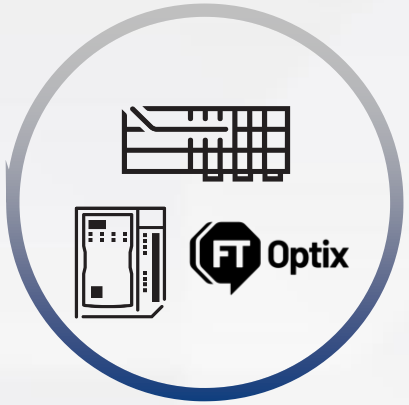
Rockwell Automation

Usage of Docker on 1756-CMEE

expanding **human possibility**®

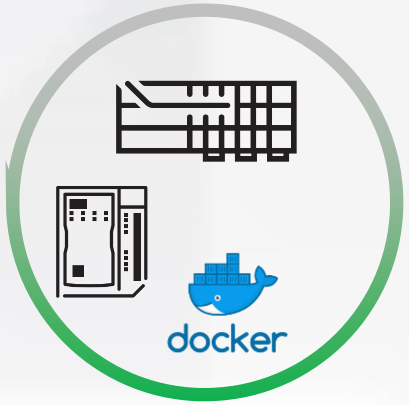
Embedded Edge Compute Program Priorities

Optimize machine & equipment control while enabling additional OT and IT technologies at the Edge



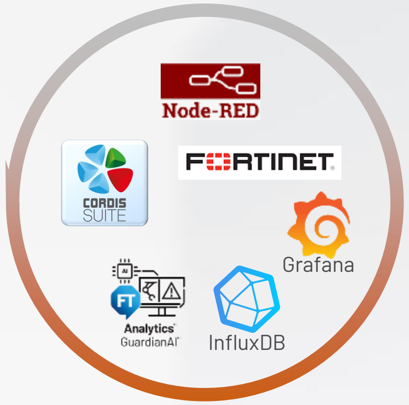
Edge and Control

Bring contemporary IT/OT technologies in-rack alongside world class multi-discipline control that is Modular, Secure and Safe



Containerized Applications

Provide flexibility to the Logix platform including capabilities to support containerized applications



Application Ecosystem

Expand the ecosystem of deployable applications to enable more complimentary value to the Logix platform

Rockwell Automation

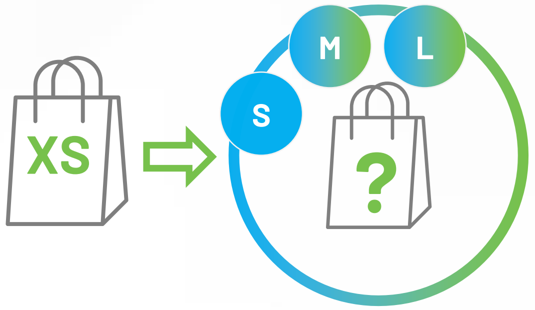
Release 1 – Q4FY23

Embedded Edge Compute™ hosting FactoryTalk® Optix™

Embedded hardware solution optimized for FactoryTalk® Optix™

Embedded Edge Compute

- Scaled to meet wide range of customer needs:
  - **Greenfield and Brownfield applications**
  - OEM focus – small to large machines, simple to complex applications
  - End User Focus – Simple to complex Machines and Applications
- **Released in September 2023**
- Includes
  - FactoryTalk® Optix™ Runtime **Xtra-Small**
  - FactoryTalk® Remote Access™ Runtime **Pro**
- Optional License Upgrade
  - FactoryTalk® Optix™ Runtime (Small-L)
- Specifications
  - ARM NXP iMX8M Plus
  - 50GB+ User Memory
  - Linux Yocto 64-bit OS



Rockwell Automation

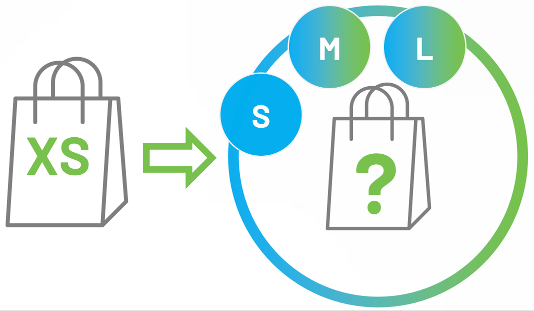
Container Support – Q1FY25

Embedded Edge Compute™ Containers enabled

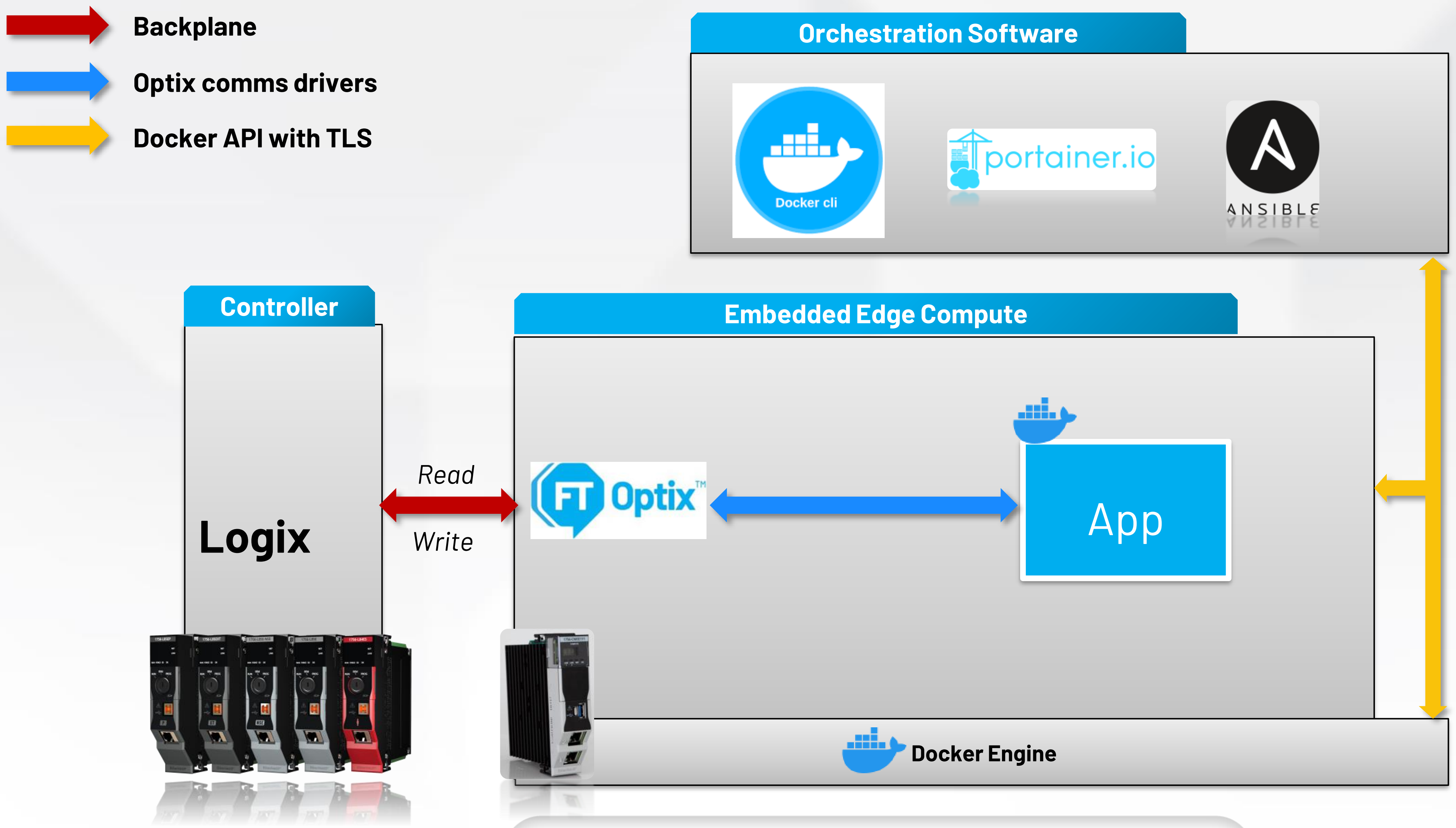
Open, Secure and IT/OT ready co-processor for Logix Controllers

Embedded Edge Compute

- Scaled to meet wide range of customer needs:
  - Greenfield and Brownfield applications
  - OEM focus – small to large machines, simple to complex applications
  - End User Focus – Simple to complex Machines and Applications
- Target Release Q1FY25
- Includes
  - FactoryTalk® Optix™ Runtime Xtra-Small
  - FactoryTalk® Remote Access™ Runtime Pro
  - **Docker Engine**
- Optional License Upgrade
  - FactoryTalk® Optix™ Runtime (Small-L)
- Specifications
  - Same as Release 1



Embedded Edge Compute Containerization



Embedded Edge Compute™ Containers enabled

Open, Secure and IT/OT ready co-processor for Logix Controllers

Open

Following software can run on the module:

- RA apps
- RA Partners apps
- Custom apps
- Third-party apps

Secure

- Closed host OS
- Encrypted Connections

IT/OT ready

- App deployment options

FT Optix™

HTML 5, OPC UA, MQTT, REST API, C#, SQLite, MySQL

FT Remote Access

docker

CORDIS SUITE, Node-RED, influxdb, Grafana, FT Analytics™ GuardianAI™, FORTINET, GitHub

Rockwell Automation

Containerization

Rockwell Automation

Workshop goals

From zero to hero with Docker and Portainer on the 1756-CMEE

Docker

Key concepts of containerization, what is a container, why to use a container, what is Docker.

Portainer

Key concepts of orchestrators, what is Portainer, why to use Portainer

Deploy a container

Enablement of Docker on the 1756-CMEE, enabling Portainer CE, deployment of a container, use of Docker CLI

Build a custom container

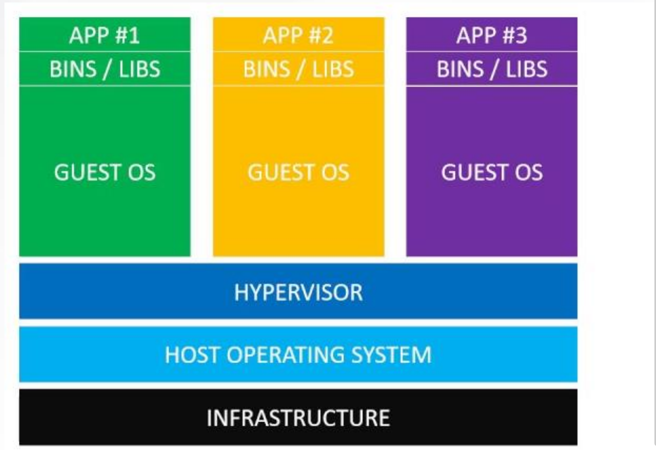
What is a Dockerfile, what is Docker Composer, create a custom container image for the 1756-CMEE.

Containerization

How to pack many things into a carry-on bag

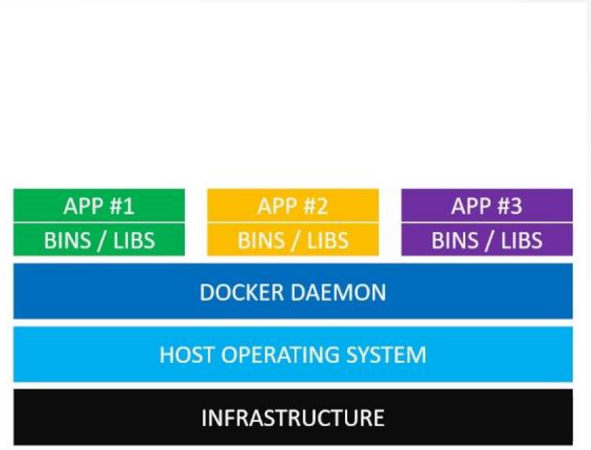
Virtual Machine

- Needs lot of resources
- Needs specific virtualization software
- Weighs some tenth of GB
- Contains many things that are not strictly necessary



Container

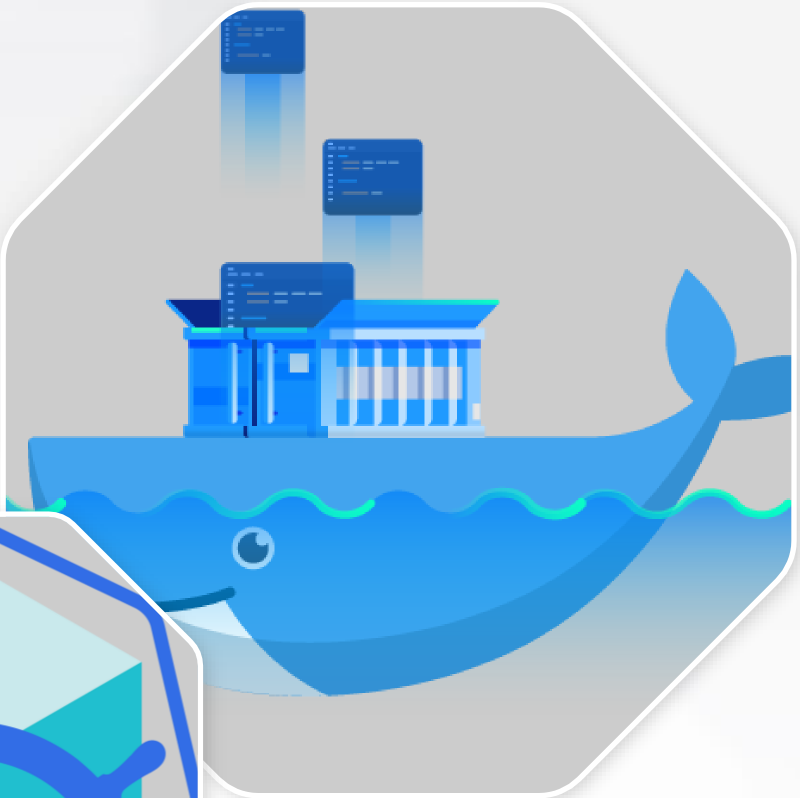
- Needs only the resources used by the executable
- Can run on any host OS with a container agent
- Weighs few KB (or even just as a single text file)
- Only contains the main executable and dependencies



Containerization softwares

Docker is not the only one, it is just one of the most famous

Podman is an open-source tool which is 1:1 compatible with Docker



Docker offers a simple and efficient approach to running and managing containers, but Kubernetes offers more complex capabilities, such as automated container deployment, scalability, and self-healing

Minikube is a lightweight Kubernetes nabagenebt tool with advanced features like load balancing and Add-Ons

Containerd is an industry-standard container runtime with an emphasis on simplicity, robustness, and portability

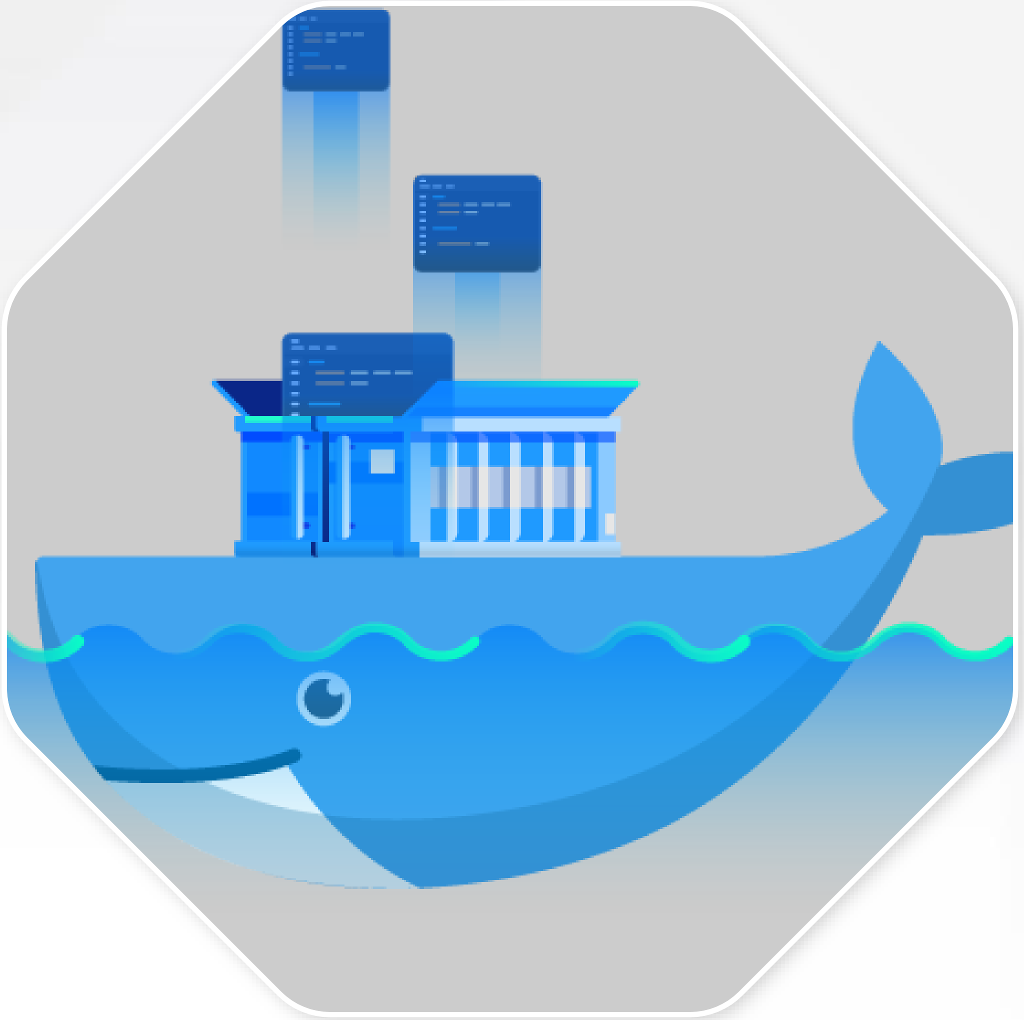
Why Docker?

Good advertisements and customer care

Mostly focused on every-day development by supporting Linux, Windows and MAC OS

Large community of both professionist and amateurs which create a strong engagement

Constantly updated and supported



Simplicity and good documentation

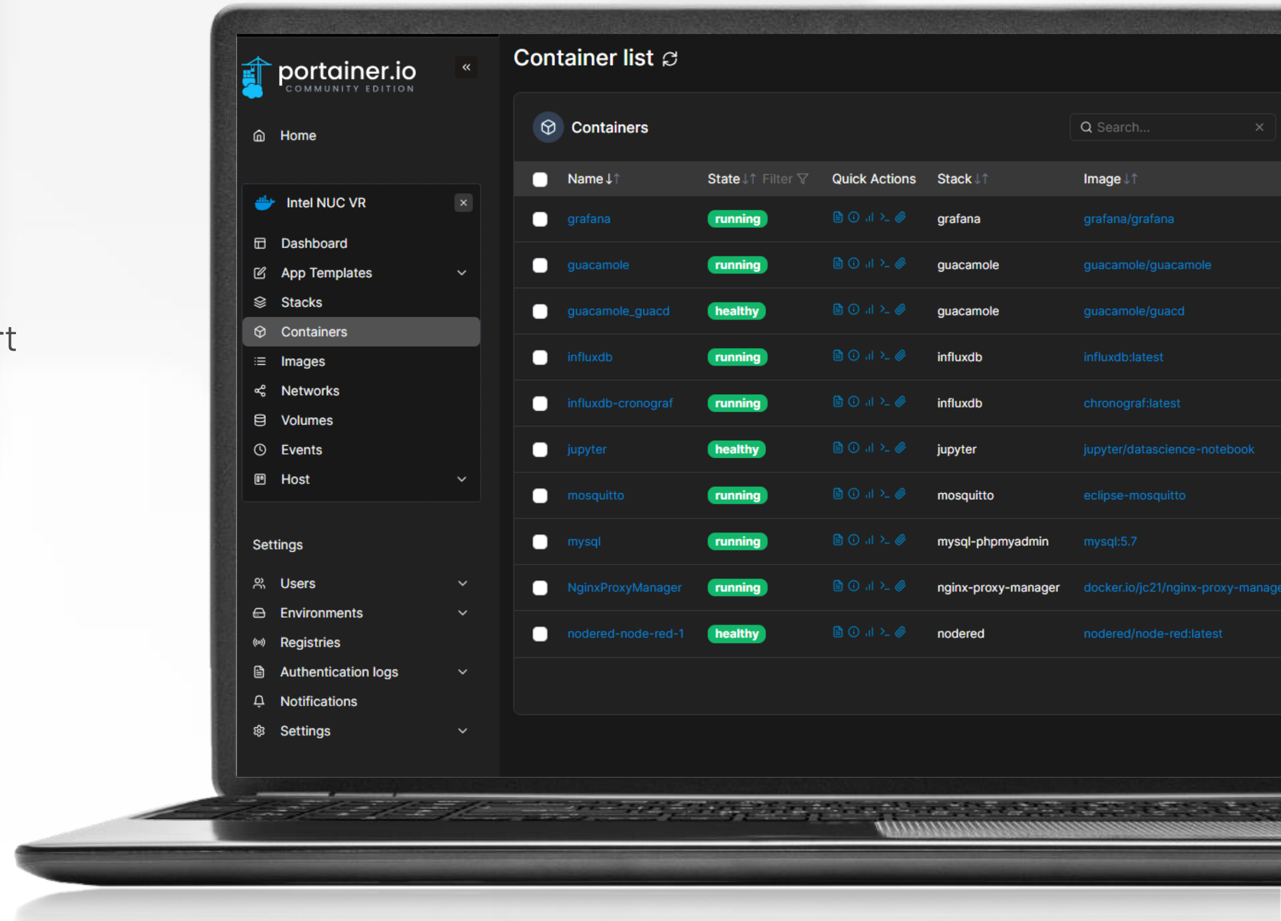
Containers can be easily ported across different systems and host OS

Support for scalability

Portainer

Portainer is actually a Docker container

- Portainer is not a containerization platform
- Portainer is a GUI for Docker
- Comes in two flavors:
  - Portainer-ce which is free with some minor features limitations
  - Portainer-business which is paid and includes all features and support
- Allows connecting to multiple agents
- Supports both Docker files and Docker compose



Container list

Containers

Name	State	Quick Actions	Stack	Image
grafana	running		grafana	grafana/grafana
guacamole	running		guacamole	guacamole/guacamole
guacamole_guacd	healthy		guacamole	guacamole/guacd
influxdb	running		influxdb	influxdb:latest
influxdb-cronograf	running		influxdb	chronograf:latest
jupyter	healthy		jupyter	jupyter/datascience-notebook
mosquitto	running		mosquitto	eclipse-mosquitto
mysql	running		mysql-phpmyadmin	mysql:5.7
NginxProxyManager	running		nginx-proxy-manager	docker.io/jc21/nginx-proxy-manage
nodered-node-red-1	healthy		nodered	nodered/node-red:latest

Docker Resources

“Before you start”

Docker: Getting started

- https://docker-curriculum.com/
- https://docs.docker.com/get-started/

Docker Workshops

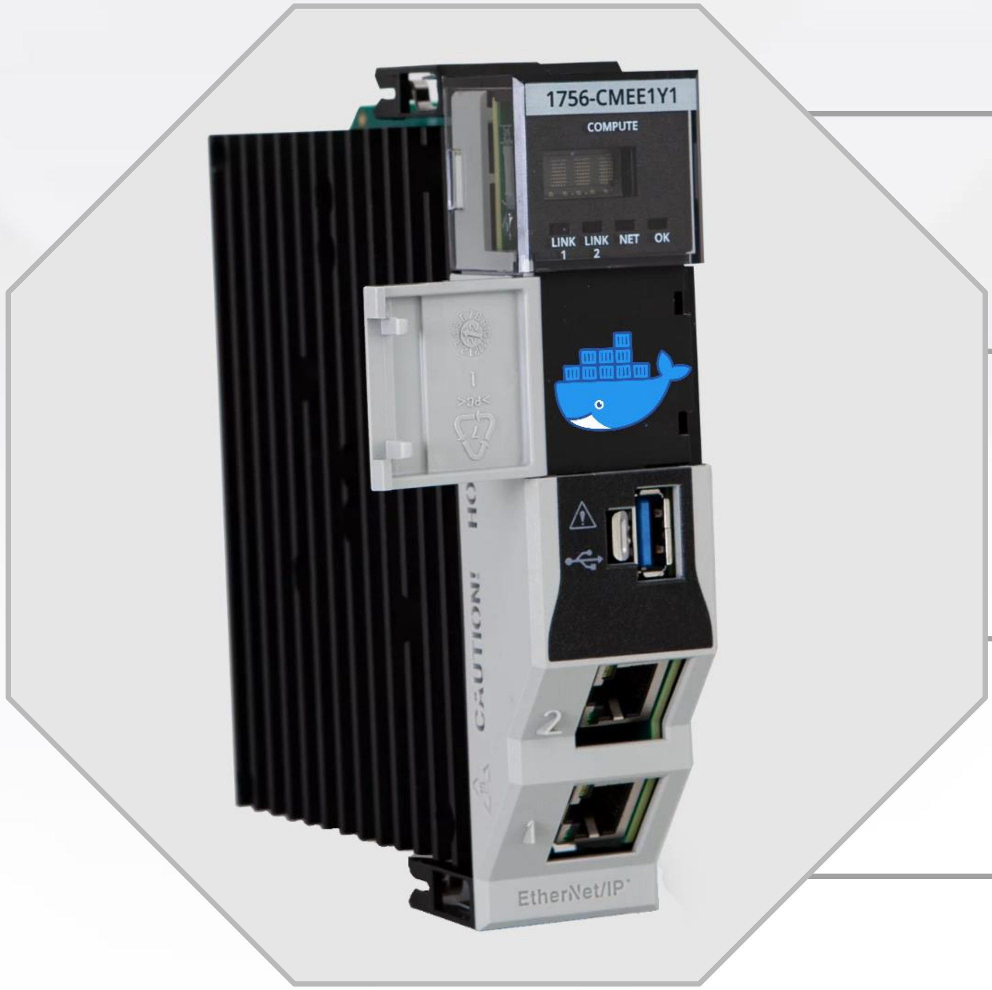
- https://docs.docker.com/get-started/workshop/02_our_app/

Docker Registry

- https://docs.docker.com/registry/

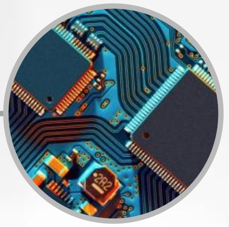
What is the 1756-CMEE

Basically, it is an OptixPanel™ without the Panel



Backplane mounting

Native communication using the backplane of the 1756 rack. No need for external power.



Motherboard

Same hardware as an OptixPanel Standard. 4GB of RAM, non-removable SD card, ARM64 CPU

I/O

1x USB 3.0 port, 2x Gigabit network interfaces

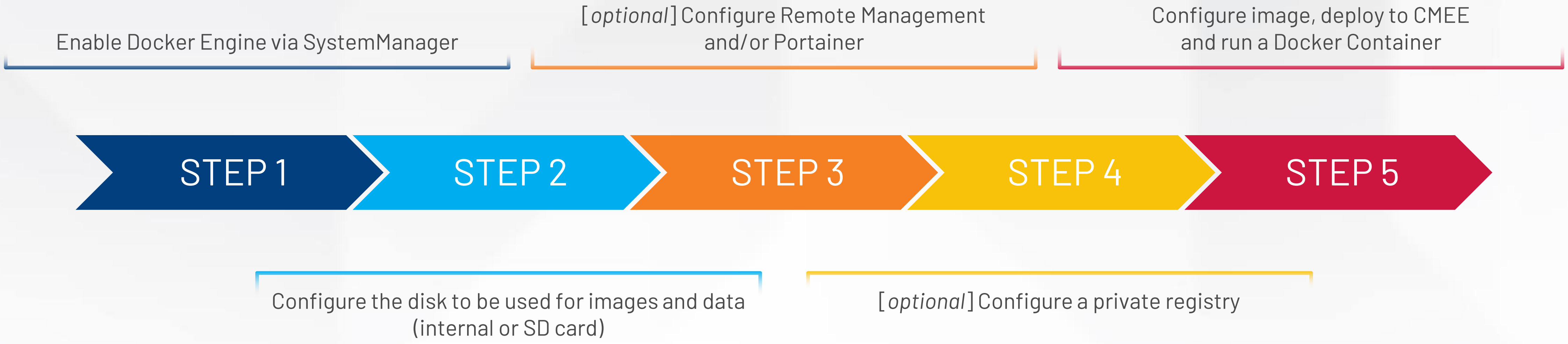
no video output, no wireless options

Software

Linux Yocto with RA SystemManager, support for Docker engine.

Enabling Docker

Docker is not activated by default



STEP 1: Enable Docker Engine via SystemManager

STEP 2: Configure the disk to be used for images and data (internal or SD card)

STEP 3: [*optional*] Configure Remote Management and/or Portainer

STEP 4: [*optional*] Configure a private registry

STEP 5: Configure image, deploy to CMEE and run a Docker Container

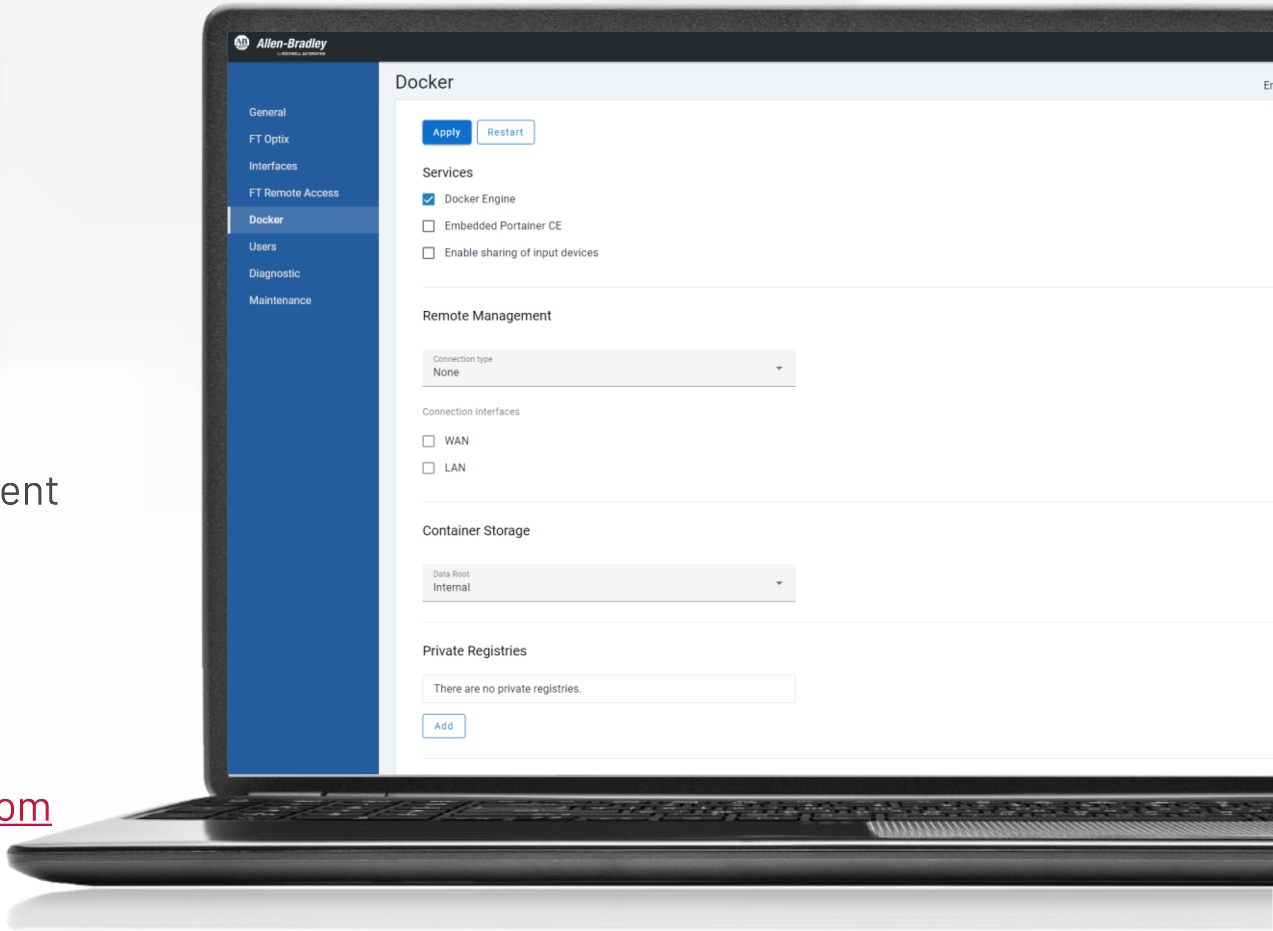
Configure Docker Engine

Rockwell Automation

Enable Docker [SystemManager]

Tick the option and Apply -> device will **reboot**

- Access the SystemManager page of the device
- Enable the **Docker Engine**
- [optional] Enable Portainer CE
  - On-device orchestrator with web-based user interface
- [optional] Configure the Remote Management
  - Exposes the Docker API to the network for remote management
- Configure the Container Storage
  - Where to store images and persistent data
- [optional] Configure Private Registries
  - Where to pull images from, if not using https://hub.docker.com
- [optional] Configure Proxy
  - If required by the network



Docker

Services

[✓] Docker Engine
[ ] Embedded Portainer CE
[ ] Enable sharing of input devices

Remote Management

Connection type: None

Connection interfaces

[ ] WAN
[ ] LAN

Container Storage

Data Root: Internal

Private Registries

There are no private registries.

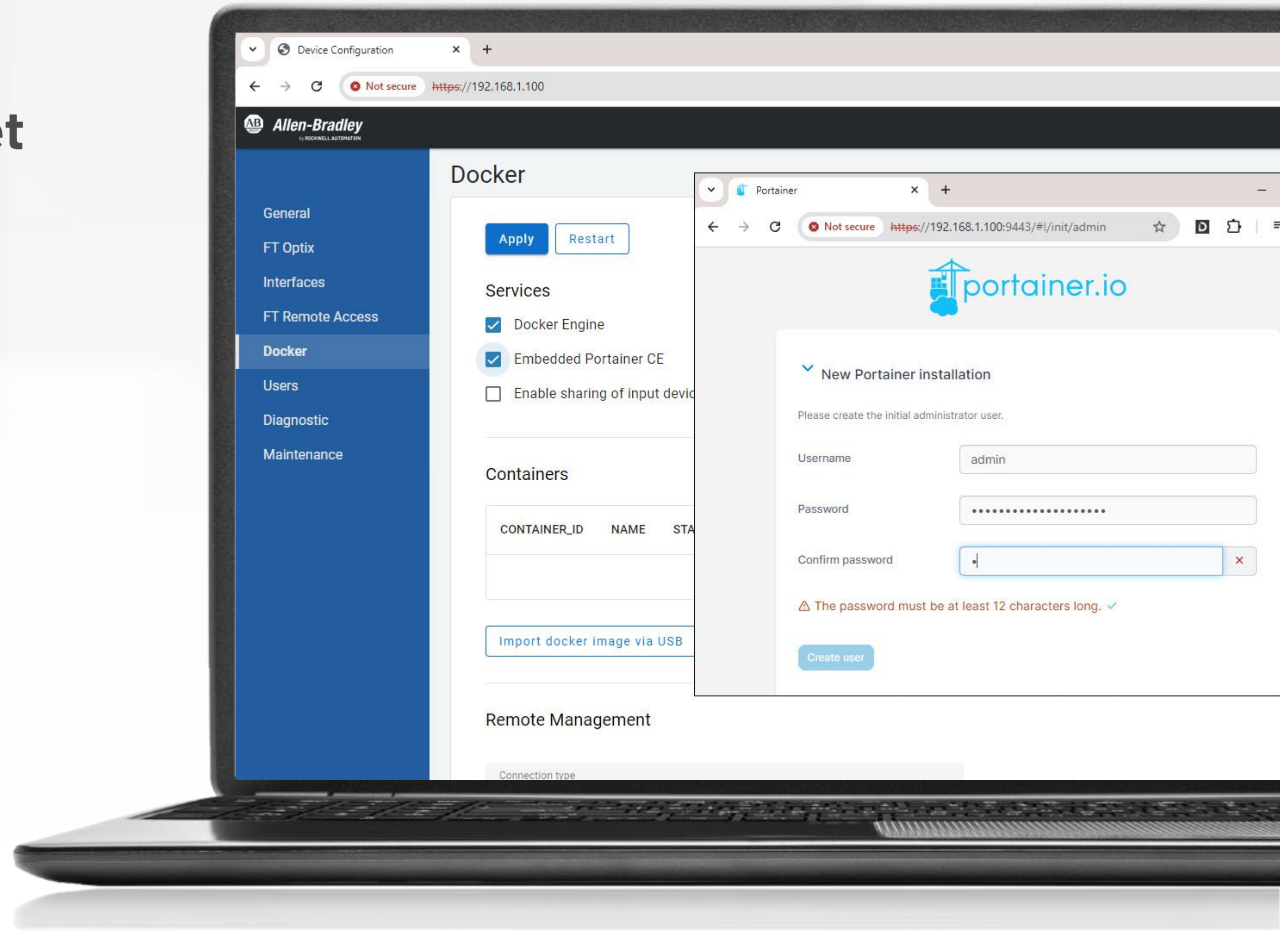
Install Portainer [SystemManager]

[Optional] -> install on-device orchestrator

- Access the SystemManager page of the device
- Make sure the module is **connected to the Internet**
  - [typically Port 1 – WAN]
- Enable the **Embedded Portainer CE**

- Restart
- Open Portainer webpage of the module
  - https://ip-address-of-the-module:9443
- Configure Admin password

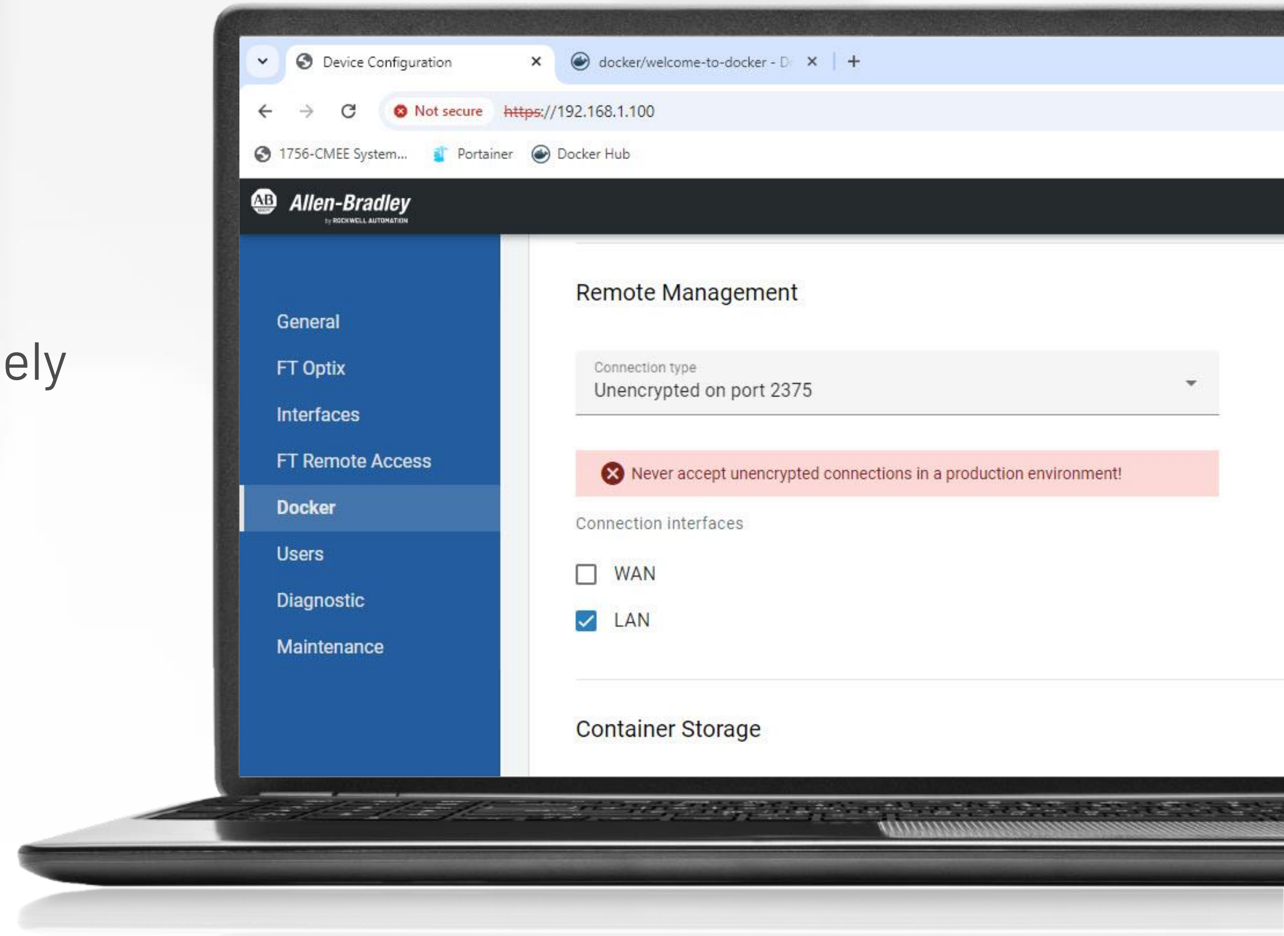
- Portainer becomes visible in the list of containers



Remote Management [SystemManager]

[Optional] -> Allow management via (e.g.) Docker CLI

- Access the SystemManager page of the device
- Enable Remote Management
  - Select which connection type to be used
  - Select which interfaces can be used to access
- Docker on 1756-CMEE can now be managed remotely (e.g. Docker CLI, or compatible orchestrator)
- Restart



Remote Management

Connection type: Unencrypted on port 2375

Never accept unencrypted connections in a production environment!

Connection interfaces

- [ ] WAN
- [x] LAN

Container Storage

Rockwell Automation

Prepare a Docker **image** [**pre-built**]

Pull image from Docker Hub

Get a pre-built image from registry

- Find the desired image at https://hub.docker.com/
  [could be pulled from any configured registry ...]

- Copy the pull command (and add `--platform linux/arm64`)

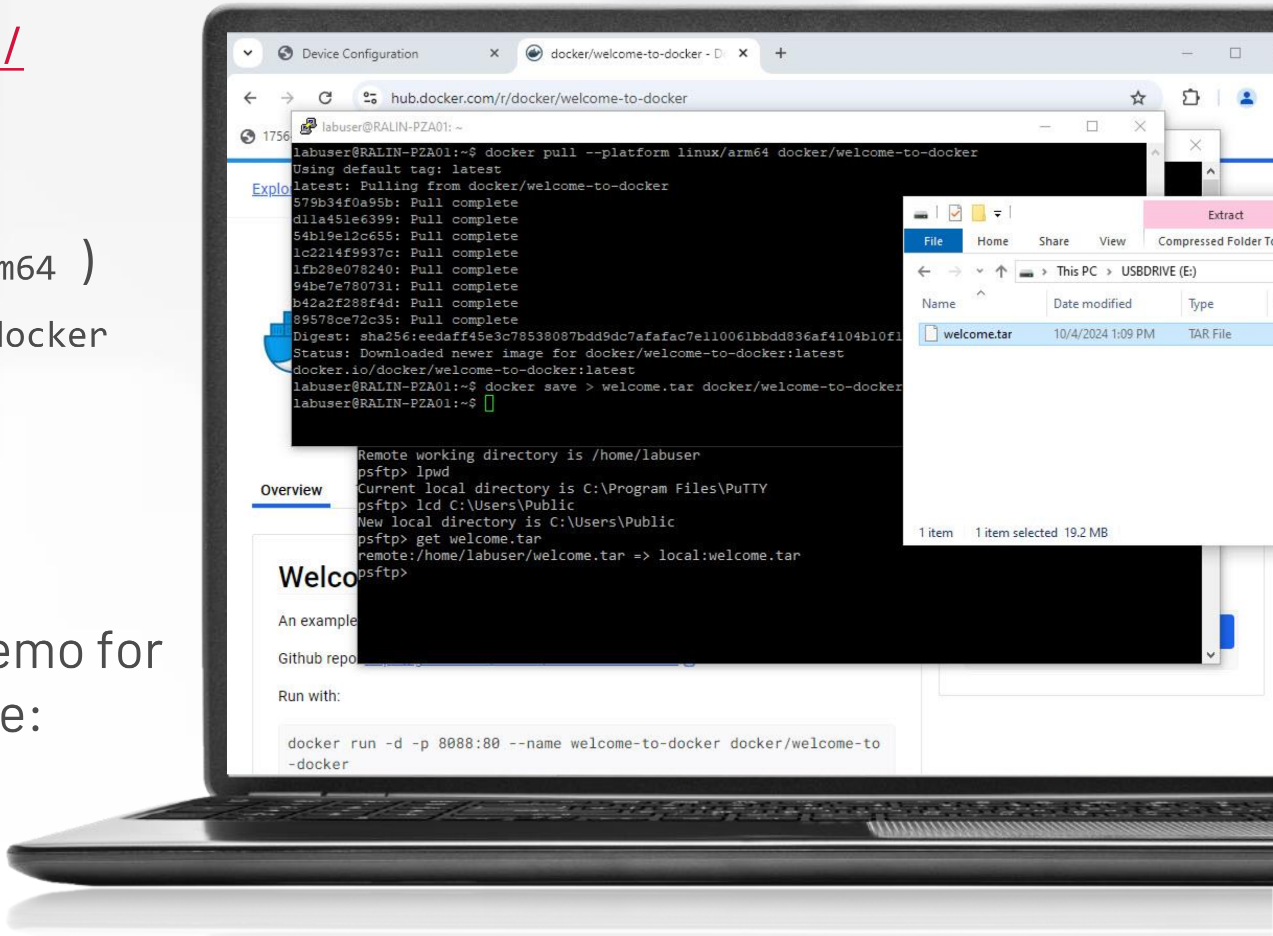
```
$ docker pull --platform linux/arm64 docker/welcome-to-docker
```

- Export image to tar file

```
$ docker save > welcome.tar docker/welcome-to-docker
```

- **<u>Tip</u>**: **PSFTP** (part of PuTTY package) used in this demo for file transfer from Linux server to Windows machine:

```
> lcd C:\Users\Public
> get welcome.tar
```



```
labuser@RALIN-PZA01:~$ docker pull --platform linux/arm64 docker/welcome-to-docker
Using default tag: latest
latest: Pulling from docker/welcome-to-docker
579b34f0a95b: Pull complete
d11a451e6399: Pull complete
54b19e12c655: Pull complete
1c2214f9937c: Pull complete
1fb28e078240: Pull complete
94be7e780731: Pull complete
b42a2f288f4d: Pull complete
89578ce72c35: Pull complete
Digest: sha256:eedaff45e3c78538087bdd9dc7afafac7e110061bbdd836af4104b10f1
Status: Downloaded newer image for docker/welcome-to-docker:latest
docker.io/docker/welcome-to-docker:latest
labuser@RALIN-PZA01:~$ docker save > welcome.tar docker/welcome-to-docker
labuser@RALIN-PZA01:~$
```

```
Remote working directory is /home/labuser
psftp> lpwd
Current local directory is C:\Program Files\PuTTY
psftp> lcd C:\Users\Public
New local directory is C:\Users\Public
psftp> get welcome.tar
remote:/home/labuser/welcome.tar => local:welcome.tar
psftp>
```

```
docker run -d -p 8088:80 --name welcome-to-docker docker/welcome-to
-docker
```

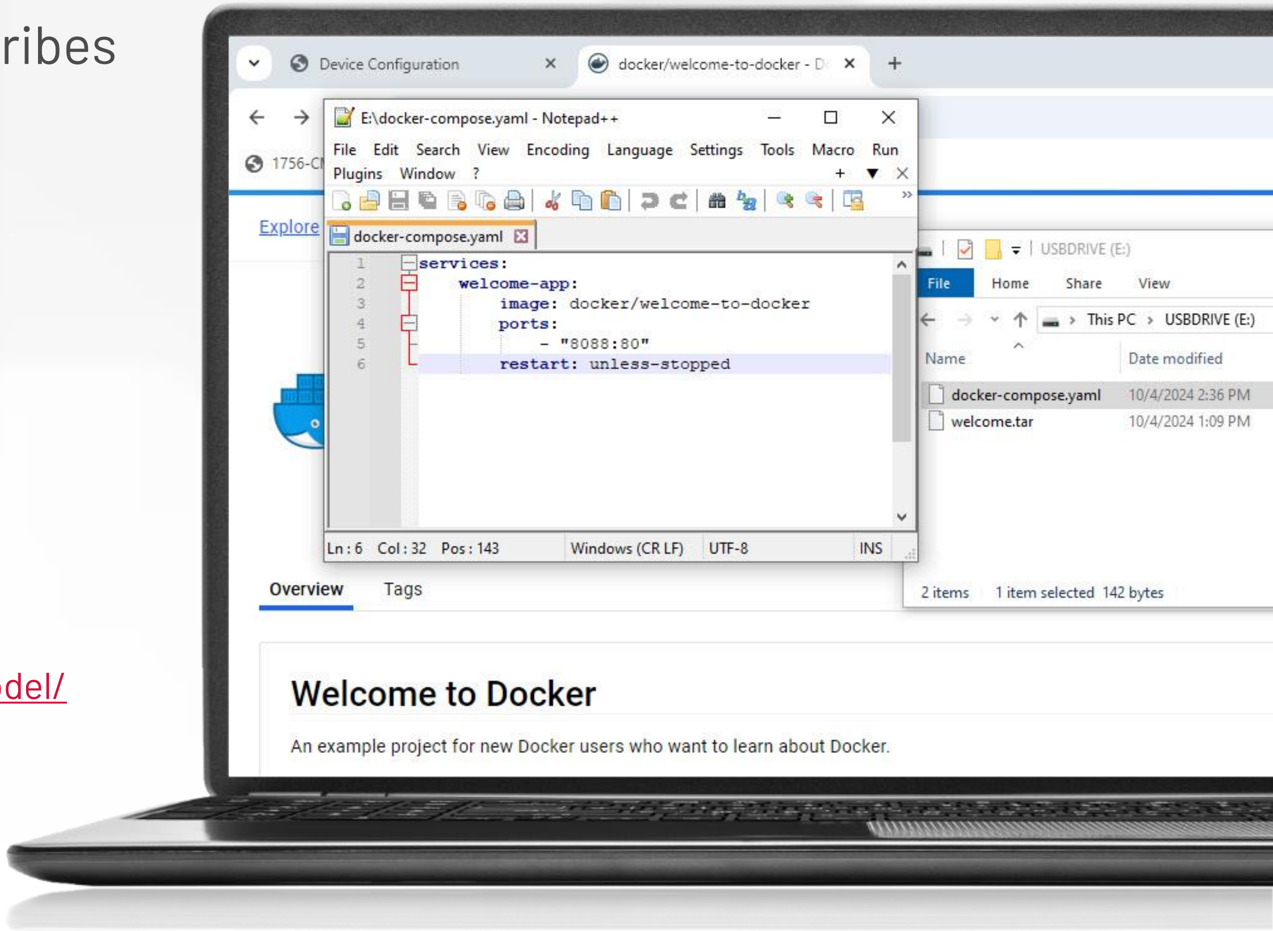
Create a Docker Compose file

Configure the container

- Create a text file **docker-compose.yaml** that describes the container startup procedure and arguments

```
services:
    welcome-app:
        image: docker/welcome-to-docker
        ports:
            - "8088:80"
        Restart: unless-stopped
```

- References:
  - https://docs.docker.com/compose/intro/compose-application-model/
  - https://docs.docker.com/reference/compose-file/



```
services:
    welcome-app:
        image: docker/welcome-to-docker
        ports:
            - "8088:80"
        restart: unless-stopped
```

Ln : 6 Col : 32 Pos : 143 Windows (CR LF) UTF-8 INS

docker-compose.yaml 10/4/2024 2:36 PM
welcome.tar 10/4/2024 1:09 PM

2 items 1 item selected 142 bytes

Welcome to Docker

An example project for new Docker users who want to learn about Docker.

Manage Containers on 1756-CMEE

Rockwell Automation

Manage Containers (on 1756-CMEE)

Three ways to manage Docker, coming with module "out of box"

Portainer CE

Comes with module (click + download)

Local orchestrator engine "onboard"

Web-based user interface

System Manager

Easy deployment via USB

Remote [Docker API]

Connect **a** Docker API "compliant" tool

Non-secured or secured connection

… -> Docker CLI used in the demo

Portainer: Add Container vs Add Stack

Multiple ways to achieve the same result

Adding new **container** instance

- Each parameter must be configured manually
  - A little longer when configuring networks
  - Create the persistent volume before (if needed)
- Containers (and settings) are not saved in the backup file
  - Persistent volumes are not backed up

Adding new container with a «**stack**»

- All parameters are loaded from the stack file
  - Stack file is actually a docker-compose
  - All container settings are loaded in a single shot
- Stacks are saved in the backup file
  - Easier to restore normal functioning
  - Persistent volumes are not backed up

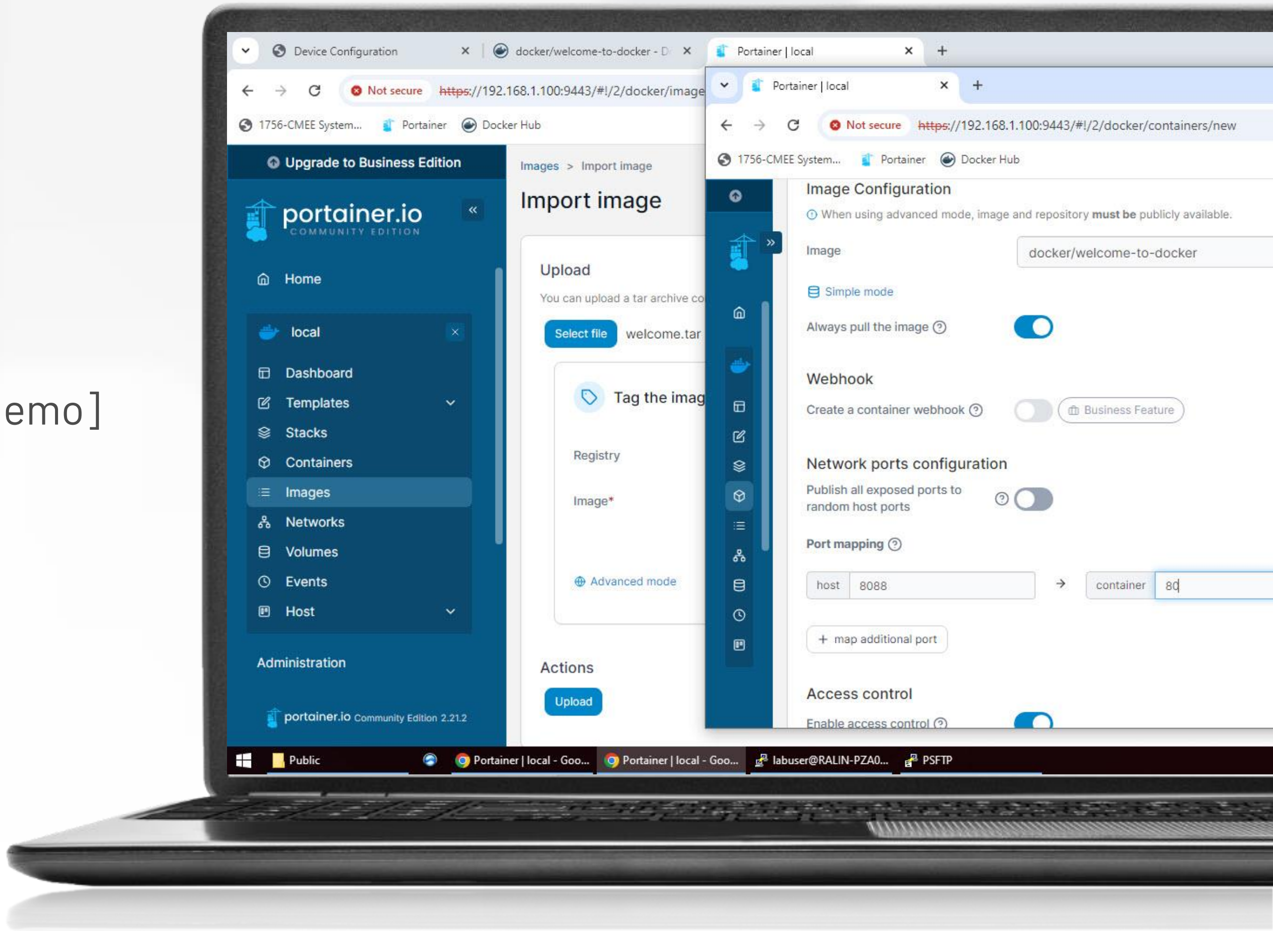
Manage with Portainer CE

Option **1a**: Add **Container**

- Acces Portainer web interface
  https://ip-address-of-the-module:9443

- Import image
  - Local -> Images -> Import
    -> select previously created image file [**welcome.tar** in this demo]

- Create Container
  - Local -> Containers -> Add Container
  - Switch to **Advanced mode**
  - Type image name [docker/welcome-to-docker in this demo]
  - Configure Port mapping [ex.: host **8088** / container **80**, **TCP**]



Upload
You can upload a tar archive co
welcome.tar

Image Configuration
When using advanced mode, image and repository **must be** publicly available.
Image: docker/welcome-to-docker
Always pull the image: on

Webhook
Create a container webhook: off (Business Feature)

Network ports configuration
Publish all exposed ports to random host ports: off
Port mapping
host 8088 → container 80

Manage with Portainer CE

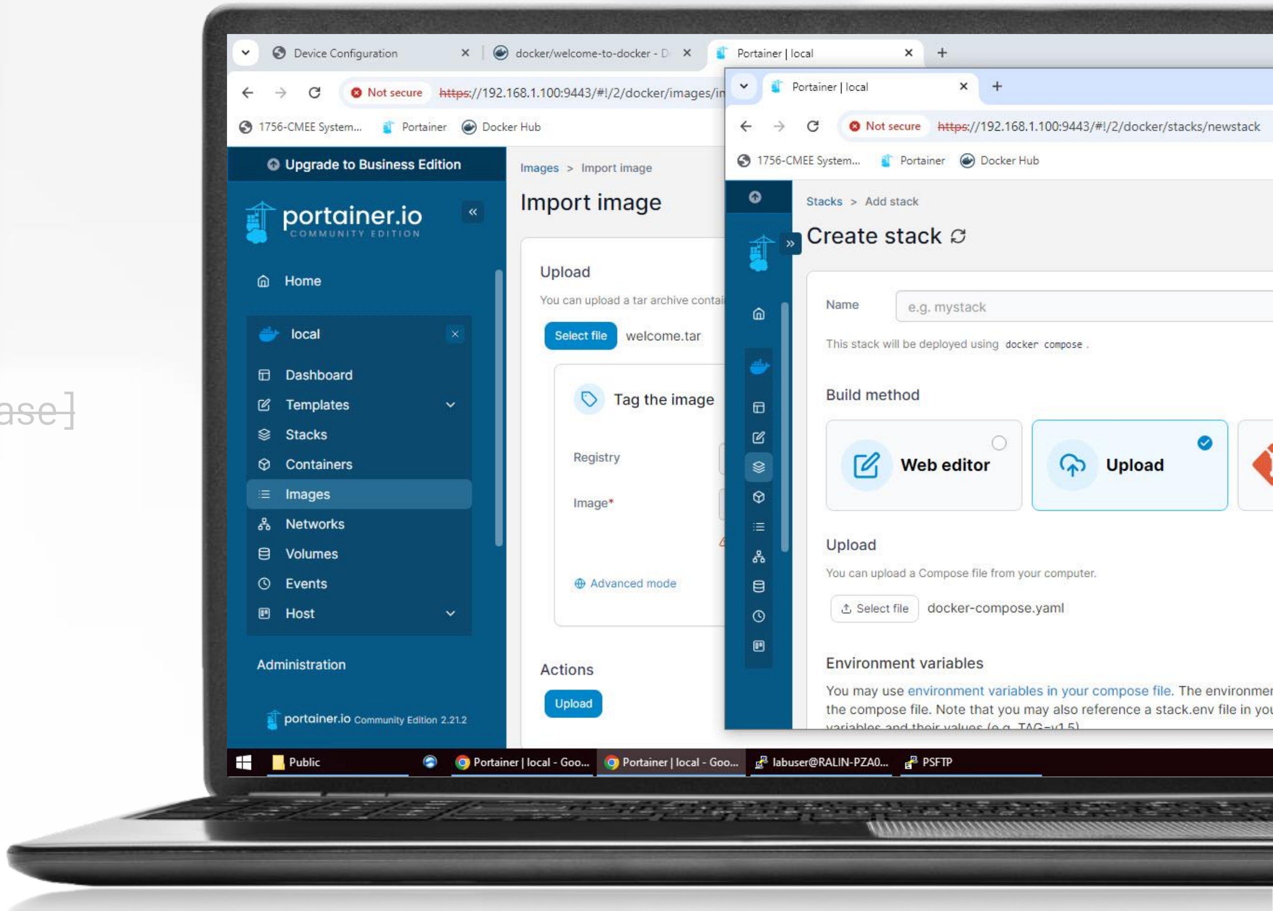
Option **1b**: Add **Stack**

- Acces Portainer web interface
  https://ip-address-of-the-module:9443

- ~~Import image~~ [already imported in previous step]
  - ~~Local -> Images -> Import~~
    ~~-> select previously created image file [**welcome.tar** in our case]~~

- Create Stack
  - Local -> Stacks -> Add Stack -> Upload
    -> select previously created **docker-compose.yaml** file

- Portainer allows much more [build, compose, ...]

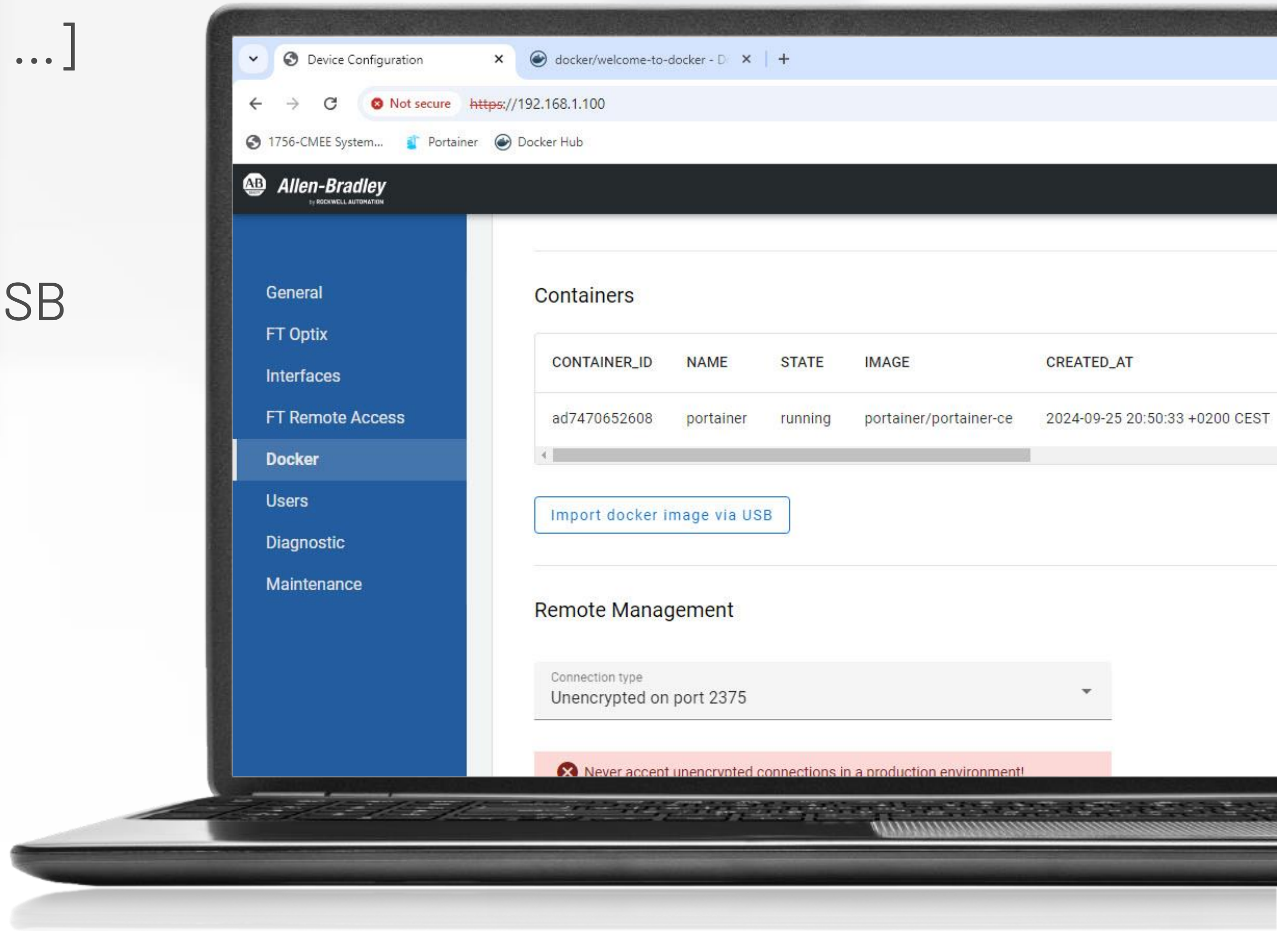

Device Configuration
docker/welcome-to-docker - D
Portainer | local
Not secure
https://192.168.1.100:9443/#!/2/docker/images/in
1756-CMEE System...
Portainer
Docker Hub
Upgrade to Business Edition
portainer.io COMMUNITY EDITION
Home
local
Dashboard
Templates
Stacks
Containers
Images
Networks
Volumes
Events
Host
Administration
portainer.io Community Edition 2.21.2
Images > Import image
Import image
Upload
You can upload a tar archive contai
Select file
welcome.tar
Tag the image
Registry
Image*
Advanced mode
Actions
Upload
Portainer | local
https://192.168.1.100:9443/#!/2/docker/stacks/newstack
Stacks > Add stack
Create stack
Name
e.g. mystack
This stack will be deployed using docker compose .
Build method
Web editor
Upload
Upload
You can upload a Compose file from your computer.
Select file
docker-compose.yaml
Environment variables
You may use environment variables in your compose file. The environme
the compose file. Note that you may also reference a stack.env file in you
Public
Portainer | local - Goo...
Portainer | local - Goo...
labuser@RALIN-PZA0...
PSFTP

“Manage” with SystemManager

Option **2**: Easy way to deploy a container [but nothing else]

- **Import image only** [no stop, start, update, remove ...]

- Prepare a USB partitioned FAT32, exFAT or ext4
- Copy **<image>.tar** and **docker-compose.yaml** to USB
- Plug USB to 1756-CMEE
- Access the SystemManager page of the device
- Import docker image via USB



Containers

CONTAINER_ID	NAME	STATE	IMAGE	CREATED_AT
ad7470652608	portainer	running	portainer/portainer-ce	2024-09-25 20:50:33 +0200 CEST

Remote Management

Connection type: Unencrypted on port 2375

Never accept unencrypted connections in a production environment!

Manage Remotely -> example with Docker CLI

Option **3**: Endless options with CLI for different platforms

- Once the image is configured and built, it makes sense to switch the CLI to 1756-CMEE
- Connect to Docker daemon on 1756-CMEE:
  - `export DOCKER_HOST=192.168.1.100:2375` [Linux]
  - `set DOCKER_HOST=192.168.1.100:2375` [Windows Command shell]
  - `$env:DOCKER_HOST="192.168.1.100:2375“` [Windows PowerShell]
- Transfer image to 1756-CMEE:

```
$ docker load --input ./welcome.tar
```

- Create a container based on the image and start it:

```
$ docker run -dp 8088:80 --restart unless-stopped --name welcome docker/welcome-to-docker
```

- Disconnect from 1756-CMEE:
  - `unset DOCKER_HOST` [Linux]
  - `set DOCKER_HOST=` [Windows Command shell]
  - `Remove-Item env:DOCKER_HOST` [Windows PowerShell]

Useful commands for Docker CLI

Some basic commands

- `docker help`

- `docker ps` (lists running containers)
- `docker ps -a` (lists all existing containers)
- `docker images` (lists existing images)
- `docker load --input <filename>` (loads image from TAR)
- `docker stop <container>`
- `docker rm <container>` (removes container)
- `docker rmi <image>` (removes image)
- `docker update` (change properties of an existing container)

Prepare a Docker **image** [containerize a **custom** app]

Rockwell Automation

Containerize an application [slide 1 of 2]

Create a custom container

- Prepare a Linux machine (see additional slide)
- Follow https://docs.docker.com/get-started/workshop/02_our_app/, with some changes to make it work on 1756-CMEE:
  - **DockerFile** contents:

    ```
    # syntax=docker/dockerfile:1
    FROM --platform=linux/arm64 node:alpine
    WORKDIR /app
    COPY . .
    RUN yarn install --production
    CMD ["node", "src/index.js"]
    EXPOSE 3000
    ```

  - Build command:

    ```
    $ docker build --platform linux/arm64 -t getting-started.
    ```

- **Leave** web guide with last paragraph before Start an app container

Containerize an application [slide **2** of 2]

Create a custom container

- Export and save the image:

```
$ docker save > getting-started.tar getting-started
```

- Connect to Docker daemon on 1756-CMEE (see one of previous slides)

```
export DOCKER_HOST=192.168.1.100:2375
```

- Transfer image to 1756-CMEE:

```
$ docker load --input ./getting-started.tar
```

- Create a container based on the image and start it :

```
$ docker run -dp 3000:3000 --restart unless-stopped --name my-name-gs getting-started
```

- [Test the application by accessing https://192.168.1.100:3000]

Docker “tools”
[Docker Engine/Desktop]

Docker Engine

Command Line Interface

- Linux
  - use package manager for specific distribution (typically something like `apt install docker`)
  - https://docs.docker.com/engine/install/ [detailed guide in the following (hidden) slide]

- Windows
  - manually unzip client and server executables
  - https://docs.docker.com/engine/install/binaries/#install-server-and-client-binaries-on-windows
  - **if** getting 1607 error durickg dockerd service start / **failed to load vmcompute.dll** in Event Viewer, look at https://poweruser.blog/docker-on-windows-10-without-hyper-v-a529897ed1cc -> basically enable **Containers**
  - does **not allow build of Linux images** -> but still can be used to manage Docker on Embedded Edge Compute via CLI
    - [load, stop, start, update, remove, ...]

Docker Desktop

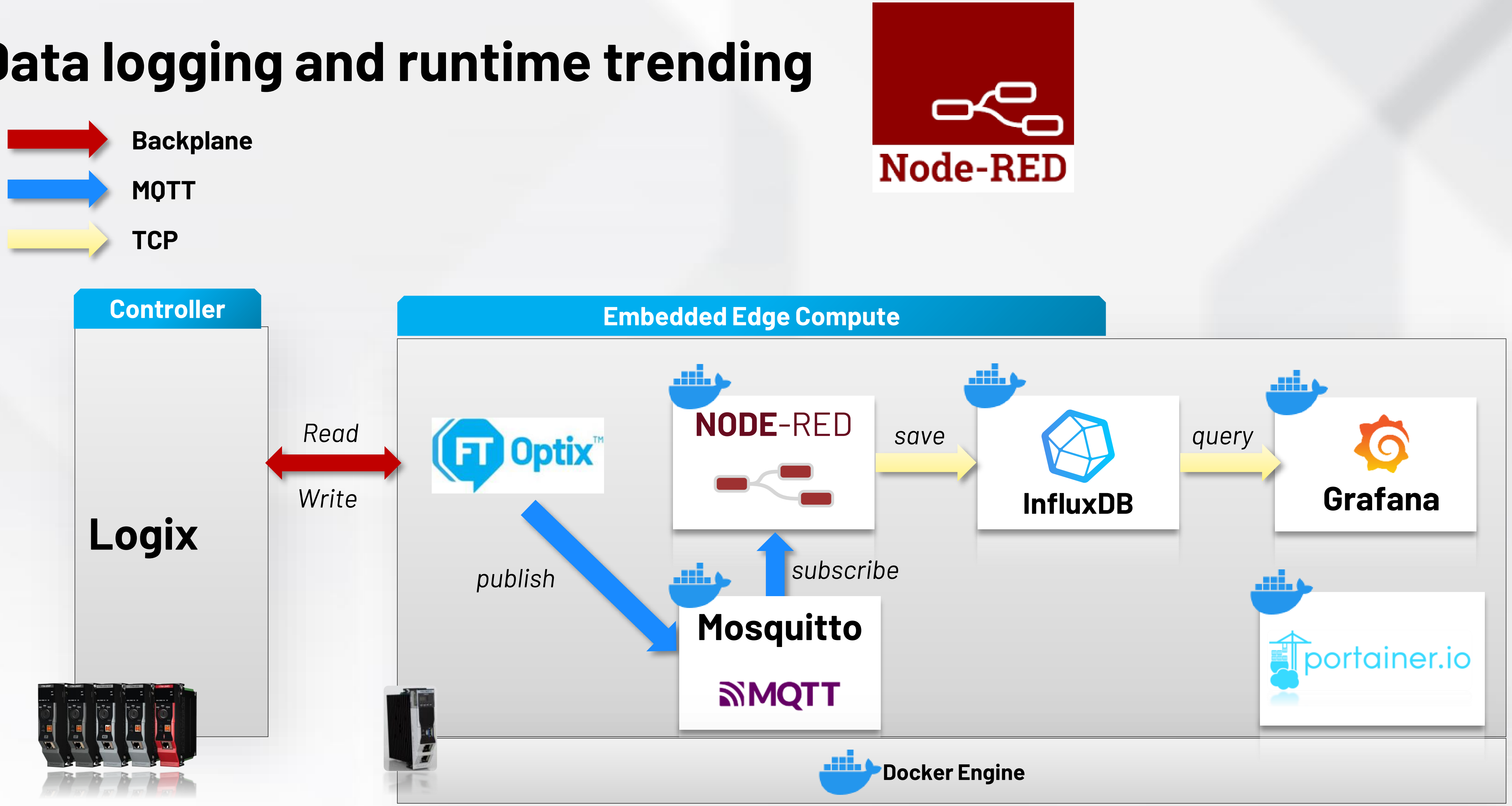
Grpahical user interface, plus samples, plus command line interface

- Needs license for business use
  - [**RA Internal**]: https://rockwellautomation.sharepoint.com/sites/BusinessAssetManagement/SitePages/Software-License-Pricing-Estimate1.aspx
- https://docs.docker.com/engine/install/

Rockwell Automation

Use cases

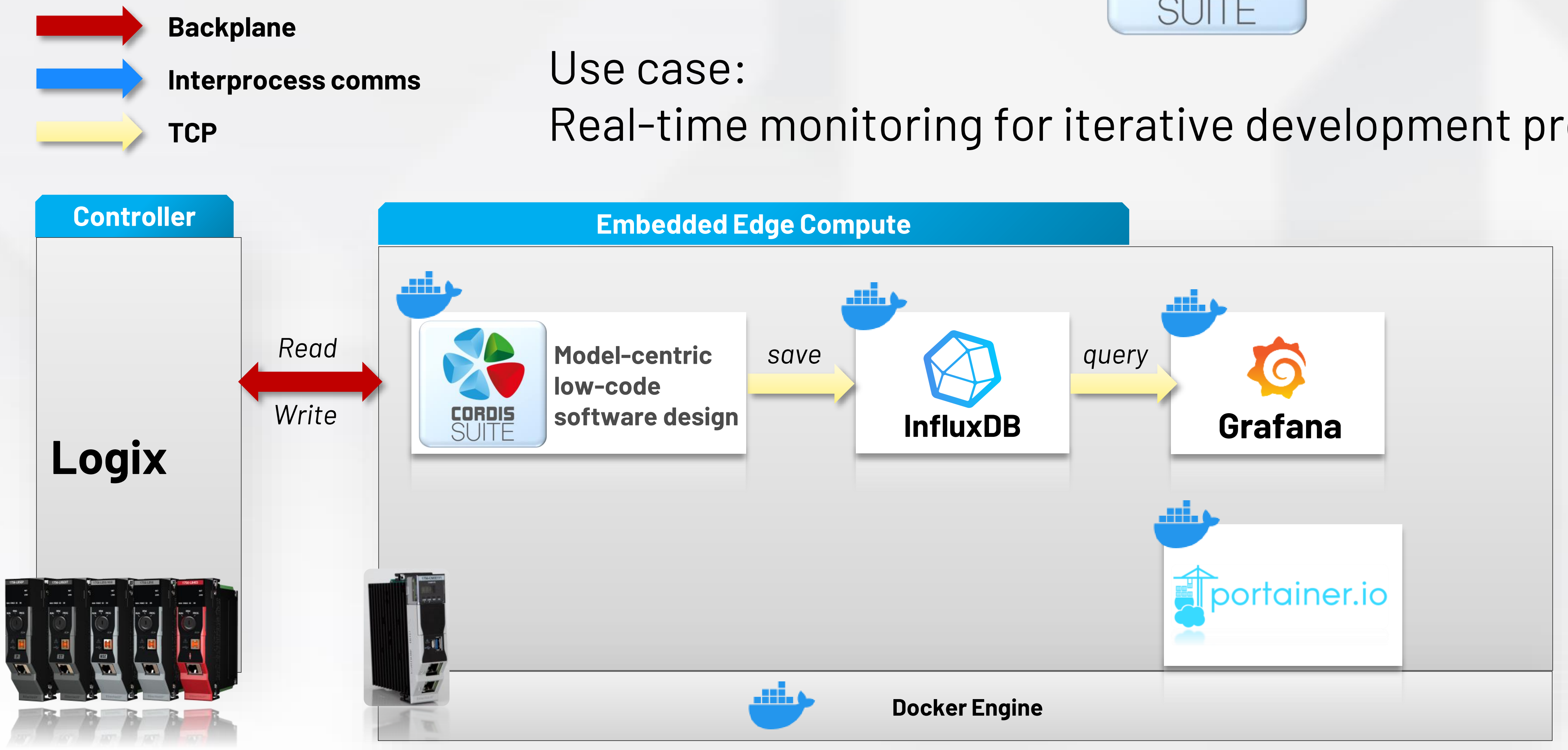
Data logging and runtime trending



Lab under construction for Automation Fair® 2025

Low-Code Software Design & Datalogging

Use case:
Real-time monitoring for iterative development process



Questions?

Rockwell Automation

expanding **human possibility**®

www.rockwellautomation.com