` и маркер `galleon-storage.json`) — содержимое артефактов.
- **Файл ключей шифрования** (`security.secret_key_file`) — без него дамп базы не восстановим: не расшифруются учётные данные внешних реестров и служебные секреты (Секреты и ключ шифрования).

Резервная копия одной только базы данных бесполезна: без каталога хранилища не будет самих артефактов, без файла ключей — доступа к зашифрованным настройкам.

Порядок снятия копии

Правило согласованности: **база данных никогда не должна опережать хранилище**.

Лишние файлы в хранилище без записей в базе только занимают место; записи в базе без файлов в хранилище означают потерянные артефакты: каждое скачивание такого артефакта завершается ошибкой.

Поэтому копия снимается в порядке: **сначала база данных, потом каталог хранилища**. Всё, что опубликуют между двумя моментами, попадёт в копию лишними файлами.

Идущая параллельно сборка мусора может нарушить это правило: если между дампом базы и копией хранилища кто-то удалит артефакт, сборка мусора успеет удалить и файл, на который ссылается уже снятый дамп.

Удалённое содержимое остаётся на диске не меньше `gc.grace_period` (по умолчанию сутки), поэтому копирование, которое укладывается в этот срок, согласовано само по себе.

Если окно копирования длиннее grace-периода или он уменьшен, приостановите сборку мусора на время работ кнопкой «Приостановить» на странице «Настройки» → «Сборка мусора» (Хранилище и сборка мусора). То же самое можно сделать запросом к API, если копирование запускается скриптом. Пауза останавливает и плановые циклы, и ручной запуск. Она действует ограниченное время, по умолчанию сутки, и снимается автоматически, поэтому забытая пауза не остановит освобождение места навсегда. Кнопка «Возобновить» снимает её сразу, как только копирование закончено.

Обе копии можно снять и атомарными снимками на один момент времени (LVM, ZFS, облачные снимки), приостановив запись.

Восстановление

Порядок восстановления:

1. Восстановите **файл ключей** — тот же, что действовал на момент копии.
2. Восстановите **базу данных**.
3. Восстановите **каталог хранилища** — копией, снятой одновременно с дампом базы данных или позже. Копия хранилища, снятая раньше дампа, оставит записи, для которых нет файлов, и такие артефакты перестанут скачиваться.
4. Перезапустите все серверы инстанса.

Каталог хранилища восстанавливайте вместе с маркером `galleon-storage.json`: без него сервер не стартует.

Восстанавливайте базу целиком: выборочное восстановление отдельных таблиц рассинхронизирует служебные секреты, каталог артефактов и метаданные образов. Лишние файлы хранилища, попавшие в копию, останутся на диске: сборка мусора их не удаляет. Удалять их вручную нельзя (Хранилище и сборка мусора).

TLS

Galleon отдаёт HTTPS сам, если в конфигурации указаны сертификат и ключ. Обратный прокси для этого не нужен.

Включение

TLS включает блок `server.tls` с двумя путями:

```
server:
  tls:
    cert_file: /etc/galleon/tls/fullchain.pem
    key_file: /etc/galleon/tls/privkey.pem
```

TLS включается, когда заданы оба пути. Те же значения принимают переменные окружения `GALLEON_SERVER_TLS_CERT_FILE` и `GALLEON_SERVER_TLS_KEY_FILE`.

Требования к файлам:

- сертификат в формате PEM. Внутри файла сначала идёт сертификат сервера, затем промежуточные сертификаты удостоверяющего центра; ACME-клиенты и cert-manager называют такой файл `fullchain`;
- ключ в формате PEM. Ключи с парольной фразой и ключи в формате OpenSSH не принимаются;
- алгоритм ключа — RSA, ECDSA или Ed25519.

После включения Galleon сообщает клиентам адреса по схеме `https`. Если инстанс доступен под другим именем, задайте его в `server.public_base_url`.

Ошибки в настройках

Сервер не стартует, если:

- задан только один из двух путей;
- файл не читается или содержит не тот формат;
- ключ защищён парольной фразой;
- внутри `server.tls` встретился неизвестный ключ;
- блок `tls` оказался вне `server`.

Ротация

Сертификат меняется перезапуском:

1. Положите новые файлы по тем же путям.
2. Перезапустите Galleon; если серверов несколько, перезапустите каждый.
3. Убедитесь, что поднялся нужный сертификат: при старте в лог пишется запись `tls: server certificate loaded` с полем `sha256`.

Параметры протокола

Принимаются соединения по TLS 1.2 и новее. Наборы шифров:

- `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`;
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`;
- `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`;
- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`;
- `TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256`;
- `TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256`;
- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256`;
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`.

В TLS 1.3 наборы шифров определяет сам протокол.

Ответы по TLS получают заголовок `Strict-Transport-Security` со сроком в один год и с `includeSubDomains`. Если позже вернуть инстанс на HTTP, браузеры, уже заходившие по HTTPS, откроют веб-интерфейс только после сброса состояния HSTS.

Диагностика

Неудачные рукопожатия TLS пишутся в лог на уровне `warn` с источником `server.http`: клиент не согласовал версию протокола или шифр. Формат записей описан в разделе Логирование.

Обратный прокси

Galleon может отдавать HTTPS сам (TLS), поэтому обратный прокси перед ним не обязателен. Если прокси всё же стоит перед сервером, настройте его по требованиям ниже: с настройками по умолчанию часть запросов к репозиториям ломается.

Требования

Любой обратный прокси должен:

- проксировать запросы на порт Galleon; терминировать TLS на прокси или передавать трафик дальше по HTTPS, если TLS настроен и в Galleon;

- **не декодировать процентные последовательности в путях**: раскрытие `%2F` в `/` меняет путь запроса, и артефакт по нему не находится;
- пропускать большие тела запросов: разрешённый прокси размер согласуйте с `server.max_upload_size` в конфигурации Galleon;
- задать тайм-ауты чтения и записи с запасом под медленные загрузки больших артефактов;
- передавать заголовки `Host`, `X-Real-IP`, `X-Forwarded-For`, `X-Forwarded-Proto`.

Доверие заголовкам

По умолчанию Galleon не доверяет заголовкам `X-Forwarded-*`: прямой клиент может указать в них произвольный адрес, и тот попадёт в лог как `client_ip`. Доверие включают два независимых условия:

```
server:
  # Условие 1: адреса прокси в нотации CIDR.
  trusted_proxies: ["172.18.0.0/16"]
  # Условие 2: общий секрет в заголовке.
  proxy_auth:
    header: X-Origin-Token
    secret_env: GALLEON_ORIGIN_TOKEN
```

Как это работает:

- задан только `trusted_proxies` — проверяется адрес, с которого пришёл запрос;
- задан только `proxy_auth` — проверяется секрет в заголовке (по умолчанию `X-Origin-Token`), адрес не важен;
- заданы оба — требуются оба одновременно;
- не задано ничего — forwarded-заголовки игнорируются у всех запросов.

Запрос, не прошедший проверку, обрабатывается как запрос прямого клиента: forwarded-заголовки в нём игнорируются.

Абсолютные адреса, которые Galleon сообщает клиентам (поле `url` в ответах API, адреса пакетов в метаданных npm), собираются из заголовка `Host` и схемы, по которой работает сам Galleon. Когда TLS терминирует прокси, такие адреса уходят клиентам со схемой `http`. Чтобы хост и схему брали из `X-Forwarded-Host` и `X-Forwarded-Proto`, включите `server.trust_forwarded_headers`. Заголовки учитываются только у запросов, прошедших проверку выше, поэтому без условий доверия этот параметр ничего не меняет.

Параметр `server.public_base_url` задаёт адрес жёстко, в обход заголовков:

```
server:
  public_base_url: https://galleon.example.com
```

Он нужен, когда клиенты должны получать один и тот же адрес независимо от того, по какому имени пришёл запрос. Например, инстанс доступен ещё и по внутреннему имени: клиент, пришедший по нему, получит в метаданных npm внутренние адреса пакетов, и они попадут в `package-lock.json`.

Пример nginx

Конфигурация ниже терминирует TLS, проксирует запросы на Galleon и передаёт заголовки, по которым сервер определяет исходного клиента:

```
upstream galleon {
  server 127.0.0.1:8080;
  keepalive 32;
}

server {
  listen 443 ssl;
  server_name galleon.example.com;

  # Лимит загрузок задаёт прокси: без этой строки nginx режет тела на 1 МБ.
  client_max_body_size 0;

  # Тайм-ауты под большие артефакты.
  proxy_read_timeout 1h;
  proxy_send_timeout 1h;

  location / {
    proxy_http_version 1.1;
    proxy_set_header Connection "";
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Origin-Token "<GALLEON_ORIGIN_TOKEN>";
    # proxy_pass без URI-части: с ней nginx декодирует %2F в путях.
    proxy_pass http://galleon;
  }
}
```

⚠ Директива `proxy_set_header` не наследуется с предыдущего уровня конфигурации, если на текущем объявлена хотя бы одна такая директива. При добавлении отдельных location повторяйте в каждом весь набор заголовков, включая секрет.

Входящий заголовок `X-Request-ID` Galleon игнорирует и всегда возвращает собственный идентификатор запроса. Чтобы сопоставлять записи прокси с записями Galleon, логируйте его в nginx через переменную `$upstream_http_x_request_id`.

Мониторинг

Для проверок живости Galleon предоставляет два эндпоинта без аутентификации:

Эндпоинт	Назначение	Ответы
`GET /healthz`	Процесс жив и отвечает	`200`
`GET /readyz`	Доступна база данных	`200`; `503` при недоступной базе

`healthz` подходит для liveness-проб, `readyz` — для readiness-проб и проверки готовности после развёртывания.

Ограничение доступа

Доступ к health-эндпоинтам ограничен списком адресов `server.monitoring_allowlist`; по умолчанию — только loopback, поэтому локальные проверки работают без настройки, а запросы из сети отклоняются.

Правила:

- проверяется адрес TCP-соединения; заголовки `X-Forwarded-*` не учитываются — запрос через обратный прокси оценивается по адресу самого прокси;
- отказ неотличим от несуществующего пути (`404`); причина видна только в логе (`deny_reason=monitoring_ip_denied`);
- явно заданный пустой список запрещает доступ всем, включая локальные проверки;
- значение `0.0.0.0/0` открывает эндпоинты всем.

```
server:
  monitoring_allowlist: ["127.0.0.0/8", "::1/128", "10.0.0.0/8"]
```

Если проверки приходят не с loopback, в списке нужны их адреса: для внешнего мониторинга через обратный прокси — адрес самого прокси. Точный адрес источника виден в access-логе отклонённого запроса — поле `client_ip` в записи с `deny_reason=monitoring_ip_denied`.

Наблюдение за инстансом

Наблюдение строится на логах и средствах PostgreSQL:

- **Лог сервера** — три поля, отсутствующие на здоровом трафике, дают готовые мониторы: `sqlstate` (ошибки базы), `deny_reason` (отказы доступа), `served_stale` (деградация внешних реестров). Подробнее — в разделе Логирование.
- **Состояние соединений с внешними реестрами** — колонка «Источник» на странице «Настройки» → «Репозитории» (Репозитории).
- **PostgreSQL** — стандартные средства: активность по именам приложений `galleon` и `galleon-oci-meta` (соединения, обслуживающие образы контейнеров, идут без метки), размер базы, медленные запросы.
- **Диск** — заполнение каталога хранилища.

Логирование

Все подсистемы Galleon пишут один поток структурированных логов в stdout — формат JSON, одна запись на строку. Ротация и доставка в систему хранения логов — забота платформы развёртывания.

Настройка

Блок `logging` конфигурации:

Ключ	По умолчанию	Назначение
`level`	`info`	Порог логов Galleon: `debug`, `info`, `warn`, `error`
`format`	`json`	`json` или `text`
`registry_level`	`warn`	Отдельный порог для записей о работе с образами контейнеров

События аудита при работе с образами контейнеров (загрузка и удаление манифестов, блобов, тегов) пишутся всегда, независимо от `registry_level`.

Access-лог

Каждый запрос пишет одну запись `msg="http request"` с полями: `source` (подсистема), `method`, `path`, `status`, `size`, `user`, `client_ip`, `duration`, время и число запросов к базе и завершающим `request_id`. В `user` стоит аутентифицированная учётная запись; у запроса без успешной аутентификации там `-`.

Тот же идентификатор Galleon возвращает клиенту в заголовке `X-Request-ID`. По нему в логе находятся все записи одного запроса. `client_ip` за обратным прокси корректен только при настроенном доверии (Обратный прокси).

Ошибки самого HTTP-сервера пишутся с источником `server.http` на уровне `warn`. Когда Galleon отдаёт HTTPS сам, в этот источник попадают неудачные TLS-рукопожатия (TLS).

Поля, которые появляются только на проблемном трафике:

- `sqlstate` — запрос столкнулся с ошибкой PostgreSQL;
- `served_stale=true` — proxy-репозиторий отдал устаревшую копию, потому что внешний реестр не ответил;
- `deny_reason` — отказ аутентификации или доступа; словарь значений ниже.

Поле `attempted_user` — логин, предъявленный клиентом: учётные данные отвергнуты или не проверялись, потому что запрос отклонён раньше. Значение `?` — авторизация не Basic или пустой логин. Неудачные входы ищите по этому полю: в `user` у таких записей `-`. Поле появляется и на обычном трафике, например когда мониторинг опрашивает открытый эндпоинт с учётными данными.

Причины отказов

Поле `deny_reason` объясняет каждый отказ, не меняя ответ клиенту (коды ответов не раскрывают существование чужих репозиториев):

Значение	Причина
`no_credentials`	Запрос без учётных данных там, где они обязательны
`bad_credentials`	Учётные данные предъявлены и отвергнуты
`anonymous_disabled`	Анонимный доступ выключен
`anonymous_forbidden`	Роли `anonymous` не хватает привилегии
`forbidden`	Пользователю не хватает привилегии на репозиторий
`admin_required`	Не хватает административной привилегии
`anon_no_admin`	Анонимный запрос к образам контейнеров, требующий административных прав
`password_change_required`	Заблокировано до принудительной смены пароля
`repository_not_found`	Целевой репозиторий не найден; ставится и на ответ `404` до проверки учётных данных
`repository_name_invalid`	Имя репозитория не разобрано, поиск не выполнялся
`wrong_format`	Репозиторий существует, но другого формата
`monitoring_ip_denied`	Адрес вне `monitoring_allowlist`
`upstream_manually_blocked`	Обращения к внешнему реестру заморожены вручную
`upstream_auto_blocked`	Обращения приостановлены автоблокировкой

Запись с `401` / `403` без `deny_reason` — признак инфраструктурного сбоя: смотрите поля `err` и `sqlstate`. Ответ `404` по репозиторию тоже несёт причину; без неё остаются ответы репозитория, выведенного из обслуживания, и отказы из-за неверной формы запроса.

События аудита

Административные и доменные события пишутся на уровне `info`:

- публикации артефактов — записи `msg="<FORMAT> artifact published"` с координатами и репозиторием, по одной на артефакт;

- операции с образами контейнеров — загрузка и удаление манифестов, блобов и тегов;
- инвалидация кеша репозитория (`source=audit.repository`);
- создание bootstrap-администратора, создание стартового набора репозиториев, итоги циклов сборки мусора, смены состояния соединений с внешними реестрами.

Секреты в лог не попадают ни на каком уровне: значения заголовков авторизации, пароли, токены и содержимое тел не логируются. Управляющие символы и некорректный UTF-8 в значениях, пришедших от клиента (путь, заголовки, имена файлов), заменяются в логе на `?` .

API

У Galleon есть встроенный REST API: все операции веб-интерфейса доступны и программно. Через него заводят репозитории и учётные записи при развёртывании, выгружают состояние инстанса в систему инвентаризации и подключают инструменты управления конфигурацией.

Справочник API встроен в сервер: Swagger UI открывается по адресу `https://<GALLEON_HOST>/docs` , машиночитаемая спецификация OpenAPI доступна по адресу `https://<GALLEON_HOST>/openapi.yaml` . Оба адреса требуют аутентификации — подойдёт любая учётная запись (браузер запросит логин и пароль).

Аутентификация

Программные запросы аутентифицируются так же, как клиенты форматов: логином и паролем либо токеном пользователя (Учётные данные). Для автоматизации заведите отдельную учётную запись и работайте от её токена — его отзывают, не трогая пароль.

Пример: список репозиториев инстанса.

```
curl -u <USERNAME>:<PASSWORD> https://<GALLEON_HOST>/service/rest/v1/repositories
```

Здесь `<USERNAME>` и `<PASSWORD>` — логин и пароль учётной записи либо «Name code» и «Pass code» токена.

Права

Операции API проверяются теми же привилегиями, что и кнопки в веб-интерфейсе: чтобы создать репозиторий программно, учётной записи нужна привилегия на создание репозиториев, а чтобы выпустить пользователя — привилегия области `users` (Привилегии). Ответ `403` означает нехватку привилегии, а не ошибку запроса.

Руководство пользователя

Раздел для пользователя Galleon: работа в веб-интерфейсе, учётные данные и подключение штатных клиентов по каждому формату.

Обзор

Пользователь Galleon работает с артефактами через веб-интерфейс и штатные клиенты форматов. В веб-интерфейсе он просматривает содержимое репозиториев, загружает файлы и управляет своим профилем. Штатные клиенты форматов (`docker`, `mvn`, `npm` и другие) работают с репозиториями напрямую, им указывается адрес Galleon.

С чего начать

Чтобы начать работу, выполните следующие действия:

1. Получите учётную запись у администратора. Если для нужных репозиториев включён анонимный доступ, читать их можно и без учётной записи.
2. Войдите в веб-интерфейс и выпустите токен пользователя — он используется вместо логина и пароля в клиентах и CI (Учётные данные).
3. Настройте клиент своего формата на адрес репозитория — инструкции по каждому формату собраны в разделе Форматы.

Что доступно пользователю

Возможности пользователя в Galleon:

- **Просмотр** — дерево компонентов и ассетов любого доступного репозитория и скачивание файлов: раздел «Обзор» веб-интерфейса.
- **Публикация** — штатными клиентами форматов или через раздел «Загрузка» веб-интерфейса.
- **Профиль** — смена пароля, токен пользователя, управление сессиями.

Набор видимых репозиториев и допустимых действий определяется ролями, которые назначает администратор. Как устроена модель доступа, описано в разделе Управление доступом.

Веб-интерфейс

Веб-интерфейс Galleon доступен по адресу инстанса (`https://<GALLEON_HOST>`). В нём пользователь просматривает содержимое репозиториев, скачивает и загружает артефакты и управляет своим профилем. Интерфейс доступен на русском и английском языках. Язык переключается списком «Язык интерфейса» в шапке, а на странице входа — на карточке входа. Выбор запоминается в браузере: он не привязан к учётной записи и не переносится на другой

браузер или устройство. При первом заходе язык определяется по настройкам браузера. Сообщения сервера, например текст ошибки при отказе, остаются английскими.

Вход

Чтобы войти, введите на странице входа «Имя пользователя» и «Пароль» и нажмите «Войти». После входа открывается раздел «Обзор».

Без входа доступен только просмотр — и только если администратор включил анонимный доступ. Ссылки «Загрузить» и «Настройки» появляются в шапке после входа.

Просмотр содержимого

Раздел «Обзор» показывает карточки доступных репозиториев (имя, формат, тип). Содержимое репозитория отображается деревом (модель компонентов и ассетов описана в разделе Репозитории и форматы):

- **папки** (▪) — группируют содержимое: группы Maven, пространства имён пакетов, образы;
- **компоненты** (◆) — версии пакетов и теги образов;
- **индексные файлы** (▤) — служебные файлы, которые Galleon генерирует сам (например, `maven-metadata.xml`).

В репозиториях Docker (OCI) содержимое образа разложено по группам: `tags/` — теги, `manifests/` — манифесты, на которые эти теги ведут (несколько тегов одного манифеста показываются одной строкой). Слои и конфигурации собраны в отдельную ветку `blobs/` в корне репозитория: они принадлежат репозиторию, а не образу, и один и тот же слой могут использовать несколько образов.

При выборе узла справа открывается панель деталей:

- у компонента — координаты, метаданные формата (описание и зависимости пакета, платформы и слои образа) и список ассетов;
- у ассета — размер, тип содержимого, контрольные суммы (SHA-256, SHA-1, MD5, SHA-512), даты, а для proxy-репозиториев — адрес источника и срок действия кеша.

Среди дат есть «Последнее скачивание» — когда файл скачивали в последний раз. Значение «Никогда» означает, что его не скачивали ни разу, а «Не отслеживается» — что скачивания не учитываются: так помечены слои образов контейнеров, потому что они общие для всех образов репозитория.

Кнопка «Скачать» скачивает ассет. Кнопка «Показать ещё» подгружает следующую страницу длинных списков.

Кнопки «Удалить компонент» и «Удалить» у отдельных ассетов требуют привилегии `delete`. При политике записи `DENY` удаление отклоняется независимо от привилегий: такой репозиторий доступен только на чтение. Удаление необратимо; при удалении последнего ассета компонент

удаляется автоматически. Для образов контейнеров действуют свои правила удаления — они описаны на странице Docker (OCI).

Загрузка

Раздел «Загрузка» загружает файлы в hosted-репозитории форматов Maven, npm, RubyGems, PyPI и raw. Форма подстраивается под формат выбранного репозитория:

- **Maven** — координаты (`groupId` , `artifactId` , `version`), файлы с полями `classifier` и `extension` , флаг `generate-pom` ;
- **npm** и **RubyGems** — только файл пакета (`.tgz` или `.gem`): имя и версия читаются из его метаданных;
- **PyPI** — файлы дистрибутивов (wheel или sdist); имя и версию можно не указывать — они читаются из метаданных;
- **raw** — целевой каталог (`directory`), имя файла (`filename`) и сам файл.

Для загрузки нужна привилегия `add` на репозиторий. Репозитории форматов Go, Docker (OCI) и Cargo принимают публикации только штатными клиентами (Форматы).

Профиль

Страница «Профиль» открывается из меню аватара и содержит секции:

- **Роли** — назначенные вам роли;
- **Учётная запись** — данные учётной записи; для локальных пользователей редактируются кнопкой «Изменить»;
- **Пароль** — смена пароля кнопкой «Смена пароля» (требуется текущий пароль);
- **Токен пользователя** — токен для клиентов и CI (Учётные данные);
- **Сессии** — активные сессии: текущую можно завершить («Выйти»), любую из остальных — отозвать («Отозвать»), а все, кроме текущей, — разом («Завершить другие сессии»).

Раздел «Настройки» в шапке предназначен для администрирования — он описан в руководстве администратора.

Учётные данные

Для аутентификации в Galleon используются пароль и токен пользователя; как устроена модель доступа, описано в разделе Управление доступом.

Пароль

Пароль используется для входа в веб-интерфейс. Сменить его можно в профиле: меню аватара → «Профиль» → «Смена пароля» (требуется текущий пароль). Если администратор потребовал смену пароля, до её завершения остальные действия в веб-интерфейсе заблокированы.

Токен пользователя

В клиентах командной строки и CI вместо пароля используйте токен: он подставляется как обычная пара «логин и пароль», а при компрометации отзывается без смены пароля учётной записи.

Чтобы создать токен, выполните следующие действия:

1. Откройте «Профиль», секцию «Токен пользователя».
2. Нажмите «Выпустить».
3. Сохраните показанные значения: «Name code» используйте как логин, «Pass code» — как пароль.

> ⚠ Pass code показывается только один раз — сразу после создания. Если значение утеряно, перевыпустите токен кнопкой «Перевыпустить».

У каждого пользователя может быть один действующий токен:

- «Перевыпустить» выпускает новый токен вместо старого — старые учётные данные перестают работать сразу;
- «Удалить» отзывает токен без выпуска нового.

Использование в клиентах

Токен (или пароль) подставляется везде, где клиент формата ожидает логин и пароль: в `docker login`, `settings.xml` Maven, `.npmrc`, `~/.pypirc`, `~/.netrc` и других. Готовые примеры для каждого клиента — в разделе Форматы.

RubyGems (файл `~/.gem/credentials`) и Cargo (команда `cargo login`) принимают учётные данные готовой строкой заголовка авторизации `Basic <BASE64>`, где `<BASE64>` — строка `<USERNAME>:<PASSWORD>` в кодировке base64. Подробности — на страницах RubyGems и Cargo.

Если клиент получает ошибку

Ответ `401` означает, что учётные данные не приняты: чаще всего токен перевыпущен кнопкой «Перевыпустить» или удалён кнопкой «Удалить», и старые значения перестали работать. Ответ `403` означает нехватку привилегии на операцию — за правами обратитесь к администратору. Ответ `404` приходит и при отсутствии артефакта, и при отсутствии прав доступа (Управление доступом), а у proxy-репозиториев — ещё и из кеша отказов (Проксирование и кеширование).

Форматы

В этом разделе собраны инструкции по каждому формату: подключение клиента к proxy-репозиторию, публикация в hosted-репозиторий и особенности формата.

В примерах команд подставьте свои значения:

- `<GALLEON_HOST>` — адрес вашего инстанса Galleon;
- `<USERNAME>` и `<PASSWORD>` — логин и пароль учётной записи либо «Name code» и «Pass code» токена пользователя (Учётные данные);
- `<REPOSITORY>` — имя репозитория. Имена в примерах условные, фактические смотрите в разделе «Обзор» или уточняйте у администратора.

Общие правила для всех форматов:

- адрес репозитория — `https://<GALLEON_HOST>/repository/<REPOSITORY>/`; исключение — Docker (OCI), о нём написано в разделе Репозитории и форматы;
- если администратор включил анонимный доступ к репозиторию, учётные данные в командах чтения можно опустить (`<USERNAME>:<PASSWORD>@` в адресе, `docker login` и подобные);
- почему только что вышедший во внешнем реестре пакет может быть виден не сразу, объясняет раздел Проксирование и кеширование.

Docker (OCI)

Репозитории формата Docker (OCI) хранят образы контейнеров и OCI-артефакты, например Helm-чарты. Hosted-репозиторий служит приватным реестром, proxy-репозиторий кеширует внешний реестр. Репозитории `docker-hosted` и `docker-hub-proxy` (кеш Docker Hub) входят в стартовый набор.

Docker (OCI) — единственный формат, который адресуется без префикса `/repository/`:

```
&lt;GALLEON_HOST&gt;/&lt;REPOSITORY&gt;/&lt;IMAGE&gt;:&lt;TAG&gt;
```

Вход в реестр

Выполните вход с логином и паролем или токеном пользователя:

```
docker login &lt;GALLEON_HOST&gt;
```

Совместимые инструменты входят аналогично: `crane auth login`, `skopeo login`, `helm registry login`, `oras login`.

Скачивание

Через proxy-репозиторий образы внешнего реестра скачиваются по общему шаблону — путь образа указывается после имени репозитория (для официальных образов Docker Hub нужен префикс `library/`):

```
docker pull &lt;GALLEON_HOST&gt;/docker-hub-proxy/library/alpine:3.20
```

Скачивание из hosted-репозитория:

```
docker pull <GALLEON_HOST>/docker-hosted/alpine:1.0
helm pull oci://<GALLEON_HOST>/docker-hosted/charts/mychart --version 0.1.0
```

Публикация

Чтобы опубликовать образ, пометьте его адресом в репозитории и отправьте:

```
docker tag alpine:3.20 <GALLEON_HOST>/docker-hosted/alpine:1.0
docker push <GALLEON_HOST>/docker-hosted/alpine:1.0
```

Helm-чарты публикуются как OCI-артефакты — версия чарта становится тегом:

```
helm push mychart-0.1.0.tgz oci://<GALLEON_HOST>/docker-hosted/charts
```

Для публикации нужны обе привилегии на репозиторий — `read` и `add`: клиент при отправке образа запрашивает у реестра сразу и чтение, и запись, поэтому одной привилегии `add` не хватит.

Удаление

Образы удаляются в разделе «Обзор» веб-интерфейса (нужна привилегия `delete`):

- удаление тега снимает только его — сам образ остаётся доступен по digest;
- удаление манифеста по digest не выполняется, пока на него ссылается тег или многоплатформенный индекс: сначала удалите ссылающиеся теги или индекс;
- слои и конфигурации (ветка `blobs/`) по отдельности не удаляются: содержимое общее для всех образов репозитория;
- при политике записи `DENY` удаление отклоняется даже с привилегией `delete` — политику меняет администратор.

Место на диске возвращает сборка мусора, поэтому оно освобождается не сразу после удаления (Хранилище и сборка мусора).

Особенности

Особенности формата:

- Повторный `push` существующего тега перезаписывает его — теги изменяемы; неизменяемая ссылка на образ — его digest. Перезапись подчиняется политике записи репозитория: при `ALLOW_ONCE` перевод существующего тега на другой образ отклоняется, при `DENY` отклоняется любая публикация.
- Поддерживаются многоплатформенные образы и OCI-артефакты: Helm-чарты, артефакты `oras`.
- Образы, скачанные через proxy-репозиторий, кешируются целиком — манифест и все слои; общее поведение кеша описано в разделе Проксирование и кеширование.

- Список допустимых медиатипов OCI-артефактов фиксирован: публикация артефакта с другим медиатипом отклоняется.
- Публикация через веб-интерфейс недоступна — используйте перечисленные клиенты.

Maven

В стартовый набор входят proxy-репозиторий `maven-central` (зеркало Maven Central) и hosted-репозитории `maven-releases` (релизы) и `maven-snapshots` (snapshot-версии).

Подключение к proxy

Направьте Maven на зеркало в `~/.m2/settings.xml`:

```
<settings>
  <mirrors>
    <mirror>
      <id>galleon-central</id>
      <url>https://<GALLEON_HOST>/repository/maven-central</url>
      <mirrorOf>central</mirrorOf>
    </mirror>
  </mirrors>
</settings>
```

Если анонимный доступ к репозиторию выключен, добавьте блок `<server>` с тем же `id`, что у зеркала, и вашими учётными данными — по образцу из раздела Публикация.

Публикация

Укажите репозиторий назначения в `pom.xml` и учётные данные в `settings.xml` — значения `id` в обоих блоках должны совпадать:

```
<!-- pom.xml -->
<distributionManagement>
  <repository>
    <id>galleon-releases</id>
    <url>https://<GALLEON_HOST>/repository/maven-releases</url>
  </repository>
</distributionManagement>
```

```
<!-- ~/.m2/settings.xml -->
<settings>
  <servers>
    <server>
      <id>galleon-releases</id>
      <username><USERNAME></username>
      <password><PASSWORD></password>
    </server>
  </servers>
</settings>
```

```
mvn deploy
```

Публикация из Gradle:

```
publishing {
  repositories {
    maven {
      url = uri("https://<GALLEON_HOST>/repository/maven-releases")
      credentials {
        username = "<USERNAME>"
        password = "<PASSWORD>"
      }
    }
  }
}
```

Использование опубликованных артефактов

Зеркало покрывает только Maven Central, поэтому hosted-репозитории подключаются к сборке отдельными записями — в `pom.xml` или в профиле `settings.xml`:

```
<repositories>
  <repository>
    <id>galleon-releases</id>
    <url>https://<GALLEON_HOST>/repository/maven-releases</url>
  </repository>
  <repository>
    <id>galleon-snapshots</id>
    <url>https://<GALLEON_HOST>/repository/maven-snapshots</url>
    <snapshots>
      <enabled>true</enabled>
    </snapshots>
  </repository>
</repositories>
```

Для Gradle:

```
repositories {
  maven { url = uri("https://<GALLEON_HOST>/repository/maven-releases") }
  maven { url = uri("https://<GALLEON_HOST>/repository/maven-snapshots") }
}
```

Учётные данные — те же блоки `<server>` (Maven) или `credentials` (Gradle), что и при публикации.

Релизы и snapshot-версии

Повторная публикация существующей релизной версии отклоняется с кодом `409`: релизы неизменяемы.

Каждая загрузка snapshot-версии получает метку времени (`YYYYMMDD.HHMMSS-N`) и попадает в `maven-metadata.xml`; запрос версии `X-SNAPSHOT` отдаёт последнюю сборку.

Особенности

Особенности формата:

- `maven-metadata.xml` генерируется сервером автоматически — публиковать его не нужно.
- Hosted-репозиторий отдаёт только те checksum-файлы, которые загрузил клиент: Maven по умолчанию публикует `.sha1` и `.md5`. Чтобы публиковать все четыре, задайте при деплое `-Daether.checksums.algorithms=SHA-1,MD5,SHA-256,SHA-512`.
- Классификаторы (`sources`, `javadoc`) работают штатно.

Go

В стартовый набор входят proxy-репозиторий `go-proxy` (зеркало proxy.golang.org), hosted-репозиторий `go-hosted` (приватные модули) и `go-sumdb` — proxy-репозиторий базы контрольных сумм sum.golang.org.

Подключение к proxy

Задайте переменные окружения (или запишите их командой `go env -w`):

```
export GOPROXY="https://<GALLEON_HOST>/repository/go-proxy"
export GOSUMDB="sum.golang.org https://<GALLEON_HOST>/repository/go-sumdb"
go get rsc.io/quote
```

`GOSUMDB` записывается двумя значениями через пробел — имя базы и адрес proxy-репозитория: клиент `go` будет проверять контрольные суммы через Galleon, не обращаясь в интернет напрямую.

Приватные модули

Приватные модули публикуются в `go-hosted`, публичные продолжают идти через `go-proxy` — для этого источники перечисляются в `GOPROXY` списком (при ответе `404` клиент переходит к следующему адресу):

```
export GOPROXY="https://<GALLEON_HOST>/repository/go-hosted,https://<GALLEON_HOST>/
repository/go-proxy"
export GOPRIVATE="example.com/private-*"
export GONOPROXY="none"
go mod download example.com/private-x/lib@v1.0.0
```

Приватных модулей нет в публичной базе контрольных сумм, поэтому `GOPRIVATE` отключает для их путей проверку сумм.

> ⚠ `GONOPROXY="none"` обязателен: без него `GOPRIVATE` направит запросы приватных модулей напрямую в систему контроля версий, минуя Galleon.

Учётные данные клиент `go` читает из файла `~/.netrc`:

```
machine <GALLEON_HOST>
login <USERNAME>
password <PASSWORD>
```

Публикация

Штатной команды публикации у Go нет — файлы модуля загружаются по путям протокола module proxy:

```
curl -u <USERNAME>:<PASSWORD> -T v1.0.0.mod https://<GALLEON_HOST>/repository/go-
hosted/example.com/foo/@v/v1.0.0.mod
curl -u <USERNAME>:<PASSWORD> -T v1.0.0.zip https://<GALLEON_HOST>/repository/go-
hosted/example.com/foo/@v/v1.0.0.zip
```

Файл `.mod` должен байт в байт совпадать с `go.mod` внутри архива `.zip`. Повторная загрузка уже опубликованной версии отклоняется с кодом `409` — версии модулей неизменяемы; для новой сборки публикуйте новую версию.

Особенности

Особенности формата:

- Файлы версий (`.zip`, `.mod`, `.info`) неизменяемы по правилам протокола Go; как proxy-репозитории кешируют содержимое и обновляют списки версий, описано в разделе Проксирование и кеширование.

- Если `GOSUMDB` не настроен на `go-sumdb`, клиент проверяет контрольные суммы напрямую через sum.golang.org — для закрытого контура настраивайте обе переменные.

npm

Репозитории формата npm работают со штатными клиентами `npm`, `yarn` и `pnpm`. В примерах используются условные имена `npm-proxy` и `npm-hosted`.

Подключение к proxy

Укажите реестр в `.npmrc`:

```
registry=https://<GALLEON_HOST>/repository/npm-proxy/
```

```
npm install lodash
```

Приватные пакеты из hosted-репозитория подключаются привязкой области видимости к реестру — публичные пакеты продолжают идти через proxy:

```
@myscope:registry=https://<GALLEON_HOST>/repository/npm-hosted/
```

Если чтение требует аутентификации, добавьте строки учётных данных и для читаемого репозитория — по образцу из раздела Публикация, с соответствующим адресом.

Публикация

Добавьте в `.npmrc` учётные данные hosted-репозитория — логин и пароль либо «Name code» и «Pass code» токена (пароль кодируется в base64):

```
//<GALLEON_HOST>/repository/npm-hosted/:username=<USERNAME>
//<GALLEON_HOST>/repository/npm-hosted/:_password=<base64(<PASSWORD>)>
//<GALLEON_HOST>/repository/npm-hosted/:email=you@example.com
//<GALLEON_HOST>/repository/npm-hosted/:always-auth=true
```

```
npm publish --registry https://<GALLEON_HOST>/repository/npm-hosted/
```

Опубликованные версии неизменяемы: повторная публикация той же версии отклоняется.

Особенности

Особенности формата:

- Теги распространения (`npm dist-tag add/rm/ls`) поддерживаются, но тег `latest` вручную переместить нельзя — он обновляется только публикацией.
- `npm deprecate` работает, в том числе для уже опубликованных версий.

- Пакеты с областью видимости (`@scope/name`) работают во всех клиентах.
- В lock-файлы записываются адреса Galleon — ссылки на внешний реестр наружу не попадают.

PyPI

Репозитории формата PyPI работают с клиентами `pip` , `uv` , `poetry` и `twine` . В примерах используются условные имена `pypi-proxy` и `pypi-hosted` .

Подключение к proxy

Укажите индекс репозитория — путь заканчивается на `/simple/` :

```
pip install --index-url https://<USERNAME>:<PASSWORD>@<GALLEON_HOST>/repository/pypi-proxy/simple/ six==1.16.0
```

Для `uv` добавьте индекс в `pyproject.toml` (учётные данные — переменные `UV_INDEX_GALLEON_USERNAME` и `UV_INDEX_GALLEON_PASSWORD`):

```
[[tool.uv.index]]
name = "galleon"
url = "https://<GALLEON_HOST>/repository/pypi-proxy/simple/"
default = true
```

Для `poetry` (учётные данные — `POETRY_HTTP_BASIC_GALLEON_USERNAME` и `POETRY_HTTP_BASIC_GALLEON_PASSWORD`):

```
poetry source add --priority=primary galleon https://<GALLEON_HOST>/repository/pypi-proxy/simple/
```

Приватные пакеты

Пакеты из hosted-репозитория подключаются дополнительным индексом рядом с proxy.

Для `pip` — дополнительный индекс:

```
pip install --index-url https://<GALLEON_HOST>/repository/pypi-proxy/simple/ \
  --extra-index-url https://<GALLEON_HOST>/repository/pypi-hosted/simple/ mypkg
```

> ⚠ При нескольких индексах pip может выбрать пакет из любого из них — имена приватных пакетов должны быть уникальны относительно публичного PyPI (защита от подмены зависимостей).

Для `uv` — второй блок `[[tool.uv.index]]` с адресом `pypi-hosted` и закрепление приватного пакета за ним через `[tool.uv.sources]`:

```
[[tool.uv.index]]
name = "galleon-private"
url = "https://<GALLEON_HOST>/repository/pypi-hosted/simple/"

[tool.uv.sources]
mypkg = { index = "galleon-private" }
```

Публикация

Поддерживаются `twine`, `uv publish` и `poetry publish`. Аутентификация — только логин и пароль (или «Name code» и «Pass code» токена); формат одиночного секрета `__token__` не поддерживается.

Публикация через `twine` (`~/.pypirc`):

```
[distutils]
index-servers = galleon

[galleon]
repository = https://<GALLEON_HOST>/repository/pypi-hosted/
username = <USERNAME>
password = <PASSWORD>
```

```
python -m build
twine upload --repository galleon dist/*
```

Публикация через `uv`:

```
UV_PUBLISH_USERNAME=<USERNAME> UV_PUBLISH_PASSWORD=<PASSWORD> \
  uv publish --publish-url https://<GALLEON_HOST>/repository/pypi-hosted/ dist/*
```

Особенности

Особенности формата:

- Опубликованные версии неизменяемы: повторная загрузка файла отклоняется с сообщением «File already exists». Для идемпотентных публикаций в CI используйте `uv publish --check-url` или `poetry publish --skip-existing` — на уже загруженных байт в байт файлах они завершаются успешно.
- Имена проектов нормализуются по правилам PyPI: любое написание (`Foo_Bar`, `foo.bar`) ведёт на одну каноническую страницу `foo-bar`.

- Метаданные wheel-файлов отдаются отдельными файлами (PEP 658) — `uv lock` собирает информацию о зависимостях без скачивания самих пакетов.

RubyGems

Репозитории формата RubyGems работают с клиентами `gem` и `bundle`. В примерах используются условные имена `rubygems-proxy` и `rubygems-hosted`.

Подключение к proxy

Направьте Bundler на proxy-репозиторий зеркалом:

```
bundle config mirror.https://rubygems.org https://<GALLEON_HOST>/repository/rubygems-proxy
```

Установка одиночного гема — порядок флагов важен, `--clear-sources` должен идти до `--source`:

```
gem install colorize --clear-sources \
  --source https://<USERNAME>:<PASSWORD>@<GALLEON_HOST>/repository/rubygems-proxy
```

Установка из hosted-репозитория

Гемы, опубликованные в hosted-репозитории, подключаются источником в `Gemfile`:

```
source "https://<GALLEON_HOST>/repository/rubygems-hosted" do
  gem "foo"
end
```

Учётные данные для Bundler задаются командой (работает и для зеркала из предыдущего раздела):

```
bundle config set --global <GALLEON_HOST> <USERNAME>:<PASSWORD>
```

Публикация

Клиент `gem` передаёт значение ключа авторизации как есть, поэтому в `~/.gem/credentials` записывается готовая строка `Basic <BASE64>` (файл должен иметь права `0600`):

```
---
:galleon: Basic <BASE64>
```

```
gem build foo.gemspec
gem push foo-1.0.0.gem --key galleon --host https://<GALLEON_HOST>/repository/
rubygems-hosted
```

Особенности

Особенности формата:

- Опубликованные версии неизменяемы: повторная публикация той же версии отклоняется.
- Команда `gem yank` не поддерживается — версия удаляется через веб-интерфейс кнопкой «Удалить компонент» (нужна привилегия `delete`).
- Разрешение зависимостей идёт по протоколу Compact Index; классические команды (`gem fetch`, `gem list --remote`) тоже работают.

Cargo

Репозитории формата Cargo используют sparse-индекс (Cargo 1.70 и новее). В примерах используются условные имена `cargo-hosted` и `cargo-proxy`.

Подключение

Опишите реестр в `~/.cargo/config.toml`; для репозиториев, требующих аутентификации, явно укажите провайдер учётных данных:

```
[registry]
global-credential-providers = ["cargo:token"]

[registries.galleon]
index = "sparse+https://<GALLEON_HOST>/repository/cargo-hosted/"
```

Токен — готовая строка `Basic <BASE64>`; сохраните её командой `cargo login`:

```
printf 'Basic <BASE64>\n' | cargo login --registry galleon
```

Proxy для crates.io

Чтобы все публичные зависимости шли через Galleon, настройте подмену источника — реестр объявляется и в `[registries]`, чтобы к нему можно было привязать токен:

```
[registries.galleon-proxy]
index = "sparse+https://<GALLEON_HOST>/repository/cargo-proxy/"

[source.crates-io]
replace-with = "galleon-proxy"

[source.galleon-proxy]
registry = "sparse+https://<GALLEON_HOST>/repository/cargo-proxy/"
```

Если proxy-репозиторий требует аутентификации, сохраните токен и для него: `cargo login --registry galleon-proxy` (та же строка `Basic <BASE64>`).

Использование

Зависимость из hosted-реестра указывается в `Cargo.toml` с именем реестра:

```
mycrate = { version = "1", registry = "galleon" }
```

Публикация

Крейт отправляется в hosted-реестр по имени, объявленному в `registries.galleon`:

```
cargo publish --registry galleon
```

Особенности

Особенности формата:

- Повторная публикация существующей версии отклоняется с кодом `409`; имена сравниваются без учёта регистра, а `-` и `_` считаются одним символом.
- `cargo yank` поддерживается: версия помечается в индексе, но файл не удаляется — сборки с точной версией в `Cargo.lock` продолжают работать. Требуется привилегия `edit`.
- `cargo search` и `cargo owner` не поддерживаются.

raw

Репозитории формата raw хранят произвольные файлы: архивы релизов, модели, документы. В примерах используется условное имя `raw-hosted`.

Публикация

Загрузите файл запросом `PUT` по нужному пути:

```
curl -u <USERNAME>:<PASSWORD> -T release.tar.gz \
  https://<GALLEON_HOST>/repository/raw-hosted/releases/v1.0.0/release.tar.gz
```

В ответе возвращаются заголовки с контрольными суммами (`X-Checksum-Sha256` и другие). Загрузка доступна и через веб-интерфейс — раздел «Загрузка».

Скачивание и удаление

Скачивание и удаление файла:

```
curl -u <USERNAME>:<PASSWORD> -O https://<GALLEON_HOST>/repository/raw-hosted/
releases/v1.0.0/release.tar.gz
curl -u <USERNAME>:<PASSWORD> -X DELETE https://<GALLEON_HOST>/repository/raw-hosted/
releases/v1.0.0/release.tar.gz
```

Контрольные суммы возвращаются в заголовках `X-Checksum-*` при скачивании и загрузке. Отдельный checksum-файл (например, `file.tar.gz.sha256`) доступен, только если он был загружен как самостоятельный файл.

Proxy-репозитории

Raw-репозиторий может быть и кеширующим зеркалом внешнего HTTP-источника: файлы скачиваются теми же путями `GET` , а загрузка и удаление недоступны. Пример из стартового набора — `go-sumdb` , proxy-репозиторий базы контрольных сумм Go.

Особенности

Особенности формата:

- Поведение браузера при открытии файлов (показать или сохранить) задаёт администратор настройкой репозитория. В репозиториях с режимом показа `inline` публикация HTML-, SVG- и JavaScript-файлов отклоняется из соображений безопасности.
- Если политика записи репозитория запрещает перезапись, повторная загрузка по существующему пути отклоняется с кодом `409` .
- В путях отклоняются конструкции вида `..` , управляющие символы и ряд специальных символов.