Kata Containers Meetup
Shanghai 2021

Kata Containers for Edge Computing

Feng Li (李枫)
hkli2013@126.com
May 22，2021

Agenda

I. Overview
- Edge Computing
- Kata Containers
- Testbed

II. LF Edge
- Overview
- EVE

III. Lightweight Kubernetes
- K3S
- Arkade
- Sandboxed Containers Operator

IV. Project ACRN
- Overview
- X86 only

V. Rethinking Infrastructure for Edge Cloud
- Linux

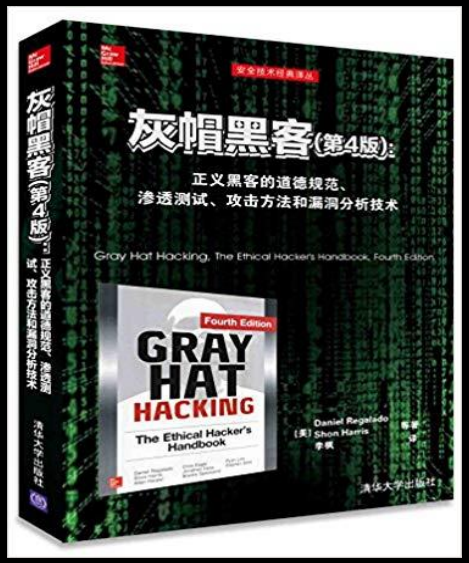
- eBPF-centric
- Rust
- WebAssembly
- Edge AI

- Serverless
- Unified Runtime
- Autonomous Vehicles
- RISC-V
- HW-SW Co-designed System Security

VI. Wrap-up

Who Am I

- The main translator of the book «Gray Hat Hacking The Ethical Hacker's Handbook, Fourth Edition» (ISBN：9787302428671) & «Linux Hardening in Hostile Networks, First Edition» (ISBN：9787115544384)



- Actively participate in various activities of the open source community
- https://github.com/XianBeiTuoBaFeng2015/MySlides
- Recently, focus on infrastructure of Edge Computing, AI, RISC-V, EDA…

I. Overview

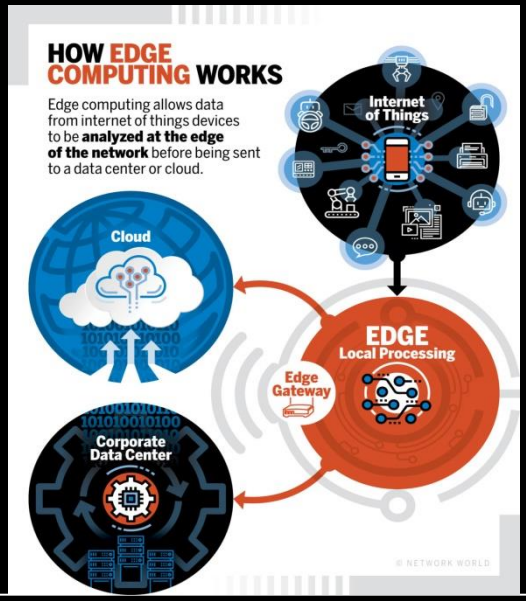
1) Edge Communicating

- https://en.wikipedia.org/wiki/Edge_computing

  a distributed computing paradigm which brings computation and data storage closer to the location where it is needed, to improve response times and save bandwidth.…

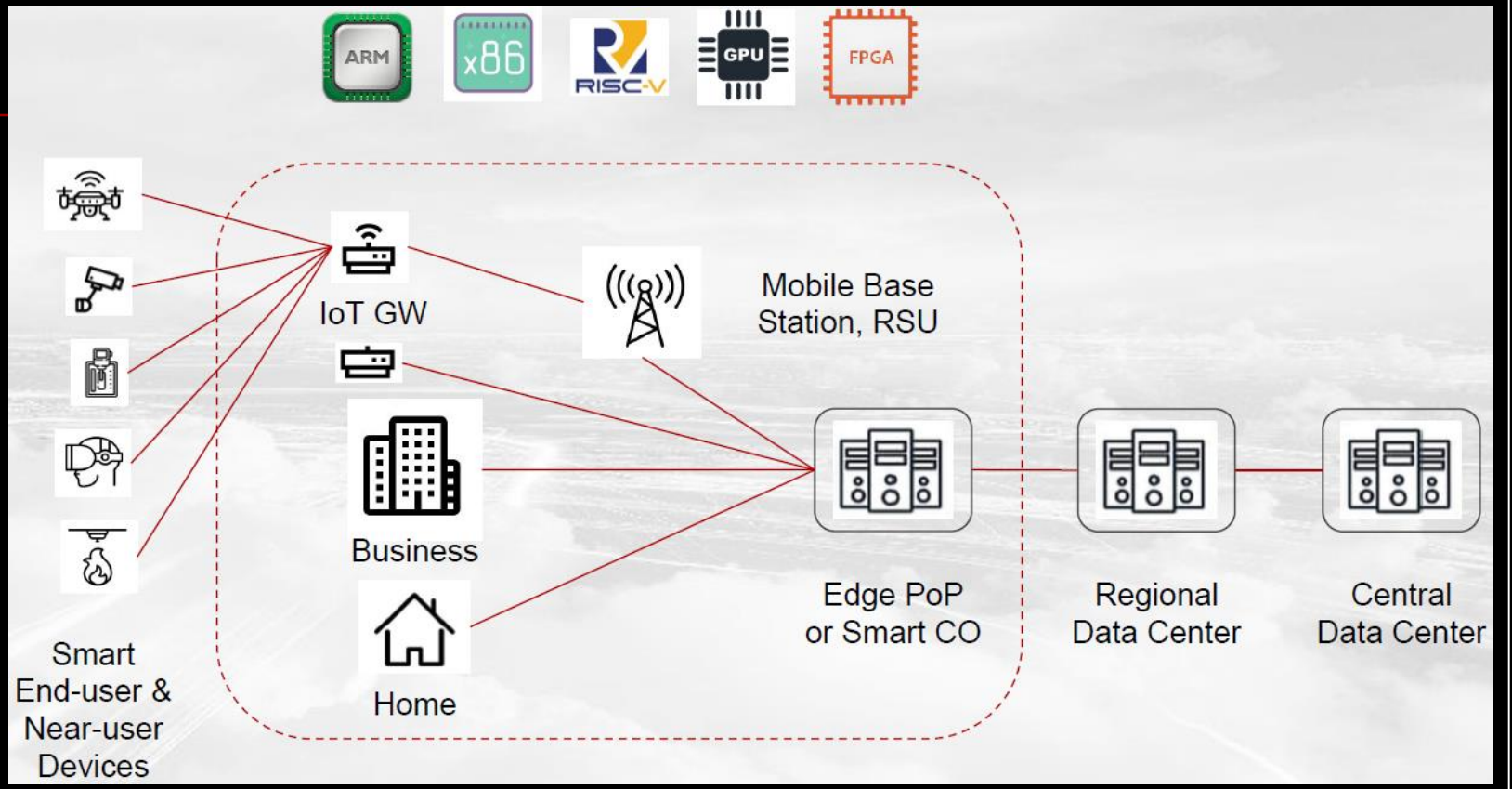
- https://www.networkworld.com/article/3224893/what-is-edge-computing-and-how-it-s-changing-the-network.html

  a way to streamline the flow of traffic from IoT devices and provide real-time local data analysis



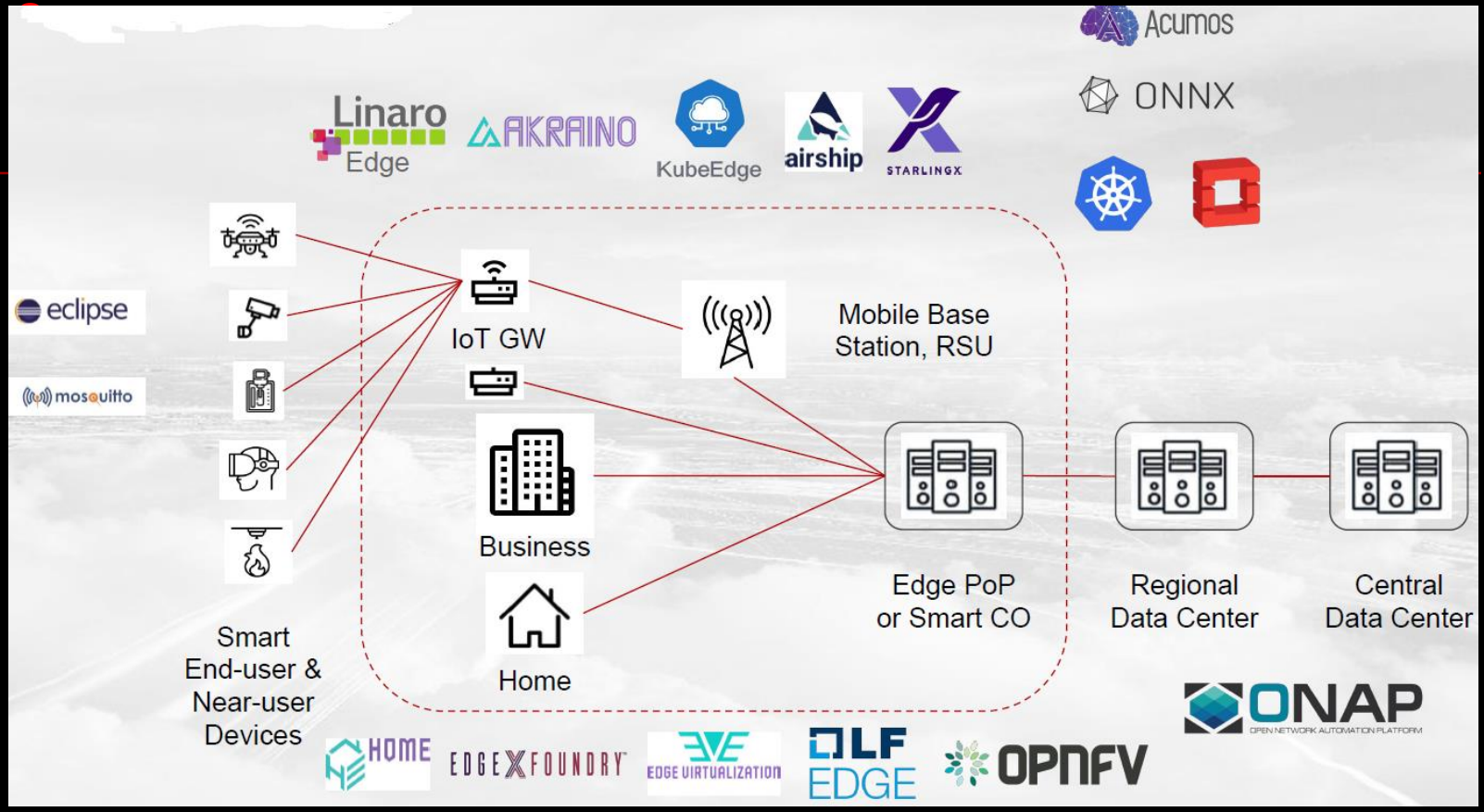
1.1 Intelligent Edge with Hardware, Software and Data

Hardware



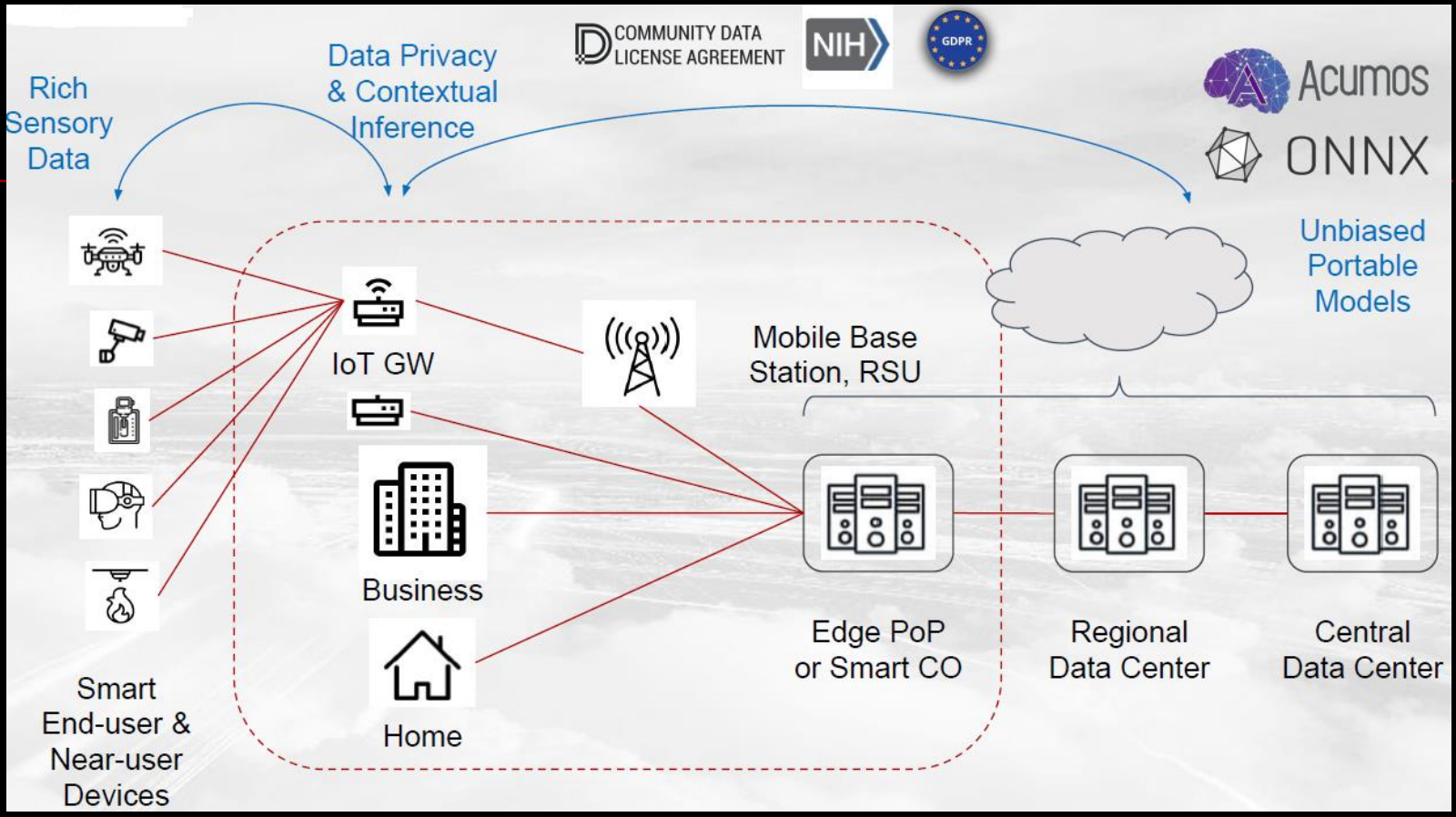
Source: "Deploying the New Intelligent Edge with Open Hardware, Software and Data", Wenjing Chu, ONS North America 2019

Software



Source: "Deploying the New Intelligent Edge with Open Hardware, Software and Data", Wenjing Chu, ONS North America 2019

Data

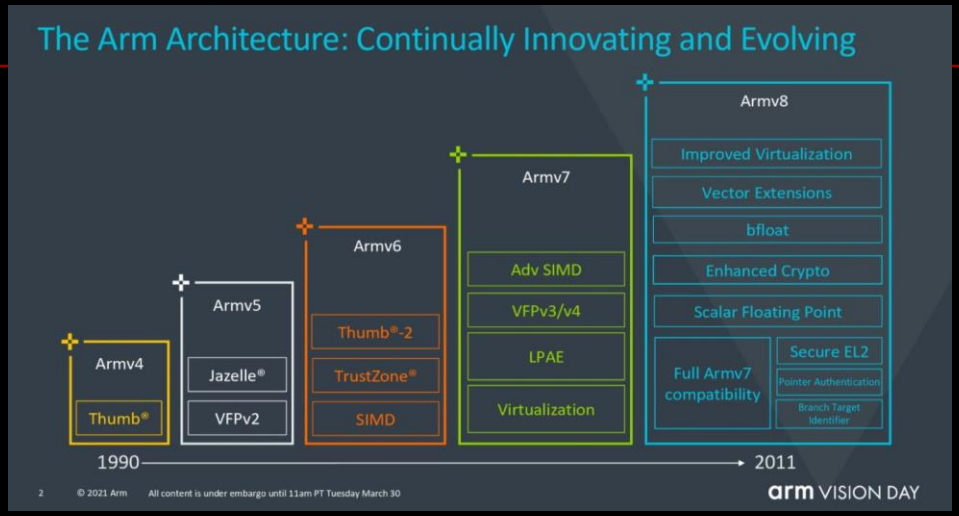


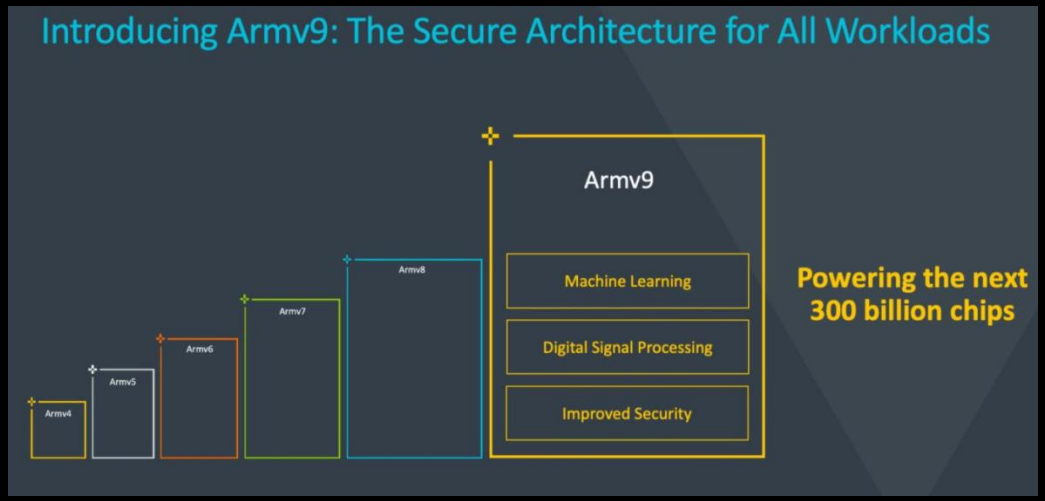
Source: "Deploying the New Intelligent Edge with Open Hardware, Software and Data", Wenjing Chu, ONS North America 2019

1.2 ARM Ecosystem for Edge Computing in 2021

ARM v9

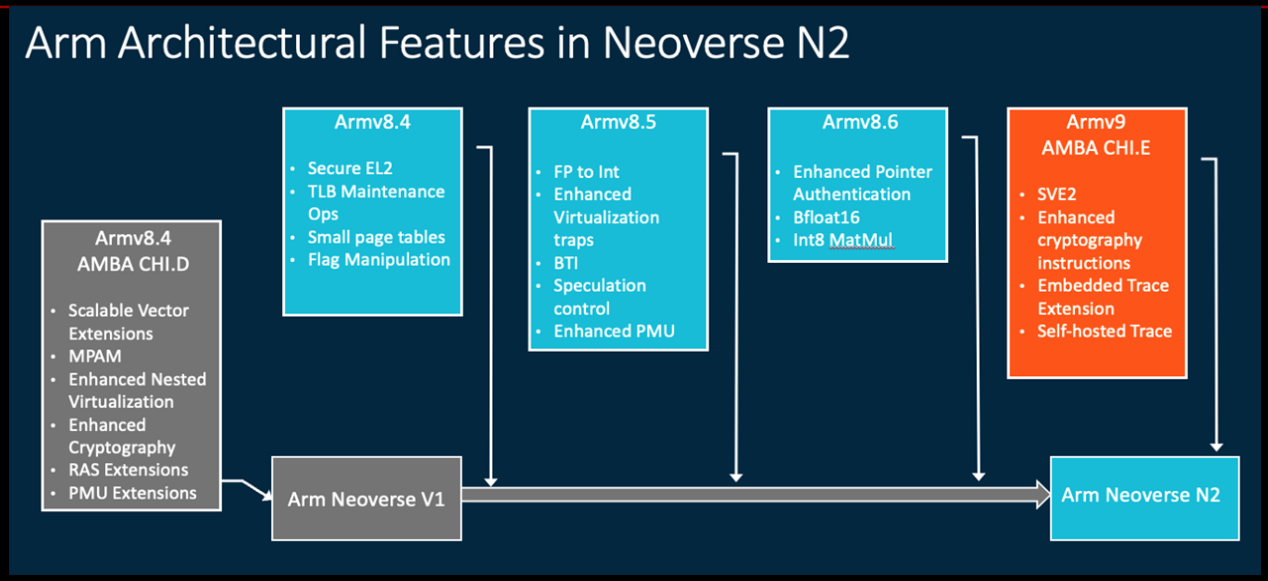
- https://www.arm.com/campaigns/arm-vision



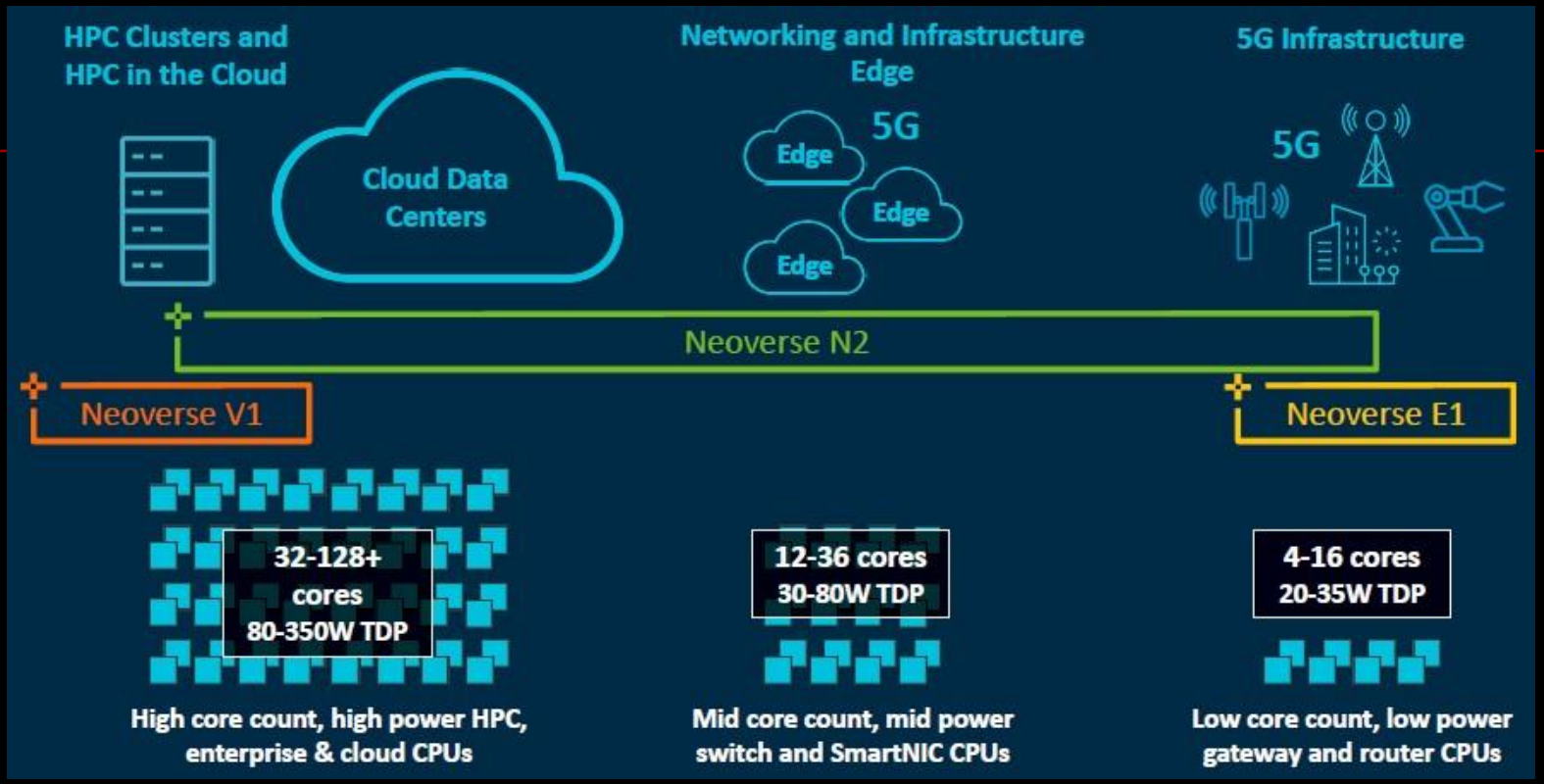


Neoverse N2

- **https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/arm-neoverse-n2-industry-leading-performance-efficiency**

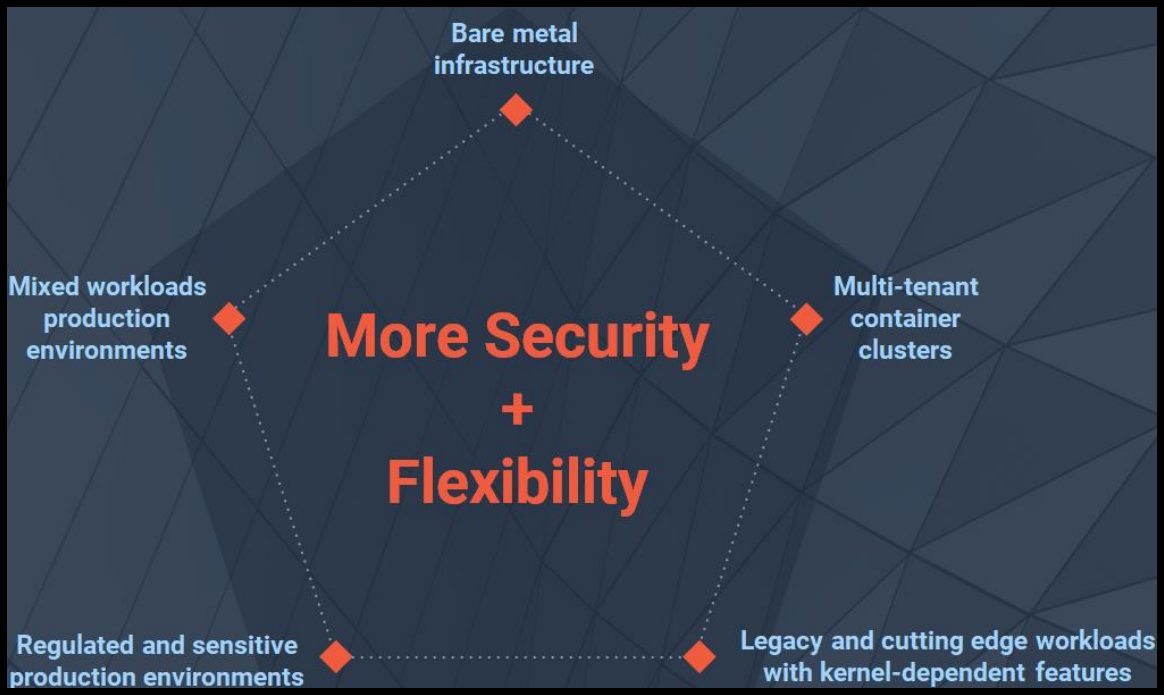


- https://www.nextplatform.com/2021/04/27/arm-puts-some-muscle-into-future-neoverse-server-cpu-designs/



2) Kata Containers

- https://katacontainers.io/
- a standard implementation of lightweight Virtual Machines (VMs) that feel and perform like containers, but provide the workload isolation and security advantages of VMs.

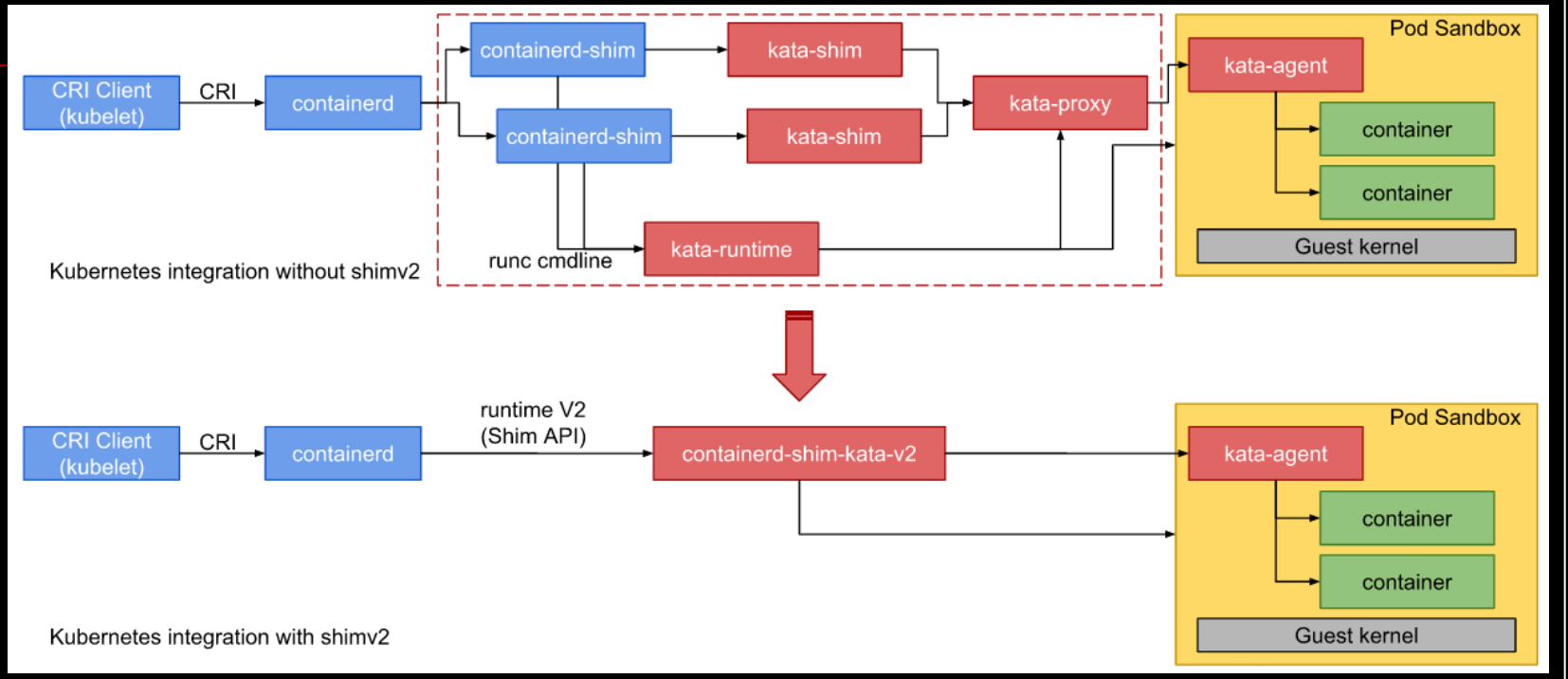


Source: https://katacontainers.io/collateral/kata-containers-onboarding-deck.pptx

- https://medium.com/kata-containers/kata-containers-2020-annual-report-c9ad7f44e0d4

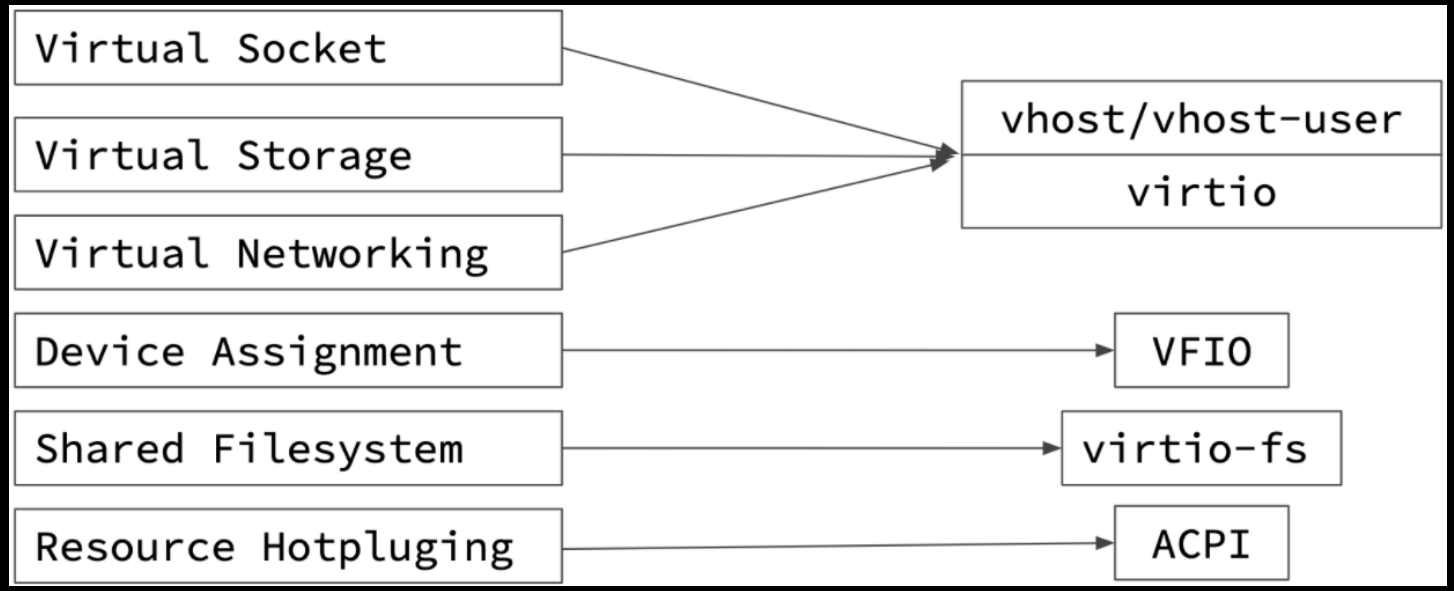
Architecture

- https://github.com/kata-containers/documentation/blob/master/design/architecture.md



Virtualization

- **https://github.com/kata-containers/documentation/blob/master/design/virtualization.md**
- **Hypervisor/VMM support:**
  - ACRN hypervisor
  - Cloud Hypervisor/KVM
  - Firecracker/KVM
  - QEMU/KVM
- **Each hypervisor/VMM varies on how or if it handles each of the specific para-virtualized devices or virtualization technologies as below:**

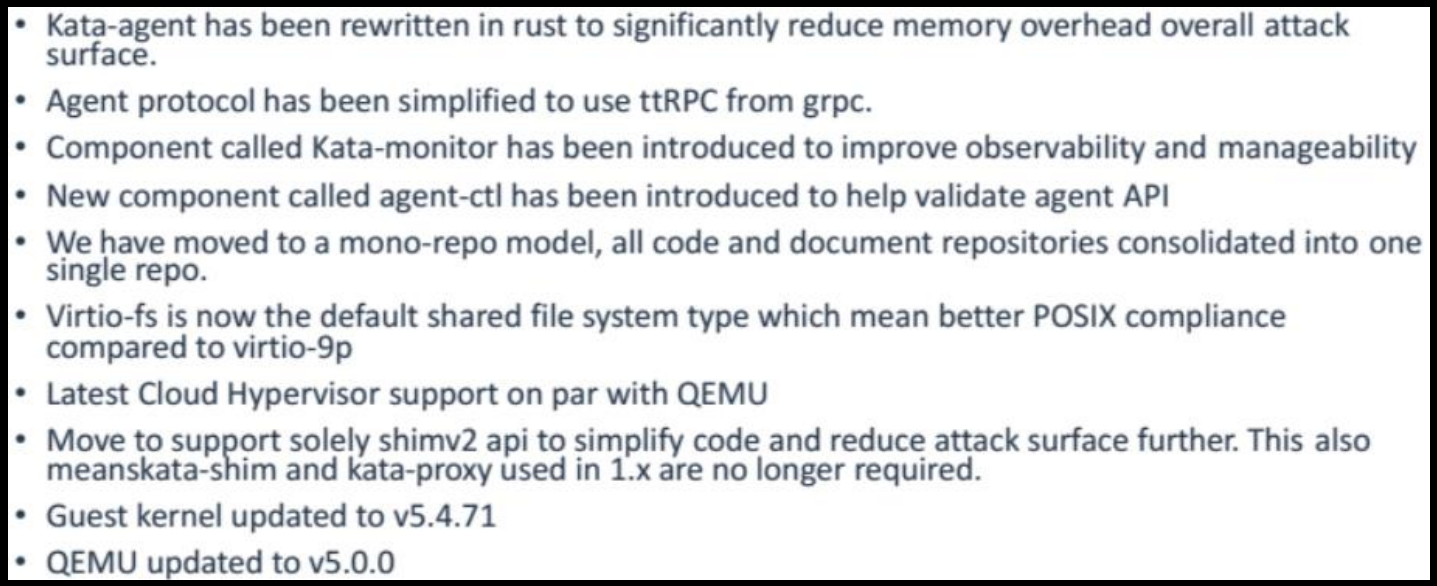


Cloud Native

- Kata Containers runtime is OCI-compatible, CRI-O-compatible, and Containerd-compatible(CNI, CSI…), allowing it to work seamlessly with both Docker and Kubernetes respectively
- https://github.com/kata-containers/documentation/blob/master/how-to/how-to-use-k8s-with-cri-containerd-and-kata.md
-

2.x

- https://github.com/kata-containers/kata-containers/releases/tag/2.0.0
- https://medium.com/kata-containers/kata-containers-version-2-0-e45df4dd328
- What's New



Source: https://openanolis-downloads.oss-cn-beijing.aliyuncs.com/meetup/kata-update-2020-11.pdf

- https://www.openstack.org/videos/summits/virtual/Observability-in-Kata-containers-2.0

Security

- https://github.com/kata-containers/documentation/blob/master/Limitations.md
- https://containerjournal.com/topics/container-security/kata-container-security-is-good-but-theres-an-achilles-heel/

- "Escaping Virtualized Containers", Yuval Avrahami, BlackHat USA 2020
- https://www.bugcrowd.com/blog/big-bugs-cve-2020-28914/
- https://github.com/kata-containers/community/tree/master/VMT/KCSA

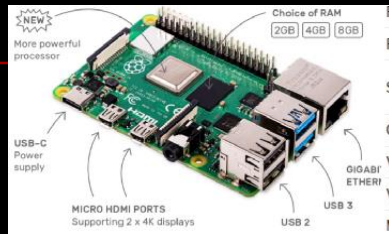
Getting Started

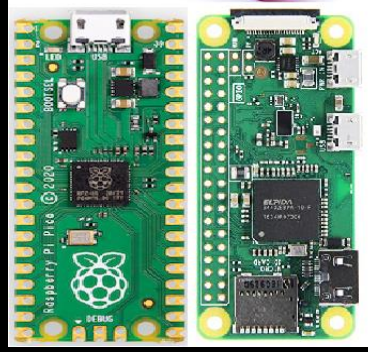
- https://github.com/kata-containers/kata-containers#getting-started
- https://github.com/kata-containers/kata-containers/tree/main/docs
- https://github.com/kata-containers/kata-containers/tree/main/docs/install
- https://github.com/kata-containers/kata-containers/blob/main/docs/install/README.md#packaged-installation-methods
- https://docs.01.org/clearlinux/latest/tutorials/kata.html
- https://github.com/kata-containers/kata-containers/tree/main/docs/install#build-from-source-installation
- https://github.com/kata-containers/documentation/blob/master/how-to/containerd-kata.md

3) Testbed

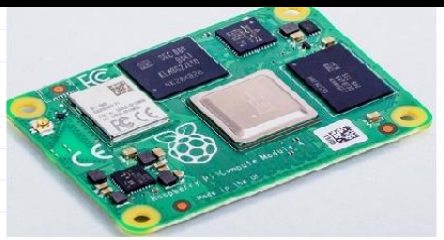
Raspberry Pi

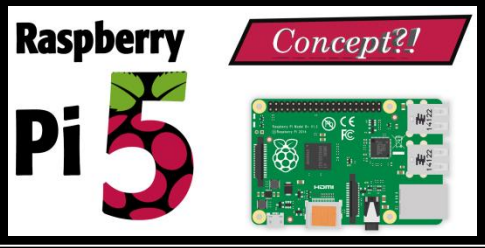
- **https://www.raspberrypi.org/products/raspberry-pi-4-model-b/**





Features/Specs	Raspberry Pi 4B	Raspberry Pi 3 B+
Release date	24th June 2019	14th March 2018
SoC	Broadcom BCM2711 quad-core Cortex-A72 @ 1.5 GHz	Broadcom BCM2837B0 quad-core Cortex-A53 @ 1.4 GHz
GPU	VideoCore VI with OpenGL ES 1.1, 2.0, 3.0	VideoCore IV with OpenGL ES 1.1, 2.0
Video Decode	H.265 4Kp60, H.264 1080p60	H.264 & MPEG-4 1080p30
Video Encode	H.264 1080p30	
Memory	1GB, 2GB, or 4GB LPDDR4	1GB LPDDR2
Storage	microSD card	
Video & Audio Output	2x micro HDMI ports up to 4Kp60 3.5mm AV port (composite + audio) MIPI DSI connector	1x HDMI 1.4 port up to 1080p60 3.5mm AV port (composite + audio) MIPI DSI connector
Camera	MIPI CSI connector	
Ethernet	Native Gigabit Ethernet	Gigabit Ethernet over USB (300 Mbps max.)
WiFi	Dual band 802.11 b/g/n/ac	
Bluetooth	Bluetooth 5.0 + BLE	Bluetooth 4.2 + BLE
USB	2x USB 3.0 + 2x USB 2.0	4x USB 2.0
Expansion	40-pin GPIO header	
Power Supply	5V via USB type-C up to 3A 5V via GPIO header up to 3A Power over Ethernet via PoE HAT	5V via micro USB up to 2.5A 5V via GPIO header up to 3A Power over Ethernet via PoE HAT
Dimensions	85×56 mm	
Default OS	Raspbian (after June 24, 2019)	Raspbian (after March 2018)
Price	$35 (1GB RAM), $45 (2GB RAM), $55 (4GB RAM)	$35 (1GB RAM)



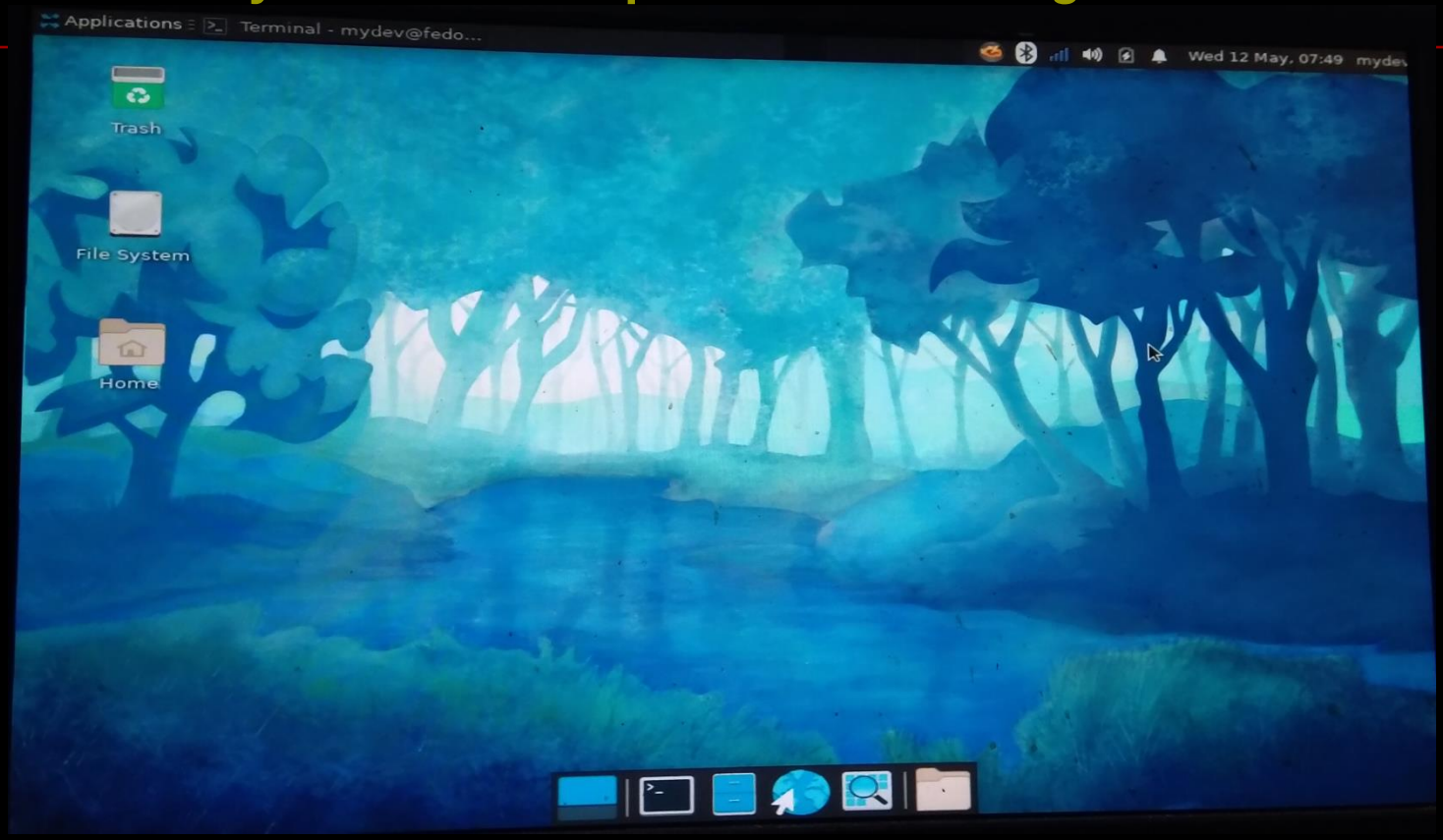


3.1 Fedora

- a Linux distribution developed by the community-supported Fedora Project which is sponsored primarily by Red Hat
- https://en.wikipedia.org/wiki/Fedora_(operating_system)
- https://getfedora.org/
- https://alt.fedoraproject.org/alt/
- https://spins.fedoraproject.org/
- https://fedoraproject.org/wiki/Architectures/ARM
- https://fedoramagazine.org/
- https://silverblue.fedoraproject.org/
- Developer friendly!

Fedora 34 AArch64 XFCE

- https://fedoramagazine.org/announcing-fedora-34/
- https://archives.fedoraproject.org/pub/fedora-secondary/releases/34/Spins/aarch64/images/




```
[mydev@fedora ~]$ uname -a
Linux fedora 5.11.20-300.fc34.aarch64 #1 SMP Wed May 12 12:27:11 UTC 2021 aarch64 aarch64 aarch64 GNU/Linux
```

```
[mydev@fedora ~]$ cat /boot/config-5.11.20-300.fc34.aarch64 |grep -i kvm
CONFIG_KVM=y
CONFIG_HAVE_KVM_IRQCHIP=y
CONFIG_HAVE_KVM_IRQFD=y
CONFIG_HAVE_KVM_IRQ_ROUTING=y
CONFIG_HAVE_KVM_EVENTFD=y
CONFIG_KVM_MMIO=y
CONFIG_HAVE_KVM_MSI=y
CONFIG_HAVE_KVM_CPU_RELAX_INTERCEPT=y
CONFIG_KVM_VFIO=y
CONFIG_HAVE_KVM_ARCH_TLB_FLUSH_ALL=y
CONFIG_KVM_GENERIC_DIRTYLOG_READ_PROTECT=y
CONFIG_HAVE_KVM_IRQ_BYPASS=y
CONFIG_HAVE_KVM_VCPU_RUN_PID_CHANGE=y
[mydev@fedora ~]$
[mydev@fedora ~]$ cat /boot/config-5.11.20-300.fc34.aarch64 |grep -i virtual
CONFIG_VIRTUALIZATION=y
# Virtual GPIO drivers
# end of Virtual GPIO drivers
CONFIG_REGULATOR_VIRTUAL_CONSUMER=m
CONFIG_FB_VIRTUAL=m
CONFIG_DMA_VIRTUAL_CHANNELS=y
CONFIG_ARCH_HAS_DEBUG_VIRTUAL=y
# CONFIG_DEBUG_VIRTUAL is not set
[mydev@fedora ~]$
```

```
[mydev@fedora ~]$ cat /boot/config-5.11.20-300.fc34.aarch64 |grep -i bpf
CONFIG_CGROUP_BPF=y
CONFIG_BPF=y
CONFIG_BPF_LSM=y
CONFIG_BPF_SYSCALL=y
CONFIG_ARCH_WANT_DEFAULT_BPF_JIT=y
CONFIG_BPF_JIT_ALWAYS_ON=y
CONFIG_BPF_JIT_DEFAULT_ON=y
CONFIG_BPF_PRELOAD=y
CONFIG_BPF_PRELOAD_UMD=m
CONFIG_IPV6_SEG6_BPF=y
CONFIG_NETFILTER_XT_MATCH_BPF=m
# CONFIG_BPFILTER is not set
CONFIG_NET_CLS_BPF=m
CONFIG_NET_ACT_BPF=m
CONFIG_BPF_JIT=y
CONFIG_BPF_STREAM_PARSER=y
CONFIG_LWTUNNEL_BPF=y
CONFIG_HAVE_EBPF_JIT=y
CONFIG_BPF_LIRC_MODE2=y
CONFIG_BPF_EVENTS=y
# CONFIG_BPF_KPROBE_OVERRIDE is not set
# CONFIG_TEST_BPF is not set
[mydev@fedora ~]$
[mydev@fedora ~]$ cat /boot/config-5.11.20-300.fc34.aarch64 |grep -i cgroup
CONFIG_CGROUPS=y
CONFIG_BLK_CGROUP=y
CONFIG_CGROUP_WRITEBACK=y
CONFIG_CGROUP_SCHED=y
CONFIG_CGROUP_PIDS=y
# CONFIG_CGROUP_RDMA is not set
CONFIG_CGROUP_FREEZER=y
# CONFIG_CGROUP_HUGETLB is not set
CONFIG_CGROUP_DEVICE=y
CONFIG_CGROUP_CPUACCT=y
CONFIG_CGROUP_PERF=y
CONFIG_CGROUP_BPF=y
# CONFIG_CGROUP_DEBUG is not set
CONFIG_SOCK_CGROUP_DATA=y
CONFIG_BLK_CGROUP_RWSTAT=y
CONFIG_BLK_CGROUP_IOLATENCY=y
CONFIG_BLK_CGROUP_IOCOST=y
# CONFIG_BFQ_CGROUP_DEBUG is not set
CONFIG_NETFILTER_XT_MATCH_CGROUP=m
CONFIG_NET_CLS_CGROUP=y
CONFIG_CGROUP_NET_PRIO=y
CONFIG_CGROUP_NET_CLASSID=y
[mydev@fedora ~]$
```

- **Three years ago**

  build Kata Containers 1.0.0 from source on Raspberry Pi 3B+ with Fedora Minimal 28 AArch64 & Go 1.10.3

```
2018-06-19 00:36:39      CLEAN   clean
2018-06-19 00:36:42      GENERATE cli/config-generated.go
2018-06-19 00:36:42      CONFIG  data/kata-collect-data.sh
2018-06-19 00:36:42 kata-runtime - version 1.0.0 (commit 42821b7c0a572bb6b1497e6a2b9a3ad6301c09bb)
2018-06-19 00:36:42
2018-06-19 00:36:42 • architecture:
2018-06-19 00:36:42     Host: aarch64
2018-06-19 00:36:42     golang: arm64
2018-06-19 00:36:42     Build: arm64
2018-06-19 00:36:42
2018-06-19 00:36:42 • golang:
2018-06-19 00:36:42     go version go1.10.3 linux/arm64
2018-06-19 00:36:42
2018-06-19 00:36:42 • Summary:
2018-06-19 00:36:42
2018-06-19 00:36:42     binary install path (DESTTARGET)      : /usr/local/bin/kata-runtime
2018-06-19 00:36:42     config install path (DESTCONFIG)      : /usr/share/defaults/kata-containers/configuration.toml
2018-06-19 00:36:42     alternate config path (DESTSYSCONFIG) : /etc/kata-containers/configuration.toml
2018-06-19 00:36:42     hypervisor path (QEMUPATH)            : /usr/bin/qemu-system-aarch64
2018-06-19 00:36:42     assets path (PKGDATADIR)              : /usr/share/kata-containers
2018-06-19 00:36:42     proxy+shim path (PKGLIBEXECDIR)       : /usr/libexec/kata-containers
2018-06-19 00:36:42
2018-06-19 00:36:42      BUILD   /opt/MyWorkSpace/MyProjs/Virtual/VM-Container/Kata/src/github.com/kata-containers/runtime
/kata-runtime
2018-06-19 00:38:11      CONFIG  cli/config/configuration.toml
```

  Source: “OpenStack on ARM”, Feng Li, OpenInfra Days 2018(Beijing)

- **Nowadays**

```
[mydev@fedora ~]$ dnf search kata*
Last metadata expiration check: 21:12:50 ago on Sun 09 May 2021 09:49:21 AM PDT.
================================================== Name & Summary Matched: kata* ==========
kata-agent.aarch64 : Kata guest agent
kata-containers.aarch64 : Kata Containers version 2.x repository
kata-ksm-throttler.aarch64 : Kata KSM throttling daemon
kata-osbuilder.aarch64 : Kata guest initrd and image build scripts
kata-runtime.aarch64 : Kata runtime to run containers in virtual machines
============================================================ Name Matched: kata* ==============
kata-proxy.aarch64 : Proxy for Kata Containers
kata-shim.aarch64 : A shim running in the host for teh Kata Containers project
[mydev@fedora ~]$
```

Kata Containers 2.x

- sudo dnf install libvirt qemu-kvm oci-kvm-hook virt-install genisoimage libvirt-client libvirt-python3 libvirt-daemon-kvm libvirt-daemon-config-network fakeroot policycoreutils-python-utils libguestfs qemu-user-static kpartx jq podman skopeo buildah wine…
- https://docs.docker.com/engine/install/fedora/
-

```
[mydev@fedora ~]$ which kata-runtime
/usr/bin/kata-runtime
[mydev@fedora ~]$
[mydev@fedora ~]$ kata-runtime check
/usr/share/kata-containers/defaults/configuration.toml: file /usr/bin/qemu-kvm does not exist
[mydev@fedora ~]$
[mydev@fedora ~]$ cd /usr/bin
[mydev@fedora bin]$ sudo ln -sf qemu-system-aarch64 qemu-kvm
[mydev@fedora bin]$
[mydev@fedora bin]$ ls -l qemu-kvm
lrwxrwxrwx. 1 root root 19 May 19 02:15 qemu-kvm -> qemu-system-aarch64
[mydev@fedora bin]$
[mydev@fedora bin]$ kata-runtime check
Invalid command "check"
[mydev@fedora bin]$ kata-runtime kata-check
Newer minor release available: 2.1.0 (url: https://github.com/kata-containers/kata-containers/releases/download/2.1.0/kata-static-2.1.0-x86_64.tar.xz,
 date: 2021-05-14T18:37:59Z)
WARN[0001] modprobe insert module failed: modprobe: ERROR: could not insert 'vhost_net': Operation not permitted  arch=arm64 error="exit status 1" mod
ule=vhost_net name= pid=19893 source=runtime
ERRO[0001] kernel property not found                     arch=arm64 description="Host kernel accelerator for virtio network" name=vhost_net pid=19893
source=runtime type=module
WARN[0001] modprobe insert module failed: modprobe: ERROR: could not insert 'vhost_vsock': Operation not permitted  arch=arm64 error="exit status 1" m
odule=vhost_vsock name= pid=19893 source=runtime
ERRO[0001] kernel property not found                     arch=arm64 description="Host Support for Linux VM Sockets" name=vhost_vsock pid=19893 source=
runtime type=module
WARN[0001] modprobe insert module failed: modprobe: ERROR: could not insert 'vhost': Operation not permitted  arch=arm64 error="exit status 1" module=
vhost name= pid=19893 source=runtime
ERRO[0001] kernel property not found                     arch=arm64 description="Host kernel accelerator for virtio" name=vhost pid=19893 source=runti
me type=module
ERRO[0001] ERROR: System is not capable of running Kata Containers  arch=arm64 name= pid=19893 source=runtime
ERROR: System is not capable of running Kata Containers
[mydev@fedora bin]$
[mydev@fedora bin]$ sudo kata-runtime kata-check
WARN[0000] Not running network checks as super user      arch=arm64 name= pid=20048 source=runtime
System is capable of running Kata Containers
System can currently create Kata Containers
[mydev@fedora bin]$
```

Env

```
[mydev@fedora bin]$ sudo kata-runtime kata-env
[Meta]
  Version = "1.0.25"

[Runtime]
  Debug = false
  Trace = false
  DisableGuestSeccomp = true
  DisableNewNetNs = false
  SandboxCgroupOnly = true
  Path = "/usr/bin/kata-runtime"
  [Runtime.Version]
    OCI = "1.0.1-dev"
    [Runtime.Version.Version]
      Semver = "2.0.3"
      Major = 2
      Minor = 0
      Patch = 3
      Commit = "77e069db5e352a4044554ab8059b70e16b51875a"
  [Runtime.Config]
    Path = "/usr/share/kata-containers/defaults/configuration.toml"

[Hypervisor]
  MachineType = "virt"
  Version = "QEMU emulator version 5.2.0 (qemu-5.2.0-5.fc34.1)\nCopyright (c) 2003-2020 Fabrice Bellard and the QEMU Project developers"
  Path = "/usr/bin/qemu-system-aarch64"
  BlockDeviceDriver = "virtio-scsi"
  EntropySource = "/dev/urandom"
  SharedFS = "virtio-fs"
  VirtioFSDaemon = "/usr/libexec/virtiofsd"
  Msize9p = 8192
  MemorySlots = 10
  PCIeRootPort = 0
  HotplugVFIOOnRootBus = false
  Debug = false
```

3.2 Manjaro

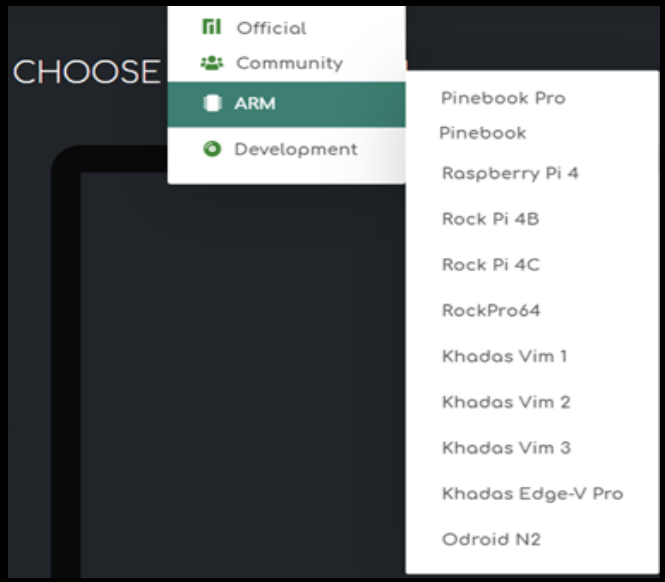
- an open-source Linux distribution based on the Arch Linux operating system
- https://en.wikipedia.org/wiki/Manjaro_Linux
- https://distrowatch.com/

Page Hit Ranking

Data span: Last 6 months Go

Rank	Distribution	HPD*
1	MX Linux	3369▼
2	Manjaro	2490▼
3	Mint	2172▼
4	Pop!_OS	1967▼
5	EndeavourOS	1643▲
6	Ubuntu	1378▼
7	Debian	1281–
8	elementary	1149▼
9	Fedora	1018▼
10	openSUSE	890▲

- https://manjaro.org



```
[mydev@MyRPi4Manjaro2 ~]$ uname -a
Linux MyRPi4Manjaro2 5.12.3-1-MANJARO-ARM #1 SMP PREEMPT Thu May 13 12:58:54 CDT 2021 aarch64 GNU/Linux
[mydev@MyRPi4Manjaro2 ~]$
[mydev@MyRPi4Manjaro2 ~]$ dmesg |grep -i kvm
[    0.739737] kvm [1]: IPA Size Limit: 44 bits
[    0.742958] kvm [1]: vgic interrupt IRQ9
[    0.745142] kvm [1]: Hyp mode initialized successfully
[mydev@MyRPi4Manjaro2 ~]$
[mydev@MyRPi4Manjaro2 ~]$ cat /lib/modules/5.12.3-1-MANJARO-ARM/build/.config |grep -i kvm
CONFIG_KVM=y
CONFIG_HAVE_KVM_IRQCHIP=y
CONFIG_HAVE_KVM_IRQFD=y
CONFIG_HAVE_KVM_IRQ_ROUTING=y
CONFIG_HAVE_KVM_EVENTFD=y
CONFIG_KVM_MMIO=y
CONFIG_HAVE_KVM_MSI=y
CONFIG_HAVE_KVM_CPU_RELAX_INTERCEPT=y
CONFIG_KVM_VFIO=y
CONFIG_HAVE_KVM_ARCH_TLB_FLUSH_ALL=y
CONFIG_KVM_GENERIC_DIRTYLOG_READ_PROTECT=y
CONFIG_HAVE_KVM_IRQ_BYPASS=y
CONFIG_HAVE_KVM_VCPU_RUN_PID_CHANGE=y
[mydev@MyRPi4Manjaro2 ~]$
[mydev@MyRPi4Manjaro2 ~]$ cat /lib/modules/5.12.3-1-MANJARO-ARM/build/.config |grep -i virtual
CONFIG_VIRTUALIZATION=y
# Virtual GPIO drivers
# end of Virtual GPIO drivers
# CONFIG_REGULATOR_VIRTUAL_CONSUMER is not set
# CONFIG_FB_VIRTUAL is not set
CONFIG_DMA_VIRTUAL_CHANNELS=y
CONFIG_ARCH_HAS_DEBUG_VIRTUAL=y
# CONFIG_DEBUG_VIRTUAL is not set
[mydev@MyRPi4Manjaro2 ~]$ _
```

- **my presentation "Python for Linux Kernel Debugging" at PyCon China Hangzhou 2019 contains instructions to build Linux Kernel directly on RPi4 Manjaro**

```
[mydev@MyRPi4Manjaro2 ~]$ cat /lib/modules/5.12.3-1-MANJARO-ARM/build/.config |grep -i bpf
CONFIG_CGROUP_BPF=y
CONFIG_BPF=y
CONFIG_BPF_SYSCALL=y
CONFIG_ARCH_WANT_DEFAULT_BPF_JIT=y
# CONFIG_BPF_PRELOAD is not set
CONFIG_NETFILTER_XT_MATCH_BPF=m
# CONFIG_BPFILTER is not set
# CONFIG_NET_CLS_BPF is not set
# CONFIG_NET_ACT_BPF is not set
# CONFIG_BPF_JIT is not set
# CONFIG_BPF_STREAM_PARSER is not set
CONFIG_LWTUNNEL_BPF=y
CONFIG_HAVE_EBPF_JIT=y
CONFIG_BPF_LIRC_MODE2=y
CONFIG_BPF_EVENTS=y
# CONFIG_BPF_KPROBE_OVERRIDE is not set
# CONFIG_TEST_BPF is not set
[mydev@MyRPi4Manjaro2 ~]$
```

```
[mydev@MyRPi4Manjaro2 ~]$ cat /lib/modules/5.12.3-1-MANJARO-ARM/build/.config |grep -i cgroup
CONFIG_CGROUPS=y
CONFIG_BLK_CGROUP=y
CONFIG_CGROUP_WRITEBACK=y
CONFIG_CGROUP_SCHED=y
CONFIG_CGROUP_PIDS=y
CONFIG_CGROUP_RDMA=y
CONFIG_CGROUP_FREEZER=y
CONFIG_CGROUP_DEVICE=y
CONFIG_CGROUP_CPUACCT=y
CONFIG_CGROUP_PERF=y
CONFIG_CGROUP_BPF=y
# CONFIG_CGROUP_DEBUG is not set
CONFIG_SOCK_CGROUP_DATA=y
CONFIG_BLK_CGROUP_RWSTAT=y
# CONFIG_BLK_CGROUP_IOLATENCY is not set
# CONFIG_BLK_CGROUP_IOCOST is not set
# CONFIG_BFQ_CGROUP_DEBUG is not set
CONFIG_NETFILTER_XT_MATCH_CGROUP=m
CONFIG_NET_CLS_CGROUP=m
CONFIG_CGROUP_NET_PRIO=y
CONFIG_CGROUP_NET_CLASSID=y
[mydev@MyRPi4Manjaro2 ~]$
```

Build Kata Containers

- **https://wiki.archlinux.org/title/Kata_Containers**
- **No available distro packages for Manjaro RPi4, just DIY it**
- **QEMU 6.x is available on Manjaro**

```
[mydev@MyRPi4Manjaro2 ~]$ which qemu-system-aarch64
/usr/bin/qemu-system-aarch64
[mydev@MyRPi4Manjaro2 ~]$
[mydev@MyRPi4Manjaro2 ~]$ /usr/bin/qemu-system-aarch64 -version
QEMU emulator version 6.0.0
Copyright (c) 2003-2021 Fabrice Bellard and the QEMU Project developers
[mydev@MyRPi4Manjaro2 ~]$
```

- **https://github.com/kata-containers/kata-containers/tree/main/docs/install#build-from-source-installation**
- **https://github.com/kata-containers/kata-containers/blob/main/versions.yaml**
- **https://github.com/kata-containers/kata-containers/blob/main/docs/Developer-Guide.md**

```
$ go get -d -u github.com/kata-containers/kata-containers
$ cd $GOPATH/src/github.com/kata-containers/kata-containers/src/runtime
$ make && sudo -E PATH=$PATH make install
```

~150 seconds on my Manjaro RPi4 with the latest Go 1.16.4

build artifacts

```
[mydev@MyRPi4Manjaro2 ~]$ ll /usr/local/bin
total 90196
drwxr-xr-x  3 root root     4096 May 21 20:06 ./
drwxr-xr-x 11 root root     4096 Sep 22  2020 ../
-rwxr-xr-x  1 root root 33401728 May 21 20:06 containerd-shim-kata-v2*
-rwxr-xr-x  1 root root    16678 May 21 20:06 kata-collect-data.sh*
-rwxr-xr-x  1 root root 24705280 May 21 20:06 kata-monitor*
-rwxr-xr-x  1 root root 34208992 May 21 20:06 kata-runtime*
-rwxr-xr-x  1 root root       26 Jan 27 06:49 stop-dmesg*
drwxr-xr-x  2 root root     4096 Apr 18 14:20 tflite/
[mydev@MyRPi4Manjaro2 ~]$
[mydev@MyRPi4Manjaro2 ~]$ file /usr/local/bin/kata-runtime
/usr/local/bin/kata-runtime: ELF 64-bit LSB pie executable, ARM aarch64, version 1 (SYSV), dynamically linked, interpreter /li
b/ld-linux-aarch64.so.1, BuildID[sha1]=a4b985d98f2e24f90fe1c56cc32212bea7ecde2a, for GNU/Linux 3.7.0, not stripped
[mydev@MyRPi4Manjaro2 ~]$
[mydev@MyRPi4Manjaro2 ~]$ ll /usr/share/defaults/kata-containers
total 60
drwxr-xr-x 2 root root  4096 May 21 20:06 ./
drwxr-xr-x 4 root root  4096 May 21 19:26 ../
-rw-r--r-- 1 root root 11106 May 21 20:06 configuration-clh.toml
-rw-r--r-- 1 root root 15614 May 21 20:06 configuration-fc.toml
-rw-r--r-- 1 root root 23174 May 21 20:06 configuration-qemu.toml
lrwxrwxrwx 1 root root    23 May 21 20:06 configuration.toml -> configuration-qemu.toml
[mydev@MyRPi4Manjaro2 ~]$
```

Kata-check failed

```
[mydev@MyRPi4Manjaro2 ~]$ kata-runtime kata-check
ERRO[0000] /usr/share/defaults/kata-containers/configuration-qemu.toml: file /usr/share/kata-containers/vmlinux.container does
 not exist  arch=arm64 name=kata-runtime pid=8982 source=runtime
/usr/share/defaults/kata-containers/configuration-qemu.toml: file /usr/share/kata-containers/vmlinux.container does not exist
[mydev@MyRPi4Manjaro2 ~]$
[mydev@MyRPi4Manjaro2 ~]$ vim /usr/share/defaults/kata-containers/configuration-qemu.toml
```

```
[hypervisor.qemu]
path = "/usr/bin/qemu-system-aarch64"
kernel = "/usr/share/kata-containers/vmlinux.container"
image = "/usr/share/kata-containers/kata-containers.img"
machine_type = "virt"
```

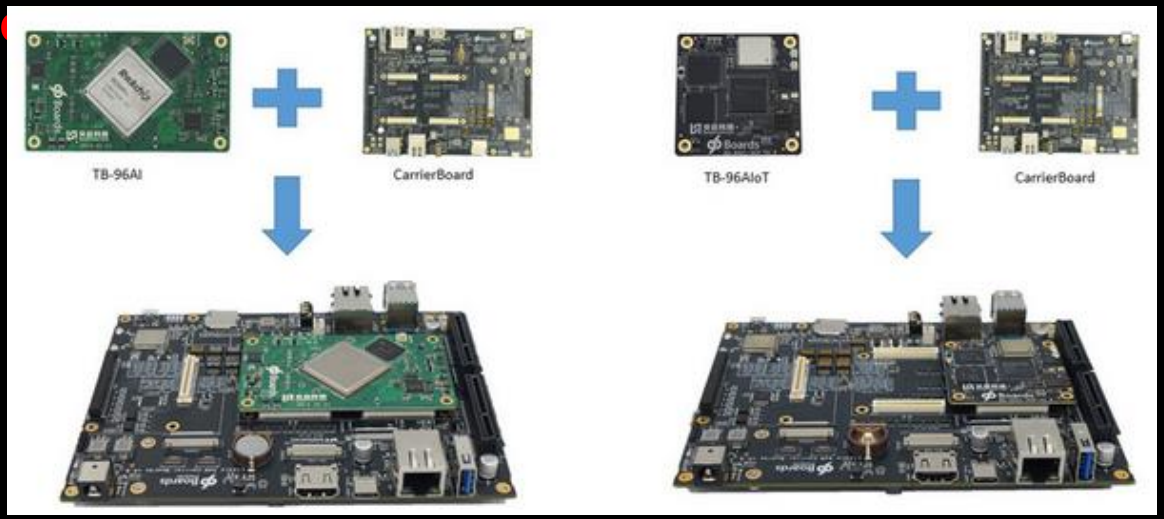
No Kernel and Rootfs for container…

- https://github.com/kata-containers/packaging/tree/master/kernel#build-kata-containers-kernel
  https://github.com/kata-containers/documentation/blob/master/Developer-Guide.md#create-a-rootfs-image
  https://github.com/kata-containers/documentation/blob/master/Developer-Guide.md#create-an-initrd-image---optional
-

3.3 Clustering

SOM/COM

- https://www.linaro.org/news/linaro-announces-launch-of-96boards-system-on-module-som-specification/
- http://linuxgizmos.com/linaro-launches-two-96boards-som-specifications/
- http://static.linaro.org/assets/specifications/96BoardsComputeSoMSpecificationV1.0.pdf
- https://static.linaro.org/assets/specifications/96BoardsWirelessSoMSpecificationV1.0.pdf
-

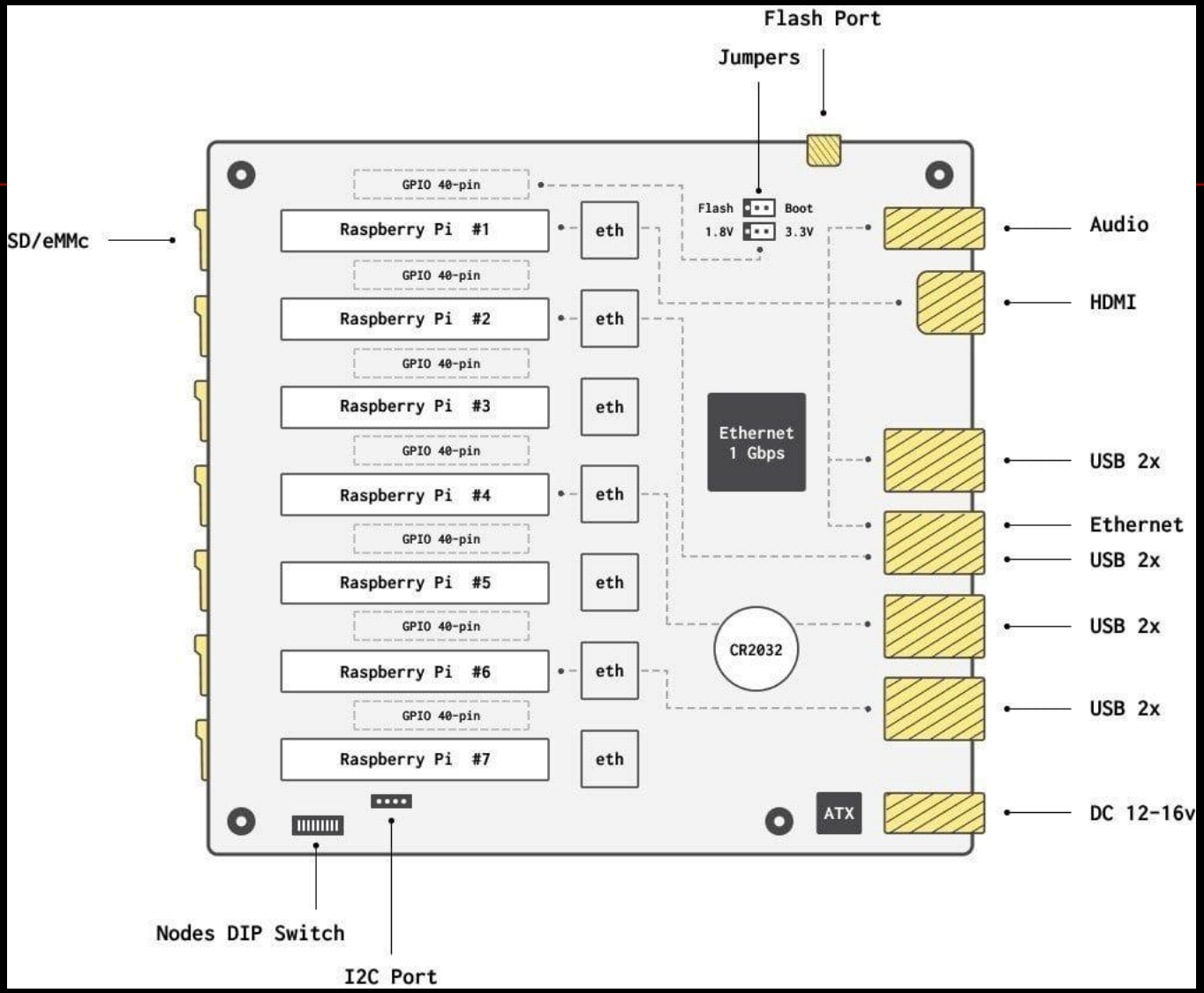


Turing Pi 1

- https://turingpi.com/
- https://docs.turingpi.com/specs
- Support Raspberry Pi Compute Module 1, 3, 3+

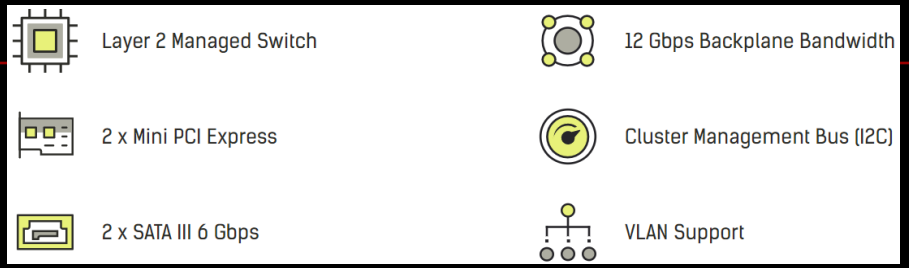
- Host Kubernetes, Containers and Cloud Native Apps locally

Scheme


Flash Port
Jumpers
SD/eMMc
GPIO 40-pin
Raspberry Pi #1
eth
Flash
Boot
1.8V
3.3V
Audio
GPIO 40-pin
Raspberry Pi #2
eth
HDMI
GPIO 40-pin
Raspberry Pi #3
eth
Ethernet
1 Gbps
USB 2x
GPIO 40-pin
Raspberry Pi #4
eth
Ethernet
GPIO 40-pin
Raspberry Pi #5
eth
USB 2x
GPIO 40-pin
Raspberry Pi #6
eth
CR2032
USB 2x
GPIO 40-pin
Raspberry Pi #7
eth
USB 2x
ATX
DC 12-16v
Nodes DIP Switch
I2C Port

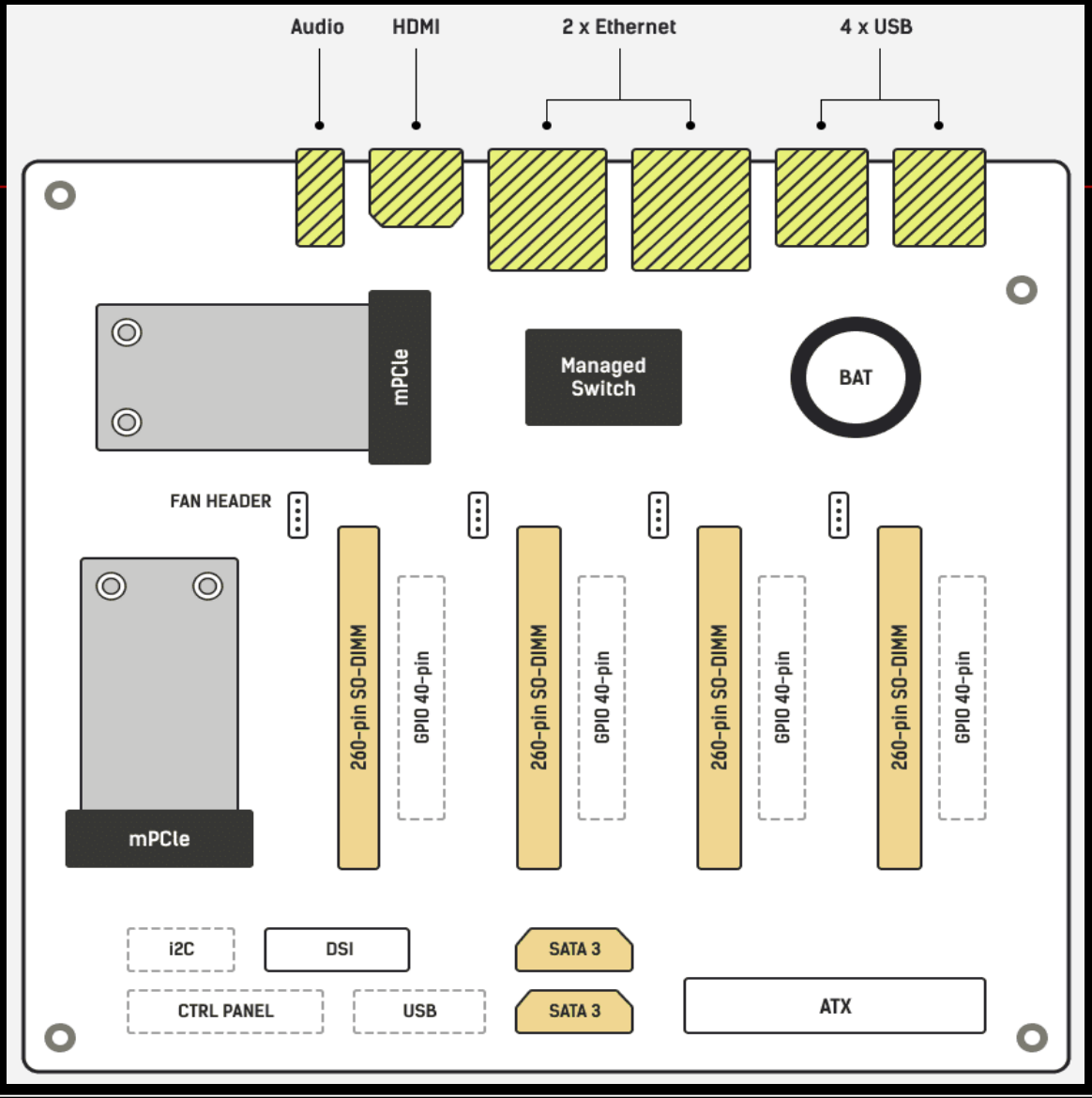
Upcoming Turing Pi 2

- **https://turingpi.com/v2/**
- **https://turingpi.com/turing-pi-2-announcement/**



- **Reliable, Scalable, Cloud-Native Infrastructure for the Edge**

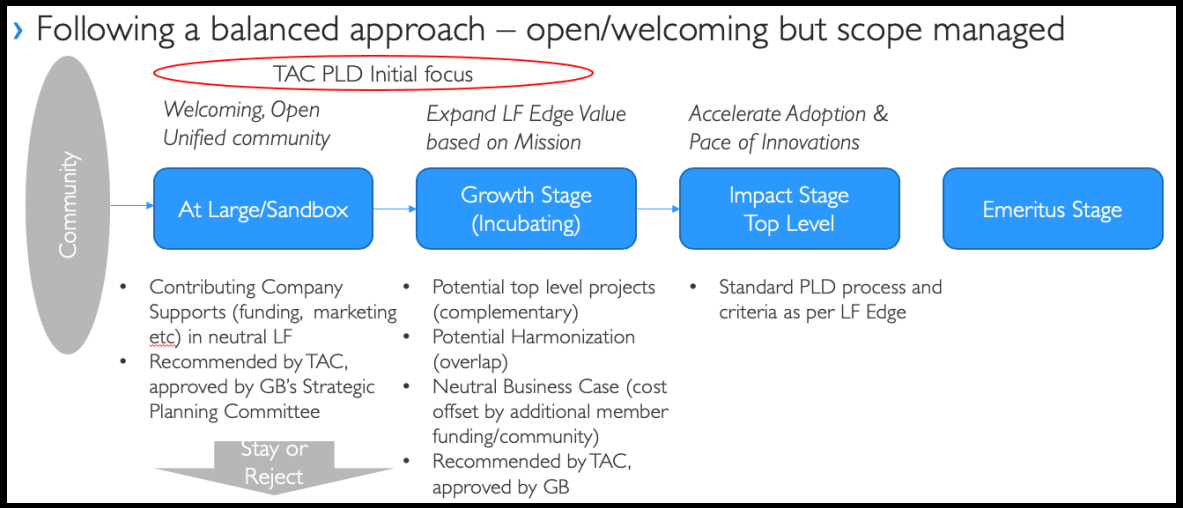
Scheme


Audio
HDMI
2 x Ethernet
4 x USB
mPCIe
Managed Switch
BAT
FAN HEADER
260-pin SO-DIMM
GPIO 40-pin
260-pin SO-DIMM
GPIO 40-pin
260-pin SO-DIMM
GPIO 40-pin
260-pin SO-DIMM
GPIO 40-pin
mPCIe
i2C
DSI
SATA 3
CTRL PANEL
USB
SATA 3
ATX

II. LF Edge

1) Overview

- https://www.lfedge.org/
- IOT + Telecom + Cloud + Enterprise + Industrial
- https://www.linuxfoundation.org/en/press-release/the-linux-foundation-launches-new-lf-edge-to-establish-a-unified-open-source-framework-for-the-edge/
- https://www.lfedge.org/resources/publications/
- https://landscape.lfedge.org/
- https://www.lfedge.org/projects/



IMPACT STAGE

AKRAINO

EDGE X FOUNDRY™

GROWTH STAGE

FLEDGE

HOME EDGE

OPEN HORIZON

STATE OF THE EDGE

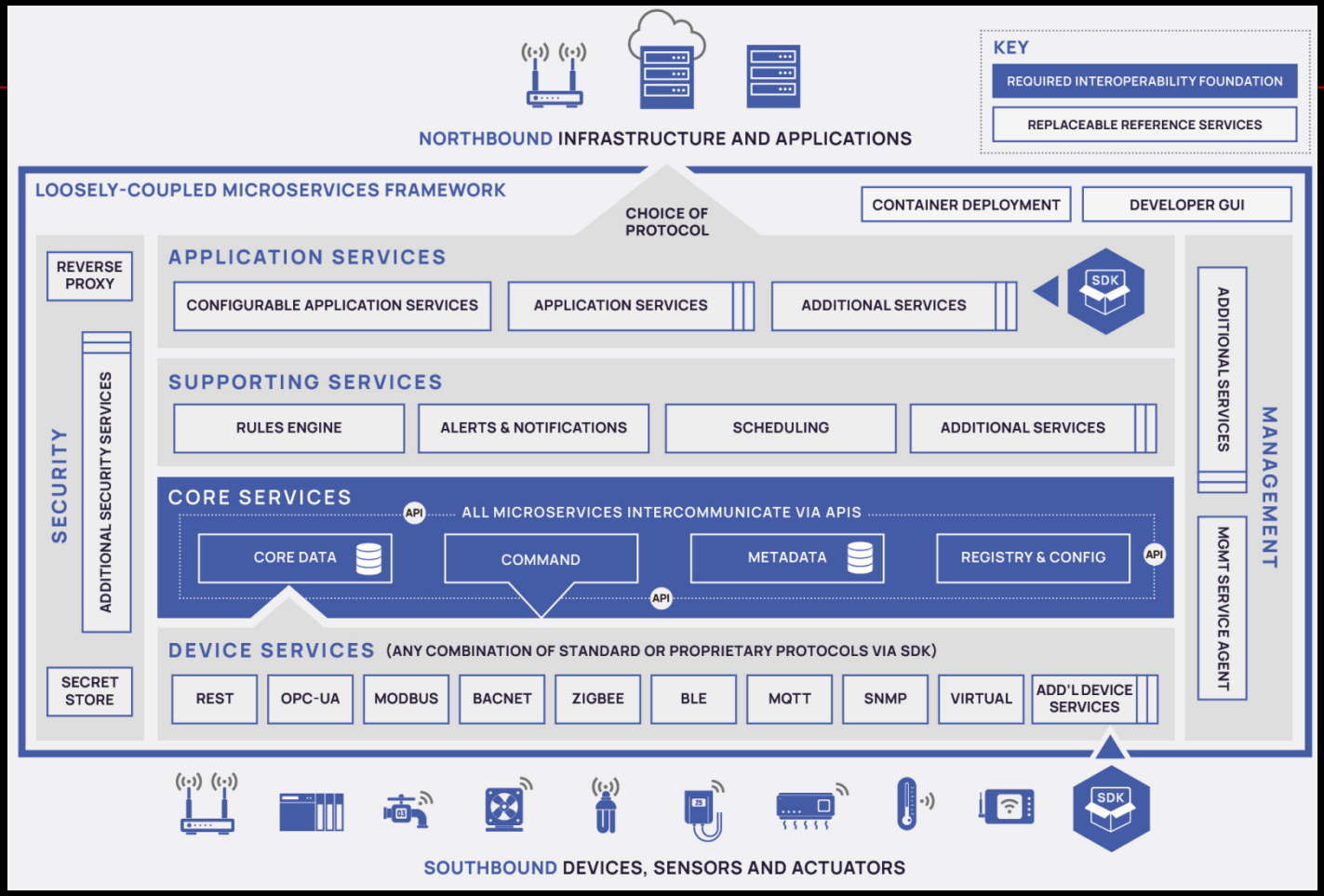
AT LARGE STAGE

BAETYL

SECURE DEVICE ONBOARD

EdgeX

- https://www.lfedge.org/projects/edgexfoundry/
- https://www.edgexfoundry.org/

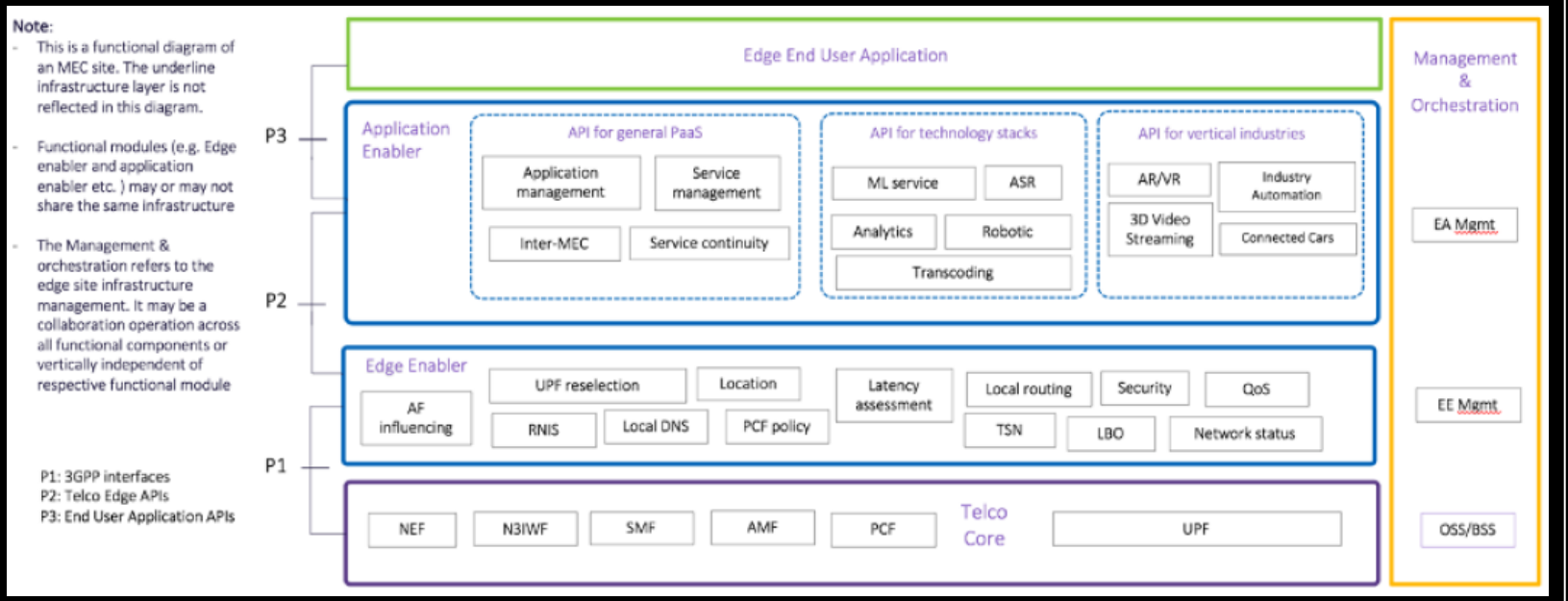


- Migrating from Java to Go since 1.0 release

Akraino

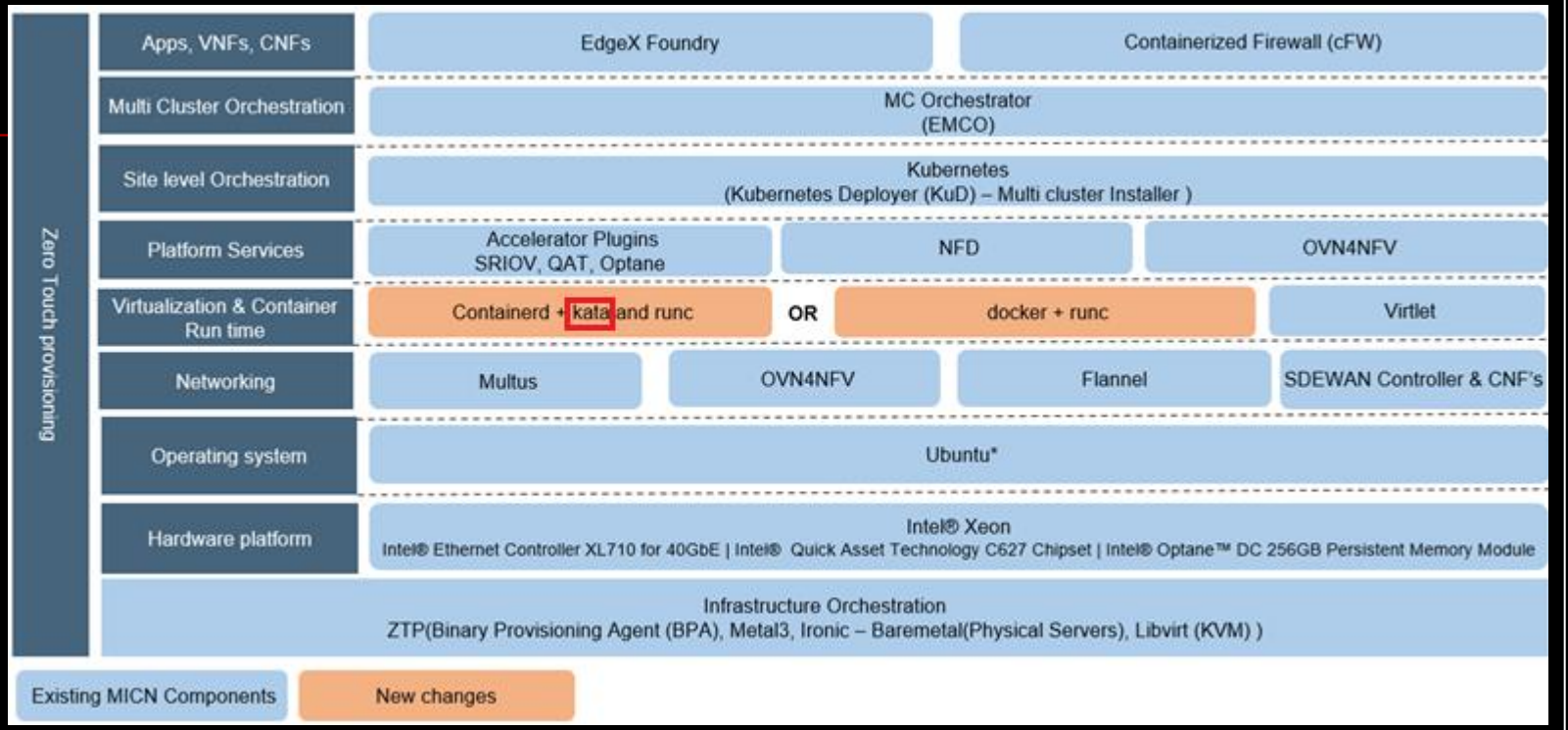
- **https://www.lfedge.org/projects/akraino/**
- **Akraino Edge Stack**
  https://wiki.akraino.org/display/AK/Akraino+Edge+Stack

- **Edge Stack Functional Layers**



- **https://www.lfedge.org/projects/akraino/release-4-2/**

- https://wiki.akraino.org/display/AK/Multi-tenant+Secure+Cloud+Native+Platform+Architecture



2) EVE

- https://www.lfedge.org/projects/eve/
- Key Capabilities

The goal of Project EVE is to enable IoT edge computing deployments with the following capabilities:

- Access to hardware root of trust (e.g. TPM) when deployed on bare metal, supporting functions such as crypto-based ID (no device usernames and passwords), measured boot, remote attestation, signed updates, encryption, etc.
- "Secure by default" deployment profile
- High efficiency and usage of device resources including remote control of CPU, memory, networking and edge device I/O ports
- Hosting of any combination of apps in virtual machines, containers and Kubernetes clusters
- Hosting of any guest operating system deployable in a virtual machine
- Ability to assign CPU cores and co-processing (e.g. GPU) to specific apps
- Ability to block unused I/O ports to prevent physical tampering
- Remote updates of entire software stack with rollback capability to prevent bricking
- Automated patching for security updates
- Automated connectivity to one or more backends (cloud or on premises)
- Distributed firewall to securely route data over networks per policy

- Project EVE is building EVE-OS, a universal, open Linux-based operating system for distributed edge computing. EVE-OS aims to do for the distributed edge what Android did for mobile by creating an open foundation that simplifies development, orchestration and security of edge computing nodes deployed on-prem and in the field. Supporting Docker containers, Kubernetes clusters and virtual machines, EVE-OS provides a flexible foundation for distributed edge deployments with choice of any hardware, application and cloud.

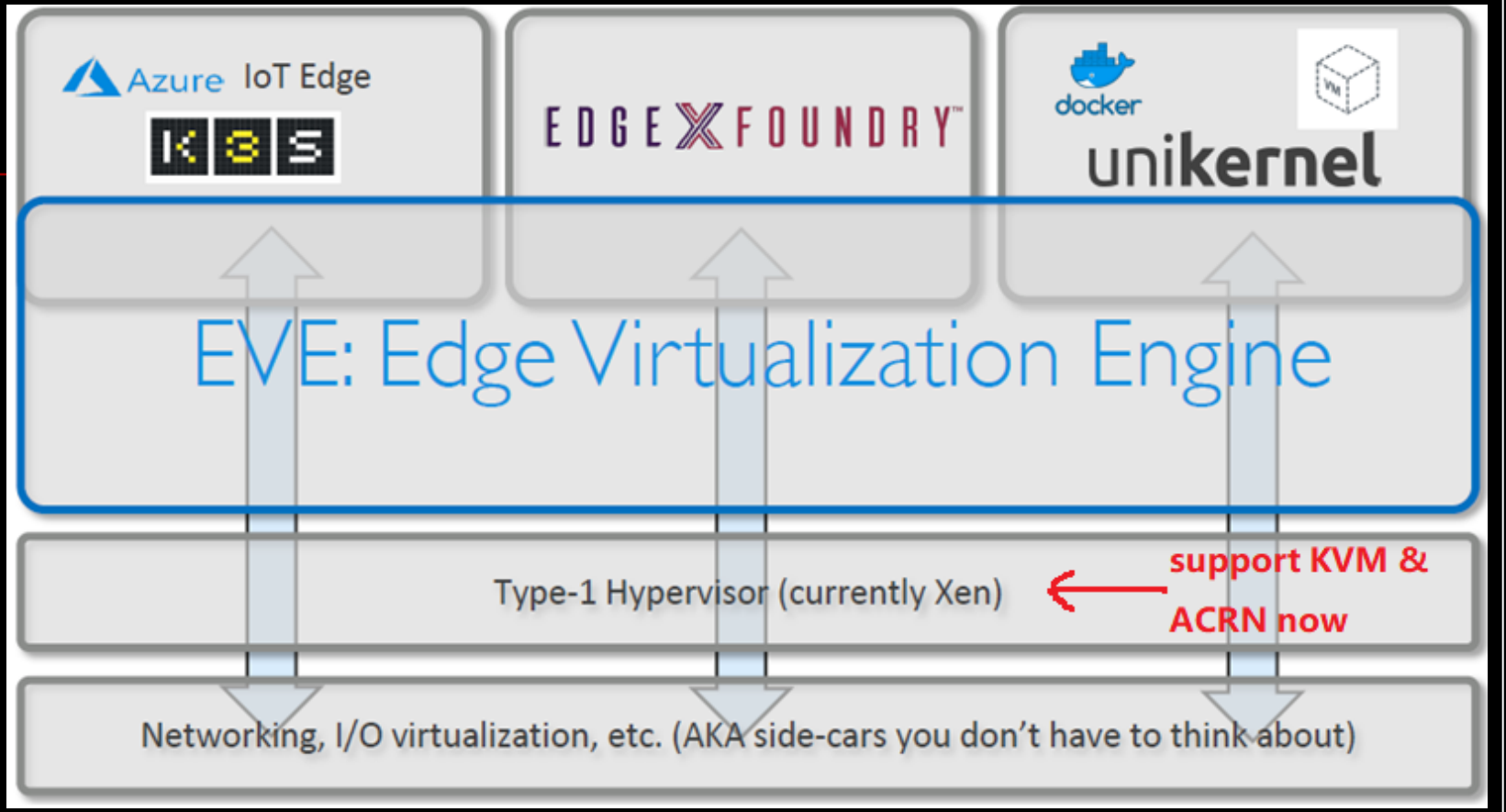
  EVE-OS can be deployed on any bare metal hardware (e.g. x86, Arm, GPU) or within a VM to provide consistent system and orchestration services and provides the ability to run applications in a variety of formats. Support for VMs enables users to continue to use existing software investments while building new containerized innovations in parallel. Compared to agent-based edge management solutions, the bare metal EVE-OS eliminates the possibility of bricking a device in the field during an update, requiring an expensive truck roll.

- https://github.com/lf-edge/eve

EVE supports both ARM and Intel architectures and requires hardware-assisted virtualization. While EVE can run on a board as small as a $20 Orange Pi, the sweet spot for its deployment are IoT Gateways and Industrial PCs.

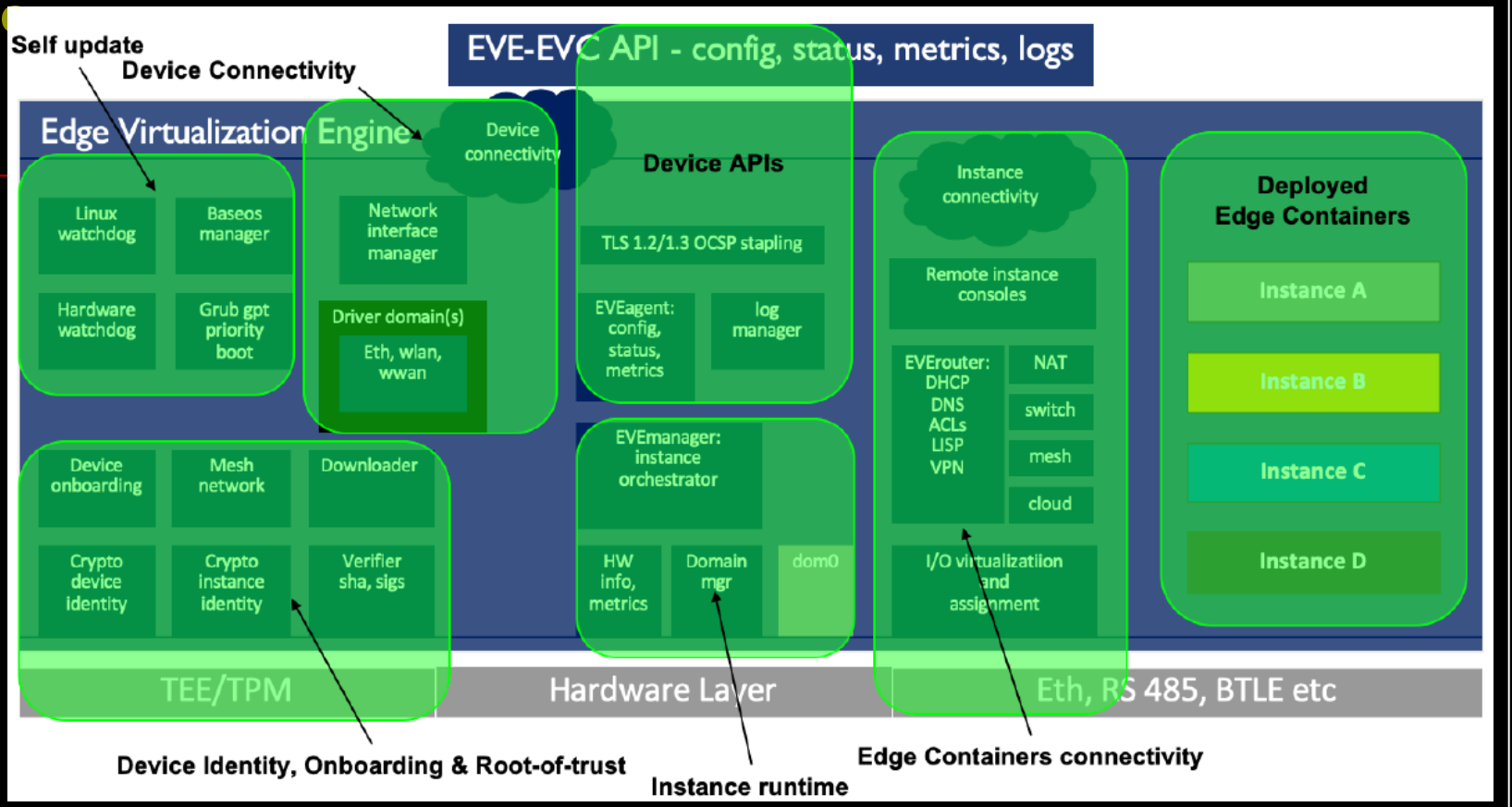
To get its job done, EVE leverages a lot of great open source projects: Xen Project, Linuxkit and Alpine Linux just to name a few. All of that functionality is being orchestrated by the Go microservices available under pkg/pillar. Why pillar? Well, because pillar is the kind of a monolith we need to break out into true, individual microservices under pkg/.

Architecture



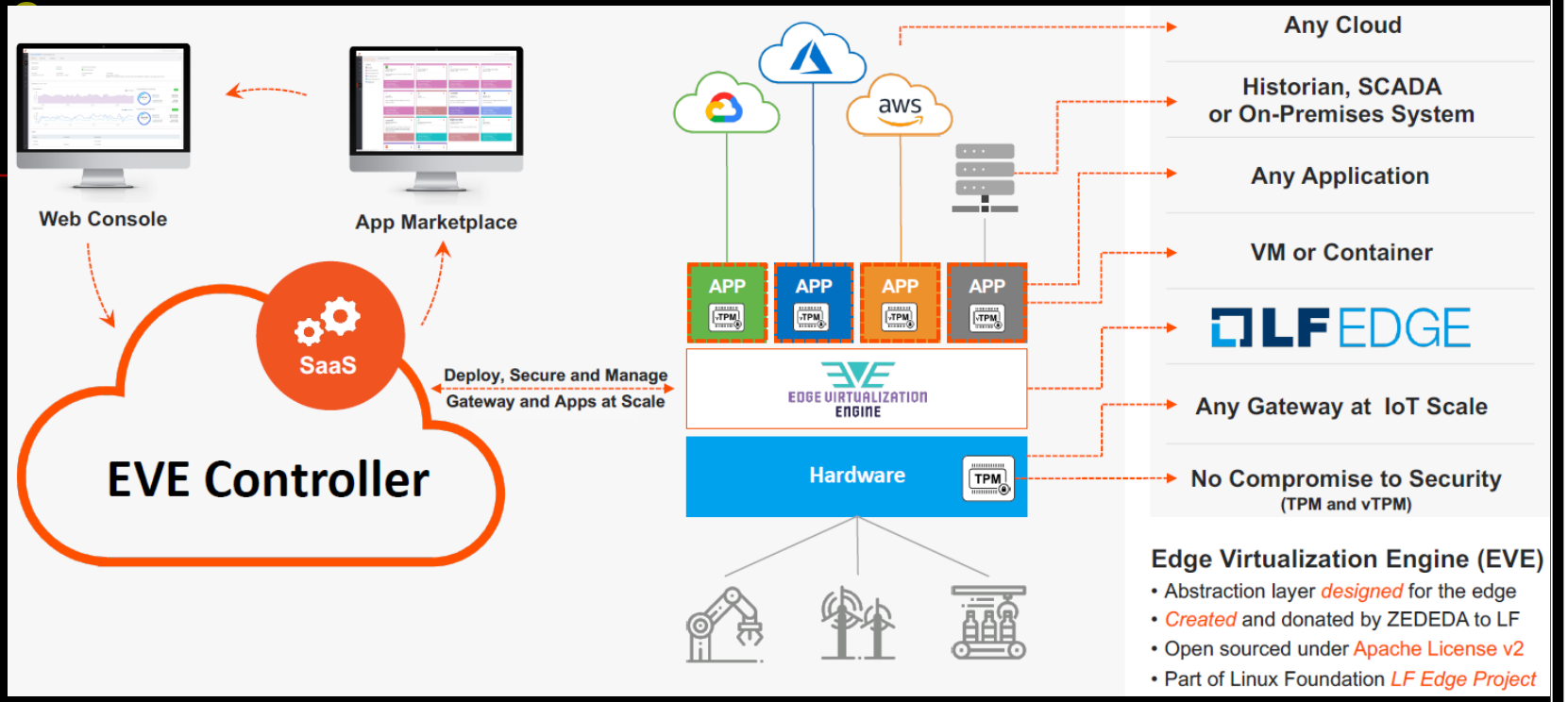
Source: "Linux Foundation's Project EVE: a Cloud-Native Edge Computing Platform", Roman Shaposhnik, QConSF2019

Internals



Source: "Linux Foundation's Project EVE: a Cloud-Native Edge Computing Platform", Roman Shaposhnik, QConSF2019

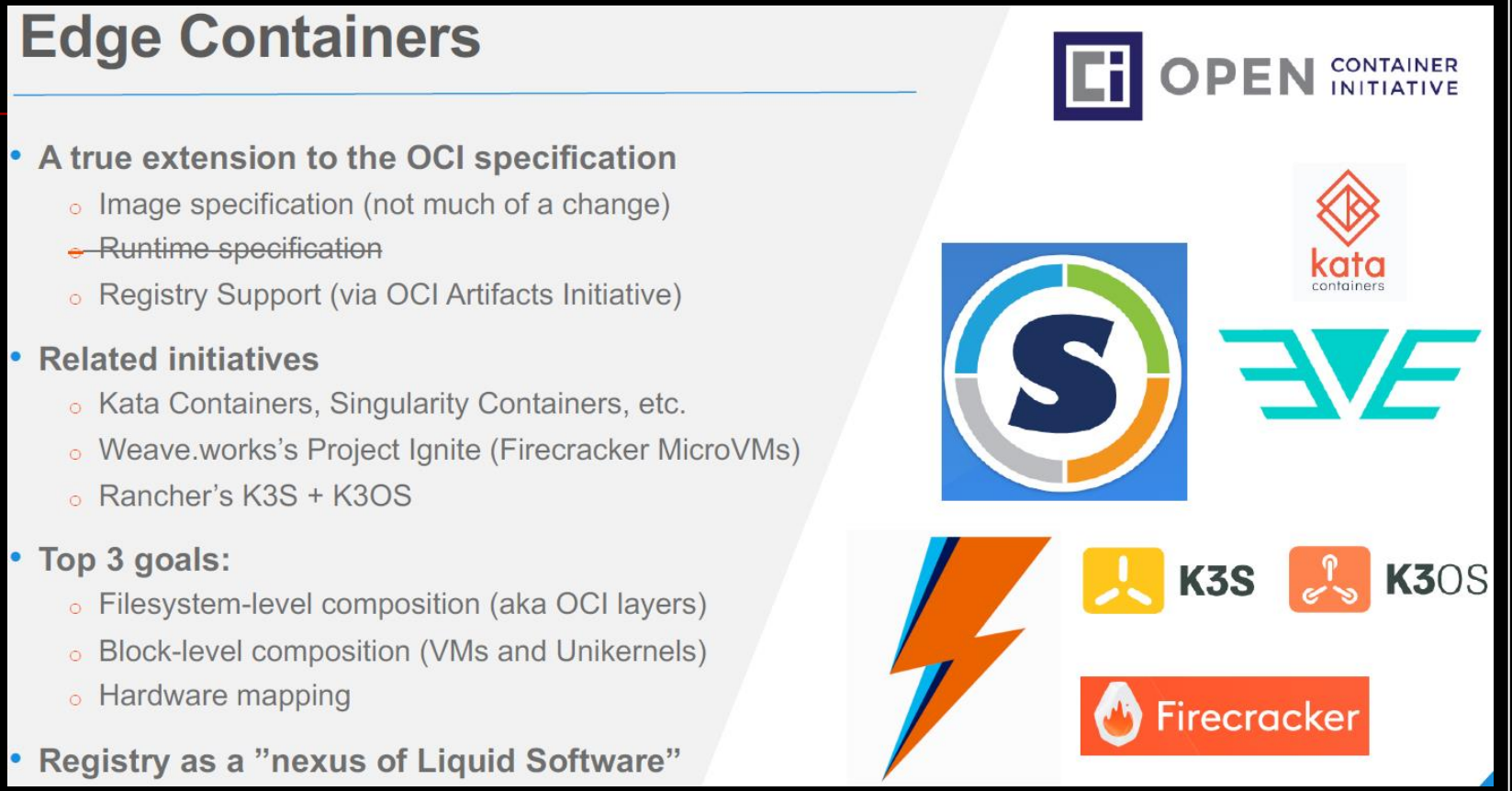
Edge Virtualization



Source: "Linux Foundation's Project EVE: a Cloud-Native Edge Computing Platform", Roman Shaposhnik, QConSF2019

2.1 Edge Containers

- https://github.com/lf-edge/edge-containers
-



Source: "Linux Foundation's Project EVE: a Cloud-Native Edge Computing Platform", Roman Shaposhnik, QConSF2019

2.2 EVE for RPi

- https://github.com/lf-edge/eve/blob/master/Makefile
-

How to use on a Raspberry Pi 4 ARM board

Raspberry Pi 4 is a tiny, but capable enough ARM board that allows EVE to run with either Xen or KVM hypervisors. While EVE would run in the lowest memory configuration (1GB) if you plan to use it for actual EVE development we strongly recommend buying a 4GB RAM option.

Since a full Raspberry Pi 4 support is only available in upstream Linux kernels starting from 5.6.0, you'll have to use that bleeding edge kernel for your build. Another peculiar aspect of this board is that it doesn't use a standard bootloader (e.g. u-boot or UEFI) so we need to trick it into using our own u-boot as UEFI environment. Thankfully, our Makefile logic tries to automate as much of it as possible. Thus, putting it all together, here are the steps to run EVE on Raspberry Pi 4:

1. Make sure you have a clean build directory (since this is a non-standard build) `rm -rf dist/arm64`
2. Build a live image `make ZARCH=arm64 HV=rpi live-raw` (or `make ZARCH=arm64 HV=rpi-kvm live-raw` if you want KVM by default)
3. Flash the `dist/arm64/live.raw` live EVE image onto your SD card by following these instructions

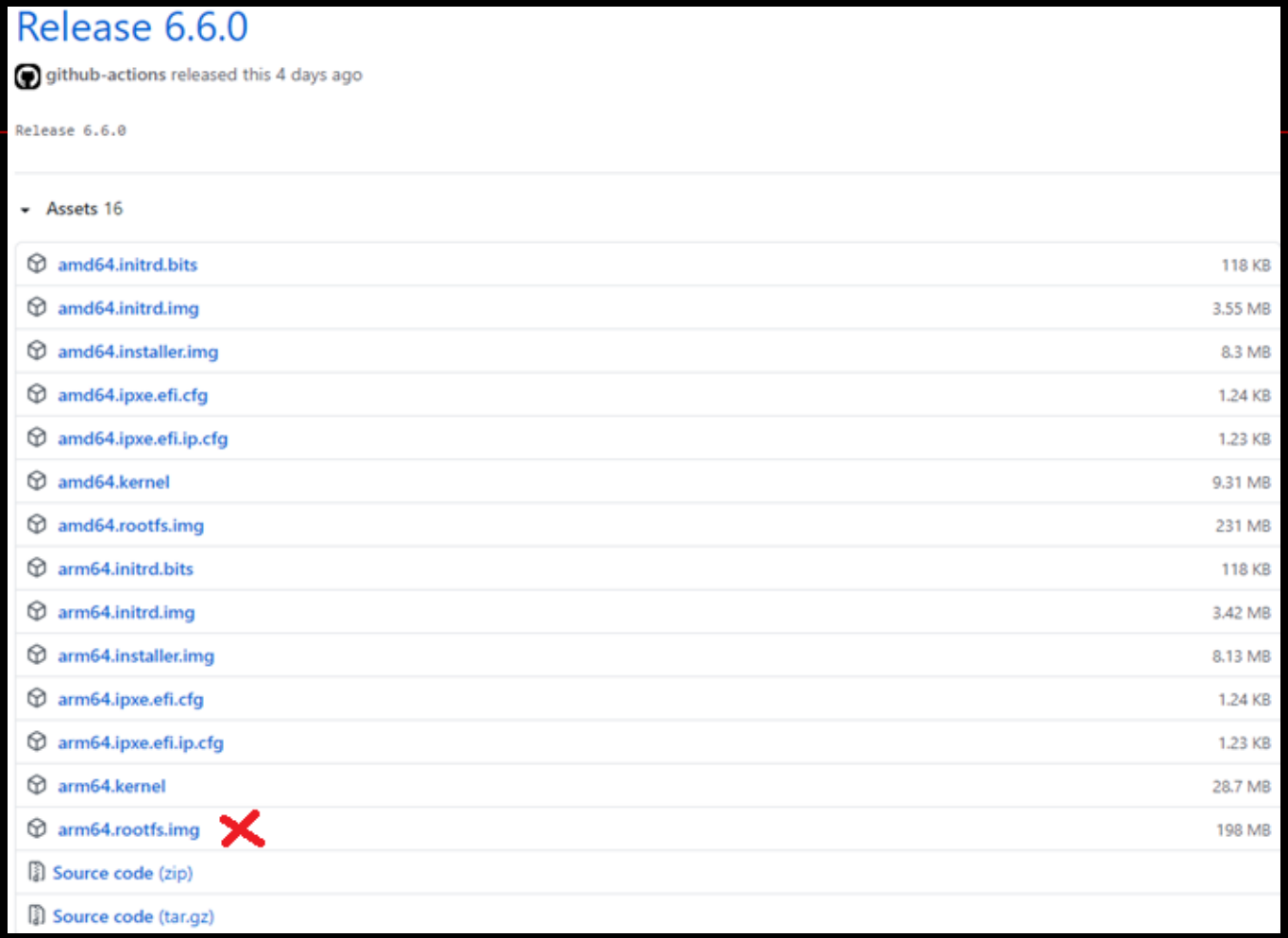
Once your Raspberry Pi 4 is happily running an EVE image you can start using EVE controller for further updates (so that you don't ever have to take an SD card out of your board). Build your rootfs by running `make ZARCH=arm64 rootfs-rpi` (or `make ZARCH=arm64 rootfs-kvm-rpi` if you want KVM by default) and give resulting `dist/arm64/installer/rootfs.img` to the controller.

One final note about Raspberry Pi 4 GPU support is that since we are running EVE in 64bit (aarch64) mode we are still waiting for the proper VC4 DRM BCM2711 drivers to be upstreamed. Currently it is expected that Kernel 5.7 may actually ship the fully functional driver.

- https://github.com/lf-edge/eve/blob/master/Makefile
- https://hub.docker.com/r/lfedge/eve

No prebuilt image for RPi

- **https://github.com/lf-edge/eve/releases**



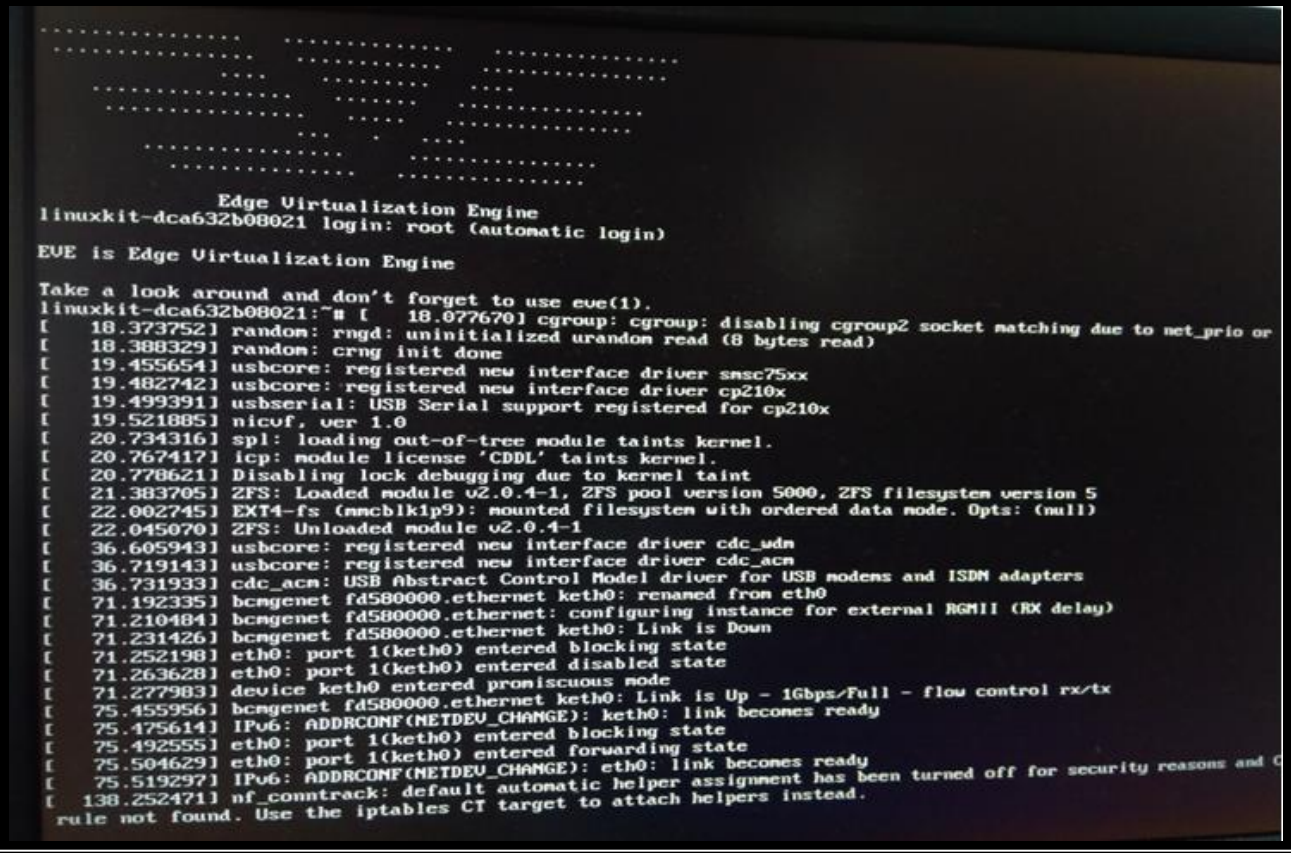
Retrieve live image from container

- **https://www.lfedge.org/2020/10/21/xen-on-raspberry-pi-4-with-project-eve/**

  e.g.:

docker pull lfedge/eve:6.6.0-rpi-xen-arm64

docker run lfedge/eve:6.6.0-rpi-xen-arm64 live > live.raw



2.3 RunX

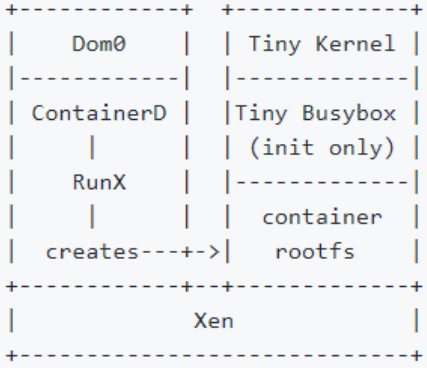
- https://github.com/lf-edge/runx
- an OCI Runtime Spec compliant containers runtime that runs containers as VMs

RunX and KataContainers

Both KataContainers and RunX are containers runtimes that use hypervisors to start containers as virtual machines. However, there are a few key differences.

KataContainers focuses on KVM-based virtual machines. RunX focuses on Xen virtual machines. KataContainers uses an agent running inside each VM, while RunX does not do that by design. RunV (KataContainers' parent) uses libxl to create Xen VMs; thus, it has a build dependency on the Xen Dom0 libraries. RunX doesn't have any build or runtime dependencies on libraries as it invokes the command-line tool `xl`.

Architecture



III. Lightweight Kubernetes

1) K3S

- https://k3s.io/
- Why is it

Perfect for Edge

K3s is a highly available, certified Kubernetes distribution designed for production workloads in unattended, resource-constrained, remote locations or inside IoT appliances.

Simplified & Secure

K3s is packaged as a single <40MB binary that reduces the dependencies and steps needed to install, run and auto-update a production Kubernetes cluster.

Optimized for ARM

Both ARM64 and ARMv7 are supported with binaries and multiarch images available for both. K3s works great from something as small as a Raspberry Pi to an AWS a1.4xlarge 32GiB server.

Additionally, K3s simplifies Kubernetes operations by maintaining functionality for:

- Managing the TLS certificates of Kubernetes componenents
- Managing the connection between worker and server nodes
- Auto-deploying Kubernetes resources from local manifests, in realtime as they are changed.
- Managing an embedded etcd cluster (work in progress)

Features

- https://github.com/k3s-io/k3s/blob/master/README.md

K3s is a fully conformant production-ready Kubernetes distribution with the following changes:

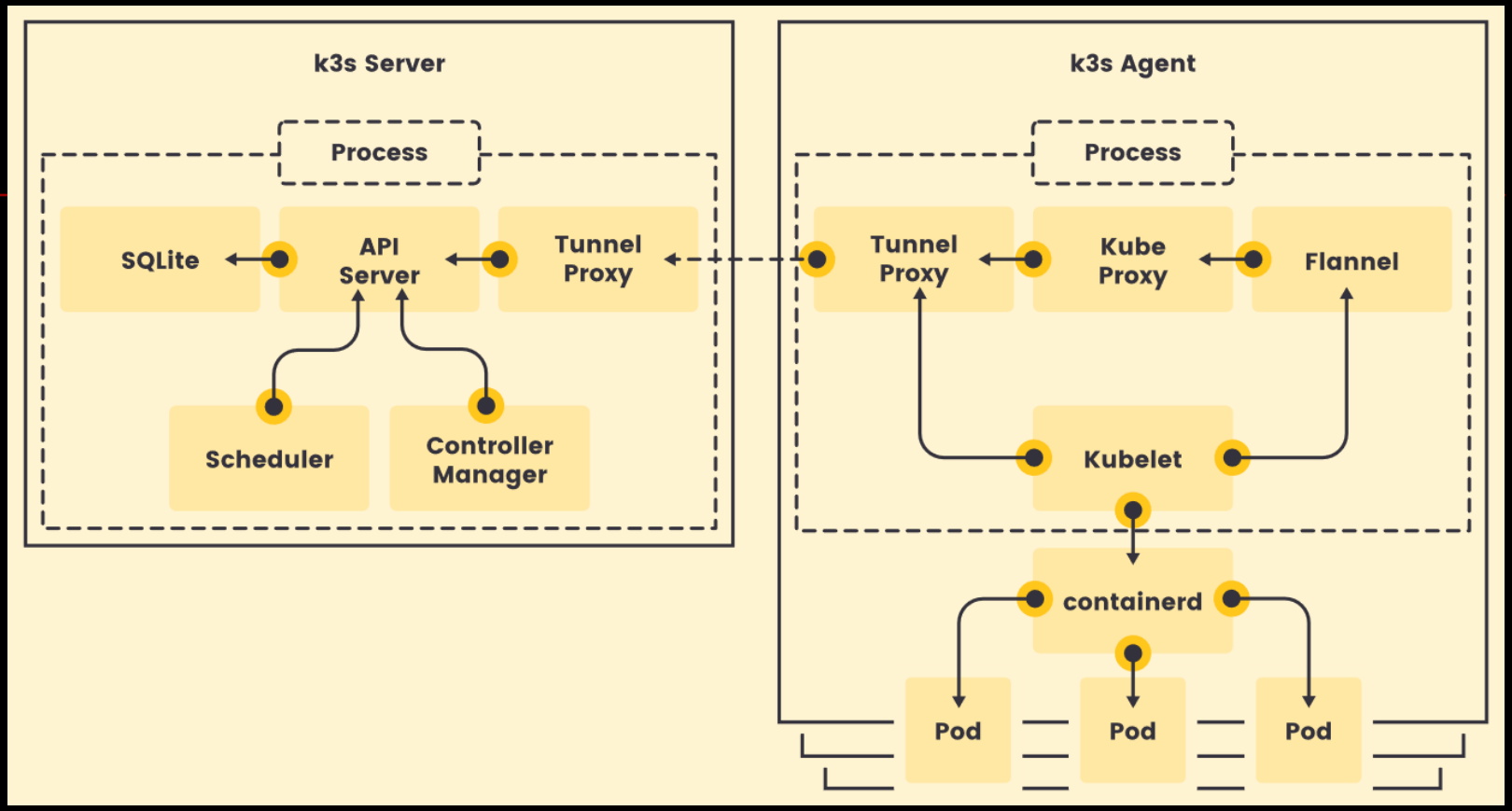
1. It is packaged as a single binary.
2. It adds support for sqlite3 as the default storage backend. Etcd3, MySQL, and Postgres are also supported.
3. It wraps Kubernetes and other components in a single, simple launcher.
4. It is secure by default with reasonable defaults for lightweight environments.
5. It has minimal to no OS dependencies (just a sane kernel and cgroup mounts needed).
6. It eliminates the need to expose a port on Kubernetes worker nodes for the kubelet API by exposing this API to the Kubernetes control plane nodes over a websocket tunnel.

K3s bundles the following technologies together into a single cohesive distribution:

- Containerd & runc
- Flannel for CNI
- CoreDNS
- Metrics Server
- Traefik for ingress
- Klipper-lb as an embedded service loadbalancer provider
- Kube-router for network policy
- Helm-controller to allow for CRD-driven deployment of helm manifests
- Kine as a datastore shim that allows etcd to be replaced with other databases
- Local-path-provisioner for provisioning volumes using local storage
- Host utilities such as iptables/nftables, ebtables, ethtool, & socat

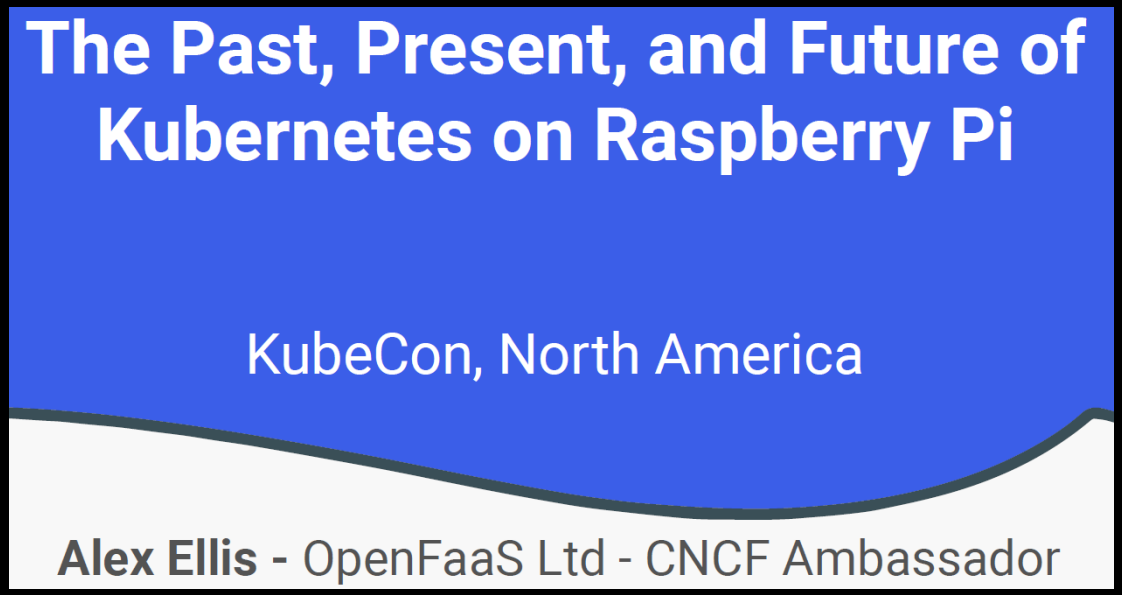
These technologies can be disabled or swapped out for technologies of your choice.

How it works


k3s Server
Process
SQLite
API Server
Tunnel Proxy
Scheduler
Controller Manager
k3s Agent
Process
Tunnel Proxy
Kube Proxy
Flannel
Kubelet
containerd
Pod
Pod
Pod

2) Arkade

- https://github.com/alexellis/arkade
- The Open Source Kubernetes Marketplace
- https://kccncna20.sched.com/event/ekAF/the-past-present-and-future-of-kubernetes-on-raspberry-pi-alex-ellis-openfaas-ltd



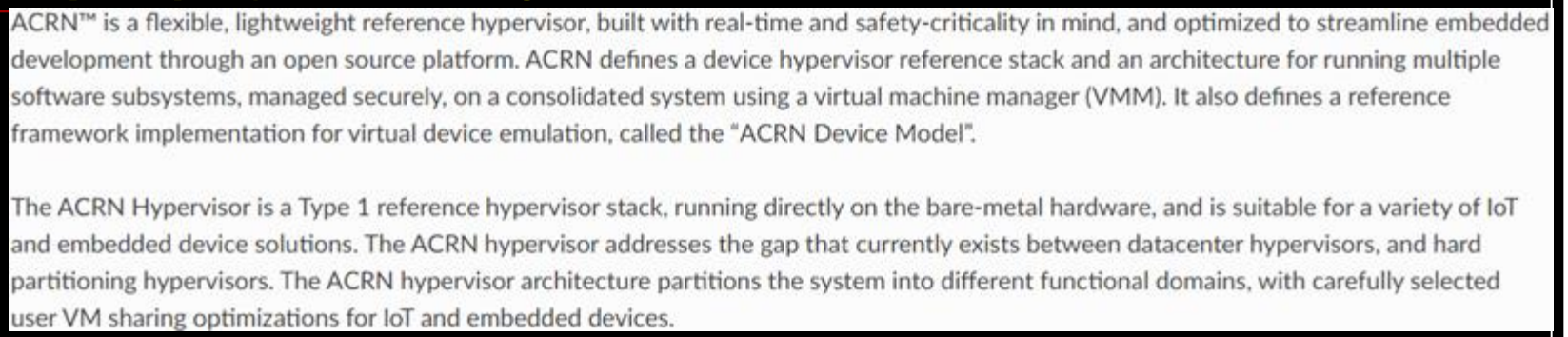
3) Sandboxed Containers Operator

- https://github.com/openshift/sandboxed-containers-operator
- An operator to enhance an Openshift/Kubernetes cluster to support running sandboxed containers

IV. Project ACRN

1) Overview

- https://projectacrn.org/



ACRN™ is a flexible, lightweight reference hypervisor, built with real-time and safety-criticality in mind, and optimized to streamline embedded development through an open source platform. ACRN defines a device hypervisor reference stack and an architecture for running multiple software subsystems, managed securely, on a consolidated system using a virtual machine manager (VMM). It also defines a reference framework implementation for virtual device emulation, called the "ACRN Device Model".

The ACRN Hypervisor is a Type 1 reference hypervisor stack, running directly on the bare-metal hardware, and is suitable for a variety of IoT and embedded device solutions. The ACRN hypervisor addresses the gap that currently exists between datacenter hypervisors, and hard partitioning hypervisors. The ACRN hypervisor architecture partitions the system into different functional domains, with carefully selected user VM sharing optimizations for IoT and embedded devices.

- https://projectacrn.github.io/latest/release_notes/release_notes_2.4.html
- https://projectacrn.org/acrn-hypervisor-service-module-upstreamed-to-linux-kernel/

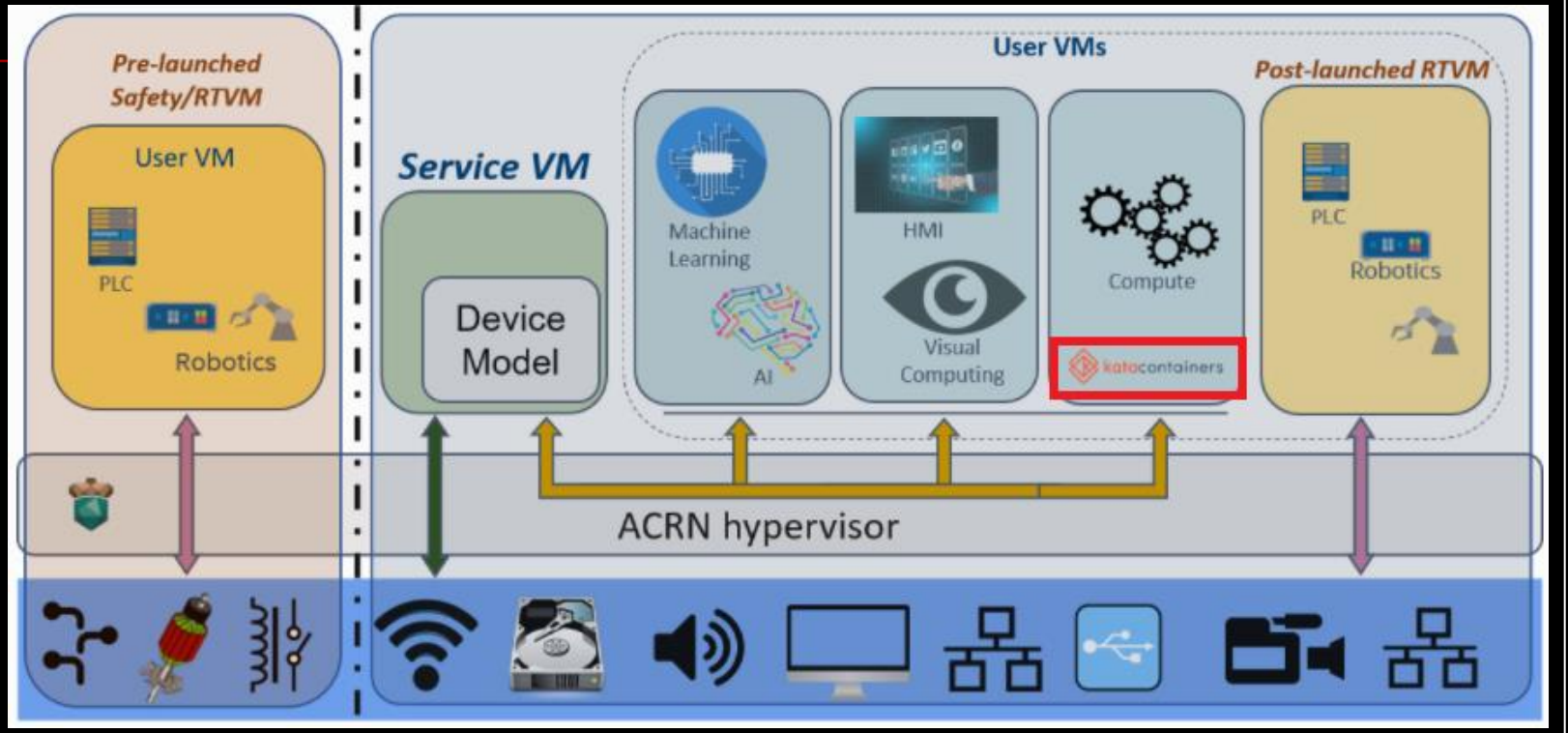
https://lwn.net/Articles/845372/ //HSM driver for ACRN hypervisor

https://www.kernel.org/doc/html/latest/virt/acrn/index.html

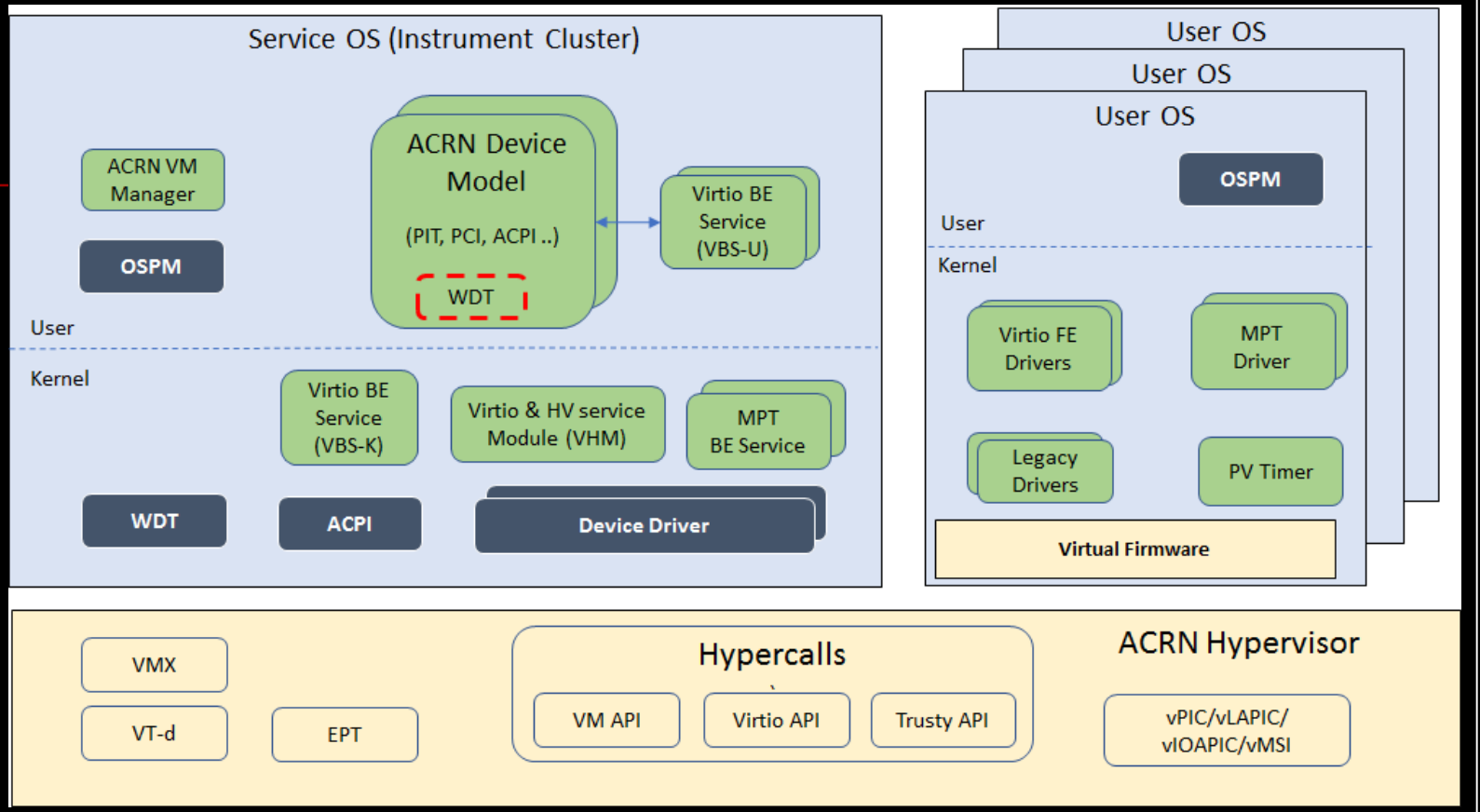
https://kernelnewbies.org/Linux_5.12

Architecture

- https://projectacrn.github.io/latest/introduction/index.html
- High-level Architecture



Hypervisor Architecture



2) X86 only

- https://projectacrn.github.io/latest/reference/hardware.html

Minimum System Requirements for Installing ACRN

Hardware	Minimum Requirements	Recommended
Processor	Compatible x86 64-bit processor	2 core with Intel Hyper-threading Technology enabled in the BIOS or more cores
System memory	4GB RAM	8GB or more (< 32G)
Storage capabilities	20GB	120GB or more

Minimum Requirements for Processor

1 GB Large pages

Known Limitations

Platforms with multiple PCI segments

ACRN assumes the following conditions are satisfied from the Platform BIOS

- All the PCI device BARs should be assigned resources, including SR-IOV VF BARs if a device supports.
- Bridge windows for PCI bridge devices and the resources for root bus, should be programmed with values that enclose resources used by all the downstream devices.
- There should be no conflict in resources among the PCI devices and also between PCI devices and other platform devices.

- https://github.com/kata-containers/kata-containers/blob/main/src/runtime/arch/arm64-options.mk

Run Kata Containers on a Service VM

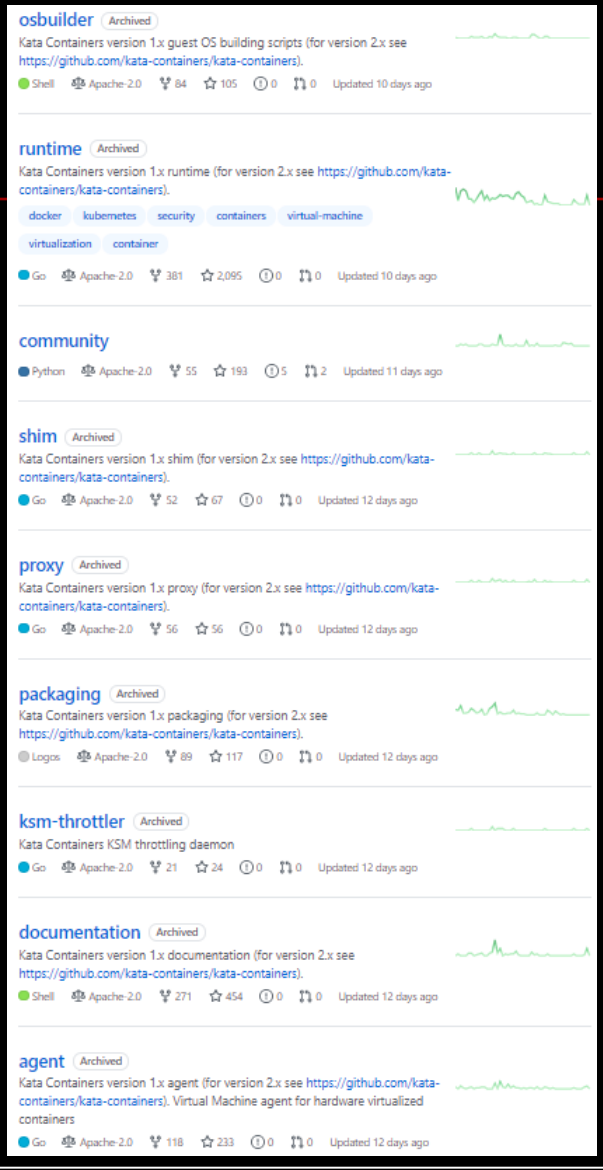
- **https://projectacrn.github.io/latest/tutorials/run_kata_containers.html**

Prerequisites

1. Refer to the ACRN supported hardware.
2. For a default prebuilt ACRN binary in the end-to-end (E2E) package, you must have 4 CPU cores or enable "CPU Hyper-threading" in order to have 4 CPU threads for 2 CPU cores.
3. Follow the Getting Started Guide for ACRN Industry Scenario With Ubuntu Service VM to set up the ACRN Service VM based on Ubuntu.
4. This tutorial is validated on the following configurations:
   - ACRN v2.0 (branch: `release_2.0`)
   - Ubuntu 20.04
5. Kata Containers are supported for ACRN hypervisors configured for the Industry or SDC scenarios.

- **https://github.com/kata-containers/packaging/tree/master/kata-deploy**
- **https://github.com/kata-containers/kata-containers/blob/main/docs/how-to/how-to-use-kata-containers-with-acrn.md**

- so many Kata projects archived recently…



osbuilder Archived
Kata Containers version 1.x guest OS building scripts (for version 2.x see https://github.com/kata-containers/kata-containers).
Shell Apache-2.0 84 105 0 0 Updated 10 days ago

runtime Archived
Kata Containers version 1.x runtime (for version 2.x see https://github.com/kata-containers/kata-containers).
docker kubernetes security containers virtual-machine virtualization container
Go Apache-2.0 381 2,095 0 0 Updated 10 days ago

community
Python Apache-2.0 55 193 5 2 Updated 11 days ago

shim Archived
Kata Containers version 1.x shim (for version 2.x see https://github.com/kata-containers/kata-containers).
Go Apache-2.0 52 67 0 0 Updated 12 days ago

proxy Archived
Kata Containers version 1.x proxy (for version 2.x see https://github.com/kata-containers/kata-containers).
Go Apache-2.0 56 56 0 0 Updated 12 days ago

packaging Archived
Kata Containers version 1.x packaging (for version 2.x see https://github.com/kata-containers/kata-containers).
Logos Apache-2.0 89 117 0 0 Updated 12 days ago

ksm-throttler Archived
Kata Containers KSM throttling daemon
Go Apache-2.0 21 24 0 0 Updated 12 days ago

documentation Archived
Kata Containers version 1.x documentation (for version 2.x see https://github.com/kata-containers/kata-containers).
Shell Apache-2.0 271 454 0 0 Updated 12 days ago

agent Archived
Kata Containers version 1.x agent (for version 2.x see https://github.com/kata-containers/kata-containers). Virtual Machine agent for hardware virtualized containers
Go Apache-2.0 118 233 0 0 Updated 12 days ago

Implementation

- **Kata adaptations to support ACRN**
  - Kata-runtime:
    - Added ACRN hypervisor as a supported hypervisor, so that kata config could pickup ACRN instead of QEMU when launching VM

```
[hypervisor.acrn]
path = "/usr/bin/acrn-dm"
kernel = "/usr/share/kata-containers/vmlinuz.container"
# initrd = "/usr/share/kata-containers/kata-containers-initrd.img"
image = "/usr/share/kata-containers/kata-containers.img"
```

    - Implemented Sandbox Management APIs such as CreateSandBox, FetchSandBox..
      - https://github.com/kata-containers/documentation/blob/master/design/kata-api-design.md#sandbox-management-api
    - Implemented Sandbox Operation APIs such as StartSandbox, StopSanbox, PauseSandBox..
      - https://github.com/kata-containers/documentation/blob/master/design/kata-api-design.md#sandbox-operation-api
    - Implemented hot-plug API (currently have a WA. Working on adding PCI device hot-plug support in ACRN-DM)
      - https://github.com/kata-containers/documentation/blob/master/design/kata-api-design.md#sandbox-hotplug-api
    - Prime the devices, image and kernel as parameters for ACRN-DM when launching the VM.

Source: https://www.slideshare.net/ProjectACRN/acrn-kata-container-on-acrn

ACRN-DM features to support Kata

- Socket Backend support:
  - Kata uses socket communication to talk between kata-runtime <->Kata-proxy, Kata-shim<->Kata-Proxy and Kata-Proxy<->Kata-agent running inside VM.
  - Implemented socket backend for virtio-console device in acrn-dm. (This feature is already merged upstream)
    - -s x,virtio-console,socket:"socket name"="socket path" where x is the PCI slot number.
- PCI device Hot-Plug support:
  - Kata containers does hot plugging of container roofs for both Docker and Kubernetes.
    - Kubernetes when creating a pod, first creates a *pause container and then subsequently creates the application container(s). During the launch of the VM, the roofs for the application container is not known, it needs to be hot-plugged.
  - Looking at both ACPI based hot-plug and PCI/PCIe hot-plug support for PCI device (container rootfs is passed as virtio-blk device). Scoped out initial design and code changes that would be needed in acrn-dm. (WIP)
- Graceful shutdown of VMs:
  - When kata containers finishes its job, expectation is that VM associated with container will be shutdown gracefully. (WIP)

Source: https://www.slideshare.net/ProjectACRN/acrn-kata-container-on-acrn

V. Rethinking Infrastructure for Edge Cloud

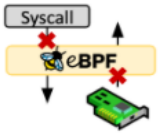
1) Why Linux is critical to Edge Computing

- https://opensource.com/article/21/2/linux-edge-computing

2) eBPF-centric

- https://en.wikipedia.org/wiki/Berkeley_Packet_Filter
- https://www.kernel.org/doc/html/latest/bpf/index.html
- https://ebpf.io/

Security

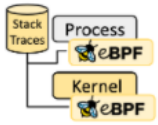


Building on the foundation of seeing and understanding all system calls and combining that with a packet and socket-level view of all networking operations allows for revolutionary new approaches to securing systems. While aspects of system call filtering, network-level filtering, and process context tracing have typically been handled by completely independent systems, eBPF allows for combining the visibility and control of all aspects to create security systems operating on more context with better level of control.

Networking

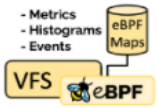
The combination of programmability and efficiency makes eBPF a natural fit for all packet processing requirements of networking solutions. The programmability of eBPF enables adding additional protocol parsers and easily program any forwarding logic to meet changing requirements without ever leaving the packet processing context of the Linux kernel. The efficiency provided by the JIT compiler provides execution performance close to that of natively compiled in-kernel code.

Tracing & Profiling

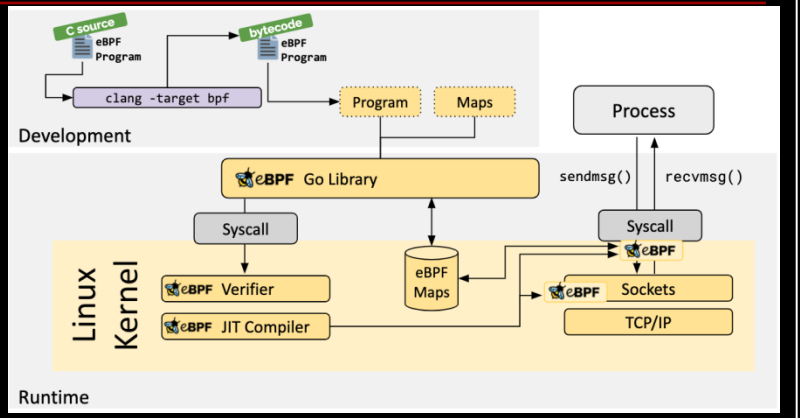


The ability to attach eBPF programs to trace points as well as kernel and user application probe points allows unprecedented visibility into the runtime behavior of applications and the system itself. By giving introspection abilities to both the application and system side, both views can be combined, allowing powerful and unique insights to troubleshoot system performance problems. Advanced statistical data structures allow to extract meaningful visibility data in an efficient manner, without requiring the export of vast amounts of sampling data as typically done by similar systems.

Observability & Monitoring



Instead of relying on static counters and gauges exposed by the operating system, eBPF enables the collection & in-kernel aggregation of custom metrics and generation of visibility events based on a wide range of possible sources. This extends the depth of visibility that can be achieved as well as reduces the overall system overhead significantly by only collecting the visibility data required and by generating histograms and similar data structures at the source of the event instead of relying on the export of samples.



List of BPF features per kernel version

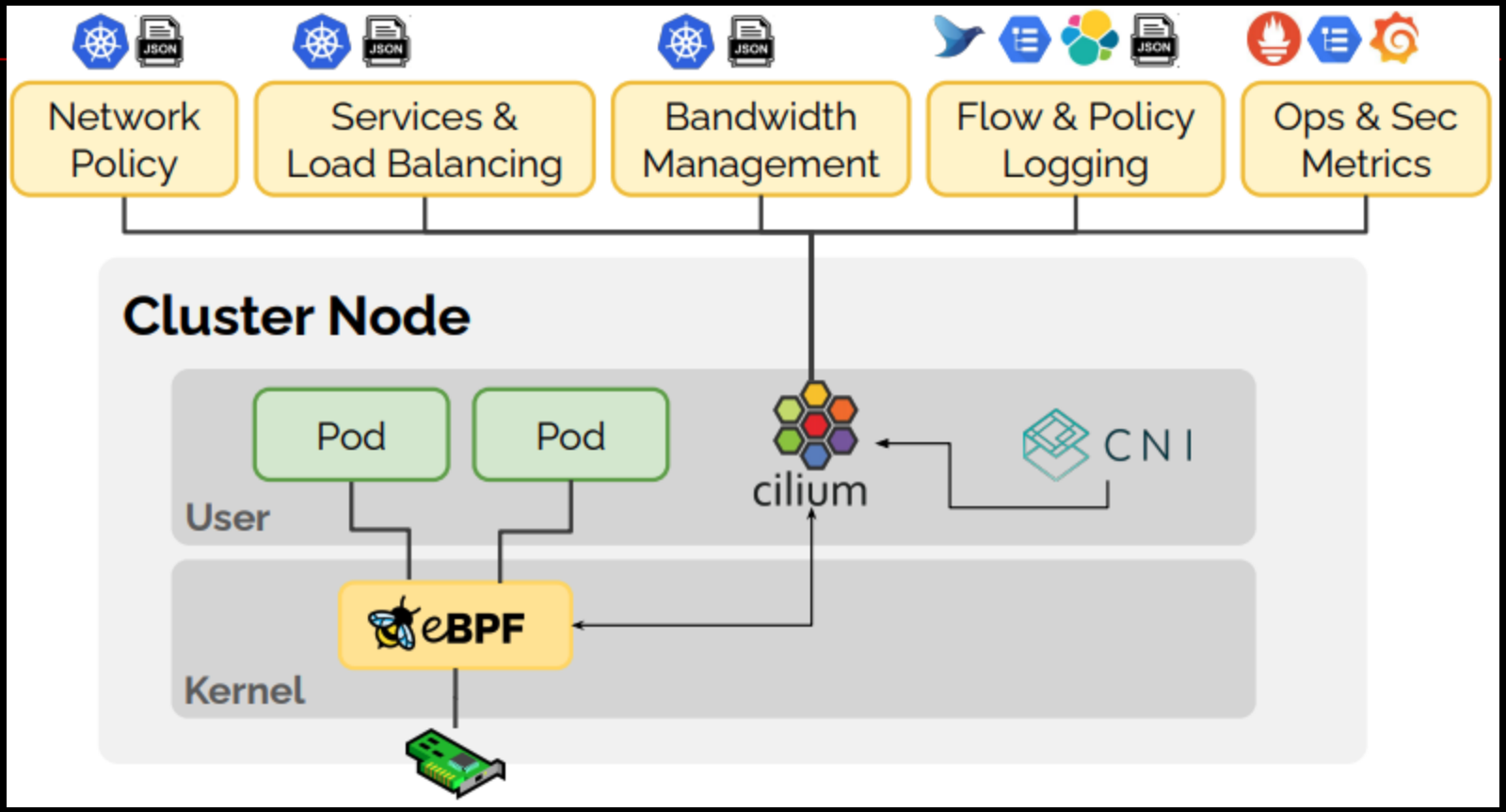
- https://github.com/iovisor/bcc/blob/master/docs/kernel-versions.md
- https://kernelnewbies.org/

2.1 *e*BPF in Kata Containers

- https://github.com/cilium/ebpf
- https://github.com/opencontainers/runc/tree/master/libcontainer/cgroups/ebpf
- https://github.com/vishvananda/netlink
- …

Cilium

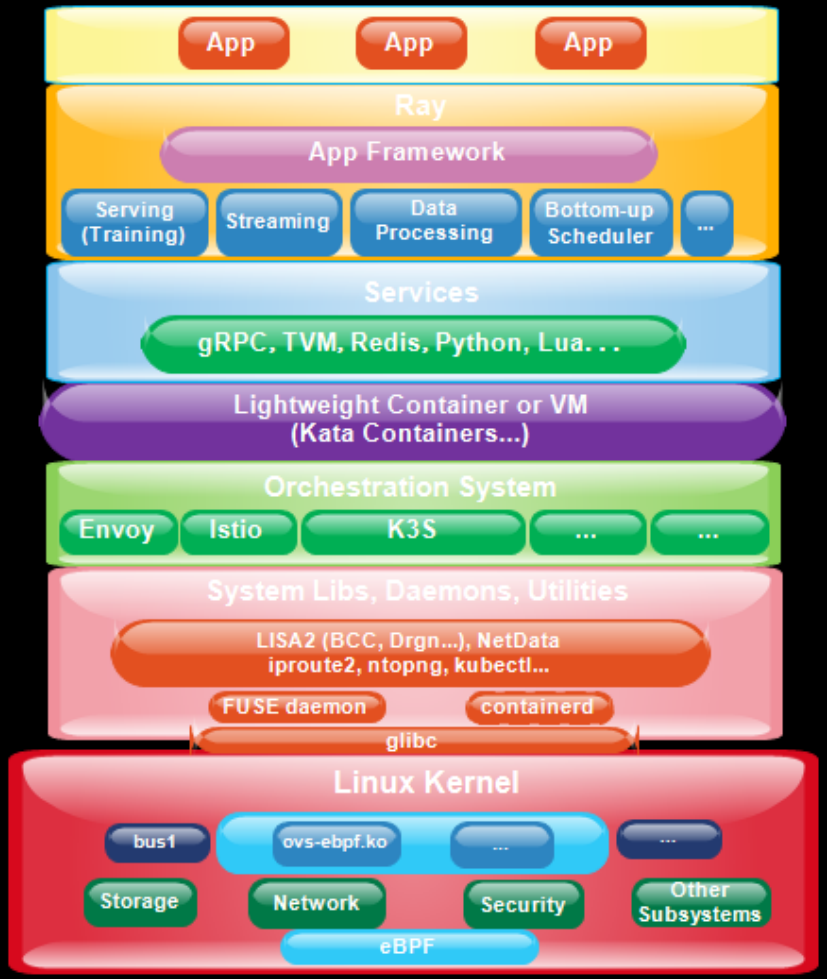
- https://cilium.io/
- eBPF-based Networking, Security, and Observability



- https://cilium.io/blog/2021/05/11/cni-benchmark

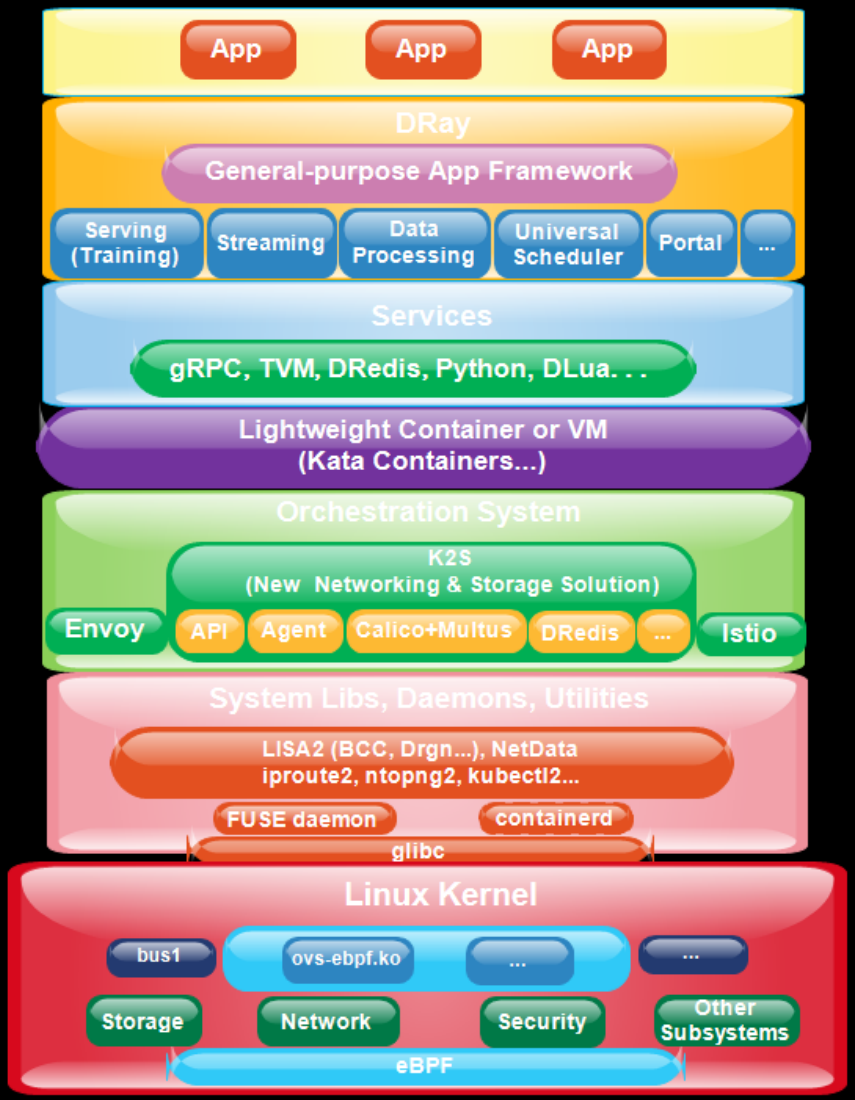
2.2 Our New Design

- an eBPF-centric new lightweight for Edge Computing
- Stage I



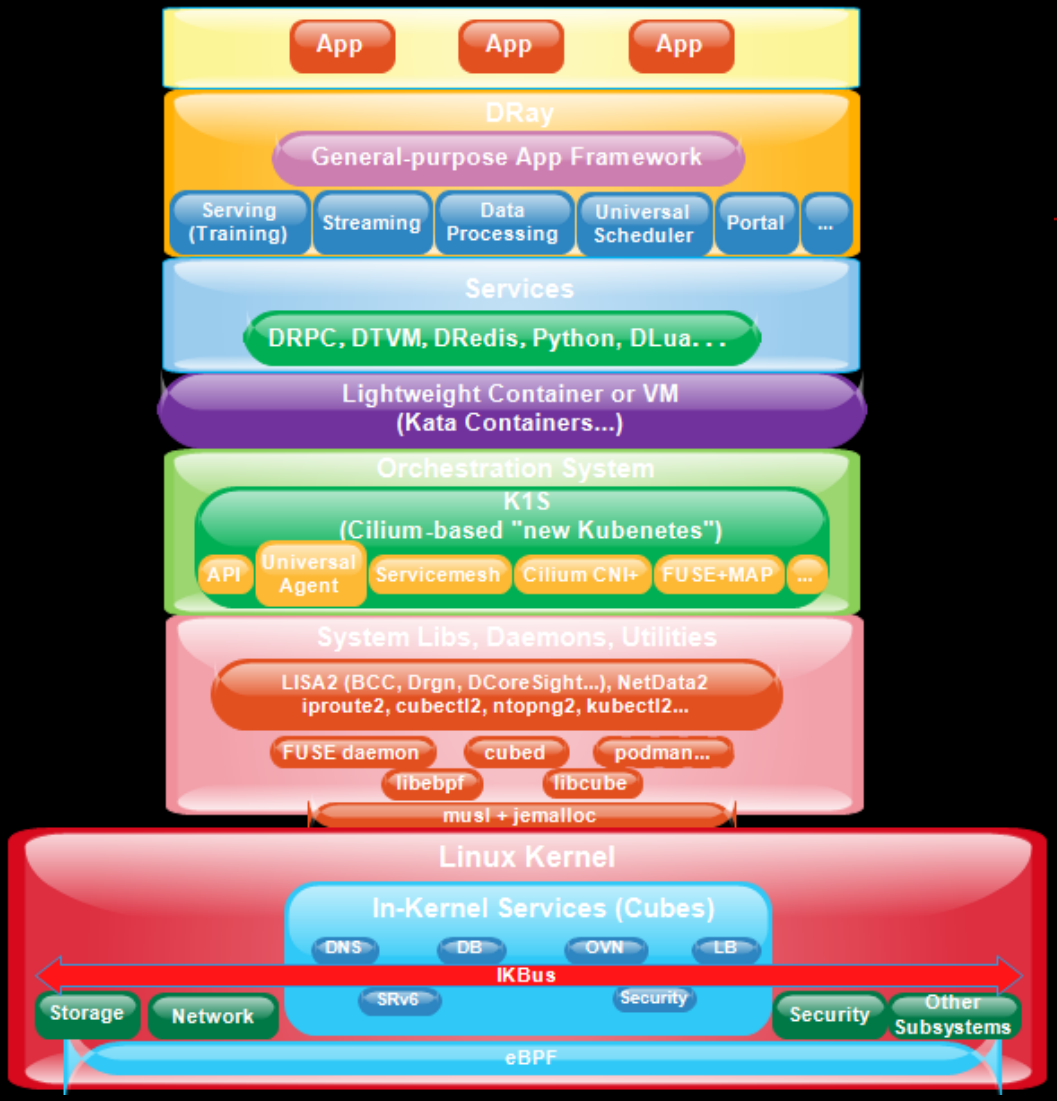
Source: "Rethinking Hyper-Converged Infrastructure for Edge Computing", Feng Li, OpenInfra Days 2019(Shanghai)

Stage II



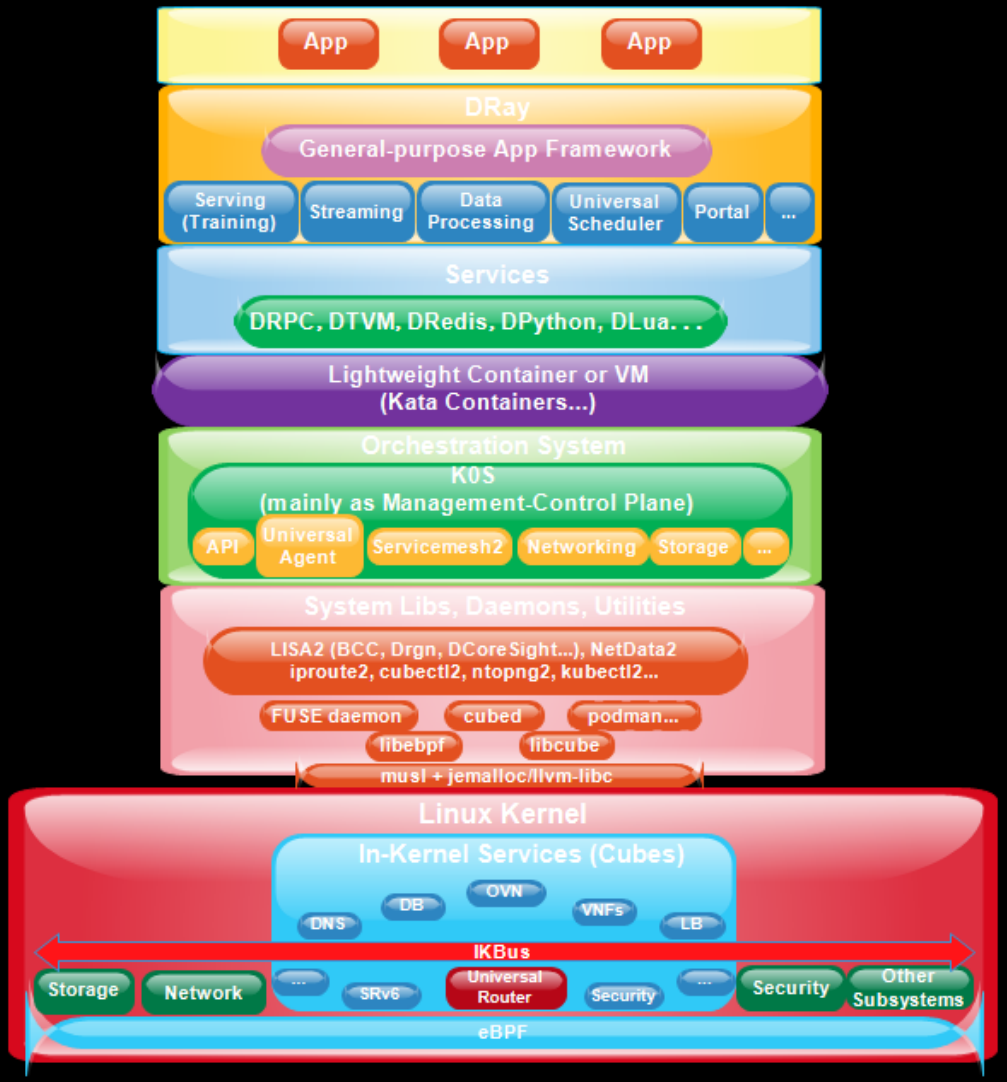
Source: "Rethinking Hyper-Converged Infrastructure for Edge Computing", Feng Li, OpenInfra Days 2019(Shanghai)

Stage III



Source: "Rethinking Hyper-Converged Infrastructure for Edge Computing", Feng Li, OpenInfra Days 2019(Shanghai)

Stage IV



Source: "Rethinking Hyper-Converged Infrastructure for Edge Computing", Feng Li, OpenInfra Days 2019(Shanghai)

3) Rust for Linux Kernel Development

- https://lwn.net/Articles/853423/ //Rust heads into the kernel?
- https://lwn.net/Articles/852704/ //Rust in the Linux kernel
- https://lwn.net/Articles/849849/ //Rust support hits linux-next
- https://lwn.net/Articles/829858/ //Supporting Linux kernel development in Rust
- https://blogs.gartner.com/manjunath-bhat/2021/01/03/why-2021-will-be-a-rusty-year-for-system-programmers/
- https://developers.slashdot.org/story/21/04/17/009241/linus-torvalds-says-rust-closer-for-linux-kernel-development-calls-c-a-crap-language
- https://www.infoq.com/news/2021/04/rust-linux-kernel-development/
- https://itwire.com/open-source/rust-support-in-linux-may-be-possible-by-5-14-release-torvalds.html

- https://github.com/libbpf/libbpf-rs //Idiomatic rust wrapper around libbpf
- https://github.com/foniod/redbpf //A Rust eBPF toolchain

Cloud Hypervisor

- **https://github.com/cloud-hypervisor/cloud-hypervisor**

Cloud Hypervisor is an open source Virtual Machine Monitor (VMM) that runs on top of KVM and the MSHV hypervisors .

The project focuses on exclusively running modern, cloud workloads, on top of a limited set of hardware architectures and platforms. Cloud workloads refers to those that are usually run by customers inside a cloud provider. For our purposes this means modern operating systems with most I/O handled by paravirtualised devices (i.e. virtio), no requirement for legacy devices, and 64-bit CPUs.

Cloud Hypervisor is implemented in Rust and is based on the rust-vmm crates.

- **Objectives**

High Level

- Runs on KVM or MSHV
- Minimal emulation
- Low latency
- Low memory footprint
- Low complexity
- High performance
- Small attack surface
- 64-bit support only
- CPU, memory, PCI hotplug
- Machine to machine migration

Architectures

Cloud Hypervisor supports the `x86-64` and `AArch64` architectures. There are some small differences in functionality between the two architectures (see #1125).

Guest OS

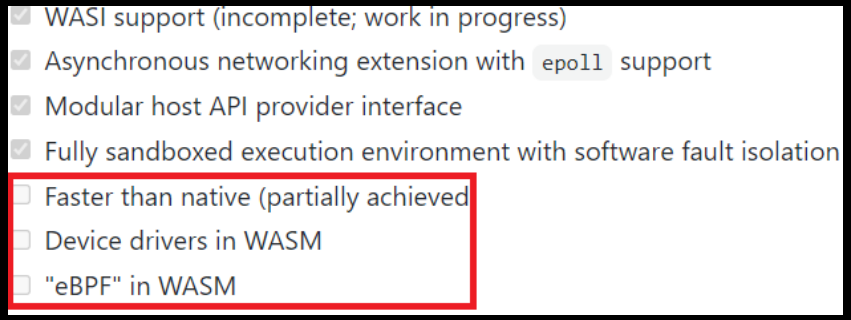
Cloud Hypervisor supports `64-bit Linux` and Windows 10/Windows Server 2019.

4) WebAssembly

- https://en.wikipedia.org/wiki/WebAssembly
- https://www.rust-lang.org/what/wasm
- https://rustwasm.github.io/book
- https://github.com/rustwasm

In-kernel WASM

- https://github.com/wasmerio/kernel-wasm
  //sandboxed kernel mode WebAssembly runtime



- https://github.com/kenny-ngo/wasmjit

Its primary target is a Linux kernel module that can host Emscripten-generated WebAssembly modules. In this configuration it runs WebAssembly modules in kernel-space (ring 0) and provides access to system calls as normal function calls. This configuration avoids user-kernel transition overhead, as well as scheduling overheads from swapping page tables. This results in a significant performance increase for syscall-bound programs like web servers or FUSE file systems. The performance increase is more pronounced in a post-Spectre/Meltdown world due to PTI overhead. Check it out:

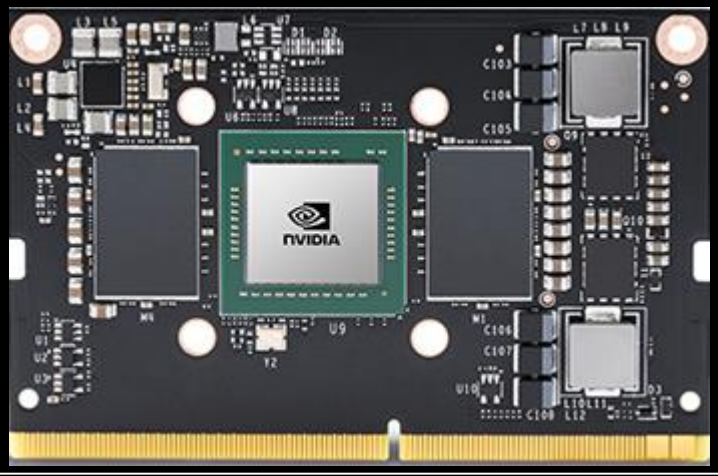
5) Edge AI

- Definition?
- https://towardsdatascience.com/edge-ai-is-the-next-wave-of-ai-a3e98b77c2d7
- https://wiki.akraino.org/display/AK/The+AI+Edge+Blueprint+Family

5.1 Nvidia Jetson

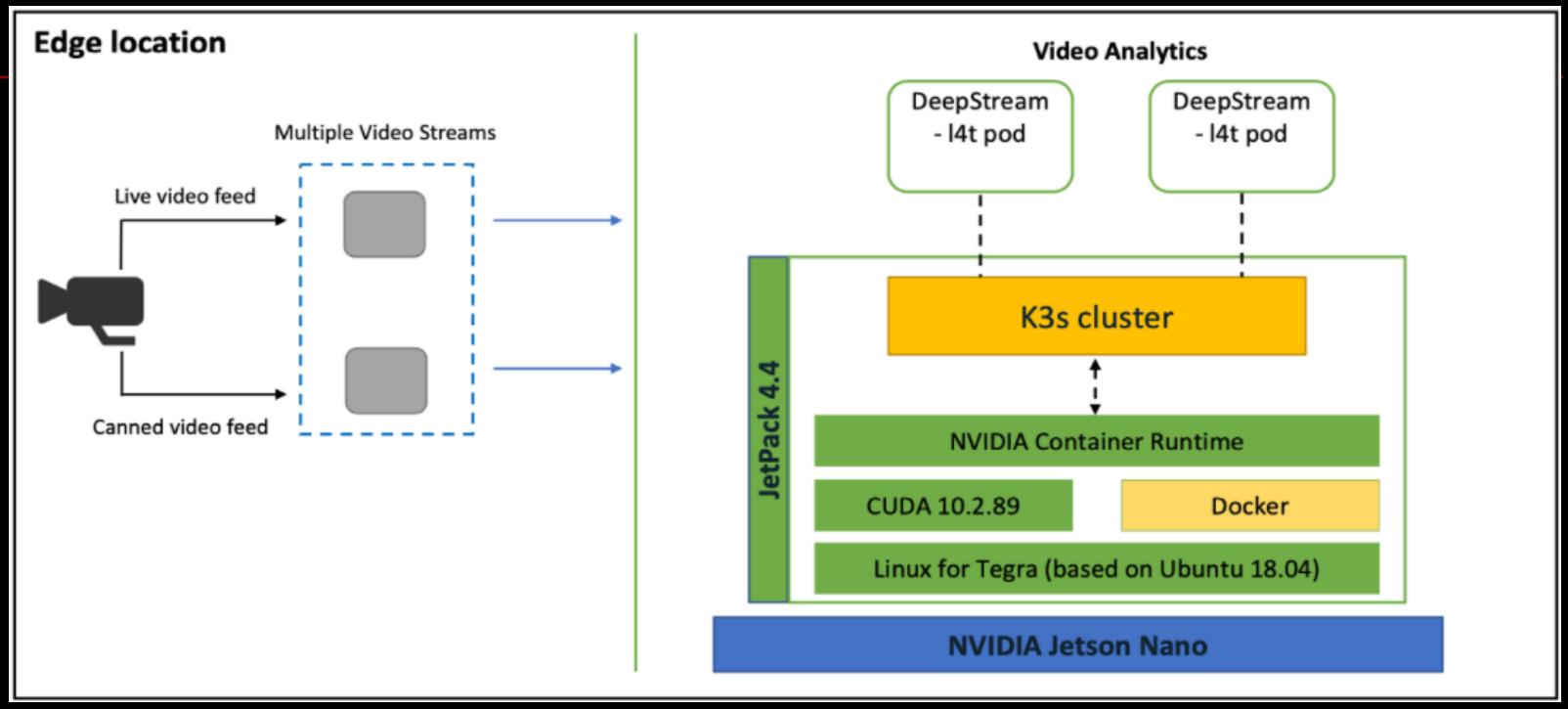
- https://www.nvidia.com/en-us/autonomous-machines/jetson-store/

  Jetson Nano; Jetson Xavier NX; Jetson AGX Xavier; Jetson TX2…



AI at the Edge with K3s and NVIDIA Jetson Nano

- **https://www.suse.com/c/ai-at-the-edge-with-k3s-nvidia-jetson-nano-object-detection-real-time-video-analytics-src/**



NVidia Jetson Lab Environment - exploring k3s on ARM

- **https://github.com/cloudxabide/jetsons.lab/**

Running Kata Containers on Jetson series?

- **Ubuntu 18.04.5 LTS**

```
mydev@mydev-jetson1:~$ uname -a
Linux mydev-jetson1 4.9.201-tegra #1 SMP PREEMPT Fri Feb 19 08:40:32 PST 2021 aarch64 aarch64 aarch64 GNU/Linux
mydev@mydev-jetson1:~$
mydev@mydev-jetson1:~$ dmesg |grep -i kvm
mydev@mydev-jetson1:~$
mydev@mydev-jetson1:~$ sudo zcat /proc/config.gz |grep -i kvm
[sudo] password for mydev:
# CONFIG_KVM is not set
mydev@mydev-jetson1:~$
mydev@mydev-jetson1:~$ sudo zcat /proc/config.gz |grep -i virtual
# CONFIG_REGULATOR_VIRTUAL_CONSUMER is not set
# CONFIG_FB_VIRTUAL is not set
CONFIG_DMA_VIRTUAL_CHANNELS=y
# CONFIG_TEGRA_VIRTUALIZATION is not set
CONFIG_VIRTUALIZATION=y
# CONFIG_CRYPTO_DEV_TEGRA_VIRTUAL_SE_INTERFACE is not set
mydev@mydev-jetson1:~$
mydev@mydev-jetson1:~$ sudo zcat /proc/config.gz |grep -i cgroup
CONFIG_CGROUPS=y
CONFIG_CGROUP_DEBUG=y
CONFIG_CGROUP_FREEZER=y
CONFIG_CGROUP_PIDS=y
CONFIG_CGROUP_DEVICE=y
CONFIG_CGROUP_CPUACCT=y
CONFIG_BLK_CGROUP=y
# CONFIG_DEBUG_BLK_CGROUP is not set
CONFIG_CGROUP_WRITEBACK=y
CONFIG_CGROUP_SCHED=y
CONFIG_CGROUP_HUGETLB=y
CONFIG_CGROUP_PERF=y
CONFIG_SOCK_CGROUP_DATA=y
# CONFIG_NETFILTER_XT_MATCH_CGROUP is not set
CONFIG_NET_CLS_CGROUP=y
CONFIG_CGROUP_NET_PRIO=y
CONFIG_CGROUP_NET_CLASSID=y
```

- **https://developer.nvidia.com/embedded/linux-tegra**

NVIDIA L4T is the board support package for Jetson. It includes Linux Kernel 4.9, bootloader, NVIDIA drivers, flashing utilities, sample filesystem based on Ubuntu 18.04, and more for the Jetson platform.

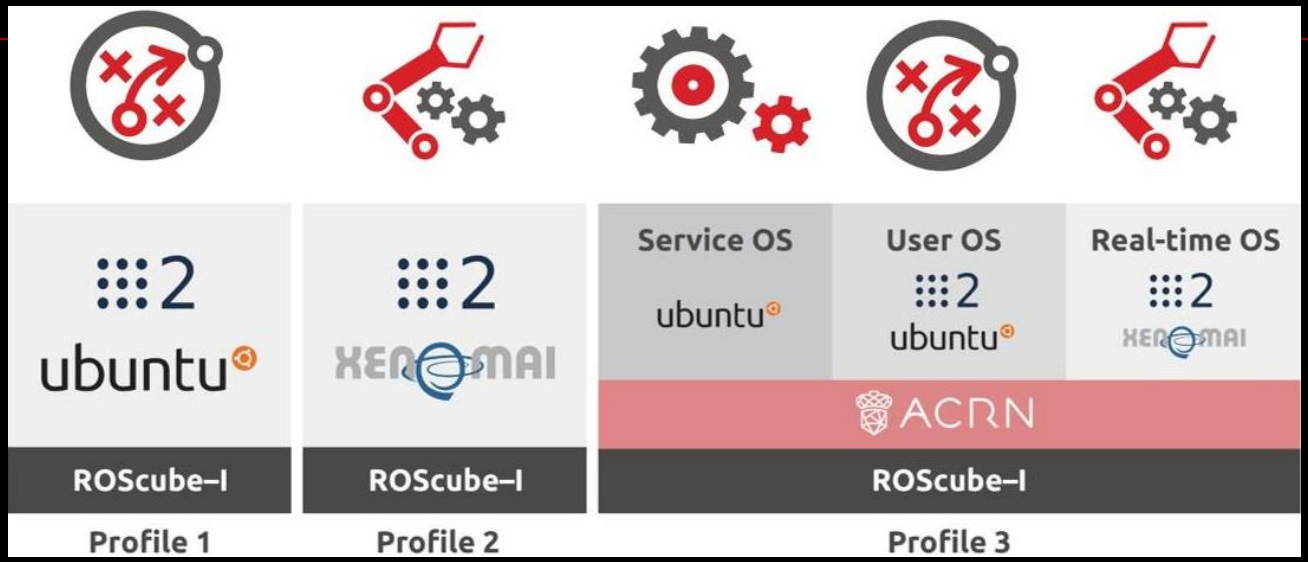
NVIDIA L4T 32.5.1 supports all Jetson modules: Jetson AGX Xavier series, Jetson Xavier NX, Jetson TX2 series, Jetson TX1, and Jetson Nano. All Jetson developer kits are also supported.

L4T 32.5.1 is included as part of JetPack 4.5.1

5.2 ROS

ROScube-I

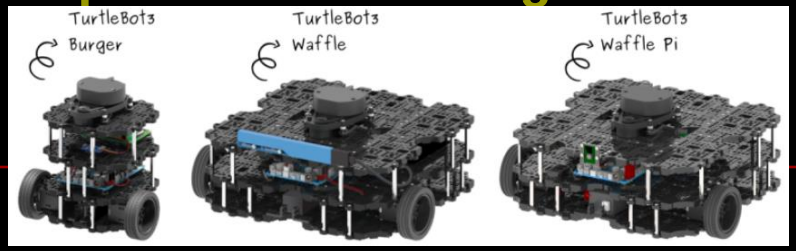
- https://www.adlinktech.com/en/News/ADLINK-ROScube-I-ROS-2-controller-now-with-ACRN-real-time-hypervisor

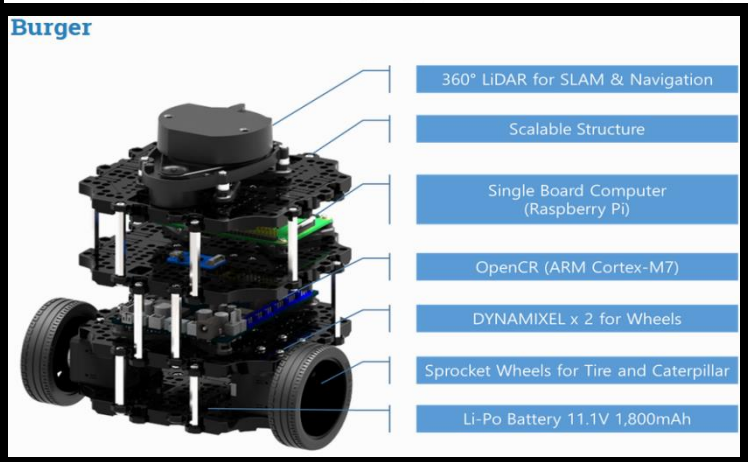


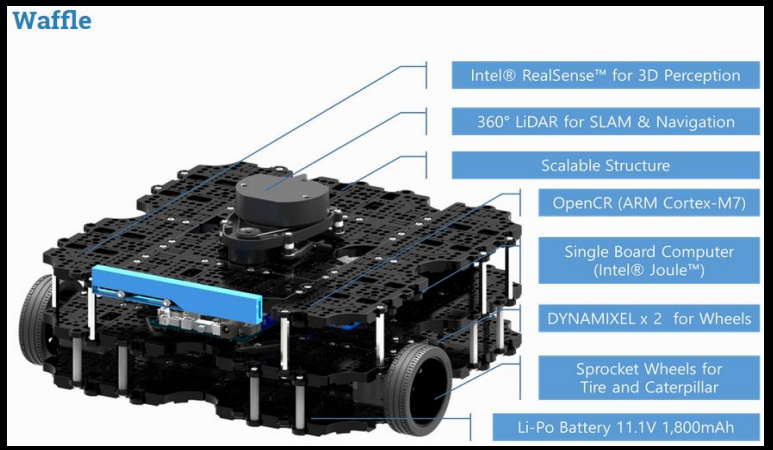
- https://projectacrn.github.io/latest/getting-started/roscube/roscube-gsg.html

Turtlebot3

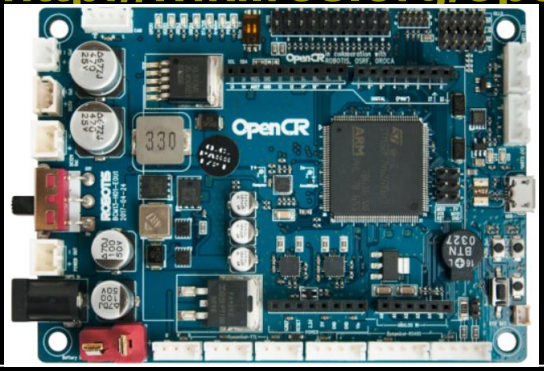
- **https://robots.ros.org/turtlebot3/**







- **http://wiki.ros.org/opencr**



Containerization of ROS2

- https://docs.ros.org/en/foxy/Guides/Run-2-nodes-in-single-or-separate-docker-containers.html
- https://developer.nvidia.com/blog/implementing-robotics-applications-with-ros-2-and-ai-on-jetson-platform-2/
- https://ubuntu.com/blog/ros-2-on-kubernetes-a-simple-talker-and-listener-setup
- https://discourse.ros.org/t/ros-2-on-kubernetes/17182

6) Serverless

- https://en.wikipedia.org/wiki/Serverless_computing
- Open Source Serverless Platforms

6.1 vAccel

- https://vaccel.org/
- Give your Serverless deployments the power of hardware accelerators with vAccel

---

- Problem: hardware acceleration in the Cloud & at the Edge

Hardware partitioning

- bound to hardware device/vendor support
- inflexible sharing of diverse accelerator resources

API remoting

- still either device/vendor API specific, or
- can incur significant performance overhead
- not fit for infrastructures with resource or performance (i.e. latency) constraints

Paravirtualization

- users have to program the hardware directly
- multiple schedulers doing the same job (VM, VMM, runtime system)
- software stack duplication

Source: "ML inference acceleration for lightweight VMMs", Anastassios Nanos, Babis Chalios, Kostis Papazafeiropoulos, Fosdem 2021

- Design Goals

- programmability / simplicity (device/vendor-agnostic)
- performance (minimal overhead)
- portability / interoperability (run anywhere)
- security / isolation (virtualization support)

Source: "ML inference acceleration for lightweight VMMs", Anastassios Nanos, Babis Chalios, Kostis Papazafeiropoulos, Fosdem 2021

Architecture & Design

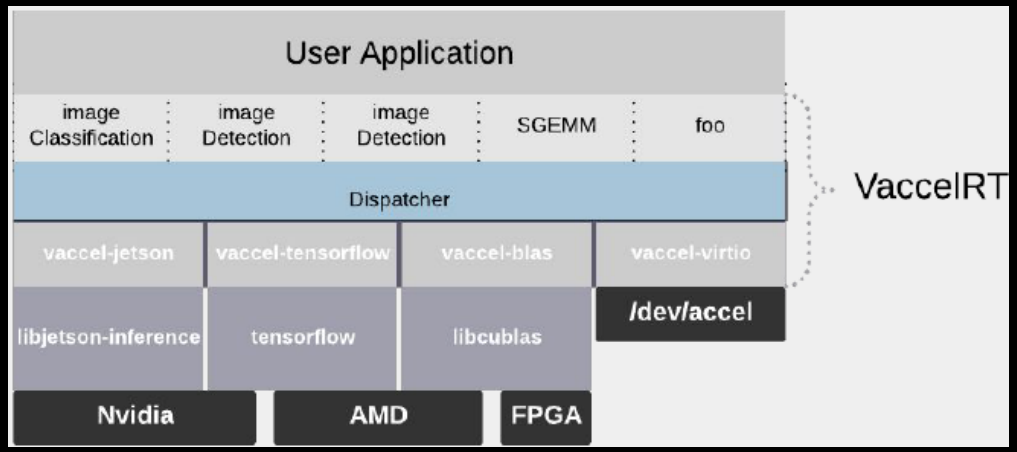
core component: **vAccelRT (vAccel runtime system)**

user-facing API: **function prototypes**

- abstracted by the underlying frameworks or
- defined by the system as a superset / subset of individual acceleration functions

hardware abstraction layer: **acceleration frameworks, transport layer**

- low-level APIs (openCL, CUDA, openACC etc.)
- higher-level frameworks (TensorRT, tensorflow, pytorch etc.)
- user-facing APIs (jetson-inference, libBLAS etc.)
- virtio-accel

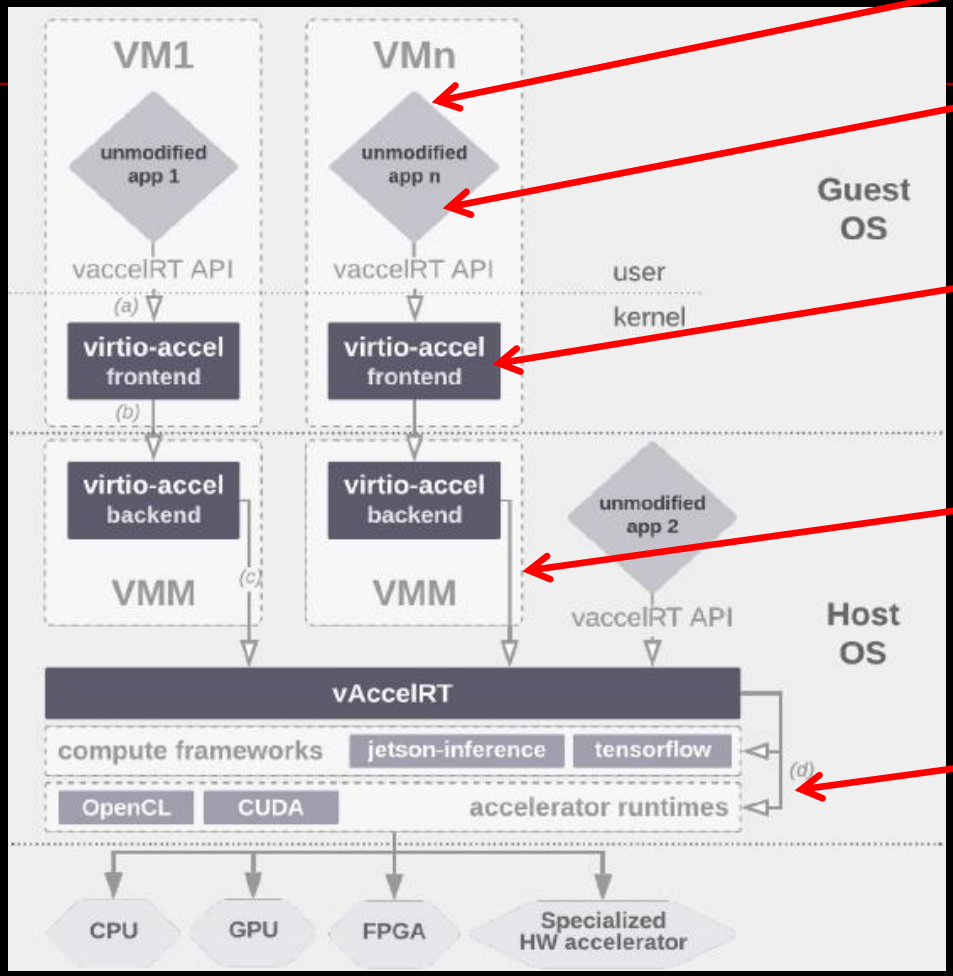


Source: "ML inference acceleration for lightweight VMMs", Anastassios Nanos, Babis Chalios, Kostis Papazafeiropoulos, Fosdem 2021

Example execution flow for a VM

- application

```
image_classify(image, model, parameters, &output)
```



VaccelRT (inside the guest)

1. Look for backend plugin that implements image classification
2. If supported, call the plugin code

vaccel-virtio plugin

1. Check that the virtio device is present
2. Prepare arguments and issue `ioctl` command to the vaccel device

vaccel-virtio front-end driver

1. Create virtio request
2. Insert user arguments inside the request
3. Insert the request inside the virtqueue
4. Kick the virtqueue

VMM

1. The guest VM exits in the VMM
2. The virtio-backend inside the VMM
   a. Parses the request
   b. Validates the request header and corresponding arguments
   c. `image_classify(image, model, parameters, &output)`

VaccelRT (in the host)

1. Look for the backend plugin that implements image classification
2. If supported, call the plugin code

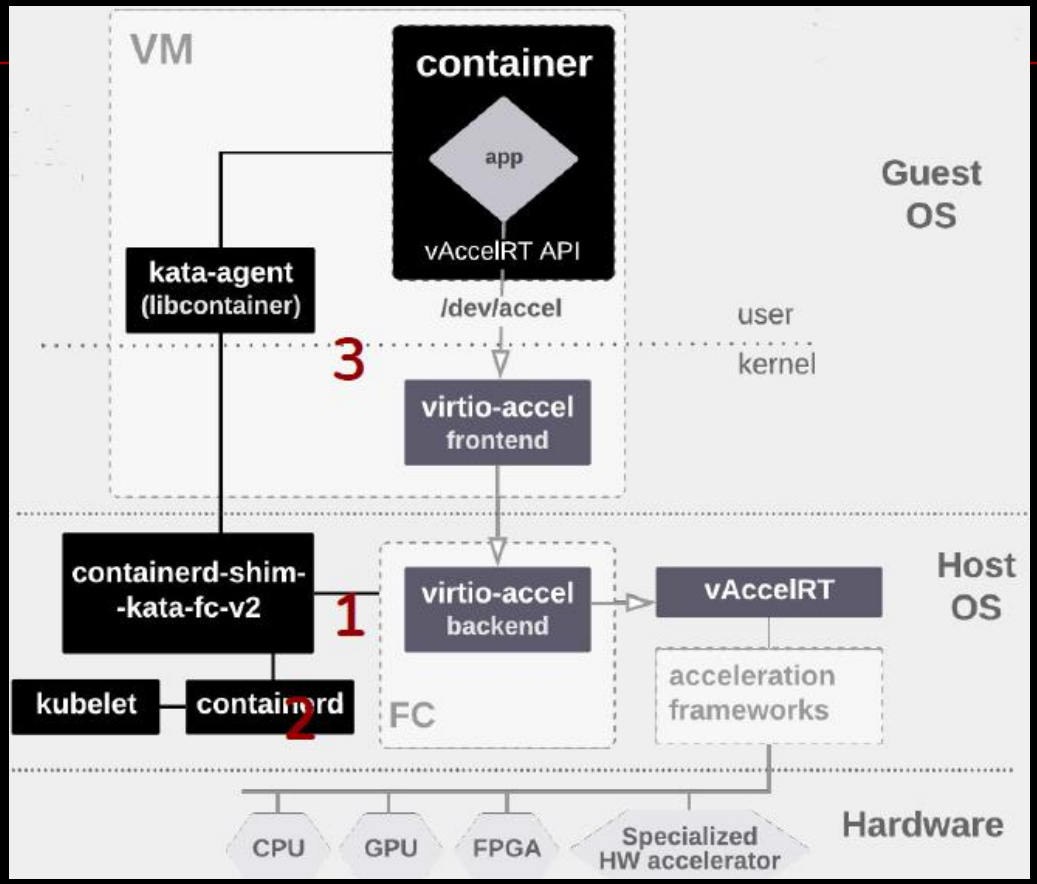
vaccel-jetson plugin

1. Perform operation using jetson-inference framework
2. Offload computation to GPU

Source: "ML inference acceleration for lightweight VMMs", Anastassios Nanos, Babis Chalios, Kostis Papazafeiropoulos, Fosdem 2021

Orchestration

- https://docs.vaccel.org/kata-vaccel/
- AWS Firecracker with vAccel on K8S(using Kata Containers)



1. kata v2 runtime (1.12 & 2.0) with Firecracker v0.21.1
2. docker
3. custom device assignment

1. kata v2 runtime Firecracker <= v0.21.1
   -> **v0.23.1**
2. docker with devmapper
   -> **containerd (still using devmapper)**
3. custom device assignment
   -> **kata-agent patch**

Source: "ML inference acceleration on K8s using kata containers & AWS Firecracker", Orestis Lagkas Nikolos, Babis Chalios, Anastassios Nanos, Fosdem 2021

- https://docs.vaccel.org/unikernels/unikraft/

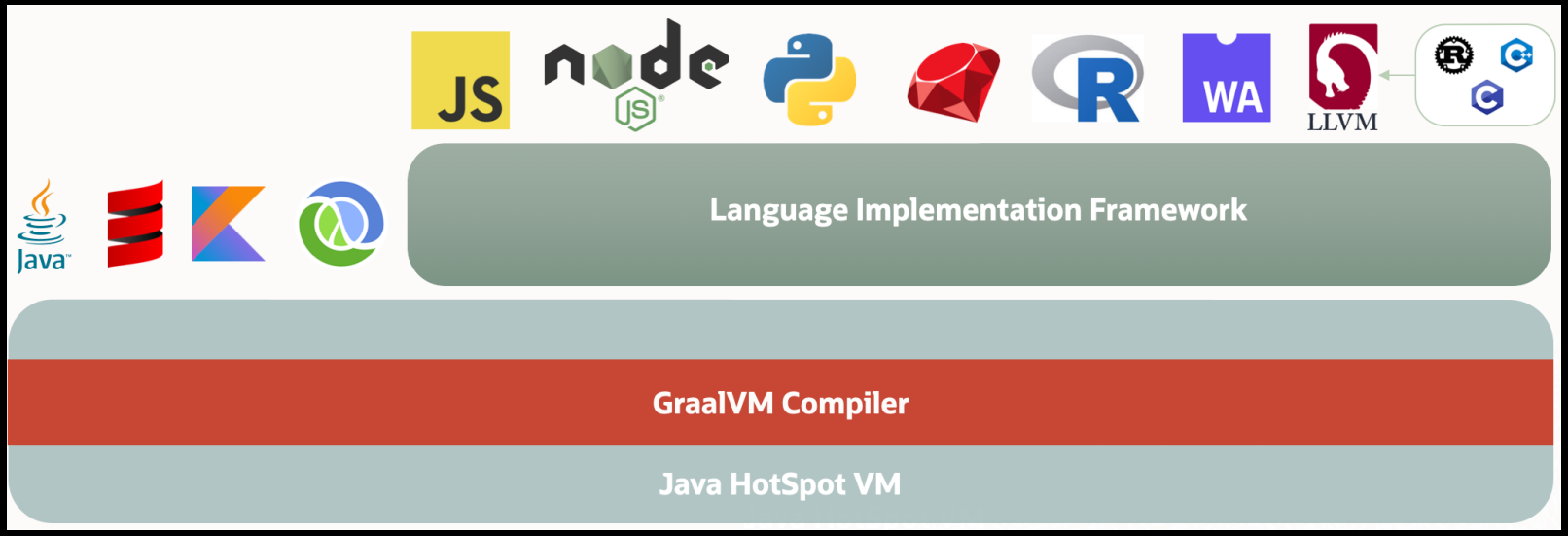
7) Unified Runtime for eBPF & WASM

In-Kernel

- AIO runtime in Rust?

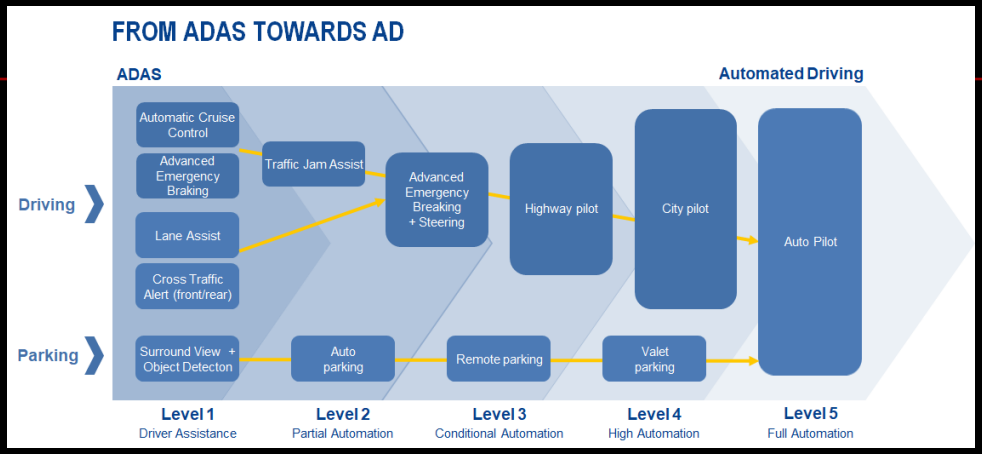
User space

- Implement uBPF in GraalVM
  https://github.com/oracle/graal/blob/master/wasm/README.md

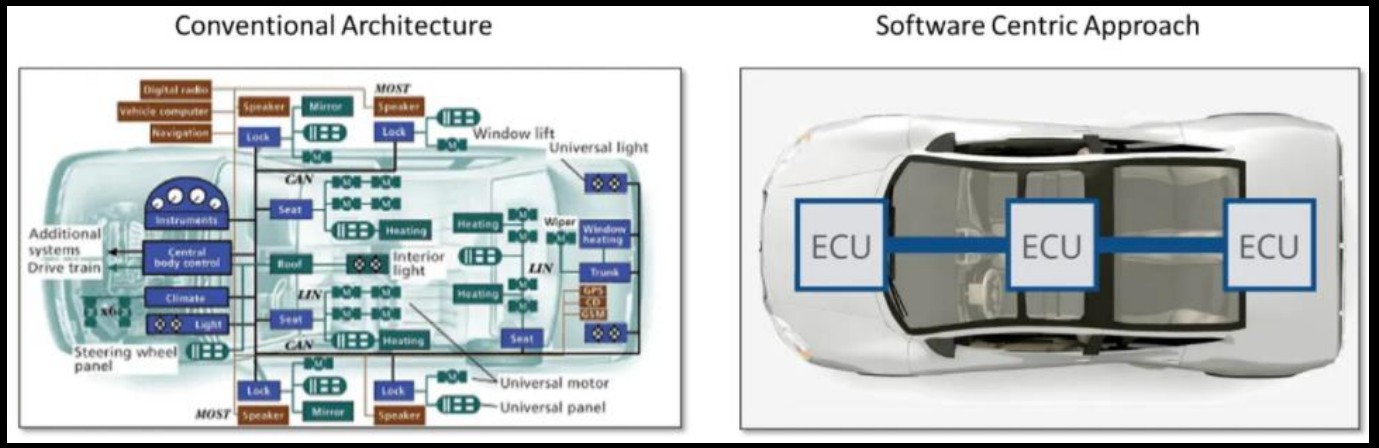


8) Autonomous Vehicles

- https://en.wikipedia.org/wiki/Self-driving_car
- https://en.wikipedia.org/wiki/Advanced_driver-assistance_systems



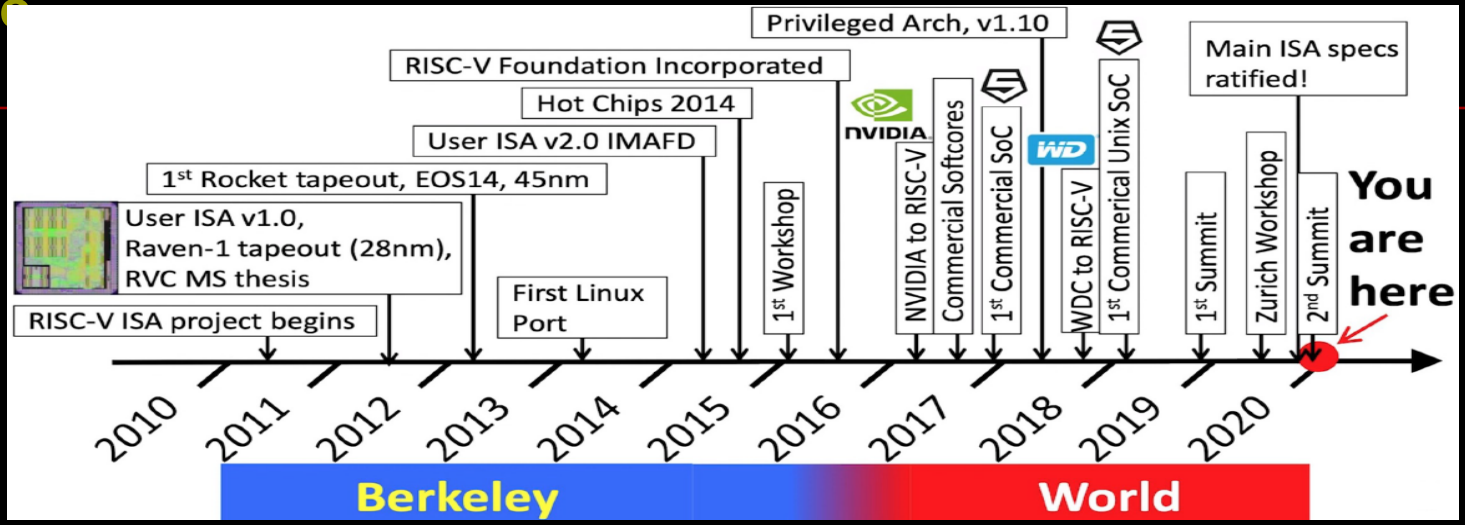
- https://www.autovision-news.com/hmi/vehicle-control/changes-vehicle-electrical-architecture/



- https://en.wikipedia.org/wiki/Time-Sensitive_Networking

9) RISC-V

- https://en.wikipedia.org/wiki/RISC-V
-



Source: "Linux on RISC-V", D. Fustimi, ELC2020

A New Golden Age for Computer Architecture

- **https://cacm.acm.org/magazines/2019/2/234352-a-new-golden-age-for-computer-architecture/fulltext**

"the Linux of the chip world"

- **https://www.zdnet.com/article/risc-v-the-linux-of-the-chip-world-is-starting-to-produce-technological-breakthroughs/**

BeagleV

- **https://beagleboard.org/beaglev**
- **https://beaglev.seeed.cc/**
- **The First Affordable RISC-V Computer Designed to Run Linux**

Powerful and Rich AI Features

$119 for 4GB RAM
$149 for 8GB RAM

The Early Version of BeagleV™ only comes with 8GB Ram at $149

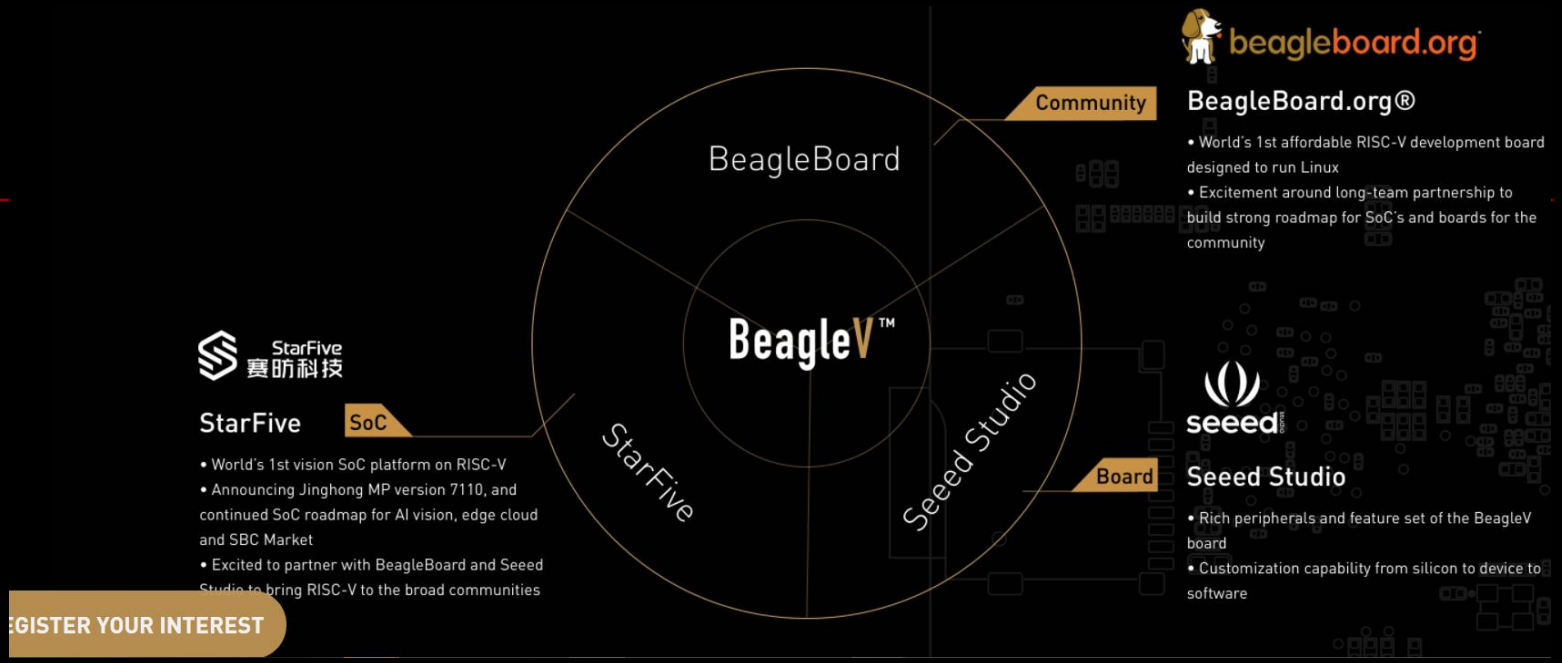
TRULY OPEN SOURCE

Open Source Software

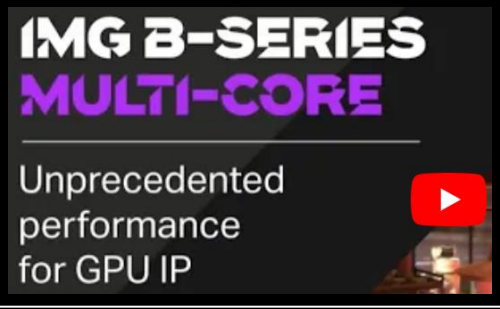
Based on RISC-V Open Architecture

Open Hardware Design

Strategy Partners



https://www.imaginationtech.com/news/press-release/imaginations-gpu-selected-by-starfive-to-create-high-performance-small-and-low-cost-beaglev-risc-v-ai-single-board-computer/

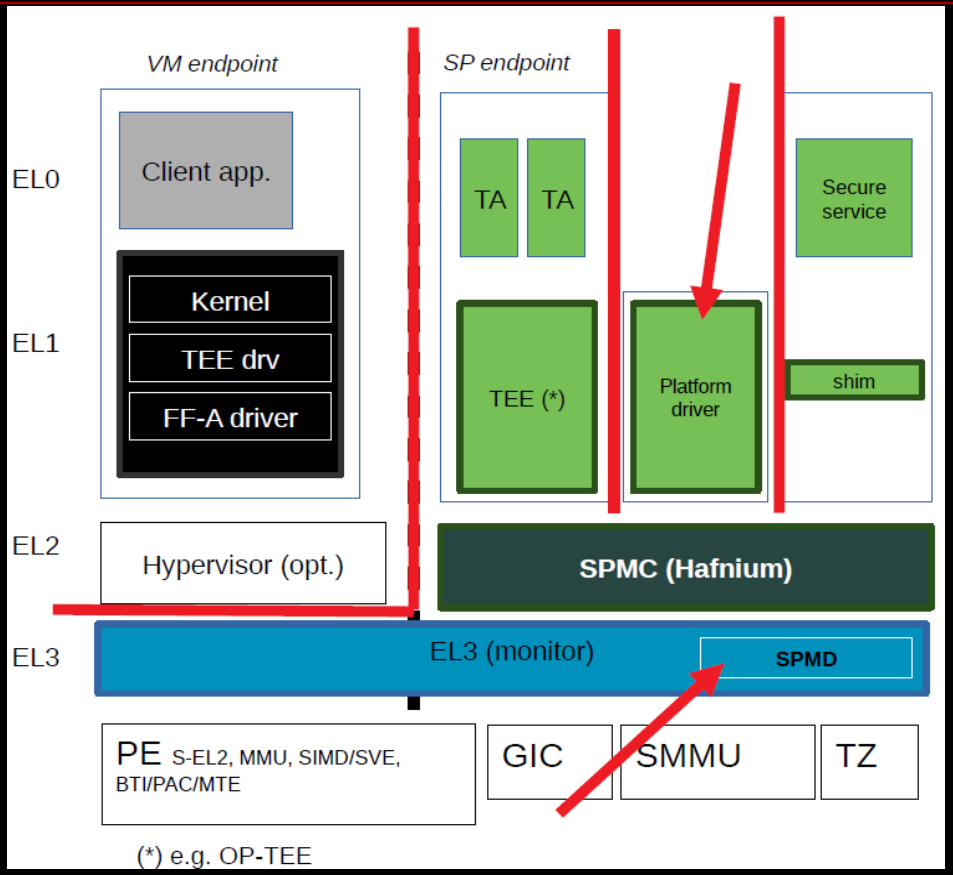


10) HW-SW Co-designed System Security

- Hardware-Software Co-designed System and System Security is the trend in the real world
- https://en.wikipedia.org/wiki/Trusted_execution_environment

10.1 Security on ARM

- https://developer.arm.com/architectures/architecture-security-features
- ARMv8.4 Secure EL2



Source: "Secure Partition Manager(Armv8.4 Secure EL2)", Olivier Deprez & Madhukar Pappireddy, LVC21-303

Project Cassini for a Cloud-Native Edge

- https://www.arm.com/solutions/infrastructure/edge-computing/project-cassini
- extends PSA to the infrastructure Edge

PSA for M-profile (constrained IoT) deliverable	PSA for Infrastructure Edge deliverable
Threat models	New Threat Model and Security Analysis (TMSA) for the Edge
Firmware Framework-M	Firmware Framework-A
Trusted Boot and Firmware Update	Same
PSA APIs (Crypto, Attestation, Secure Storage)	Same A new open-source security microservice: PARSEC
Trusted Firmware-M	Trusted Firmware-A New features being added to support the needs of infrastructure
Certification scheme	In progress – Working to understand market need
-	New SBSG – Server Base Security Guide

Source: https://www.arm.com/-/media/global/solutions/artificial-intelligence/Project_Cassini.pdf

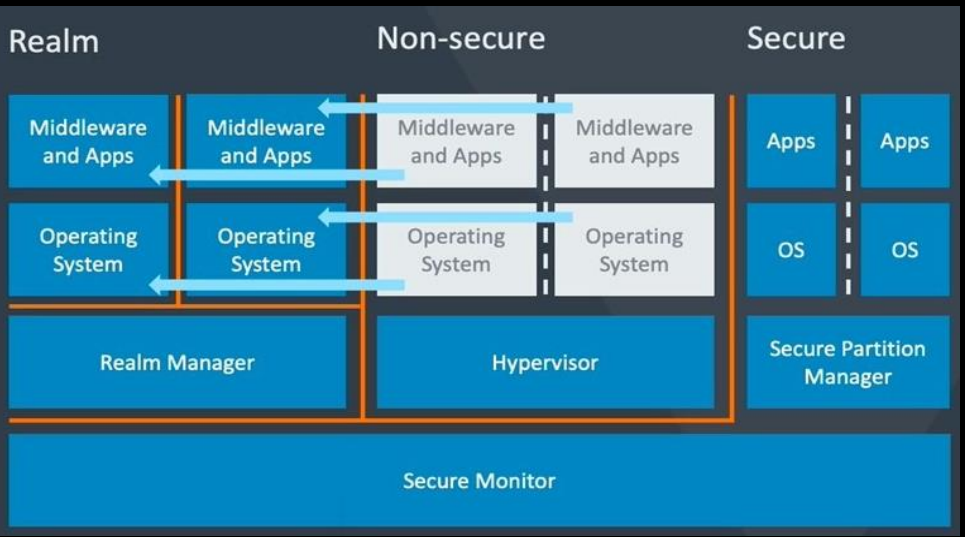
- PSA (Platform Security Architecture) https://www.arm.com/why-arm/architecture/platform-security-architecture

ARM v9

- https://www.embedded.com/new-armv9-takes-on-security-and-ai/

- CCA (Confidential Compute Architecture)

- **Realms** enable you to run an application or service in such a way that your data is protected from:
  - The host
  - The hardware manufacturer or service supplier
  - Other software running on the device or sharing the machine



VI. Wrap-up

- Lightweight VM like Kata Containers is well suited for Edge- and Serverless- computing

---

- Linux is a key player in the edge-computing revolution
- Unified runtime for eBPF & WASM?
- Rethinking App Runtime – User space and Kernel space Repartitioning or Unifying

Q & A

Thanks!

Reference

Slides/materials from many and varied sources:

- http://en.wikipedia.org/wiki/
- http://www.slideshare.net/
- https://katacontainers.io/collateral/kata-containers-onboarding-deck.pptx
- https://www.openstack.org/videos/summits/denver-2019/kata-containers-story-of-a-container-runtime-1
- https://www.lfedge.org/resources/publications/
- https://medium.com/wasmer/running-webassembly-on-the-kernel-8e04761f1d8e
- https://www.arm.com/products/silicon-ip-security
- https://developer.arm.com/documentation/102142/latest/Secure-virtualization
- https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/project-cassini-ensuring-a-cloud-native-experience-for-secure-edge?_ga=2.45148257.414898915.1620708938-1735275506.1620708938
- https://github.com/firecracker-microvm/firecracker-containerd/blob/main/docs/architecture.md
- https://github.com/cloud-hypervisor/

- https://github.com/firecracker-microvm
- https://github.com/edgexfoundry
- https://gerrit.akraino.org/r/q/status:open+-is:wip
- https://containerd.io/

- https://github.com/opencontainers
- https://www.graalvm.org/docs/introduction/
- …