EB Corbos Linux - SDK

Build Target Image(s)

Status	Proposed
Type	User Guide
Author	Marcus Schäfer <marcus.schaefer@elektrobit.com>
Sponsor	Petersohn Jens < Jens.Petersohn@elektrobit.com>
Updated	2022-10-02

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.

Select SDK Setup Method

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

- SDK as OCI container (x86_64, aarch64)
- SDK as AWS EC2 Cloud Instance (x86_64, aarch64)
- SDK on RaspberryPI (aarch64)
- SDK on Ubuntu based build machine (x86_64, aarch64)

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

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)
```

Setup the EBCL repos as follows:

```
sudo ebcl-repo-setup --production --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-get install reference-image-description-crinit
```

Installation: SDK Container

Install EBCL SDK container as follows:

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

Once done continue with Image Building

SDK as AWS EC2 Cloud Instance (x86_64, aarch64)

There are Amazon EC2 images for x86_64 and aarch64 available at: https://ebs.ebgroup.elektrobit.com/package/show/nautilos:2.0:devel:images/nautilos-builder-ec2.
The images can be uploaded and registered as AMI's (Amazon Machine Image). For further details how to register images in EC2 please get in contact with Marcus Schäfer

<marcus.schaefer@elektrobit.com>

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
```

```
sudo ebcl-repo-setup --production --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: http://repos.ebgroup.elektrobit.com/nautilos:/2.0:/devel:/images/images/nautilos-builder-rpi.aarch64.raw.xz on a SD card and boot it up on a **Raspberry Pi 4 Model B** with either 4G (better 8G) of RAM space.

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
```

```
sudo ebcl-repo-setup --production --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 consult EB for further details.

System packages

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

```
sudo apt-get 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 be to be installed, reference example image description. The following steps describes how to install the needed components.

Repository Setup

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)
```

Setup the EBCL repos as follows:

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

Installation: Image Builder

Install EBCL SDK components as follows:

```
sudo apt-get 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-get 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 and how to work with the image description can be found in the Customization Use Cases user guide.

Image Deployment

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

```
sudo -i
source ~/.ebcl
ebcl-sync --user $A_USER --password $A_SECRET
```

The command creates a local mirror sync of the Artifactory repositories below:
`/var/tmp/nautilos/artifactory.elektrobit.com/eb_corbos_linux_nautilos2.0-debian-remote`

After the operation the `--repo-server` argument when building an image can be changed to point to:

```
--repo-server dir:///var/tmp/nautilos/artifactory.elektrobit.com
--dist-prefix eb_corbos_linux_nautilos2.0-debian-remote
```

WARNING Using a local repository mirror disconnects the image builder from updates until **ebcl-sync** is called again to refresh the local copy

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 (WIP)**
  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 (WIP)**
  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 (WIP)**
  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 (WIP)**
  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 is a list of use cases for which we provide instructions how to apply them based on the existing reference example image descriptions. The information covers topics like how to handle security keys, how to add a container image, how to change the system configuration and more.

http://reference_documentation_customer_use_cases

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