EB Corbos Linux - SDK

Build Target Image(s)

The EB Corbos Linux (EBCL) OS, codename NautilOS, will be delivered as software repositories including binary and source packages based on the Ubuntu distribution. One part of EBCL consists out of a collection of reference example image descriptions which can be directly deployed to the target hardware. The main purpose of these images descriptions is to provide a straight forward path to get started and to outline the main features of EBCL.

For producing target image(s), it is required to provide an environment that allows to build and modify them. Together with packaging and compiler tools, this system represents the EBCL SDK.

The following chapters are designed as user guide, describing how to setup a SDK host and how to get started building target image(s).

Prerequisites

EBCL software repositories

The EBCL software repositories are delivered via https://artifactory.elektrobit.com Please make sure to have your account details (username and password) at hand.

SDK hardware requirements

No matter if the SDK runs on virtual- or real hardware, a minimum of 4G of RAM space and 10G of **free** storage space are required !

Quick Start

For the quick setup process, the `ebcl-ui` utility is provided here: ebcl-ui

NOTE	Access is still protected even though we plan to opensource it. In case access is denied please get in Contact with the Elektrobit Linux team.

Fetch `ebcl-ui` and store it on the desired SDK machine. Once available call:

```
ebcl-ui
```

The program guides the user through a dialog based process. At its end a menu shows up which allows for the following actions:

1. **Create/Update Local Repository Mirror**

In case the SDK workflow should be offline, this selection creates a local mirror of the EBCL software repositories. During runtime a connection to Artifactory is required. Subsequent selections of this menu entry leads to an incremental update of the data.

2. **Run SDK Setup**
   In case there is no pre-provided SDK configuration available this selection will configure the SDK using the SDK as OCI container (x86_64, aarch64) method. If no local mirror exists this requires a connection to Artifactory
3. **Build Reference Image(s)**
   This selection searches for the installed reference image descriptions and offers them in a list view from which the user can select. The selected image will then be build. If no local mirror exists this requires a connection to Artifactory

SDK Setup Methods

The following chapters describes the SDK and image building setup procedures on a commandline tool based workflow. For a quick start without further details about the SDK tooling and features go to Quick Start

There are currently four setup methods available. Please select one from the list below:

NOTE	The description on the `ebcl-repo-setup` tool is provided in production mode which will access the data from https://artifactory.elektrobit.com. In case the SDK workloads should be done in offline mode, an `ebcl-sync` process as described in Offline Image Building should be performed first.

- SDK as OCI container (x86_64, aarch64)
  Requires on premise Ubuntu system, possible to be integrated in kubernetes cluser.
- SDK as AWS EC2 Cloud Instance (x86_64, aarch64)
  Requires AWS cloud account with EC2 permissions.
- SDK on RaspberryPI (aarch64)
  Requires on premise **Raspberry Pi 4 Model B** hardware.
- SDK on Ubuntu based build machine (x86_64, aarch64)
  Requires on premise hardware for custom installation and setup procedure.

SDK as OCI container (x86_64, aarch64)

EBCL software packages and containers are created for the Ubuntu 22.04 (jammy) distribution. Thus EBCL components should be used with an Ubuntu 22.04 (jammy) based host system.

NOTE	It is not mandatory to use Ubuntu 22.04 (jammy) as build host OS. However, different instructions to install the required components are needed. In this case please consult EB for further details.

There are OCI containers for the **x86_64** and **aarch64** architectures available. The containers are provided as Debian packages and can be installed using the native **apt** toolchain. At install time of

the container they will be automatically loaded into the local registry and `ebcl-build-ARCH` commands will be registered using the **oci-pilot** container application manager. The registered `ebcl-build-ARCH` commands are pre-configured with the following settings:

- Container runs in privileged mode.
- Container shares `/var/tmp` with the host, which is the default output directory for the image builds.
- Container shares `/usr/share/kiwi/nautilos` with the host, which is the default location for the image descriptions.
- Container will be removed when it exits.

WARNING	Running image build processes inside of a container has its limitations. Even in privileged mode the container shares the kernel with the host. This means all device operations must be compatible with the host kernel. In addition it is not possible to run container workloads inside of a container. If this sort of issue arises please select another SDK setup method.

Repository Setup

If not already done, download and install the latest `nautilos-repo-setup` package as follows:

```
A_USER=ARTIFACTORY_USER_NAME
```

```
A_SECRET=ARTIFACTORY_SECRET
```

```
wget --user $A_USER --password $A_SECRET \
    --recursive --no-directories --no-parent -A 'nautilos-repo-setup*.deb' \
https://artifactory.elektrobit.com/eb_corbos_linux_nautilos2.0-debian-
remote/release/xUbuntu_22.04_debbuild/all/
```

```
sudo dpkg --install $(ls -1t nautilos-repo-setup* | head -n1)
```

Check for available releases as follows:

```
ebcl-releases --user $A_USER --password $A_SECRET
```

Select the release date of your choice and setup the EBCL repos as follows:

```
sudo ebcl-repo-setup --production YYYY-MM-DD --user $A_USER --password $A_SECRET
```

Installation: Reference Example Image Description

Install one ore more of the provided example descriptions as follows. For further details see: Reference Example Image Descriptions

```
sudo apt install reference-image-description-crinit
```

Installation: SDK Container

Install EBCL SDK container as follows:

```
sudo apt install oci-pilot
sudo apt install nautilos-builder-oci-flake
```

NOTE

For **arm64** (multiarch) support on **x86_64** it's required to apply the binfmt registration from the **multiarch/qemu-user-static** container as follows:

```
sudo podman run --rm --privileged \
    docker.io/multiarch/qemu-user-static --reset -p yes
```

Next install the **arm64** SDK Container as follows:

```
sudo apt -o Dpkg::Options::="--force-architecture" \
    install elektrobit-arm64-sdk-image:arm64
```

Once done continue with Image Building

SDK as AWS EC2 Cloud Instance (x86_64, aarch64)

In the Continental/Elektrobit organisation at Amazon, there are registrations for AWS EC2 AMI's (Amazon Machine Images) for the x86_64 and aarch64 architectures. Along with the business contract the respective AMI IDs can be shared with the customer account. Once this is done start an instance of the SDK AMI either through the AWS web console or using the commandline interface as follows:

```
aws ec2 run-instances \
  --count 1 \
  --image-id SDK-AMI-ID \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=SDK}]' \
  --instance-type AWS-EC2-INSTANCE-TYPE \
```

```
  --key-name SSH-KEY-PAIR-NAME
```

Environment Setup

Once the EC2 SDK instance is running, login via ssh (user ubuntu) and setup the `~/.ebcl` environment as follows:

```
A_USER=ARTIFACTORY_USER_NAME
```

```
A_SECRET=ARTIFACTORY_SECRET
```

Check for available releases as follows:

```
ebcl-releases --user $A_USER --password $A_SECRET
```

Select the release date of your choice and setup the EBCL repos as follows:

```
sudo ebcl-repo-setup --production YYYY-MM-DD --user $A_USER --password $A_SECRET --no
-init
```

Once done continue with Image Building

SDK on RaspberryPI (aarch64)

To setup a SDK system on a **RaspberryPI** dump the **uncompressed** image from: nautilos-builder-rpi on a SD card and boot it up on a **Raspberry Pi 4 Model B** with either 4G (better 8G) of RAM space.

NOTE	Access to the **RaspberryPI** image requires access to the Elektrobit Build Service. In case access is denied please get in Contact with the Elektrobit Linux team.

Environment Setup

Once the Raspberry Pi is running, login (user root, default pass: **linux**) and setup the `~/.ebcl` environment as follows:

```
A_USER=ARTIFACTORY_USER_NAME
```

```
A_SECRET=ARTIFACTORY_SECRET
```

Check for available releases as follows:

```
ebcl-releases --user $A_USER --password $A_SECRET
```

Select the release date of your choice and setup the EBCL repos as follows:

```
sudo ebcl-repo-setup --production YYYY-MM-DD --user $A_USER --password $A_SECRET --no
-init
```

Once done continue with Image Building

SDK on Ubuntu based build machine (x86_64, aarch64)

EBCL software packages are created for the Ubuntu 22.04 (jammy) distribution. Thus EBCL components should be used with an Ubuntu 22.04 (jammy) based host system.

NOTE	It is not mandatory to use Ubuntu 22.04 (jammy) as build host OS. However, different instructions to install the required components are needed. In this case please get in Contact with the Elektrobit Linux team.

System packages

Prior installing EBCL SDK components double check for required system packages to be installed:

```
sudo apt install \
    dpkg-dev apt-utils git gnupg netcat gawk \
    jing qemu-kvm qemu-system-x86 qemu-system-arm \
    python3-solv libsolv-tools fdisk device-tree-compiler \
    file runc podman rust-all libxml2-utils iptables
```

The SDK setup process covers among others the installation of the **KIWI** (https://osinside.github.io/kiwi) appliance builder which is used to build the image from an also to be installed, reference example image description. The following steps describes how to install the needed components.

Repository Setup

If not already done, download and install the latest `nautilos-repo-setup` package as follows:

```
A_USER=ARTIFACTORY_USER_NAME
```

```
A_SECRET=ARTIFACTORY_SECRET
```

```
wget --user $A_USER --password $A_SECRET \
    --recursive --no-directories --no-parent -A 'nautilos-repo-setup*.deb' \
https://artifactory.elektrobit.com/eb_corbos_linux_nautilos2.0-debian-
remote/release/xUbuntu_22.04_debbuild/all/
```

```
sudo dpkg --install $(ls -1t nautilos-repo-setup* | head -n1)
```

Check for available releases as follows:

```
ebcl-releases --user $A_USER --password $A_SECRET
```

Select the release date of your choice and setup the EBCL repos as follows:

```
sudo ebcl-repo-setup --production YYYY-MM-DD --user $A_USER --password $A_SECRET
```

Installation: Image Builder

Install EBCL SDK components as follows:

```
sudo apt install \
    python3-kiwi kiwi-systemdeps python-kiwi_boxed_plugin debbuild \
    oci-pilot oci-deb nautilos-uboot-tools signing-key-tools \
    storage-key-tools
```

Installation: Reference Example Image Description

Install one ore more of the provided example descriptions as follows. For further details see: Reference Example Image Descriptions

```
sudo apt install reference-image-description-crinit
```

All reference example image description(s) will be installed to the `/usr/share/kiwi/nautilos/` directory. Each image description contains the same `build` script which can be used to build the image for its configured targets. Copy any of the installed build script as follows:

```
test $(uname -m) = "aarch64" && build=-arm64
test $(uname -m) = "x86_64" && build=-amd64
sudo cp \
```

```
    /usr/share/kiwi/nautilos/reference-image-description-crinit/nxp/build \
    /usr/bin/ebcl-build${build}
```

Image Building

Prior building an image the following command lists the available image descriptions:

```
sudo ebcl-build-arm64 --list
```

NOTE	The above command exists on systems that supports the arm64(aarch64) architecture. On x86 systems the `ebcl-build-amd64` exists.

Building the minimal crinit based system image for the NXP(arm64) board as an example can be done as follows.

```
sudo -i
```

```
source ~/.ebcl
ebcl-build-arm64 \
    --description /usr/share/kiwi/nautilos/reference-image-description-crinit/nxp \
    --repo-server $A_REPO_SERVER \
    --dist-prefix none \
    --profile RO \
    --local
```

NOTE	Depending on the image setup additional **build** options might be required. For example building images with custom signing and or security keys requires them to be provided and referenced at call time. Details about customization options can be found at: Customization Use Cases

Image Deployment

The result of the **ebcl-build-arm64** call is a binary image that runs on the target as it is. This means the deployment of the image to the target can be done as follows:

```
cd /var/tmp/test-image-embedded-crinit-nxp
sudo dd if=test-image-embedded-crinit-nxp.aarch64-1.0.2.raw of=/dev/SD-CARD
```

Offline Image Building

When building an image the `--repo-server` argument of the `ebcl-build-ARCH` command specifies the

repository endpoint of the Artifactory server within the Elektrobit domain. On an SDK machine which should work completely offline it is therefore required to sync that repositories to the local system first. To run this sync operation, call **ebcl-sync** as follows:

Setup Artifactory access credentials:

```
A_USER=ARTIFACTORY_USER_NAME
```

```
A_SECRET=ARTIFACTORY_SECRET
```

If not already done, download and install the latest `nautilos-repo-setup` package as follows:

```
wget --user $A_USER --password $A_SECRET \
    --recursive --no-directories --no-parent -A 'nautilos-repo-setup*.deb' \
https://artifactory.elektrobit.com/eb_corbos_linux_nautilos2.0-debian-
remote/release/xUbuntu_22.04_debbuild/all/
```

```
sudo dpkg --install $(ls -1t nautilos-repo-setup* | head -n1)
```

Check for available releases as follows:

```
ebcl-releases --user $A_USER --password $A_SECRET
```

Select the release date of your choice and sync the repos to your local system. This process takes some time to complete:

```
sudo ebcl-sync --production YYYY-MM-DD --user $A_USER --password $A_SECRET
```

The `ebcl-sync` command created a local mirror sync of the Artifactory repositories below `/var/tmp/nautilos`. After the operation call `ebcl-repo-setup` as follows:

```
sudo ebcl-repo-setup --local \
    /var/tmp/nautilos/artifactory.elektrobit.com/eb_corbos_linux_nautilos2.0-releases-
generic/YYYY-MM-DD/
```

Reference Example Image Descriptions

There are the following image descriptions provided by EB:

- **reference-image-description-jammy**
  Simple ready to run disk image based on standard Linux components, e.g systemd. Fits into

128M and uses a compressed read-only squashfs root filesystem. Allows for ssh access into a mini home filesystem. It's main purpose is to get started with EBCL.

- **reference-image-description-crinit**
  Based on the `reference-image-description-jammy` image but replaces systemd with crinit as alternative init system. It's main use case is to demo the crinit process manager.
- **reference-image-description-verity**
  Based on the `reference-image-description-crinit`, adding a runtime verity hash check on top of the root device using **dm-verity**. It's main use case is to demo the verity hash device mapper capability on a read-only rootfs
- **reference-image-description-integrity**
  Based on the `reference-image-description-crinit`, but using a writable root filesystem (EXT4 or XFS) which is runtime integrity checked on top of the root device using **dm-integrity**. It's main use case is to demo the integrity hash checking on a read-write rootfs
- **reference-image-description-crypt**
  Based on the `reference-image-description-jammy`, but using an encrypted and integrity checked root filesystem using **LUKS**. It's main use case is to demo full rootfs encryption on a read-write rootfs
- **reference-image-description-container-app**
  Based on the `reference-image-description-verity`, but using a a specific partition layout and tools to serve as immutable container based operating system. More information about the design and tooling can be found at
  Invisible Container Fortress

Customization Use Cases

A KIWI based image description allows users to create all kinds of images with different configurations, features and functionality. In the scope of the automotive industry and with EBCL as the OS platform, there are a number of possible scenarios like how to handle security keys, how to add a container image, how to change the system configuration and more. In the scope of the SDK we support customers with these tasks by providing tools, reference example image descriptions and consulting. For further details how to produce a production ready image please get in Contact with the Elektrobit Linux team

WARNING	Provided example images are subject to change and has no warranty by Elektrobit. New use cases are always open to discussions with Elektrobit.

Contact

Elektrobit Linux team	sales@elektrobit.com