Документация Deckhouse Airlock

Дата генерации: 26.10.2026

Документация

Обзор продукта **4**
- Компоненты системы 6
- Архитектура 8
  - Взаимодействие компонентов 8
  - Аутентификация 15
  - Авторизация 17
  - Proxy Service 21

Установка и настройка **23**
- Установка 23
  - Установка бинарных файлов 23
  - Развёртывание сервера на Linux 25
  - Развёртывание в Kubernetes 28
- Агенты Airlock 31
  - Подключение по join-токену 33
  - SSH-узел Linux 37
  - OpenSSH без агента 39
  - Агент Kubernetes 42
  - Доступ к базам данных 44
  - PostgreSQL 46
  - MySQL / MariaDB 48
  - OpenSearch / Elasticsearch 50
  - Веб-приложения 52
  - RDP и Windows 56
- Аутентификация 59
  - Локальные пользователи 59
  - Аутентификация OIDC 63
  - Аутентификация SAML 68
  - Двухфакторная аутентификация (OTP/TOTP) 72
  - WebAuthn (второй фактор) 74
- Политики доступа (RBAC) 77
  - Доступ к SSH-хостам 80
  - Доступ к Kubernetes 83
  - Доступ к базам данных 87

Руководство пользователя **91**
- Утилита airsh 91

Запросы доступа 98
Веб-интерфейс 103
Справочник 106
Справочник airsh 106
Справочник airctl 112

Обзор продукта

Airlock — это безопасный и удобный способ получать доступ к инфраструктуре и защищать её.

Платформа Airlock Access обеспечивает доступ по принципу наименьших привилегий на основе криптографической идентификации и модели Zero Trust, с встроенным управлением политиками и аудитом.

Сценарии использования

Организации используют Airlock для:

- **Устранения разрозненных систем доступа:** единая платформа для ролевого управления доступом, аудита и подключения ко всей инфраструктуре — от серверов до кластеров Kubernetes и баз данных.
- **Внедрения Zero Trust с устойчивыми к компрометации учётными данными:** аутентификация доступа к инфраструктуре с помощью краткосрочных сертификатов, проверяемых на каждом конечном узле.
- **Соответствия требованиям безопасности:** централизованный аудит, запись сессий и гибкое управление политиками доступа.

Возможности платформы

Airlock Access обеспечивает Zero Trust-подключение ко всей инфраструктуре. Пользователи получают доступ к серверам, базам данных, кластерам Kubernetes и другим ресурсам через защищённый прокси, даже если ресурсы находятся за межсетевым экраном.

Все ресурсы инфраструктуры входят в единый каталог с общей системой ролевого управления доступом (RBAC). Подключение выполняется через краткосрочные сертификаты, которые компоненты Airlock проверяют криптографическими методами.

Запросы доступа (Access Requests) — встроенная возможность платформы: пользователи могут запрашивать временное повышение привилегий к ролям или отдельным ресурсам, а назначенные рецензенты одобряют или отклоняют запросы через веб-интерфейс или CLI. Подробнее см. руководство пользователя.

Airlock Identity (дополнительный модуль) расширяет возможности управления доступом:

- **Списки доступа (Access Lists):** регулярный аудит и контроль членства в ролях.
- **Доверие к устройствам (Device Trust):** требование аутентификации с зарегистрированного устройства.
- **Блокировка сессий и идентичностей:** мгновенная остановка активности скомпрометированных пользователей.

Архитектура

Минимальный кластер Airlock состоит из **Auth Service** (служба аутентификации) и **Proxy Service** (прокси-сервис). Агенты Airlock, работающие на серверах или в Kubernetes, проксируют доступ к ресурсам инфраструктуры.

Пользователи аутентифицируются через Proxy Service с помощью краткосрочных сертификатов. Сертификаты содержат атрибуты пользователя Airlock, что позволяет агентам применять политики RBAC.

Основные компоненты

Компонент	Назначение
Auth Service	Управление пользователями, ролями, конфигурацией и центрами сертификации
Proxy Service	Точка входа для пользователей: веб-интерфейс, SSH, Kubernetes, базы данных
SSH Service	Доступ к серверам по SSH с аудитом и записью сессий
Kubernetes Service	Проксирование доступа к API Kubernetes
Database Service	Проксирование доступа к PostgreSQL, MySQL и другим СУБД
Application Service	Проксирование HTTP/TCP-трафика к внутренним приложениям
Desktop Service	Доступ к рабочим столам Windows через RDP

Подробнее см. раздел Компоненты системы и Архитектура.

Клиентские утилиты

Утилита	Назначение
`airsh`	Клиент для пользователей: вход в кластер, SSH, Kubernetes, базы данных
`airctl`	Инструмент администратора: управление пользователями, ролями, ресурсами
`airlock`	Сервер и агент: Auth Service, Proxy Service, сервисы доступа

Компоненты системы

Здесь описаны основные понятия и компоненты развёртывания Airlock. Эти термины встречаются при настройке и эксплуатации платформы.

Кластер Airlock

Ключевое понятие архитектуры Airlock — **кластер**. Кластер состоит из **Auth Service** и **Proxy Service**, а также **сервисов Airlock**, которые управляют доступом к ресурсам инфраструктуры: кластерам Kubernetes, рабочим столам Windows, базам данных и приложениям.

Минимальный кластер включает Auth Service и Proxy Service. В демонстрационной среде оба сервиса могут работать в одном процессе `airlock` на Linux-хосте.

Подробнее о том, как компоненты взаимодействуют между собой: Взаимодействие компонентов.

Auth Service

Auth Service управляет **локальными пользователями** и **ресурсами конфигурации** кластера. Он поддерживает центры сертификации (CA), которые позволяют пользователям и сервисам аутентифицироваться в кластере. Auth Service выдаёт сертификаты клиентам и ведёт журнал аудита.

Auth Service — единственный компонент кластера, которому требуется подключение к бэкенду для хранения состояния кластера и закрытых ключей CA. **Сервисы Airlock** не хранят состояние и взаимодействуют с Auth Service через gRPC API.

Для высокой доступности можно запустить несколько экземпляров Auth Service.

Подробнее: Аутентификация, Авторизация.

Proxy Service

Proxy Service обеспечивает безопасный доступ к ресурсам инфраструктуры из публичной сети без VPN.

Он устанавливает обратные туннели к Auth Service и сервисам Airlock, которые могут работать в частных сетях. В минимальной конфигурации достаточно открыть в интернет только порт `443`, а остальную инфраструктуру держать в частной сети.

Для production можно запустить **несколько экземпляров Proxy Service** за балансировщиком — это повышает доступность, распределяет нагрузку и позволяет снизить задержку за счёт размещения прокси ближе к пользователям или агентам.

Клиенты также могут подключаться к ресурсам напрямую с сертификатами Airlock, минуя Proxy Service.

Подробнее: Proxy Service.

Сервисы Airlock

Сервис Airlock управляет доступом к ресурсам инфраструктуры. Один процесс `airlock` может запускать один или несколько сервисов в зависимости от конфигурации.

Application Service

Проксирует HTTP- и TCP-трафик к настраиваемым конечным точкам — внутренним веб-приложениям и другим сервисам.

Database Service

Проксирует TCP-трафик в нативных протоколах популярных СУБД, включая PostgreSQL и MySQL.

Desktop Service

Проксирует трафик Remote Desktop Protocol (RDP) к рабочим столам Windows.

Kubernetes Service

Проксирует HTTP-трафик к API-серверу Kubernetes.

SSH Service

Реализация SSH-сервера, позволяющая выполнять команды на удалённых машинах с встроенным управлением доступом, аудитом и записью сессий.

Агент

Экземпляр сервиса Airlock называется **агентом**. Для всех сервисов, кроме SSH Service, один агент может обеспечивать доступ к нескольким ресурсам и работать на отдельном хосте от целевых ресурсов. Все агенты должны находиться в той же сети, что и целевые ресурсы.

Ресурсы конфигурации

Ресурс конфигурации — документ, хранящийся в бэкенде Auth Service и задающий параметры кластера. Примеры: **роли**, **локальные пользователи**, **коннекторы аутентификации**.

Роль

Роль — ресурс конфигурации, который предоставляет **пользователям Airlock** привилегии в кластере. RBAC в Airlock ограничивает доступ по умолчанию: пользователю нужны явные разрешения для доступа к ресурсу или выполнения административных действий.

Пользователи Airlock

Airlock поддерживает два типа пользователей:

- **Локальные пользователи** — соответствуют ресурсу `user`, хранящемуся в бэкенде Auth Service.
- **Пользователи SSO** — хранятся во внешнем провайдере идентификации (GitHub, SAML, OIDC). При аутентификации через SSO Airlock выдаёт сертификат и создаёт временного локального пользователя на срок действия сертификата.

В итоге пользователь Airlock — это субъект сертификата, выданного Auth Service. Auth Service проверяет наличие действительного сертификата Airlock и использует субъект сертификата — имя пользователя и роли — для авторизации.

Коннектор аутентификации

Коннектор аутентификации — ресурс конфигурации, позволяющий пользователям входить в Airlock через внешний провайдер идентификации (SSO).

Доверенные кластеры

Airlock поддерживает **отношение доверенного кластера** между **корневым кластером** и одним или несколькими **листовыми кластерами**, которые доверяют CA корневого кластера. Пользователи, аутентифицированные в корневом кластере, могут получать доступ к ресурсам листового кластера. Корневой и листовые кластеры работают независимо со своими пользователями, ролями и ресурсами; доверие позволяет сопоставлять роли корневого кластера с ролями и разрешениями листового.

Архитектура

Руководства по архитектуре компонентов Airlock.

Взаимодействие компонентов

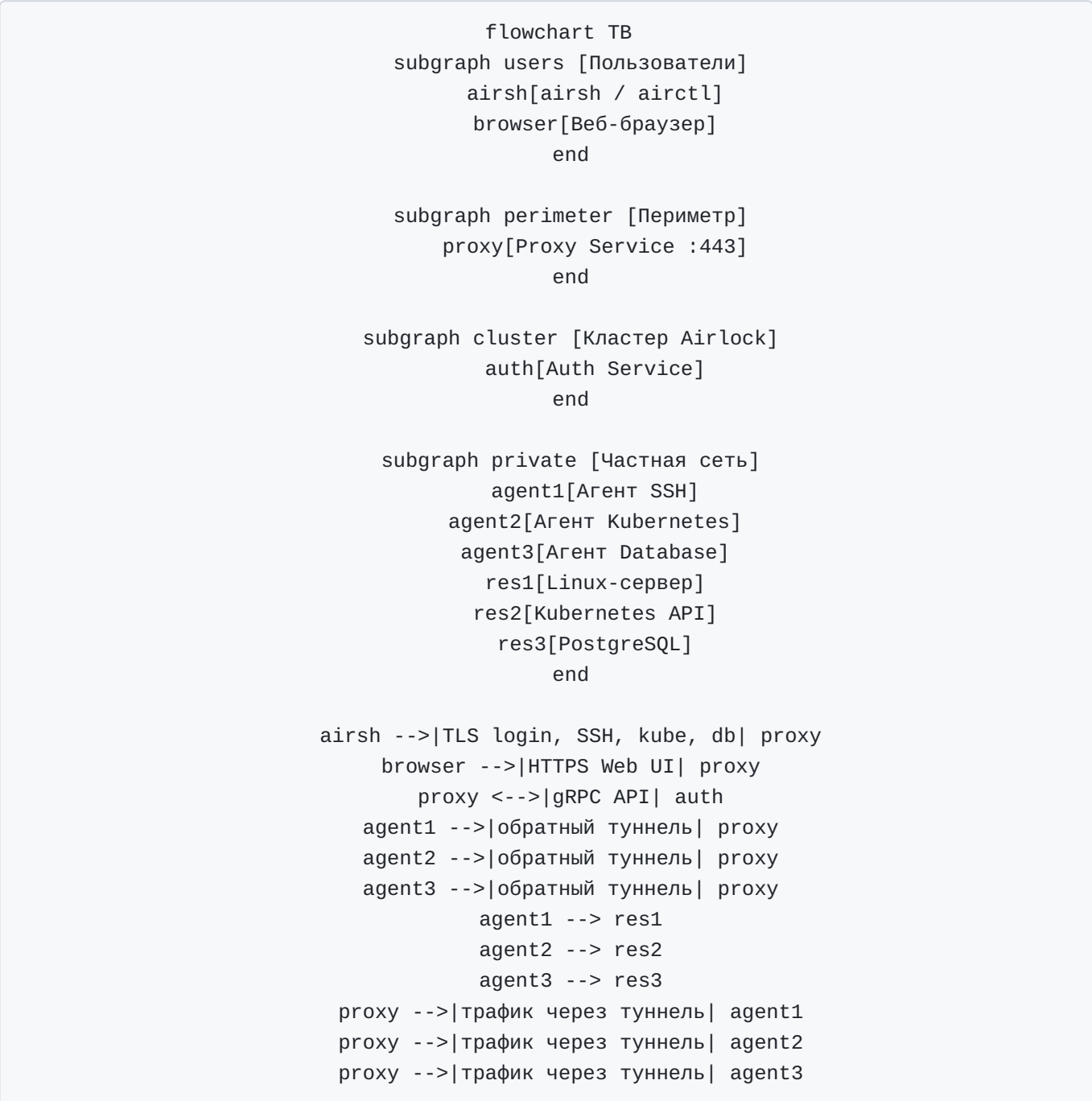
В этом документе описано, как основные компоненты Airlock взаимодействуют при установке, подключении агентов и доступе пользователей к ресурсам.

Роли компонентов

Компонент	Где работает	Назначение
Auth Service	Контрольный узел (частная сеть)	Центры сертификации, пользователи, роли, конфигурация, журнал аудита
Proxy Service	Периметр (порт 443)	Точка входа для пользователей, веб-интерфейс, маршрутизация трафика к агентам
Агент	Рядом с ресурсами (частная сеть)	SSH, Kubernetes, базы данных, приложения, RDP — проксирование к конкретным ресурсам
Пользователь	Рабочая станция	`airsh`, браузер — аутентификация и подключение к ресурсам

Auth Service — единственный компонент кластера с постоянным хранилищем состояния. Proxy Service и агенты не хранят конфигурацию кластера: они получают учётные данные от Auth Service и поддерживают соединение с ним через Proxy Service.

Общая схема



Пользователи обращаются только к **Proxy Service** (обычно `https://airlock.example.com`). Ресурсы и агенты в частной сети **не открывают** входящие порты в интернет — агенты сами устанавливают исходящие обратные туннели к прокси.

Auth Service

Auth Service выполняет функции центра доверия кластера:

- Хранит пользователей, роли, коннекторы SSO и описания ресурсов.
- Управляет центрами сертификации (CA) для пользователей, узлов и сервисов.

- Выдаёт краткосрочные сертификаты после успешной аутентификации.
- Принимает запросы на присоединение агентов (join-токены) и выдаёт агентам долгосрочные сертификаты.
- Записывает события аудита.

Auth Service не принимает пользовательский трафик к SSH, Kubernetes или базам данных напрямую. Клиенты и агенты обращаются к нему через gRPC API; для пользователей этот путь проходит через Proxy Service.

Proxy Service

Proxy Service — единая точка входа:

- **Веб-интерфейс** — вход пользователя, список ресурсов, запуск приложений, терминал SSH и RDP в браузере.
- **Identity-Aware Proxy (IAP)** — выдача пользовательских сертификатов после SSO или локального входа; клиент `airsh` использует эти сертификаты для последующих подключений.
- **Маршрутизация** — перенаправление SSH, Kubernetes, HTTPS приложений и трафика баз данных к нужному агенту через обратный туннель.
- **TLS-маршрутизация** — несколько протоколов через один порт 443.

Proxy Service поддерживает постоянные **обратные туннели** от агентов. Когда пользователь подключается к ресурсу, прокси выбирает агента, зарегистрированного для этого ресурса, и передаёт трафик по уже установленному туннелю.

Несколько экземпляров Proxy Service

В production можно запустить **несколько экземпляров Proxy Service** — на разных узлах, в разных зонах доступности или регионах. Это даёт:

- **Высокую доступность** — при отказе одного прокси пользователи и агенты продолжают работать через оставшиеся экземпляры.
- **Снижение нагрузки** — входящий трафик распределяется между прокси (балансировщик или DNS).
- **Меньшую задержку (latency)** — прокси можно разместить ближе к пользователям или к агентам, чтобы сократить сетевой путь.

Типовая схема:

```
flowchart TB
  users[Пользователи]
lb[Балансировщик / DNS]
  p1[Proxy 1]
  p2[Proxy 2]
  p3[Proxy N]
 auth[Auth Service]
  agent[Агент]

  users --> lb
   lb --> p1
   lb --> p2
   lb --> p3
  p1 --> auth
  p2 --> auth
  p3 --> auth
agent -->|туннель| p1
agent -->|туннель| p2
agent -->|туннель| p3
```

Рекомендации:

- Перед прокси — **балансировщик нагрузки** или DNS с несколькими A-записями; алгоритм **round-robin**, без sticky session.
- Агенты при подключении устанавливают обратный туннель **к каждому** Proxy Service в кластере — см. подключение по join-токену.
- Все экземпляры Proxy Service используют один Auth Service и общие TLS-сертификаты для одного домена (`airlock.example.com`).
- Auth Service также может работать в нескольких экземплярах для отказоустойчивости; состояние кластера хранится в общем бэкенде.

Размещение прокси в регионах, близких к командам пользователей или к пулам агентов, уменьшает RTT при входе и при проксировании сессий.

Агенты

Агент — процесс `airlock` с одним или несколькими сервисами (SSH, Kubernetes, Database, Application, Desktop), развёрнутый рядом с защищаемыми ресурсами.

Подключение агента к кластеру

1. Администратор создаёт join-токен: `airctl tokens add --type=node` (или `kube` , `db` , `app` и т.д.).
2. Агент запускается с токеном и адресом Proxy Service.
3. Агент устанавливает TLS-соединение с Proxy Service и регистрируется в Auth Service.
4. Auth Service выдаёт агенту сертификат; агент открывает обратный туннель к прокси.
5. Агент периодически обновляет сертификат и поддерживает туннель.

Агенту не нужен публичный `IP-адрес` или открытый порт — достаточно исходящего доступа к Proxy Service на порту `443`.

Работа агента при доступе пользователя

1. Пользователь аутентифицируется и получает сертификат с ролями и атрибутами.
2. Пользователь инициирует подключение (например, `airsh ssh user@host` или `airsh db connect`).
3. Proxy Service проверяет сертификат пользователя и RBAC на стороне Auth Service.
4. Прокси направляет трафик через туннель к агенту, обслуживающему ресурс.
5. Агент повторно проверяет сертификат и политики, затем устанавливает соединение с целевым ресурсом (SSH, API Kubernetes, PostgreSQL и т.д.).
6. События сессии записываются в журнал аудита Auth Service.

Пользователи

Вход и получение сертификата

Пользователь аутентифицируется одним из способов:

- **Локальная учётная запись** — логин и пароль (+ OTP или WebAuthn при включённом MFA).
- **SSO (OIDC / SAML)** — внешний провайдер идентификации.

После успешного входа Auth Service выдаёт **краткосрочный сертификат**, содержащий имя пользователя и роли. Клиент `airsh` сохраняет сертификат локально; браузер использует сессию для веб-интерфейса.

```
airsh login --proxy=airlock.example.com --user=alice
```

Доступ к ресурсам

Способ	Инструмент	Путь трафика
SSH	`airsh ssh`	Пользователь → Proxy → агент SSH → сервер
Kubernetes	`airsh kube login`	Пользователь → Proxy → агент Kubernetes → API-сервер
База данных	`airsh db connect`	Пользователь → Proxy → агент Database → СУБД
Приложение	Браузер / `airsh`	Пользователь → Proxy → агент Application → HTTP-сервис
RDP	Веб-UI / `airsh`	Пользователь → Proxy → агент Desktop → Windows

На каждом этапе проверяется сертификат Airlock; статические пароли к целевым ресурсам пользователю не передаются.

Администратор

Администратор использует `airctl` для управления кластером: пользователи, роли, токены, ресурсы. `airctl` может работать:

- на хосте Auth Service (с локальной конфигурацией);
- с рабочей станции после `airsh login` (через Proxy Service).

Типичные сценарии

Первоначальное развёртывание

1. Установите Auth Service и Proxy Service на контрольном узле (или в Kubernetes).
2. Настройте DNS и TLS для `airlock.example.com`.
3. Создайте локальных пользователей и роли через `airctl`.
4. Установите агентов на серверы, в кластеры Kubernetes и рядом с базами данных.
5. Пользователи выполняют `airsh login` и подключаются к ресурсам.

Ресурс за межсетевым экраном

Ресурсы и агенты остаются в частной сети без входящего доступа из интернета. Агент инициирует исходящее соединение к Proxy Service; пользователи из интернета подключаются только к прокси, который передаёт трафик по туннелю.

Прямое подключение (без прокси)

В некоторых конфигурациях клиент с действительным сертификатом Airlock может подключаться к агенту напрямую, минуя Proxy Service. Это снижает задержку, но требует сетевой доступности агента для клиента. По умолчанию рекомендуется маршрутизация через прокси.

Аутентификация

Аутентификация и авторизация

Airlock выполняет и аутентификацию, и авторизацию:

- **Аутентификация** — подтверждение личности пользователя или сервиса.
- **Авторизация** — проверка прав доступа к ресурсу.

В этом документе описана аутентификация с помощью краткосрочных сертификатов.

Краткосрочные сертификаты

Центры сертификации (CA) и краткосрочные сертификаты — основа аутентификации Airlock. В начале каждого подключения пользователь или сервис предъявляет действительный сертификат, выданный доверенным CA. Клиенты всегда инициируют взаимную TLS- или SSH-аутентификацию.

CA Airlock выдаёт краткосрочные сертификаты X.509 для веб-сервисов, баз данных, кластеров Kubernetes, рабочих столов и SSH-сертификаты для OpenSSH-совместимых серверов.

Преимущества сертификатов

- Сертификаты привязаны к идентичности пользователя или сервиса — каждое действие можно отследить.
- Краткосрочные сертификаты автоматически истекают, отзыв не требуется.
- Сертификаты решают проблему «доверия при первом подключении» (TOFU): все серверы в кластере имеют собственные идентичности и сертификаты и не принимают клиентские сертификаты, подписанные недоверенным CA.
- Сертификаты обеспечивают взаимную аутентификацию (mTLS), что снижает риск подмены, атак «человек посередине» и перебора учётных данных.
- Сертификаты лучше масштабируются: каждому сервису достаточно проверить подпись CA, без копирования учётных данных на все узлы.

Airlock выдаёт сертификаты со сроком действия от нескольких часов до нескольких минут. Чем короче срок — тем лучше. Идеально — выдавать сертификат только на время сессии. На практике допустимы несколько часов или рабочий день. Дату истечения в сертификате нельзя подделать без его аннулирования.

Сертификаты X.509

Сертификаты X.509 — те же сертификаты, что используются при доступе к веб-сайтам через браузер. Они связывают идентичность с открытым ключом подписью CA.

Airlock использует X.509 для Kubernetes, баз данных, веб-сервисов и внутренних компонентов (прокси, Auth Service) для установки взаимной TLS-аутентификации (mTLS).

Сертификаты OpenSSH

SSH-сертификаты аналогичны X.509 и также связывают идентичность пользователя или сервера с открытым ключом подписью CA.

SSH-сертификат содержит метаданные для аутентификации:

- Список principals (идентичностей), которым принадлежит сертификат.
- Подпись CA, выдавшего сертификат.
- Дату истечения (TTL).
- Дополнительные данные (например, роль узла) в расширениях сертификата.

Время как фактор безопасности

Истечение срока действия — встроенное свойство сертификатов. SSH- и X.509-сертификаты содержат дату истечения, которую серверы проверяют вместе с подписью.

Airlock выдаёт сертификаты на срок от нескольких часов до минут; по истечении они автоматически становятся недействительными. Вместо списков отзыва Airlock полагается на время.

> Если истечения сертификата недостаточно (например, при инциденте безопасности), Proxy Service может немедленно завершить активные соединения с помощью блокировки сессий и идентичностей.

Сертификаты для пользователей

Для выдачи сертификата пользователю Airlock открывает экран входа, выдаёт сертификат и доставляет его на компьютер пользователя.

Рекомендуется использовать SSO (GitHub, SAML, OIDC или другой провайдер идентификации) для получения краткосрочного сертификата.

Внутренние сертификаты

Внутренние сервисы Airlock — Auth, Proxy и узлы — используют сертификаты для идентификации в кластере. Для подключения прокси и узлов к кластеру администраторы используют краткосрочные токены присоединения.

В отличие от пользователей, внутренние сервисы получают долгосрочные сертификаты. Для их обновления администраторы выполняют ротацию CA — аннулирование всех ранее выданных сертификатов независимо от срока действия и выпуск новых с новым CA.

> Чтобы быстро заблокировать скомпрометированный узел, прокси или Auth Service без ротации сертификатов всего кластера, используйте блокировку сессий и идентичностей.

Авторизация

Аутентификация и авторизация

Airlock выполняет и аутентификацию, и авторизацию:

- **Аутентификация** — подтверждение личности пользователя или сервиса.
- **Авторизация** — проверка прав доступа к ресурсу.

В этом документе описана авторизация пользователей и сервисов на основе RBAC.

Пользователи и роли

Airlock поддерживает несколько типов учётных записей:

- Интерактивные и неинтерактивные.
- Локальные и внешние.

После успешной аутентификации каждому пользователю назначается одна или несколько ролей.

Интерактивные пользователи

Интерактивные пользователи могут быть локальными или внешними. Локальные учётные записи хранят хеш пароля и данные MFA в бэкенде Airlock. Внешние учётные записи аутентифицируются через провайдер идентификации по протоколам SSO — OAuth 2.0, OIDC или SAML.

Внешние пользователи из SSO

При каждом входе пользователя SSO Airlock создаёт временную запись учётной записи, которая автоматически истекает вместе с SSO-сессией, и записывает события в журнал аудита. Это предотвращает конфликты имён с локальными пользователями.

Внешние пользователи из других кластеров

Пользователь может быть внешним по отношению к кластеру Airlock, если другой кластер или CA выдаёт сертификат, которому доверяет этот кластер. В этом случае Airlock применяет логику сопоставления ролей доверенных кластеров.

Локальные интерактивные пользователи

Локальные интерактивные пользователи имеют запись в бэкенде Airlock с учётными данными. Администратор создаёт учётные записи через `airctl users add` или API.

Каждый локальный пользователь Airlock должен быть связан с одной или несколькими ролями — это называется «сопоставлением ролей».

Неинтерактивные пользователи

Airlock поддерживает неинтерактивных пользователей для сервисов автоматизации (например, CI/CD или микросервисов). Локальные неинтерактивные пользователи имеют запись, сопоставляющую имя с ролями, но без учётных данных в базе данных.

Ролевое управление доступом (RBAC)

Каждому пользователю Airlock назначается одна или несколько ролей, которые определяют доступ к ресурсам и API Airlock.

Правила allow и deny

Каждая роль содержит два списка правил: `allow` и `deny`:

- По умолчанию всё запрещено.
- Правила `deny` проверяются первыми и имеют приоритет.
- Правило состоит из ресурсов и глаголов (verbs).

Пример правила `allow` для просмотра списка записанных SSH- или Kubernetes-сессий:

```
allow:
  - resources: [session]
    verbs: [list]
```

Principals

Роли определяют, какие principals (например, пользователи ОС Linux или группы Kubernetes) могут использовать пользователи с этой ролью:

```
spec:
  allow:
    logins: [ubuntu]
    kubernetes_groups: [viewer]
```

Если у пользователя несколько ролей, списки principals объединяются.

Метки

Метки ролей определяют, к каким ресурсам применяются правила. Пример доступа к SSH-узлам и кластерам Kubernetes:

```
spec:
  allow:
    node_labels:
      'environment': '^test|staging$'
    kubernetes_labels:
      'region': 'us-west-*'
      'cluster_name': '^us.*\.example\.com$'
```

Правила применения меток:

- Для совпадения правила `allow` должны совпасть все метки в правиле.
- Для совпадения правила `deny` достаточно совпадения любой метки.

Пример: пользователь Alice имеет роли `dev` и `prod`. Роль `dev` разрешает SSH как `root` и полный доступ к Kubernetes как `system:masters` для ресурсов с метками `test` или `stage`. Роль `prod` разрешает SSH как `ubuntu` и доступ к Kubernetes как `view` для ресурсов с меткой `prod`.

- Alice может подключиться по SSH как `root` к серверу с меткой `test` или `stage`.
- Alice не может подключиться как `root` к серверу с меткой `prod` — роль `prod` разрешает только `ubuntu`.

Шаблоны ролей

Роли поддерживают переменные шаблонов:

```
spec:
  allow:
    logins: ['{{internal.logins}}']
    kubernetes_groups: ['{{external.groups}}']
```

Любая роль с интерполяцией переменных считается шаблоном роли.

Правила интерполяции:

- Если `external.groups` — список `["dev", "prod"]`, выражение `["{{external.groups}}"]` интерполируется в `["dev", "prod"]`.
- Если переменная отсутствует, выражение `"{{external.groups}}"` даёт пустую строку.
- Недопустимые значения (например, `-foo` как Unix-логин) опускаются.

Шаблоны ролей вычисляются в момент доступа к ресурсу прокси или узлом. Переменные из провайдера идентификации кодируются в расширениях X.509-сертификата.

Условия ролей

Поле `where` ограничивает доступ по условиям. Пример — доступ только к собственным сессиям:

```
kind: role
metadata:
  name: only-own-sessions
spec:
  allow:
    rules:
    - resources: [session]
      verbs: [list, read]
      where: contains(session.participants, user.metadata.name)
```

Параметры ролей

Помимо правил `allow` и `deny`, роли задают параметры подключения:

```
kind: role
version: v5
metadata:
  name: relaxed
spec:
  options:
    max_session_ttl: 8h
    lock: strict
```

При конфликте параметров из нескольких ролей Airlock выбирает наиболее строгое значение (например, меньший `max_session_ttl` или режим `strict` вместо `best_effort`).

Запросы доступа (Access Requests)

Роли могут разрешать запрос повышенных привилегий — других ролей или отдельных ресурсов. Роли определяют, кто может рассматривать запросы, и сколько одобрений или отклонений требуется. В Airlock эта возможность доступна через веб-интерфейс, `airsh` и предустановленные роли `requester` и `reviewer`.

```
spec:
  allow:
    review_requests:
      roles: ['dbadmin']
    request:
      roles: ['common', 'dev-*']
      thresholds:
        - approve: 2
          deny: 1
```

Proxy Service

Proxy Service Airlock — это прокси с учётом идентичности и веб-интерфейсом. Основные возможности:

- Аутентификация пользователей через SSO или локальные учётные данные для доступа к SSH и рабочим столам Windows через веб-интерфейс.
- Перехват трафика по протоколам SSH, Kubernetes, HTTPS и баз данных с проверкой аутентификации клиентов.
- Запись команд, API-вызовов и запросов с отправкой в журнал аудита.
- Обратные туннели для подключения серверов и прокси за межсетевыми экранами.
- TLS-маршрутизация — сжатие всех портов и протоколов в один TLS-порт.

> Минимальный кластер Airlock состоит из Auth Service и Proxy Service. В домашней лаборатории оба сервиса можно запустить одним бинарным файлом `airlock`.

Веб-интерфейс

В веб-интерфейсе Proxy Service использует WSS (защищённые веб-сокеты) для проксирования целевого ресурса — SSH-сервера или рабочего стола.

> При использовании веб-интерфейса Proxy Service завершает TLS-трафик и переформатирует данные для клиентского соединения.

Режим Identity-Aware Proxy (IAP)

В режиме IAP пользователи инициируют SSO или локальный вход для подписи открытых ключей на клиентской машине.

> Режим IAP считается более безопасным, чем доступ через веб-интерфейс: закрытые ключи не покидают клиент, соединение с ресурсом взаимно аутентифицировано, браузер используется меньше — снижается риск CSRF и перехвата файлов cookie.

Туннели

В этом режиме ресурсы за межсетевым экраном устанавливают обратные туннели к прокси. Прокси перенаправляет клиентские соединения к целевым ресурсам через эти туннели.

Например, пользователь подключается к кластеру Kubernetes за межсетевым экраном через два туннеля.

> Все описанные режимы включены в Proxy Service по умолчанию. Специальная настройка не требуется, если нужно отключить отдельные режимы.

Несколько экземпляров Proxy Service

Кластер Airlock может включать **несколько Proxy Service**, работающих параллельно. Это стандартный способ масштабирования и повышения отказоустойчивости.

Цель	Как помогает несколько прокси
Доступность	Отказ одного узла не блокирует доступ: трафик переключается на другие экземпляры
Нагрузка	Балансировщик распределяет сессии пользователей и соединения между прокси
Latency	Прокси в разных ДЦ/регионах сокращают путь до пользователей и до агентов

Пользователи подключаются к одному адресу (например, `https://airlock.example.com`) через балансировщик. Агенты при регистрации открывают обратные туннели ко **всем** Proxy Service кластера, поэтому любой прокси может маршрутизировать трафик к нужному агенту.

> ⚠ Балансировщик должен использовать round-robin (или аналог без привязки сессии к одному узлу). Sticky session для Proxy Service не подходит.

Подробнее о подключении агентов к нескольким прокси: подключение по join-токену.

Установка и настройка

Инструкции по установке и настройке Airlock для администраторов.

- Установка
- Агенты
- Аутентификация
- Политики доступа

Установка

- Установка бинарных файлов — `airlock`, `airsh`, `airctl`
- Сервер на Linux — демонстрационный кластер Auth + Proxy
- Сервер в Kubernetes — развёртывание через Helm

Установка бинарных файлов

В этом руководстве описана установка бинарных файлов Airlock на Linux:

- `airlock` — сервер и агент
- `airsh` — клиент командной строки
- `airctl` — инструмент администрирования
- `airbot` — агент Machine ID

Клиентские инструменты (`airsh`, `airctl`, `airbot`) должны быть той же мажорной версии, что и кластер, к которому они подключаются. Серверы Airlock совместимы с клиентами той же мажорной версии или на одну мажорную версию старше. Клиенты более новой мажорной версии не поддерживаются.

Поддерживаемые платформы

Airlock поддерживается на следующих дистрибутивах Linux. На всех перечисленных платформах доступны компоненты `airlock`, `airctl`, `airsh`, `airbot` и веб-интерфейс.

Поддерживаемые дистрибутивы Linux

Дистрибутив	Поддерживаемые версии
ALT Linux	8 СП (релиз 10), p10, 10.0, 10.1, 10.2, 11
Astra Linux Special Edition	1.7, 1.8
CentOS	7, 8, 9, 10*
Debian	10, 11, 12, 13
openSUSE	15.4, 15.5, 15.6
Rocky Linux	8, 9
Ubuntu	18.04, 20.04, 22.04, 24.04, 26.04
РЕД ОС	7.3, 8.0
РОСА Сервер	7.9, 12.4, 12.5, 12.6
МОС ОС	15.4, 15.5

Расширенная запись сессий требует ядра Linux v5.8+. Для установки через пакетные менеджеры (`apt` , `yum` , `zypper`) требуется systemd. Репозитории не содержат пакеты для всех вариантов дистрибутивов — при установке может потребоваться заменить `ID` на `ID_LIKE` для ближайшего поддерживаемого дистрибутива.

Установка на Linux

Установка включает все компоненты: `airlock` , `airsh` , `airctl` и `airbot` .

Установка через скрипт

Выполните на целевом хосте:

```
curl https://deckhouse.ru/downloads/airlock/install.sh | bash -s
```

Скрипт обновит репозиторий пакетного менеджера и установит последнюю версию Airlock.

Установка конкретной версии

Чтобы установить определённую версию, укажите её в конце команды:

```
curl https://deckhouse.ru/downloads/airlock/install.sh | bash -s v15.0.0
```

Проверка установки

```
airlock version
airsh version
airctl version
```

Установка из tar-архива

Альтернативно можно скачать архив с официальной страницы загрузок:

```
export version=v15.0.0
export os=linux
export arch=amd64
curl -O https://deckhouse.ru/downloads/airlock/airlock-${version}-${os}-${arch}-
bin.tar.gz
tar -xzf airlock-${version}-${os}-${arch}-bin.tar.gz
sudo ./install
```

Контрольные суммы

Для проверки целостности бинарного файла доступны SHA256-контрольные суммы на странице загрузок:

```
export version=v15.0.0
export os=linux
export arch=amd64
curl https://deckhouse.ru/downloads/airlock/airlock-${version}-${os}-${arch}-
bin.tar.gz.sha256
```

Следующие шаги

После установки бинарных файлов:

- Разверните кластер на Linux-сервере или в Kubernetes
- Подключите инфраструктуру с помощью агентов

Развёртывание сервера на Linux

Airlock обеспечивает подключение, аутентификацию, контроль доступа и аудит для инфраструктуры. Платформа включает прокси с учётом идентичности, центр сертификации с короткоживущими сертификатами, единую систему контроля доступа и туннелирование для доступа к ресурсам за брандмауэром.

В этом руководстве показано, как развернуть одноузловой кластер Airlock на Linux-сервере. После развёртывания можно настроить RBAC, зарегистрировать ресурсы и защитить тестовые или домашние окружения.

Компоненты демо-кластера

- **Auth Service** — центр сертификации кластера. Выпускает сертификаты и проводит проверку аутентификации. Обычно недоступен извне частной сети.
- **Proxy Service** — фронтенд кластера: обрабатывает запросы пользователей, передаёт учётные данные в Auth Service и взаимодействует с агентами.
- **SSH Service** — SSH-сервер с короткоживущими сертификатами, RBAC, записью сессий и другими возможностями Airlock.

Предварительные требования

- Linux-хост с открытым портом `443` для входящего трафика. Необходима возможность установки и запуска ПО. Для первоначальной настройки потребуется SSH-доступ (откройте порт SSH в дополнение к `443`).
- Приложение для многофакторной аутентификации (Authy, Google Authenticator, 1Password и т.п.).
- **Один** из вариантов:
  - Зарегистрированное доменное имя.
  - Авторитетный DNS-сервер организации и существующий центр сертификации. Браузер должен использовать DNS-сервер организации.

Шаг 1 из 4. Настройка DNS

Airlock использует TLS для защиты Proxy Service и Auth Service, поэтому нужно доменное имя для проверки сертификата. Создайте две DNS-записи типа `A`, указывающие на IP-адрес Linux-хоста. Для домена `airlock.example.com`:

Домен	Назначение
`airlock.example.com`	Трафик пользователей и сервисов к Proxy Service
`*.airlock.example.com`	Трафик к веб-приложениям, зарегистрированным в Airlock

Шаг 2 из 4. Установка Airlock на Linux-хост

Установка

```
curl https://deckhouse.ru/downloads/airlock/install.sh | bash -s
```

Настройка

Сгенерируйте конфигурационный файл с помощью `airlock configure`. Команда требует TLS-сертификат и закрытый ключ.

Развёртывание в публичной сети с Let’s Encrypt

Let’s Encrypt проверяет владение доменом через HTTPS на порту 443 Proxy Service. Замените `airlock.example.com` на домен кластера, а `admin@example.com` — на email для уведомлений:

```
sudo airlock configure -o file \
  --acme --acme-email=admin@example.com \
  --cluster-name=airlock.example.com
```

Порт `443` на хосте Proxy Service должен принимать трафик из любых источников.

Развёртывание в частной сети

Разместите закрытый ключ и цепочку сертификатов в `/var/lib/airlock/privkey.pem` и `/var/lib/airlock/fullchain.pem`. Сертификат должен соответствовать домену кластера.

```
sudo airlock configure -o file \
  --cluster-name=airlock.example.com \
  --public-addr=airlock.example.com:443 \
  --cert-file=/var/lib/airlock/fullchain.pem \
  --key-file=/var/lib/airlock/privkey.pem
```

Запуск

```
sudo systemctl enable airlock
sudo systemctl start airlock
sudo systemctl status airlock
```

Откройте веб-интерфейс по HTTPS: `https://airlock.example.com`. Должен отобразиться экран приветствия.

Шаг 3 из 4. Создание пользователя и MFA

Создайте администратора `airlock-admin` с правами входа как `root`, `ubuntu` или другие локальные учётные записи:

```
sudo airctl users add airlock-admin --roles=editor,access --logins=root,ubuntu
```

Команда выведет одноразовую ссылку для завершения регистрации пользователя (действует 1 час):

```
User "airlock-admin" has been created but requires a password. Share this URL with the
user to complete user setup, link is valid for 1h:
https://airlock.example.com:443/web/invite/...
```

Перейдите по ссылке, задайте пароль и настройте OTP по QR-коду.

> Учётные записи из флага `--logins` должны существовать на Linux-хосте. Создайте их через `adduser <login>` или укажите текущего пользователя: `airctl users add airlock-admin $(whoami)`.

Вход через CLI

Установите `airsh` на рабочую станцию (если ещё не установлен) и войдите в кластер:

```
airsh login --proxy=airlock.example.com --user=airlock-admin
```

Пример вывода:

```
Profile URL:        https://airlock.example.com:443
Logged in as:       airlock-admin
Cluster:            airlock.example.com
Roles:              access, editor
Logins:             root, ubuntu
Kubernetes:         enabled
Valid until:        ...
```

Шаг 4 из 4. Регистрация инфраструктуры

В веб-интерфейсе откройте раздел нужного типа ресурса (**Servers**, **Applications**, **Kubernetes** и т.д.) и следуйте инструкциям по регистрации.

На вкладке **Servers** уже будет виден Linux-сервер, на котором запущен кластер.

Следующие шаги

Агенты Airlock проксируют трафик к серверам, базам данных, кластерам Kubernetes и рабочим столам Windows. Подробнее см. раздел Агенты и подключение по join-токену.

Развёртывание в Kubernetes

Airlock обеспечивает единый защищённый доступ к Kubernetes-кластерам. В этом руководстве показано развёртывание Auth Service и Proxy Service в Kubernetes с помощью Helm.

Если Auth Service и Proxy Service уже работают на другой платформе, используйте существующий кластер Airlock и подключите Kubernetes по руководству по агенту Kubernetes.

Предварительные требования

- Зарегистрированное доменное имя (для TLS через Let’s Encrypt и проверки Proxy Service клиентами).
- Kubernetes-кластер с поддержкой балансировщика нагрузки и постоянных томов.
- Persistent Volume для хранения состояния Auth Service:

```
kubectl get pv
```

Если томов нет, включите динамическое выделение томов или создайте PV вручную. Проверьте наличие StorageClass по умолчанию:

```
kubectl get storageclasses
```

- Клиент `airsh` установлен на рабочей станции.
- Kubernetes >= 1.28, Helm >= 3.14:

```
helm version
kubectl version
```

Шаг 1 из 2. Установка Airlock

Добавление Helm-репозитория

```
helm repo add airlock https://charts.airlock.deckhouse.ru
helm repo update
```

Создание пространства имён

```
kubectl create namespace airlock-cluster
kubectl label namespace airlock-cluster 'pod-security.kubernetes.io/enforce=baseline'
kubectl config set-context --current --namespace=airlock-cluster
```

Файл значений Helm

Задайте `clusterName` (например, `airlock.example.com`) и `acmeEmail` для Let's Encrypt:

```
cat << EOF > airlock-cluster-values.yaml
clusterName: airlock.example.com
proxyListenerMode: multiplex
acme: true
acmeEmail: admin@example.com
EOF
```

Установка chart `airlock-cluster`

```
helm install airlock-cluster airlock/airlock-cluster \
  --create-namespace \
  --values airlock-cluster-values.yaml
```

Дождитесь запуска подов Auth Service и Proxy Service:

```
kubectl get pods
```

Ожидаемый вывод:

```
NAME                                        READY   STATUS    RESTARTS   AGE
airlock-cluster-auth-000000000-00000     1/1     Running   0          114s
airlock-cluster-proxy-0000000000-00000   1/1     Running   0          114s
```

Настройка DNS

Получите адрес внешнего балансировщика:

```
kubectl get services/airlock-cluster
```

Создайте DNS-записи для `airlock.example.com` и `*.airlock.example.com`:

Тип записи	Домен	Значение
A или CNAME	`airlock.example.com`	IP или домен балансировщика
A или CNAME	`*.airlock.example.com`	IP или домен балансировщика

Проверьте доступность кластера:

```
curl https://airlock.example.com/webapi/ping
```

Шаг 2 из 2. Создание локального пользователя

Локальные пользователи — надёжный резерв, когда SSO недоступен. Создайте роль `member` с доступом к группе Kubernetes `system:masters`:

```
kind: role
version: v7
metadata:
  name: member
spec:
  allow:
    kubernetes_labels:
      '*': '*'
    kubernetes_groups:
    - system:masters
```

```
kubectl exec -i deployment/airlock-cluster-auth -- airctl create -f - < member.yaml
```

Создайте пользователя и получите ссылку-приглашение:

```
kubectl exec -ti deployment/airlock-cluster-auth -- airctl users add myuser --roles=me
mber,access,editor
```

Перейдите по ссылке из вывода команды и активируйте учётную запись.

Войдите через `airsh`:

```
airsh login --proxy=airlock.example.com:443 --user=myuser
airsh kube ls
```

Для безопасной работы с kubeconfig используйте отдельный файл:

```
KUBECONFIG=$HOME/airlock-kubeconfig.yaml airsh kube login airlock.example.com
KUBECONFIG=$HOME/airlock-kubeconfig.yaml kubectl get -n airlock-cluster pods
```

Устранение неполадок

Если подключение не работает, проверьте статус подов:

```
kubectl get pods -n airlock-cluster
```

Под Auth Service может оставаться в состоянии `Pending`, если не удаётся выделить PersistentVolume Claim. Используйте `kubectl describe pod` и `kubectl logs` для диагностики:

```
kubectl get events --sort-by='.metadata.creationTimestamp' -A
```

Следующие шаги

- Настройте единый вход (SSO) для production-развёртываний.
- Подключите дополнительные Kubernetes-кластеры через агент Kubernetes.
- Ограничьте права доступа к Kubernetes с помощью RBAC Airlock вместо `system:masters`.

Агенты Airlock

Агенты Airlock — это экземпляры Airlock, настроенные на проксирование трафика к ресурсам инфраструктуры: серверам, базам данных, кластерам Kubernetes, рабочим столам Windows и веб-приложениям.

Обзор архитектуры

Сервисы

Каждый процесс Airlock может запускать один или несколько **сервисов**. Сервис включается в конфигурационном файле экземпляра.

Пулы агентов

Агенты обычно работают в тех же частных сетях, что и проксируемые ресурсы. Они должны быть единственными клиентами, имеющими прямой доступ к ресурсу без Airlock.

Агенты устанавливают обратные SSH-туннели к Proxy Service. Proxy Service доступен из интернета по HTTPS, а агентам не нужны открытые порты или публичный адрес.

Proxy Service использует обратные туннели для пересылки трафика по поддерживаемым протоколам на доступного агента. Агент применяет правила RBAC и передаёт трафик ресурсам.

Подключение агентов

Первоначальное подключение к кластеру

Агентам нужно установить доверие с Auth Service. Доступны несколько способов подключения для автоматизации в вашей среде — см. подключение по join-токену.

При первом запуске процесс Airlock читает конфигурацию и определяет включённые сервисы. Каждый сервис отдельно подключается к Auth Service, который проверяет наличие **join-токена** для этого сервиса. При успехе Auth Service выдаёт учётные данные, подписанные для конкретного сервиса.

Добавление нового сервиса на существующий агент

Учётные данные агентов подписаны для конкретных сервисов. Чтобы запустить новые сервисы, повторите процедуру подключения:

1. Сгенерируйте новый join-токен для всех сервисов агента, включая новые.
2. Обновите токен в конфигурации (`airlock.join_params` или `airlock.auth_token`).
3. Удалите каталог состояния агента (по умолчанию `/var/lib/airlock`, поле `airlock.data_dir`).
4. Перезапустите агент.

Рекомендуется развёртывать агентов через инфраструктуру как код и обновлять конфигурацию сервисов в проекте IaC с последующим передеплоем.

Регистрация инфраструктуры

Два способа зарегистрировать ресурсы через агентов:

- **Статический** — указать ресурс в конфигурационном файле агента.
- **Динамический** — применить ресурс конфигурации (YAML через `airctl`), чтобы Airlock управлял проксированием.

Подробные инструкции по типам ресурсов:

- SSH-узел — серверы Linux
- OpenSSH — legacy-серверы с OpenSSH
- Агент Kubernetes — кластеры Kubernetes
- RDP Windows — рабочие столы Windows
- Базы данных — PostgreSQL, MySQL, OpenSearch
- Веб-приложения — HTTP/HTTPS-приложения

Руководства в этом разделе

Документ	Описание
join-token	Подключение агента по одноразовому токену
ssh-node	Регистрация Linux-сервера с SSH Service
openssh	Доступ к серверам OpenSSH без агента
kubernetes-agent	Подключение Kubernetes-кластера
rdp-windows	Доступ к рабочим столам Windows по RDP
databases	Обзор Database Service
database-postgresql	PostgreSQL
database-mysql	MySQL / MariaDB
database-opensearch	OpenSearch / Elasticsearch
applications	Внутренние веб-приложения

Подключение по join-токену

В этом руководстве показано, как зарегистрировать процесс Airlock с одним или несколькими сервисами в кластере, предъявив **join-токен**.

Предварительные требования

- Linux-сервер для процесса Airlock (виртуальная машина или контейнер на базе Linux).

- В этом руководстве регистрируется SSH Service (`node`); тот же подход применим к Kubernetes Service, Database Service, Proxy Service и другим сервисам.
- На административной машине установлен `airctl` с доступом к кластеру.

> Join-токен работает и с одним Proxy Service, и с несколькими за LB или DNS с несколькими A-записями. Процесс при подключении устанавливает туннель к каждому Proxy Service. Балансировщик должен использовать round-robin, а не sticky session.

Шаг 1 из 3. Установка Airlock

На Linux-хосте:

```
curl https://deckhouse.ru/downloads/airlock/install.sh | bash -s
```

Шаг 2 из 3. Подключение к кластеру

Генерация токена

На локальной машине с `airctl`:

```
airctl tokens add --ttl=5m --type=node
```

Пример вывода:

```
The invite token: abc123...
This token will expire in 5 minutes.

Run this on the new node to join the cluster:

> airlock start \
   --roles=node \
   --token=abc123... \
   --ca-pin=sha256:... \
   --auth-server=192.0.2.0:3025
```

Поддерживаемые типы токенов

Тип	Сервис Airlock
`app`	Application Service
`auth`	Auth Service
`bot`	Machine ID
`db`	Database Service
`discovery`	Discovery Service
`kube`	Kubernetes Service
`node`	SSH Service
`proxy`	Proxy Service
`windowsdesktop`	Windows Desktop Service

Список активных токенов:

```
airctl tokens ls
```

> ⚠ Предпочитайте короткоживущие токены статическим. Статические токены в конфигурации Auth Service проще украсть или они могут утечь.

Запуск процесса с токеном

На хосте нового агента (замените `JOIN_TOKEN` и `airlock.example.com`):

```
sudo airlock configure \
  --roles=node \
  --token=JOIN_TOKEN \
  --proxy=airlock.example.com:443 \
  -o file
```

Для SSH Service можно использовать `airlock node configure` без флага `--roles=node`:

```
sudo airlock node configure \
  --output=file:///etc/airlock.yaml \
  --token=JOIN_TOKEN \
  --proxy=airlock.example.com:443
```

Запустите сервис:

```
sudo systemctl enable airlock
sudo systemctl start airlock
```

Проверьте регистрацию узла:

```
airctl nodes ls
```

Прямое подключение к Auth Service

Если Proxy Service недоступен с хоста агента, подключайтесь напрямую к Auth Service только при отсутствии других вариантов. Минимизируйте входящий трафик к Auth Service.

Получите CA pin:

```
airctl status
```

Настройте агент:

```
sudo airlock configure \
  --roles=node \
  --token=JOIN_TOKEN \
  --auth-server=airlock.example.com:3025 \
  -o file
```

Укажите CA pin в `/etc/airlock.yaml` (поле `airlock.ca_pin`):

```
sudo sed -i 's|  ca_pin: ""|  ca_pin: "sha256:..."|' /etc/airlock.yaml
sudo systemctl restart airlock
```

> CA pin становится недействительным после ротации CA (`airctl auth rotate`).

Шаг 3 из 3. Отзыв токена

Чтобы запретить использование токена:

```
airctl tokens rm TOKEN_TO_DELETE
```

Следующие шаги

- SSH-узел — полный сценарий доступа к серверу
- Агент Kubernetes — подключение кластера Kubernetes

SSH-узел Linux

Сервер защищается запуском SSH Service на нём и регистрацией в кластере Airlock. RBAC контролирует доступ, а журнал аудита фиксирует действия.

SSH Service открывает обратный SSH-туннель к Proxy Service; клиентский трафик проходит через этот туннель — аналогично схеме bastion-хоста.

> ⚠ Не запускайте SSH Service как pod Kubernetes для доступа к узлам кластера — pod может оказаться не на том сервере, к которому нужен доступ.

Предварительные требования

- Linux-хост (Ubuntu 20.04, CentOS 8, Debian 10 и т.п.) — будущий узел Airlock.
- Работающий кластер Airlock и `airctl` на административной машине.
- Временно открытый порт 22 для первоначальной настройки.

Шаг 1 из 4. Установка Airlock на сервере

На целевом хосте:

```
curl https://deckhouse.ru/downloads/airlock/install.sh | bash -s
```

Шаг 2 из 4. Добавление сервера в кластер

Создание join-токена

На рабочей станции сохраните токен в файл:

```
airctl tokens add --type=node --format=text > token.file
```

Подключение сервера

Скопируйте `token.file` на сервер в защищённый каталог и сгенерируйте конфигурацию (замените `airlock.example.com`):

```
sudo airlock node configure \
  --output=file:///etc/airlock.yaml \
  --token=/path/to/token.file \
  --proxy=airlock.example.com:443
```

Запустите SSH Service:

```
sudo systemctl enable airlock
sudo systemctl start airlock
sudo systemctl status airlock
```

Пользователь для веб-интерфейса

```
airctl users add myuser --roles=editor,access,reviewer --logins=root,ubuntu
```

Перейдите по ссылке из вывода команды, задайте пароль и MFA. Сервер появится в веб-интерфейсе на вкладке **Servers**.

Шаг 3 из 4. SSH на сервер

Вход в кластер

```
airsh login --proxy=airlock.example.com --user=myuser
```

Пример профиля:

```
Profile URL:      https://airlock.example.com:443
Logged in as:     myuser
Cluster:          airlock.example.com
Roles:            access, editor, reviewer
Logins:           root, ubuntu
```

Список узлов

```
airsh ls
```

```
Node Name        Address        Labels
---------------- -------------- ---------------------------------------
bastion-host     ← Tunnel
web-server       127.0.0.1:3022 env=prod, hostname=web-server
```

Подключение

```
airsh ssh root@web-server
```

Эквивалент через стандартный `ssh`:

```
airsh config > ssh_config_airlock
ssh -F ssh_config_airlock root@web-server.airlock.example.com
```

Все команды записываются в журнал аудита.

Шаг 4 из 4. Обзор ресурсов кластера

```
airctl nodes ls
```

Фильтр по метке:

```
airsh ls env=prod
```

Выполнение команды на всех узлах с меткой:

```
airsh ssh root@env=prod ls
```

Усиление защиты (опционально)

После настройки закройте порт 22 на сервере — доступ останется через `airsh ssh`. На хосте Proxy Service порт 22 также можно закрыть.

OpenSSH без агента

Airlock может проксировать SSH к существующим серверам **OpenSSH** (`sshd`) без установки бинарника `airlock` на узел. Серверы принимают динамически выданные сертификаты от CA Airlock.

Это быстрый способ подключить legacy-системы. В перспективе рекомендуется заменить `sshd` на `airlock` для полной поддержки RBAC, записи сессий без разрыва соединения, совместного доступа к сессиям и расширенной записи.

Proxy Service проверяет роли пользователя, аутентифицируется на OpenSSH-узле сертификатом CA Airlock и ведёт аудит.

Предварительные требования

- OpenSSH 6.9+ на клиентской машине: `ssh -V`
- Linux-хост с `sshd` 7.4+, **без** Airlock; порт SSH открыт для Proxy Service
- `airctl` и `airsh` на административной машине

Шаг 1. Ресурс узла в кластере

Создайте `openssh-node-resource.yaml`:

```
kind: node
version: v2
sub_kind: openssh
metadata:
  name: a100fdd0-52db-4eca-a7ab-c3afa7a1564a
  labels:
    env: prod
spec:
  addr: 1.2.3.4:22
  hostname: openssh-node
```

- `spec.addr` — IP и порт SSH
- `spec.hostname` — имя для пользователей в Airlock
- `metadata.name` — UUID (`uuidgen`)

```
airctl create -f openssh-node-resource.yaml
```

Шаг 2. Доверие sshd к CA Airlock

На хосте с `sshd` (замените `airlock.example.com`):

```
export KEY=$(curl 'https://airlock.example.com/webapi/auth/export?type=openssh' | sed
"s/cert-authority\ //")
sudo bash -c "echo \"$KEY\" > /etc/ssh/airlock_openssh_ca.pub"
sudo bash -c "echo 'TrustedUserCAKeys /etc/ssh/airlock_openssh_ca.pub' >> /etc/ssh/
sshd_config"
sudo systemctl restart sshd
```

Шаг 3. Сертификат хоста

Создайте роль для выпуска host-сертификатов (`host-certifier.yaml`):

```
kind: role
version: v5
metadata:
  name: host-certifier
spec:
  allow:
    rules:
      - resources: [host_cert]
        verbs: [list, create, read, update, delete]
```

```
airctl create -f host-certifier.yaml
```

Назначьте роль своему пользователю через `airctl users update`.

UUID узла:

```
airctl get node/openssh-node --format=json | jq -r ".[0].metadata.name"
```

Выпустите сертификат хоста:

```
ADDR=1.2.3.4,openssh-node,a100fdd0-52db-4eca-a7ab-c3afa7a1564a
airctl auth sign --host=${ADDR} --format=openssh --out=myhost
```

Скопируйте `myhost` и `myhost-cert.pub` на сервер в `/etc/ssh`, права `0600`, добавьте в `sshd_config`:

```
HostKey /etc/ssh/myhost
HostCertificate /etc/ssh/myhost-cert.pub
```

```
sudo systemctl restart sshd
```

Шаг 4. Конфигурация SSH-клиента

Войдите в кластер:

```
airsh login --proxy=airlock.example.com --user=myuser
airsh config > ssh_config_airlock
```

Шаг 5. Подключение

```
USER=ubuntu
CLUSTER=airlock.example.com
PORT=22
ADDR_NODE=openssh-node
ssh -p ${PORT} -F ssh_config_airlock "${USER}@${ADDR_NODE}.${CLUSTER}"
```

Порт 22 переопределяется явно: у OpenSSH-узла нет обратного туннеля на 3022, Proxy Service подключается напрямую к `spec.addr`.

> Airlock использует сертификаты OpenSSH. В разделе `Principals` сертификата должны быть все имена и адреса, по которым вы подключаетесь.

Следующие шаги

Для новых серверов предпочтительнее SSH-узел с агентом.

Агент Kubernetes

В этом руководстве показана регистрация Kubernetes-кластера путём развёртывания Kubernetes Service (chart `airlock-kube-agent`) на защищаемом кластере.

Pod Kubernetes Service определяет, что работает в Kubernetes, и автоматически регистрирует кластер в Airlock.

Предварительные требования

- Работающий кластер Airlock, `airctl` и `airsh` совместимой с сервером версии:

```
airctl version
airsh version
```

- Kubernetes >= 1.28, Helm >= 3.14, `kubectl` с доступом к целевому кластеру:

```
helm version
kubectl version
```

- Доступ администратора к веб-интерфейсу Airlock (для мастера регистрации) или возможность выполнить шаги вручную ниже.

Шаг 1 из 3. RBAC в Airlock и Kubernetes

Роль Airlock

Создайте `kube-access.yaml`:

```
kind: role
metadata:
  name: kube-access
version: v6
spec:
  allow:
    kubernetes_labels:
      '*': '*'
    kubernetes_resources:
      - kind: 'pod'
        namespace: '*'
        name: '*'
    kubernetes_groups:
    - viewers
  deny: {}
```

```
airctl create -f kube-access.yaml
```

Назначьте роль пользователю через `airctl users update`.

ClusterRoleBinding в Kubernetes

Группа `viewers` должна иметь права в кластере. Создайте `viewers-bind.yaml`:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: viewers-crb
subjects:
- kind: Group
  name: viewers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: view
  apiGroup: rbac.authorization.k8s.io
```

```
kubectl apply -f viewers-bind.yaml
```

Шаг 2 из 3. Развёртывание Kubernetes Service

Helm-репозиторий

```
helm repo add airlock https://charts.airlock.deckhouse.ru
helm repo update
```

Ручная установка

Сгенерируйте join-токен:

```
airctl tokens add --type=kube --ttl=1h
```

Создайте файл значений `airlock-kube-agent-values.yaml` (подставьте токен и адрес proxy):

```
roles: kube
proxyAddr: airlock.example.com:443
authToken: "YOUR_KUBE_JOIN_TOKEN"
kubeClusterName: my-k8s-cluster
```

Установите chart в namespace `airlock-agent`:

```
kubectl create namespace airlock-agent
helm install airlock-agent airlock/airlock-kube-agent \
  --namespace airlock-agent \
  --values airlock-kube-agent-values.yaml
```

Проверьте pod:

```
kubectl get pods -n airlock-agent
```

Через веб-интерфейс: **Enroll New Resource** → **Kubernetes** — скопируйте сгенерированную команду и выполните её на рабочей станции.

Шаг 3 из 3. Проверка доступа

Войдите в Airlock:

```
airsh login --proxy=airlock.example.com:443 --user=myuser
```

Список кластеров:

```
airsh kube ls
```

Получите учётные данные для `kubectl`:

```
airsh kube login my-k8s-cluster
kubectl get pods -n airlock-agent
```

Ожидаемый вывод:

```
NAME               READY   STATUS    RESTARTS   AGE
airlock-agent-0   1/1     Running   0          8m
```

Следующие шаги

- Ограничьте доступ через метки Kubernetes и роли Airlock вместо широких прав `view` на все namespace.
- Если Auth Service и Proxy Service уже развёрнуты отдельно, достаточно агента без chart `airlock-cluster`.

Доступ к базам данных

Airlock Database Service проксирует TCP-трафик к базам данных в их нативных протоколах. Пользователи подключаются через `airsh db connect`, а Airlock проверяет сертификаты и применяет RBAC.

Поддерживаемые self-hosted СУБД

СУБД	Протокол Airlock	Руководство
PostgreSQL	`postgres`	PostgreSQL
MySQL / MariaDB	`mysql`	MySQL / MariaDB
OpenSearch / Elasticsearch	`elastic`	OpenSearch

Общая схема

1. Database Service подключается к кластеру Airlock (join-токен типа `db`).
2. Между Database Service и СУБД настраивается взаимная TLS-аутентификация (mTLS).
3. Администратор создаёт роли RBAC с правами `db_labels`, `db_names`, `db_users`.
4. Пользователь выполняет `airsh db ls` и `airsh db connect`.

Join-токен для Database Service

```
airctl tokens add --type=db --format=text > /tmp/db-token
```

Запуск Database Service (Linux)

```
sudo airlock db configure create \
  --token=/tmp/db-token \
  --name=example-db \
  --proxy=airlock.example.com:443 \
  --protocol=postgres \
  --uri=postgres.example.com:5432 \
  --output file:///etc/airlock.yaml

sudo systemctl enable --now airlock
```

Роль для доступа к базам

```
airctl create <<EOF
kind: role
version: v3
metadata:
  name: db
spec:
  allow:
    db_labels:
      '*': '*'
    db_names:
    - '*'
    db_users:
    - '*'
EOF

airctl users add --roles=access,db alice
```

Подробнее о политиках: Политики доступа к базам данных.

Подключение пользователя

```
airsh login --proxy=airlock.example.com --user=alice
airsh db ls
airsh db connect --db-user=postgres --db-name=postgres example-db
```

PostgreSQL

Руководство описывает подключение self-hosted PostgreSQL к Airlock через Database Service с взаимной TLS-аутентификацией.

Предварительные требования

- Экземпляр PostgreSQL в вашей инфраструктуре.
- Клиент `psql` в `PATH` на рабочей станции пользователя.
- Хост для Database Service (Linux или Kubernetes).
- Доступ к `airctl` на машине администратора.

Шаг 1. Join-токен и пользователь Airlock

```
airctl tokens add --type=db --format=text > /tmp/token

airctl users add --roles=access,db alice
```

Шаг 2. Сертификаты для mTLS

Database Service подключается к PostgreSQL по TLS. Сгенерируйте сертификат для хоста, с которого агент достигает PostgreSQL:

```
airctl auth sign --format=db --host=db.example.com --out=server --ttl=2190h
```

Будут созданы файлы `server.cas`, `server.crt`, `server.key` — их нужно разместить на сервере PostgreSQL.

Шаг 3. Настройка PostgreSQL

В `postgresql.conf`:

```
ssl = on
ssl_cert_file = '/path/to/server.crt'
ssl_key_file = '/path/to/server.key'
ssl_ca_file = '/path/to/server.cas'
```

В `pg_hba.conf` добавьте аутентификацию по клиентскому сертификату:

```
hostssl all             all             ::/0                    cert
hostssl all             all             0.0.0.0/0               cert
```

Убедитесь, что нет правил `md5` с более высоким приоритетом для тех же подключений — иначе Airlock не сможет войти по сертификату.

Перезапустите PostgreSQL.

Шаг 4. Database Service

Linux

```
curl https://deckhouse.ru/downloads/airlock/install.sh | bash -s

sudo airlock db configure create \
  --token=/tmp/token \
  --name=example-postgres \
  --proxy=airlock.example.com:443 \
  --protocol=postgres \
  --uri=postgres.example.com:5432 \
  --output file:///etc/airlock.yaml

sudo systemctl enable --now airlock
```

Kubernetes

Установите chart `airlock-kube-agent` с ролью `db` и параметрами базы данных (см. Агент Kubernetes для добавления Helm-репозитория).

Шаг 5. Подключение

```
airsh login --proxy=airlock.example.com --user=alice
airsh db ls
airsh db connect --db-user=postgres --db-name=postgres example-postgres
```

Завершение сессии:

```
airsh db logout example-postgres
```

Список доступных баз зависит от роли RBAC — см. Политики доступа к базам.

MySQL / MariaDB

Руководство описывает подключение self-hosted MySQL или MariaDB к Airlock через Database Service.

Предварительные требования

- Экземпляр MySQL или MariaDB.
- Хост для Database Service.
- Доступ к `airctl`.

Шаг 1. Join-токен

```
airctl tokens add --type=db --format=text > /tmp/token
```

Шаг 2. Сертификаты для mTLS

```
airctl auth sign --format=db --host=db.example.com --out=server --ttl=2190h
```

Скопируйте `server.cas`, `server.crt`, `server.key` на хост с MySQL/MariaDB.

Шаг 3. Настройка MySQL

В `mysql.cnf`:

```
[mysqld]
require_secure_transport=ON
ssl-ca=/path/to/server.cas
ssl-cert=/path/to/server.crt
ssl-key=/path/to/server.key
```

Шаг 4. Настройка MariaDB

В `mysql.cnf`:

```
[mariadb]
require_secure_transport=ON
ssl-ca=/path/to/server.cas
ssl-cert=/path/to/server.crt
ssl-key=/path/to/server.key
```

Пользователь базы данных

Создайте пользователя с аутентификацией по сертификату:

```
CREATE USER 'alice'@'%' REQUIRE SUBJECT '/CN=alice';
GRANT ALL ON `mydb`.* TO 'alice'@'%';
```

Для существующего пользователя:

```
ALTER USER 'alice'@'%' REQUIRE SUBJECT '/CN=alice';
```

Airlock аутентифицирует пользователей по сертификатам; пароль для таких учётных записей не используется.

Шаг 5. Database Service и пользователь Airlock

```
airctl users add --roles=access,db alice

sudo airlock db configure create \
  --token=/tmp/token \
  --name=example-mysql \
  --proxy=airlock.example.com:443 \
  --protocol=mysql \
  --uri=mysql.example.com:3306 \
  --output file:///etc/airlock.yaml

sudo systemctl enable --now airlock
```

Шаг 6. Подключение

```
airsh login --proxy=airlock.example.com --user=alice
airsh db ls
airsh db connect --db-user=root --db-name=mysql example-mysql
```

Для подключения нужен клиент `mysql` или `mariadb` в `PATH`.

```
airsh db logout example-mysql
```

OpenSearch / Elasticsearch

Airlock поддерживает self-hosted OpenSearch и Elasticsearch через протокол `elastic`. Требуется mutual TLS с клиентскими сертификатами.

Предварительные требования

- Self-hosted OpenSearch или Elasticsearch (управляемые облачные сервисы без поддержки клиентских сертификатов не подходят).
- Хост для Database Service.
- Доступ к `airctl`.

Шаг 1. Join-токен и Database Service

```
airctl tokens add --type=db --format=text > /tmp/token

sudo airlock db configure create \
  --token=/tmp/token \
  --name=my-opensearch \
  --proxy=airlock.example.com:443 \
  --protocol=elastic \
  --uri=opensearch.example.com:9200 \
  --output file:///etc/airlock.yaml

sudo systemctl enable --now airlock
```

Шаг 2. Пользователь Airlock

```
airctl users add --roles=access,db alice
```

Шаг 3. Role mapping в OpenSearch

Сопоставьте пользователя Airlock с ролью OpenSearch:

```
curl -u admin:password -X POST "https://opensearch.example.com:9200/_plugins/
_security/api/rolesmapping/all_access" \
  -H 'Content-Type: application/json' -d '{
  "users": ["alice"]
}'
```

Для Elasticsearch используйте API `_security/role_mapping/` с правилом по полю `username`.

Пример для Elasticsearch:

```
curl -u elastic:password -X POST "https://opensearch.example.com:9200/_security/
role_mapping/mapping1?pretty" \
  -H 'Content-Type: application/json' -d '{
  "roles": ["user"],
  "enabled": true,
  "rules": {
    "field": { "username": "alice" }
  }
}'
```

Шаг 4. Mutual TLS

```
airctl auth sign --format=elasticsearch --host=opensearch.example.com --out=elastic --
ttl=2160h
```

Файлы `elastic.cas`, `elastic.crt`, `elastic.key` — настройте их в конфигурации кластера.

Для OpenSearch в `opensearch.yml`:

```
plugins.security.ssl.http.enabled: true
plugins.security.ssl.http.pemcert_filepath: /path/to/elastic.crt
plugins.security.ssl.http.pemkey_filepath: /path/to/elastic.key
plugins.security.ssl.http.pemtrustedcas_filepath: /path/to/elastic.cas
plugins.security.ssl.http.clientauth_mode: REQUIRE
```

Для Elasticsearch 7.x+ с X-Pack в `elasticsearch.yml`:

```
xpack.security.http.ssl:
  certificate_authorities: /path/to/elastic.cas
  certificate: /path/to/elastic.crt
  key: /path/to/elastic.key
  enabled: true
  client_authentication: required
  verification_mode: certificate

xpack.security.authc.realms.pki.pki1:
  order: 1
  enabled: true
  certificate_authorities: /path/to/elastic.cas
```

Перезапустите кластер.

Шаг 5. Подключение

```
airsh login --proxy=airlock.example.com --user=alice
airsh db ls
airsh db connect my-opensearch --db-user=alice
```

Туннель для GUI-клиентов:

```
airsh proxy db my-opensearch --db-user=alice --tunnel
```

Подключите клиент (OpenSearch Dashboards, Elasticvue и т.п.) к `http://localhost:<порт>`.

Веб-приложения

Airlock Application Service проксирует HTTP/HTTPS-трафик к внутренним веб-приложениям: панелям мониторинга, CI/CD, wiki, Kubernetes Dashboard и другим сервисам, доступным только из частной сети.

Пользователи открывают приложение через веб-интерфейс Airlock или по адресу `https://<имя-приложения>.<домен-прокси>`.

Предварительные требования

- Работающий кластер Airlock (Auth + Proxy).
- DNS-запись `*.airlock.example.com`, указывающая на Proxy Service (для поддоменов приложений).
- TLS-сертификат для Proxy Service и wildcard для поддоменов приложений.
- Хост, Docker-контейнер или Kubernetes для приложения и Application Service.
- Доступ к `airctl`.

Шаг 1. Join-токен

```
airsh login --user=admin --proxy=airlock.example.com
airctl tokens add \
  --type=app \
  --app-name=grafana \
  --app-uri=http://localhost:3000 \
  --format=text > /tmp/token
```

Шаг 2. Запуск приложения

Пример Grafana в Docker:

```
docker run --detach --name grafana --publish 3000:3000 grafana/grafana
```

В `grafana.ini` укажите домен Proxy Service:

```
[server]
domain = airlock.example.com
```

Пример Grafana в Kubernetes:

```
helm repo add grafana https://grafana.github.io/helm-charts
helm install example-grafana grafana/grafana \
  --create-namespace \
  --namespace example-grafana
```

Внутренний URI в кластере: `http://example-grafana.example-grafana.svc.cluster.local`.

Шаг 3. Application Service

Linux / Docker

```
sudo airlock configure \
  --output=file \
  --proxy=airlock.example.com:443 \
  --token=/tmp/token \
  --roles=app \
  --app-name=grafana \
  --app-uri=http://localhost:3000

sudo systemctl enable --now airlock
```

Конфигурационный файл

```
version: v3
airlock:
  data_dir: /var/lib/airlock-app
  auth_token: /tmp/token
  proxy_server: airlock.example.com:443
app_service:
  enabled: yes
  apps:
  - name: grafana
    uri: http://localhost:3000
    public_addr: grafana.airlock.example.com
    labels:
      env: prod
auth_service:
  enabled: "no"
ssh_service:
  enabled: "no"
proxy_service:
  enabled: "no"
```

```
sudo airlock start --config=/etc/airlock.yaml
```

Kubernetes

```
JOIN_TOKEN=$(cat /tmp/token)
helm install airlock-kube-agent airlock/airlock-kube-agent \
  --create-namespace \
  --namespace airlock-agent \
  --set roles=app \
  --set proxyAddr=airlock.example.com:443 \
  --set authToken=${JOIN_TOKEN} \
  --set "apps[0].name=grafana" \
  --set "apps[0].uri=http://example-grafana.example-grafana.svc.cluster.local" \
  --set "apps[0].labels.env=dev"
```

Шаг 4. Пользователь и доступ

```
airctl users add --roles=access appuser
```

Способы доступа:

1. Веб-UI Airlock → вкладка **Applications** → **Launch**.
2. Прямой URL: `https://grafana.airlock.example.com`.

Имя приложения

Имя приложения (`--app-name`) должно быть допустимым поддоменом: до 63 символов, только `a-z` , `0-9` , `-` .

После регистрации приложение доступно по адресу `<app-name>.<proxy-host>` , например `grafana.airlock.example.com` . Можно переопределить через `public_addr` в конфигурации.

Дополнительные параметры

Приложение в подкаталоге

```
- name: k8s-dashboard
  uri: "http://10.0.1.60:8001/api/v1/namespaces/kube-system/services/https:kubernetes-
dashboard:/proxy/"
  public_addr: k8s.airlock.example.com
```

Перезапись redirect

Для приложений с внутренними редиректами:

```
- name: jenkins
  uri: https://localhost:8001
  public_addr: jenkins.airlock.example.com
  rewrite:
    redirect:
    - localhost
    - jenkins.internal.dev
```

Self-signed сертификат приложения

```
- name: app
  uri: https://localhost:8443
  public_addr: app.airlock.example.com
  insecure_skip_verify: true
```

⚠ Не рекомендуется для production.

Заголовки и JWT

Airlock может добавлять заголовки к запросам, в том числе JWT с информацией о пользователе:

```
- name: dashboard
  uri: https://localhost:4321
  public_addr: dashboard.airlock.example.com
  rewrite:
    headers:
    - "Authorization: Bearer {{internal.jwt}}"
```

Выход из приложения

Принудительный выход до истечения сессии:

```
https://grafana.airlock.example.com/airlock-logout
```

TCP-приложения

Для приложений без HTTP (plain TCP) используйте TCP App Access — укажите протокол TCP в конфигурации Application Service.

Политики доступа

Доступ к приложениям настраивается через RBAC (`app_labels` в ролях). См. Политики доступа.

RDP и Windows

Airlock обеспечивает безопасный доступ без пароля к рабочим столам Microsoft Windows для локальных пользователей Windows (не домен Active Directory).

Предварительные требования

- Linux-сервер для Windows Desktop Service (можно добавить сервис к существующему экземпляру Airlock с новым join-токеном).
- Физическая или виртуальная машина Windows с включённым Remote Desktop; порт RDP (обычно 3389) открыт для Linux-сервера с агентом.
- `airctl` на административной машине с доступом к кластеру.

Шаг 1 из 4. Подготовка Windows

На компьютере Windows в командной строке (`cmd.exe`):

1. Экспорт CA пользовательских сертификатов Airlock:

```
curl.exe -fo airlock.cer https://airlock.example.com/webapi/auth/export?type=windows
```

1. Загрузка установщика Airlock Windows Auth (подставьте актуальную версию):

```
curl.exe -fo airlock-windows-auth-setup-amd64.exe https://deckhouse.ru/downloads/
airlock/airlock-windows-auth-setup-amd64.exe
```

1. Запуск установщика: выберите экспортированный сертификат `airlock.cer`. Установщик:
   - доверяет CA Airlock;
   - устанавливает DLL для интеграции;
   - отключает NLA для RDP;
   - при необходимости включает RemoteFX.
2. Перезагрузите компьютер.

Автоматическая установка из elevated cmd/PowerShell:

```
airlock-windows-auth-setup-amd64.exe install --cert=airlock.cer -r
```

Шаг 2 из 4. Windows Desktop Service

1. Сгенерируйте join-токен:

```
airctl tokens add --type=windowsdesktop
```

1. Сохраните токен в файл на Linux-сервере, например `/tmp/token`.
2. Убедитесь, что Airlock установлен:

```
airlock version
```

При необходимости:

```
curl https://deckhouse.ru/downloads/airlock/install.sh | bash -s
```

1. Настройте `/etc/airlock.yaml`:

```
version: v3
airlock:
  nodename: windows-agent.example.com
  proxy_server: airlock.example.com:443
  auth_token: /tmp/token
windows_desktop_service:
  enabled: yes
  static_hosts:
  - name: win-desktop-1
    ad: false
    addr: 192.0.2.155
    labels:
      datacenter: dc1
auth_service:
  enabled: no
proxy_service:
  enabled: no
ssh_service:
  enabled: no
```

- `proxy_server` — адрес Proxy Service кластера.
- `static_hosts` / `non_ad_hosts` — IP или имена Windows-машин (без AD автообнаружение недоступно).

1. Запуск:

```
sudo systemctl enable airlock
sudo systemctl start airlock
sudo systemctl status airlock
```

Шаг 3 из 4. Роль для доступа к рабочим столам

Создайте `windows-desktop-admins.yaml`:

```
kind: role
version: v6
metadata:
  name: windows-desktop-admins
spec:
  allow:
    windows_desktop_labels:
      "*": "*"
    windows_desktop_logins: ["Administrator", "alice"]
```

```
airctl create -f windows-desktop-admins.yaml
```

Назначьте роль пользователю через `airctl users update`.

Шаг 4 из 4. Подключение

1. Войдите в веб-интерфейс Airlock под учётной записью с ролью `windows-desktop-admins`.
2. **Resources** → фильтр **Desktops** → **Connect** у нужного рабочего стола → выберите логин Windows.
3. Сессия RDP открывается в браузере и записывается. Для отключения используйте **Disconnect**.
4. Запись просматривается в **Management** → **Session Recordings**.

Следующие шаги

- Сузьте доступ через `windows_desktop_labels` и отдельные роли для разных групп пользователей.
- Для нескольких рабочих столов используйте метки в `static_hosts` и RBAC по меткам.

Аутентификация

Раздел описывает способы аутентификации пользователей в Airlock: локальные учётные записи, протоколы единого входа и многофакторную аутентификацию.

- Локальные пользователи — создание учётных записей, назначение ролей, пользовательские политики
- OIDC — настройка OpenID Connect (Keycloak и другие IdP)
- SAML — единый вход через SAML (ADFS и другие IdP)
- OTP/TOTP — одноразовые коды как второй фактор
- WebAuthn — аппаратные ключи и биометрия как второй фактор

Локальные пользователи

В Airlock любому локальному, SSO- или робот-пользователю можно назначить одну или несколько ролей. Роли определяют доступ к базам данных, SSH-серверам, кластерам Kubernetes, рабочим столам Windows и веб-приложениям.

Локальные пользователи — это учётные записи, управляемые непосредственно через Airlock, а не через внешний провайдер идентификации. Все локальные пользователи хранятся в backend состояния кластера: имя, роли, traits и bcrypt-хеш пароля.

Предустановленные роли

Airlock предоставляет несколько предустановленных ролей:

Роль	Описание
`access`	Доступ к ресурсам кластера
`editor`	Редактирование настроек конфигурации кластера
`auditor`	Чтение событий кластера, аудит-логов и воспроизведение записей сессий
`requester`	Создание запросов на доступ (Access Requests)
`reviewer`	Рассмотрение и одобрение запросов на доступ

Добавление локальных пользователей

Для управления локальными пользователями используется утилита `airctl`. Добавьте пользователя Alice с ролью `editor`:

```
airctl users add alice --roles=editor
```

При необходимости можно указать несколько ролей и разрешённые OS-логины:

```
airctl users add joe --logins=joe,root --roles=access,editor
```

Airlock генерирует одноразовую ссылку-приглашение (действительна один час). Пользователь переходит по ссылке, задаёт пароль и при необходимости настраивает MFA. После регистрации учётная запись отображается в списке:

```
airctl users ls
```

Пример вывода:

```
User                 Roles
-------------------- --------------
alice                editor
joe                  access, editor
```

Обновление ролей

Обновите роли пользователя командой `airctl users update`:

```
airctl users update alice --set-roles=editor,auditor
```

Если у пользователя несколько ролей, разрешения объединяются (логическое «ИЛИ»). Пользователь может одновременно выполнять функции администратора и аудитора.

Редактирование и удаление

Получить определение пользователя для редактирования:

```
airctl get user/joe > joe.yaml
# отредактируйте joe.yaml
airctl create -f joe.yaml
```

Удалить локального пользователя:

```
airctl users rm joe
```

Создание пользовательской роли

Создайте роль для стажёров с доступом к тестовым и staging SSH-серверам, мониторинговым приложениям и dev-кластеру Kubernetes.

Сохраните роль в файл `interns.yaml`:

```
kind: role
version: v6
metadata:
  name: interns
spec:
  allow:
    logins: ['readonly']
    kubernetes_groups: ["view"]
    node_labels:
      'env': ['staging', 'test']
    kubernetes_labels:
      'env': 'dev'
    kubernetes_resources:
      - kind: 'pod'
        namespace: "*"
        name: "*"
    app_labels:
      'type': ['monitoring']
  deny:
    node_labels:
      'env': 'prod'
    kubernetes_labels:
      'env': 'prod'
    kubernetes_resources:
      - kind: 'pod'
        namespace: 'prod'
        name: '*'
    db_labels:
      'env': 'prod'
    app_labels:
      'env': 'prod'
```

Создайте роль:

```
airctl create -f interns.yaml
airctl get roles --format text
```

Принципы RBAC

Роль Airlock содержит два списка правил: `allow` и `deny`.

- По умолчанию доступ запрещён.
- Правила `deny` проверяются первыми и имеют приоритет.

Доступ к ресурсам настраивается через метки (labels) на ресурсах и соответствующие правила в роли. Несколько меток в одном правиле интерпретируются как логическое «И».

Блокировка после неудачных попыток входа

Локальный пользователь блокируется на 20 минут, если за 30 минут было несколько неудачных попыток входа или сброса пароля.

Разблокировать пользователя может администратор с правами на ресурсы `user` (роль `editor`):

```
airctl get users/alice > user.yaml
```

Отредактируйте поле `status.is_locked` на `false` и примените изменения:

```
airctl create -f user.yaml
```

Следующие шаги

- Настройте OIDC или SAML для единого входа
- Включите OTP или WebAuthn для двухфакторной аутентификации
- Ознакомьтесь с политиками доступа для SSH, Kubernetes и баз данных

Аутентификация OIDC

Это руководство описывает настройку SSO-провайдера на базе OpenID Connect (OIDC) для выдачи учётных данных Airlock группам пользователей. В сочетании с RBAC администраторы могут задавать политики:

- Только члены группы «DBA» могут подключаться к PostgreSQL.
- Разработчикам запрещён SSH-доступ к production-серверам.

Предварительные требования

- Административный доступ к SSO/IdP с пользователями, распределёнными по группам.
- Роль Airlock с правом управления ресурсами `oidc` (доступна в роли `editor` по умолчанию).
- Установленная утилита `airctl` и доступ к кластеру.

Регистрация приложения в IdP

Зарегистрируйте Airlock во внешнем провайдере идентификации и получите `client_id` и `client_secret`.

URL перенаправления OIDC

OIDC использует HTTP-редиректы для возврата управления в Airlock после аутентификации. URL перенаправления должен быть заранее согласован с IdP.

Формат URL:

```
https://<адрес-proxy>:443/v1/webapi/oidc/callback
```

Замените `<адрес-proxy>` на публичный адрес Proxy Service Airlock (например, `airlock.example.com`).

Настройка OIDC-коннектора

Коннекторы создаются, тестируются и удаляются через `airctl` или веб-интерфейс Airlock.

Создайте файл `client-secret.txt`, содержащий только client secret. Сгенерируйте YAML-конфигурацию коннектора:

```
airctl sso configure oidc --name keycloak \
  --issuer-url https://keycloak.example.com/realms/master \
  --id airlock \
  --secret $(cat client-secret.txt) \
  --claims-to-roles groups,/users,access \
  --claims-to-roles groups,/admins,editor > oidc-connector.yaml
```

Параметр	Описание
`--name`	Имя коннектора в Airlock (обычно имя IdP)
`--issuer-url`	Базовый URL OIDC-провайдера без `.well-known/openid-configuration`
`--id`	Client ID, заданный в IdP
`--secret`	Client secret от IdP
`--claims-to-roles`	Сопоставление claim/значения с ролями Airlock

Пример готового коннектора:

```
kind: oidc
metadata:
  name: keycloak
spec:
  claims_to_roles:
  - claim: groups
    roles:
    - access
    value: /users
  - claim: groups
    roles:
    - editor
    value: /admins
  client_id: airlock
  client_secret: abc123...
  issuer_url: https://keycloak.example.com/realms/master
  redirect_url: https://airlock.example.com:443/v1/webapi/oidc/callback
version: v3
```

Пример: Keycloak

В Keycloak на `keycloak.example.com` создайте клиент `airlock`. В client scope `airlock-dedicated` добавьте mapper «Group Membership», чтобы claim `groups` передавался в токен.

Тестирование и применение

Перед применением проверьте коннектор:

```
cat oidc-connector.yaml | airctl sso test
```

Команда откроет браузер и попытается выполнить вход через IdP. При ошибке изучите вывод — раздел «[OIDC] Claims» показывает все claims пользователя от IdP.

После успешного теста создайте коннектор:

```
airctl create -f oidc-connector.yaml
```

Вход через OIDC:

```
airsh --proxy=airlock.example.com login --auth=keycloak
```

Дополнительные параметры

ACR Values

Airlock может отправлять Authentication Context Class Reference (ACR) при получении authorization code. Если поле `acr_values` задано, Airlock ожидает то же значение в claim `acr`.

```
kind: oidc
version: v2
metadata:
  name: oidc-connector
spec:
  issuer_url: "https://oidc.example.com"
  client_id: "xxxxxxxxxxxxxxxxxxxxxxx.example.com"
  client_secret: "zzzzzzzzzzzzzzzzzzzzzzz"
  redirect_url: "https://airlock.example.com/v1/webapi/oidc/callback"
  display: "Login with Example"
  acr_values: "foo/bar"
  scope: [ "group" ]
  claims_to_roles:
     - claim: "group"
       value: "editor"
       roles: [ "editor" ]
     - claim: "group"
       value: "user"
       roles: [ "access" ]
```

Max age

Поле `max_age` задаёт максимальный возраст сессии OIDC до принудительной повторной аутентификации. По умолчанию не задано. Значение `0s` принуждает к повторной аутентификации при каждом входе.

```
spec:
  max_age: 24h
```

Prompt

Управляет поведением Authorization Server при повторной аутентификации и согласии. По умолчанию Airlock использует `select_account`.

```
spec:
  # Допустимые значения: none, select_account, login, consent
  prompt: 'login'
```

Отключение проверки email

По умолчанию Airlock проверяет claim `email_verified`. Для тестирования можно включить `allow_unverified_email` (снижает безопасность):

```
kind: oidc
version: v2
metadata:
  name: connector
spec:
  allow_unverified_email: true
```

Username claim

По умолчанию в качестве имени пользователя Airlock использует email. Альтернативный claim:

```
spec:
  username_claim: preferred_username
```

Аутентификация по умолчанию

Чтобы OIDC стал методом входа по умолчанию, обновите `cluster_auth_preference`:

```
airctl get cap > cap.yaml
```

```
kind: cluster_auth_preference
metadata:
  name: cluster-auth-preference
spec:
  type: oidc
version: v2
```

```
airctl create -f cap.yaml
```

Устранение неполадок

При ошибке `Failed to calculate user attributes` проверьте сопоставление `claims_to_roles` — значения claim должны точно совпадать с тем, что возвращает IdP. Используйте `airctl sso test` для просмотра всех claims.

Аутентификация SAML

Это руководство описывает настройку SAML для единого входа (SSO) в Airlock. В сочетании с RBAC администраторы могут задавать политики:

- Только члены группы «DBA» могут подключаться к PostgreSQL.
- Разработчикам запрещён SSH-доступ к production-серверам.

Airlock выступает в роли SAML Service Provider (SP) и принимает аутентификацию от внешнего Identity Provider (IdP).

Предварительные требования

- Развёрнутый SAML IdP (например, Active Directory Federation Services) с пользователями в группах.
- Роль Airlock с правом управления ресурсами `saml` (доступна в роли `editor`).
- Установленная утилита `airctl`.

Общие принципы SAML

SAML-аутентификация включает:

1. **IdP (Identity Provider)** — провайдер, который аутентифицирует пользователя и выпускает SAML assertion.
2. **SP (Service Provider)** — Airlock, который принимает assertion и создаёт сессию.
3. **ACS URL** — endpoint Airlock, куда IdP отправляет SAML-ответ: `https://<proxy>/v1/webapi/saml/acs`.
4. **Entity Descriptor** — XML-метаданные IdP с сертификатами и endpoints.
5. **Атрибуты (claims)** — группы, email, имя пользователя — сопоставляются с ролями Airlock через `attributes_to_roles`.

Шаг 1. Настройка IdP (пример ADFS)

В Active Directory Federation Services настройте:

Claims Provider Trust — входящие claims от Active Directory:

- `Name ID`: LDAP-атрибут `E-Mail-Addresses` → `Name ID`
- `Group`: claim членства в группах для сопоставления с ролями
- При необходимости: `SAM-Account-Name` → `Windows account name`, `E-Mail-Addresses` → `UPN`

Relying Party Trust — доверие к Airlock:

- Display name: `Airlock`
- SAML 2.0 Web SSO URL: `https://airlock.example.com/v1/webapi/saml/acs`
- Relying party trust identifier: тот же ACS URL
- Claim Issuance Policy: отправлять как минимум `Name ID` и `Group`

Убедитесь, что у пользователей Active Directory заполнено поле email.

Шаг 2. Создание ролей Airlock

Создайте роли для администраторов и обычных пользователей.

```
# admin-role.yaml
kind: role
version: v3
metadata:
  name: admin
spec:
  options:
    max_session_ttl: "8h0m0s"
  allow:
    logins: [ root ]
    node_labels:
      "*": "*"
    rules:
      - resources: ["*"]
        verbs: ["*"]
```

```
# user-role.yaml
kind: role
version: v3
metadata:
  name: dev
spec:
  options:
    max_session_ttl: "1h"
  allow:
    logins:
      - '{{external["http://schemas.microsoft.com/ws/2008/06/identity/claims/
windowsaccountname"]}}'
      - ubuntu
    node_labels:
      "access": "relaxed"
```

Роль `dev` разрешает вход только на узлы с меткой `access: relaxed`. Шаблон `{{external[...]}}` подставляет значение SAML-атрибута как разрешённый OS-логин.

Шаг 3. Создание SAML-коннектора

Сгенерируйте конфигурацию коннектора:

```
airctl sso configure saml --acs https://airlock.example.com/v1/webapi/saml/acs \
  --preset adfs \
  --entity-descriptor https://adfs.example.com/FederationMetadata/2007-06/
FederationMetadata.xml \
  --attributes-to-roles http://schemas.xmlsoap.org/claims/Group,teleadmins,editor \
  --attributes-to-roles http://schemas.xmlsoap.org/claims/Group,Users,access \
  > adfs.yaml
```

Параметр	Описание
`--acs`	Assertion Consumer Service URL (должен совпадать с настройкой IdP)
`--entity-descriptor`	URL или путь к XML-метаданным IdP
`--attributes-to-roles`	Сопоставление атрибута, значения и роли Airlock

Проверьте коннектор перед применением:

```
cat adfs.yaml | airctl sso test
```

Примените коннектор:

```
airctl create -f adfs.yaml
```

Экспорт ключа подписи

Экспортируйте сертификат подписи Airlock и добавьте его в IdP как certificate for signature verification:

```
airctl saml export adfs > saml.cer
```

Вход через SAML

```
airsh --proxy=airlock.example.com login
```

При нескольких SAML-коннекторах укажите имя:

```
airsh login --auth=adfs
```

Сопоставление атрибутов

Поле `attributes_to_roles` связывает SAML-атрибуты с ролями Airlock:

```
kind: saml
version: v2
metadata:
  name: saml-connector
spec:
  acs: https://airlock.example.com/v1/webapi/saml/acs
  entity_descriptor: |
    <?xml version="1.0"?>
    <!-- XML-метаданные IdP -->
  attributes_to_roles:
    - name: "groups"
      value: "admins"
      roles: ["editor"]
    - name: "groups"
      value: "developers"
      roles: ["access"]
```

Полное имя атрибута зависит от IdP. Для ADFS группы часто имеют URI `http://schemas.xmlsoap.org/claims/Group`.

Динамические traits

Значения SAML-атрибутов доступны в ролях через `{{external["имя-атрибута"]}}`. Это позволяет назначать OS-логины, метки окружений и другие параметры на основе данных IdP.

Аутентификация по умолчанию

Чтобы SAML стал методом входа по умолчанию:

```
kind: cluster_auth_preference
metadata:
  name: cluster-auth-preference
spec:
  type: saml
version: v2
```

```
airctl get cap > cap.yaml
# отредактируйте cap.yaml
airctl create -f cap.yaml
```

Устранение неполадок

- Убедитесь, что ACS URL в IdP и коннекторе совпадают.
- Проверьте, что IdP отправляет необходимые атрибуты (группы, email).
- Используйте `airctl sso test` для диагностики SAML-потока.
- При ошибках подписи проверьте, что сертификат Airlock добавлен в IdP.

Двухфакторная аутентификация (OTP/TOTP)

Airlock поддерживает многофакторную аутентификацию (MFA) для локальных пользователей. Тип `otp` реализует стандарт TOTP — одноразовые коды на основе времени.

Для генерации кодов можно использовать любой TOTP-клиент: Authy, FreeOTP, KeePassXC и аналоги.

Доступные режимы second_factor

Значение	Описание
`otp`	Только TOTP (по умолчанию)
`webauthn`	Только WebAuthn (аппаратные ключи, биометрия)
`on`	TOTP и WebAuthn; у каждого пользователя должно быть хотя бы одно MFA-устройство
`optional`	TOTP и WebAuthn доступны, но не обязательны
`off`	MFA отключена

> Для пользователей SSO MFA настраивается на стороне IdP. Airlock не запрашивает MFA при SSO-входе, хотя пользователи могут зарегистрировать устройства.

Включение TOTP

Динамическая конфигурация

Получите текущий ресурс `cluster_auth_preference`:

```
airctl get cap > cap.yaml
```

Установите `second_factor`:

```
kind: cluster_auth_preference
metadata:
  name: cluster-auth-preference
spec:
  type: local
  second_factor: "otp"
version: v2
```

Примените:

```
airctl create -f cap.yaml
```

Для обязательного MFA используйте `second_factor: "on"`. Для постепенного внедрения — `"optional"`.

Статическая конфигурация

Добавьте в `/etc/airlock.yaml`:

```
auth_service:
  authentication:
    type: local
    second_factor: otp
```

Перезапустите Auth Service после изменения.

Регистрация TOTP-устройства

Пользователь регистрирует TOTP через `airsh`:

```
airsh mfa add
# Choose device type [TOTP, WEBAUTHN]: totp
# Enter device name: phone
# Scan QR code with your TOTP app
# Enter an OTP token from the device: 123456
# MFA device "phone" added.
```

Список зарегистрированных устройств:

```
airsh mfa ls
```

Удаление устройства:

```
airsh mfa rm phone
```

Вход с TOTP

После регистрации устройства при входе запрашивается одноразовый код:

```
airsh login --proxy=airlock.example.com
# Enter password for Airlock user alice:
# Enter an OTP token from a registered device: 654321
```

Блокировка после неудачных попыток

Локальный пользователь блокируется на 20 минут при нескольких неудачных попытках входа или сброса пароля за 30 минут.

Разблокировка администратором:

```
airctl get users/alice > user.yaml
```

Установите `status.is_locked: false` и примените:

```
airctl create -f user.yaml
```

Следующие шаги

- Настройте WebAuthn для аппаратных ключей
- Ознакомьтесь с локальными пользователями и управлением ролями

WebAuthn (второй фактор)

Airlock поддерживает WebAuthn как второй фактор аутентификации. WebAuthn используется при входе в Airlock (`airsh login` или веб-интерфейс), а также при подключении к SSH-узлам и Kubernetes (`airsh ssh`, `kubectl`).

Поддерживаются аппаратные ключи (YubiKey, SoloKey) и биометрические аутентификаторы (Touch ID, Windows Hello).

Предварительные требования

- Аппаратный ключ WebAuthn или встроенный биометрический аутентификатор.
- Браузер с поддержкой WebAuthn.
- Утилита `airctl` для настройки кластера.

Шаг 1. Включение WebAuthn

Динамическая конфигурация

Отредактируйте ресурс `cluster_auth_preference`:

```
airctl edit cap
```

Пример конфигурации:

```
kind: cluster_auth_preference
version: v2
metadata:
  name: cluster-auth-preference
spec:
  type: local
  second_factor: "on"
  webauthn:
    rp_id: airlock.example.com
    attestation_allowed_cas:
    - "/path/to/allowed_ca.pem"
    attestation_denied_cas:
    - "/path/to/denied_ca.pem"
```

Статическая конфигурация

Фрагмент `/etc/airlock.yaml`:

```
auth_service:
  authentication:
    type: local
    second_factor: on
    webauthn:
      rp_id: airlock.example.com
      attestation_allowed_cas:
      - "/path/to/allowed_ca.pem"
      attestation_denied_cas:
      - "/path/to/denied_ca.pem"
```

Перезапустите Auth Service после изменения.

Поля webauthn

Поле	Описание
`rp_id`	Публичный домен Proxy Service (без `https://` и порта)
`attestation_allowed_cas`	Allow list CA для проверки устройств при регистрации
`attestation_denied_cas`	Deny list CA — устройства с этими CA отклоняются

По умолчанию все устройства разрешены. Для ограничения моделей используйте `attestation_denied_cas`.

> Начиная с Airlock v10, WebAuthn заменяет U2F. Существующие U2F-устройства поддерживаются автоматически.

Шаг 2. Регистрация устройства

Пользователь регистрирует WebAuthn-устройство:

```
airsh mfa add
# Choose device type [TOTP, WEBAUTHN]: webauthn
# Enter device name: desktop yubikey
# Tap any *registered* security key or enter a code from a *registered* OTP device:
# Tap your *new* security key
# MFA device "desktop yubikey" added.
```

Если при регистрации нужен только OTP как подтверждение (без WebAuthn):

```
airsh mfa add --mfa-mode=otp
```

Шаг 3. Вход через WebAuthn

После регистрации устройства при входе запрашивается подтверждение:

```
airsh login --proxy=airlock.example.com
# Enter password for Airlock user alice:
# Tap any security key or enter a code from a OTP device:
# > Profile URL:        https://airlock.example.com
#   Logged in as:       alice
#   Cluster:            airlock.example.com
#   Roles:              access, editor
#   Logins:             alice
#   Kubernetes:         enabled
#   Valid until:        2026-07-05 14:00:00 +0300 MSK [valid for 12h0m0s]
```

> WebAuthn при входе в Airlock требуется только для локальных пользователей. Для SSO настройте MFA на стороне IdP.

Миграция с U2F

Если ранее использовался U2F, Airlock автоматически выводит конфигурацию WebAuthn из U2F. Сохраните поле `u2f.app_id`, чтобы не перерегистрировать устройства:

```
auth_service:
  authentication:
    type: local
    second_factor: on
    u2f:
      app_id: https://airlock.example.com
    webauthn:
      rp_id: airlock.example.com
      attestation_allowed_cas:
      - "/path/to/u2f_attestation_ca.pem"
```

Следующие шаги

- Настройте OTP/TOTP как альтернативный второй фактор
- Ознакомьтесь с политиками доступа

Политики доступа (RBAC)

Airlock использует ролевую модель доступа (RBAC). Роль содержит два списка правил: `allow` и `deny`.

- По умолчанию доступ запрещён.
- Правила `deny` проверяются первыми и имеют приоритет.

Роли и другие динамические ресурсы управляются через веб-интерфейс Airlock, утилиту `airctl` или API.

Просмотр ролей

```
airsh login --user=admin --proxy=airlock.example.com
airctl get roles
```

Предустановленные роли

Роль	Описание
`access`	Доступ к ресурсам кластера
`editor`	Редактирование настроек конфигурации кластера
`auditor`	Чтение событий, аудит-логов и воспроизведение сессий
`requester`	Создание запросов на доступ (Access Requests)
`reviewer`	Рассмотрение и одобрение запросов на доступ

Версии ролей

Поддерживаются версии `v3` – `v7`. Версии `v4` и выше обратно совместимы с `v3`; отличие — значения по умолчанию для незаданных полей.

Метка	По умолчанию `v3`	По умолчанию `v4+`
`node_labels`	`[{"*": "*"}]` при наличии logins, иначе `[]`	`[]`
`app_labels`	`[{"*": "*"}]`	`[]`
`kubernetes_labels`	`[{"*": "*"}]`	`[]`
`db_labels`	`[{"*": "*"}]`	`[]`

Версия `v6` добавила поле `kubernetes_resources` для детального контроля ресурсов Kubernetes.

RBAC для инфраструктурных ресурсов

Роль определяет, к каким ресурсам (приложения, серверы, базы данных) пользователь имеет доступ. Механизм: метки на ресурсах + правила `allow` / `deny` в роли.

Пример: инфраструктура разделена метками `environment=production` и `environment=staging`. Роль стажёра может разрешать доступ только к `environment=staging`.

Пример роли с метками

```
kind: role
version: v5
metadata:
  name: example-role
spec:
  allow:
    node_labels:
      'env': 'stage'
  deny:
    node_labels:
      'workload': ['database', 'backup']
```

Несколько меток в одном правиле интерпретируются как логическое «И»:

```
    db_labels:
      'env': 'prod'
      'region': ['eu-central-1', 'eu-west-1']
```

Расширенный синтаксис меток

```
kind: role
version: v5
metadata:
  name: example-role
spec:
  allow:
    node_labels:
      'environment': 'test'
      '*': '*'
      'environment': ['test', 'staging']
      'environment': '^test|staging$'
```

Опции роли

Опции ограничивают параметры сессий. При нескольких ролях применяются комбинированные правила:

Опция	Описание	Поведение при нескольких ролях
`max_session_ttl`	Максимальный TTL SSH-сертификата	Минимальный TTL
`forward_agent`	SSH agent forwarding	Логическое «ИЛИ»
`port_forwarding`	TCP port forwarding	Логическое «ИЛИ»
`ssh_file_copy`	SCP/SFTP	Логическое «И»
`client_idle_timeout`	Завершение неактивных сессий	Минимальное значение
`require_session_mfa`	MFA на каждую сессию	Логическое «ИЛИ»
`pin_source_ip`	Привязка к IP источника	Логическое «ИЛИ»

Создание роли

```
airctl create -f my-role.yaml
airctl get roles --format text
```

Назначение роли пользователю:

```
airctl users update alice --set-roles=access,my-role
```

Разделы документации

- SSH-хосты — логины, метки узлов, опции SSH-сессий
- Kubernetes — кластеры, группы, пользователи и ресурсы Kubernetes
- Базы данных — метки БД, учётные записи и имена баз

Доступ к SSH-хостам

RBAC Airlock позволяет задавать детальные разрешения для аутентификации на Linux-серверах, подключённых к Airlock.

Пример политики: администраторы серверов имеют доступ ко всему, QA и инженеры — полный доступ к staging, инженеры могут получить временный доступ к production в экстренных случаях.

Конфигурация роли

Ресурс `role` предоставляет следующие средства ограничения доступа к серверам:

```
kind: role
version: v5
metadata:
  name: developer
spec:
  allow:
    logins: [ubuntu, debian, '{{internal.logins}}']
    node_labels:
      'env': 'test'
      '*': '*'
      'region': ['eu-central-1', 'eu-west-1']
      'reg': '^eu-central-1|eu-west-1$'
    host_groups: [ubuntu, nginx, other]
    host_sudoers:
    - 'ALL=(ALL) NOPASSWD: ALL'
  deny:
    logins:
    - root
```

⚠ Правила deny имеют приоритет. В примере выше попытка входа под `root` будет отклонена.

Шаблонные переменные

Поля роли поддерживают шаблонные переменные.

Переменные `{{external.xyz}}` заменяются значениями от SSO-провайдера. Для OIDC `xyz` — claim; для SAML — assertion.

Пример: назначение окружений и логинов из SAML-атрибутов:

```
spec:
  allow:
    node_labels:
    - env: '{{external.environments}}'
    logins:
    - '{{external.allowedlogins}}'
```

Переменная `{{internal.logins}}` применяется к локальным пользователям и trusted clusters. Локальный пользователь с trait `logins: jeff` получит логины `jeff` и `ubuntu`:

```
spec:
  allow:
    logins: ['{{internal.logins}}', ubuntu]
```

Опции роли для SSH

Блок `options` задаёт возможности Airlock для пользователей с ролью:

```
spec:
  options:
    create_host_user_mode: drop
    forward_agent: true
    port_forwarding: true
    ssh_file_copy: false
    client_idle_timeout: never
    disconnect_expired_cert: no
    max_sessions: 10
    enhanced_recording:
    - command
    - disk
    - network
    permit_x11_forwarding: true
    max_connections: 2
    record_session:
      default: best_effort
      ssh: best_effort
    require_session_mfa: true
    pin_source_ip: true
    cert_extensions:
     - type: ssh
       mode: extension
       name: login@example.com
       value: "{{ external.login }}"
```

Опция	Описание
`create_host_user_mode`	Автосоздание пользователя на хосте (`drop` , `keep` , `off`)
`forward_agent`	SSH agent forwarding
`port_forwarding`	TCP port forwarding
`ssh_file_copy`	SCP/SFTP (по умолчанию `true`)
`require_session_mfa`	Дополнительный MFA при начале сессии
`pin_source_ip`	Привязка сертификата к IP входа

Пример: разделение окружений

```
kind: role
version: v5
metadata:
  name: staging-access
spec:
  allow:
    logins: [ubuntu, deploy]
    node_labels:
      'env': 'staging'
  deny:
    node_labels:
      'env': 'prod'
```

Создание и назначение:

```
airctl create -f staging-access.yaml
airctl users update developer --set-roles=access,staging-access
```

Проверка доступа

Пользователь видит доступные серверы:

```
airsh ls
```

Подключение к серверу:

```
airsh ssh user@hostname
```

Доступ к Kubernetes

Kubernetes Service Airlock — прокси между пользователями Kubernetes и одним или несколькими кластерами.

При аутентификации пользователь получает kubeconfig для отправки запросов к авторизованным кластерам через Kubernetes Service. Сервис проверяет, изменяет или отклоняет запросы в зависимости от ролей Airlock.

Как это работает

1. Пользователь аутентифицируется в Airlock.
2. Kubernetes Service определяет права по ролям пользователя.
3. Для разрешённых запросов сервис переписывает заголовки для impersonation нужного Kubernetes-пользователя и групп.

4. Запрос передаётся API-серверу Kubernetes.

Поля роли для Kubernetes

```
kind: role
version: v6
metadata:
  name: kube-access
spec:
  allow:
    kubernetes_labels:
      'region': '*'
      'platform': 'production'
    kubernetes_resources:
      - kind: pod
        namespace: "production"
        name: "^webapp-[a-z0-9-]+$"
      - kind: pod
        namespace: "development"
        name: "*"
    kubernetes_groups:
    - developers
    kubernetes_users:
    - admin
  deny: {}
```

kubernetes_labels

Определяет, к каким зарегистрированным кластерам Kubernetes пользователь может подключаться. Метки задаются при регистрации агента:

```
--set labels.region=local --set labels.platform=production
```

kubernetes_resources

Поле версии `v6` и выше. Ограничивает доступ к конкретным ресурсам по kind, namespace и name. Имена поддерживают регулярные выражения (с `^` и `$`).

Версия роли	Значение по умолчанию `kubernetes_resources`
`v3`, `v4`, `v5`	Все pod во всех namespace
`v6`	Пустой список (нет доступа)
`v7`	Все ресурсы во всех namespace

kubernetes_groups и kubernetes_users

Определяют, от чьего имени пользователь действует в Kubernetes. Группы должны быть связаны с Kubernetes RBAC (ClusterRoleBinding) в целевом кластере.

Пример Kubernetes RBAC в кластере:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: pod-viewer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: pod-viewer
subjects:
- kind: Group
  name: developers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: pod-viewer
  apiGroup: rbac.authorization.k8s.io
```

Пример: ограничение по namespace

```
kind: role
version: v6
metadata:
  name: dev-kube
spec:
  allow:
    kubernetes_labels:
      'env': 'dev'
    kubernetes_resources:
      - kind: pod
        namespace: "development"
        name: "*"
      - kind: deployment
        namespace: "development"
        name: "*"
    kubernetes_groups:
    - developers
    kubernetes_users:
    - dev-user
  deny:
    kubernetes_resources:
      - kind: '*'
        namespace: 'production'
        name: '*'
```

Создание роли:

```
airctl create -f dev-kube.yaml
airctl users update developer --set-roles=access,dev-kube
```

Подключение к кластеру

Список доступных кластеров:

```
airsh kube ls
```

Аутентификация и обновление kubeconfig:

```
airsh kube login my-cluster
d8 k get pods --all-namespaces
```

Kubernetes Service фильтрует вывод — пользователь видит только ресурсы, разрешённые его ролью.

Проверка прав:

```
d8 k auth can-i create pods
```

Комбинирование Airlock RBAC и Kubernetes RBAC

Airlock RBAC определяет, к каким кластерам и ресурсам пользователь может обращаться. Kubernetes RBAC в целевом кластере определяет, что пользователь может делать с ресурсами (verbs: get, list, create и т.д.).

Рекомендуемый подход:

1. Создайте Kubernetes ClusterRole с нужными правами.
2. Привяжите его к группе, указанной в `kubernetes_groups` роли Airlock.
3. Ограничьте доступ Airlock через `kubernetes_labels` и `kubernetes_resources`.

Доступ к базам данных

RBAC позволяет администраторам задавать детальные политики доступа к базам данных, подключённым к Airlock.

Пример политики: администраторы БД имеют доступ ко всему, QA и инженеры — полный доступ к staging, инженеры могут получить временный доступ к production в экстренных случаях.

Конфигурация роли

Ресурс `role` предоставляет следующие средства ограничения доступа к базам данных:

```
kind: role
version: v5
metadata:
  name: developer
spec:
  allow:
    db_labels:
      environment: ["dev", "stage"]
    db_users: ["viewer", "editor"]
    db_names: ["main", "metrics", "postgres"]
```

Подстановочные знаки

Можно использовать `*` для любых имён пользователей и баз:

```
kind: role
version: v5
metadata:
  name: prod-reader
spec:
  allow:
    db_labels:
      environment: ["prod"]
    db_users: ["*"]
    db_names: ["*"]
  deny:
    db_users: ["postgres"]
    db_names: ["postgres"]
```

⚠ Правила deny имеют приоритет. Попытка подключения с учётной записью `postgres` или к базе `postgres` будет отклонена.

Имена баз данных

Поведение поля `db_names` зависит от СУБД.

PostgreSQL поддерживает несколько логических баз. При подключении `db_names` проверяется против имени целевой базы.

MySQL — термины «база» и «схема» синонимичны; права определяются grants на учётную запись. Поле `db_names` при подключении к MySQL не проверяется.

Шаблонные переменные

Поля `db_*` поддерживают шаблонные переменные.

`{{external.xyz}}` заменяется значениями от SSO: для OIDC — claim, для SAML — assertion.

Пример:

```
spec:
  allow:
    db_names: ["{{external.databases}}"]
```

Переменные `{{internal.db_users}}` и `{{internal.db_names}}` используются в trusted clusters для передачи разрешённых учётных записей и имён баз с root-кластера на leaf.

Root-кластер:

```
spec:
  allow:
    db_users: ["postgres"]
    db_names: ["postgres"]
```

Leaf-кластер:

```
spec:
  allow:
    db_users: ["{{internal.db_users}}"]
    db_names: ["{{internal.db_names}}"]
```

Опция create_db_user_mode

Автоматическое создание пользователя БД при сессии:

Значение	Описание
`off`	Автосоздание отключено
`keep`	Пользователь отключается в конце сессии
`best_effort_drop`	Попытка удалить пользователя; при неудаче — отключение

При нескольких ролях применяется логическое «ИЛИ» — если хотя бы одна роль разрешает, автосоздание включено.

```
spec:
  options:
    create_db_user_mode: keep
```

Пример: разделение окружений

```
kind: role
version: v5
metadata:
  name: db-staging
spec:
  allow:
    db_labels:
      environment: ["staging"]
    db_users: ["app_user", "readonly"]
    db_names: ["app", "analytics"]
  deny:
    db_labels:
      environment: ["prod"]
```

Создание и назначение:

```
airctl create -f db-staging.yaml
airctl users update developer --set-roles=access,db-staging
```

Подключение к базе данных

Список доступных баз:

```
airsh db ls
```

Подключение:

```
airsh db connect --db-user=readonly --db-name=app postgres-instance
```

Руководство пользователя

Руководство для конечных пользователей платформы Airlock.

- Утилита airsh — вход в кластер, SSH, Kubernetes, базы данных
- Запросы доступа — создание, согласование и использование временного доступа
- Веб-интерфейс — активные сессии, таймаут бездействия

Утилита airsh

В этом руководстве описано использование клиентской утилиты Airlock — `airsh`.

Вы узнаете, как:

- Войти в кластер и получить сертификат пользователя.
- Подключаться к серверам по SSH.
- Работать с кластерами Kubernetes через `kubectl`.
- Подключаться к базам данных.
- Копировать файлы, просматривать узлы и записи сессий.

Полный список команд доступен также через `airsh help` в терминале.

Быстрый старт

```
# Вход в кластер Airlock. Сертификат сохраняется в ~/.airsh/airlock.example.com
airsh login --proxy=airlock.example.com

# SSH-подключение к узлу
airsh ssh user@node

# airsh ssh принимает те же аргументы, что и OpenSSH:
airsh ssh -o ForwardAgent=yes user@node

# Можно создать символическую ссылку:
ln -s /path/to/airsh /path/to/ssh
ssh user@host

# Удаление сертификатов с локальной машины:
airsh logout
```

Airlock полностью совместим с существующими SSH-процессами: достаточно выполнять `airsh login` перед работой.

Установка airsh

Установите бинарный файл `airsh` той же мажорной версии, что и сервер Airlock в вашем кластере.

Чтобы узнать версию сервера:

- В веб-интерфейсе выберите имя пользователя в правом верхнем углу → **Help & Support**. Версия кластера указана в разделе **CLUSTER INFORMATION**.
- Через API:

```
curl https://airlock.example.com/webapi/find | jq '.server_version'
```

Идентификация пользователей

Идентичность пользователя в Airlock существует в рамках кластера. На узлах кластера могут быть разные пользователи ОС. Администратор Airlock назначает разрешённые логины каждой учётной записи.

При подключении к удалённому узлу нужно указать логин Airlock (флаг `--user`) и логин ОС в формате `login@host`:

```
# Аутентификация в кластере «work» как joe, вход на узел как root:
airsh ssh --proxy=work.example.com --user=joe root@node
```

Вход в кластер

Для получения сертификата выполните:

```
# Полная форма:
airsh login --proxy=proxy_host:<https_proxy_port>

# Порты по умолчанию:
airsh login --proxy=work.example.com

# Нестандартный HTTPS-порт:
airsh login --proxy=work.example.com:5000
```

Порт	Описание
https_proxy_port	HTTPS-порт прокси (по умолчанию `443` или `3080`)

Команда `login` сохраняет сертификат в `~/.airsh` и в ssh-agent, если он запущен. По умолчанию сертификаты действуют 12 часов.

> Выполняйте `airsh login` перед другими командами — тогда флаг `--proxy` можно не указывать: `airsh ssh user@host` будет работать.

Кластер может поддерживать несколько источников идентификации. Для выбора коннектора используйте флаг `--auth`:

```
# Локальный пользователь admin:
airsh --proxy=proxy.example.com --auth=local --user=admin login

# SSO через GitHub (коннектор «github»):
airsh --proxy=proxy.example.com --auth=github login
```

При внешней аутентификации `airsh` открывает браузер. Чтобы подавить это поведение:

```
airsh login --proxy=work.example.com --browser=none
```

Ссылка для входа будет выведена в терминал — скопируйте её в браузер.

Просмотр сертификата

```
airsh status

# > Profile URL:  https://proxy.example.com:3080
#   Logged in as: johndoe
#   Cluster:      proxy.example.com
#   Roles:        access, auditor, editor
#   Logins:       root, admin, guest
#   Kubernetes:   enabled
#   Valid until:  2017-04-25 15:02:30 -0700 PDT [valid for 1h0m0s]
#   Extensions:   permit-agent-forwarding, permit-port-forwarding, permit-pty
```

Интеграция с SSH-agent

Если запущен ssh-agent, `airsh login` сохраняет сертификат в агенте. Проверка:

```
ssh-add -L
```

Чтобы отключить интеграцию с агентом, передайте `--no-use-local-ssh-agent` или установите `AIRLOCK_USE_LOCAL_SSH_AGENT=false`.

Файлы идентичности

```
# Сохранить сертификат в файл joe.pem:
airsh login --proxy=proxy.example.com --out=joe

# Использовать файл для входа:
airsh ssh --proxy=proxy.example.com -i joe joe@db
```

Для совместимости с OpenSSH добавьте `--format=openssh` — будут созданы файлы `joe` и `joe-cert.pub`.

SSH-доступ

Просмотр узлов

```
airsh ls

# Node Name     Address            Labels
# ---------     -------            ------
# turing        ← Tunnel          os=linux
# graviton      10.1.0.7:3022      os=osx

# Фильтр по меткам:
airsh ls os=osx
```

Интерактивная оболочка

```
airsh ssh user@node
airsh ssh -p 6122 user@node ls
airsh ssh -o ForwardAgent=yes user@node
```

`airsh ssh` поддерживает флаги OpenSSH: `-p`, `-l`, `-L`, `-A` и другие.

Проброс портов

```
airsh ssh -L 5000:web.remote:80 node
```

Флаг `--local` выполняет локальную команду через туннель:

```
airsh ssh -L 5000:example.com:80 --local node curl http://localhost:5000
```

Jump host

```
airsh ssh -J proxy.example.com telenode
```

Поддерживается только один jump host; при `-J user@proxy` используется проброс портов вместо прокси Airlock.

Разрешение имён узлов

- По IP-адресу или DNS.
- По имени узла (`nodename` в конфигурации агента).
- По меткам: `airsh ssh os=osx` (если узел один).

Копирование файлов

```
airsh scp example.txt root@node:/path/to/dest
scp -P 61122 -r files root@node:/path/to/dest
```

Airlock поддерживает протоколы SCP и SFTP.

Совместное использование сессий

На удалённом сервере выполните:

```
airlock status
# Session ID : 7645d523-60cb-436d-b732-99c5df14b7c4
# Session URL: https://work:3080/web/sessions/7645d523-...
```

Другой пользователь может присоединиться:

```
airsh join <session_ID>
```

> Для присоединения к сессиям нужны специальные разрешения, настраиваемые администратором.

Доступ к Kubernetes

Список доступных кластеров:

```
airsh kube ls

# Kube Cluster Name Labels                      Selected
# ---------------- --------------------------- --------
# mycluster         env=dev
```

Вход в кластер:

```
airsh kube login mycluster
# Logged into kubernetes cluster "mycluster". Try 'kubectl version' to test the
connection.
```

После `airsh kube login` обновляется kubeconfig. Далее используйте `kubectl`:

```
kubectl get pods
```

Флаги `airsh kube login`:

Флаг	Описание
`--all`	Сгенерировать kubeconfig для всех доступных кластеров
`--as`	Kubernetes-пользователь для impersonation
`--as-groups`	Kubernetes-группа для impersonation
`--cluster`	Имя кластера Airlock (для листовых кластеров)
`-n, --kube-namespace`	Namespace по умолчанию

Доступ к базам данных

Список доступных баз данных:

```
airsh db ls

# Name                    Description  Allowed Users   Labels  Connect
# mysql-server1           ...          [alice]         env=dev airsh db connect mysql-
server1
```

Прямое подключение через airsh

Для MySQL/MariaDB нужен клиент `mysql`:

```
airsh db connect --db-user=alice --db-name=airlock_example mysql-server1
```

Локальный туннель

Для графических клиентов или других СУБД создайте локальный туннель:

```
airsh proxy db mysql-db --db-user=alice --tunnel
# Started authenticated tunnel for the MySQL database "mysql-db" on 127.0.0.1:49415.

# Подключение:
mysql --port 49415 --host localhost --protocol TCP
```

С TLS-маршрутизацией:

```
airsh proxy db --port 10700 mysql-db
# Started DB proxy on 127.0.0.1:10700
```

Записи сессий

```
airsh recordings ls

airsh play <session-id>
```

Конфигурация airsh

Файлы конфигурации:

- `/etc/airsh.yaml` — глобальная конфигурация (переопределяется через `AIRLOCK_GLOBAL_AIRSH_CONFIG`).
- `~/.airsh/config/config.yaml` — пользовательская конфигурация.

Пользовательская конфигурация имеет приоритет над глобальной.

Псевдонимы (aliases)

```
aliases:
    "l": "airsh login --auth=local"
    "status": "airsh status --format=json"
```

Дополнительные HTTP-заголовки

```
add_headers:
  - proxy: "*.example.com"
    headers:
      foo: bar
```

Доверенные кластеры

```
airsh --proxy=work clusters

# Cluster Name     Status
# staging          online
# production       offline

airsh --proxy=work ls --cluster=production
airsh --proxy=work ssh --cluster=production db-1
```

Запросы доступа

Запросы доступа (Access Requests) позволяют получить **временное** повышение привилегий — к дополнительным ролям или к конкретным ресурсам. Запрос проходит согласование: один или несколько рецензентов одобряют или отклоняют его.

В Airlock запросы доступа доступны через веб-интерфейс, CLI (`airsh`) и роли `requester` / `reviewer` . Поддерживаются запросы к ролям, к отдельным ресурсам (SSH, базы данных, Kubernetes, приложения) и согласование без участия администратора на сервере Auth Service.

Такой подход реализует принцип наименьших привилегий: постоянных администраторов с широкими правами становится меньше, а повышенный доступ выдаётся только на ограниченный срок и под конкретную задачу.

Airlock поддерживает два сценария:

Тип	Когда использовать
Запрос роли (Role Access Request)	Нужны дополнительные роли для системных задач — например, роль `dba` для миграции базы данных
Запрос ресурса (Resource Access Request)	Нужен доступ к конкретному серверу, базе, приложению или объекту Kubernetes без знания внутренней RBAC-модели

В одном запросе можно указать **либо** роли, **либо** ресурсы — смешивать оба типа нельзя.

Роли requester и reviewer

Для работы с запросами доступа администратор назначает пользователям предустановленные роли:

Роль	Назначение
`requester`	Создание запросов на доступ (Access Requests)
`reviewer`	Рассмотрение и одобрение или отклонение запросов на доступ

Если вашей учётной записи не хватает одной из этих ролей, обратитесь к администратору кластера.

> ⚠ Выдача роли, которая может редактировать другие роли, может позволить пользователю **постоянно** повысить свои привилегии. Перед одобрением внимательно проверяйте запрашиваемые роли и итоговые права.

Создание запроса через веб-интерфейс

Запрос роли

1. В левой панели выберите **Resources** → **Access Requests** → **New Request**.
2. В выпадающем списке выберите **Roles**.
3. Отметьте нужные роли и нажмите **ADD TO REQUEST**.
4. Нажмите **PROCEED TO REQUEST**, укажите причину запроса и отправьте его.

Запрос ресурса

1. Откройте **Resources** → **Access Requests** → **New Request**.
2. Выберите **Resources** вместо ролей.
3. Найдите нужный сервер, базу данных, приложение или объект Kubernetes.
4. Добавьте ресурсы в запрос, укажите причину и отправьте.

Статус запроса можно отслеживать на странице **Access Requests**.

Создание запроса через airsh

Запрос роли

```
# Вход в кластер
airsh login --proxy=airlock.example.com --user=alice

# Создание запроса на роль dba
airsh request create \
    --roles=dba \
    --reason="миграция базы данных сегодня вечером"
```

По умолчанию команда ожидает одобрения. Чтобы отправить запрос без ожидания, добавьте флаг `--nowait`.

Можно запросить роль сразу при входе:

```
airsh login --user=alice --request-roles=dba
# Seeking request approval... (id: bc8ca931-fec9-4b15-9a6f-20c13c5641a9)
```

После одобрения сертификат будет выдан с запрошенной ролью. Чтобы войти без ожидания, используйте `--request-nowait` — обычные роли выдаются сразу, а повышенный доступ активируется после одобрения.

Запрос ресурса

Сначала найдите ресурс:

```
airsh login --proxy=airlock.example.com --user=alice

# Поиск SSH-узлов (пустой список airsh ls — нормален без прямого доступа)
airsh request search --kind node

# Поиск с фильтром
airsh request search --kind node --search iot
```

Поддерживаемые типы ресурсов: `node`, `kube_cluster`, `db`, `app`, `windows_desktop`, а также объекты внутри кластеров Kubernetes (`pod`, `namespace`, `deployment` и другие).

Создайте запрос по идентификатору ресурса из вывода `airsh request search`:

```
airsh request create \
    --resource /airlock.example.com/node/b1168402-9340-421a-a344-af66a6675738 \
    --reason="реагирование на инцидент 123"
```

Команда выведет идентификатор запроса и будет ждать одобрения:

```
Creating request...
Request ID: f406f5d8-3c2a-428f-8547-a1d091a4ddab
Status:     PENDING

Waiting for request approval...
```

Автоматический запрос при отказе в доступе

Если настроены запросы к ресурсам, `airsh ssh` может создать запрос автоматически при отказе в доступе:

```
airsh ssh alice@iot
# ERROR: access denied to alice connecting to iot on cluster airlock.example.com
#
# You do not currently have access to alice@iot, attempting to request access.
#
# Enter request reason: нужен доступ для диагностики
# Creating request...
# Waiting for request approval...
```

После одобрения подключение продолжится без дополнительных действий.

Рассмотрение запросов

Через веб-интерфейс

Пользователи с ролью `reviewer` открывают **Management → Access Requests → Review Requests** и видят список ожидающих запросов. Для каждого запроса доступны одобрение или отклонение.

Через airsh

```
airsh login --proxy=airlock.example.com --user=bob

# Список запросов, доступных для рассмотрения
airsh request ls --reviewable

# Подробности запроса
airsh request show f406f5d8-3c2a-428f-8547-a1d091a4ddab

# Одобрение
airsh request review --approve f406f5d8-3c2a-428f-8547-a1d091a4ddab

# Отклонение
airsh request review --deny f406f5d8-3c2a-428f-8547-a1d091a4ddab
```

Администратор кластера при необходимости может рассматривать запросы через `airctl` на сервере Auth Service:

```
airctl requests ls
airctl request approve f406f5d8-3c2a-428f-8547-a1d091a4ddab
airctl request deny --reason="задача отменена" f406f5d8-3c2a-428f-8547-a1d091a4ddab
```

Использование одобренного запроса

Через airsh

После одобрения войдите с идентификатором запроса:

```
airsh login --request-id=f406f5d8-3c2a-428f-8547-a1d091a4ddab
```

Проверьте активный профиль:

```
airsh status
#   Active requests:    f406f5d8-3c2a-428f-8547-a1d091a4ddab
#   Roles:              access, requester
#   Allowed Resources:  ["/airlock.example.com/node/b1168402-..."]
```

При запросе **роли** права **добавляются** к текущим — пользователь сохраняет свои обычные роли плюс одобренные.

При запросе **ресурса** сертификат ограничен только одобренными ресурсами — доступ к остальным ресурсам временно блокируется.

Пример доступа к одобренному SSH-узлу:

```
airsh ls
airsh ssh alice@iot
```

Через веб-интерфейс

Откройте одобренный запрос на странице **Review Requests** и нажмите **ASSUME ROLES** (для запросов ролей) или подключитесь к ресурсу из раздела **Resources**.

Пока активен повышенный доступ, в верхней части страницы отображается баннер. Чтобы вернуться к обычным правам, нажмите **Switch Back**.

Завершение повышенного доступа

Когда временный доступ больше не нужен, откажитесь от активного запроса:

```
airsh request drop
```

Можно указать конкретный идентификатор:

```
airsh request drop f406f5d8-3c2a-428f-8547-a1d091a4ddab
```

В веб-интерфейсе используйте кнопку **Switch Back** в баннере активного запроса.

Просмотр своих запросов

```
# Все запросы текущего пользователя
airsh request ls --my-requests

# Детали конкретного запроса
airsh request show <request-id>
```

Kubernetes: поиск и запрос объектов

Поиск подов в кластере:

```
airsh request search --kind=pod --kube-cluster=mycluster --kube-namespace=default
```

Запрос доступа к поду:

```
airsh request create \
    --resource /airlock.example.com/pod/mycluster/development/nginx-1 \
    --reason="отладка приложения"
```

Для доступа ко всем подам в namespace можно использовать wildcard или регулярное выражение:

```
airsh request create \
    --resource /airlock.example.com/pod/mycluster/*/^nginx-[a-z0-9-]+$
```

Веб-интерфейс

Веб-интерфейс Airlock — это браузерный интерфейс для доступа к ресурсам, просмотра активных сессий и записей, управления запросами доступа, пользователями и ролями.

Присоединение к активной сессии

Веб-интерфейс позволяет просматривать и присоединяться к активным SSH-сессиям через терминал в браузере.

Активные SSH-сессии, к которым у вас есть доступ, отображаются на странице **Active Sessions** в левой панели навигации. Если вкладка **Active Sessions** отсутствует, вашей роли не хватает разрешения `list` для ресурса `ssh_session`. Обратитесь к администратору для настройки роли.

При нажатии **Join** нужно выбрать режим участника:

- **observer** — только просмотр вывода, без возможности ввода.
- **moderator** — просмотр и принудительное завершение или пауза сессии, без ввода.
- **peer** — совместная работа: просмотр и ввод.

Если кнопка присоединения отсутствует, у вас нет разрешений ни в одном из режимов.

Таймаут бездействия

После входа веб-интерфейс каждые 30 секунд проверяет активность сессии. Если более 10 минут не было взаимодействия с любой вкладкой браузера (клавиатура, мышь), выполняется выход.

Чтобы изменить таймаут по умолчанию (10 минут), администратор настраивает параметр `web_idle_timeout` в конфигурации Auth Service.

Динамическая конфигурация

```
airctl edit cluster_networking_config
```

Измените значение `spec.web_idle_timeout`:

```
kind: cluster_networking_config
metadata:
  ...
spec:
  ...
  web_idle_timeout: 10m0s
  ...
version: v2
```

После сохранения `airctl` обновит ресурс:

```
cluster networking configuration has been updated
```

Статическая конфигурация

Обновите `/etc/airlock.yaml` в секции `auth_service` и перезапустите демон `airlock`:

```
auth_service:
  web_idle_timeout: 10m0s
```

Разделы веб-интерфейса

Раздел	Назначение
Resources	Список доступных серверов, баз данных, кластеров Kubernetes
Active Sessions	Активные SSH-сессии и присоединение к ним
Session Recordings	Записанные сессии SSH и Kubernetes
Access Requests	Создание и рассмотрение запросов доступа (см. Запросы доступа)
Management	Управление пользователями, ролями и настройками (для администраторов)

Kubernetes через веб-интерфейс

На вкладке **Kubernetes** отображаются кластеры, к которым у вас есть доступ. Нажмите **Connect** — откроется окно с командами для подключения через `airsh kube login` в терминале.

Базы данных через веб-интерфейс

Веб-интерфейс не устанавливает прямое подключение к базам данных, но показывает доступные СУБД на вкладке **Databases** и предоставляет команды `airsh` для подключения из локального терминала.

Справочник

Справочная информация по утилитам командной строки.

- Справочник airsh — команды клиента для пользователей
- Справочник airctl — команды администратора

Справочник airsh

`airsh` — CLI-клиент для пользователей Airlock: вход в кластер, SSH, Kubernetes, базы данных, копирование файлов.

Полный справочник: `airsh help <команда>`.

Переменные окружения

Переменная	Описание	Пример
`AIRLOCK_AUTH`	Имя коннектора аутентификации (SAML, OIDC, GitHub, local)	github
`AIRLOCK_CLUSTER`	Имя корневого или листового кластера	cluster.example.com
`AIRLOCK_LOGIN`	Логин ОС по умолчанию на удалённом хосте	root
`AIRLOCK_LOGIN_BROWSER`	`none` — не открывать браузер при SSO	none
`AIRLOCK_PROXY`	Адрес Proxy Service	cluster.example.com:3080
`AIRLOCK_HOME`	Каталог конфигурации airsh	~/.airsh
`AIRLOCK_USER`	Имя пользователя Airlock	alice
`AIRLOCK_USE_LOCAL_SSH_AGENT`	Интеграция с ssh-agent	true, false
`AIRLOCK_GLOBAL_AIRSH_CONFIG`	Путь к глобальному конфигу	/etc/airsh.yaml
`AIRLOCK_IDENTITY_FILE`	Путь к файлу идентичности	/opt/identity

Глобальные флаги

Флаг	Описание
`-l, --login`	Логин ОС на удалённом хосте
`--proxy`	Адрес Proxy Service: `host:https_port[,ssh_proxy_port]`
`--user`	Имя пользователя Airlock
`--ttl`	Срок действия сертификата в минутах (по умолчанию 720)
`-i, --identity`	Файл идентичности
`--auth`	Коннектор аутентификации (по умолчанию `local`)
`--insecure`	Не проверять сертификат сервера (только для тестов)
`-d, --debug`	Подробный вывод
`-J, --jumphost`	SSH jump host

airsh login

Вход в кластер. Сертификат сохраняется в `~/.airsh`.

```
airsh login [<flags>] [<cluster>]
```

Флаг	Описание
`-o, --out`	Сохранить идентичность в файл
`--format`	`file`, `openssh`, `kubernetes`
`--browser`	`none` — не открывать браузер
`--no-use-local-ssh-agent`	Не загружать сертификат в ssh-agent

```
airsh --proxy=proxy.example.com login
airsh --proxy=proxy.example.com --auth=local --user=admin login
airsh --proxy=proxy.example.com --auth=github login
airsh --ttl=1 login
airsh login --format=kubernetes -o kubeconfig
```

airsh logout

Удаляет сертификат клиента.

```
airsh logout
```

airsh status

Показывает текущий профиль, роли, логины и срок действия сертификата.

```
airsh status
```

airsh ls

Список узлов кластера.

```
airsh ls [<flags>] [<label>]
```

Флаг	Описание
`-v, --verbose`	Показать Node ID
`--cluster`	Листовой кластер

```
airsh ls
airsh ls os=linux
airsh ls -v
```

airsh ssh

SSH-подключение или выполнение команды.

```
airsh ssh [<flags>] <[user@]host> [<command>...]
```

Флаг	Описание
`-p, --port`	SSH-порт
`-A, --forward-agent`	Проброс агента
`-L, --forward`	Локальный проброс портов
`-D, --dynamic-forward`	SOCKS5-прокси
`--cluster`	Листовой кластер
`-X, --x11-untrusted`	X11 forwarding
`-Y, --x11-trusted`	Доверенный X11 forwarding

```
airsh ssh root@grav-00
airsh ssh -o ForwardAgent=yes root@grav-00
airsh ssh root@env=prod hostname
```

airsh scp

Копирование файлов.

```
airsh scp [<flags>] <source>... <dest>
```

Флаг	Описание
`-r, --recursive`	Рекурсивное копирование
`-P, --port`	Порт
`--cluster`	Листовой кластер

airsh join

Присоединение к активной сессии.

```
airsh join [<flags>] <session-id>
```

airsh clusters

Список доверенных кластеров.

```
airsh clusters [--quiet]
```

airsh kube ls

Список доступных кластеров Kubernetes.

```
airsh kube ls
```

airsh kube login

Вход в кластер Kubernetes, обновление kubeconfig.

```
airsh kube login <kube-cluster>
```

Флаг	Описание
`--all`	Kubeconfig для всех доступных кластеров
`--as`	Impersonation пользователя Kubernetes
`--as-groups`	Impersonation группы Kubernetes
`--cluster`	Кластер Airlock
`-n, --kube-namespace`	Namespace по умолчанию

airsh db ls

Список доступных баз данных.

```
airsh db ls
```

airsh db connect

Подключение к базе данных через встроенный клиент.

```
airsh db connect --db-user=<user> --db-name=<db> <database>
```

airsh proxy db

Локальный прокси для подключения к базе данных.

```
airsh proxy db [<flags>] <db>
```

Флаг	Описание
`--db-user`	Пользователь БД
`--db-name`	Имя базы данных
`--port`	Локальный порт
`--tunnel`	Аутентифицированный туннель без дополнительных учётных данных
`--cluster`	Листовой кластер

```
airsh proxy db mysql-db
airsh proxy db --port 10700 mysql-db
airsh proxy db --tunnel mysql-db
```

airsh recordings ls

Список записанных сессий.

```
airsh recordings ls [--last=6h0m0s] [--from-utc=YYYY-MM-DD] [--to-utc=YYYY-MM-DD]
```

airsh play

Воспроизведение записанной сессии.

```
airsh play <session-id>
```

airsh config

Вывод конфигурации OpenSSH для использования с Airlock.

```
airsh config >> ~/.ssh/config
```

airsh proxy ssh

Локальный TLS-прокси для SSH (режим TLS-маршрутизации).

```
airsh proxy ssh [<flags>] <[user@]host>
```

airsh version

Версия клиента и сервера.

```
airsh version [--format=json] [--client]
```

airsh mfa add / ls / rm

Управление устройствами MFA.

```
airsh mfa add
airsh mfa ls
airsh mfa rm <type>
```

airsh request create / ls / review / show / drop / search

Управление запросами доступа (Access Requests).

```
airsh request create --roles=prod --reason="..."
airsh request ls [--my-requests] [--reviewable]
airsh request review --approve <request-id>
airsh request show <request-id>
airsh request drop [<request-id>...]
airsh request search --kind=node --labels=env=dev
```

Справочник airctl

`airctl` — CLI-инструмент администратора для управления ресурсами кластера: пользователями, узлами, токенами, сертификатами и конфигурацией.

Аутентификация airctl

Перед выполнением команд администратор должен аутентифицироваться в кластере.

С файлом идентичности на удалённом хосте

```
airctl --identity=identity.pem --auth-server=proxy.example.com:443 status
```

Файл идентичности создаётся через `airctl auth sign` или `airsh login --out=<path>`.

На том же хосте, что и Auth Service

Если на хосте есть `/etc/airlock.yaml`, `airctl` аутентифицируется к Auth Service из конфигурации.

> При локальном запуске на Auth Service в журнале аудита действие будет отображаться как выполненное самим Auth Service. Рекомендуется ограничить прямой SSH-доступ к Auth Service.

После airsh login на удалённом хосте

Если нет локального конфига и файла идентичности, `airctl` использует профиль `airsh`, созданный командой `airsh login`. Убедитесь, что `AIRLOCK_CONFIG_FILE` пуст или `/etc/airlock.yaml` отсутствует.

Глобальные флаги

Флаг	Описание
`-d, --debug`	Подробный вывод
`-c, --config`	Путь к конфигурации (по умолчанию `/etc/airlock.yaml`)
`--auth-server`	Адрес Auth/Proxy Service
`-i, --identity`	Файл идентичности
`--insecure`	Не проверять TLS-сертификат (опасно)

airctl status

Статус кластера.

```
airctl status
airctl status --auth-server=192.168.99.102:3025 --identity=identity.pem
```

airctl users add

Создание пользователя и токена приглашения.

```
airctl users add [<flags>] <account>
```

Флаг	Описание
`--roles`	Роли (через запятую)
`--logins`	Разрешённые SSH-логины
`--kubernetes-groups`	Группы Kubernetes
`--kubernetes-users`	Пользователи Kubernetes
`--db-users`	Пользователи БД
`--db-names`	Имена баз данных
`--ttl`	Срок действия токена (макс. 48ч)

```
airctl users add joe --roles=access,editor --logins=joe,ubuntu
```

airctl users ls

Список пользователей.

```
airctl users ls
```

airctl users rm

Удаление пользователей.

```
airctl users rm <logins>
airctl users rm sally,tim
```

airctl users reset

Сброс пароля локального пользователя.

```
airctl users reset <account> [--ttl=8h]
```

airctl users update

Обновление учётной записи.

```
airctl users update [<flags>] <account>
```

airctl nodes add

Генерация токена для присоединения узла.

```
airctl nodes add [<flags>]
```

Флаг	Описание
`--roles`	`proxy`, `auth`, `node`, `db`, `app`, `windowsdesktop`
`--ttl`	Срок действия токена (по умолчанию 30m)
`--token`	Пользовательское значение токена

```
airctl nodes add
airctl nodes add --roles=node,proxy --ttl=1h
```

airctl nodes ls

Список активных SSH-узлов.

```
airctl nodes ls [--namespace=<ns>]
```

airctl tokens add

Создание токена приглашения.

```
airctl tokens add --type=TYPE [<flags>]
```

Флаг	Описание
`--type`	`proxy`, `auth`, `trusted_cluster`, `node`, `db`, `kube`, `app`, `windowsdesktop`
`--ttl`	Срок действия (по умолчанию 1h)
`--labels`	Метки токена
`--value`	Значение токена

```
airctl tokens add --type=node
airctl tokens add --type=trusted_cluster --ttl=5m
airctl tokens add --type=kube
airctl tokens add --type=node,db
```

airctl tokens ls

Список токенов.

```
airctl tokens ls [--format=json]
```

airctl tokens rm

Удаление (отзыв) токена.

```
airctl tokens rm <token>
```

airctl get

Получение ресурсов в YAML/JSON.

```
airctl get [<flags>] <resource-type/resource-name>,...
```

Флаг	Описание
`--format`	`yaml`, `json`, `text`
`--with-secrets`	Включить секреты

```
airctl get users
airctl get user/joe > joe.yaml
airctl get cluster/east
airctl get clusters,users
airctl get all > state.yaml
```

airctl create

Создание ресурса из файла.

```
airctl create -f <file.yaml>
```

airctl edit

Редактирование ресурса в текстовом редакторе.

```
airctl edit <resource-type/resource-name>
```

Редактор выбирается по приоритету: `AIRLOCK_EDITOR` → `VISUAL` → `EDITOR` → `vi`.

```
airctl edit role/sre
AIRLOCK_EDITOR=nano airctl edit user/alice
```

airctl rm

Удаление ресурса.

```
airctl rm <resource-type/resource-name>
```

```
airctl rm users/admin
airctl rm role/old-role
```

airctl auth export

Экспорт публичных CA-сертификатов кластера.

```
airctl auth export [--type=tls-host] [--fingerprint=SHA256:...]
```

Флаг	Описание
`--type`	`user`, `host`, `tls-host`, `tls-user`, `db`, `openssh`, `saml-idp`
`--fingerprint`	Фильтр по отпечатку
`--keys`	Экспорт закрытых ключей

airctl auth sign

Создание файла идентичности для пользователя или хоста.

```
airctl auth sign -o <filepath> [--user <user> | --host <host>] [<flags>]
```

Флаг	Описание
`--user`	Имя пользователя Airlock
`--host`	Имя хоста Airlock
`-o, --out`	Путь к файлу
`--format`	`file`, `openssh`, `tls`, `kubernetes`
`--ttl`	Срок действия (по умолчанию 12h)

```
airctl auth sign --ttl=24h --user=jenkins --out=jenkins.pem
airctl auth sign --format=openssh --host grav-00
airctl auth sign --user=admin --out=identity.pem
```

airctl auth rotate

Ротация центров сертификации.

```
airctl auth rotate [--type=user|host] [--grace-period=200h]
```

airctl request create / approve / deny / ls / rm

Управление запросами доступа.

```
airctl request create myuser --roles=prod --reason="Fix an outage"
airctl request approve request-id-1,request-id-2
airctl request deny request-id-1
airctl request ls
airctl request rm request-id-1
```

airctl alerts create / list / ack

Управление оповещениями кластера.

```
airctl alerts create --severity=LOW "System under maintenance"
airctl alerts list
airctl alerts ack --reason="scheduled" upgrade-suggestion
airctl alerts ack --clear upgrade-suggestion
```

airctl top

Диагностическая информация (требуется `airlock start --diag-addr=<bind-addr>`).

```
airctl top http://127.0.0.1:3000 5s
```

airctl help

```
airctl help
airctl help users add
```