Rockwell Automation

Usage of Docker and Portainer on the 1756-CMEE

Luca Bennati • 20•08•24

expanding **human possibility**®

Workshop goals

Rockwell Automation

expanding **human possibility**®

PUBLIC

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 standalone, enabling Portainer agent, deployment of a container.

Build a custom container

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

Prerequirements

Rockwell Automation

expanding **human possibility**®

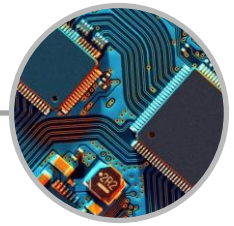
PUBLIC

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

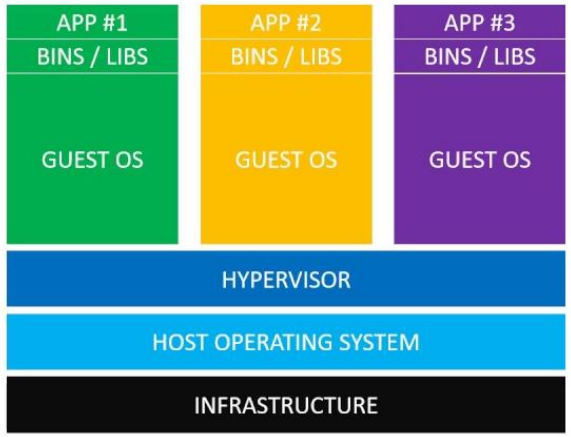
Linux Yocto with RA SystemManager, support for Docker engine.

Containerization

How to pack a lot of things into a carry-on bag

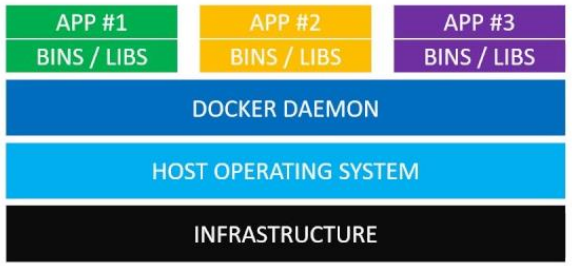
Virtual Machine

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



Container

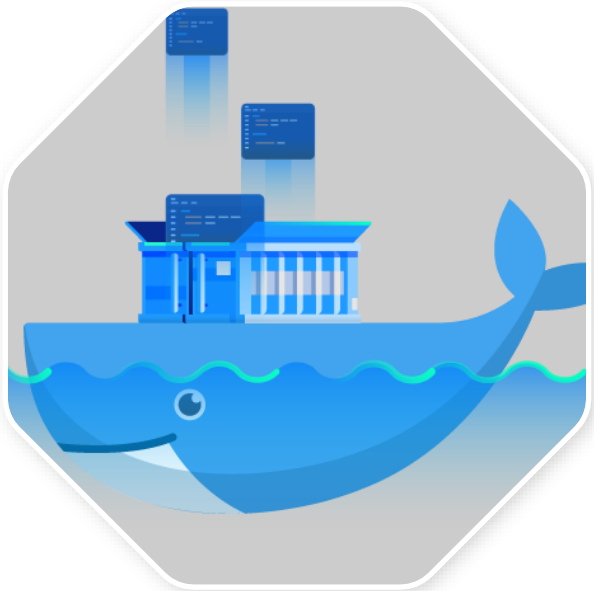
- Needs only the resources used by the executable
- Can run on any host os with a container agent
- Weighs few KB (or even just as a single text file)
- Only contains the main executable and dependencies



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 tools with advanced features like load balancing and addons

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

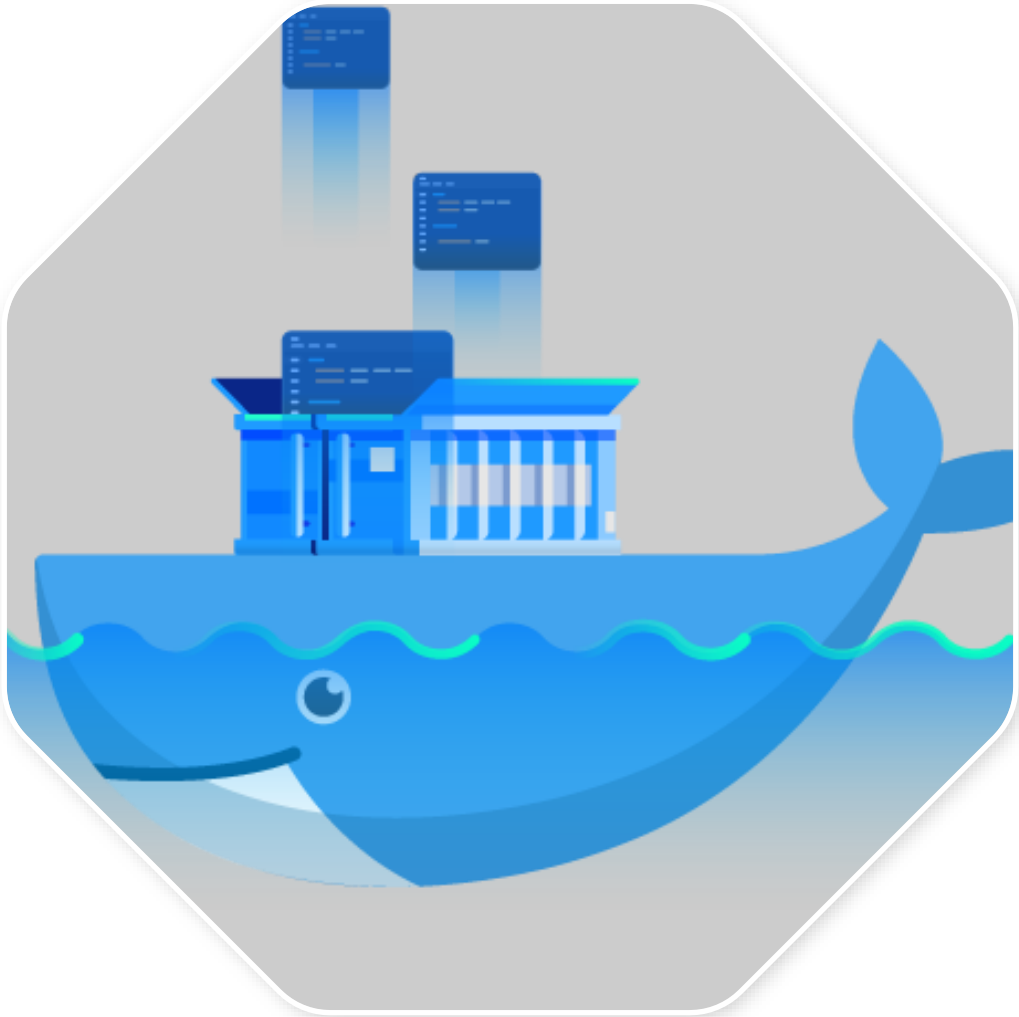
Why Docker?

Good advertisment and customer care

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

Big community of both professionist and amateurs which creates a strong engagement

Constantly updated and supported



Simplicity and good documentation

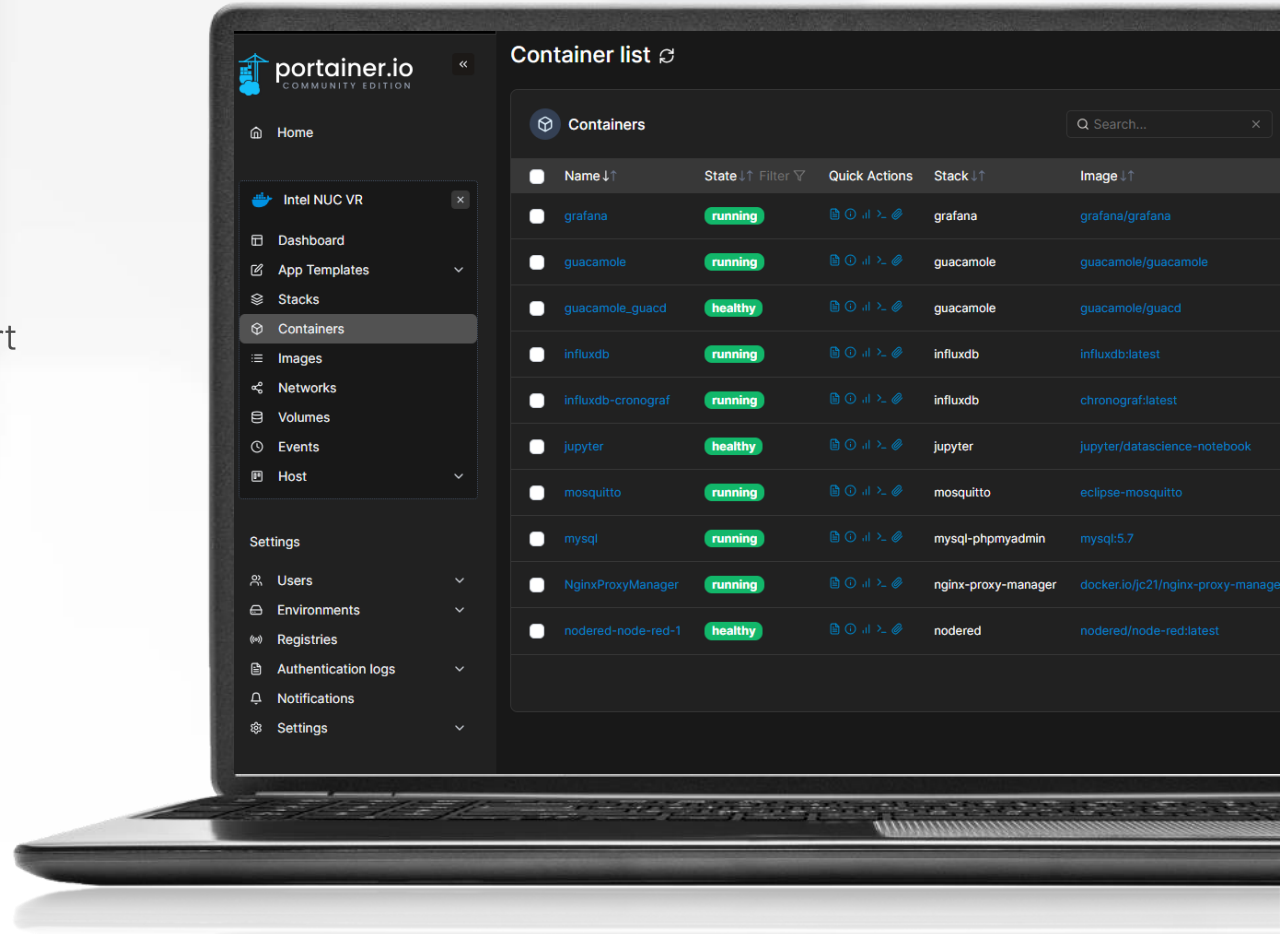
Containers can be easily ported across different systems and host OS

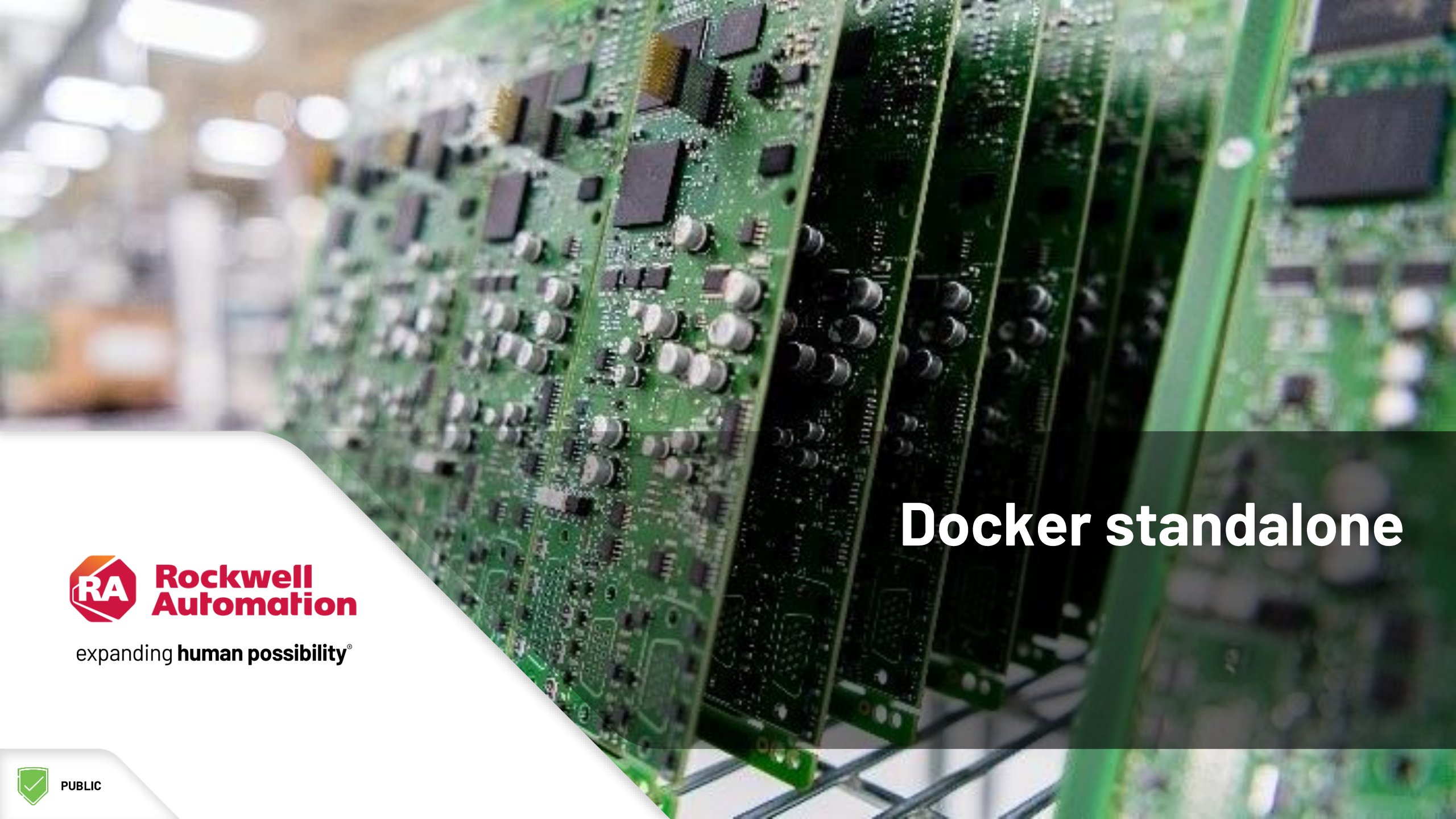
Support for scalability

Portainer

Portainer is actually a Docker container

- Portainer is not a containerization platform
- Portainer is a GUI for Docker
- Comes in two flavours:
  - 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 Dockerfiles and Docker compose





Docker standalone

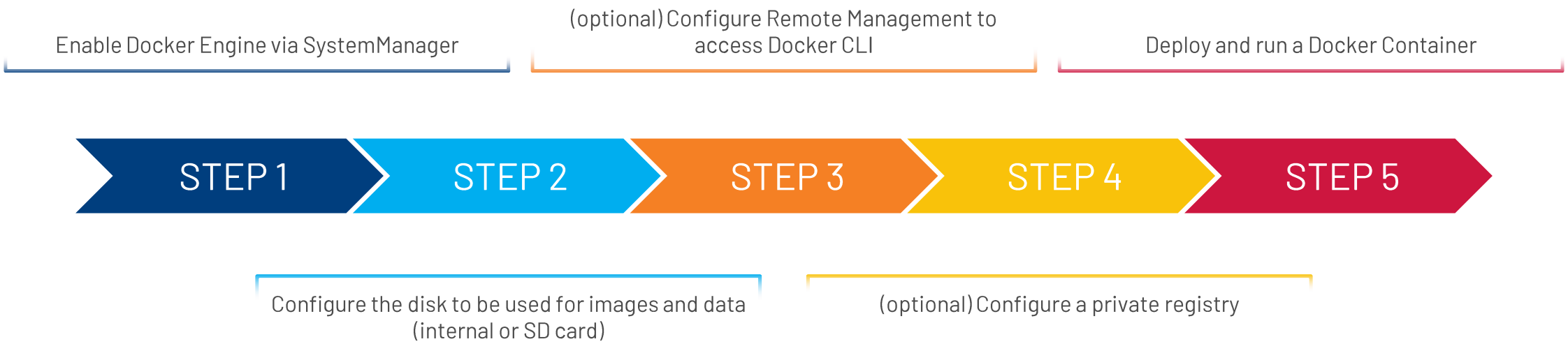
Rockwell Automation

expanding **human possibility**®

PUBLIC

Enabling Docker

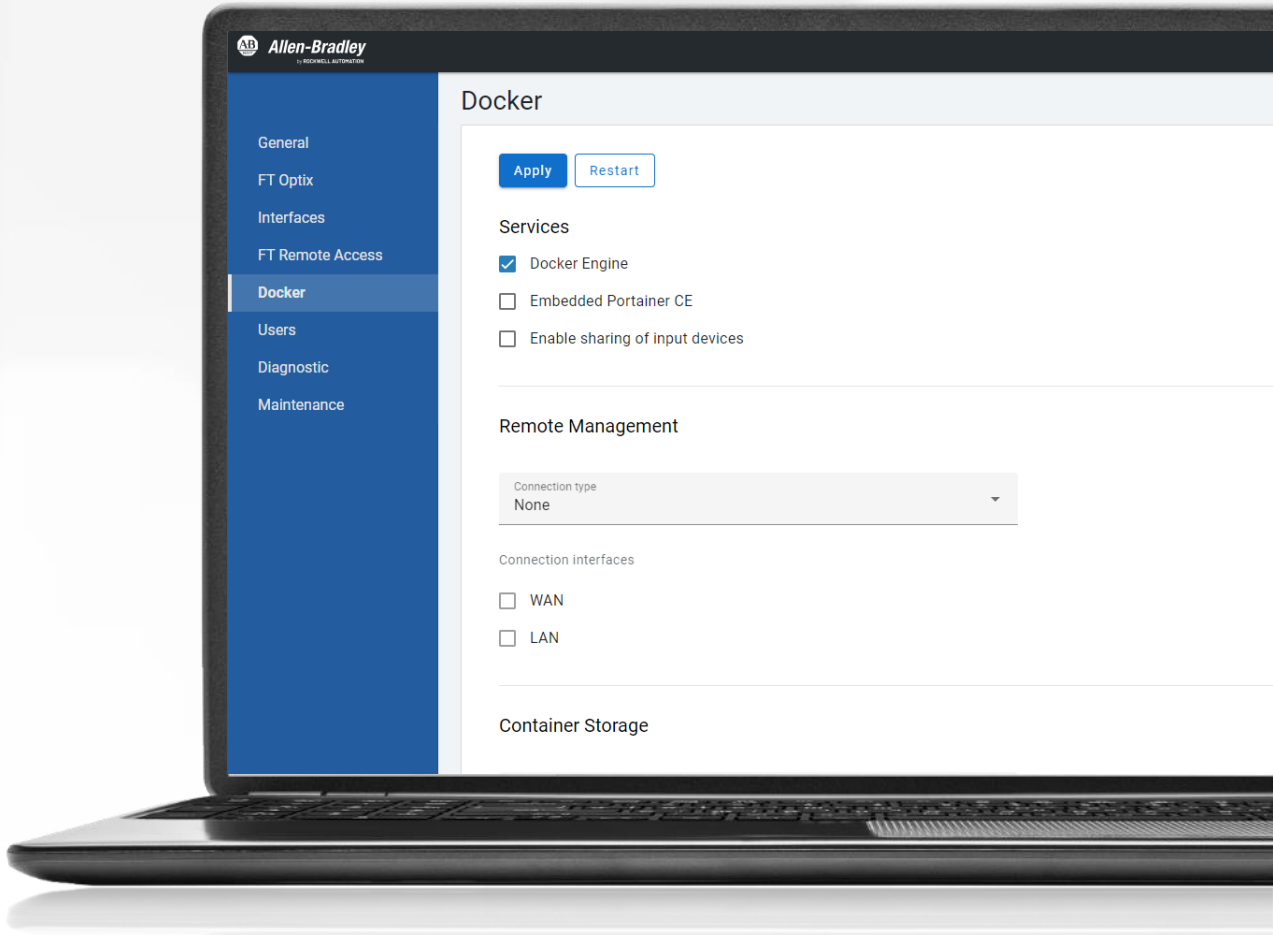
Docker is not activated by default



Enable Docker from SystemManager

Tick the option and reboot

- Access the SystemManager page of the device
- Enable the Docker Engine
- Configure the Remote Management
  - Exposes the Docker CLI to the network
- Configure the Container Storage
  - Where to store images and persistent data
- Configure Private Registries
  - Where to pull images if not listed on hub.docker.com
- Configure Proxy
  - Some networks require this parameter



Loading a Docker container with System Manager

Rockwell Automation

expanding **human possibility**®

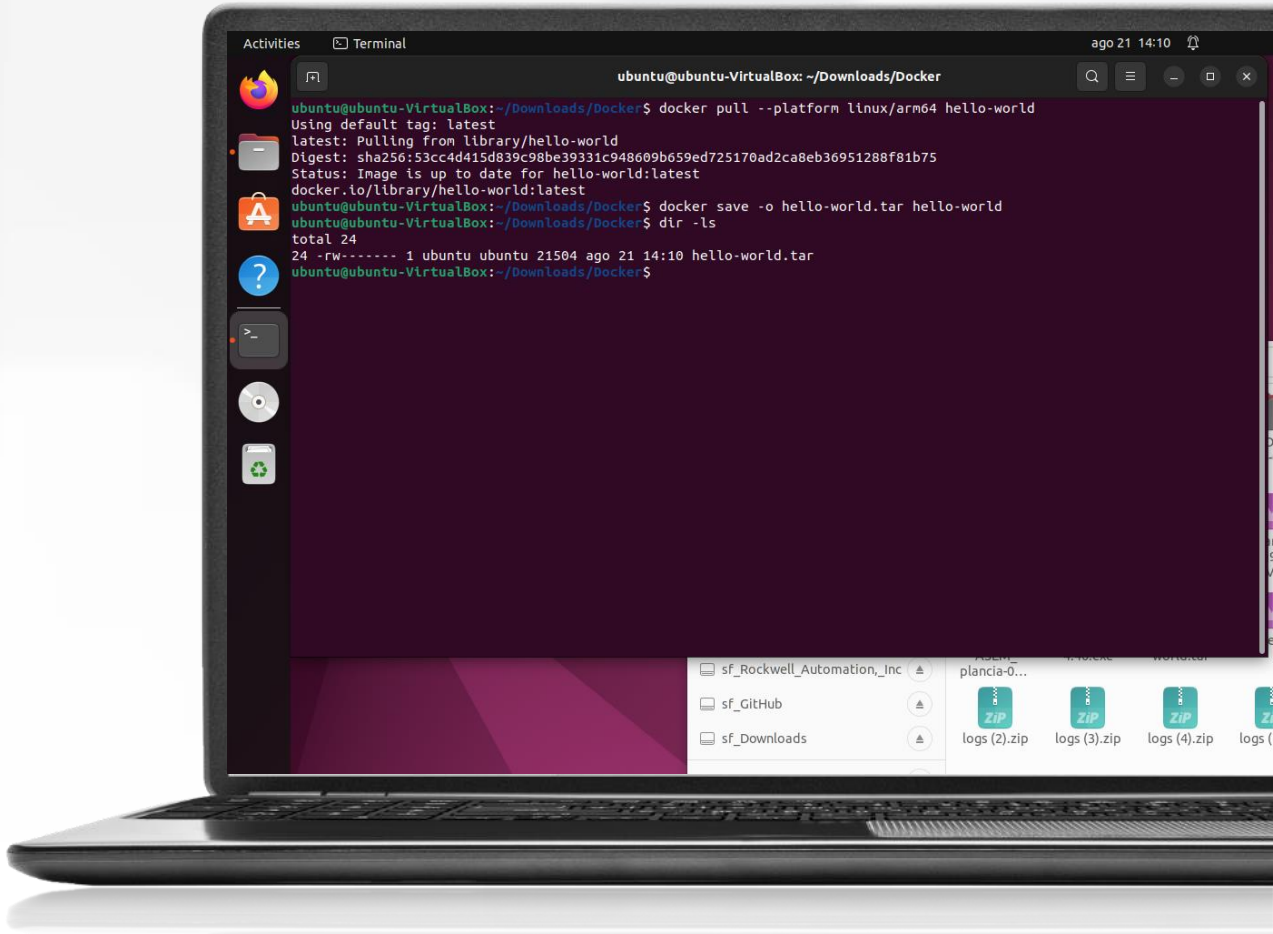
PUBLIC

Loading containers from USB

Load a docker-compose.yaml and the relevant .tar image

- Pull the image from any registry
- Make sure to specify «linux/arm64» as target
- Export image to tar file
- Copy to USB
- Do not compress it!

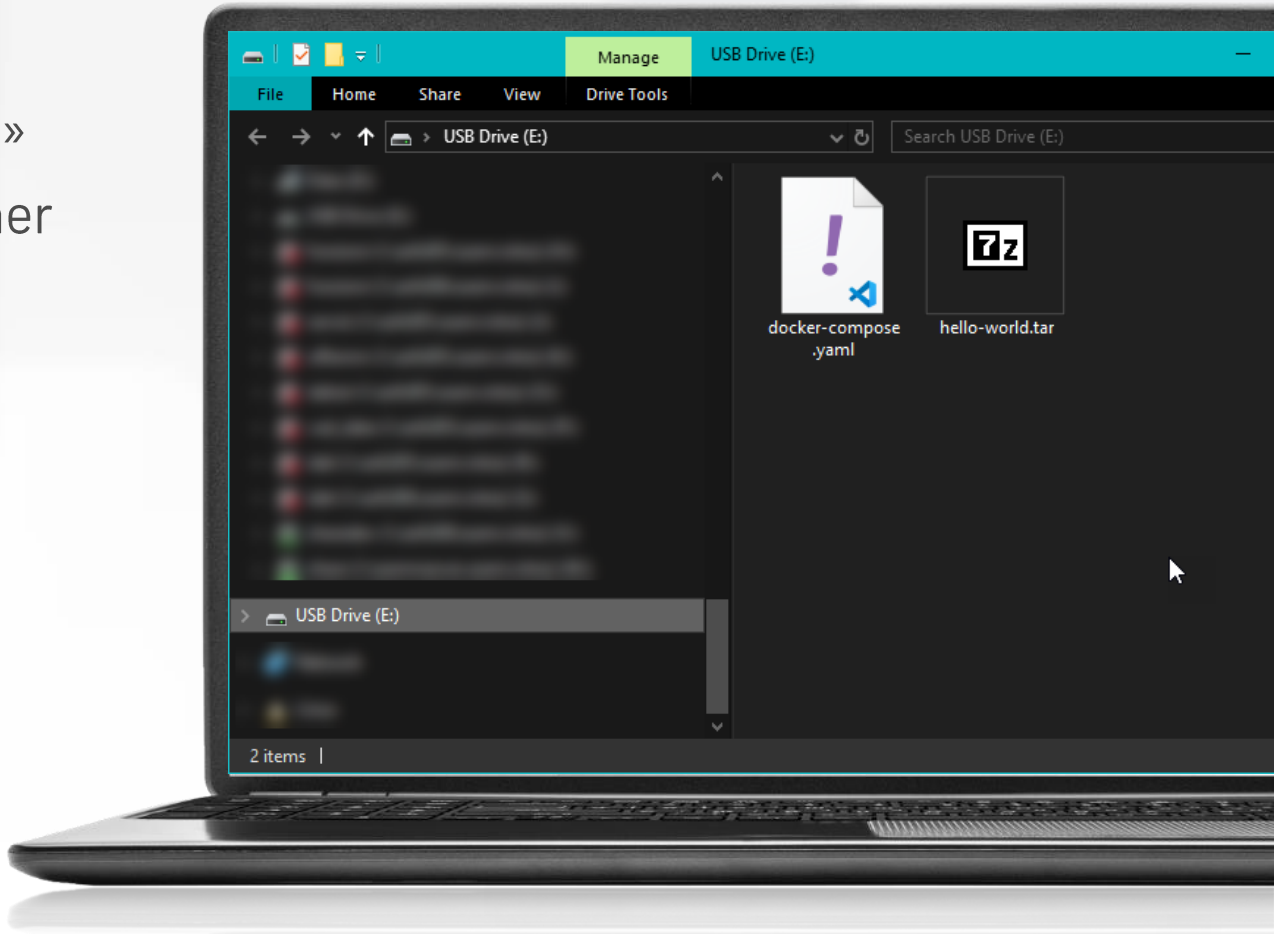
```
$ docker pull --platform linux/arm64 hello-world
$ docker save -o hello-world.tar hello-world
```



Loading containers from USB

Load a docker-compose.yaml and the relevant .tar image

- USB partition must be FAT32, exFAT or ext4
- The image «tar» file comes from the «docker save»
- A docker-compose.yaml file describes the container startup procedure and arguments



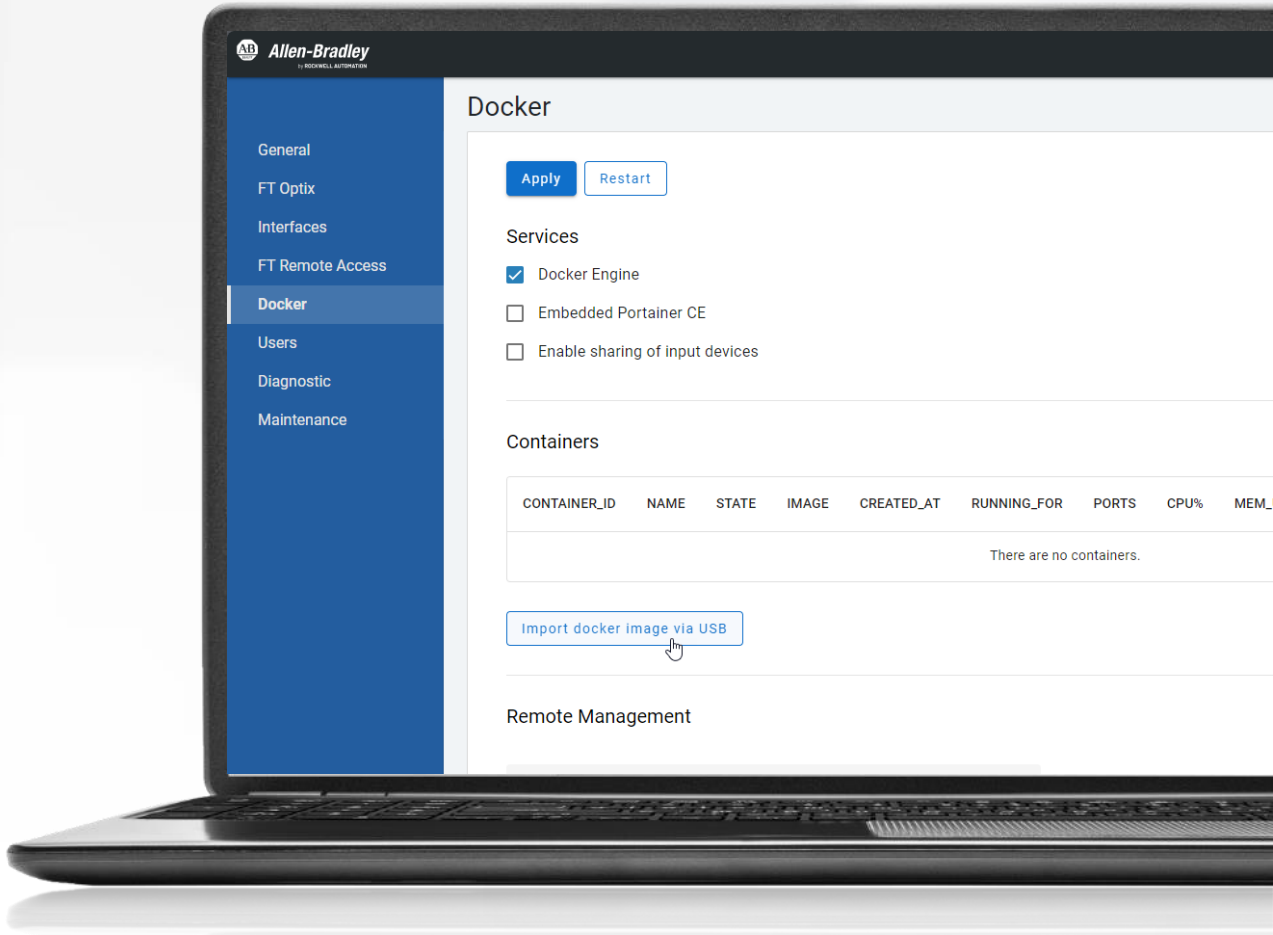

```
#file: docker-compose.yaml
version: '3.8'

services:
  hello-world:
    image: hello-world
```

Loading containers from USB

Load a docker-compose.yaml and the relevant .tar image

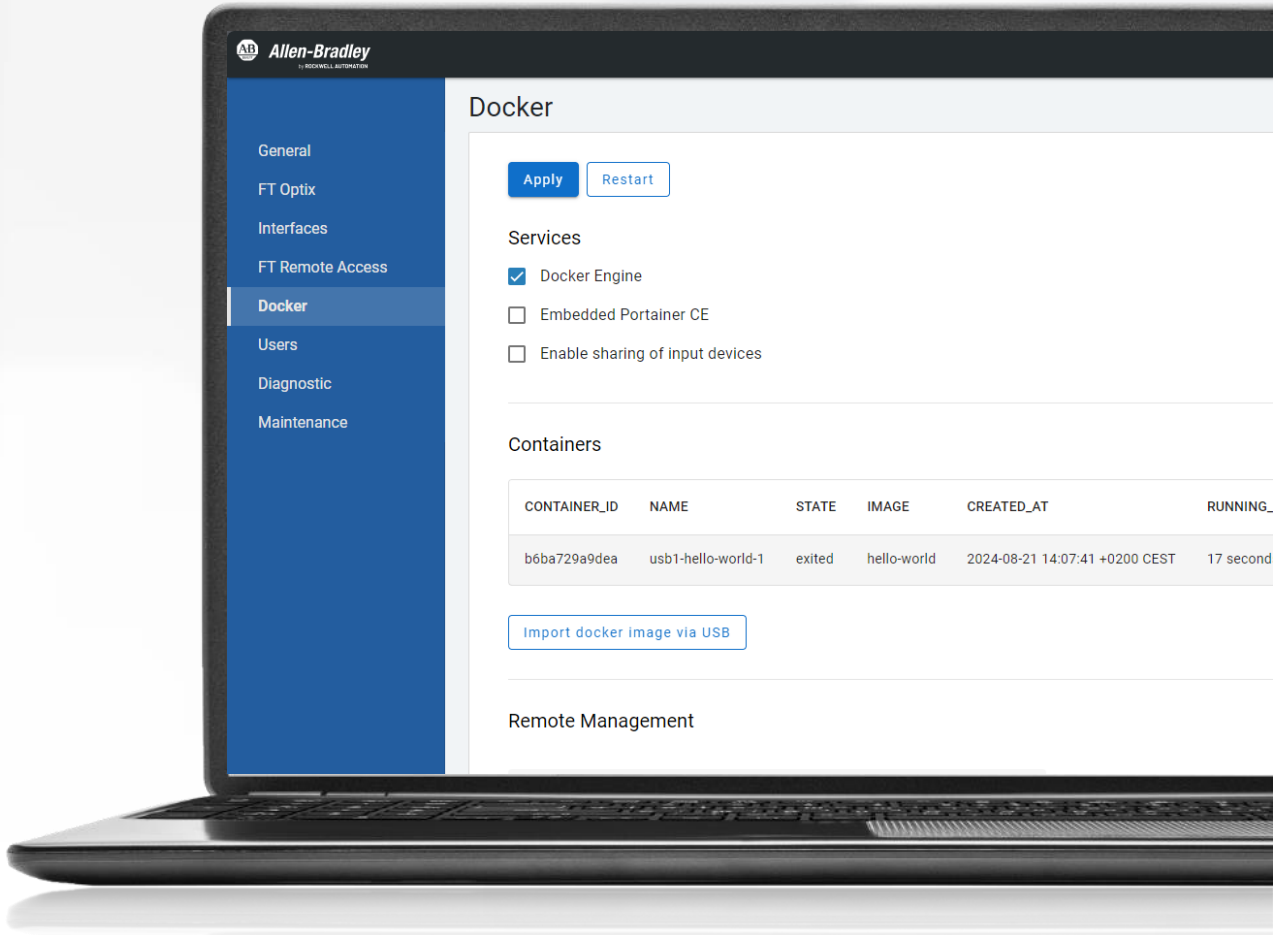
- Plug the USB key
- Import the container



Loading containers from USB

Load a docker-compose.yaml and the relevant .tar image

- Check the status



Example of nginx server

Services can be exposed by forwarding a TCP port

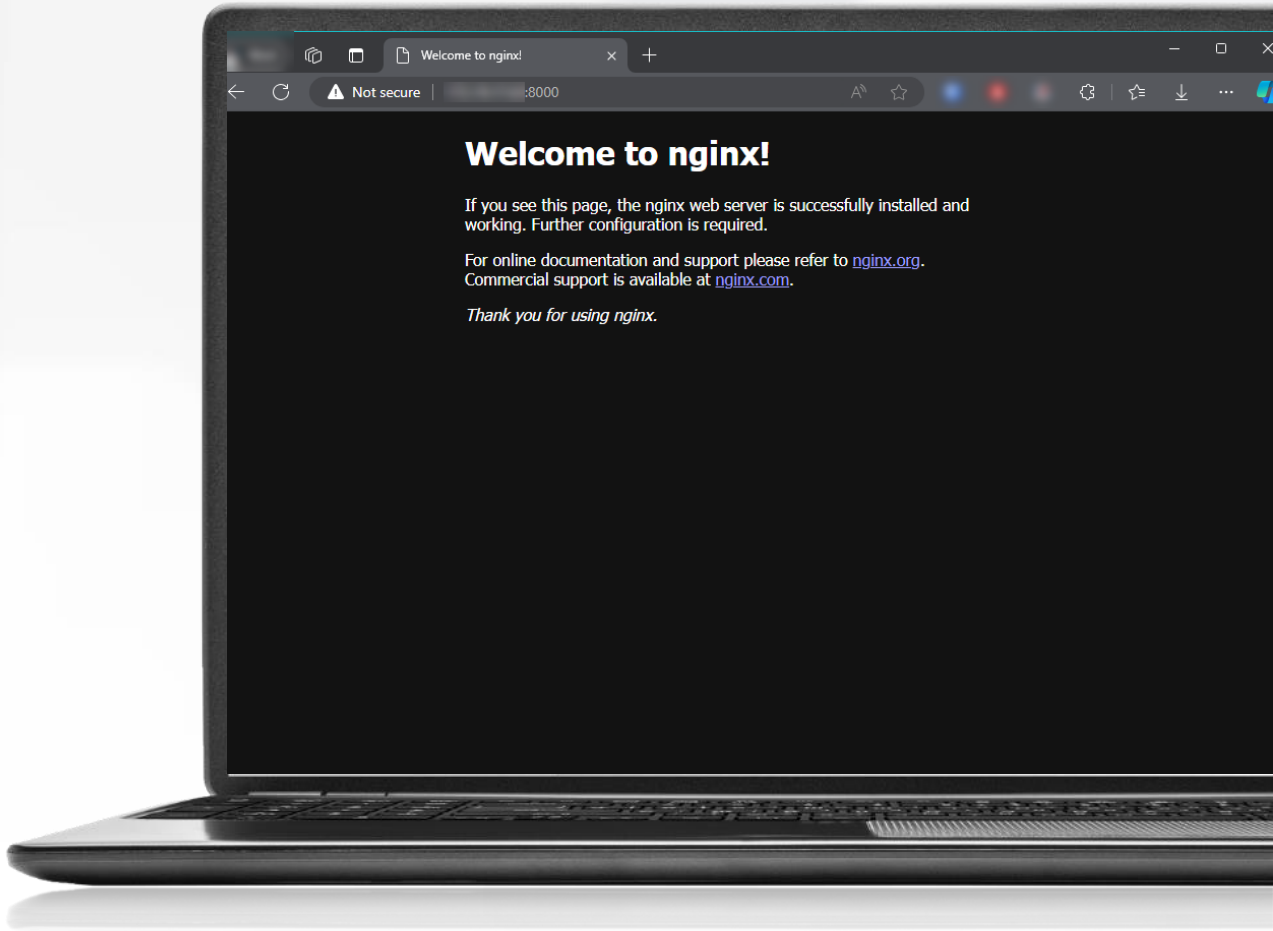
- Pull the image

```
$ docker pull --platform linux/arm64 nginx
$ docker save -o nginx.tar nginx
```

- Create the «docker-compose.yaml»
  - Network is set to «NAT» by default
  - Public address is the same as the 1756-CMEE module
  - Ports are forwarded in the stack file

```
version: '3.8'

services:
  web-server:
    image: nginx
    ports:
      - "8000:80"
```



Docker and Portainer

Rockwell Automation

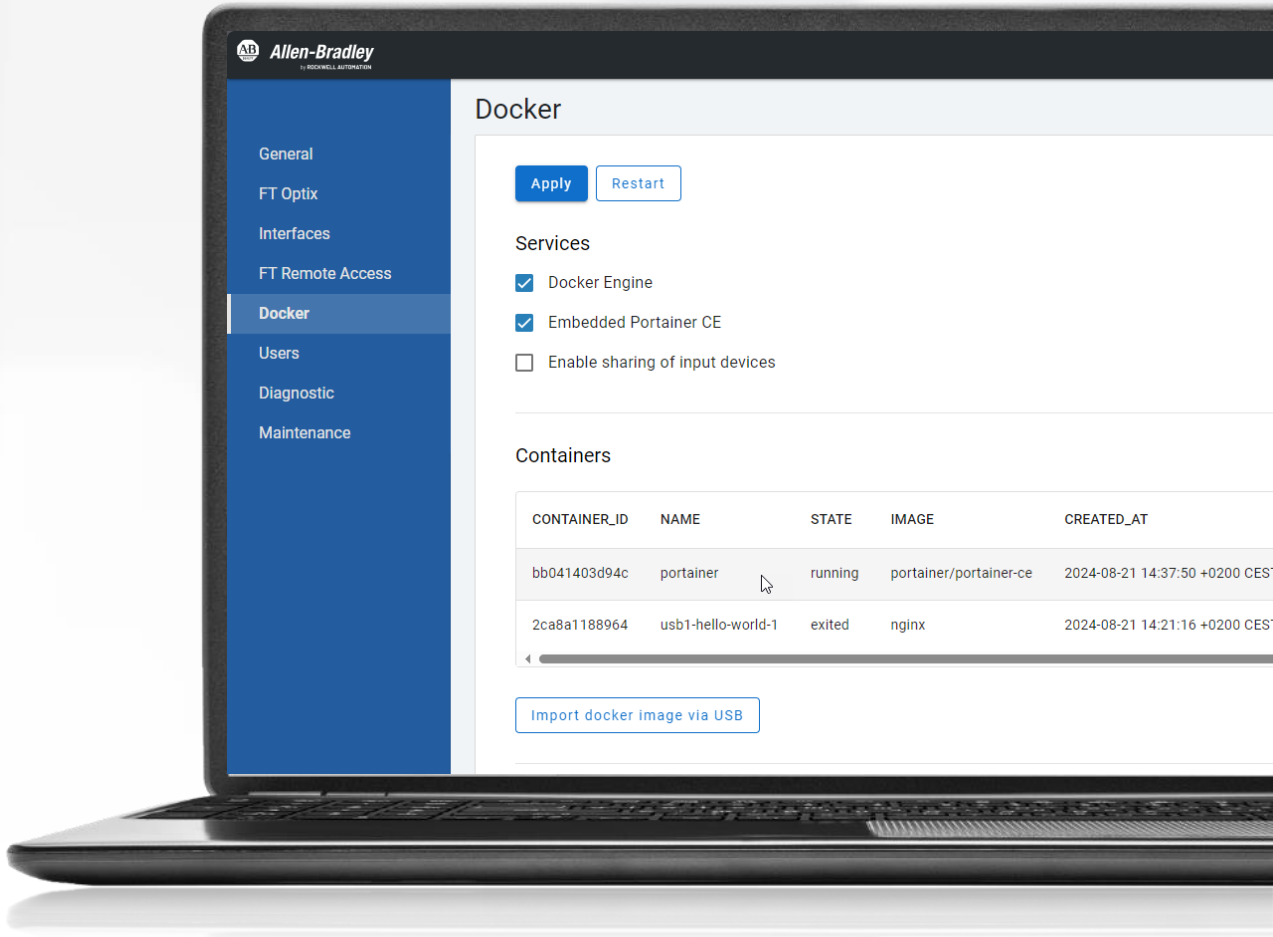
expanding **human possibility**®

PUBLIC

Enable Portainer from SystemManager

Tick the option and access the page

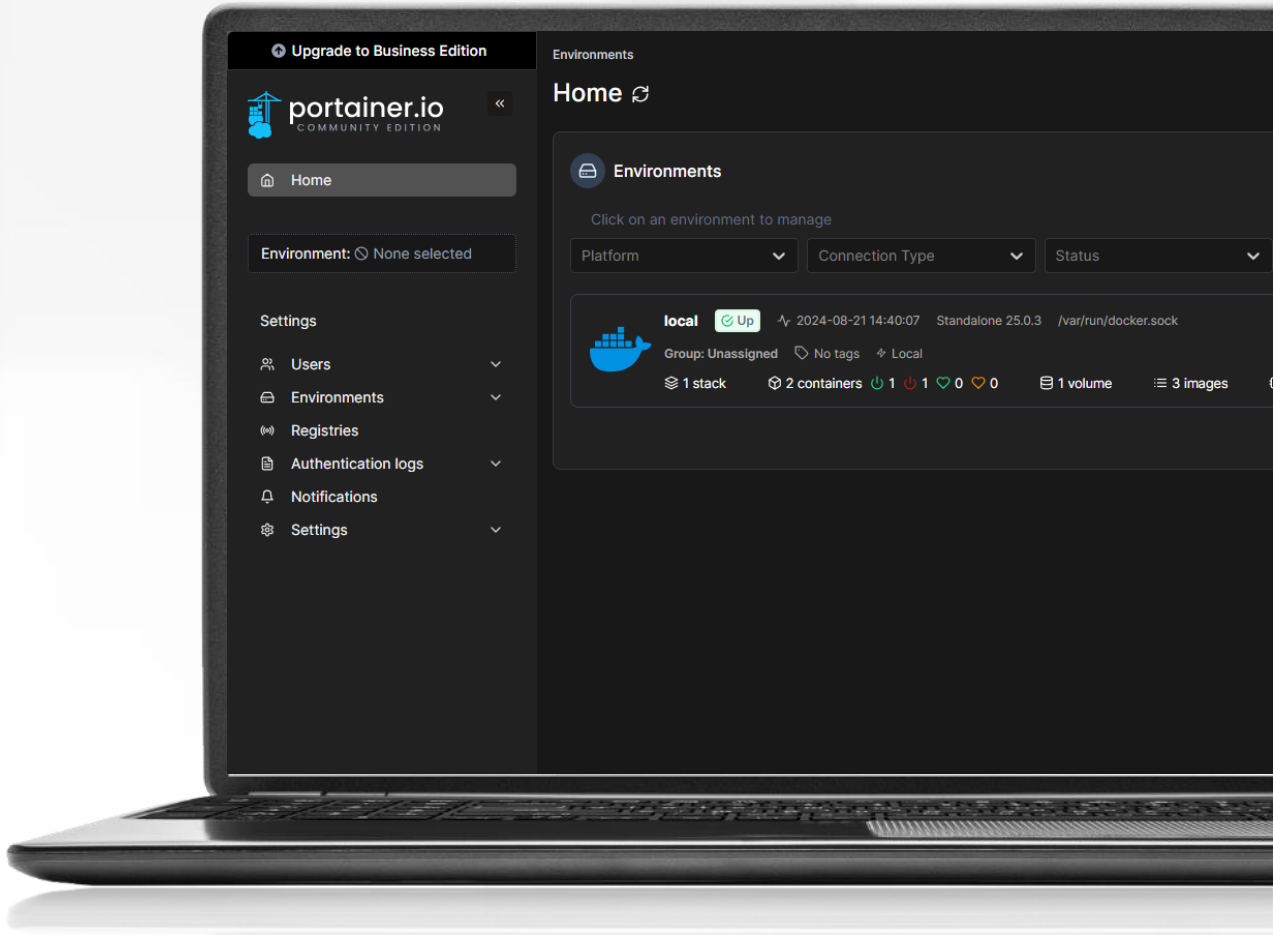
- Enable Portainer CE
- Reboot the device



Enable Portainer from SystemManager

Tick the option and access the page

- Access Portainer interface
  - https://ipaddress:9443
- Trust the self-signed certificate
- Configure admin password
  - Password complexity can be reduced later in user settings
- The «local» environment comes preconfigured



Starting containers with Portainer

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

Loading a Docker container with Portainer

Rockwell Automation

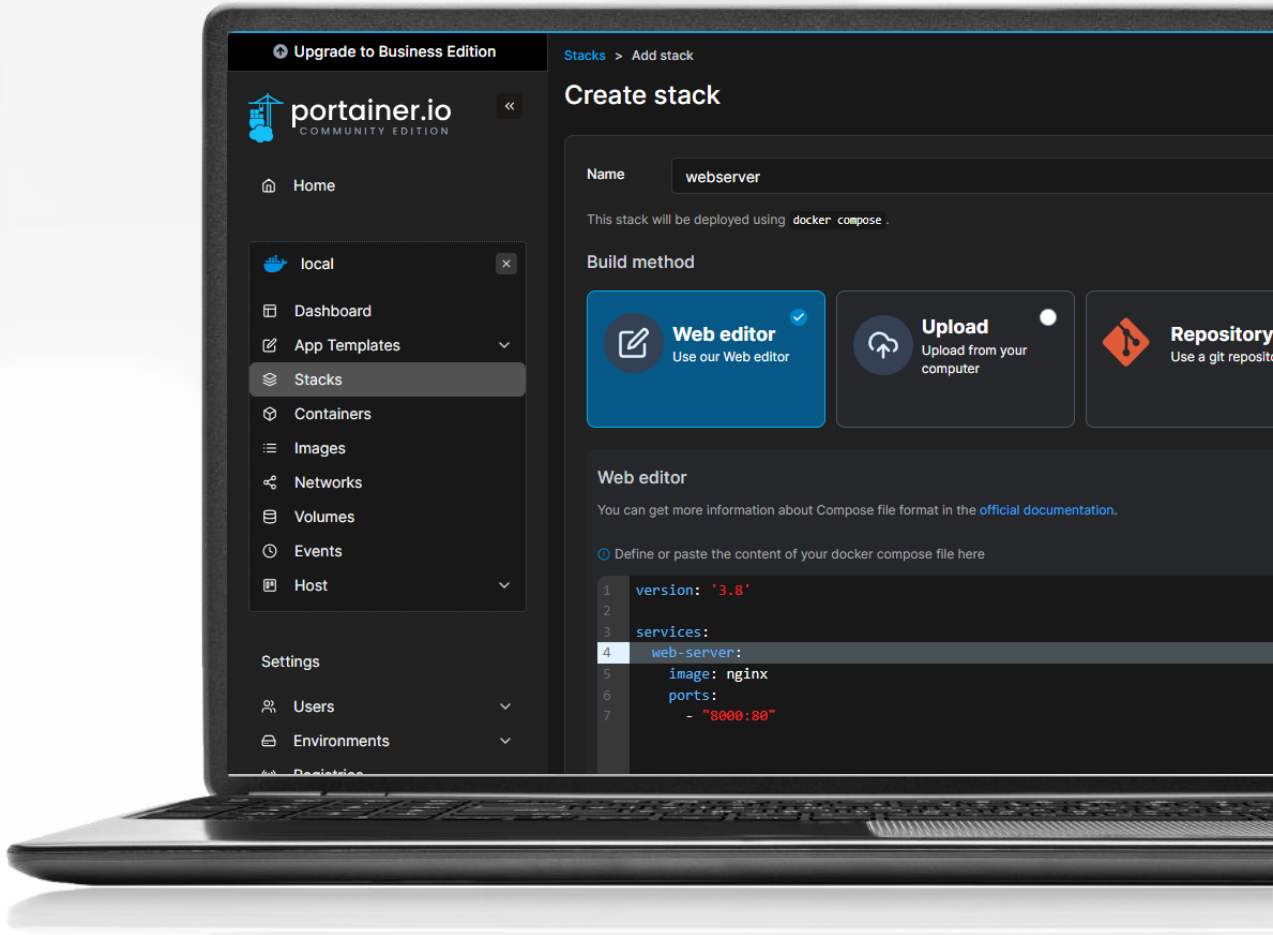
expanding **human possibility**®

PUBLIC

Creating a new stack

Stacks are just Docker compose files

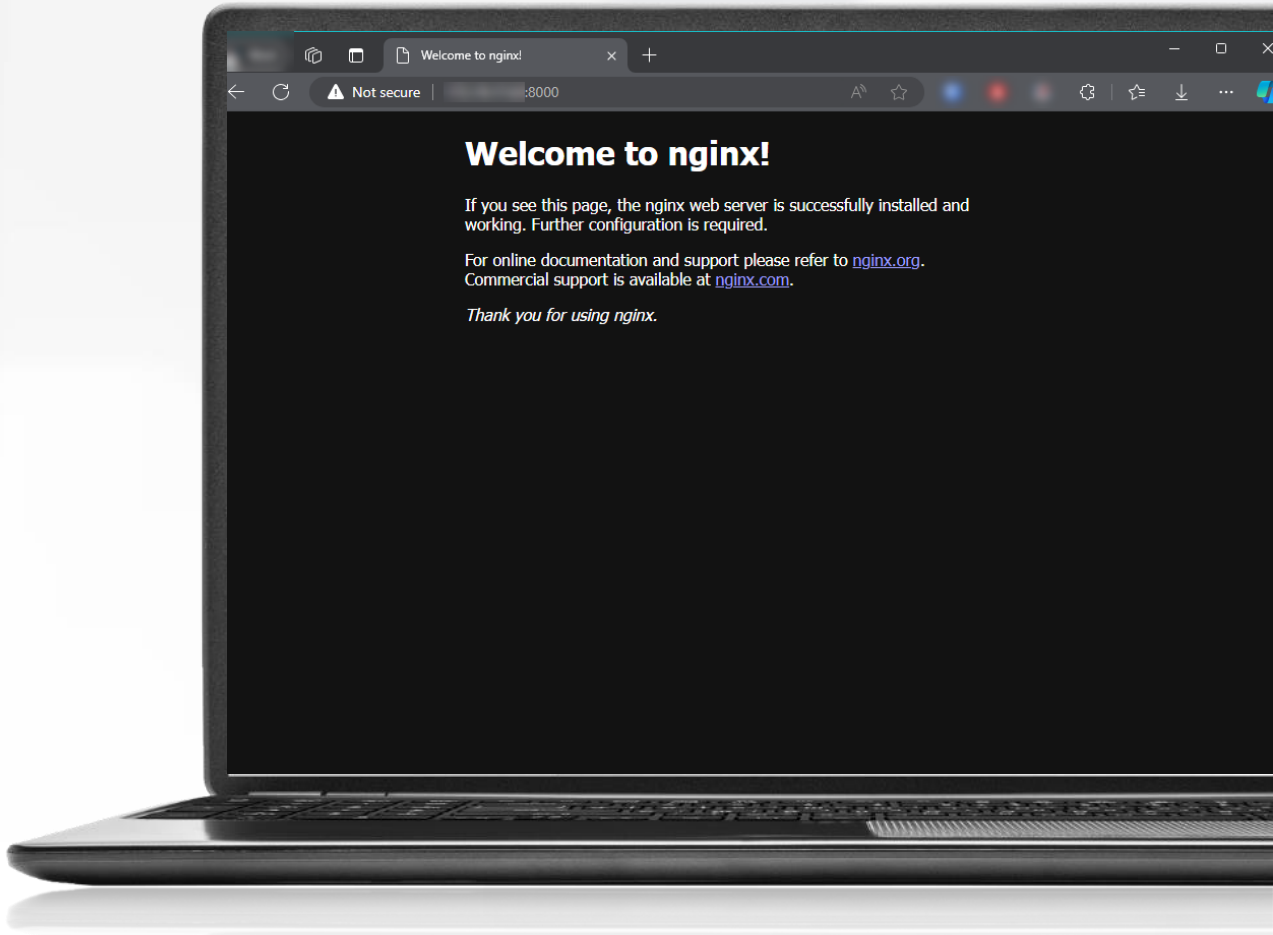
- Select the «local» environment
- Move to the «stacks» section
- Click «Add stack» in the up-right corner
  - Provide a stack name
  - Write the stack details
- Scroll down and deploy the stack



Example of nginx server

Same as before but started with Portainer

- Open the web server page
  - Network is set to «NAT» by default
  - Public address is the same as the 1756-CMEE module
  - Ports are forwarded in the stack file



Docker CLI

Rockwell Automation

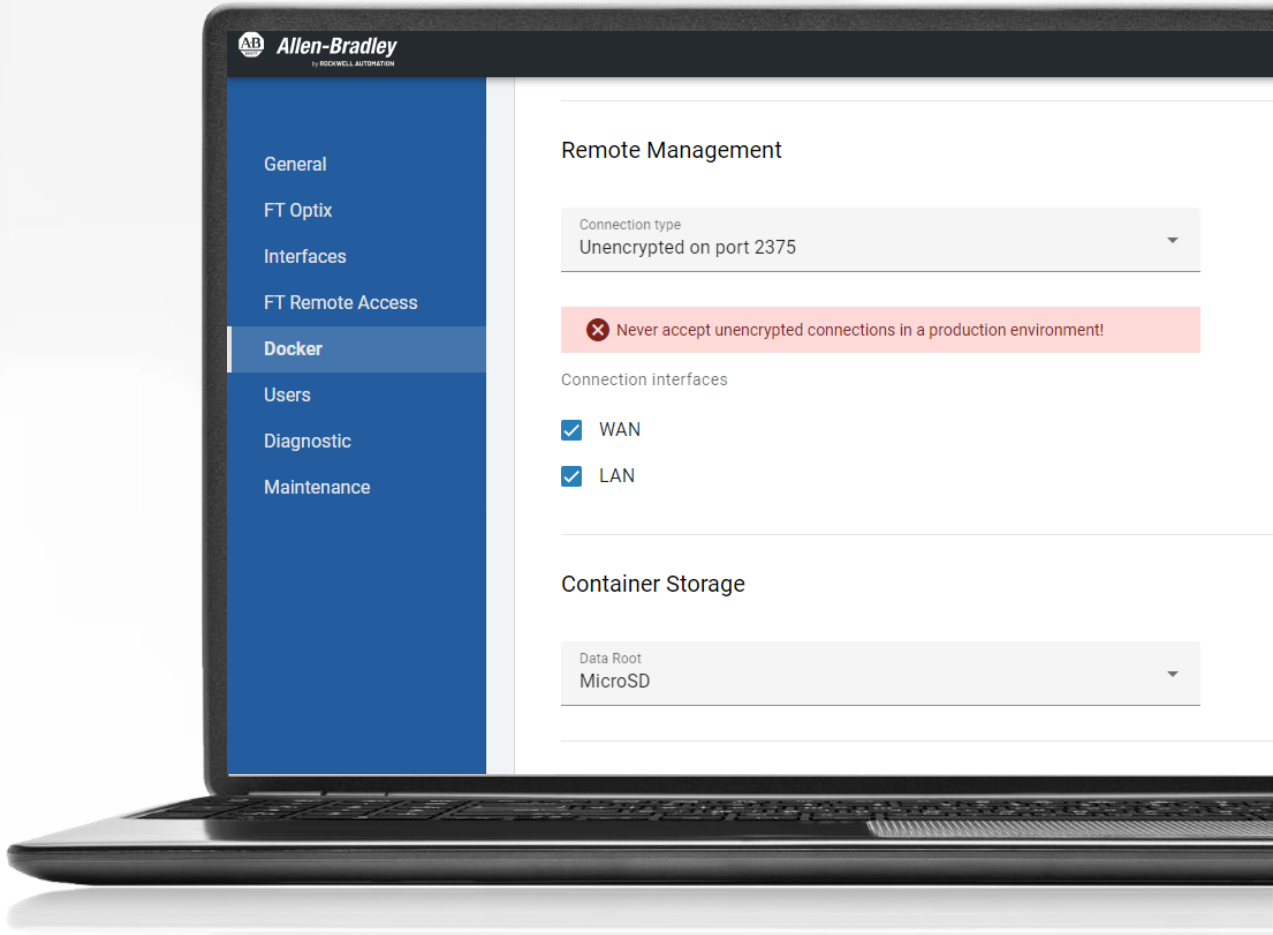
expanding **human possibility**®

PUBLIC

Enable Docker CLI SystemManager

Docker can connect to third-parties agents

- Enable Remote Management
  - Select which connection type to be used
  - Select which interfaces can be used to access the CLI

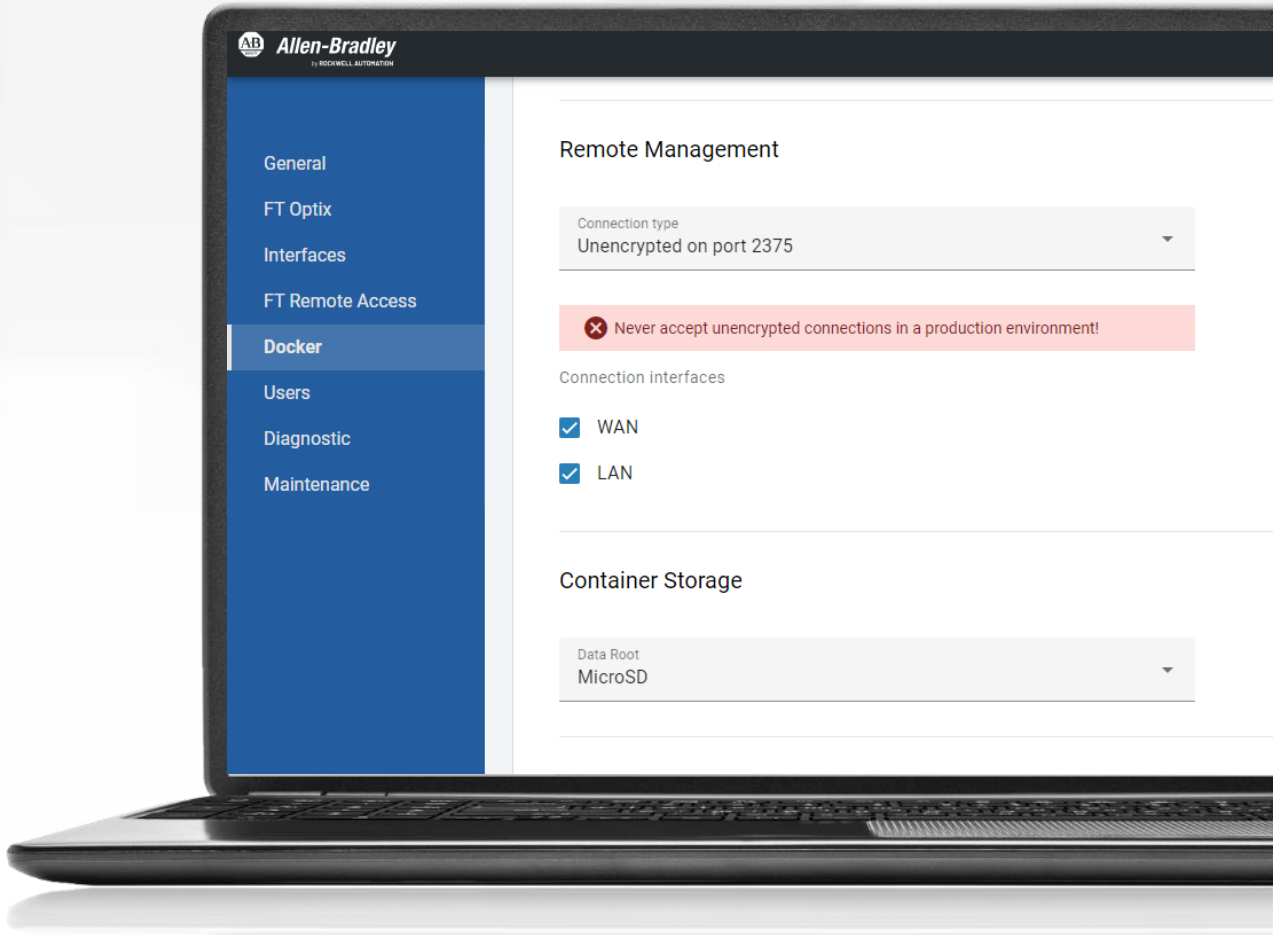


Connect the docker daemon to the agent

Docker can connect to third-parties agents

- Set the environment variable on the local machine
  - Unset the variable to restore normal functioning

```
$ export DOCKER_HOST=ipaddress:2375
$ […]
$ unset DOCKER_HOST
```



Loading a container with Docker CLI

Rockwell Automation

expanding **human possibility**®

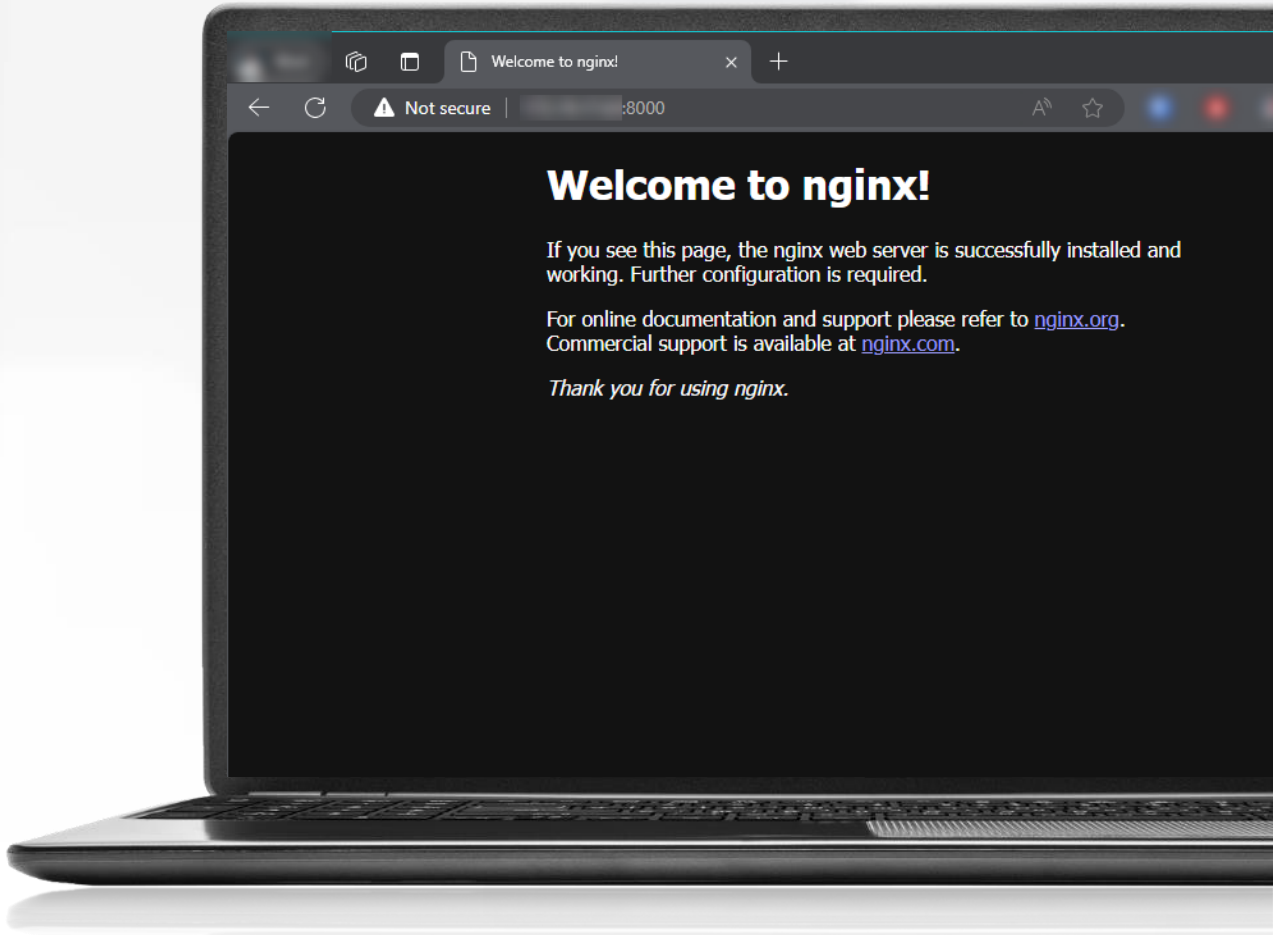
PUBLIC

Sending commands to unencrypted agent

Works just like a local docker daemon

- Set the environment variable
- Send some commands to the agent
  - Works just like a local docker machine
  - Commands are forwarded to the 1756-CMEE automatically

```
$ export DOCKER_HOST=‘ipaddress:2375’ # Example: “192.168.0.1:2375”
$ docker ps –a # List the running containers
$ docker run --name web-server -d -p 8000:80 nginx # Starts nginx
$ unset DOCKER_HOST # Disconnect from module
```

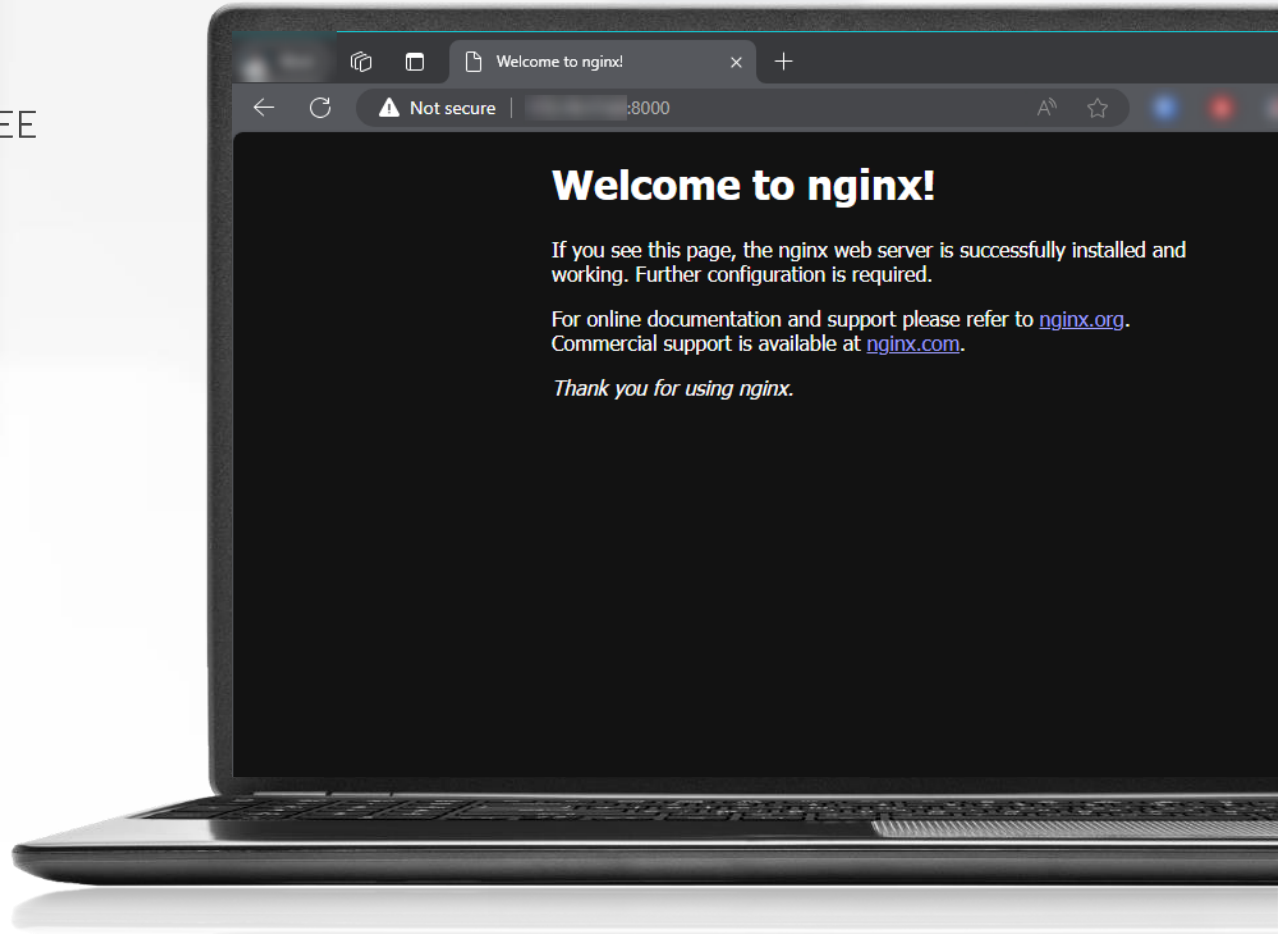


Sending commands to encrypted agent

Works just like a local docker daemon

- Generate (or get) a set of certificates
  - Needs CA, public key and private key
    - CA and server certificates are the same for both client and 1756-CMEE
    - Private key files are different for client and 1756-CMEE (asymmetric encryption)
  - All files must be «.pem» format
- Set Remote Management to «Encrypted»
  - Load certificate files
  - Select which interfaces to listen

```
$ export DOCKER_HOST='ipaddress:2376’
$ export DOCKER_CERT_PATH=/path/to/client/certificates
$ export DOCKER_TLS_VERIFY=1
$ docker ps –a # List the running containers
$ docker run --name web-server -d -p 8000:80 nginx # Starts nginx
$ unset DOCKER_HOST # Disconnect from module
```



Access USB devices

Rockwell Automation

expanding **human possibility**®

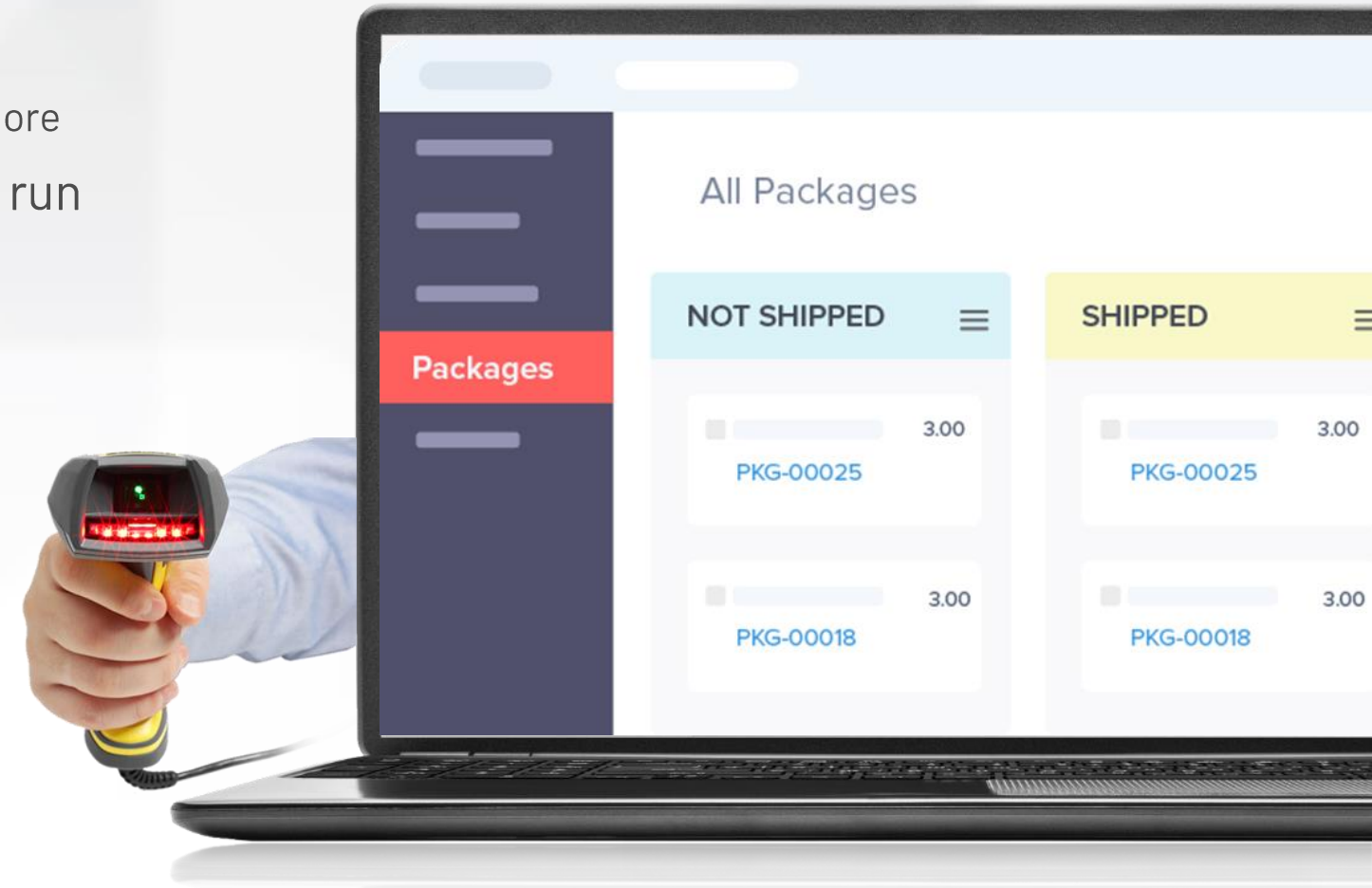
PUBLIC

Enabling Docker access to USB

Containers can access devices on the USB port

- Enable the feature in SystemManager
  - Can be used to access disk drives, usb sticks and more
- No need to add the device argument to the run command

```
$ export DOCKER_HOST=ipaddress:2375
$ docker run -it ubuntu bash
Ubuntu# apt update && apt install –y usbutils
Ubuntu# lsusb
1: Bus 001 Device 002: ID 23a9:ef18 USB DISK
Ubuntu# exit
$ unset DOCKER_HOST
```



Building a custom image

Rockwell Automation

expanding **human possibility**®

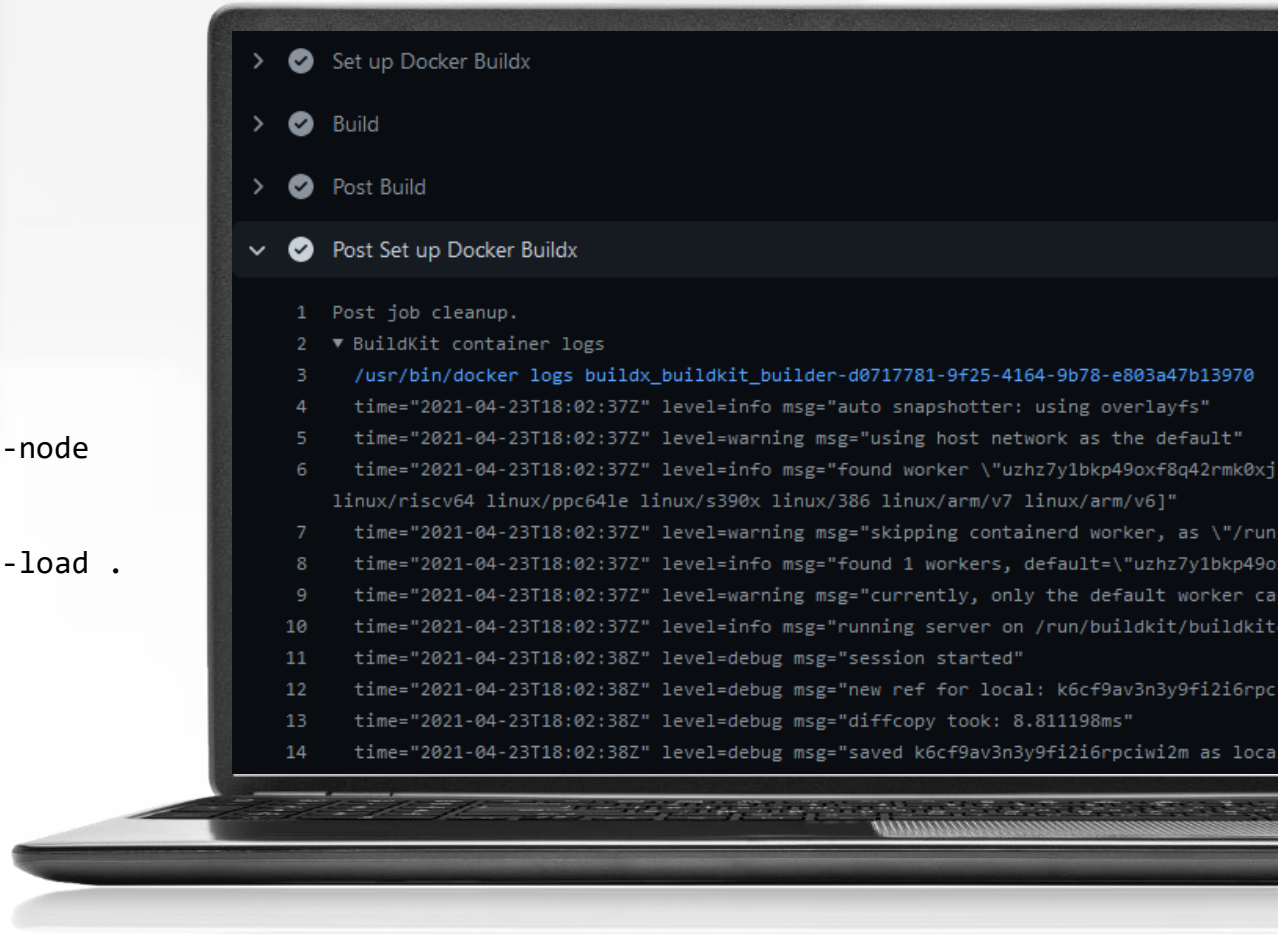
PUBLIC

Building a custom image

Make sure to cross compile with buildx

- Install buildx and dependencies
- Use buildx to create the image
- Export the tar file
  - Output file can be imported via USB, Docker CLI or Portainer

```
$ sudo apt-get install -y qemu-user-static
$ sudo docker buildx create --name eec1756 --platform linux/arm64 --node
node_eec1756 --driver docker-container --bootstrap –use
$ sudo docker buildx use eec1756
$ sudo docker buildx build --platform linux/arm64 -t eec1756_test --load .
$ sudo docker save -o eec1756_test.tar eec1756_test
```



Accessing the Backplane

Rockwell Automation

expanding **human possibility**®

PUBLIC

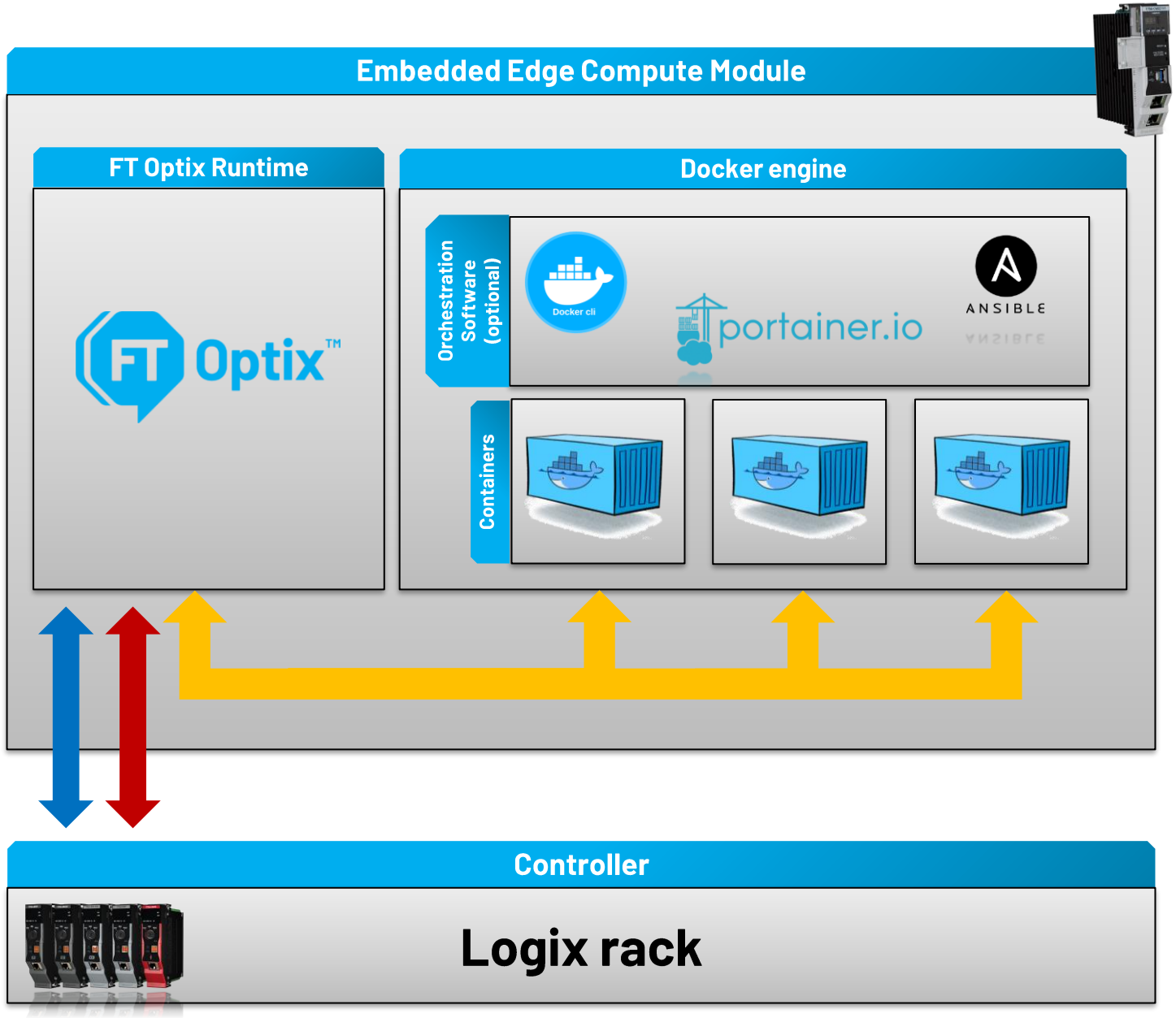
Communication overview

1756-CMEE Backplane

FT Optix RA EthernetIP driver

IP protocol

- An external Logix controller can be accessed either via Ethernet/IP or via Backplane
- Optix communicates with the controller using either the Backplane or the Ethernet/IP protocol, then expose the tags using some other protocol (e.g. OPC/UA)
- Direct access to Backplane from conainers will come soon



expanding **human possibility**®