KubeCon | CloudNativeCon

Europe 2023

Love, Death and Robots - with Wasm & K8s on Boston Dynamics Spot

Who are we?

Max Körbächer

Founder and Cloud Native Advisor

- ➔ CNCF Ambassador
- ➔ Co-Chair CNCF TAG Environmental Sustainability
- ➔ Contributed to the Kubernetes release team for 3y and related K8s technologies

Kevin Hawryluk

Head of Technology

- ➔ Can’t be with us today

REPLY LIQUID

REPLY ROBOVERSE

Meet Spot

Spot is a mobile robot that can carry around **14kg of equipment**.

Can go almost anywhere, by controller or **autonomously**.

Is **customizable** by different mountable hardware, via an SDK and pre build solutions.

Source: BostonDynamics

So containers can run anywhere, huh?

For me, Spot looks like an agile server with cameras, legs and a Nintendo controller.

Can I put a containers & Kubernetes on it?

Source: BostonDynamics

Everything starts with a stupid idea

Source: BostonDynamics

Spot Core & Spot Core IO

Spot itself is a black box but extensible

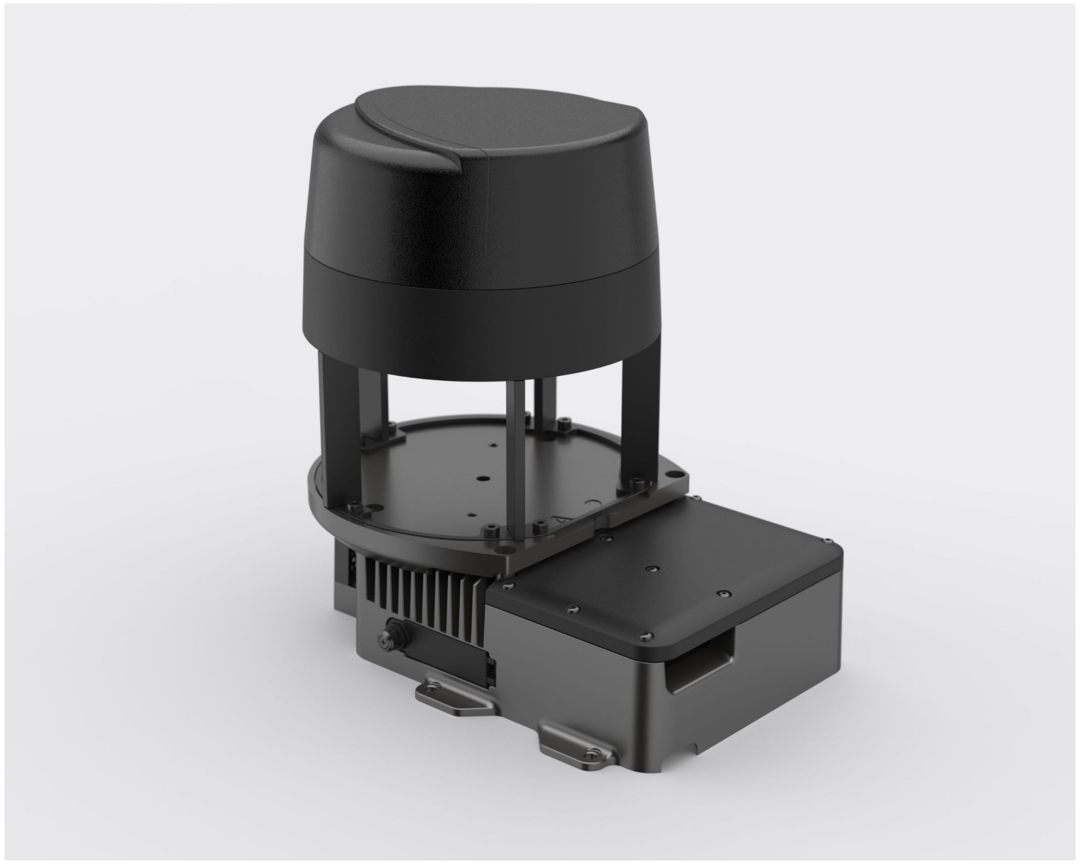
Item	Description
Motherboard	i5 Intel® 8th Gen (Whiskey Lake-U) Core™ CCG Lifecycle
Memory (RAM)	16 GB SO-DIMM DDR4 2666
Primary storage	512 GB M.2 22x60 SSD
Physical interfaces	1 x RS-232 ports 3 x USB 3.1 2 x USB 2.0 2 x DisplayPort 1 x HDMI
Accessories	Computer port & dust blocking kit Power adapter DC 12 V, 5A, 60 W Level VI
Operating system	Ubuntu Desktop 18.04 LTS 64-bit

Source: BostonDynamics

Spot Core & Spot Core IO

Spot itself is a black box but extensible

Item	Description
CPU	6-core NVIDIA Carmel ARM V8.2 64-bit CPU with 6MB Lw + 4MB L3 cache
GPU	384-core NVIDIA Volta GPU with 48 Tensor cores
Memory	16GB 128-bit LPDDR4x at 51.2 GB/s
5G/LTE	User-installable SIM card
Disk encryption	SSD encrypted with standard LUKS technology
Network encryption	Connections encrypted with TLS 1.2 and 1.3
Authentication	Access to services restricted to authenticated users
Secure boot	Tamper-proof filesystem with hardware root of trust
Firmware verification	Firmware updates must be cryptographically signed



Source: BostonDynamics

Optimization Potential

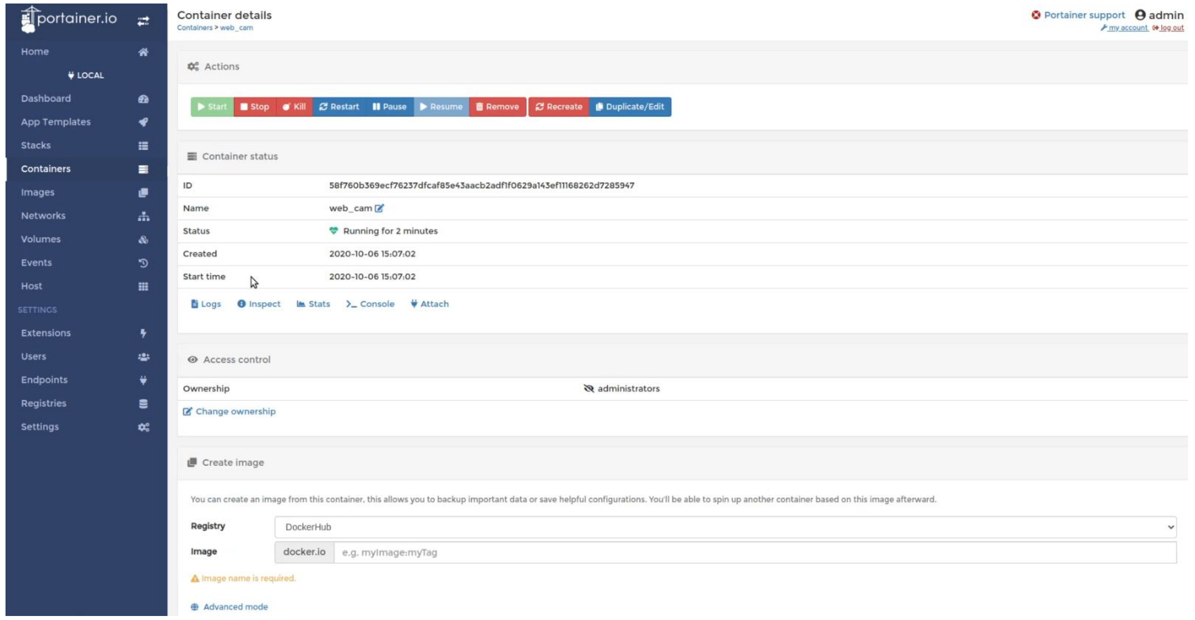
REPLY
LIQUID

Surprise!
Containers are already there

On Spot Core + Core I/O we have:

- Ubuntu 18.04
- **Portainer on spot Core**
- **Custom Docker interface on Core I/O**
- Different processor architectures
- GPUs
- Accessible and secured storage

Core is going to be deprecated in future.



Container details

Containers > web_cam

Actions

Container status

ID	58f760b369ecf76237dfcaf85e43aacb2adf1f0629a143ef11168262d7285947
Name	web_cam
Status	Running for 2 minutes
Created	2020-10-06 15:07:02
Start time	2020-10-06 15:07:02

Access control

Ownership	administrators

Create image

You can create an image from this container, this allows you to backup important data or save helpful configurations. You'll be able to spin up another container based on this image afterward.

Registry	DockerHub
Image	docker.io e.g. myImage:myTag

Image name is required.

Some downsides of this setup

- No in depth observability given
- No internal security scans possible (yet)
- Health checks are missing
- Node awareness and handling not given
- No custom roles
- Can’t change settings of the cluster

It is a good starting point, but many basic features you most of the time wish to have, are missing.

Current development process

For Spot Core I/O

Development	Continuous Integration	Continuous Deployment	Configuration	Test & Execution
Most part of the software build by Roboverse is written in Python. Development happens locally.	If the software is ready to be deployed, it needs to be packages as a tar ball.	And gets then manually uploaded through a GUI to the Spot.	Unpacked, the new software must be configured and ensured that all configs are set correctly.	Lastly, we are good to go for testing and executing the newly written functions.

- No automation possible
- The dev cycle is slowed down dramatically
- Containers are not very secure

Our dirty road of changes

Trigger warning!

What we are doing here is experimental as we try to discover better approaches

Our target setup

Before we were aware of recent changes of Core I/O

- ➔ Increase system visibility to all extends (full monitoring & logging of the node, pods and any other actions happening on the core)
- ➔ Optimize the development cycle by speeding up the process and deployment to spot -> also interesting for field operations & autonomy
- ➔ Provide a high level of availability and self healing capabilities
- ➔ Security, security & security

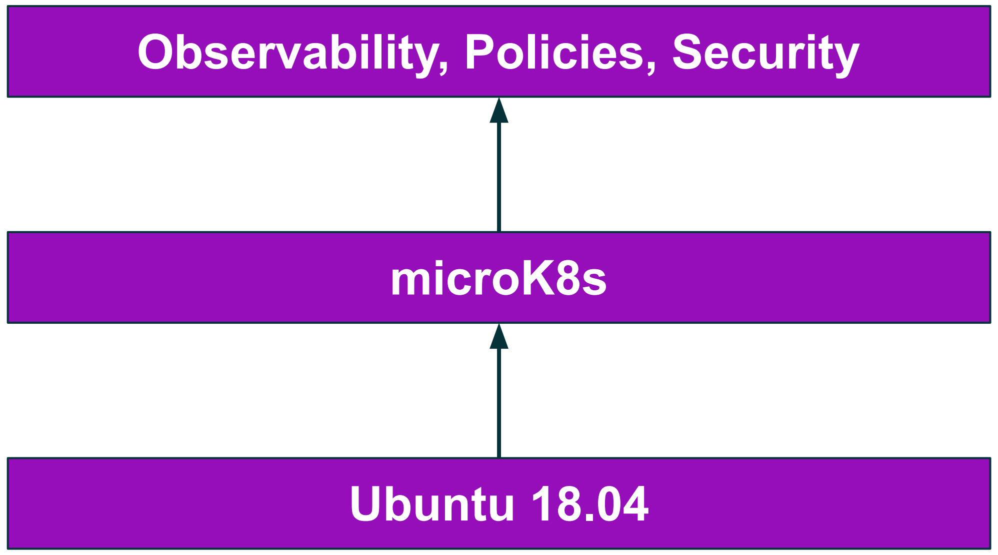
Replacing Portainer

Besides the mentioned downsides we also have:

Portainer/Compose is failing when sensors are missing/not connected.

The error handling causes drastical miss behaviour of the entire stack.

```
sudo snap install microk8s --classic
--channel=1.27
```

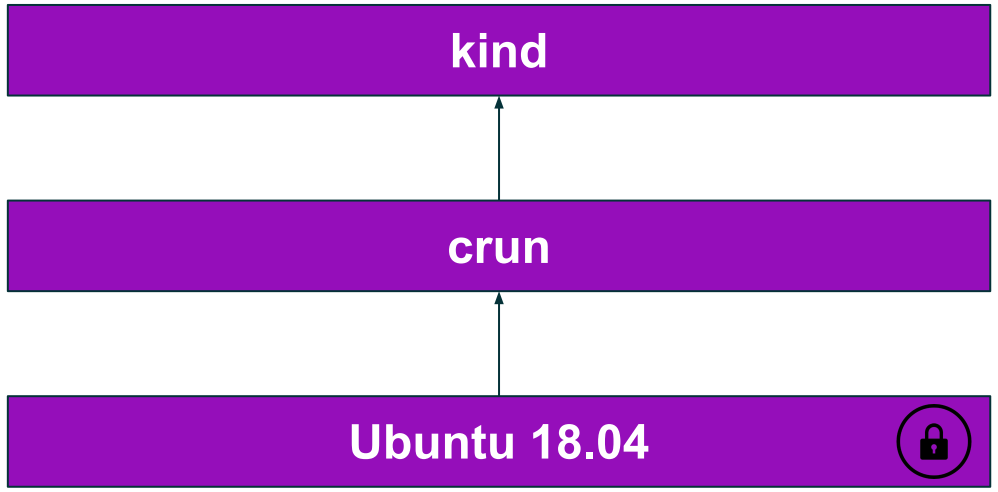


Replacing Core I/O interface

The latest version of Core - I/O got rid of Portainer to, and runs an own container runtime.

This solution prevents at the moment running anything then a simple container.

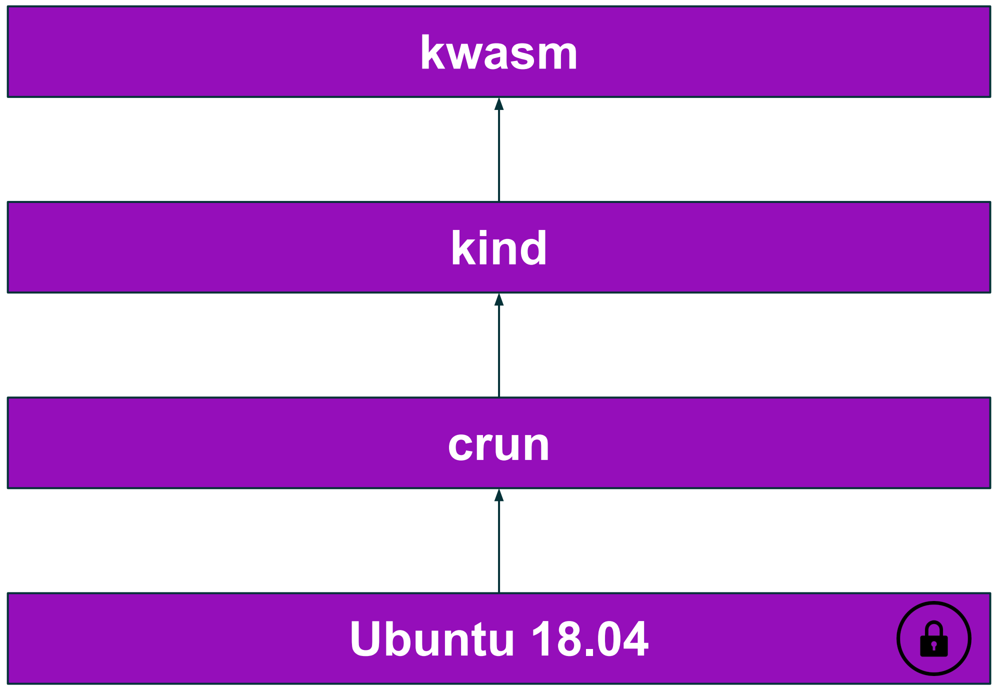
soooo….



Extending Core I/O with wasm? because we can

Why to do that?

- Different target architectures provide a pain point during development
- Having many different versions of containers at the same time eating up space
- Increase the security limit



Providing WASM as additional runtime

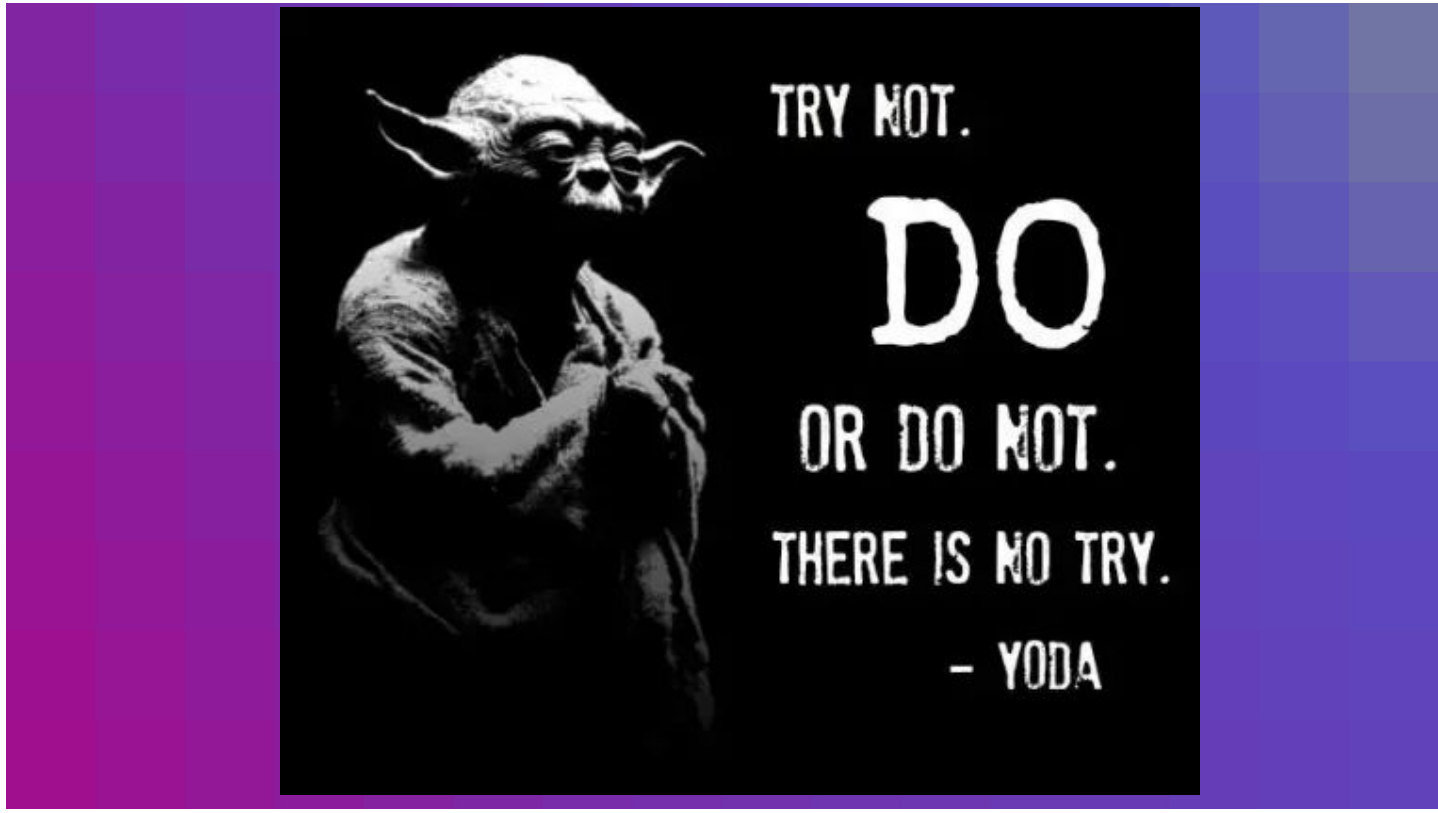
```
# Add helm repo

helm repo add kwasm http://kwasm.sh/kwasm-operator/

# Install operator

helm install -n kwasm --create-namespace kwasm-operator
kwasm/kwasm-operator

# Annotate single node

kubectl annotate node kind-worker kwasm.sh/kwasm-node=true

# Run example

kubectl apply -f examples/kind/runtimeclass.yaml
kubectl apply -f examples/kind/pod.yaml
```

Issues!

Still WIP

Bringing the different machine learning models into wasmSpot written yet in Python is kinda tricky.

Simple apps are no issue, but the complexity of different dependencies are not yet solved.



GitOps in Spot

What we want to achieve:

- Running ArgoCD in kind looking for new images whenever it has connection (5G/Wifi) or idles (charging station, planned break in a route).
- During development, this speed things up drastically.
- During working in the field and client side, this will allow to reduce the maintenance costs.

The dirty way we took:

- Quick and dirty implementation, no user management, just defaults
- We didn’t evaluate “the right” solution, might be replaced in future
- Have to build a scaling process around it

Newly gained capabilities

Improved development cycle

Bringing GitOps to the spot, right now ArgoCD works fine for this and via wasm the custom written software run on any robot.

Increased Developer experience and faster deployment of new versions.

Security (again)

Sealing of critical control mechanisms in wasm to prevent miss usage. Forbidding activities that a container shouldn't do.

Spot is a very sensitive robo-dog (sorry), catch it, plug a cable, upload your image -> not any more!

Autonomy

Event driven starting of wasm application & ArgoCD updates via Mobile/Wifi connection.

No further more tar uploads via the backdoor cable and a use case for real field operations.

Opened for fine grained extensibility

Fine tuned access management, restrict policies, only signed images are allowed to be executed. Plus whatever else you want!

Getting control of the platform while improving security. In addition, we can now optimize the workload on Spot.

What we learned:

Spot is really just a walking server with sensors (and more).

It’s almost disappointing how easy Kubernetes & wasm got already.

Machine learning is not yet ready for wasm.

Stupid ideas can turn in something relevant, BUT

Keep in mind, this is experimental, we are not done yet!

Our cloud native ecosystem is maturing. To implement stuff getting way to easy.

REPLY
LIQUID

KubeCon | CloudNativeCon

Europe 2023

Feedback & Questions