AXIOM JDK

Axiom NIK Pro Контейнеризация нативных образов

Axiom NIK Pro I Март 2025

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

Технология нативных образов набирает популярность среди разработчиков, основной целью которых является ускорение запуска приложений. В этом документе мы расскажем вам о том, как создавать нативные образы в контейнерах для дальнейшего развертывания в облаке. Мы будем использовать Инструментарий Нативных Образов Axiom NIK Pro в качестве компилятора нативных образов и Axiom Linux в качестве базового образа.

2. Создание приложения

Создайте папку для вашего демонстрационного проекта. Затем перейдите в папку проекта и создайте простое Java-приложение в консоли:

```
cat >./Demo.java <<EOL
public class Demo {
    public static void main(String[] args) {
        System.out.println("Привет от Axiom NIK Pro!");
    }
}
EOL
```

3. Загрузка образа Docker

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" и документ Axiom Linux: Руководство по работе с репозиторием и контейнерами.

Например, следующая команда скачивает образ Axiom NIK Pro 22.3 для Java 11 с musl libc:

```
docker run --rm -it <URL адрес репозитория>/axiom-native-image-kit-
container:jdk-11-nik-22.3-musl
```

Вы можете пропустить этот шаг, но мы рекомендуем загрузить образ, если вы не хотите делать это каждый раз при повторении процесса сборки.

4. Создание файла Dockerfile

Мы создадим нативный образ непосредственно в контейнере, что полезно, когда архитектура разработки и развертывания отличается. Это также полезно, если вы хотите создать образ на основе musl, который занимает меньше памяти, чем образ на основе glibc.

Нам нужно создать файл типа Dockerfile для создания контейнера изображений Docker. Поместите следующий файл в папку приложения:

```
FROM <URL адрес репозитория>/axiom-native-image-kit-container:jdk-11-nik-
22.3-musl
WORKDIR /home/myapp
COPY Demo.java /home/myapp/
RUN javac Demo.java
RUN native-image Demo
FROM <URL адрес репозитория>/axiom-linux-base:stream-musl
WORKDIR /home/myapp
COPY --from=0 /home/myapp/demo .
CMD [“./demo”]
```

Приведенный выше код выполняет следующие действия:

1. Определяет базовый образ для создания нативного образа;
2. Указывает на каталог, в котором образ будет выполняться внутри Docker;
3. Копирует программу в каталог;
4. Запускает компилятор javac для создания байт-кода нашего приложения;
5. Запускает инструмент нативных образов для создания образа;
6. Создает другой образ с помощью базового образа Axiom Linux (для создания нативного образа не требуется JVM);
7. Указывает каталог исполняемого файла;
8. Копирует приложение в новый образ;
9. Запускает программу внутри контейнера.

5. Создание контейнера с нативным образом

Перед созданием контейнера рекомендуется закрыть браузер, IDE и другие программы, занимающие много памяти. Чтобы сгенерировать нативный образ и поместить его в контейнер, запустите следующую команду:

```
docker build .
```

Убедитесь, что образ был создан с помощью следующей команды:

```
docker images
REPOSITORY     TAG            IMAGE ID       CREATED              SIZE
<none>         <none>         2921e3483bb2   21 seconds ago       18.4MB
```

Пометьте тегом только что созданный образ:

```
docker tag 2921e3483bb2 nik-example
```

Теперь вы можете запустить образ с помощью следующей команды:

```
docker run -it -rm 2921e3483bb2
Привет от Axiom NIK Pro!
```

6. Примечание

Мы использовали простую программу, которая не требует ручной настройки. Однако динамические функции Java, такие, как JNI, сериализация и т.д. не поддерживаются GraalVM, поэтому вы должны дополнительно настроить инструмент создания нативных образов для использования этих функций.

AXIOM JDK

Axiom NIK Pro
Контейнеризация нативных образов