AXIOM JDK

Axiom Linux
Руководство по работе с репозиторием и контейнерами

Axiom Linux | Март 2026

Программное обеспечение АКСИОМ содержит программное обеспечение с открытым исходным кодом. Дополнительная информация о коде сторонних разработчиков доступна на сайте https://axiomjdk.ru/third_party_licenses. Для дополнительной информации о том, как получить копию исходного кода, можно обратиться по адресу info@axiomjdk.ru.

ДАННАЯ ИНФОРМАЦИЯ МОЖЕТ ИЗМЕНЯТЬСЯ БЕЗ ПРЕДВАРИТЕЛЬНОГО УВЕДОМЛЕНИЯ. АКСИОМ ПРЕДОСТАВЛЯЕТ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ "КАК ЕСТЬ" БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, АКСИОМ ПРЯМО ОТКАЗЫВАЕТСЯ ОТ ВСЕХ ПОДРАЗУМЕВАЕМЫХ ГАРАНТИЙ, ВКЛЮЧАЯ, НО НЕ ОГРАНИЧИВАЯСЬ ПОДРАЗУМЕВАЕМЫМИ ГАРАНТИЯМИ ТОВАРНОЙ ПРИГОДНОСТИ И ПРИГОДНОСТИ ДЛЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ.

АКСИОМ НИ ПРИ КАКИХ ОБСТОЯТЕЛЬСТВАХ НЕ НЕСЕТ ОТВЕТСТВЕННОСТИ ЗА ЛЮБЫЕ КОСВЕННЫЕ, СЛУЧАЙНЫЕ, СПЕЦИАЛЬНЫЕ, ШТРАФНЫЕ ИЛИ КОСВЕННЫЕ УБЫТКИ, ИЛИ УБЫТКИ ОТ ПОТЕРИ ПРИБЫЛИ, ДОХОДА, ДАННЫХ ИЛИ ИСПОЛЬЗОВАНИЯ ДАННЫХ, ПОНЕСЕННЫЕ ВАМИ ИЛИ ЛЮБОЙ ТРЕТЬЕЙ СТОРОНОЙ, БУДЬ ТО В РЕЗУЛЬТАТЕ ДЕЙСТВИЯ ДОГОВОРА ИЛИ ДЕЛИКТА, ДАЖЕ ЕСЛИ АКСИОМ БЫЛО ПРЕДУПРЕЖДЕНО О ВОЗМОЖНОСТИ ТАКИХ УБЫТКОВ.

Использование любого программного продукта АКСИОМ регулируется соответствующим лицензионным соглашением, которое никоим образом не изменяется условиями данного уведомления. Программные продукты и фирменные наименования: Axiom JDK, Axiom JDK Pro, Axiom Runtime Container Pro, Axiom Linux, Libercat, Libercat Certified и АКСИОМ принадлежат АКСИОМ и их использование допускается только с разрешения правообладателя.

Товарный знак Linux® используется в соответствии с сублицензией от Linux Foundation, эксклюзивного лицензиата Линуса Торвальдса, владельца знака на всемирной основе. Java и OpenJDK являются товарными знаками или зарегистрированными товарными знаками компании Oracle и/или ее аффилированных лиц. Другие торговые марки являются собственностью их соответствующих владельцев и используются только в целях идентификации.

Содержание

1. Введение 5

Docker 5

Podman 5

Axiom JDK 5

Axiom Linux 5

2. Образы 7

Доступ к реестру 7

Авторизация для работы с репозиторием apk 8

Создание контейнера и запуск 8

Перечень образов 9

- axiom-linux-base 9
- axiom-runtime-container-pro 10
- axiom-linux-python 11
- axiom-linux-gcc 11
- axiom-native-image-kit-container 11
- axiom-linux-node 12
- axiom-linux-go 12

Создание модифицированных образов на базе Axiom Linux 13

Настройка Podman для запуска контейнеров. 13

1. Введение

Настоящий документ содержит описание работы с контейнерами и инструментами Docker.

Docker

Docker — это платформа контейнеризации для создания, совместного использования и запуска ваших приложений.

Docker позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть развёрнут на любой системе, а также предоставляет набор команд для управления этими контейнерами. Приложения Docker можно использовать в любой системе: ноутбуке, настольных системах, серверах или в облаке. Контейнер — это стандартная единица программного обеспечения, которая упаковывает код и все его зависимости, чтобы приложение быстро и надежно переносилось из одной вычислительной среды в другую. Docker предоставляет набор инструментов для создания и совместного использования образов контейнеров, а также запуска контейнеров в любом масштабе.

Для получения дополнительной информации посетите сайт Docker.

Podman

Podman — это инструмент с открытым исходным кодом для поиска, сборки, передачи и запуска приложений. Является утилитой командной строки с аналогичными docker командами, однако не требует дополнительный сервис для работы и может работать без прав доступа root.

Для получения дополнительной информации посетите сайт Podman.

Axiom JDK

Среда разработки и исполнения JAVA™ Axiom JDK — это Java SE дистрибутив с экспертной технической поддержкой, прошедший проверку на соответствие спецификациям Java SE. Axiom JDK — отечественная альтернатива инструментальным средствам разработки приложений на Java™ для коммерческих и государственных компаний.

Axiom Linux

Axiom Linux — это современная полнофункциональная операционная система. Данную ОС

характеризуют следующие отличительные свойства.

- Надежная многозадачная многопользовательская ОС для серверных задач и хостинга контейнерных нагрузок, в том числе на базе Java;
- Предлагает варианты установки для использования libc библиотек musl или glibc;
- Осуществляет эффективное управление памятью и предоставляет дополнительные реализации malloc, которые могут быть полезны для различных типов нагрузок;
- Поддерживает различные файловые системы во время установки ОС и для настройки дополнительных хранилищ;
- Предоставляет сетевые возможности.

2. Образы

Вы можете получить прямую ссылку на образ или использовать образы размещенные в библиотеке/репозитории образов контейнеров в реестре cr.yandex. Если вы используете реестр образов cr.yandex, то путь к репозиторию можно увидеть и скопировать в личном кабинете (ЛК) на портале поддержки под заголовком "Поддержка" в строке **Путь к репозиторию для скачивания образов**.

Для получения дополнительной информации см. документ "Инструкция по работе с порталом поддержки".

В репозиториях представлены образы с поддержкой архитектур aarch64, x86_64.

> **Примечание:**
>
> В данной главе, в качестве примеров, описывается работа с образами контейнеров в инструменте Docker. Работа в Podman делается аналогично. Для настройки Axiom Linux для работы с Podman см. главу Настройка Podman для запуска контейнеров.

Доступ к реестру

В случае получения прямой ссылки на образ, используйте эту ссылку в командах для скачивания и запуска образа.

Для использования реестра образов cr.yandex на каждом устройстве, с которого будет выполняться скачивание образов, требуется настроить аутентификацию в Docker клиенте.

Для настройки аутентификации скопируйте полученный json файл на устройство и выполните команду:

```
cat <имя файла>.json | docker login --username json_key --password-stdin
cr.yandex
```

После чего будет доступно скачивание и запуск контейнеров.

Получить список тегов (образов) в репозитории можно программно, например, с помощью утилиты skopeo версии 1.7.0 или новее как на примере ниже:

```
skopeo list-tags docker://<URL адрес репозитория>/<имя репозитория>
```

Отключить аутентификацию в реестре cr.yandex можно командой:

```
docker logout cr.yandex
```

Авторизация для работы с репозиторием apk

Для доступа к репозиторию с помощью которого вы можете установить новое программное обеспечение и обновить текущее из Axiom Runtime Container Pro, необходимо включить HTTP аутентификацию. То есть, перед выполнением команды `apk add` или `apk update` надо будет настроить переменную окружения HTTP_AUTH. В HTTP_AUTH должно быть записано имя пользователя - это имя проекта и пароль - это токен.

```
export HTTP_AUTH="basic:*:<имя проекта>:<токен полученный в ЛК>"
```

Имя проекта можно увидеть и скопировать в личном кабинете (ЛК) на портале поддержки под заголовком "Поддержка" в строке **Имя проекта**. Токен необходимо создать в том же личном кабинете на портале поддержки. Инструкция как получить токен доступна в документе "Инструкция по работе с порталом поддержки" в разделе "Аутентификация" главы "Автоматизация загрузок с партнерским API".

Вы также можете добавить информацию для авторизации непосредственно в URL репозитория в файле `/etc/apk/repositories`.

Например, если в `/etc/apk/repositories` содержаться следующие пути к репозиторию:

```
https://pkg.axiomjdk.ru/axiom/musl/23/core
https://pkg.axiomjdk.ru/axiom/musl/23/universe
```

После изменения, строки должны выглядеть следующим образом:

```
https://<имя проекта>:<токен>@pkg.axiomjdk.ru/axiom/musl/23/core
https://<имя проекта>:<токен>@pkg.axiomjdk.ru/axiom/musl/23/universe
```

Создание контейнера и запуск

Чтобы создать контейнер из образа с тегом `<тег образа>` из репозитория `<имя репозитория>`, выполните одну из следующих команд.

Скачайте образ используя команду:

```
docker image pull <URL адрес репозитория>/<имя репозитория>:<тег образа>
```

и запустите контейнер с помощью команды docker run:

```
docker container run --rm -it <URL адрес репозитория>/<имя репозитория>:<тег образа>
```

Вы также можете объединить две предыдущие операции в одну команду docker run:

```
docker run --rm -it <URL адрес репозитория>/<имя репозитория>:<тег образа>
```

Если часть `:<тег образа>` опущена, она заменяется на `:latest`, то есть команда работает с тегом `latest`.

Например, следующие команды скачивают и запускают контейнеры с Axiom Linux на musl libc.

```
docker pull <URL адрес репозитория>/axiom-runtime-container-pro:jre-17-musl
docker run --rm <URL адрес репозитория>/axiom-runtime-container-pro:jre-17-musl java -version
```

Для запуска какого-либо приложения вы можете создать Dockerfile на основе образа `axiom-runtime-container-pro` или смонтировать том с вашим кодом/приложением, например:

```
docker run -it --rm -v /home/user/project/:/data <URL адрес репозитория>/<имя репозитория>:<тег образа> java -jar /data/MyApp.jar
```

Описание доступных образов представлено ниже.

Перечень образов

axiom-linux-base

Репозиторий содержит базовые образы Axiom Linux с минимальным количеством установленных пакетов.

Ниже приведены доступные теги для образов на основе версий Axiom Linux musl или glibc:

- musl
- glibc

axiom-runtime-container-pro

Репозиторий содержит образы с Axiom JDK.

Для Java 8 представлены JRE и JDK образы с Axiom JDK. Для Java 11 и более новых версий представлены JRE и JDK образы с Axiom JDK Lite.

Теги доступных образов определяются следующим шаблоном.

`<jdk_type>-<java_version>[-crac][-cds][-slim]-<libc>`

где:

- `<jdk_type>` определяет тип поставляемых Java пакетов:
  - jre - среда выполнения Java-приложений
  - jdk - jre плюс инструменты для разработки и отладки Java-приложений
  - jdk-all - jdk плюс Java Modules, Client и Minimal виртуальные машины Java
- `<java_version>` указывает версию Java. Допустимые варианты:
  - Основная версия, например 11, 17 или 21
  - Версия выпуска, например 11.0.20
  - Полная версия (с номером сборки), например 11.0.20_9, что соответствует версии 11.0.20+9
- `<libc>` задает версию C библиотеки и может принимать значения:
  - glibc
  - musl
- `crac` в теге означает, что JDK в образе поддерживает Coordinated Restore at Checkpoint (CRAC).
- `cds` присутствует в тегах образов с Class Data Sharing (для ускорения первого запуска JVM).
- Образы с `slim` меньше по размеру, тк не включают менеджер пакетов ОС.

Например, для обычного JRE образа с последней версией Java 17 на базе glibc находящегося в репозитории cr.yandex полный путь будет:

```
<URL адрес репозитория>/axiom-runtime-container-pro:jre-17-glibc
```

а для slim образа с JDK Java 11 на базе musl полный путь будет:

```
<URL адрес репозитория>/axiom-runtime-container-pro:jdk-11-slim-musl
```

axiom-linux-python

Репозиторий содержит образы с Python 3.11 и базовыми Python утилитами (pip, setuptools, wheel).

Доступные образы:

- 3.11-glibc
- 3.11-musl

axiom-linux-gcc

Репозиторий содержит образы с инструментами для C/C++ разработки на базе GCC 12.2.

Доступные образы:

- 12.2-glibc
- 12.2-musl

axiom-native-image-kit-container

Репозиторий содержит образы с Axiom NIK 23, 24 и 25.

Теги доступных образов определяются следующим шаблоном.

```
jdk-<java_version>-nik-<nik_version>[-debug]-<libc>
```

где:

- `<nik_version>` и `<java_version>` определяют версию Axiom NIK и Java. Допустимые варианты:
  - Axiom NIK 23 представлен в вариантах с Java 17 и 21
  - Axiom NIK 24 представлен в варианте с Java 24
  - Axiom NIK 25 представлен в варианте с Java 25
- `<libc>` задает версию C библиотеки и может принимать значения:

    - glibc
    - musl
- `debug` в теге означает, что в образ добавлены Apache Maven и отладчик GDB.

Примеры:

- jdk-17-nik-23-musl
- jdk-25-nik-25-debug-glibc

axiom-linux-node

Репозиторий содержит образы с Node.js и базовыми утилитами (npm):

Доступные образы:

- X-glibc, X.Y-glibc, X.Y.Z-glibc - указанная версия Node.js с Glibc
- X-musl, X.Y-musl, X.Y.Z-musl - указанная версия Node.js с Musl
- glibc - указывает на последнюю версию Node.js с Glibc
- musl, latest - указывает на последнюю версию Node.js с Musl

axiom-linux-go

Репозиторий содержит образы с установленным компилятором Go и вспомогательными утилитами и библиотеками. Образы предназначены для сборки и отладки программ написанных на Go. Для каждого типа libc есть свои теги:

- 1.X-glibc, 1.X.Y-glibc - указанная версия Go с Glibc
- 1.X-musl, 1.X.Y-musl - указанная версия Go с Musl
- glibc - указывает на последнюю версию Go с Glibc
- musl, latest - указывает на последнюю версию Go с Musl

Например:

- 1.23.9-glibc
- 1.23-musl
- glibc

- musl

Создание модифицированных образов на базе Axiom Linux

Для минимизации риска возможных уязвимостей и размера в предоставляемых образах поставляется минимальный набор компонентов.

В случае, если для работы вашего приложения требуются дополнительные компоненты, рекомендуется создание своих дочерних образов, в которых доустанавливаются эти компоненты.

Как описано в разделе Авторизация для работы с репозиторием apk, необходимо установить переменную `HTTP_AUTH` перед любыми операциями с `apk` репозиториями. Ниже представлен пример как это можно сделать в Dockerfile.

К примеру, вы работаете с образом `axiom-runtime-container-pro:jdk-21-glibc`, но для вашего приложения также необходима утилита `curl`, которая в этом образе отсутствует.

1. В первую очередь, создайте Dockerfile:

```
FROM <URL репозитория>/axiom-runtime-container-pro:jdk-21-glibc

RUN --mount=type=secret,id=http_auth \
    export HTTP_AUTH=$(cat "/run/secrets/http_auth"); \
    apk add curl
```

2. Затем необходимо создать файл `http_auth`, в который записывается содержимое переменной `HTTP_AUTH`:

```
basic:*:<имя проекта>:<токен полученный в ЛК>
```

3. Далее запускаете процедуру сборки, указывая путь к этому файлу:

```
docker build --secret id=http_auth,src=http_auth .
```

Таким образом, вы создали новый образ, и информация об используемом токене не будет доступна для пользователей этого образа.

Настройка Podman для запуска контейнеров

Axiom Linux может выступать в роли хост системы для запуска контейнеров. При этом в поставку включены инструменты Docker и Podman, которые можно поставить из репозитория пакетов. Настройка среды отличается при использовании различных инструментов. Ниже

рассматривается настройка инструмента Podman.

Настройка включает в себя следующие шаги:

1. Выполните следующую команду `sudo apk add fuse-overlay podman`.
2. Задайте значения `/etc/subuid` и `/etc/subgid`, чтобы указать диапазон UID для пользователей и позволить Podman запускать контейнеры в соответствующем пользовательском пространстве имен.
3. Так как в Axiom Linux не используется `systemd`, то может понадобиться выполнить команду `sudo mount –make-rshared /`.
4. Настройте cgroups и выполните команду `sudo rc-update add cgroups; sudo rc-service cgroups start`.
5. Если требуется включение cgroups V2, тогда отредактируйте `/etc/rc.conf` и включите параметр “unified” для “rc_cgroup_mode”.
6. Следующие модули не загружаются по-умолчанию и их надо загрузить явным образом с помощью команды: `sudo modprobe tun; sudo modprobe fuse`.

AXIOM JDK

Axiom Linux
Руководство по работе с репозиторием и контейнерами