Armadillo Base OSセミナー

株式会社アットマークテクノ

www.atmark-techno.com

目次

第１部：Armadillo Base OSについて
第２部：Armadilloの準備
第３部：アプリケーションの作成
第４部：ソフトウェアアップデート設定
第５部：インストールディスクの作成
第６部：参考情報

第３部：アプリケーションの作成

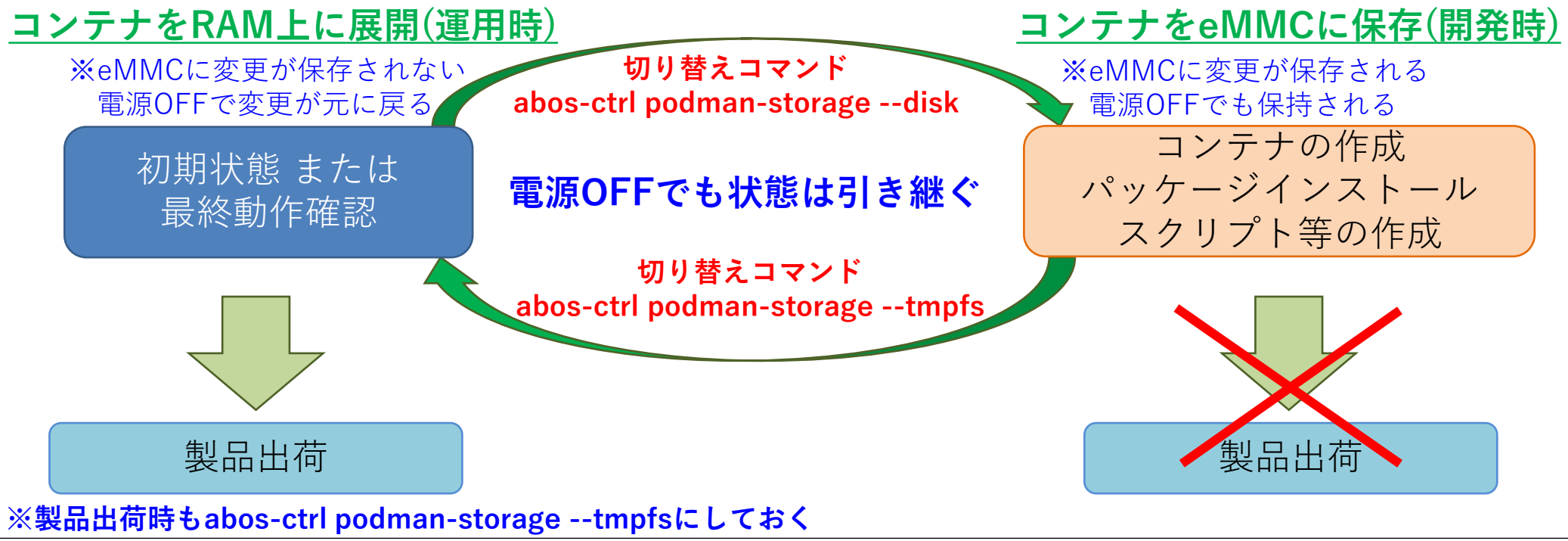
第３部の内容

- 開発時と運用時のコンテナ保存先について
- アプリケーション開発手順

開発時と運用時のコンテナについて

アプリケーション開発時：abos-ctrl podman-storage --diskコマンドを実行
⇒コンテナ及びコンテナイメージをeMMCに保存する設定に変更

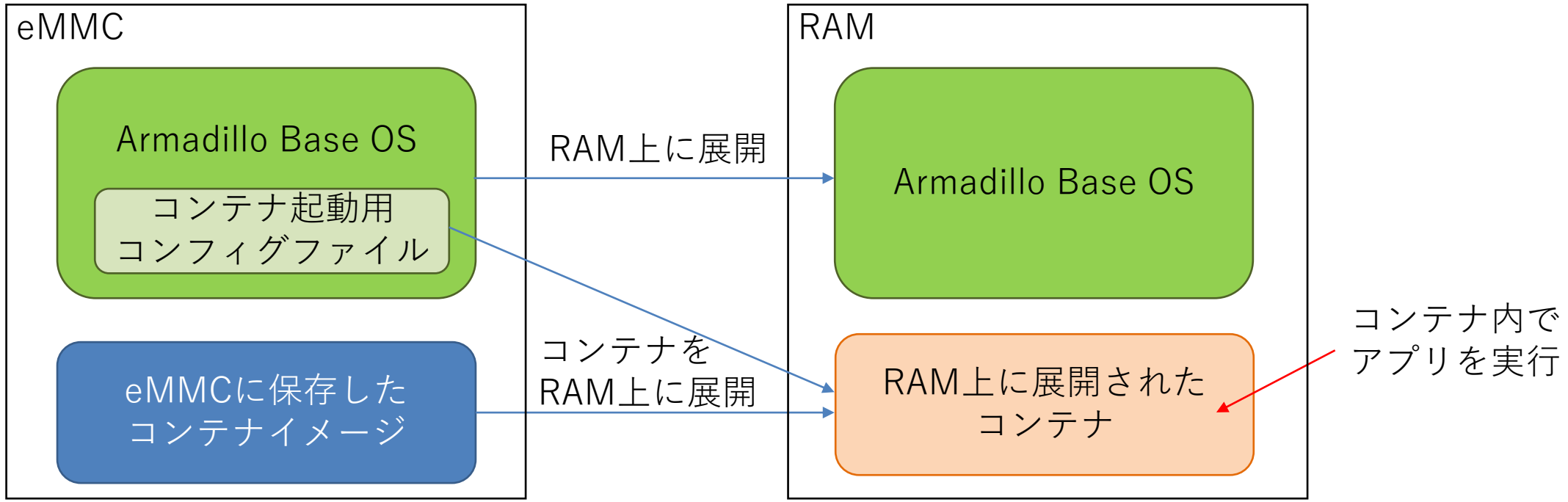
コンテナ完成後の動作確認：abos-ctrl podman-storage --tmpfsコマンドを実行
⇒実運用と同じコンテナをRAM上に保存する設定に変更



※製品出荷時もabos-ctrl podman-storage --tmpfsにしておく

実運用のコンテナの展開イメージ

実運用ではeMMCに保存されたコンテナイメージから、コンテナ起動用のコンフィグファイル(後述)を元にRAM上にコンテナを展開する



アプリケーション開発では、このコンテナイメージとコンフィグファイルを作成する

動作中のコンテナの停止

■アプリケーション作成の前準備①

起動中のコンテナがあるとabos-ctrl podman-storage --diskコマンドが実行出来ない為、
起動中のコンテナが無いか確認
※製品によっては出荷状態でもArmadillo起動時にコンテナが起動するものがあります

起動中のコンテナを確認

```
[armadillo]# podman ps                                          //起動中コンテナの確認
CONTAINER ID  IMAGE                           COMMAND    CREATED       STATUS          PORTS       NAMES
7da471537e94  docker.io/library/debian:bullseye  /bin/bash  46 seconds ago  Up 28 seconds ago           a6e-gw-container
```

上記の様に起動中のコンテナがある場合はそのコンテナを停止

起動中のコンテナ停止

```
[armadillo]# podman stop a6e-gw-container          //実行中コンテナの停止
[armadillo]# podman ps                             //起動中のコンテナ再確認
CONTAINER ID  IMAGE     COMMAND    CREATED    STATUS     PORTS       NAMES
```

ゲートウェイコンテナの停止(A6E)

■アプリケーション作成の前準備②

今回はA6Eのゲートウェイコンテナを使用しない為、自動起動設定をOFFに設定しておく
(既に自動起動をOFFにしている場合は省略)

ゲートウェイコンテナのコンフィグファイルを変更

```
[armadillo]# echo set_autostart no >> /etc/atmark/containers/a6e-gw-container.conf
                                                    //自動起動無効化
[armadillo]# persist_file /etc/atmark/containers/a6e-gw-container.conf
                                                    //eMMCに保存
```

ゲートウェイコンテナのコンフィグを戻す方法

```
[armadillo]# vi /etc/atmark/containers/a6e-gw-container.conf    //ファイル編集
＜中略＞
set_autostart no                                                //この行を削除
[armadillo]# persist_file /etc/atmark/containers/a6e-gw-container.conf
                                                    //eMMCに保存
```

コンテナのeMMC保存設定

■アプリケーション作成の前準備③

アプリケーションを開発する為、コンテナをeMMCに保存する設定に変更
--diskモードではコンテナはeMMCに保存され、電源OFFでも保存される

```
[armadillo]# abos-ctrl podman-storage --status       //現状のステータス確認
Currently in tmpfs mode, run with --disk to switch
[armadillo]# abos-ctrl podman-storage --disk         //eMMC保存に変更
Creating configuration for persistent container storage
Create subvolume '/mnt/containers_storage'
```

RAMに展開する設定に戻す場合は下記コマンドを実行します。
--tmpfsモードではコンテナはRAM上に展開され、電源OFFで消去される

```
[armadillo]# abos-ctrl podman-storage --tmpfs        //RAM保存に変更
Switching back to tmpfs container storage.
```

コンテナ(アプリ)開発手順

■コンテナ作成手順

①コンテナイメージの取得
②コンフィグファイルの作成
　(podman run/docker runコマンドに相当するファイル)
③podman_startでコンテナを新規作成
④コンテナ内でアプリケーションを開発～動作確認
⑤コンテナをコンテナイメージとして保存
⑥コンテナ自動起動設定
⑦コンテナの自動起動確認

本セミナーで作成するコンテナは下記2つ
　コンテナ①：コンテナ外へのファイルアクセス(Read/Write)
　コンテナ②：リアルセミナーのみ製品別に個別課題

アプリの内容

■コンテナ①：コンテナ外へのファイルアクセス(Read/Write)

目的：コンテナ作成の一連の流れを覚える
　　コンテナ外へのファイルアクセス方法を覚える

アプリケーションの内容

Armadillo起動時にスクリプトで現在の日時とArmadillo Base OSのバージョンを記載したファイルを作成・保存する

①Armadillo Base OSのバージョン情報：/etc/atmark-release
※ここでは変更する必要が無い為、Read Onlyとする
コンテナ外のファイルアクセスの為、権限付与が必要

②ファイルの保存先は、/var/app/volumesにversionという名前で保存する
※コンテナ外のファイルアクセスの為、権限付与が必要

《参考情報》

■volumesディレクトリについて

volumesディレクトリはapp領域にあるeMMC保存可能な領域(常時OverlayFS無効)

- /var/app/volumes
  ロールバックの有無に関わらずファイルを保持

- /var/app/rollback/volumes
  ロールバック発生時、ロールバック先の状態(前の状態)に戻る

コンテナはRAM上で動作する為、コンテナ内のデータは電源OFFでクリアされる
電源OFFでもデータを保持したい場合は本ディレクトリを使用する

①コンテナイメージの取得

■コンテナ作成手順

①コンテナイメージの取得

②コンフィグファイルの作成

③コンテナを新規作成

④アプリ開発～動作確認

⑤コンテナをイメージとして保存

⑥コンテナ自動起動設定

⑦コンテナの自動起動確認

①コンテナイメージの取得

■コンテナイメージをDocker Hubから探す

コンテナイメージはDocker Hub(https://hub.docker.com/)から取得可能だが、
使用するArmadilloに合ったアーキテクチャなのか調べる必要がある
本セミナーではDebian 11(bullseye)を使用する為、その手順を記載

検索手順

①Docker HubのURLにアクセス
②右上のExploreをクリックし、上の検索窓で"debian"と入力
③debianが上位に出てくるため、debianのアイコンをクリック
④Tagsを選択し、Filter Tagsに"bullseye"を入力
⑤下の方に”bullseye"があるのでそのDIGESTの下の+moreをクリック
⑥OS/ARCHにArmadilloと同じアーキテクチャがあれば使用可能
G4/X2の場合 ：linux/arm64/v8
A6Eの場合 ：linux/arm/v7

参考ブログ：Armadillo-IoT G4：コンテナイメージの取得方法
https://armadillo.atmark-techno.com/blog/15349/12089

①コンテナイメージの取得

■コンテナイメージの取得

※大阪会場ではネットワーク設定の都合上、下記コマンドが実行出来ません。
　お手数ですが、「command-list_oosaka.txt」のコマンドをご使用下さい。

先ほど確認したコンテナイメージを取得する
OS/ARCHは現在のアーキテクチャに合ったものを自動取得できる

```
[armadillo]# podman pull docker.io/debian:bullseye
```

ATDE環境の様なArmadilloと異なるアーキテクチャの場合は、下記の様に指定する事で指定したアーキテクチャとして取得できる（下記はarm64/v8の場合）

```
[armadillo]# podman pull docker.io/arm64v8/debian:bullseye
```

■コンテナイメージの確認

取得したコンテナイメージはpodman imagesコマンドで確認

```
[armadillo]# podman images
REPOSITORY                  TAG       IMAGE ID     CREATED     SIZE
docker.io/library/debian  bullseye    54a149f6821e  2 days ago  123 MB
```

②コンフィグファイルの作成

■コンテナ作成手順

①コンテナイメージの取得
②コンフィグファイルの作成
③コンテナを新規作成
④アプリ開発～動作確認
⑤コンテナをイメージとして保存
⑥コンテナ自動起動設定
⑦コンテナの自動起動確認

②コンフィグファイルの作成

Atmark Techno

■コンフィグファイルに記載する情報(コンテナ開発時)

・使用するコンテナイメージを指定(必須項目)

⇒set_image debian:bullseye

・アクセス権限を与えるファイルやディレクトリを指定(任意項目)

⇒add_volumes /etc/atmark-release:/etc/atmark-release:ro
add_volumes /var/app/volumes:/root/test

コロン":"の左側がArmadillo Base OS領域のファイル/ディレクトリパス
右側がコンテナ内のファイル/ディレクトリパス
右端の":ro"はRead Only

・その他オプション等(開発時は必須項目)

⇒set_args -it
set_autostart no

・コンテナ内での実行コマンド(必須項目)

⇒set_command /bin/bash

②コンフィグファイルの作成

■コンフィグファイルの作成場所

コンフィグファイルはArmadilloの/etc/atmark/containers/に作成する
ここでは/etc/atmark/containers/example1.confを作成する
※example1.confの場合、コンテナ名はexample1となる

コンフィグファイルの書式は次頁または下記ブログを参照
参考ブログ：Armadillo Base OS：コンテナの自動起動とconfファイルの説明（改訂版）
https://armadillo.atmark-techno.com/index.php/blog/15349/12098

《参考》コンフィグファイル書式

Atmark Techno

■コンフィグファイルの書式一覧

設定項目	デフォルト値	説明	設定例
set_image	無し（省略不可）	コンテナイメージの設定	set_image docker.io/alpine:latest
add_volumes	無し	マウント設定、--volumeオプションと同じ ro(read-only), nodev, nosuid,noexec, shared, slave設定可能	add_volumes /run/udev:/run/udev:ro /opt/firmware:/opt/firmware
add_devices	無し	デバイス設定、--deviceと同じ	add_devices /dev/tty1 /dev/input /dev/dri /dev/galcore
add_ports	無し	ポート設定、--publishと同じ、/udpでUDP設定も可能	add_ports 80:80 443:443 2222:22/udp
set_readonly	no	読み取り専用オプション、--read-onlyと同じ	set_readonly yes
set_pull	no	imageをダウンロードして使用する(missingは無い場合のみ)	set_pull always , set_pull missing
set_autostart	yes	自動起動不要な場合はno、自動起動時同名のコンテナは上書き	set_autostart no
set_restart	on-failure	コンテナ再起動設定	set_restart always , set_restart no
set_pod	無し	pod設定	set_pod mypod
set_network	無し	ネットワークの選択	set_network mynetwork , set_network none , set_network host
set_ip	無し	コンテナのIPアドレス設定	set_ip 10.88.0.100
add_args	無し	その他オプション設定	add_args --cap-add=SYS_TTY_CONFIG --env=XDG_RUNTIME_DIR=/tmp --net=host
set_command	bash	コンテナ起動時のコマンド設定	set_command python3 read_meter.py
set_infra_imager	無し	podのインフラコンテナイメージ	set_infra_imager k8s.gcr.io/pause:3.5
set_subnet	無し	サブネットの設定	set_subnet 192.168.100.0/24

②コンフィグファイルの作成

■コンフィグファイル作成（コンテナ作成用）

”example1”コンテナを作成する為のコンフィグファイルをvi等で作成

```
[armadillo]# vi /etc/atmark/containers/example1.conf        //ファイル作成
```

example1.conf

```
set_image debian:bullseye                                   //使用イメージ
add_volumes /etc/atmark-release:/etc/atmark-release:ro      //マウント設定
add_volumes /var/app/volumes:/root/test                     //マウント設定
add_args -it                                                //開発時は必須
set_autostart no                                            //開発時は必須
set_command /bin/bash                                       //開発時は必須
```

作成したコンフィグファイルをeMMCに保存

```
[armadillo]# persist_file /etc/atmark/containers/example1.conf   //eMMCに保存
```

③コンテナを新規作成

■コンテナ作成手順

①コンテナイメージの取得
②コンフィグファイルの作成
③コンテナを新規作成
④アプリ開発～動作確認
⑤コンテナをイメージとして保存
⑥コンテナ自動起動設定
⑦コンテナの自動起動確認

③コンテナを新規作成

■コンテナ作成コマンド

コンテナイメージからコンテナを新規作成するにはpodman_startコマンドを実行

```
[armadillo]# podman_start example1                    //コンテナ作成～起動
Starting 'example1'
0a68053d4a3d8a5f1105aaf73d4aa170862956a0f28734bae33e1477022dac19
```

《注意》
podman_startコマンドは初回コンテナを作る際と、コンテナを自動起動する場合のみ使用します。
作成済みコンテナを起動する場合はpodman start(アンダーバー無し)で起動する様にしてください。

podman ps -aコマンドでコンテナが動いている事を確認(STATUSがUpで動作中)

```
[armadillo]# podman ps -a
CONTAINER ID  IMAGE                                COMMAND    CREATED       STATUS           PORTS       NAMES
0a68053d4a3d  docker.io/library/debian:bullseye  /bin/bash   6 seconds ago  Up 6 seconds ago             example1
```

次のコマンドを実行して起動中のコンテナ内に入る

```
[armadillo]# podman attach example1        //起動中のコンテナに入るコマンド
```

④アプリ開発～動作確認

■コンテナ作成手順

①コンテナイメージの取得
②コンフィグファイルの作成
③コンテナを新規作成
④アプリ開発～動作確認
⑤コンテナをイメージとして保存
⑥コンテナ自動起動設定
⑦コンテナの自動起動確認

④アプリ開発～動作確認

■コンテナの確認

最初に、コンテナ作成時に設定した下記のアクセス権限について確認する

確認内容

・コンテナ外の/etc/atmark-releaseが読み取れること
・コンテナ外の/var/app/volumesにマウントした/root/testディレクトリが作成されている事

```
[container]# cd                                    //rootディレクトリに移動
[container]# cat /etc/atmark-release               //atmark-releaseの読み取り確認
3.16.1-at.2                                        //バージョンは適宜読み替えてください
[container]# ls -l                          //volumeにマウントしたtestディレクトリの確認
total 0
drwxr-xr-x 1 root root 14 Aug 26 05:03 test
[container]# ls -l test
total 0                                     //volumesにファイルが無い場合は表示無し
```

上記問題が無ければ、次にアプリケーションを実装する

④アプリ開発～動作確認

■パッケージインストール～スクリプト作成

コンテナには最低限のパッケージしかインストールされていない為、開発に必要なパッケージをインストールする必要がある。ここではviエディタをインストールし、スクリプトを作成する(catコマンド等で作成でも可)

```
[container]# apt update && apt upgrade -y
[container]# apt install -y vim
[container]# vi check.sh
```

※TeraTermの場合、右クリックでは張り付けられない為、
ここでは Alt+V で貼り付けます。

check.shの中身

```
#!/bin/bash

date=`date`
ver=`cat /etc/atmark-release`
echo -e "$date¥¥n$ver" >> /root/test/version
```

④アプリ開発～動作確認

■スクリプト実行～結果確認

作成したスクリプトは実行権限が無い為、実行権限付与してスクリプトを実行

```
[container]# chmod +x check.sh
[container]# ./check.sh
```

/root/testディレクトリの中にversionという下記内容のファイルが作成されていれば成功

```
[container]# cat test/version
Fri Aug 26 06:07:58 UTC 2022    //コンテナはデフォルトではUTC表記
3.16.1-at.2                     //JST表記の方法は第6部のTips集を参照
```

このversionファイルがコンテナ外の/var/app/volumes/versionと同じことを確認する
下記ファイルがあればコンテナ①の開発は完了

```
[container]# exit                                   //コンテナを抜ける(コンテナ停止)
[armadillo]# cat /var/app/volumes/version           //同じファイルがeMMCに保存
Fri Aug 26 06:07:58 UTC 2022                        //されているか確認する
3.16.1-at.2
```

《補足事項》

■その他コンテナ関連のコマンド

作成したコンテナの確認

```
[armadillo]# podman ps -a     //全てのコンテナを表示
[armadillo]# podman ps        //起動中のコンテナのみ表示
```

"作成済み"のコンテナ起動方法と停止方法

```
[armadillo]# podman start example1          //コンテナ起動
[armadillo]# podman start -ai example1      //コンテナ起動+コンテナに入る
[armadillo]# podman attach example1         //起動中のコンテナに入る
[armadillo]# podman stop example1           //起動中のコンテナを停止
```

コンテナ外部から起動中のコンテナ内部のスクリプトを実行

```
[armadillo]# podman exec -it example1 ./root/check.sh
```

⑤コンテナをイメージとして保存

■コンテナ作成手順

①コンテナイメージの取得
②コンフィグファイルの作成
③コンテナを新規作成
④アプリ開発～動作確認
⑤コンテナをイメージとして保存
⑥コンテナ自動起動設定
⑦コンテナの自動起動確認

⑤コンテナをイメージとして保存

■コンテナをコンテナイメージで保存

作成したコンテナをコンテナイメージとして保存する
ここではコンテナ"example1"をコンテナイメージ名"image_example1"
タグ名を"v1.0.0"とする

```
[armadillo]# podman ps -a
CONTAINER ID  IMAGE                                COMMAND     CREATED       STATUS                   PORTS       NAMES
e4ec31724ec1  docker.io/library/debian:bullseye  /bin/bash      25 minutes ago  Exited (0) 16 minutes ago             example1
```

```
[armadillo]# podman commit example1 image_example1:v1.0.0
[armadillo]# podman images
REPOSITORY                  TAG         IMAGE ID      CREATED         SIZE
localhost/image_example1    v1.0.0      96400f71538d  7 seconds ago   123 MB
docker.io/library/debian    bullseye    54a149f6821e  2 days ago      123 MB
```

⑤コンテナをイメージとして保存

■コンテナイメージを外部ストレージにバックアップ保存

保存したコンテナイメージは外部ストレージにバックアップを推奨
コンテナイメージはサイズが大きい事が予想される為、直接外部ストレージに保存を推奨

ここでは保存するアーカイブ名を"example1_image_v1.0.0.tar"とし、外部ストレージを/dev/sda1の場合として記載

```
[armadillo]# mount /dev/sda1 /mnt                                    //sda1は適宜変更
[armadillo]# podman save -o /mnt/example1_image_v1.0.0.tar ￥
image_example1:v1.0.0
[armadillo]# umount /mnt
```

保存したバックアップを使用する場合は下記のloadコマンドを実行する
※イメージ名やタグ名は保存前と同じ名前

```
[armadillo]# mount /dev/sda1 /mnt                                    //sda1は適宜変更
[armadillo]# podman load -i /mnt/example1_image_v1.0.0.tar
[armadillo]# umount /mnt
```

⑥コンテナ自動起動設定

■コンテナ作成手順

①コンテナイメージの取得
②コンフィグファイルの作成
③コンテナを新規作成
④アプリ開発～動作確認
⑤コンテナをイメージとして保存
⑥コンテナ自動起動設定
⑦コンテナの自動起動確認

⑥コンテナ自動起動設定

■コンフィグファイルの変更（自動起動用）

コンフィグファイルを完成したコンテナイメージを使用して、自動起動する設定に変更する

```
[armadillo]# vi /etc/atmark/containers/example1.conf       //ファイルを編集
```

example1.conf

```
set_image localhost/image_example1:v1.0.0                   //保存したイメージ名に変更
add_volumes /var/app/volumes:/root/test
add_volumes /etc/atmark-release:/etc/atmark-release:ro
#add_args -it                                               //コメントアウトor削除
#set_autostart no                                           //コメントアウトor削除
set_command ./root/check.sh                                 //スクリプト実行に変更
```

変更したら永続化する

```
[armadillo]# persist_file /etc/atmark/containers/example1.conf    //変更を永続化
```

⑥コンテナ自動起動設定

■コンテナの削除

作成したコンテナをコンテナイメージに保存が完了したら、不要なコンテナを削除する

```
[armadillo]# podman rm example1                          //コンテナを削除
[armadillo]# podman ps -a
CONTAINER ID  IMAGE                 COMMAND     CREATED     STATUS                 PORTS     NAMES
                                                                                   //コンテナが無い事を確認
```

⑦コンテナの自動起動確認

■コンテナ作成手順

①コンテナイメージの取得
②コンフィグファイルの作成
③コンテナを新規作成
④アプリ開発～動作確認
⑤コンテナをイメージとして保存
⑥コンテナ自動起動設定
⑦コンテナの自動起動確認

⑦コンテナの自動起動確認

■コンテナをRAM上に展開する設定への切り替え

下記コマンドでコンテナをRAMに展開する設定に切り替える
切り替え時に下記メッセージが出た場合は、Cを入力する

```
[armadillo]# abos-ctrl podman-storage --tmpfs
There are images configured on development storage:
REPOSITORY                TAG         IMAGE ID      CREATED             SIZE
localhost/image_example1  v1.0.0      d213cdea3978  About a minute ago  123 MB
docker.io/library/debian  bullseye    59ed139d178b  2 weeks ago         123 MB

What should we do? ([C]opy (default), [N]othing, [D]elete)
C                                          //ここではCを選択（Nは移行しない、Dはイメージ削除）
```

```
[armadillo]# podman images
REPOSITORY                TAG         IMAGE ID      CREATED             SIZE       R/O
localhost/image_example1  v1.0.0      6ff88b29c281  About a minute ago  123 MB     true
docker.io/library/debian  bullseye    59ed139d178b  2 weeks ago         123 MB     true
```

⑦コンテナの自動起動確認

■コンテナの自動起動確認

下記コマンドでコンテナを起動して動作確認

```
[armadillo]# podman_start example1
Starting 'example1'
1c84105e8ec7a5f481ef0d018c7a8338c3f8e6402b0e3b09fd749fede61b8b39
```

起動してスクリプトを実行するだけのコンテナの為、スクリプトが終了すると停止する

```
[armadillo]# podman ps -a
CONTAINER ID  IMAGE                              COMMAND       CREATED         STATUS                     PORTS       NAMES
1c84105e8ec7  localhost/image_example1:v1.0.0  ./root/check.sh  5 seconds ago  Exited (0) 6 seconds ago             example1
```

下記ファイルが最新の日付(UTC表記)で追加書き込みされていれば成功

```
[armadillo]# cat /var/app/volumes/version
Fri Aug 26 06:07:58 UTC 2022
3.16.1-at.2
Fri Aug 26 06:17:21 UTC 2022
3.16.1-at.2
```

⑦コンテナの自動起動確認

■コンテナの自動起動確認

最後に、Armadilloをリブートして自動起動で問題が無いかを確認する
Armadillo起動後/var/app/volumes/versionが現在の時間で追加書き込みされている事を
確認して開発完了となる

```
[armadillo]# reboot
[armadillo]# cat /var/app/volumes/version          //再起動後に確認
```

これでコンテナ①の開発は完了です。

《備考》
NTPサーバーとの同期が遅れ、コンテナの起動タイミングが先になるとファイルの
日時が1970年1月1日になる事があります。
その場合、コンテナ内で時刻取得のタイミングを遅らせるか、NTPサーバ同期後に
コンテナ起動するなどで回避できます。

コンテナ作成課題

コンテナ作成課題：A6E

■間欠動作をするコンテナを作成する

目的：A6Eの間欠動作方法を習得(SLEEP⇔ACTIVE)
※間欠動作で消費電力を低減（Cat.M1モデルSLEEP時：約120mW）

コンテナ動作
①コンテナ起動後、下記内容のスクリプトを実行
・DI1がLowを検出するまで待機（APPのLEDを消灯）
・DI1がLowを検出するとコンテナ終了（APPのLEDを点灯）
②コンテナ終了後、A6EをSLEEPモードに移行
③SW1押下でA6EがACTIVEになり、コンテナを起動

間欠動作のトリガー
・コンテナ終了(SLEEPモード移行)：DI1がLowになった時
・SLEEPからの復帰(コンテナ実行)：SW1押下した時

《作成方法》
まずは①を作り、最後に②③はP.50-51の設定を行う

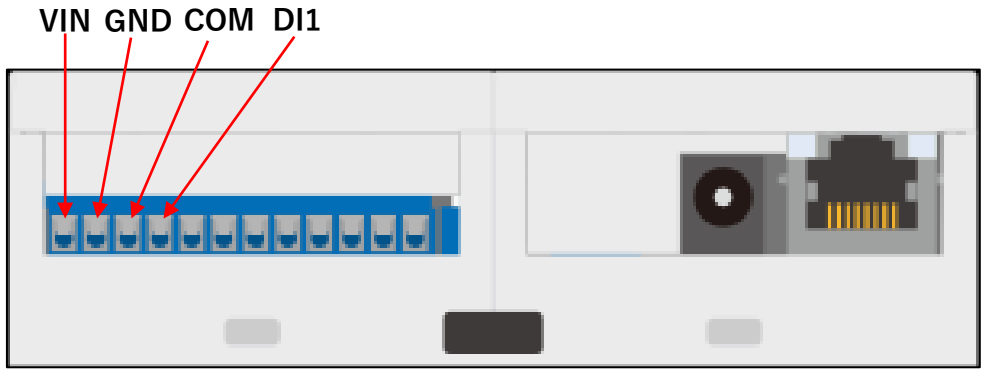
配線図

配線作業は安全を考慮し、Armadilloをpoweroffしてシャットダウンした後、
ACアダプターを抜いた状態で実施下さい。配線が完了したら電源投入します。

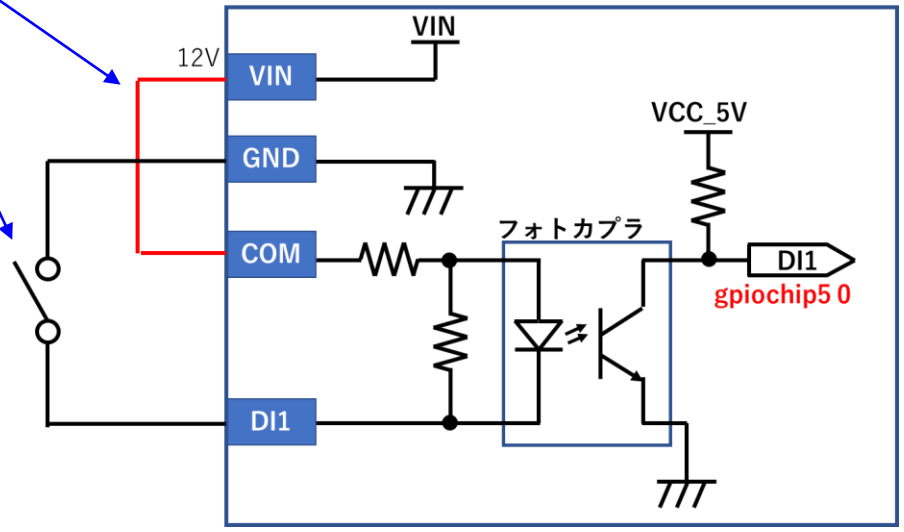
```
[armadillo]# poweroff                                    //パワーオフ
```

この様に配線する
※VINとGNDショートには注意

スイッチON時は配線をショート



Armadillo-IoT A6E(端子台側)



Armadillo-IoT A6E　DI部(簡略図)

コンテナ作成手順

■コンテナ作成手順

①コンテナイメージの取得
②コンフィグファイルの作成
③コンテナを新規作成
④アプリ開発～動作確認
⑤コンテナをイメージとして保存
⑥コンテナ自動起動設定
⑦コンテナの自動起動確認

コンテナ関連

■コンテナの保存先設定を変更

```
[armadillo]# abos-ctrl podman-storage --disk        //eMMC保存に変更
Creating configuration for persistent container storage
Create subvolume '/mnt/containers_storage'
```

①コンテナイメージの取得

※大阪会場ではP.15で取得した下記のイメージを使用します。
使用するコンテナイメージ：alpine:latest

使用するコンテナイメージに特に制限はないが、ここでは最小サイズのイメージである
Alpineのコンテナイメージを使用する

■Alpineコンテナイメージを取得

```
[armadillo]# podman pull docker.io/alpine:latest     //コンテナイメージ取得
[armadillo]# podman images
```

コンテナ関連

②コンフィグファイルの作成

■コンフィグファイル作成(コンテナ作成用)

```
[armadillo]# vi /etc/atmark/containers/example2.conf    //ファイル作成
```

example2.conf

```
set_image alpine:latest                                  //使用イメージ
add_devices /dev/gpiochip5                               //DI1のデバイス
add_volumes /sys                                         //LED制御用
add_args -it                                             //デバッグ用
set_autostart no                                         //デバッグ用
set_command /bin/sh                                      //bashがないのでsh
```

■コンフィグファイルの永続化

```
[armadillo]# persist_file /etc/atmark/containers/example2.conf
```

コンテナ関連

③コンテナを新規作成

■コンテナの新規作成～コンテナに入る

```
[armadillo]# podman_start example2
[armadillo]# podman attach example2
```

④アプリ開発～動作確認

ここからコンテナ内で作業

■コンテナの中でパッケージインストール

```
[container]# apk update && apk upgrade && apk add libgpiod vim
```

libgpiod　・・・GPIO制御用パッケージ

スクリプト例

■スクリプト作成

```
[container]# cd                                  // /root/ディレクトリに移動
[container]# vi input.sh
```

input.sh

```
#!/bin/sh

echo 0 > /sys/class/leds/app/brightness      //LED(APP)をOFF

while :
do
 result=$(gpioget gpiochip5 0)                //gpiochip5 0の値を取得
 if [ ${result} = 0 ]; then                   //result=0 (DIがLowの場合)
  echo 1 > /sys/class/leds/app/brightness     //LED(APP)をON
  break                                       //LOOPを抜ける
 fi
done
```

コンテナ関連

■スクリプト動作確認

```
[container]# chmod +x input.sh
[container]# ./input.sh
```

・input.sh実行後、LED(APP)が消灯（スクリプトは継続中）
・DI1をGNDとショートさせるとLED(APP)が点灯(スクリプトが終了)

上記確認ができればスクリプトは確認は完了

動作確認が完了したらコンテナから出る

```
[container]# exit
```

コンテナ関連

⑤コンテナをイメージとして保存

■コンテナイメージ保存

```
[armadillo]# podman ps -a
[armadillo]# podman commit example2 image_example2:v1.0.0
[armadillo]# podman images
```

※必要に応じてバックアップ保存(P.30参照)

■不要なコンテナ、コンテナイメージ削除

```
[armadillo]# podman ps -a
[armadillo]# podman rm example2
[armadillo]# podman rmi alpine:latest
```

コンテナ関連

⑥コンテナ自動起動設定

■コンフィグファイル修正(自動起動用)

```
[armadillo]# vi /etc/atmark/containers/example2.conf
```

example2.conf

```
set_image image_example2:v1.0.0                    //イメージ名を変更
add_devices /dev/gpiochip5
add_volumes /sys
#add_args -it                                      //コメントアウトor削除
#set_autostart no                                  //コメントアウトor削除
set_command /bin/sh /root/input.sh                 //スクリプト実行に変更
```

■コンフィグファイルの永続化

```
[armadillo]# persist_file /etc/atmark/containers/example2.conf
```

コンテナ関連

⑦コンテナの自動起動確認

■コンテナの保存先変更

```
[armadillo]# abos-ctrl podman-storage --tmpfs                    //選択肢はCを入力
```

■自動起動の動作確認

```
[armadillo]# podman_start example2
```

・podman_start example2実行後、podman psでコンテナが動作中か確認
・DI1をGNDとショートさせるとコンテナが終了している事を確認
※再度確認する場合はもう一度podman_startを実行

《注意》上記が正常に動作している事を確認してから次に進んでください。

コンテナ終了/復帰設定

■最後に下記設定を実施する

②コンテナ終了後、A6EをSLEEPモードに移行
③SW1押下でA6EがACTIVEになり、コンテナを起動
⇒power-utils.confを編集する事で設定可能

■ power-utils.confを編集する

```
[armadillo]# vi /etc/atmark/power-utils.conf
```

power-utils.conf

```
TARGET='example2'                    //操作するコンテナ名
MODE='SLEEP'                         //コンテナ終了時の動作
WAKEUP='SW1'                         //コンテナ復帰時のトリガー
```

■ power-utils.confを永続化する

```
[armadillo]# persist_file /etc/atmark/power-utils.conf
```

コンテナ終了/復帰設定

■コンフィグファイル修正(自動起動＋コンテナ終了検知用)

```
[armadillo]# vi /etc/atmark/containers/example2.conf
```

example2.conf

```
set_image image_example2:v1.0.0
add_devices /dev/gpiochip5
add_volumes /sys
#add_args -it
#set_autostart no
set_command /bin/sh /root/input.sh
add_args --hooks-dir=/etc/containers/aiot_gw_container_hooks.d    //追加
```

コンテナ終了時のトリガー検出

add_args --hooks-dir=/etc/containers/aiot_gw_container_hooks.d
※コンテナ終了を検出してSLEEPモードに移行する為に使用

■コンフィグファイルの永続化

```
[armadillo]# persist_file /etc/atmark/containers/example2.conf
```

間欠動作の動作確認

■動作確認

```
[armadillo]# podman_start example2
```

①DI1をLowに設定　⇒　LEDのAPPが点灯した後、ArmadilloがSLEEPに入る
（全LEDが消え、コンソールが無効）

②SW1を押下する　⇒　SLEEPから復帰(コンテナ実行)
（LEDのSYSが点灯、APPが消灯、コンソールが有効）

以上で間欠動作するコンテナ作成は完了です。

間欠動作から抜ける作業

動作確認が終了後、デバッグを継続する場合は以下の手順を行っておく
※コンテナを終了させるとSLEEPに入ってしまい、コマンドを受け付けない為

①コンテナを起動しておく（SLEEPに入る前の状態にする）
②起動中にpower-utils.confのMODEをSLEEPからNONEに変更
③power-utils.confを永続化しておく

```
[armadillo]# vi /etc/atmark/power-utils.conf
TARGET='example2'
MODE='NONE'                                          //コンテナ終了時の動作を変更
WAKEUP='SW1'
[armadillo]# persist_file /etc/atmark/power-utils.conf
```

《参考》ゲートウェイコンテナ設定

冒頭に無効化したゲートウェイコンテナの設定を元に戻す場合は、以下の様に
コンフィグファイルを開いてset_autostart noの行を削除する
(インストールディスクで初期化も可)

ゲートウェイコンテナのコンフィグを戻す

```
[armadillo]# vi /etc/atmark/containers/a6e-gw-container.conf    //ファイル編集
＜中略＞
set_autostart no                                                //この行を削除
[armadillo]# persist_file /etc/atmark/containers/a6e-gw-container.conf
                                                                //eMMCに保存
```

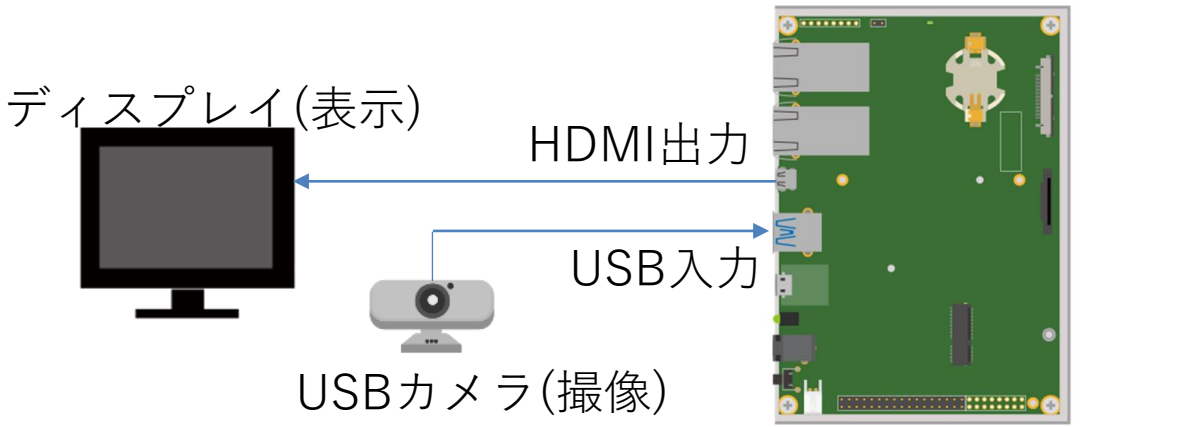
コンテナ作成課題：G4/X2

■カメラを使って録画とディスプレイ表示するコンテナを作成する

目的：VPUを使用する方法を習得
　　カメラ画像の取得と表示方法を習得

コンテナ動作
gstreamerを使用してUSB カメラからの映像をHDMI表示＋H.264 で録画(音声無し)
録画ファイルは/var/app/volumes/に保存

■追加のデバイスを接続（右図）
・USBカメラ
・ディスプレイ



Armadillo-IoT G4の例

コンテナ作成手順

■コンテナ作成手順

①コンテナイメージの取得
②コンフィグファイルの作成
③コンテナを新規作成
④アプリ開発～動作確認
⑤コンテナをイメージとして保存
⑥コンテナ自動起動設定
⑦コンテナの自動起動確認

コンテナ作成課題：G4/X2

■コンテナの保存先設定を変更

```
[armadillo]# abos-ctrl podman-storage --disk        //eMMC保存に変更
Creating configuration for persistent container storage
Create subvolume '/mnt/containers_storage'
```

①コンテナイメージの取得

今回はアットマークテクノのコンテナイメージを使用する

- westonをインストール済み（設定含む）
- VPUやNPUを使用出来る

※大阪会場ではP.15で取得した下記のイメージを使用します。

使用するコンテナイメージ：at-debian-image:latest

```
[armadillo]# podman load -i https://armadillo.atmark-techno.com/files/¥
downloads/armadillo-iot-g4/container/at-debian-image-v1.0.8.tar
[armadillo]# podman images
```

コンテナ関連

②コンフィグファイルの作成

■コンフィグファイル（コンテナ作成時）

```
[armadillo]# vi /etc/atmark/containers/example2.conf
```

example2.conf

```
set_image localhost/at-debian-image:latest                  //使用イメージ
add_devices /dev/dri   /dev/galcore                         //画面描画用
add_devices /dev/mxc_hantro  /dev/mxc_hantro_vc8000e        //VPU用
add_devices /dev/ion                                        //VPU用
add_devices /dev/input   /dev/tty7                          //入出力用
add_hotplugs video4linux                                    //USBカメラ(hotplug対応)
add_volumes /var/app/volumes:/root/rec                      //録画ファイル保存先
add_volumes /run/udev:/run/udev:ro                          //デバイス管理
add_volumes /opt/firmware:/opt/firmware:ro                  //ファームウェア
add_args --env=XDG_RUNTIME_DIR=/run/xdg_home                //westonの設定
add_args --cap-add=SYS_TTY_CONFIG                           //tty操作権限
set_autostart no                                            //開発時は必須
add_args -it                                                //開発時は必須
set_command /bin/bash                                       //開発時は必須
```

コンテナ関連

■コンフィグファイルの永続化

```
[armadillo]# persist_file /etc/atmark/containers/example2.conf
```

③コンテナを新規作成

■コンテナの起動～コンテナに入る

```
[armadillo]# podman_start example2
[armadillo]# podman attach example2
```

④アプリ開発～動作確認

■パッケージインストール

```
[container]# apt update && apt install -y gstreamer1.0-imx ¥
gstreamer1.0-imx-tools gstreamer1.0-tools gstreamer1.0-plugins-good ¥
gstreamer1.0-plugins-bad vim
```

コンテナ関連

■スクリプト作成

```
[container]# cd                                          // /rootディレクトリに移動
[container]# vi camera.sh
```

camera.sh

```
#!/bin/bash

weston --tty=7 &                                         //weston起動(tty=7を指定)
sleep 10                                                 //10sec待機

gst-launch-1.0 -e v4l2src device=/dev/video2 ¥           //gstreamer起動
! video/x-raw,width=640,height=480,framerate=30/1 ¥      //カメラ設定
! tee name=t1 ! queue ! vpuenc_h264 ! h264parse ! queue ¥
! qtmux ! filesink location=/root/rec/output.mp4 t1. ! queue ¥
! waylandsink window-width=1920 window-height=1080      //表示設定
```

※カメラの解像度やフレームレートはカメラの仕様によって適宜変更します。

コンテナ関連

■動作確認

```
[container]# chmod +x camera.sh
[container]# ./camera.sh
```

スクリプト実行後、画面が灰色になる（Weston起動）
10秒後、USBカメラの画像が表示される

終了する場合はCtrl+C

```
[container]# ls rec
output.mp4                                    //録画ファイルがある事を確認
```

動作確認完了後、コンテナから抜ける

```
[container]# exit
```

コンテナイメージの保存

⑤コンテナをイメージとして保存

■コンテナをコンテナイメージとして保存

```
[armadillo]# podman ps -a
[armadillo]# podman commit example2 image_example2:v1.0.0
[armadillo]# podman images
```

※必要に応じてバックアップ保存(P.30参照)

■不要なコンテナ、コンテナイメージ削除([version]は適宜調整)

```
[armadillo]# podman ps -a
[armadillo]# podman rm example2
[armadillo]# podman rmi at-debian-image:latest
[armadillo]# podman rmi at-debian-image:[version]
```

自動起動設定

⑥コンテナ自動起動設定

■コンフィグファイル（コンテナ完成後）

```
[armadillo]# vi /etc/atmark/containers/example2.conf
```

example2.conf

```
set_image localhost/image_example2:v1.0.0                    //使用イメージ変更
add_devices /dev/dri   /dev/galcore
add_devices /dev/mxc_hantro  /dev/mxc_hantro_vc8000e
add_devices /dev/ion
add_devices /dev/input   /dev/tty7
add_hotplugs video4linux
add_volumes /var/app/volumes:/root/rec
add_volumes /run/udev:/run/udev:ro
add_volumes /opt/firmware:/opt/firmware:ro
add_args --env=XDG_RUNTIME_DIR=/run/xdg_home
add_args --cap-add=SYS_TTY_CONFIG
#set_autostart no                                            //コメントアウトor削除
#add_args -it                                                //コメントアウトor削除
set_command /bin/bash /root/camera.sh                        //スクリプト実行に変更
```

■コンフィグファイルの永続化

```
[armadillo]# persist_file /etc/atmark/containers/example2.conf
```

自動起動確認

⑦コンテナの自動起動確認

■コンテナの保存先変更

```
[armadillo]# abos-ctrl podman-storage --tmpfs          //選択肢はCを入力
```

■自動起動の動作確認

```
[armadillo]# podman_start example2
[armadillo]# podman stop example2                      //コンテナ終了
```

・カメラ画像がディスプレイに表示されている事を確認
・コンテナ終了後、録画ファイルが/var/app/volumesに保存されている事を確認

■rebootして自動起動を確認(接続はそのまま)

```
[armadillo]# reboot
```

自動起動で動作すればコンテナ作成は完了

《参考》録画ファイルの再生

動画再生コンテナ

■camera.shを下記に変更すると、本セミナーで録画したファイルを再生できます。

```
[container]# vi camera.sh
```

camera.sh

```
#!/bin/bash

weston --tty=7 &
sleep 10

gst-launch-1.0 filesrc location=rec/output.mp4 ¥
! qtdemux ! h264parse ! vpudec ! queue ! ¥
waylandsink window-width=1920 window-height=1080
```

第３部まとめ

- 開発時と運用時のコンテナ保存先について
  - ・コンテナ保存先の切り替え
  - ・実運用でのコンテナの展開

- アプリケーション開発手順
  - ・コンフィグファイル作成
  - ・コンテナ作成
  - ・アプリケーション開発～動作確認
  - ・自動起動確認