Hardware Optimized OCI Images via MetaHub Registry Proxy

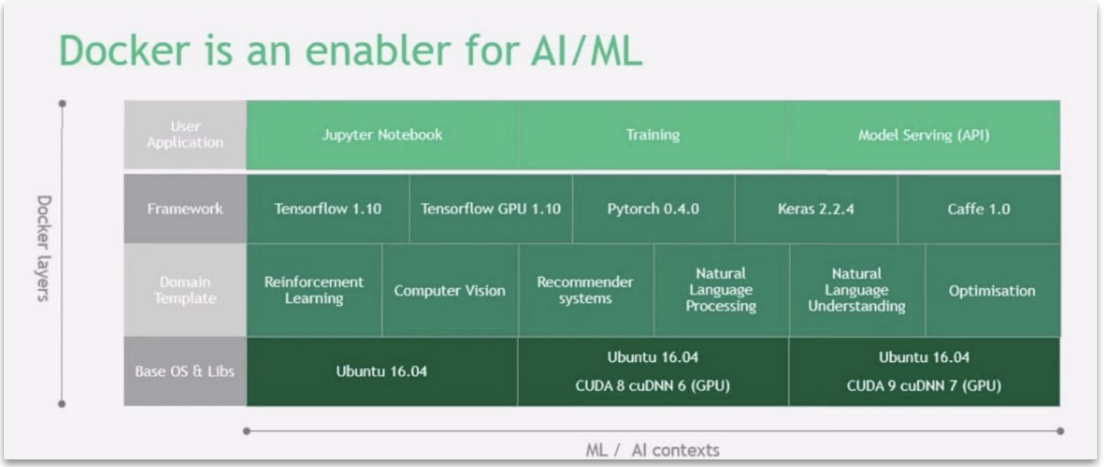
High Performance Container Workshop - ISC19

Christian Kniep, QNIB Solutions
christian@qnib.org

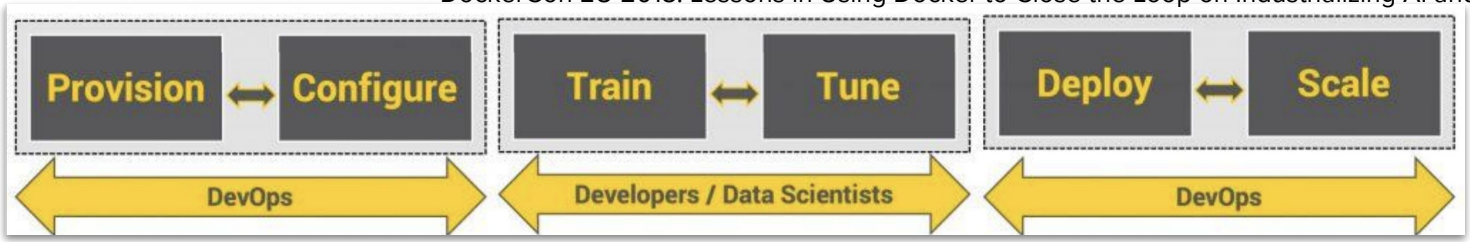
Image Lifecycle

Image Lifecycle needs Composability and Workflow

- Multiple phases in ML livecycle
- Modular Assembly of Software Stacks
- With drivers and growing set of hardware and software, curating these stacks is hard.



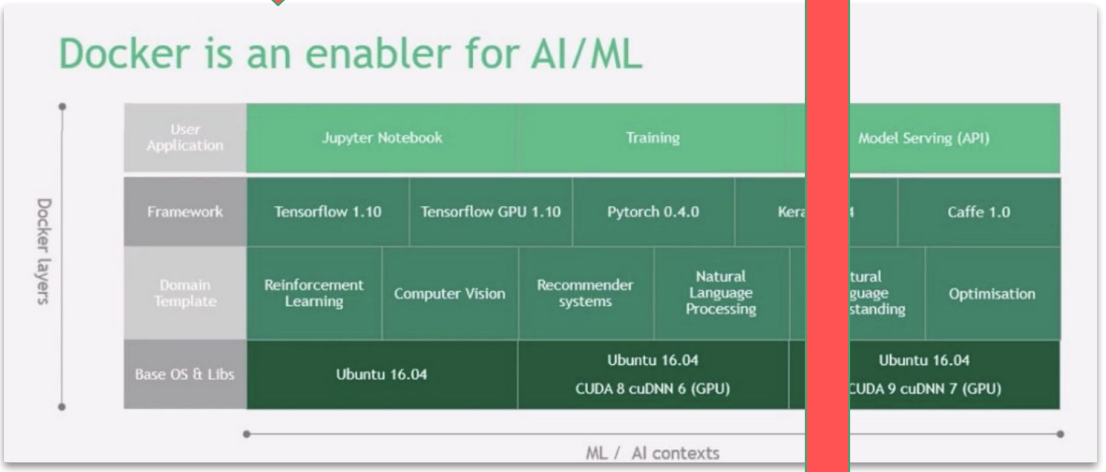
DockerCon EU 2018: Lessons in Using Docker to Close the Loop on Industrializing AI and ML Applications



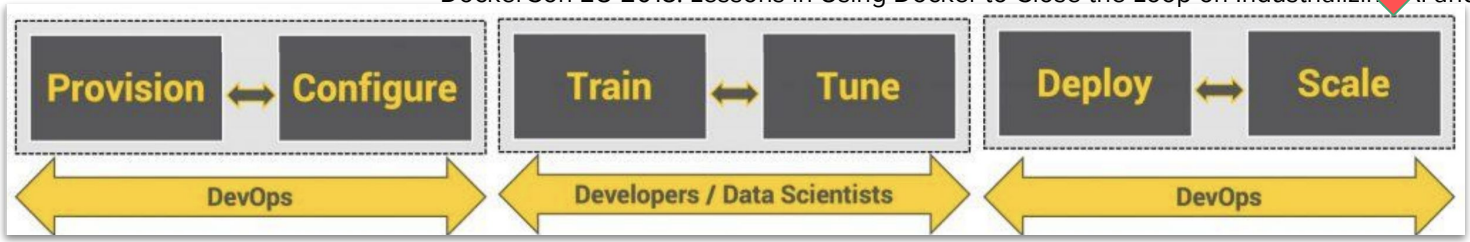
source: https://thenewstack.io/an-introduction-to-the-machine-learning-platform-as-a-service/

Image Lifecycle needs **Composability** and **Workflow**

- Multiple phases in ML livecycle
- Modular Assembly of Software Stacks



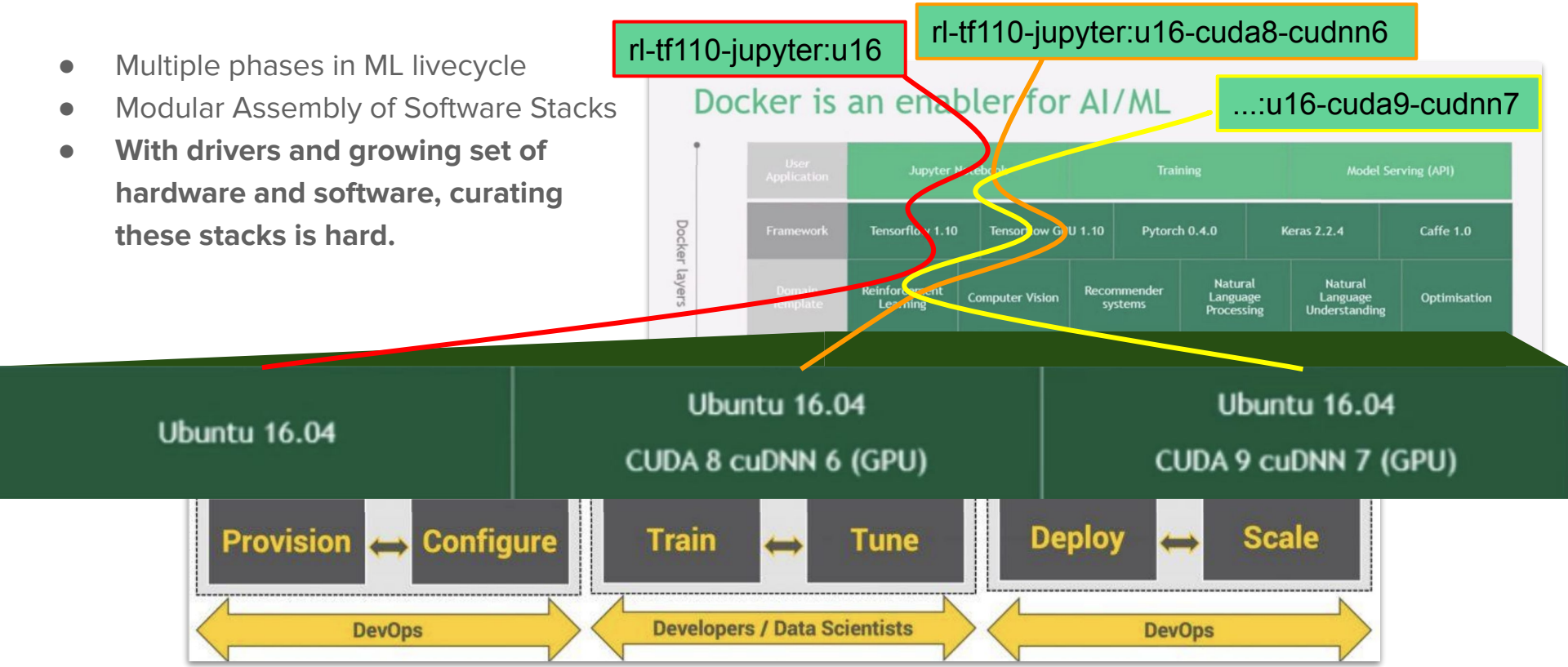
DockerCon EU 2018: Lessons in Using Docker to Close the Loop on Industrializi AI and ML Applications



source: https://thenewstack.io/an-introduction-to-the-machine-learning-platform-as-a-service/

Image Lifecycle needs Composability and Workflow

- Multiple phases in ML livecycle
- Modular Assembly of Software Stacks
- **With drivers and growing set of hardware and software, curating these stacks is hard.**



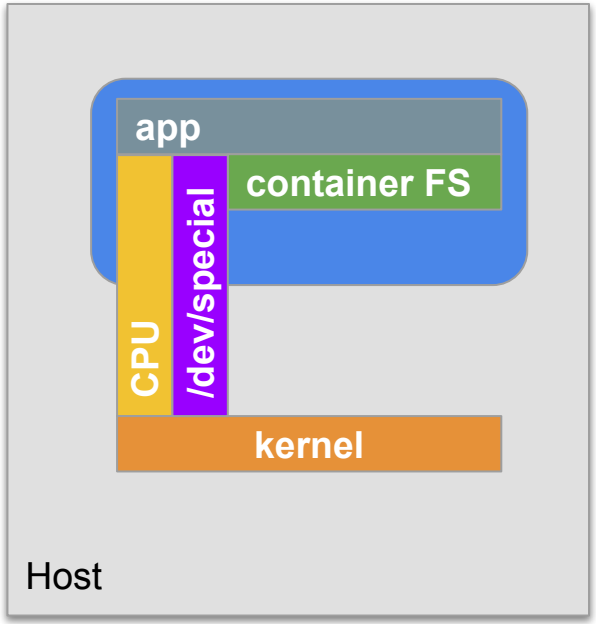
source: https://thenewstack.io/an-introduction-to-the-machine-learning-platform-as-a-service/

Hardware Optimization & Matching

Host-Agnostic vs. Host-Specific

Containers use Kernel-level isolation; being portable among many different systems as a result.
But the portability only guarantees executing, not performance.

Optimization for CPU architectures are only possible if they do not limit the execution.
avx512 is not necessarily available everywhere.



Platform Definition within OCI

The platform object within the “OCI Image Spec” differentiate the architecture by using $GOARCH.

But a container runtime can not pick a container tailored to a microarchitectures.

$GOOS	$GOARCH
linux	386
linux	amd64
linux	arm
linux	arm64
linux	ppc64
linux	ppc64le
linux	mips

- `platform` *object*

  This OPTIONAL property describes the minimum runtime requirements of the image. This property SHOULD be present if its target is platform-specific.

  - `architecture` *string*

    This REQUIRED property specifies the CPU architecture. Image indexes SHOULD use, and implementations SHOULD understand, values listed in the Go Language document for `GOARCH`.

  - `os` *string*

    This REQUIRED property specifies the operating system. Image indexes SHOULD use, and implementations SHOULD understand, values listed in the Go Language document for `GOOS`.

https://github.com/opencontainers/image-spec/blob/master/image-index.md

Platform Definition within OCI

The platform object within the “OCI Image Spec” differentiate the architecture by using $GOARCH.

But a container runtime can not pick a container tailored to a microarchitectures.

$GOOS	$GOARCH
linux	386
linux	amd64
linux	arm
linux	arm64
linux	ppc64
linux	ppc64le
linux	mips

- `platform` *object*

  This OPTIONAL property describes the minimum runtime requirements of the image. This property SHOULD be present if its target is platform-specific.

  - `architecture` *string*

    This REQUIRED property specifies the CPU architecture. Image indexes SHOULD use, and implementations SHOULD understand, values listed in the Go Language document for `GOARCH`.

  - `os` *string*

    This REQUIRED property specifies the operating system. Image indexes SHOULD use, and implementations SHOULD understand, values listed in the Go Language document for `GOOS`.

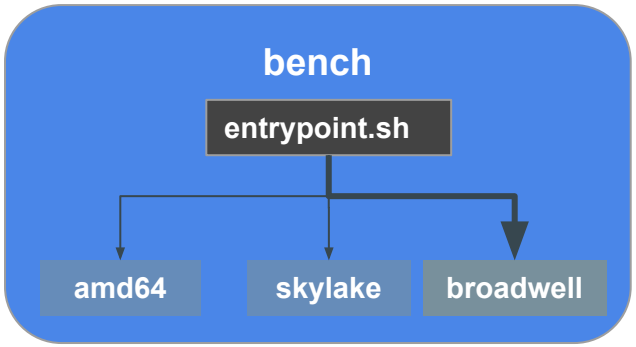
https://github.com/opencontainers/image-spec/blob/master/image-index.md

Year	Micro-architecture	Pipeline stages	max. Clock	Tech process
2018	Cannon Lake	14 (16 with fetch/retire)	3200 MHz	10 nm
2018	Whiskey Lake	14 (16 with fetch/retire)	4600 MHz	14 nm
2018	Amber Lake	14 (16 with fetch/retire)	4200 MHz	14 nm
2017	Coffee Lake	14 (16 with fetch/retire)	5000 MHz	14 nm
2017	*Goldmont Plus*	? 20 unified with branch prediction ?	2800 MHz	14 nm
2016	*Goldmont*	20 unified with branch prediction	2600 MHz	14 nm
2016	Kaby Lake	14 (16 with fetch/retire)	4500 MHz	14 nm
2015	*Airmont*	14-17 (16-19 with fetch/retire)	2640 MHz	14 nm
2015	Skylake	14 (16 with fetch/retire)	4200 MHz	14 nm
2014	Broadwell	14 (16 with fetch/retire)	3700 MHz	14 nm

Current State of Affairs

For the container to be portable, it needs to fall back to the lowest common denominator of the target systems.
Worst case: amd64 (x86_64).

Some try to mitigate the problem by using an entrypoint, which chooses the appropriate binary at runtime.



But the image still needs to be changed every time a new hardware configuration is added.

Current State of Affairs [cont]

Images optimized for different CPU μ-arch are picked by name.

As described in the previous slides.


```
$ docker run -ti --rm qnib/bench:cpu-broadwell
>> This container is optimized for: cpu:broadwell
```

```
$ docker run -ti --rm qnib/bench:cpu-skylake
>> This container is optimized for: cpu:skylake
```

```
$ docker run -ti --rm qnib/bench:generic
>> This container is not optimized for a specific microarchitecture
```

```
$ docker run -ti --rm qnib/bench:cpu_broadwell-nvcap_52
>> This container is optimized for: cpu:broadwell,nvcap:5.2
```

Thank you!