АРМ С3000

Установка образов Docker в ОС Linux

2023

Оглавление

Введение

Руководство предназначено для системных администраторов, выполняющих установку и начальную настройку **АРМ С3000** с использованием Docker (https://www.docker.io) в операционных системах семейства Linux.

Соглашения и условные обозначения

- **Полужирным** выделяются названия программных продуктов и аппаратных средств.
- *Курсив* применяется для обозначения технических терминов и в иных случаях для выделения частей текста.
- `Моноширинный шрифт` применяется для имен файлов, команд и их параметров, а также для примеров выполнения и вывода команд.
- В примерах выполнения команд, символ приглашения командной строки (command prompt) `$` говорит о том, что команда выполняется от имени непривилегированного пользователя. `#` используется для команд, выполняемых суперпользователем (`root`, администратором системы).
- В соответствии с принятыми в документации для систем семейства **UNIX** соглашениями, имена команд записываются с указанием в скобках соответствующего номера раздела страниц руководства (man pages), например: `lsusb(8)`, `dmesg(1)`.
- **Примечание:** – краткие аннотации к основному тексту.

Сокращения

- ОС – операционная система
- ПО – программное обеспечение

Системные требования

Поддерживаются следующие операционные системы:

- Astra Linux Special Edition 1.7 («Орел», «Воронеж», «Смоленск»)
- Ubuntu Linux 20.04 LTS («Focal Fossa»), 22.04 LTS («Jammy Jellyfish»)

Для каждой из ОС предоставляется свой образ **Docker**. При установке на других системах рекомендуется использовать образ, предназначенный для `Astra Linux`.

Установка ПО Docker

Astra Linux

Установить пакет `docker.io`:

```
$ sudo apt install docker.io
```

Запустить службу **Docker**:

```
$ sudo systemctl start docker
```

Включить автоматический запуск службы:

```
$ sudo systemctl enable docker
```

При необходимости, разрешить работу с **Docker** непривилегированным пользователям. Например, для пользователя *USER_NAME*:

```
$ sudo usermod -a -G docker USER_NAME
```

Для использования **Docker** в *непривилегированном* (*rootless*) режиме (служба **Docker** запускается без прав суперпользователя, `root`):

- Установить пакет `rootless-helper-astra`:
  ```
  $ sudo apt install rootless-helper-astra
  ```
- Запустить службу **Docker** от имени пользователя *USER_NAME*:
  ```
  $ sudo systemctl start rootless-docker@USER_NAME
  ```
- Включить автоматический запуск службы от имени пользователя *USER_NAME*:
  ```
  $ sudo systemctl start rootless-docker@USER_NAME
  ```

Для запуска команд **Docker** в непривилегированном режиме следует использовать `rootlessenv(1)` из пакета `rootless-helper-astra`, например: `rootlessenv docker run ...`, `rootlessenv docker volume create ...`, и т.д.

Официальная документация: Установка и администрирование Docker в Astra Linux 1.7 (https://wiki.astralinux.ru/pages/viewpage.action?pageId=158601444).

Ubuntu Linux

Предпочтительно устанавливать **Docker** из официального репозитория.

Прежде всего, следует удалить пакеты **Docker**, установленные из репозиториев Ubuntu:

```
$ sudo apt purge         \
    docker-ce            \
    docker-ce-cli        \
    containerd.io        \
    docker-buildx-plugin \
    docker-compose-plugin
```

Установить необходимые пакеты:

```
$ sudo apt install  \
    ca-certificates \
```

```
    curl            \
    gnupg           \
    lsb-release
```

Загрузить и добавить GPG-ключ репозитория:

```
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
  | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Добавить репозиторий:

```
$ echo "deb [arch=$(dpkg --print-architecture)                        \
  signed-by=/etc/apt/keyrings/docker.gpg]                             \
  https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" \
  | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

После выполнения команды будет создан файл `/etc/apt/sources.list.d/docker.list` следующего вида (приведён пример для Ubuntu 22.04 LTS «Jammy Jellyfish» на машине с архитектурой `amd64`):

```
deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian jammy stable
```

Обновить список пакетов:

```
$ sudo apt update
```

Установить пакеты **Docker**:

```
$ sudo apt install \
    docker-ce      \
    docker-ce-cli  \
    containerd.io  \
    docker-compose-plugin
```

В случае успешной установки, служба `docker` будет запущена и добавлена в автоматическую загрузку. Для проверки следует выполнить следующие команды и убедиться в соответствии их вывода приведённому ниже:

```
$ systemctl status docker

docker.service - Docker Application Container Engine
    Loaded: loaded (/lib/systemd/system/docker.service; enabled; ...
    Active: active (running) since Thu 2023-08-10 22:35:32 MSK; ...
    TriggeredBy: docker.socket
    ...
$ systemctl list-unit-files | grep -i docker

docker.service    enabled    enabled
docker.socket     enabled    enabled
```

При необходимости, разрешить работу с **Docker** непривилегированным пользователям. Например, для пользователя *USER_NAME*:

```
$ sudo usermod -a -G docker USER_NAME
```

Проверка работоспособности ПО Docker

Прежде чем приступить к работе с образами **АРМ С3000**, рекомендуется произвести проверку **Docker** с использованием специально предназначенного для этой цели контейнера `hello-world`:

Убедиться в наличии подключения к сети Интернет.

Выполнить команду:

```
$ sudo docker run hello-world
```

В случае правильной установки и настройки **Docker**, вывод должен быть таким:

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:dcba6daec718f547568c562956fa47e1b03673dd010fe6ee58ca806767031d1c
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
...
```

Подготовка контейнера

Импортировать образ в локальный репозиторий **Docker**:

```
$ sudo docker load --input arm-s3000-astra-smolensk_1.7-VERSION.tar.xz
```

Здесь и далее, *VERSION* в имени файла следует заменить на номер версии образа, с которым фактически происходит работа. Например, для версии `1.01.654.182`, имя файла будет выглядеть как `arm-s3000-astra-smolensk_1.7-1.01.654.182.tar.xz`.

Создать том **Docker** для хранения данных контейнера. `arm-s3000-volume` в команде – произвольное имя тома (должно быть уникальным в пределах локальной ОС):

```
$ sudo docker volume create arm-s3000-volume
```

Запуск контейнера

Запуск производится командой:

```
$ sudo docker run                       \
    --name arm-s3000                    \
    --volume arm-s3000-volume:/persist  \
    --restart=always                    \
    --publish 20080:80                  \
    --publish 20043:443                 \
    arm-s3000-astra-smolensk_1.7:VERSION
```

Команде `docker run` передаются следующие параметры:

- `--name arm-s3000` Произвольное имя контейнера для использования в командах `docker(1)`.
- `--volume arm-s3000-volume:/persist` Имя тома, созданного командой `docker volume create`. `/persist` – папка в контейнере, где будет смонтирован том.
- `--restart=always` Автоматический перезапуск контейнера в случае завершения его работы.
- `--publish 20080:80 --publish 20043:443` Перенаправление портов TCP. Соединение с портом, указанным до `:`, на локальной системе будет перенаправлено на порт, указанный после `:`, в контейнере.
- `arm-s3000-astra-smolensk_1.7:VERSION` Имя образа **Docker**. Про `VERSION` см. раздел «Подготовка контейнера» выше.

После успешного запуска контейнера соединение с системой **АРМ С3000** возможно на всех сетевых интерфейсах и заданных портах, например: `http://127.0.0.1:20080` или `https://127.0.0.1:20043`.

Перенаправление портов UDP

Перенаправление портов UDP может потребоваться:

- При подключении приборов к **АРМ С3000** через устройство **С2000-Ethernet** в том случае, если в настройках **С2000-Ethernet** отключен параметр «Использовать один UDP-порт на чтение и запись».
- В случае возникновения проблем при использовании NAT.

Для этого необходимо передать команде `docker run` параметр вида `--publish 20500:60500/udp`. Где до `:` указан порт на локальной системе, а после `:` – порт в контейнере.

Номера портов на локальной системе могут принимать значения от 2048 до 65535.

Работа с ключом защиты

При запуске системы **АРМ С3000** происходит поиск ключа защиты, подключенного к порту USB. Без ключа **АРМ С3000** будет работать в демонстрационном режиме. Для того, чтобы ОС, запущенная в контейнере, смогла найти это устройство, команде `docker run` необходимо передать параметр `--device`, например:

```
--device=/dev/bus/usb/002/003
```

Здесь `/dev/bus/usb/001/002` указывает путь к файлу устройства ключа защиты. Определить этот путь позволяет утилита `lsusb(8)`:

```
$ lsusb -v

Bus 002 Device 003: ID 04d8:053f Microchip Technology, Inc.
Device Descriptor:
  bLength 18
  ...
  iProduct 2 Bolid security dongle
  ...
```

В выводе команды `lsusb -v` нужно найти запись со значение поля `iProduct` равным «Bolid security dongle». Поля `Bus` и `Device` этой записи позволяют сформировать путь к файлу устройства. Например, для `Bus 002` и `Device 003` путь будет таким:

```
/dev/bus/usb/002/003
```

Использование преобразователей USB в RS-232 и RS-485

Для работы с такими преобразователями команде `docker run` необходимо передать параметр `--device`, указывающий путь к файлу используемого устройства. Этот путь можно определить с помощью команд `lsusb(8)`, `dmesg(1)`, а также из файла `/proc/tty/drivers`:

```
$ cat /proc/tty/drivers

/dev/tty        /dev/tty         5  0        system:/dev/tty
/dev/console    /dev/console     5  1        system:console
...
usbserial       /dev/ttyUSB    188  0-511    serial
...
```

Здесь мы видим файл устройства `/dev/ttyUSB` (подключен преобразователь USB в RS-485 **C2000-USB**), соответственно, параметр для команды `docker run` будет таким: `--device=/dev/ttyUSB`. Этот же путь следует указать в Web-интерфейсе системы **АРМ С3000**: поле «Устройство» в настройках порта на вкладке «Порты RS» («Настройки портов и протоколов»).

Остановка и удаление контейнера

Остановить контейнер:

```
# docker stop arm-s3000
```

Удалить том **Docker** (`arm-s3000-volume` – имя тома):

```
# docker volume rm arm-s3000-volume
```

Удалить образ **Docker**:

```
# docker image rm arm-s3000-astra-smolensk_1.7:VERSION
```

Восстановление и сброс паролей

В случае утери пароля для встроенной учетной записи, а также при необходимости изменения паролей других пользователей без использования web-интерфейса, используется команда `password-reset`, запускаемая в контейнере.

Вызванная без параметров, она восстанавливает пароль по умолчанию (*armS3000*) для пользователя *admin*. При вызове с ключом `-u` команда меняет пароль для пользователя с указанным именем учетной записи.

Сначала нужно остановить контейнер (`arm-s3000` – имя контейнера):

```
# docker stop arm-s3000
```

Восстановить пароль пользователя *admin*:

```
# docker run                                \
    --name arm-s3000                        \
    --volume arm-s3000-volume:/persist      \
    --rm                                    \
    arm-s3000-astra-smolensk_1.7:VERSION \
    password-reset
```

Описание параметров команды `docker run` приведены в разделе «Запуск контейнера».

Задать новый пароль *new_password* для пользователя *user_name*:

```
# docker run                                \
    --name arm-s3000                        \
    --volume arm-s3000-volume:/persist      \
    --rm                                    \
    arm-s3000-astra-smolensk_1.7:VERSION \
    password-reset -u "user_name" "new_password"
```

Примечание:
Если пользователь с именем, переданным команде, не существует, он будет создан; *роль* новой учетной записи – `service`.

При последующем запуске контейнера вступят в действие новые пароли.