AXIOM JDK

# Инструкция по работе с Axiom Trusted Images 25 LTS

АО «АКСИОМ» I Сентябрь 2026

Copyright © 2019-2026 Все права защищены АО "АКСИОМ" (АКСИОМ)

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

2. Образы ..... 6

3. Доступ к реестру ..... 7

Windows ..... 7

Linux и MacOS ..... 8

4. Проверка подписей ..... 9

5. Выгрузка SBOM ..... 10

6. Перечень образов ..... 11

25-trusted-base ..... 11

25-trusted-axiom-runtime-container-pro ..... 11

25-trusted-axiom-native-image-kit-container ..... 13

25-trusted-gcc ..... 14

25-trusted-go . . . . . 15

25-trusted-python . . . . . 16

25-trusted-nginx . . . . . 17

25-trusted-node . . . . . 18

# 1. Введение

Axiom Trusted Images 25 LTS - это набор готовых контейнеров с образами на базе ОС Axiom Linux 25 LTS. Образы могут включать Axiom JDK Pro, Axiom NIK Pro, интерпретатор языка Python, компиляторы GCC и Go, среду выполнения NodeJS и другие утилиты и библиотеки. См. Перечень образов.

Существует несколько видов контейнеров, каждый из которых обеспечивает разное соотношение между возможностями отладки и безопасности - обычные, nonroot, distroless. Различия между ними приведены в таблице:

| | **обычный** | **nonroot** | **distroless** |
|---|---|---|---|
| пользователь | root | appuser | appuser |
| шелл | да | да | нет |
| entrypoint | /bin/sh | /bin/sh | приложение |

Во всех образах удалены инструменты для установки нового ПО (apk-tools).

В образах nonroot и distroless настроен непривилегированный пользователь `appuser` с домашней директорией. От этого пользователя выполняется запуск entrypoint.

В образах distroless отсутствует командная строка, и поставляется только минимальный набор библиотек и других зависимостей, необходимых для запуска приложений.

Все 3 вида образов размещаются в одном репозитории, но по разным тегам. Если в теге присутствует слово `nonroot` - это nonroot образ, если `distroless` - это distroless образ, в других случаях - это обычный образ.

# 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.

# 3. Доступ к реестру

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

## Windows

Если вы используете систему Windows, вы можете настроить доступ к реестру cr.yandex через интерфейс командной строки Yandex Cloud (CLI).

Выполните следующие шаги для доступа к репозиторию Axiom Trusted Images 25 LTS из Windows PowerShell.

1. Установите Yandex CLI.

```
iex (New-Object
System.Net.WebClient).DownloadString('https://storage.yandexcloud.net/yand
excloud-yc/install.ps1')
```

2. Создайте и настройте профиль для работы.

```
yc config profile create main-profile
```

3. Для настройки аутентификации скопируйте полученный json файл на устройство и выполните команду для авторизации.

```
yc config set service-account-key .\<имя файла>.json
```

4. Настройте folder id.

```
yc config set folder-id b1ggrgs5h8c36smobplc
```

5. Настройте Docker.

```
yc container registry configure-docker
```

Теперь вы можете работать с образами в реестре cr.yandex используя Windows PowerShell.

## Linux и MacOS

Используйте командную строку вашего экземпляра Linux или MacOS.

Для настройки аутентификации скопируйте полученный json файл на устройство и выполните следующие команды для авторизации:

```
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 и 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
```

# 4. Проверка подписей

Все образы в представленных репозиториях подписаны ключом АО «АКСИОМ» с использованием программы cosign.

Для проверки подписи необходимо выполнить следущие шаги:

1. Скачайте SSL сертификат с ключом

```
curl -O https://download.axiomjdk.ru/ssl-certificates/axiom-jsc-image-signing.crt
```

2. Скачайте корневой сертификат к нему

```
curl -O https://download.axiomjdk.ru/ssl-certificates/axiom-jsc-ca.crt
```

3. Проверьте, что сертификат с ключом действительно выпущен корневым сертификатом

```
openssl verify -CAfile axiom-jsc-ca.crt -show_chain axiom-jsc-image-signing.crt
```

4. Извлеките публичный ключ из сертификата

```
openssl x509 -in axiom-jsc-image-signing.crt -noout -pubkey -out public.key
```

5. Проверьте подпись образа

```
cosign verify --key public.key --private-infrastructure=true <идентификатор образа>
```

# 5. Выгрузка SBOM

К каждому образу прикреплены 2 файла SBOM в форматах:

- CycloneDX 1.6
- SPDX 2.3

Эти файлы содержат подписанные списки компонент, из которых состоит образ.

Для выгрузки SBOM потребуется программа `cosign` и публичный ключ из раздела Проверка подписей.

Для получения файла в формате CycloneDX выполните команду:

```
cosign verify-attestation \
    --key public.key \
    --type cyclonedx \
    <идентификатор образа> | jq -r '.payload' | base64 -d | jq '.predicate'
```

Для получения файла в формате SPDX выполните команду:

```
cosign verify-attestation \
    --key public.key \
    --type spdx \
    <идентификатор образа> | jq -r '.payload' | base64 -d | jq '.predicate'
```

# 6. Перечень образов

## 25-trusted-base

Репозиторий содержит образы с минимальным набором компонент ОС для запуска статических слинкованных программ, либо программ, зависящих только от стандартной библиотеки C.

Теги доступных образов определяются следующим шаблоном:

`[nonroot-|distroless-]<libc>`

где:

- `<libc>` задает версию C библиотеки и может принимать значения:
  - glibc
  - musl

Примеры использования этих образов приведены в последующих разделах.

## 25-trusted-axiom-runtime-container-pro

Репозиторий содержит образы с Axiom JDK Pro.

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

Теги доступных образов определяются следующим шаблоном.

`<jdk_type>-<java_version>[-crac][-cds][-nonroot|-distroless]-<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 согласованное восстановление из контрольной точки (CRAC).
- `cds` присутствует в тегах образов с Class Data Sharing (для ускорения первого запуска JVM).

Примеры тегов:

- jre-25-distroless-musl
- jdk-all-11-crac-cds-glibc

Ниже представлен пример Dockerfile, в котором Java приложение сначала компилируется в `jdk nonroot` образе, после чего создается образ с приложением на базе `jre distroless` варианте:

```
FROM <...>/25-trusted-axiom-runtime-container-pro:jdk-11-nonroot-glibc AS
builder

COPY <<'EOF' HelloAxiom.java
public class HelloAxiom {
    public static void main(String[] args) {
        System.out.println("Hello Axiom!");
    }
}
EOF

RUN javac HelloAxiom.java

FROM <...>/25-trusted-axiom-runtime-container-pro:jre-11-distroless-glibc

COPY --from=builder /home/appuser/HelloAxiom.class .

ENTRYPOINT [ "/usr/bin/java" ]
CMD [ "HelloAxiom"]
```

# 25-trusted-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>[-nonroot|-distroless]-<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

Примеры тегов:

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

Ниже представлен пример Dockerfile, в котором Java приложение с помощью NIK компилируется в бинарный файл, который затем запускается в базовом distroless образе:

```
FROM <...>/25-trusted-axiom-native-image-kit-container:jdk-25-nik-25-nonroot-
musl AS builder

COPY <<'EOF' HelloAxiom.java
public class HelloAxiom {
    public static void main(String[] args) {
        System.out.println("Hello Axiom!");
    }
}
EOF

RUN javac HelloAxiom.java
RUN native-image --no-fallback -o hello-axiom HelloAxiom
```

```
FROM <...>/25-trusted-base:distroless-musl

COPY --from=builder /home/appuser/hello-axiom .

ENTRYPOINT [ "/home/appuser/hello-axiom" ]
```

## 25-trusted-gcc

Репозиторий содержит образы с инструментами для C/C++ разработки на базе GCC 14.3.

Теги доступных образов определяются следующим шаблоном.

`14.3[-nonroot|-distroless]-<libc>`,

где:

- `<libc>` задает версию C библиотеки и может принимать значения:
  - glibc
  - musl

Ниже представлен пример Dockerfile со сборкой C приложения с помощью nonroot образа с последующим запуском в базовом distroless образе:

```
FROM <...>/25-trusted-gcc:14.3-nonroot-musl AS builder

COPY <<'EOF' hello_axiom.c
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("Hello Axiom!\n");
}
EOF

RUN gcc -Wall -o hello-axiom hello_axiom.c

FROM <...>/25-trusted-base:distroless-musl

COPY --from=builder /home/appuser/hello-axiom .

ENTRYPOINT [ "/home/appuser/hello-axiom" ]
```

# 25-trusted-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:

Предоставляются образы только с версией Go, поддерживаемой разработчиками языка Go (https://go.dev).

Теги доступных образов определяются следующим шаблоном:

`<go_version>[-nonroot|-distroless]-<libc>`

где:

- `<go_version>` - версия Go в формате `X.Y`, `X.Y.Z`
- `<libc>` задает версию C библиотеки и может принимать значения:
  - glibc
  - musl

Кроме того, представлены теги, указывающие на последнюю версию Go. Они определяются шаблоном:

`[nonroot][distroless]-<libc>`

Примеры тегов:

- 1.25.9-glibc
- 1.25-musl
- distroless-glibc
- glibc
- musl

Ниже представлен пример Dockerfile со сборкой Go приложения с помощью nonroot образа с последующим запуском в базовом distroless образе:

```
FROM <...>/25-trusted-go:nonroot-musl AS builder

COPY <<'EOF' hello_axiom.go
package main
```

```
import "fmt"

func main() {
    fmt.Println("Hello Axiom")
}
EOF

RUN go build -o hello-axiom hello_axiom.go

FROM <...>/25-trusted-base:distroless-musl

COPY --from=builder /home/appuser/hello-axiom .

ENTRYPOINT [ "/home/appuser/hello-axiom" ]
```

## 25-trusted-python

Репозиторий содержит образы с Python 3.12 и базовыми Python утилитами (pip, setuptools, wheel).

Теги доступных образов определяются следующим шаблоном:

`3.12[-nonroot|-distroless]-<libc>` где:

- `<libc>` задает версию C библиотеки и может принимать значения:
    - glibc
    - musl

Пример Dockerfile с запуском приложения в distroless образе:

```
FROM <...>/25-trusted-python:3.12-distroless-glibc

COPY <<'EOF' hello_axiom.py
if __name__ == "__main__":
    print("Hello Axiom")
EOF

CMD [ "hello_axiom.py" ]
```

## 25-trusted-nginx

Репозиторий содержит образы с веб-сервером Nginx.

Теги доступных образов определяются следующим шаблоном.

`[<bundle_type>-]-<nginx_version>[-nonroot|-distroless]-<libc>`

- `<bundle_type>` это тип поставляемых компонентов.

  Если `<bundle_type>` опущен, то он трактуется как **default**. Существуют следующие варианты `<bundle_type>`:

  - **default**

    Содержит минимальный набор пакетов, необходимых для работы `nginx`. Не содержит никаких динамических модулей `nginx`.

  - **dynamic**

    Как **default** версия, но также содержит следующие динамические модули:

    - `ngx_http_brotli_filter`
    - `ngx_http_brotli_static`
    - `ngx_http_geoip`
    - `ngx_http_image_filter`
    - `ngx_http_js`
    - `ngx_http_xslt_filter`
    - `ngx_stream_geoip`
    - `ngx_stream_js`
    - `ngx_stream`

- `<nginx_version>` это версия Nginx. Она имет формат `X.Y`, например, `1.30`.
- `<libc>` задает версию C библиотеки и может принимать значения:
  - glibc
  - musl

Примеры тегов:

- 1.30-nonroot-glibc
- dynamic-1.30-distroless-glibc

Следующий пример кода демонстрирует Dockerfile, который создает образ для размещения простого статического контента:

```
FROM bellsoft/hardened-nginx:distroless
COPY static-html-directory /var/lib/nginx/html
```

Выполните следующие команды в директории, где находится директория `static-html-directory`, чтобы собрать и запустить контейнер:

```
$ docker build -t my-nginx .
$ docker run -d -p 8080:8080 my-nginx
```

После этого вы сможете получить доступ к http://localhost:8080.

## 25-trusted-node

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

Предоставляются образы только с версией Node.js, поддерживаемой разработчиками Node.js (https://nodejs.org).

Для корректной обработки сигналов ОС рекомендуется в Dockerfile запускать процесс `node` не напрямую, а используя прослойку `/usr/sbin/tini`, которая уже предустановлена во всех образах.

Теги доступных образов определяются следующим шаблоном:

`<nodejs_version>[-nonroot|-distroless]-<libc>`

где:

- `<nodejs_version>` - версия Node.js в формате `X`, `X.Y`, `X.Y.Z`
- `<libc>` задает версию C библиотеки и может принимать значения:
  - glibc
  - musl

Кроме того, представлены теги, указывающие на последнюю версию Node.js. Они определяются шаблоном:

```
[nonroot][distroless]-<libc>
```

Примеры тегов:

- 22.22.2-glibc
- 22.22-nonroot-glibc
- 22-musl
- distroless-glibc
- glibc
- musl

Пример Dockerfile с запуском приложения в distroless образе:

```
FROM <...>:distroless-musl

COPY <<'EOF' hello_axiom.js
import { createServer } from 'node:http';

const server = createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello Axiom!\n');
});

server.listen(5000, '0.0.0.0', () => {
  console.log('Listening on 0.0.0.0:5000');
});
EOF

ENTRYPOINT [ "/usr/sbin/tini", "--"]
CMD [ "node", "hello_axiom.js"]
```

AXIOM JDK

# Инструкция по работе с Axiom Trusted Images 25 LTS