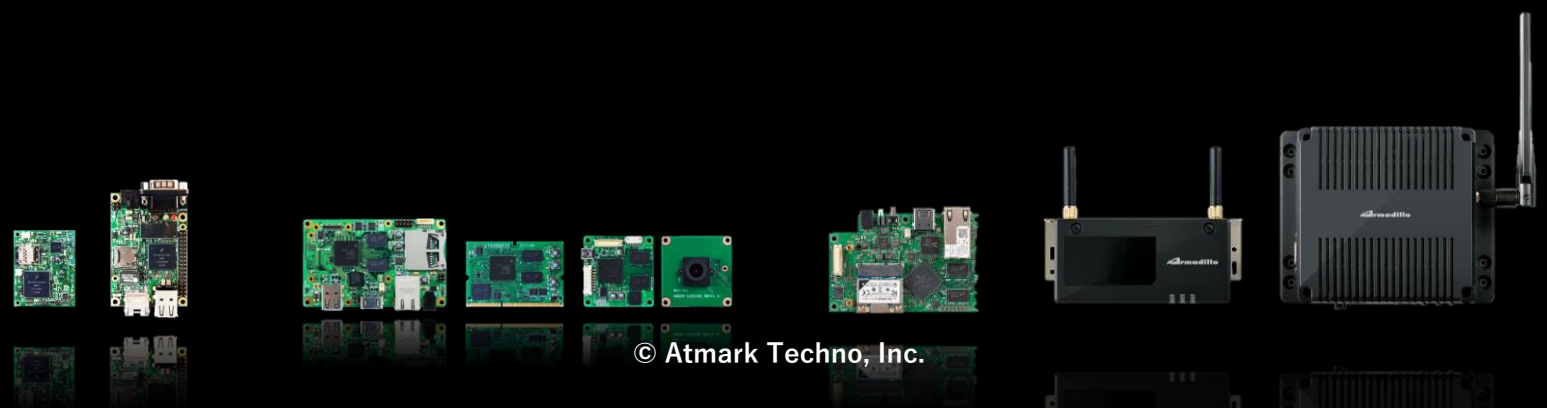
Armadillo-IoT G4 発売記念！Edge AI作品開発コンテスト meet up！

物体認識ハンズオン

株式会社アットマークテクノ

www.atmark-techno.com

ハンズオン



ハンズオン

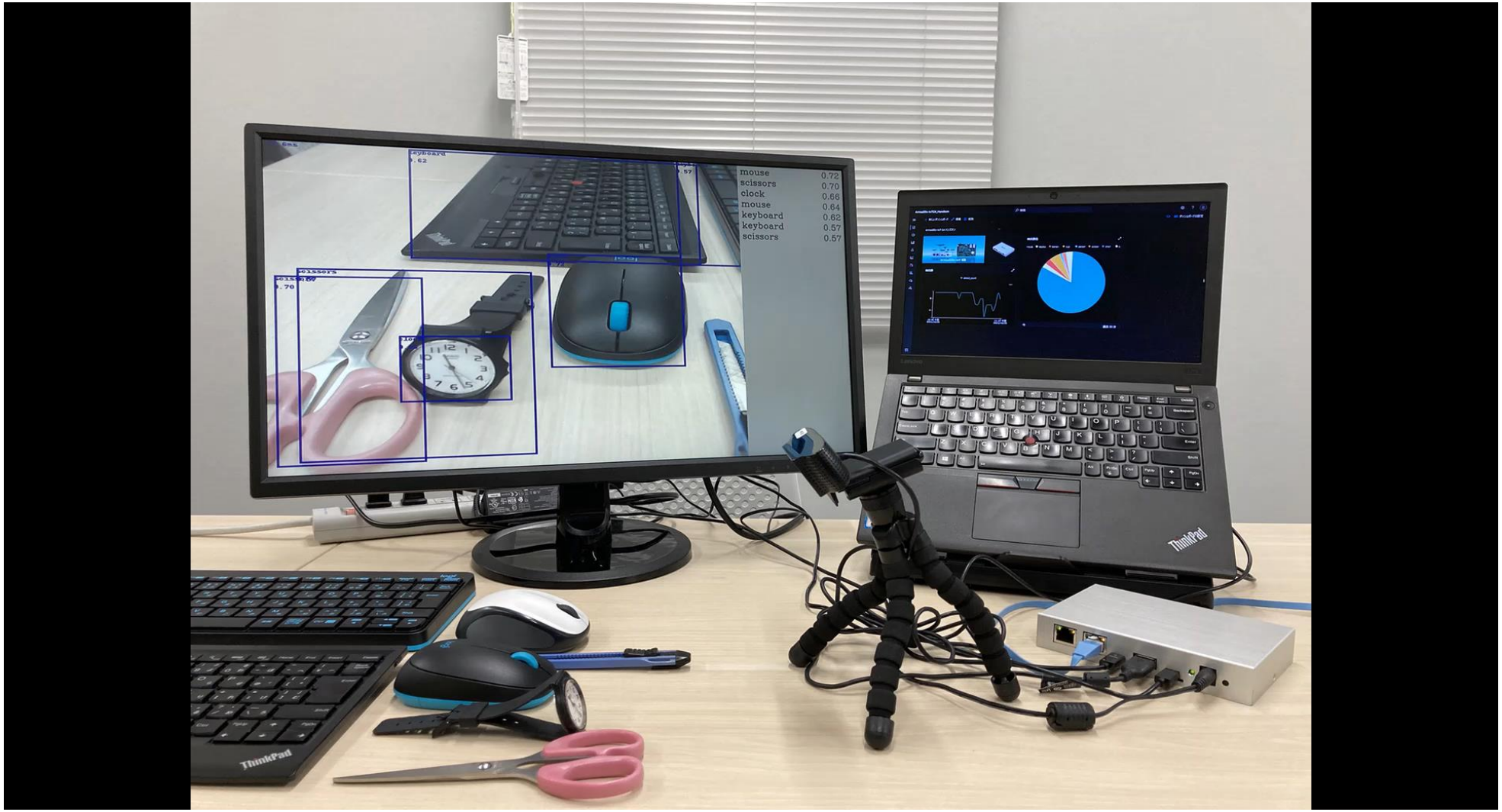
ハンズオン事前説明
・ハンズオン概要説明
・事前準備確認
・ハンズオンの進め方説明

Armadillo-IoT ゲートウェイ G4 セットアップ
・ハードウェア接続、開発環境確認
・podman ネットワーク設定、データ保存先設定

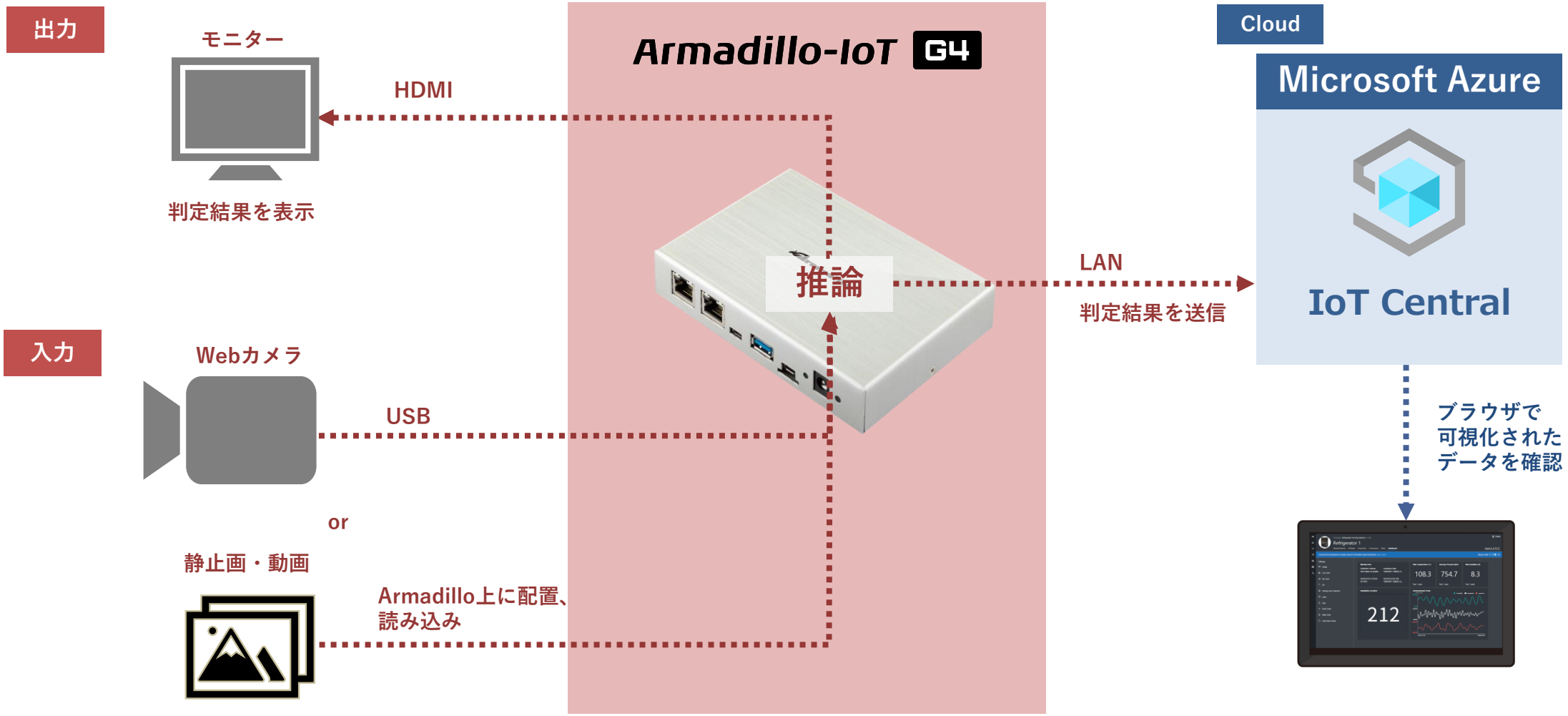
物体認識
・コンテナ作成
・実行

物体認識結果 Azure IoT Central 送信
・コンテナ作成
・実行

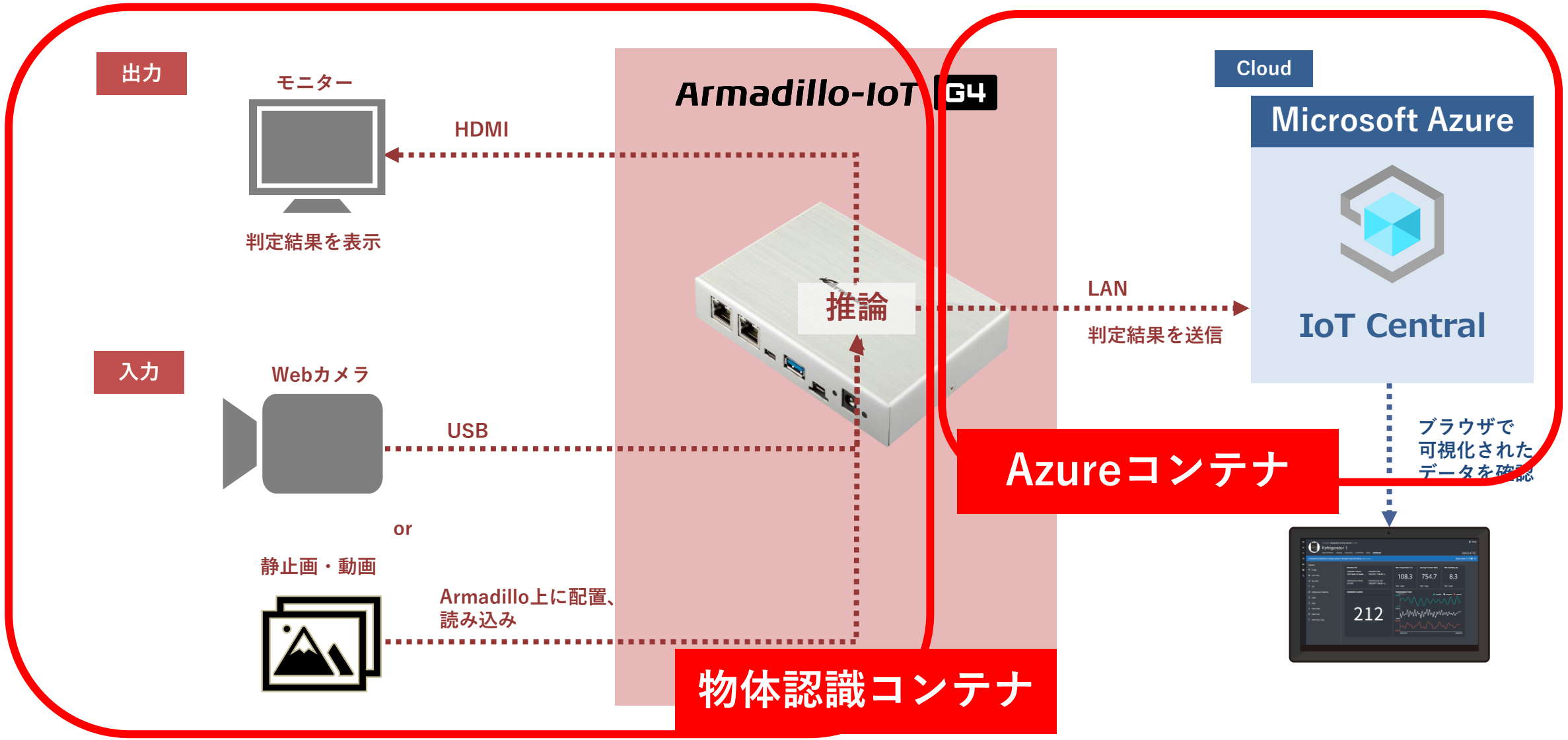
ハンズオンの目標



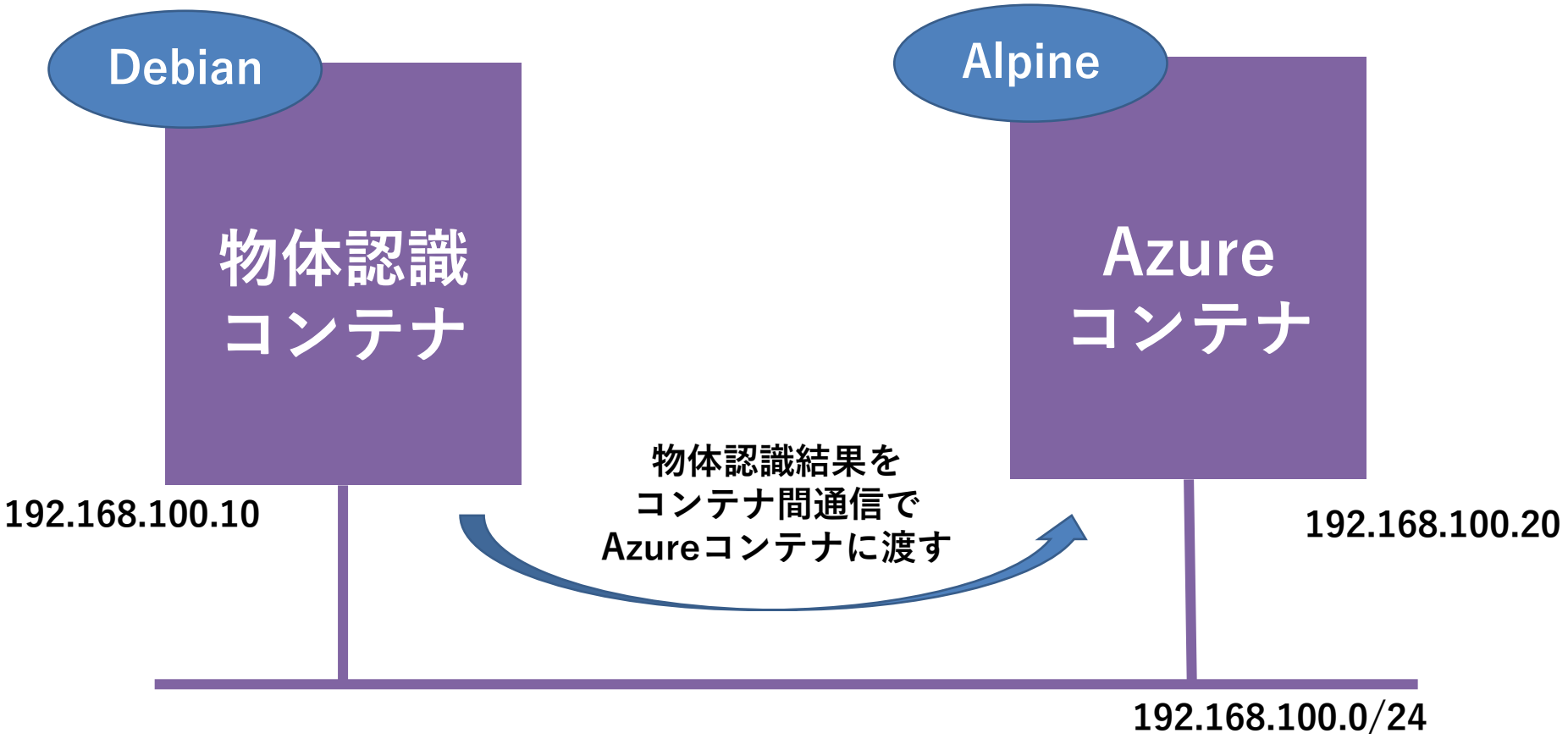
ハンズオン・コンテンツシステム図



コンテナの構成



コンテナの構成



事前準備・確認

■プレゼント品

すべて揃っていますか？

事前準備・確認

■必要な機材

- ◆ USB（タイプA）が接続可能なPC
  （Mac およびWindows11は非対象）
  - ターミナルソフト（Tera Termなど）を
    あらかじめインストールしてください
    → ハンズオンでは例として Tera Termを使用します
  - Windows PCの場合は、下記の手順に従って
    USBコンソール用のドライバをインストールしてください

    https://armadillo.atmark-techno.com/howto/g4-windows-serial
- ◆ HDMI接続可能な fullHD 以上のモニタ
- ◆ LANケーブル 1本
- ◆ UVC規格に対応した USB Webカメラ
  → なくてもOKですが、あった方がより
  リアルタイムの物体認識を体感できます！

事前準備・確認

- 必要な環境
  - DHCP接続可能な有線LAN環境(100BaseT以上推奨)
  - Microsoft アカウント
    - Azure サブスクリプション

事前準備・確認

- 下記にあるファイルを全てダウンロードしてください

URLをコピーしてアドレスバーに
貼り付ける場合は、念のためURLが合っているか
ご確認ください

■ダウンロードURL　★ 2022年1月4日以降公開します★

https://download.atmark-techno.com/sample/algyan/220108_g4_meetup/file/

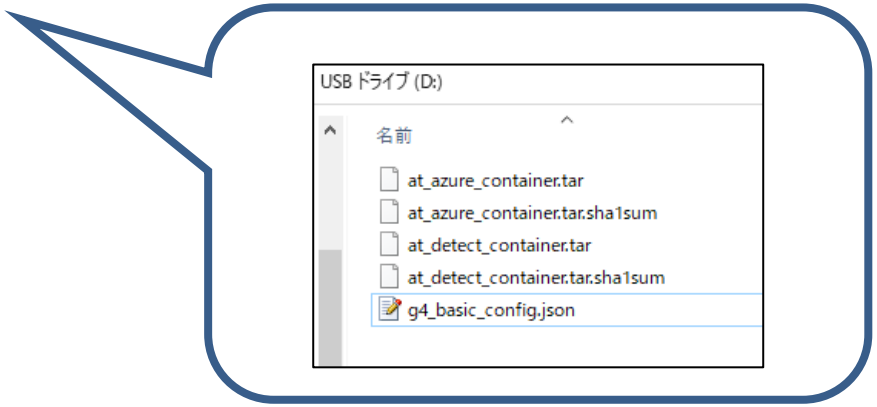
- 以下のファイルをUSBメモリのルートディレクトリに置いてください

ファイル構成

/

at_detect_container.tar	物体認識コンテナ ビルド済イメージ
at_azure_container.tar	Azureコンテナ ビルド済イメージ
at_detect_container.tar.sha1sum	物体認識コンテナ チェックサム
at_azure_container.tar.sha1sum	Azureコンテナ チェックサム
g4_basic_config.json	IoT Central接続設定ファイル

⚠ at_detect_container.tar は
ファイルサイズが 1GBほどあります。



- 以下のファイルはPC上に置いてください

220108_commandlist.txt	実行コマンドリスト
iotc_g4_meetup_handson.json	IoT Centralデバイステンプレート

事前準備・確認

- 下記の手順に従ってArmadillo-IoT ゲートウェイ G4 の初期化を行ってください

microSDカードが必要です

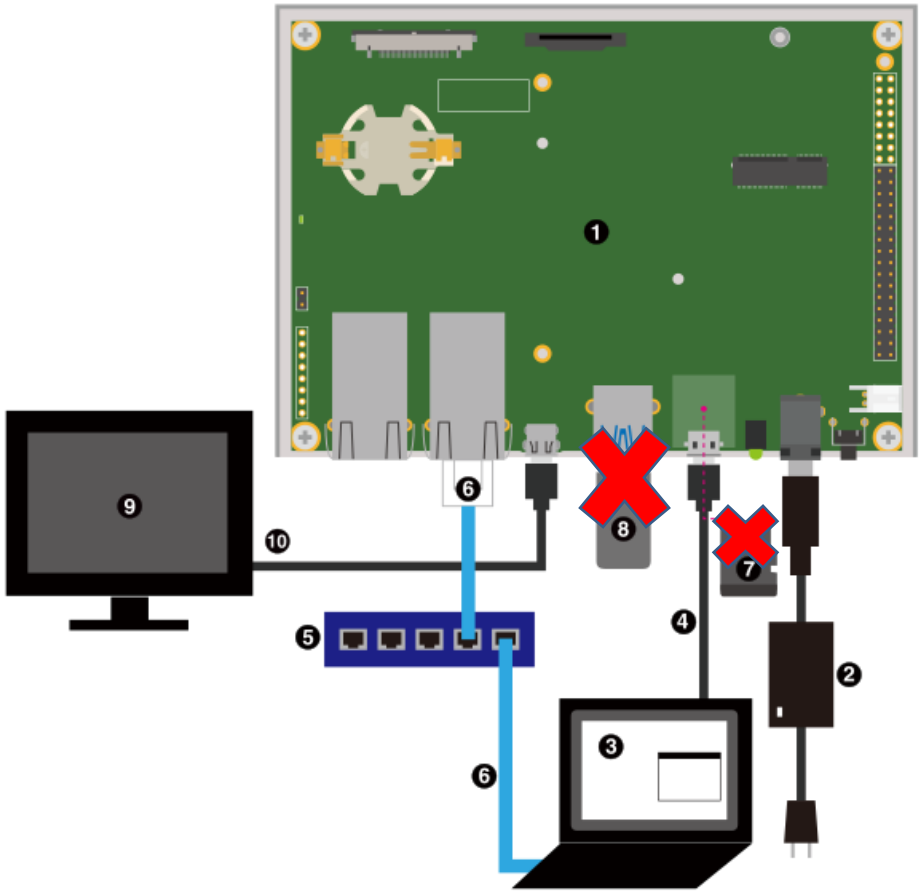
■Armadillo-IoT ゲートウェイ G4 製品マニュアル

9.6.2. インストールディスクを使用した初期化

https://armadillo.atmark-techno.com/resources/documents/armadillo-iot-g4/manuals

事前準備・確認

■下記の通り周辺機器を接続



① Armadillo-IoT ゲートウェイ G4
② ACアダプタ(5V/3A)
③ 作業用PC
④ シリアル通信用USBケーブル(A-microB)
⑤ LAN HUB
⑥ Ethernetケーブル
⑦ ※今回のハンズオン手順では不要です
⑧ ※まだ何も挿さないでください
⑨ ディスプレイ(HDMI対応)
⑩ HDMIケーブル

ACアダプタはハンズオン中に指示があるまで挿さないでください。

ハンズオンの進め方

- ハンズオン手順中、コンソールに入力するコマンドは
  220108_commandlist.txt にまとめているので
  是非コピペにご活用ください。

ハンズオンの進め方

- ハンズオンでは エディタ、ブラウザ、ターミナルソフト、ハンズオン資料など複数のウインドウを切り替えながら作業を行う事となります。
  スムーズに作業を行うためには、
  - モニター上にそれぞれのウインドウを並べながら作業する
  - PCのサブモニターを用意する
  - 資料を事前に印刷する

  などがおすすめです。

Armadillo-IoT ゲートウェイ G4 セットアップ

このセクションで実施すること

Armadillo起動確認

・ログイン
・ログインパスワード保存

各設定、確認

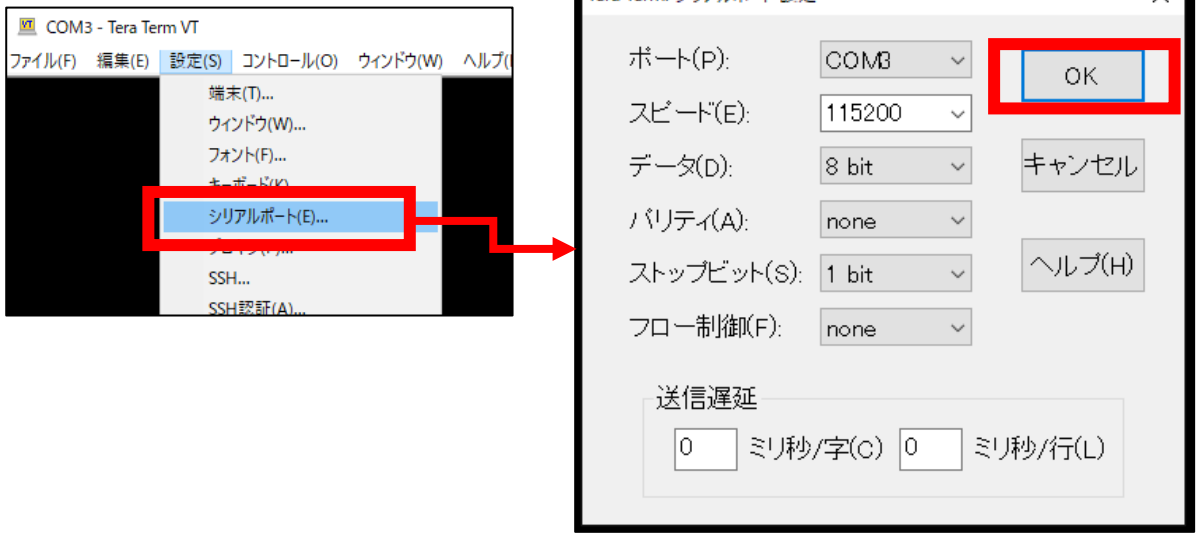
・ネットワーク接続確認
・コンテナ設定
  ・データ保存先
  ・ユーザー定義ネットワーク

起動確認

①ターミナルソフト通信設定

項目	設定
転送レート	115,200 bps
データ長	8bit
ストップビット	1bit
パリティ	なし
フロー制御	なし

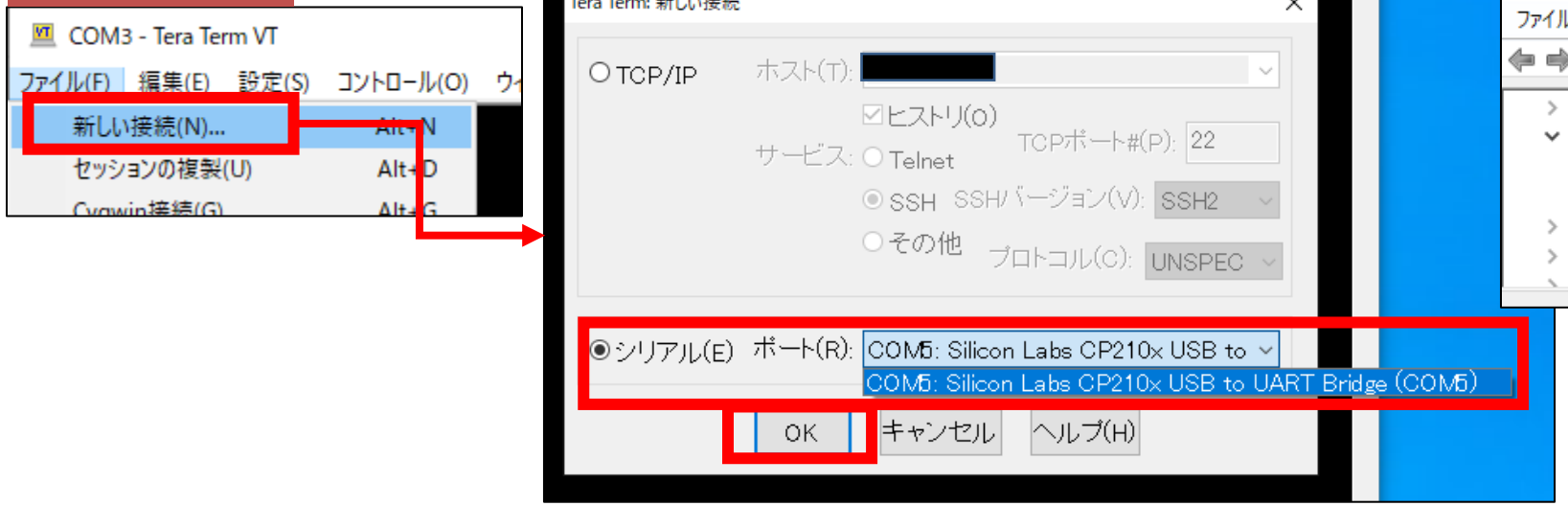
TeraTermの場合



起動確認

②ターミナルソフト接続

TeraTermの場合



“Silicon Labs CP210x USB to UART Bridge”を選択

起動確認

③Armadilloに電源を投入

電源投入前、ArmadilloのUSBポートには何も挿していないことを確認してください。

```
Welcome to Alpine Linux 3.14
Kernel 5.10.35-3-at on an aarch64 (/dev/ttymxc1)


armadillo login:
```

このログが表示されたら
起動OK

Armadilloを安全に終了させる場合は、次のコマンドを実行し「reboot: Power down」と表示されたことを確認してから電源を切断してください。

```
armadillo:~# poweroff
:
:
[ 62.862470] reboot: Power down
```

ログイン

root ログイン

初回ログインではパスワードの変更が必要です

```
armadillo login: root
You are required to change your password immediately (administrator enforced).
New password: 新しいパスワードを入力
Retype new password: 新しいパスワードを再入力
Welcome to Alpine!
```

設定するパスワードには以下を含めることができます
- 大文字のアルファベット
- 小文字のアルファベット
- 0から9までの数字
- 記号、句読点など

パスワード保存

- 以下のコマンドでパスワードを保存

```
armadillo:~# persist_file /etc/shadow
```

Armadillo Base OS ではルートファイルシステムに overlayfs を採用しているため
システムをOFFすると内容が消えてしまいます。
persist_file コマンドを使用すると、ファイル単位で永続化をすることができます。

overlayfs とは:

分離したReadonlyなlower層と、Writableなupper層をマージして1つのファイルシステムに見せる仕組み。Armadilloでは tmpfs(メモリ上に作成するファイルシステム)を upper層にしています。

ネットワーク接続確認

- 以下のコマンドで確認

```
armadillo:~# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
(省略)
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether 00:11:0c:00:0a:bb brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.68/16 brd 172.16.255.255 scope global dynamic noprefixroute eth0
       valid_lft 69192sec preferred_lft 69192sec
    inet6 fe80::8480:cb92:98a7:ec8b/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: eth1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN qlen 1000
    link/ether 00:11:0c:00:0a:bc brd ff:ff:ff:ff:ff:ff
```

“eth0” の方に
IPアドレスが振られている
ことを確認

Podman データ保存先変更

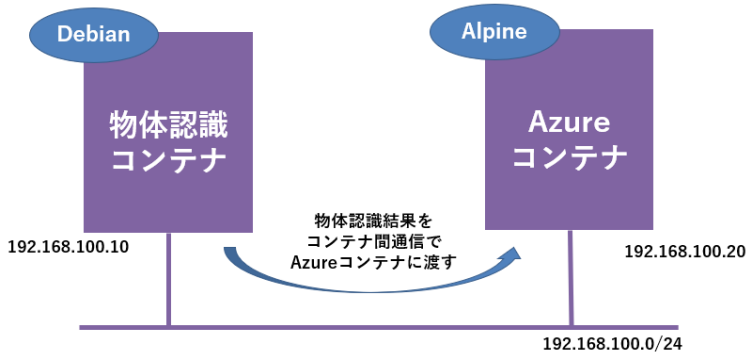
- 以下のコマンドでPodmanのデータ保存先を eMMCに変更

```
armadillo:~# podman_switch_storage --disk
Creating configuration for persistent container storage
Create subvolume '/mnt/containers_storage'
```

デフォルトでは Podman のデータは tmpfs(メモリ上に作成するファイルシステム)
に保存されるため、電源をOFFするとデータが消えてしまいます。
そのため、本ハンズオンでは保存先を eMMC に変更していますが、実際の運用では
eMMCの書き込みを最小限にする観点から、運用時は tmpfsに保存するのが適切です。
オプションに –status をつけると、現在の設定内容を確認することができます。

```
armadillo:~# podman_switch_storage --status
Currently in disk mode, run with --tmpfs to switch
```

コンテナのネットワーク設定



ご利用のネットワーク環境と
IPアドレスが被っている場合、
お知らせください

※こちらのコマンドで確認できます

```
armadillo:~# nmcli
```

- **各コンテナに固定のIPアドレスを持たせるため、**
  下記のコマンドでユーザー定義のネットワークを作成

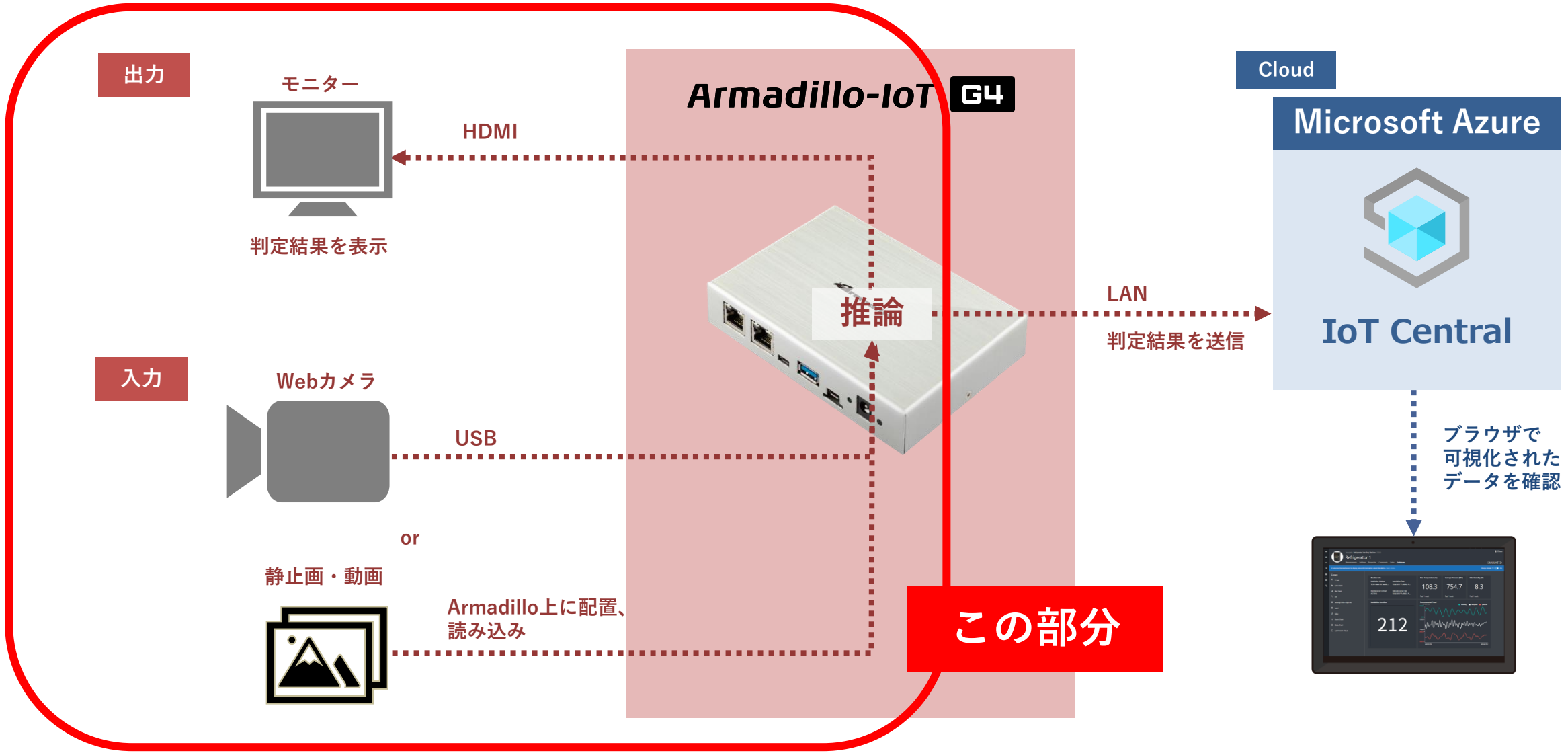
```
armadillo:~# podman network create --subnet=192.168.100.0/24 my_network
```

- **ネットワーク設定ファイルが**
  再起動で消えてしまわないよう永続化する

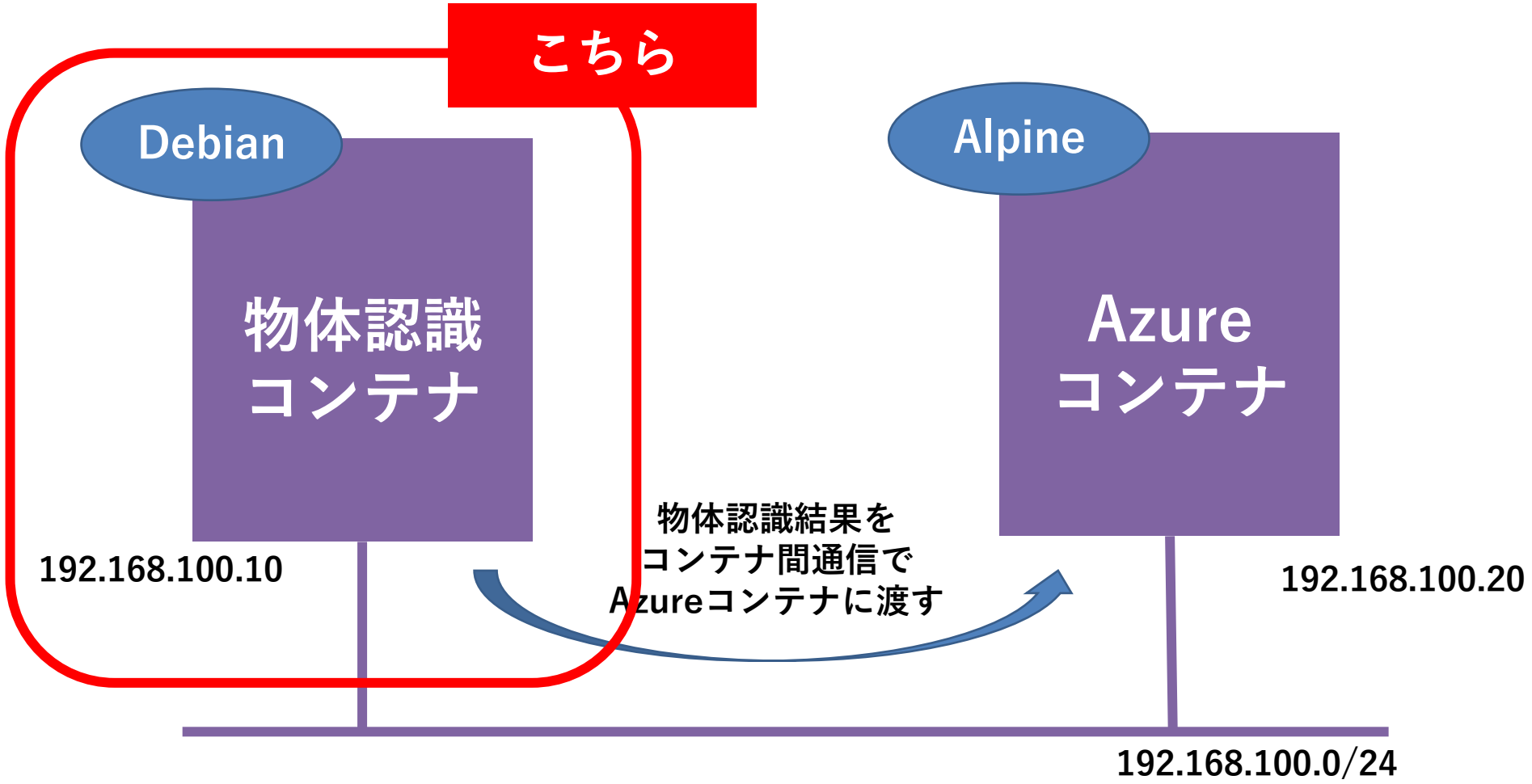
```
armadillo:~# persist_file /etc/cni/net.d/my_network.conflist
```

物体認識

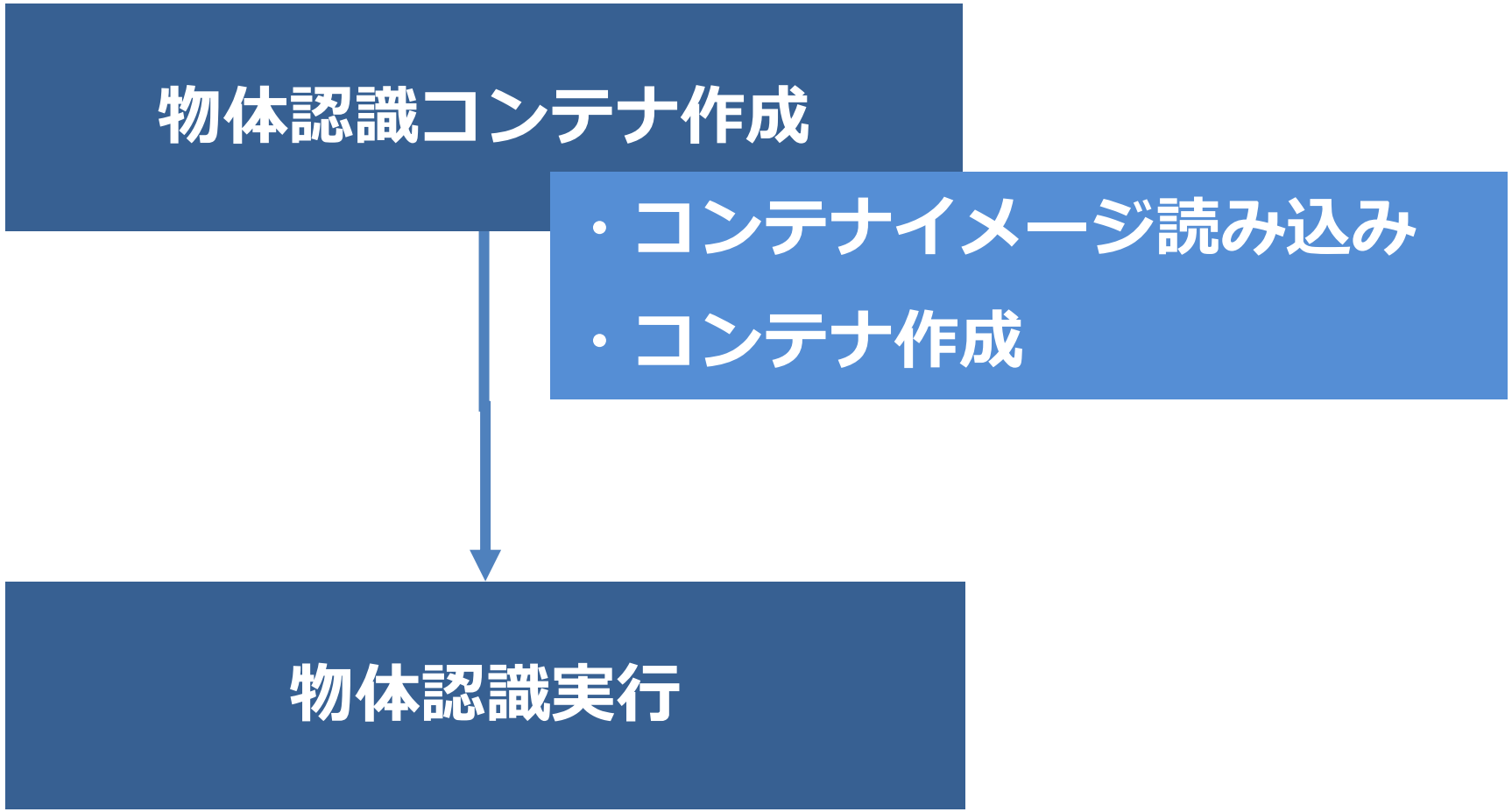
物体認識コンテナ



物体認識コンテナ



このセクションで実施すること

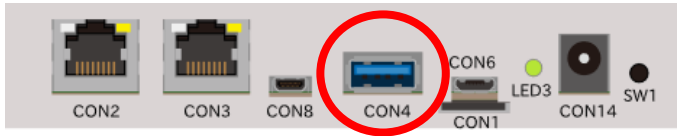


コンテナ作成(Webカメラ有無不問)

CON4

USBメモリ

①USBメモリをArmadilloのUSBコネクタに挿す



USBメモリをArmadilloに挿すと
このようなログが表示されます。

```
armadillo:~# [16577.714566] usb 2-1: new SuperSpeed Gen 1 USB device number 4 using xhci-hcd
[16577.749031] usb-storage 2-1:1.0: USB Mass Storage device detected
[16577.756937] scsi host0: usb-storage 2-1:1.0
[16578.793638] scsi 0:0:0:0: Direct-Access     BUFFALO  USB Flash Disk   1.00 PQ: 0 ANSI: 6
[16578.806287] sd 0:0:0:0: [sda] 60555264 512-byte logical blocks: (31.0 GB/28.9 GiB)
[16578.815266] sd 0:0:0:0: [sda] Write Protect is off
[16578.820606] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[16578.952278]  sda: sda1
[16578.958885] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

②USBメモリをマウントする

```
armadillo:~# mount /dev/sda1 /mnt
```

③マウント先に移動

```
armadillo:~# cd /mnt
```

④ビルド済みのイメージがあるか確認

```
armadillo:/mnt# ls at_detect_container.tar
at_detect_container.tar
```

コンテナ作成（Webカメラ有無不問）

⑤チェックサムを確認

```
armadillo:/mnt# sha1sum -c at_detect_container.tar.sha1sum
at_detect_container.tar: OK
```

⑥イメージを読み込む

2分弱かかります

```
armadillo:/mnt# podman load -i at_detect_container.tar
```

⑦イメージが読み込めたか確認

```
armadillo:/mnt# podman images
REPOSITORY                    TAG         IMAGE ID      CREATED       SIZE
localhost/at_detect_container  latest      87a2fcf04d24  13 days ago   1.2 GB
```

⑧ルートに戻る

```
armadillo:/mnt# cd
```

⑨USBメモリをアンマウントし、Armadilloから取り外す

```
armadillo:~# umount /mnt
```

Webカメラなしの場合→ p.37 にジャンプ

コンテナ作成（Webカメラあり）

Webカメラ

⑩WebカメラをUSBコネクタに挿す

⑪Webカメラ認識状態を確認する

```
armadillo:~# ls -la /dev/v4l/by-id
total 0
drwxr-xr-x 2 root root 80 Jan  1 09:04 .
drwxr-xr-x 4 root root 80 Jan  1 09:04 ..
lrwxrwxrwx 1 root root 12 Jan  1 09:04 usb-046d_HD_Pro_Webcam_C920_3E1FF2AF-video-index0 -> ../../video3
lrwxrwxrwx 1 root root 12 Jan  1 09:04 usb-046d_HD_Pro_Webcam_C920_3E1FF2AF-video-index1 -> ../../video4
```

xxx-index0 のリンク先パスを確認
※この場合は /dev/video3

コンテナ作成（Webカメラあり）

CON4

Webカメラ

⑫コンテナを作成する

```
armadillo:~# podman run -itd --name=detect ¥
        --env=XDG_RUNTIME_DIR=/tmp ¥
        --env=LD_LIBRARY_PATH=/opt/firmware/usr/lib/aarch64-linux-gnu ¥
        --env=QT_QPA_PLATFORM=wayland ¥
        --device=/dev/tty1 ¥
        --device=/dev/input  ¥
        --device=/dev/dri ¥
        --device=/dev/galcore ¥
        --volume /run/udev:/run/udev:ro ¥
        --cap-add "CAP_SYS_TTY_CONFIG" ¥
        --device=/dev/video3:/dev/video3  ¥
        --device=/dev/mxc_hantro ¥
        --device=/dev/mxc_hantro_vc8000e ¥
        --device=/dev/ion ¥
        --volume=/opt/firmware:/opt/firmware:ro ¥
        --network=my_network --ip 192.168.100.10 ¥
        localhost/at_detect_container /bin/bash
```

・Webカメラあり

この行を p.34 ⑪で確認したデバイスのパスに
合わせて適宜修正してください。

/dev/video2 の場合：
--device=/dev/video2:/dev/video3 ¥

/dev/video3 の場合：
--device=/dev/video3:/dev/video3 ¥

コンテナ作成(Webカメラあり)

CON4

Webカメラ

⑬コンテナが作成できたか確認する

```
armadillo:~# podman ps -a
CONTAINER ID  IMAGE                                COMMAND     CREATED      STATUS              PORTS       NAMES
21571d0fc440  localhost/ at_detect_container:latest  /bin/bash   8 weeks ago  Up 2 hours ago                  detect
```

p.39 にジャンプ

作成したコンテナは下記のコマンドで削除することができます。

```
armadillo:~# podman rm detect
```

コンテナ作成（Webカメラなし）

⑩コンテナを作成する

Webカメラなし

```
armadillo:~# podman run -itd --name=detect ¥
        --env=XDG_RUNTIME_DIR=/tmp ¥
        --env=LD_LIBRARY_PATH=/opt/firmware/usr/lib/aarch64-linux-gnu ¥
        --env=QT_QPA_PLATFORM=wayland ¥
        --device=/dev/tty1 ¥
        --device=/dev/input  ¥
        --device=/dev/dri ¥
        --device=/dev/galcore ¥
        --volume /run/udev:/run/udev:ro ¥
        --cap-add "CAP_SYS_TTY_CONFIG" ¥
        --device=/dev/mxc_hantro ¥
        --device=/dev/mxc_hantro_vc8000e ¥
        --device=/dev/ion ¥
        --volume=/opt/firmware:/opt/firmware:ro ¥
        --network=my_network --ip 192.168.100.10 ¥
        localhost/at_detect_container /bin/bash
```

コンテナ作成(Webカメラなし)

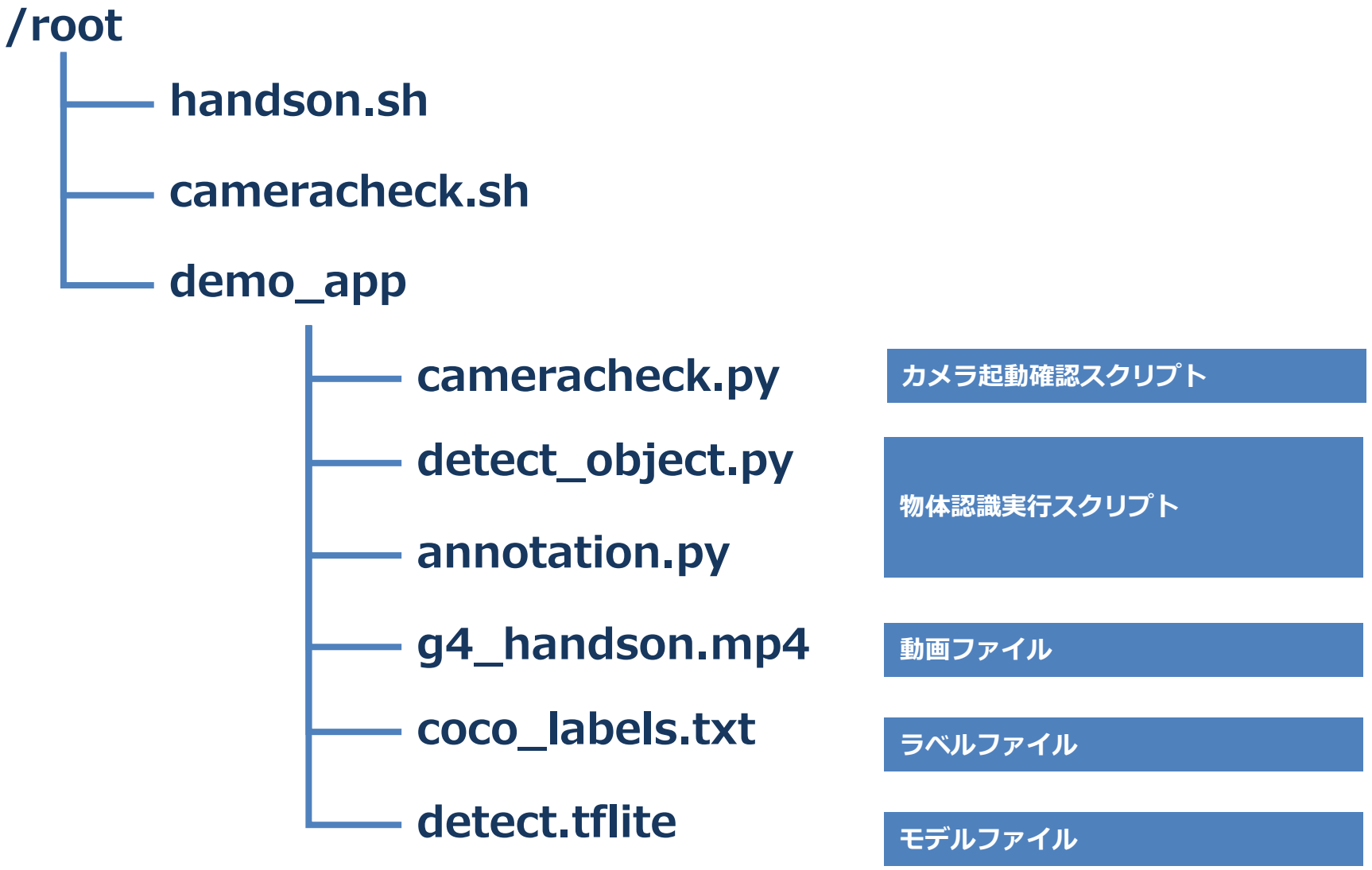
⑪コンテナが作成できたか確認する

```
armadillo:~# podman ps -a
CONTAINER ID  IMAGE                                COMMAND     CREATED      STATUS              PORTS       NAMES
21571d0fc440  localhost/ at_detect_container:latest  /bin/bash   8 weeks ago  Up 2 hours ago                  detect
```

作成したコンテナは下記のコマンドで削除することができます。

```
armadillo:~# podman rm detect
```

コンテナ内コード構成



カメラ/HDMI出力動作確認

Webカメラ

Webカメラありの場合のみ実行

①コンテナの中に入る

```
armadillo:~# podman exec -it detect /bin/bash
root@f6107fd74be7:/#
```

②スクリプト実行

```
root@f6107fd74be7:~# /root/cameracheck.sh
```

③HDMIで接続されたモニタにWebカメラから取得した動画が表示されることを確認

エラーが発生する場合、今回は「Webカメラなし」パターンでの実行をお願いします。
「Webカメラあり」のコンテナでも「カメラなし」パターンは動作するので、コンテナを作り直す必要はありません。

④コンテナから出る

```
root@f6107fd74be7:/# exit
exit
armadillo:~#
```

物体認識 実行

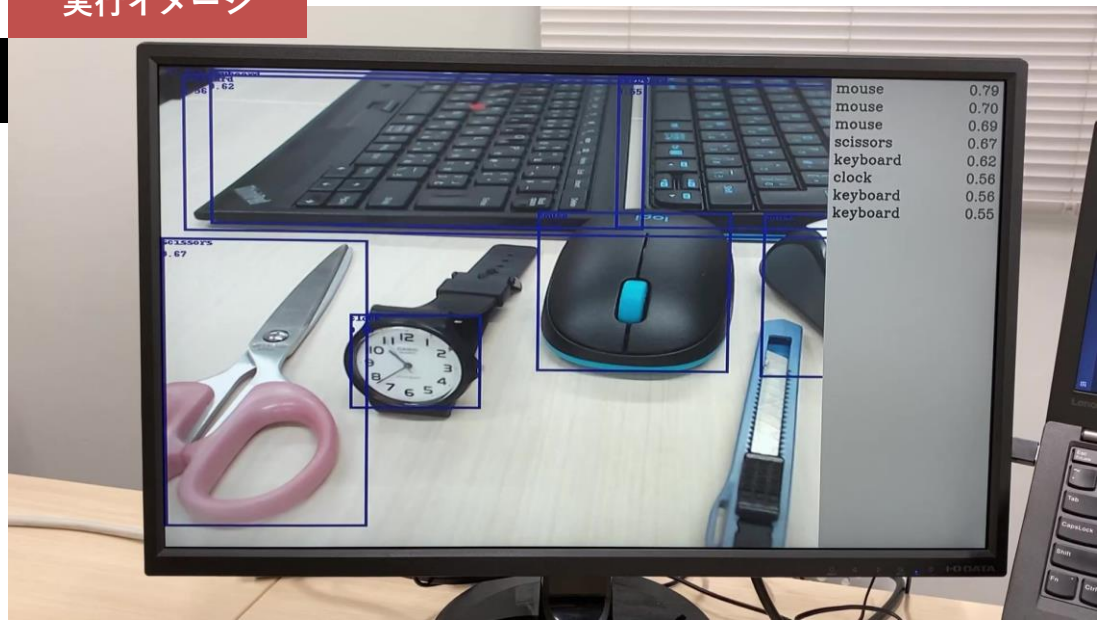
■Webカメラあり

```
armadillo:~# podman exec -d detect /root/handson.sh --camera
```

■Webカメラなし(動画を使用)

```
armadillo:~# podman exec -d detect /root/handson.sh
```

実行イメージ

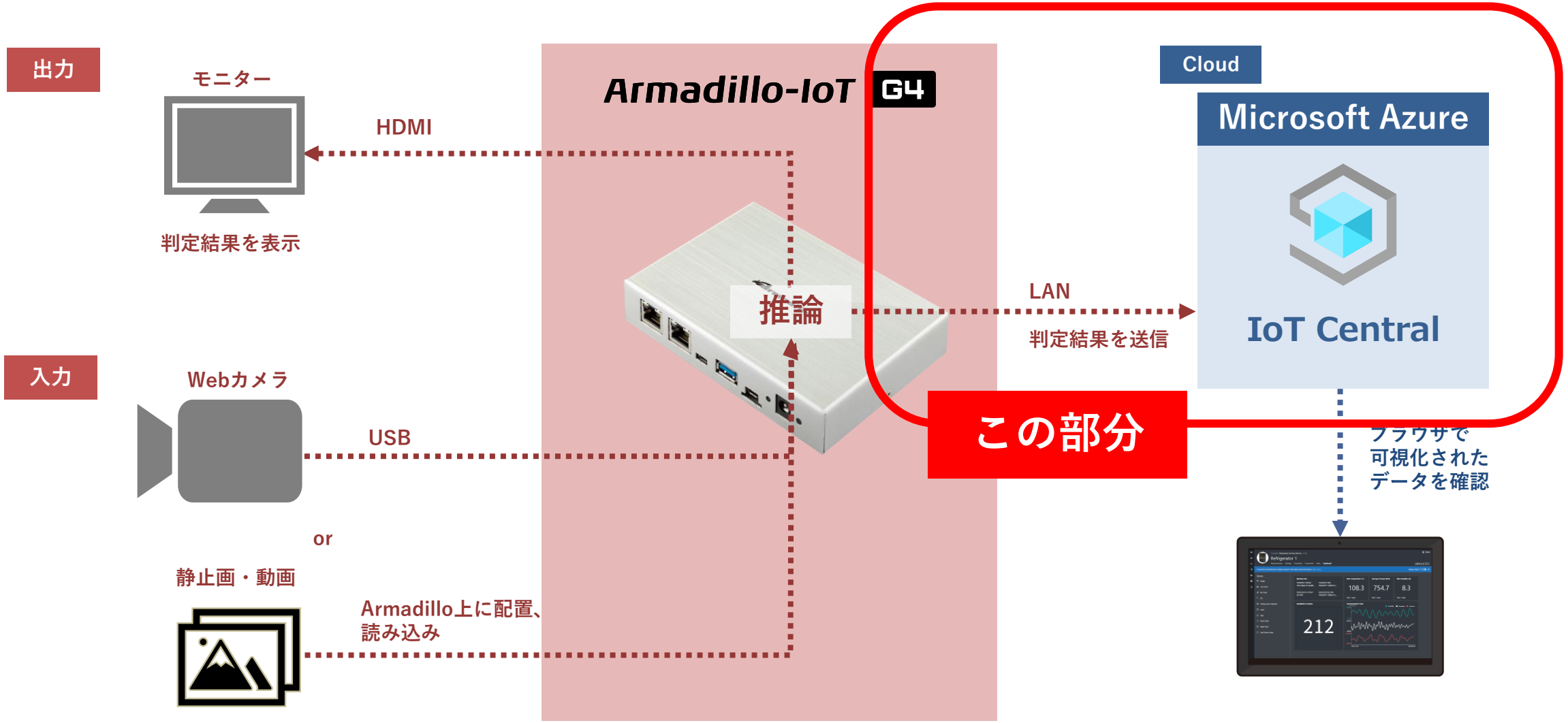


次のセクションに移るため、下記のコマンドで物体認識コンテナを再起動します。

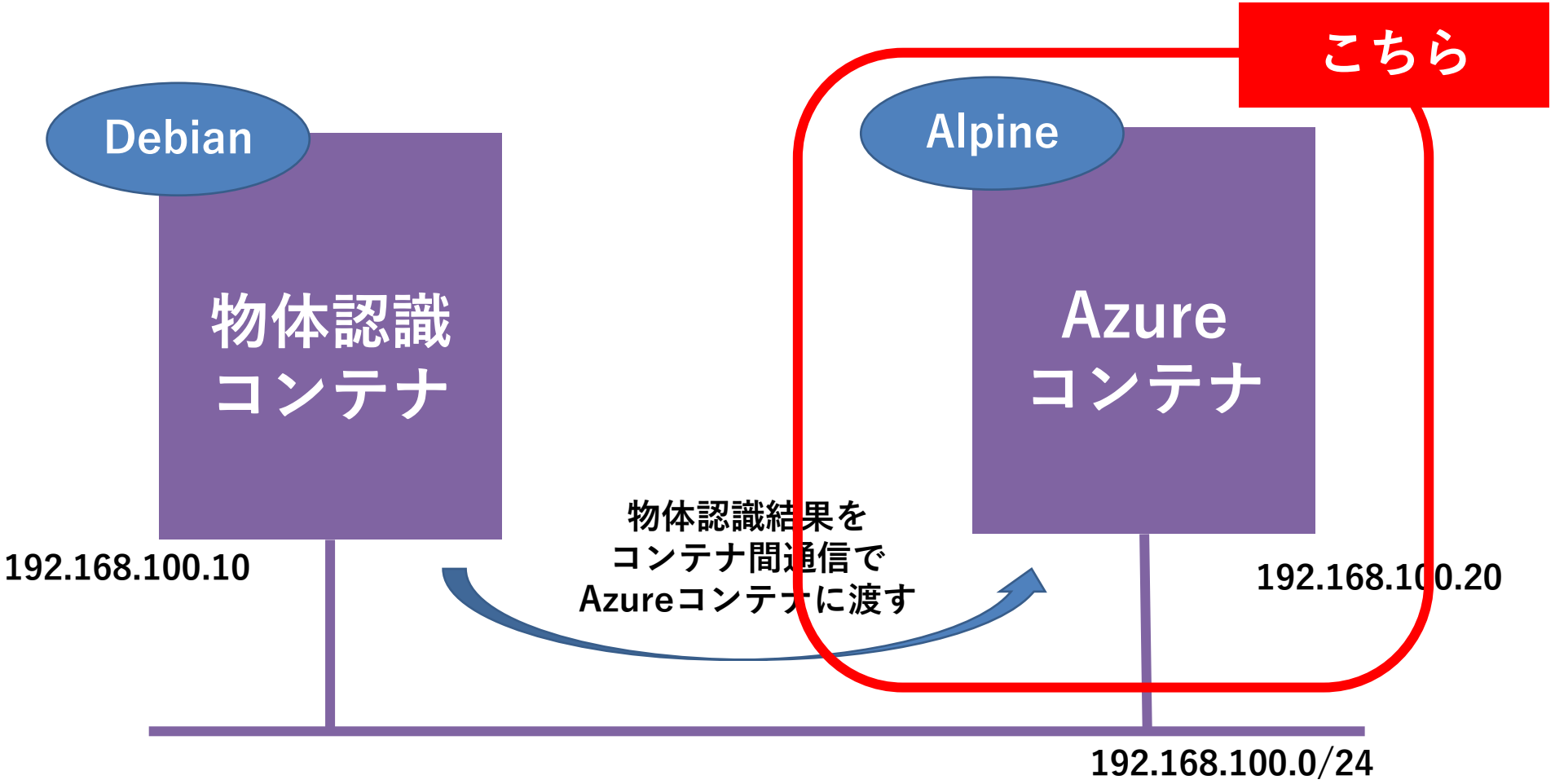
```
armadillo:~# podman restart detect
```

物体認識結果
Azure IoT Central送信

Azureコンテナ



Azureコンテナ



このセクションで実施すること

Azureコンテナ作成

・コンテナイメージ読み込み
・コンテナ作成

IoT Centralアプリケーション
セットアップ

・アプリケーション作成
・デバイステンプレート作成
・ビュー作成
・デバイス登録

接続情報設定

・接続情報取得
・設定ファイル編集/書き込み
・接続確認

物体認識結果
Azure 送信処理実行

・ブラウザでデータ可視化確認

コンテナ作成

USBメモリ

①ArmadilloからWebカメラを取り外し、USBメモリを挿す

②USBメモリをマウントする

```
armadillo:~# mount /dev/sda1 /mnt
```

③マウント先に移動

```
armadillo:~# cd /mnt
```

④ビルド済みのイメージがあるか確認

```
armadillo:/mnt# ls at_azure_container.tar
at_azure_container.tar
```

⑤チェックサムを確認

```
armadillo:/mnt# sha1sum -c at_azure_container.tar.sha1sum
at_azure_container.tar: OK
```

コンテナ作成

USBメモリ

⑥イメージを読み込む

```
armadillo:/mnt# podman load -i at_azure_container.tar
```

⑦イメージが読み込めたか確認

```
armadillo:/mnt# podman images
REPOSITORY                      TAG         IMAGE ID      CREATED         SIZE
localhost/at_azure_container    latest      c6ca6cf31232  46 hours ago    91.6 MB
localhost/at_detect_container   latest      87a2fcf04d24  13 days ago      1.2 GB
```

⑧ルートに戻る

```
armadillo:/mnt# cd
```

⑨USBメモリをアンマウントし、Armadilloから取り外す

```
armadillo:~# umount /mnt
```

コンテナ作成

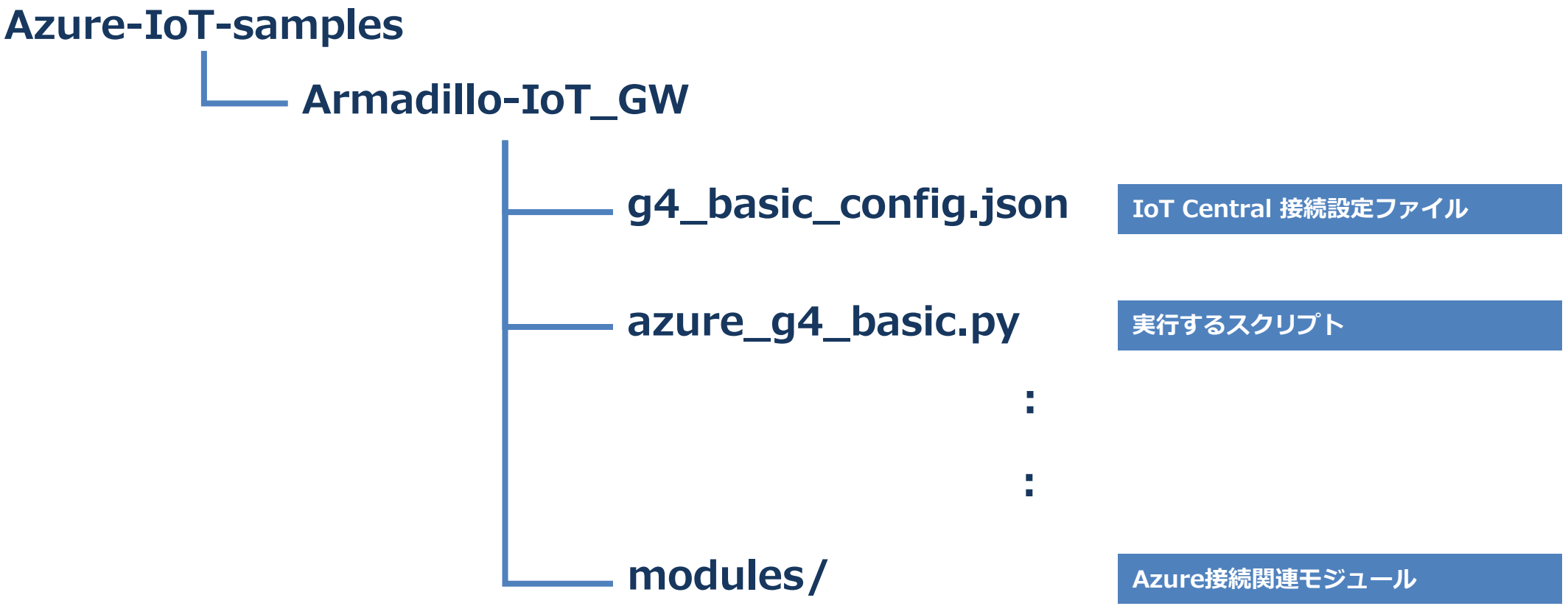
⑩コンテナを作成する

```
armadillo:~# podman run -itd --name=azure --env=NUMBER=ALGYAN0108 ¥
--net=my_network --ip=192.168.100.20 ¥
localhost/at_azure_container /bin/sh
```

⑪コンテナが作成できたか確認する

```
armadillo:~# podman ps -a
CONTAINER ID  IMAGE                                COMMAND     CREATED      STATUS              PORTS       NAMES
d6c973e30dc6  localhost/at_detect_container        /bin/bash   2 days ago   Up About an hour ago             detect
6c9b7777673c  localhost/ at_azure_container:latest  /bin/sh     2 days ago   Up About an hour ago             azure
```

コンテナ内コード構成



IoT Central アプリケーション作成

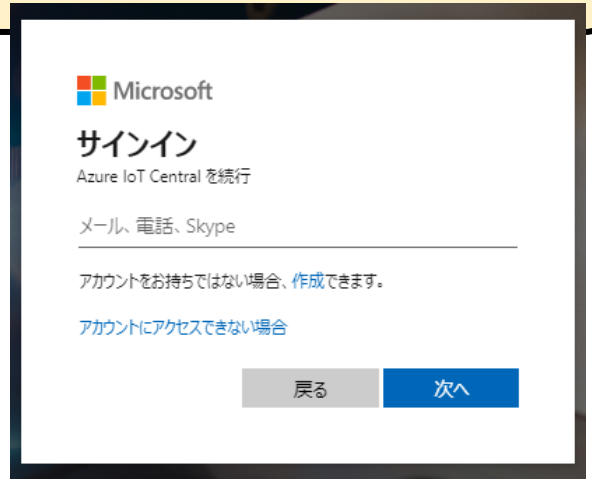
①下記の URL にアクセス

■Azure IoT Central – IoT アプリケーションのビルド

https://apps.azureiotcentral.com/build

②カスタムアプリ「アプリの作成」を選択

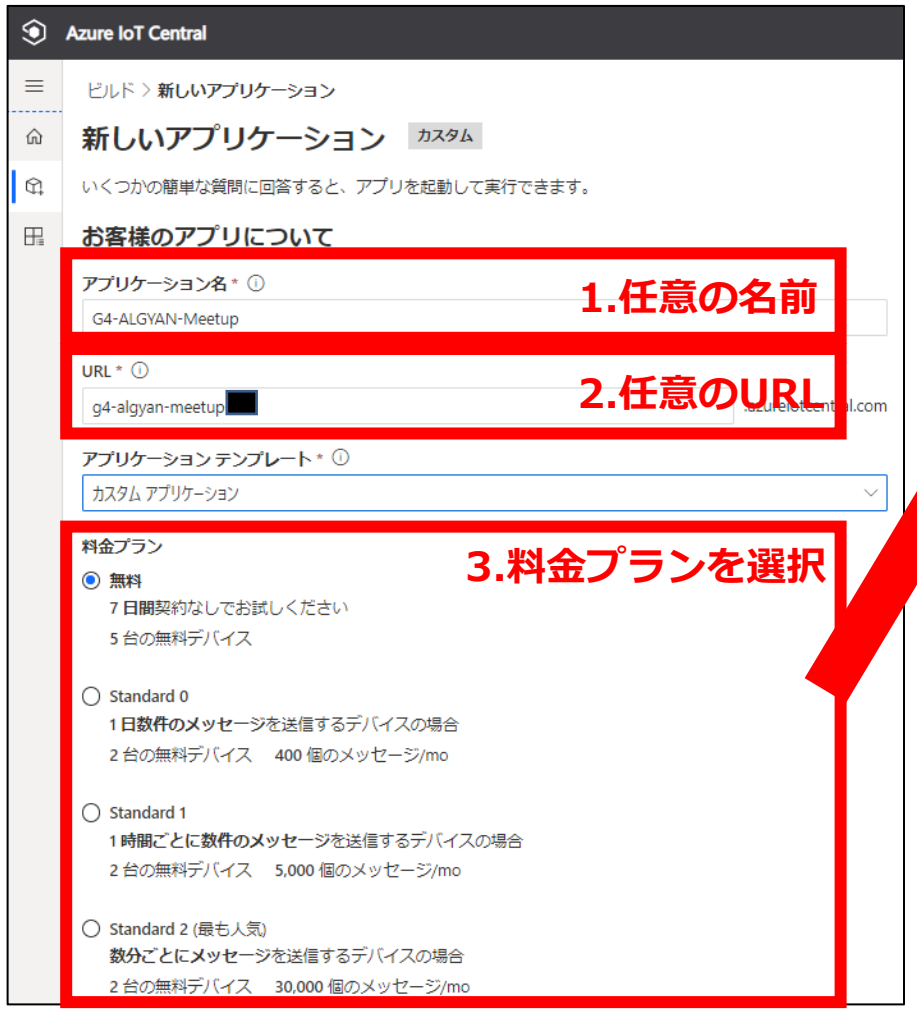
次の画面でサインインを求められたら、
お持ちの Azure アカウントでログインしてください



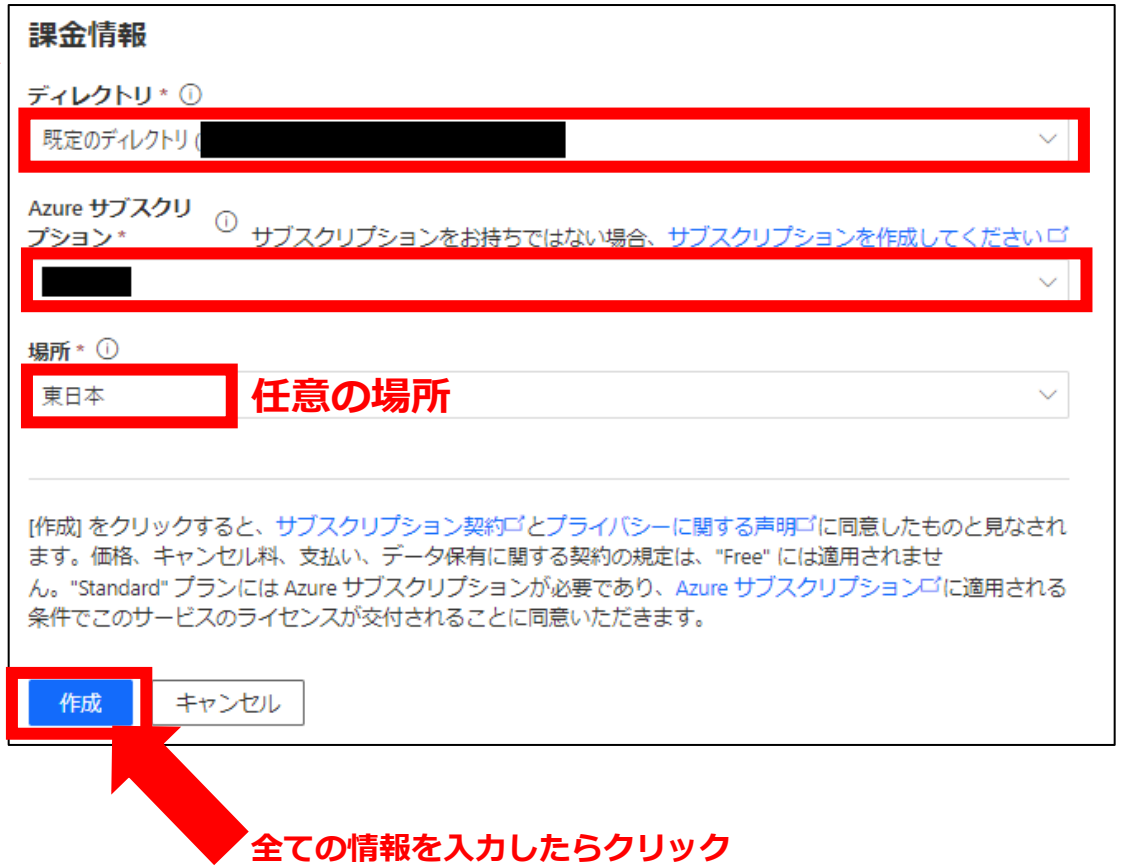
IoT Central アプリケーション作成

Atmark Techno

1アカウントにつき、1つまで7日間無料のアプリケーションを作成することができます



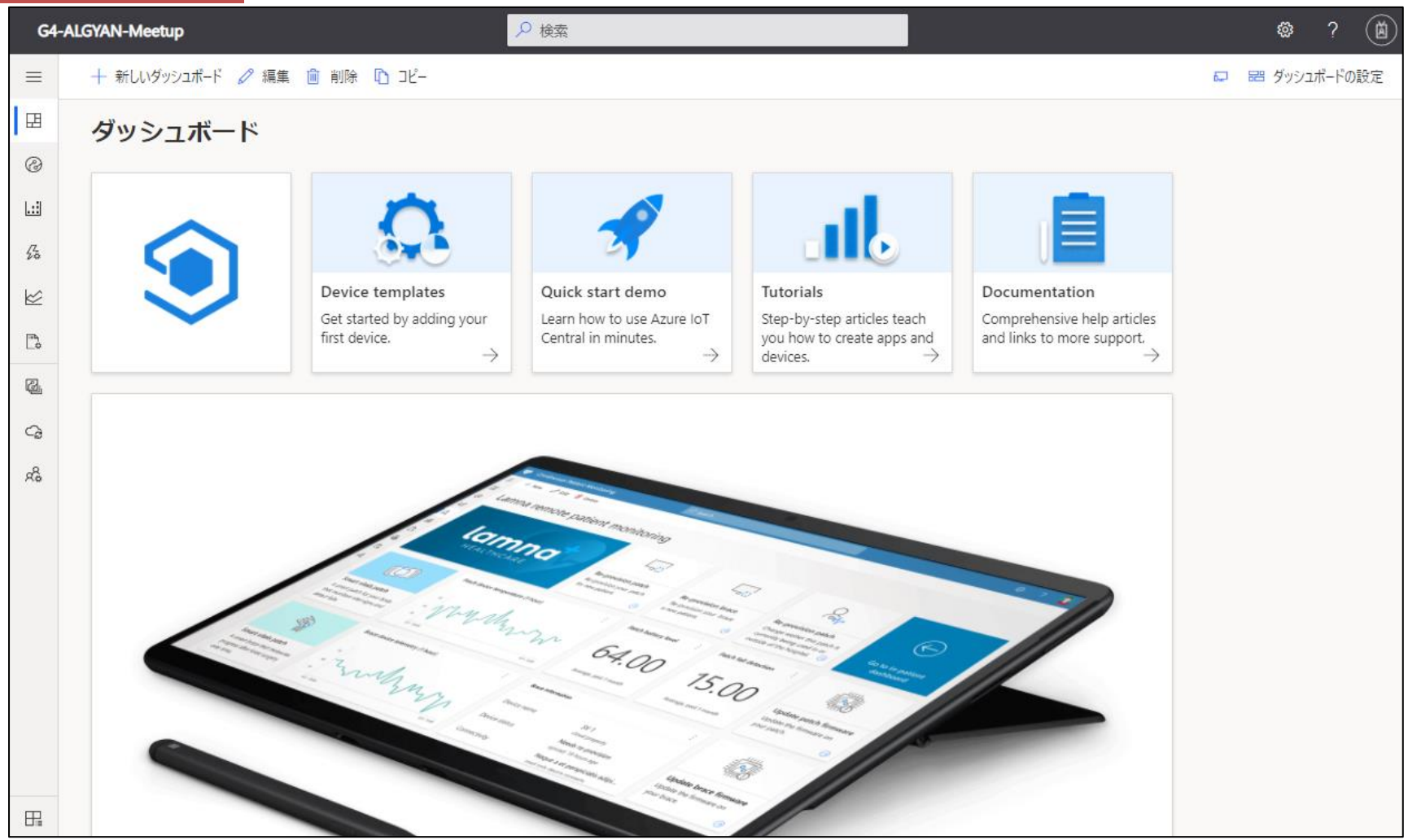
●有料を選択した場合、サブスクリプション情報を入力



IoT Central アプリケーション作成

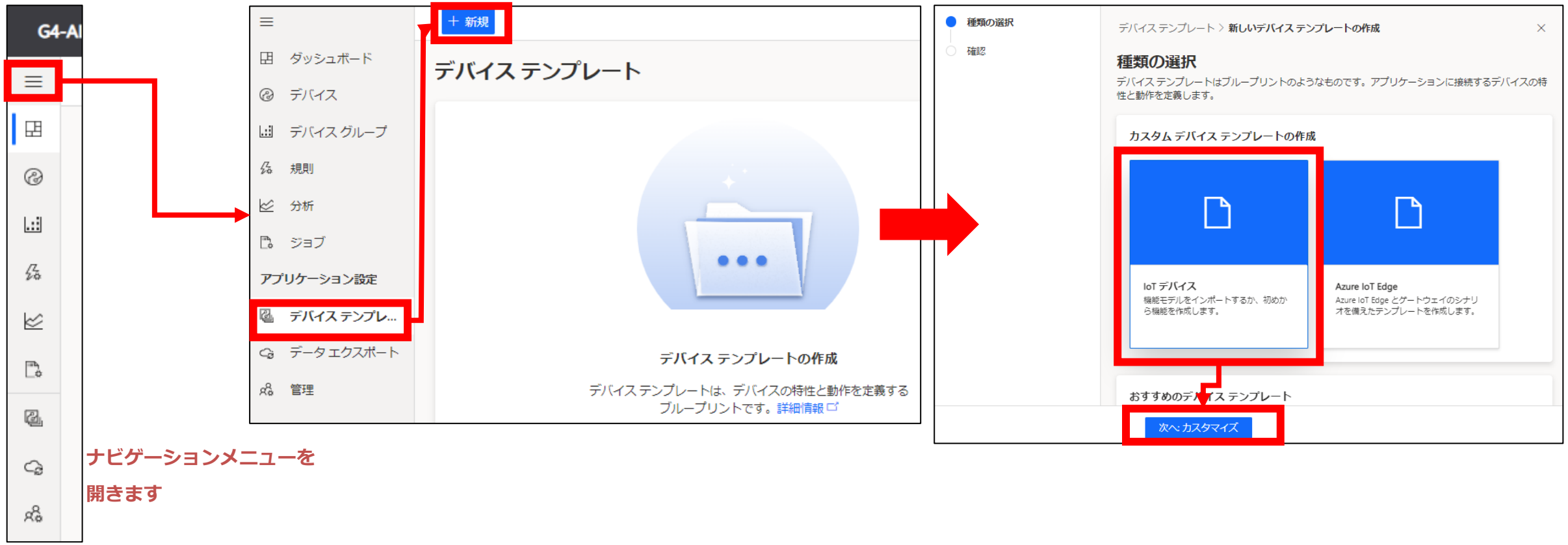
Atmark Techno

アプリケーションTOP画面



デバイステンプレート作成

デバイステンプレート： デバイスが IoT Central とどのようにやり取りするか記述されたモデル



ナビゲーションメニューを開きます

デバイステンプレート作成

デバイステンプレート名は
「G4Meetup」と入力

デバイステンプレートの種類が IoTデバイス、
デバイステンプレート名が「G4Meetup」に
なっていることを確認してください

モデルの作成

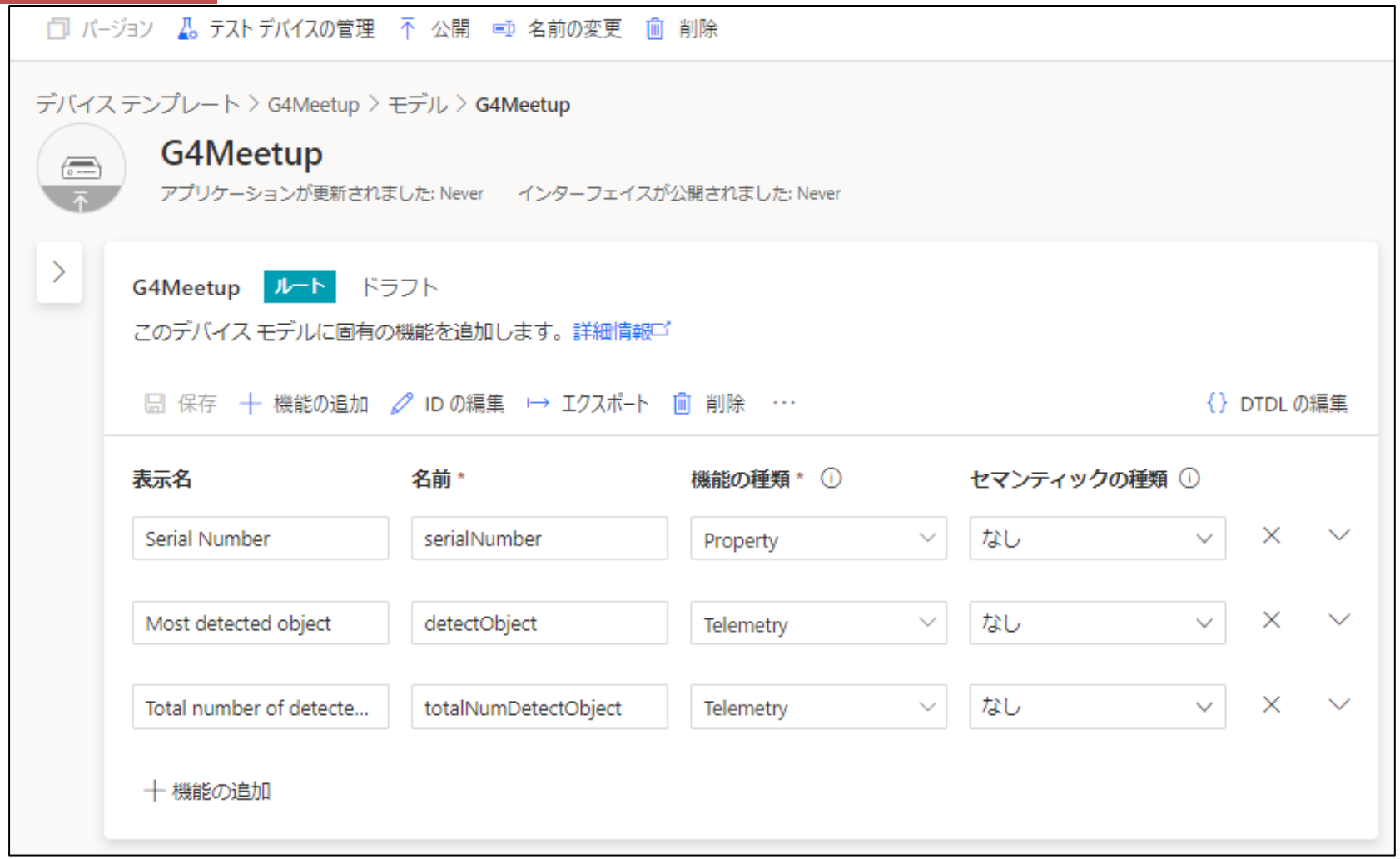
ファイル選択画面が表示されるので
予めダウンロードした
iotc_g4_meetup_handson.json
を選択

【注意】
ダウンロードしたファイルにはjsonファイルが2つあります。指定するファイルを間違えないよう注意してください。

iotc_g4_meetup_handson.json には
デバイスが IoT Central に送信する
テレメトリ等の定義を記述しています。

デバイステンプレート作成

モデルインポート後画面



デバイステンプレート作成

各パラメータに関して

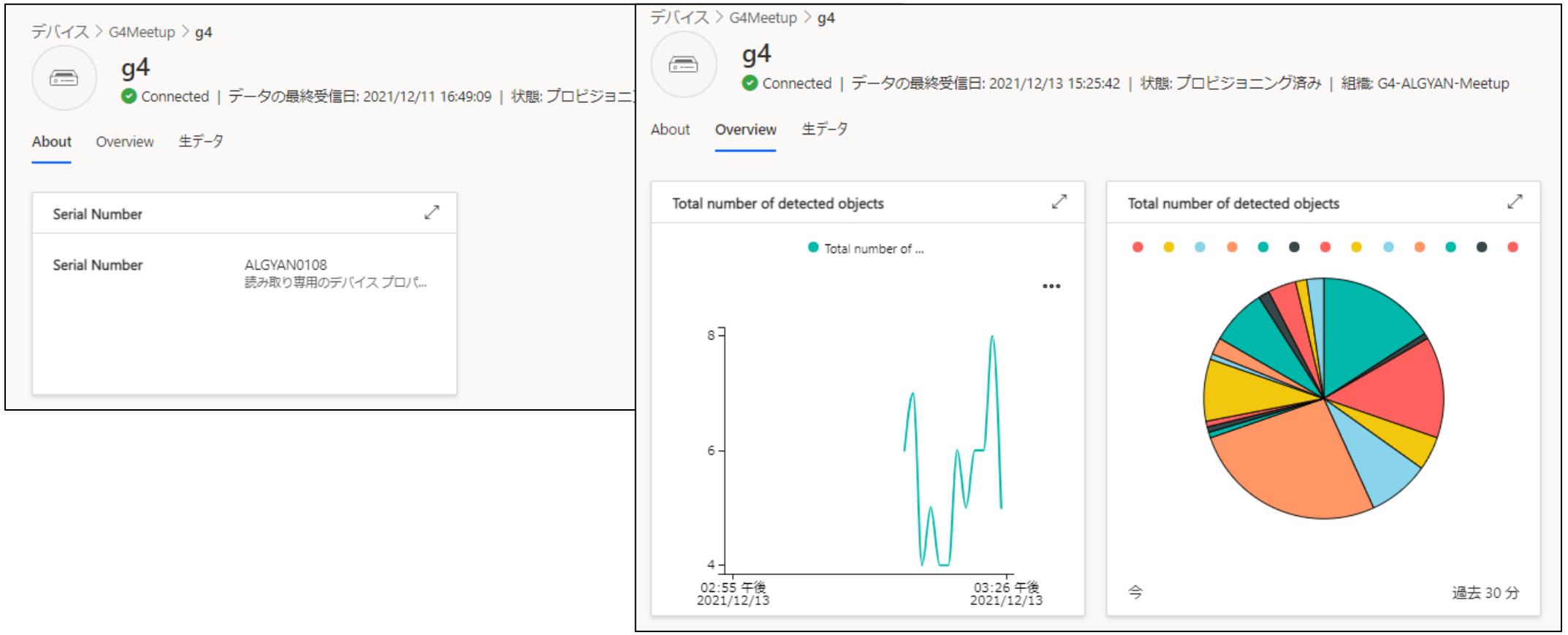
デバイス固有の番号。
今回のハンズオンではコンテナ作成時に
指定しています。
IoT Centralに接続した後に1回だけ送信します。

物体検出結果。Object型で定義しています。
5秒ごとに一番多く検出した物体名・その数を
送信します。

物体検出総数。5秒ごとに検出した物体の総数
を送信します。

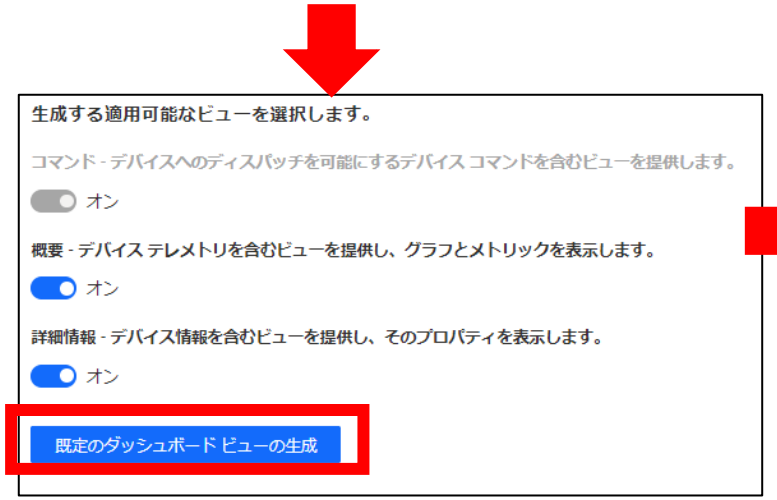
ビュー作成

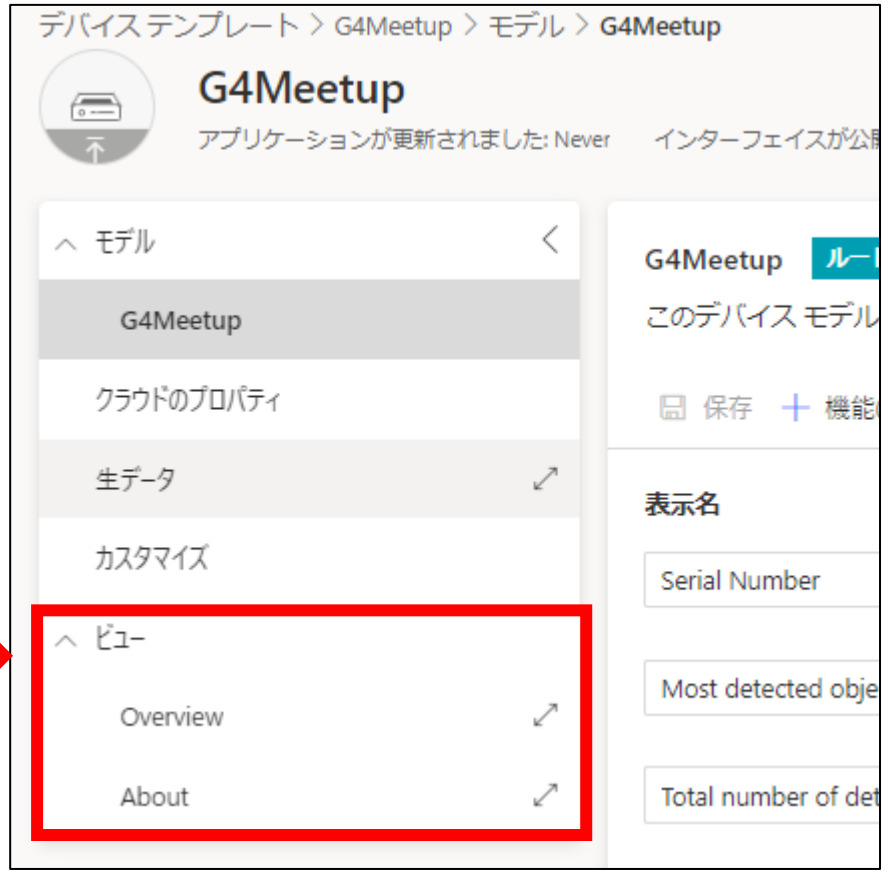
ビュー完成図



ビュー作成

規定のビューの作成

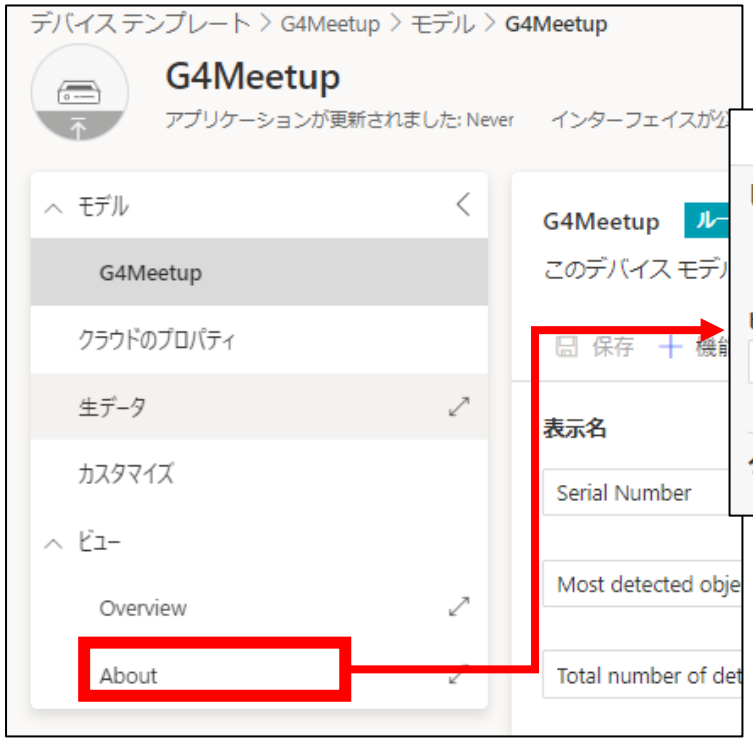




Overview, About ビューが自動的に生成されます。

ビュー作成

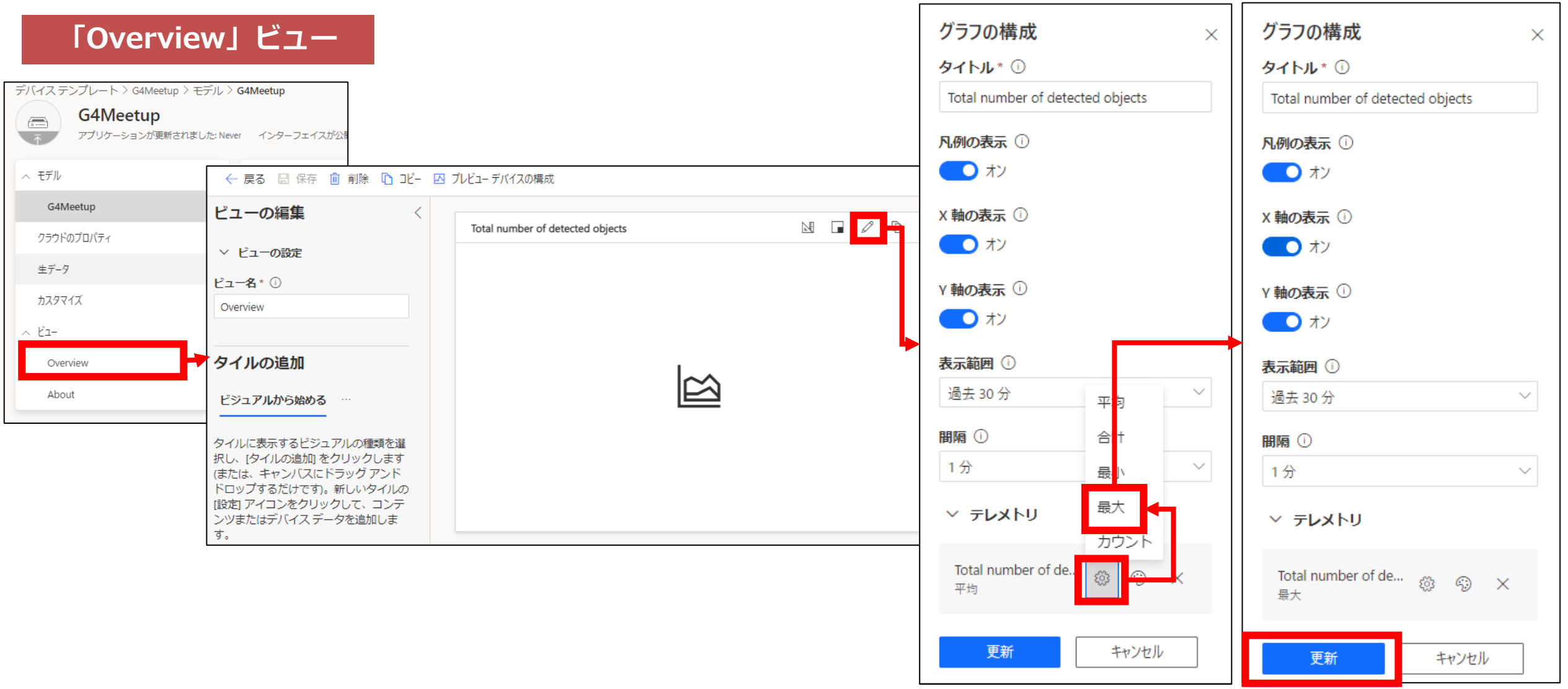
「About」ビュー



内容の確認だけでOKです。

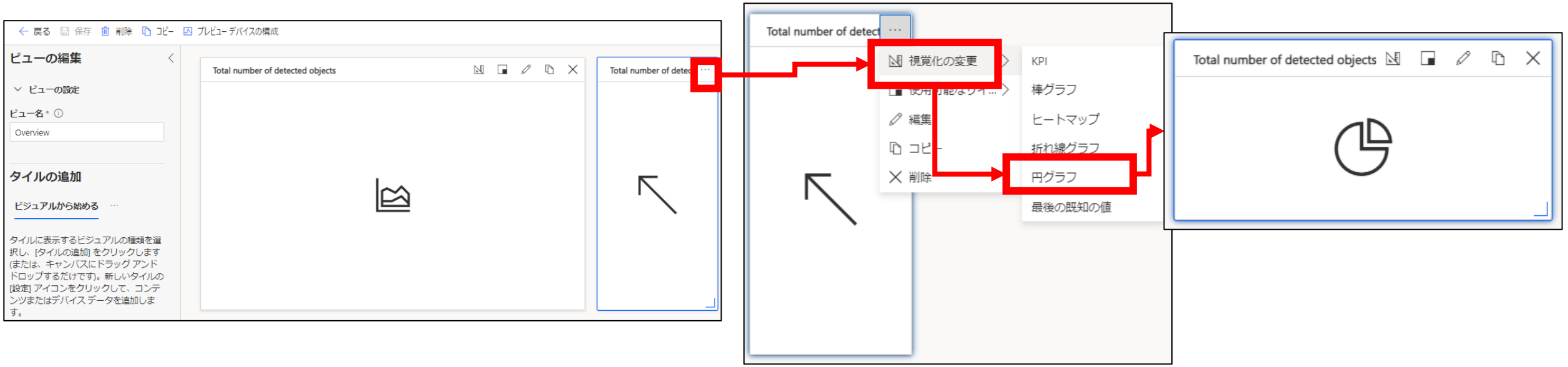
ビュー作成

「Overview」ビュー



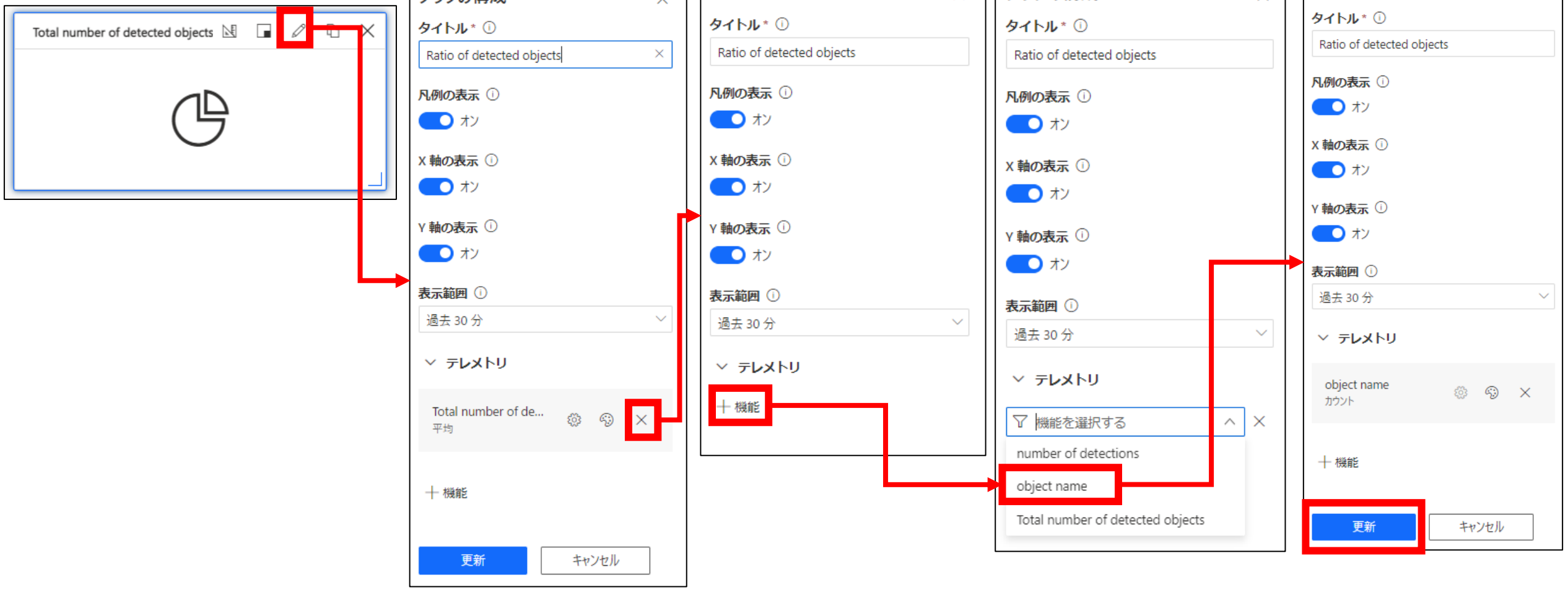
ビュー作成

「Overview」ビュー



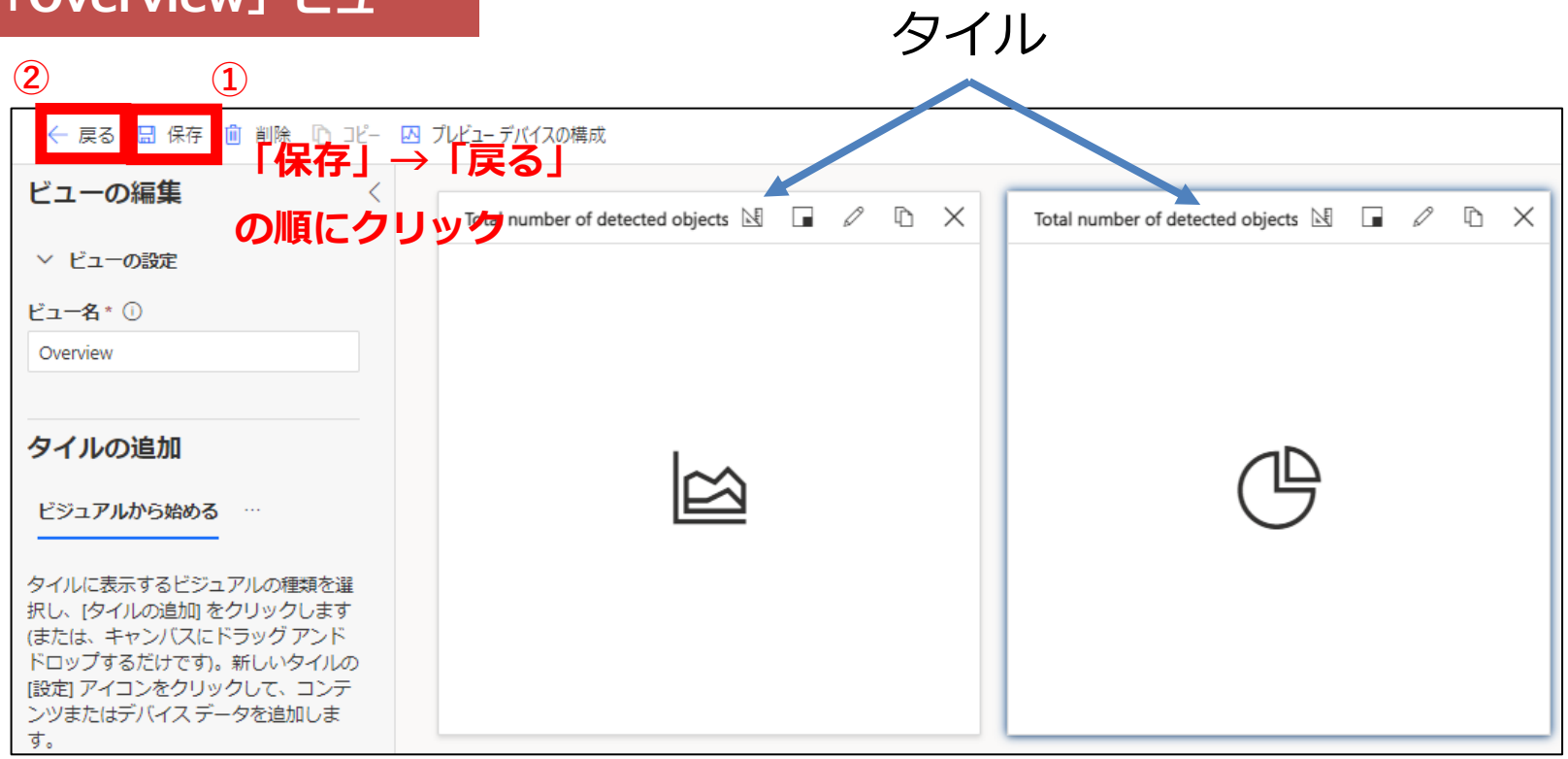
ビュー作成

「Overview」ビュー



ビュー作成

「Overview」ビュー

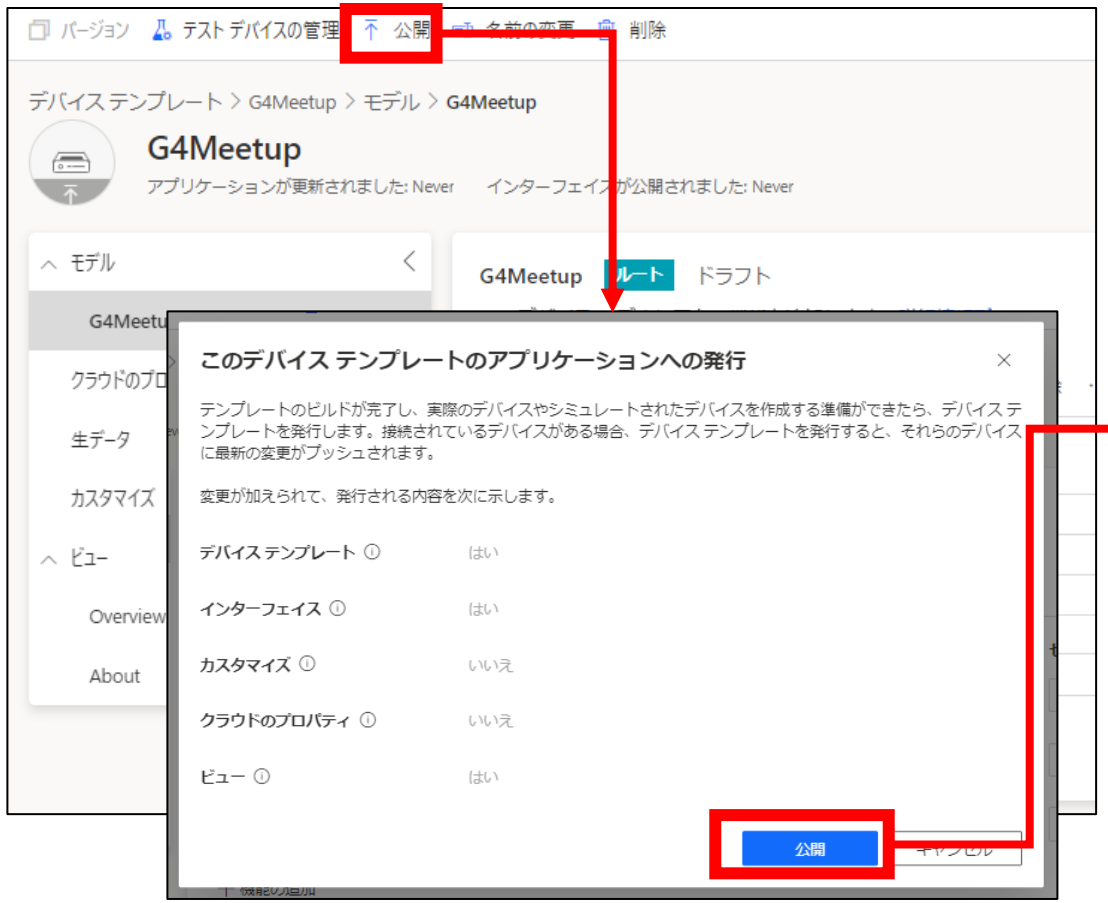


ウインドウサイズ調整の要領で

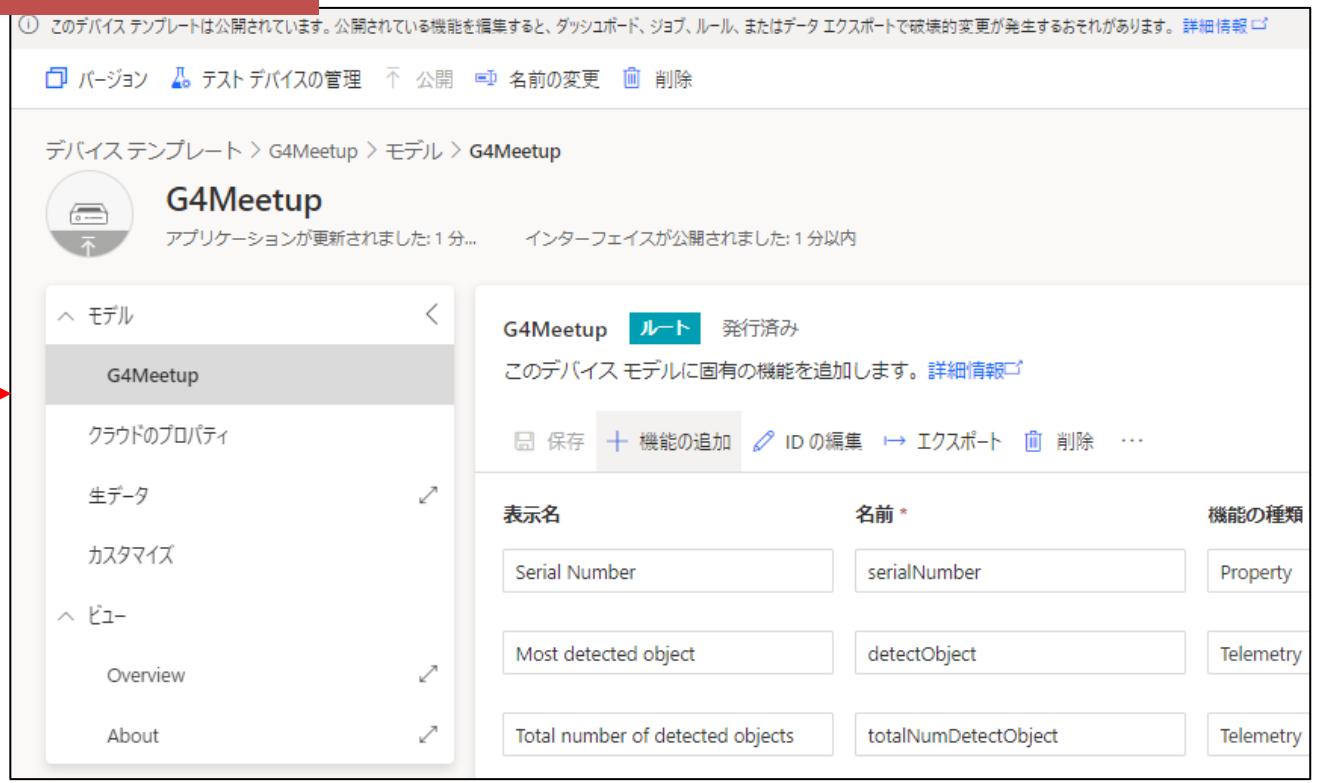
各タイルの大きさの調整をすることができます

デバイステンプレート公開

デバイスを接続する前に、デバイステンプレートを「公開(publish)」する必要があります。

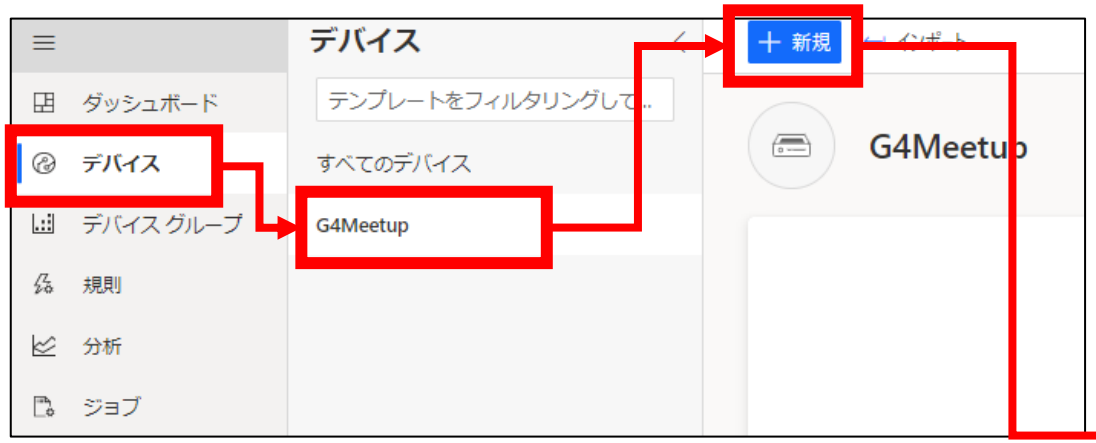


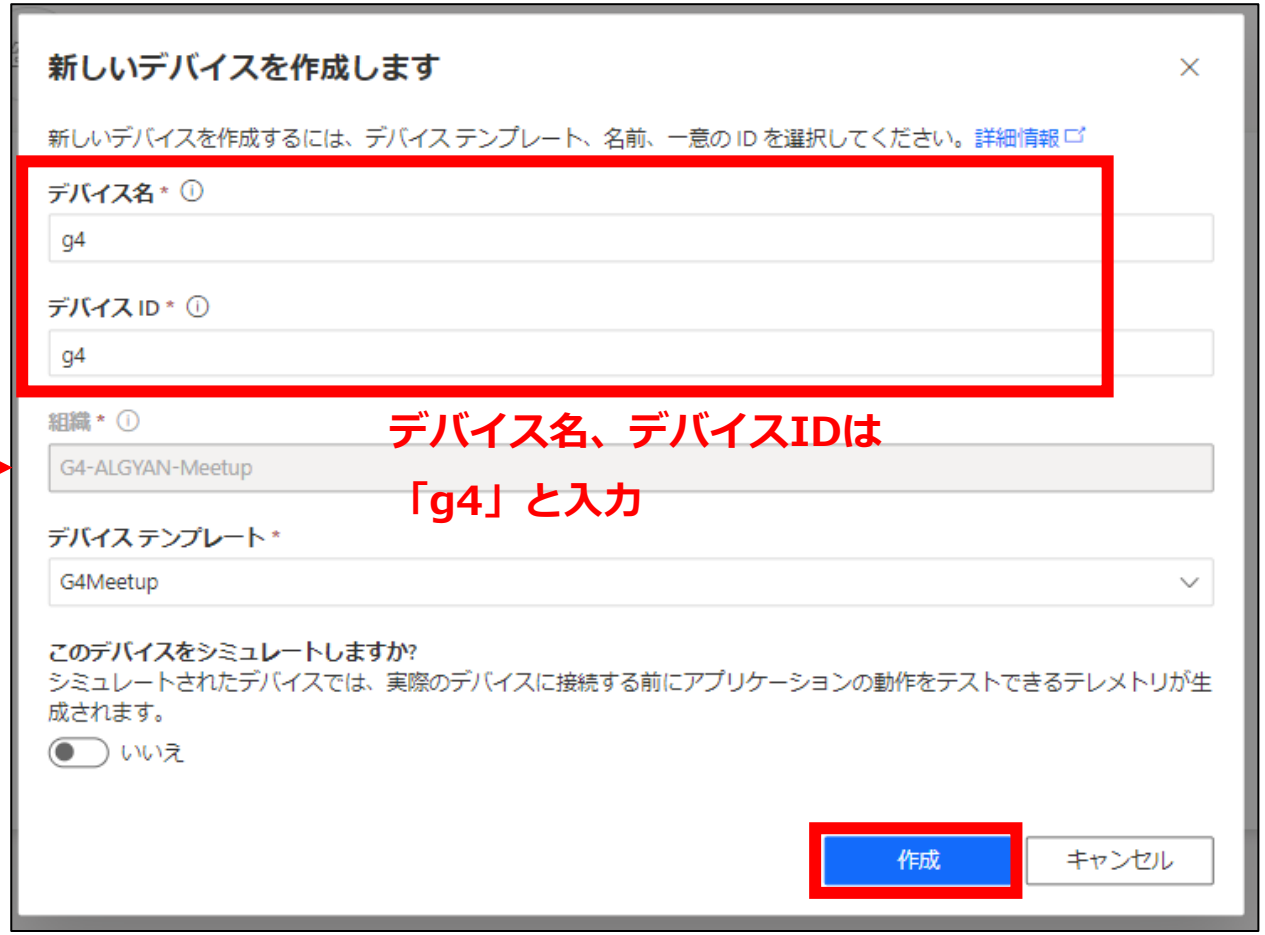
公開完了



デバイス作成

接続情報を取得するため、デバイスを登録します。



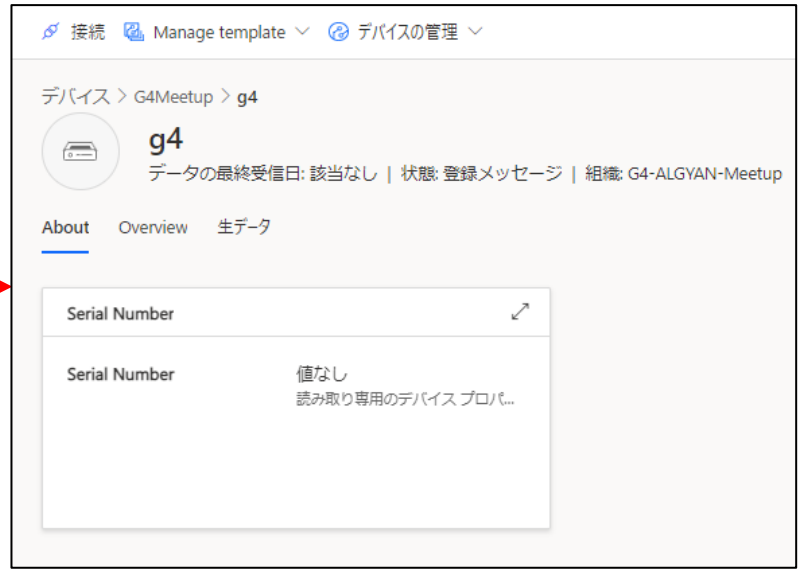


デバイス名、デバイスIDは
「g4」と入力

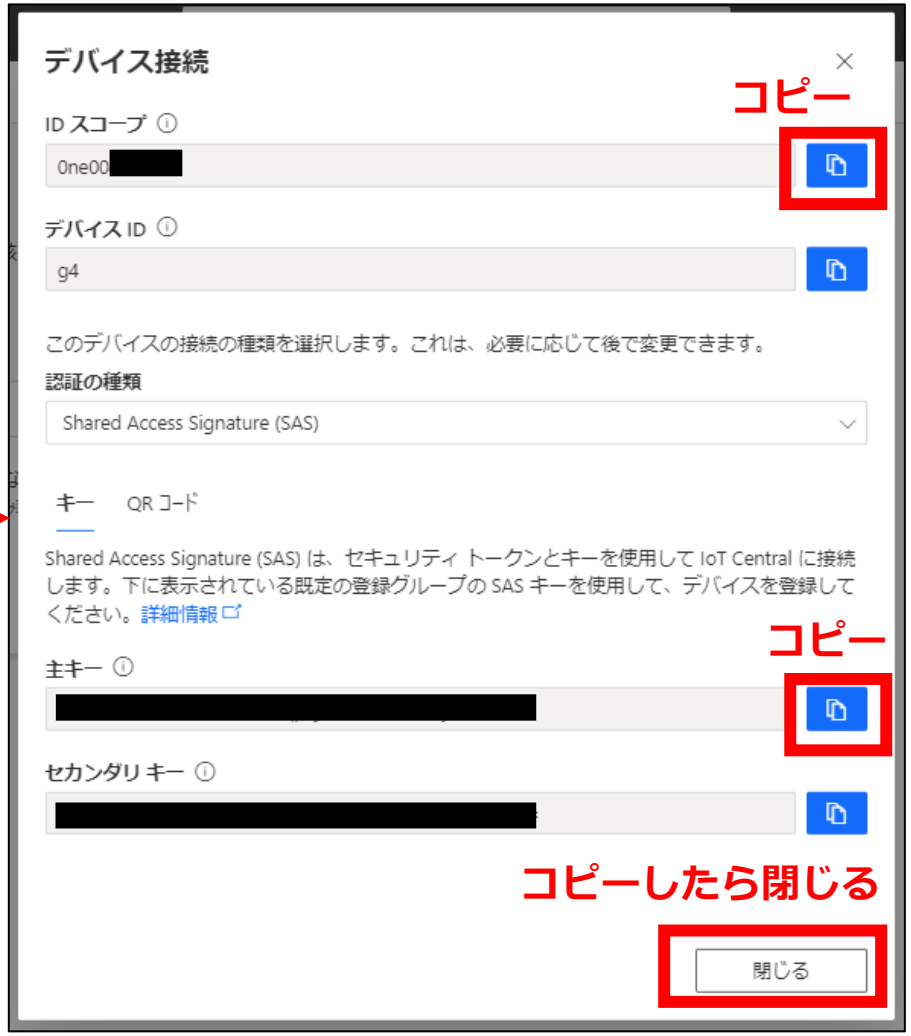
デバイス作成

デバイス作成後画面

デバイス名をクリックすると、
詳細を見ることができます。



デバイス接続情報取得



必要な情報は下記の3つ

- IDスコープ
- デバイスID　※今回は固定
- 主キー

接続設定ファイル編集

①PCにUSBメモリを挿し、予めDLした g4_basic_config.json をエディタで開く

②下記の箇所を編集、保存する

編集前

```
{
 "auth": {
  "IOTHUB_DEVICE_DPS_ENDPOINT": "global.azure-devices-provisioning.net",
  "IOTHUB_DEVICE_DPS_ID_SCOPE": "",
  "IOTHUB_DEVICE_DPS_DEVICE_ID": "g4",
  "IOTHUB_DEVICE_DPS_DEVICE_KEY": ""
 },
  (省略)
}
```

IDスコープ

デバイス主キー

編集後

```
{
 "auth": {
  "IOTHUB_DEVICE_DPS_ENDPOINT": "global.azure-devices-provisioning.net",
  "IOTHUB_DEVICE_DPS_ID_SCOPE": "0ne00xxxxxx",
  "IOTHUB_DEVICE_DPS_DEVICE_ID": "g4",
  "IOTHUB_DEVICE_DPS_DEVICE_KEY": "xxxxxx/xxxxxxxxxxxxxxxxxxxxx"
 },
  (省略)
}
```

・コピペ誤りなど、編集箇所を間違うと Armadillo が IoT Centralに接続できないためご注意ください

・エディタに Visual Studio Code を使用している場合は自動的にjson構文チェックが行われます

接続設定ファイル書き込み

USBメモリ

①PCからUSBメモリを取り外し、ArmadilloのUSBコネクタに挿す

②USBメモリをマウントする

```
armadillo:~# mount /dev/sda1 /mnt
```

③前作業で編集した設定ファイルがあるか確認

```
armadillo:~# ls /mnt/g4_basic_config.json
/mnt/g4_basic_config.json
```

④設定ファイルを コンテナ内にコピー

```
armadillo:~# podman cp /mnt/g4_basic_config.json azure:/root/Azure-IoT-samples/Armadillo-IoT_GW/.
```

接続設定ファイル書き込み

USBメモリ

⑤コピーできているか確認

```
armadillo:~# podman exec azure cat /root/Azure-IoT-samples/Armadillo-IoT_GW/g4_basic_config.json
{
  "auth": {
    "IOTHUB_DEVICE_DPS_ENDPOINT": "global.azure-devices-provisioning.net",
    "IOTHUB_DEVICE_DPS_ID_SCOPE"  " 0ne00xxxxxx",
    "IOTHUB_DEVICE_DPS_DEVICE_ID": "g4",
    "IOTHUB_DEVICE_DPS_DEVICE_KEY"  "xxxxxx/xxxxxxxxxxxxxxxxxxxxxx"
  },
(省略)
}
```

IDスコープ

デバイス主キー

⑥USBメモリをアンマウントし、Armadilloから取り外す

```
armadillo:~# umount /mnt
```

IoT Central接続確認

①Azureコンテナの中に入る

```
armadillo:~# podman exec -it azure /bin/sh
/ #
```

②スクリプト実行

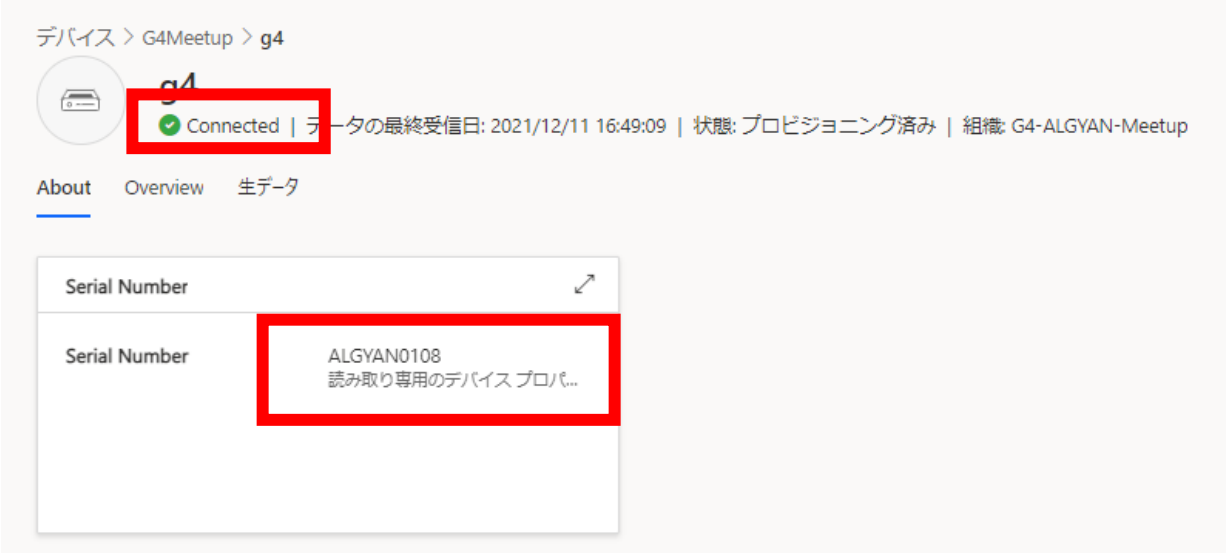
```
/ # python3 /root/Azure-IoT-samples/Armadillo-IoT_GW/azure_g4_basic.py
```

デバイスが接続されると以下のログが表示されます。

```
Device was assigned
iotc-xxx-xxx-xxx-xxx-xxx.azure-devices.net
g4
Press Q to quit
```

IoT Central 接続確認

③ IoT Central 上で接続を確認する



④スクリプト停止

```
Device was assigned
iotc-xxx-xxx-xxx-xxx-xxx.azure-devices.net
g4
Press Q to quit
Q
Quitting...
all threads are quited, then shutdown the IoT client...
```

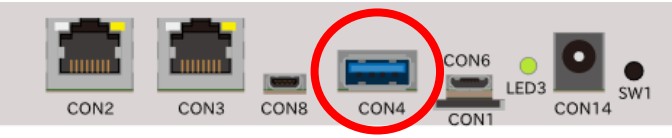
Q を入力してエンター

物体認識 + Azure 実行

①Azureコンテナから出る

```
/ # exit
armadillo:~#
```

②ArmadilloにWebカメラを接続する



③Azure 接続スクリプトを実行する

```
armadillo:~# podman exec -d azure python3 /root/Azure-IoT-samples/Armadillo-IoT_GW/azure_g4_basic.py -d
```

④物体認識を実行する

- **Webカメラあり**

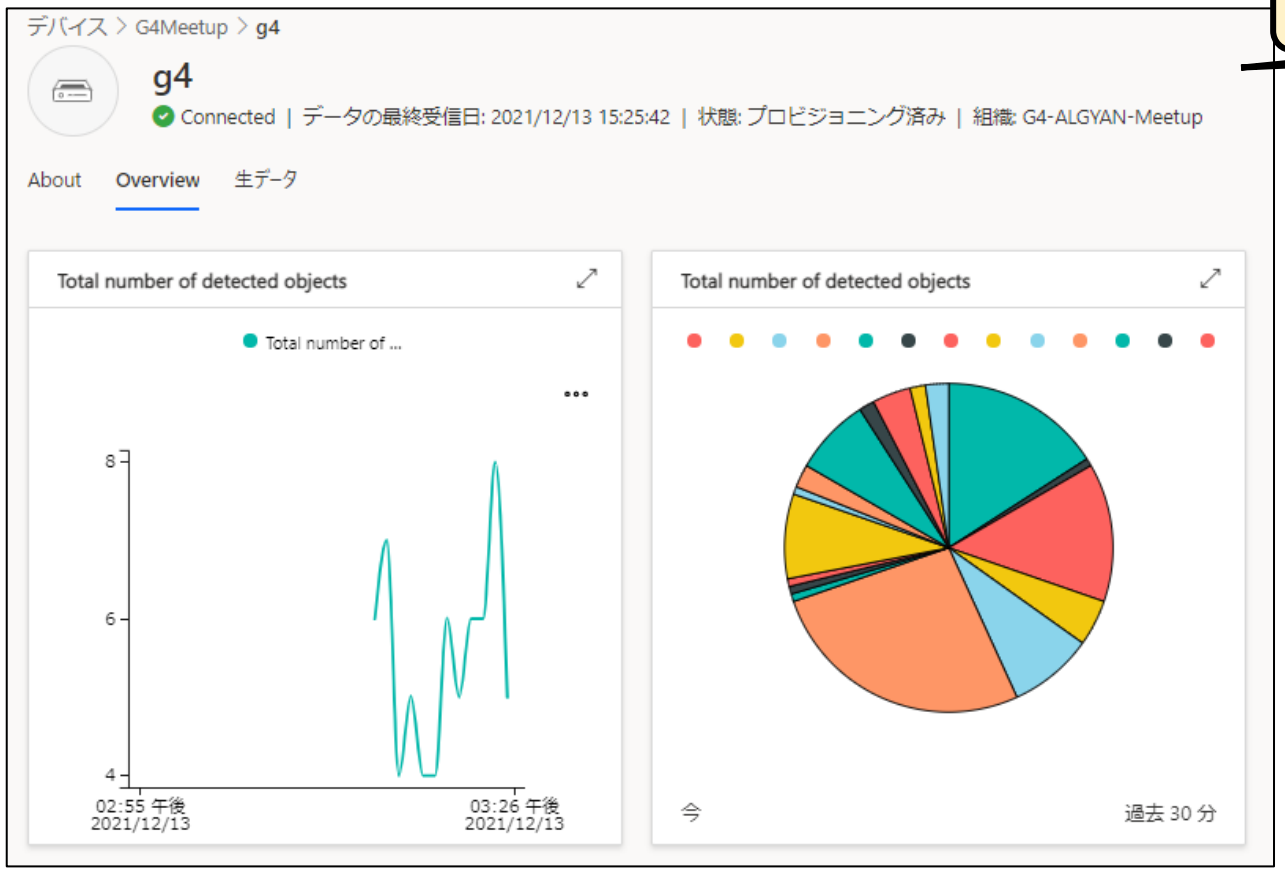
```
armadillo:~# podman exec -d detect /root/handson.sh --camera --cloud
```

- **Webカメラなし(動画を使用)**

```
armadillo:~# podman exec -d detect /root/handson.sh --cloud
```

物体認識 + Azure 実行

IoT Central デバイス画面



およそ 1分周期で表示が更新されます