
KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023

Extending Cloud Native Boundaries! Bringing Cloud Native Workloads to AndroidOS with KubeEdge

鲍玥 Yue Bao

温福才 Fucai Wen

Bio

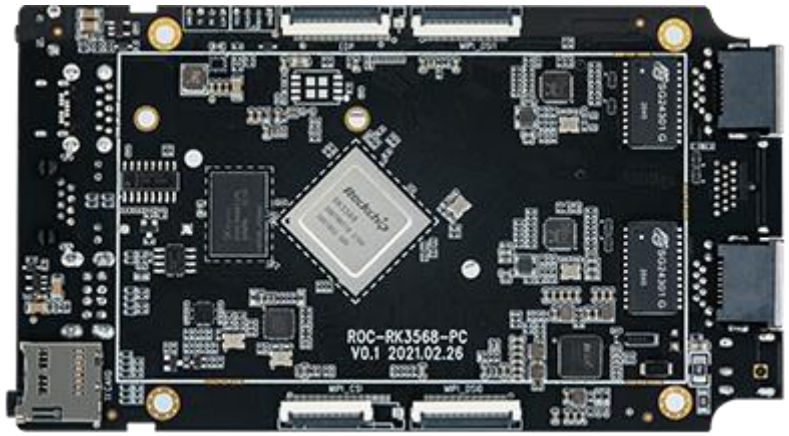
鲍玥/Yue Bao
Software Engineer
Huawei Cloud

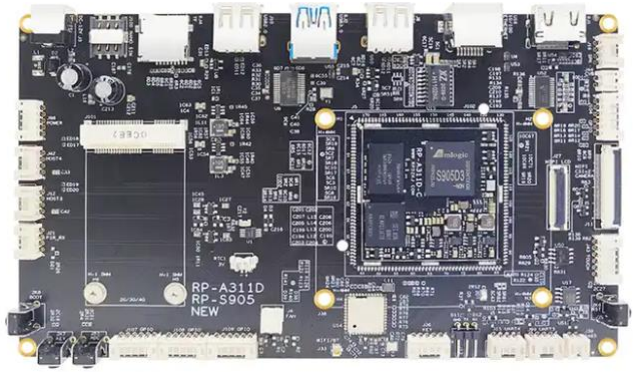
温福才/Fucai Wen
Architect
Beijing KongJie Smart Technology Corporation

Background

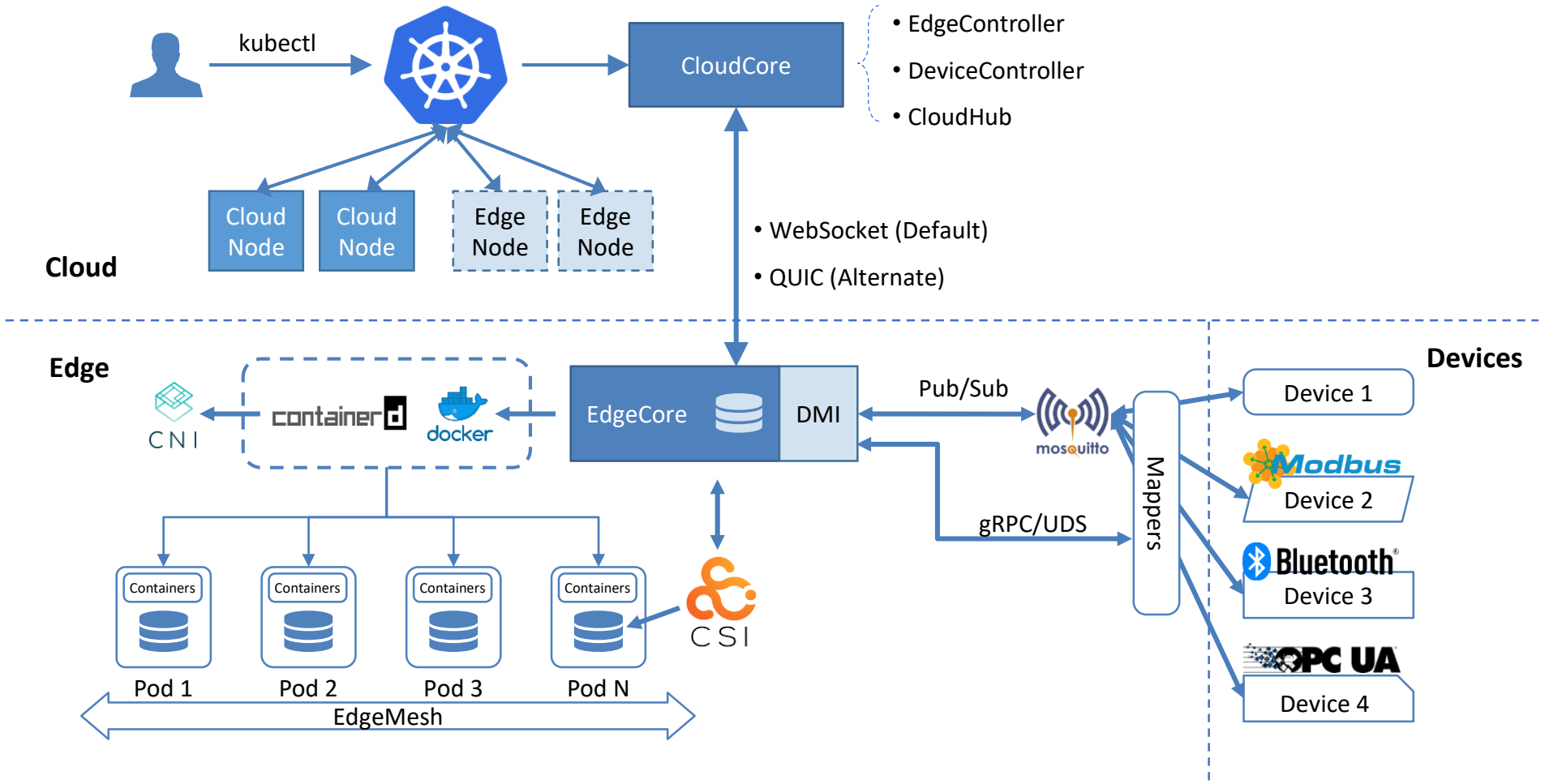
Devices based on AndroidOS are widely used in various business domains such as set-top boxes, advertising screens, robots, and SDV(Software Defined Vehicle). This highlights the extensive application of Android-based mobile devices in the current landscape. Furthermore, in the context of edge IoT scenarios, there is a strong demand for cloud-native technologies in mobile smart devices. We are exploring:

- Leveraging the hardware capabilities of Android devices and software resources such as OCI images;
- Applying cloud-edge collaborative edge computing technologies to Android devices;
- Further **expanding the application scenarios of cloud-native edge computing with KubeEdge in the mobile domain, extending cloud-native boundaries**.





KubeEdge Architecture



KubeEdge Core Concept

➢ Open Ecology

- **100% compatibility with native Kubernetes capabilities**, supporting users to manager edge applications uniformly using native Kubenetes APIs.
- **Reliable list-watch interfaces for the edge**, integrating with the native ecosystem.
- Edge device communication protocols that support MQTT, Modbus, Bluetooth, Wi-Fi, OPC-UA and etc., with the ability to extend edge device protocols through custom plugins.

➢ Support managing massive edge devices

- Lightweight of the components for resource-constrained environments, **~70MB memory footprint**
- Pluggable edge device management framework, support user-defined extension, **decoupling the underlying device communication.**

➢ Support complex edge-cloud network environment

- **Bidirectional multiplexed edge-cloud messaging channels,** support edge nodes located in private networks.
- **Application-layer reliable incremental synchronization mechanism**, support operation in high-latency, low-quality network environments.

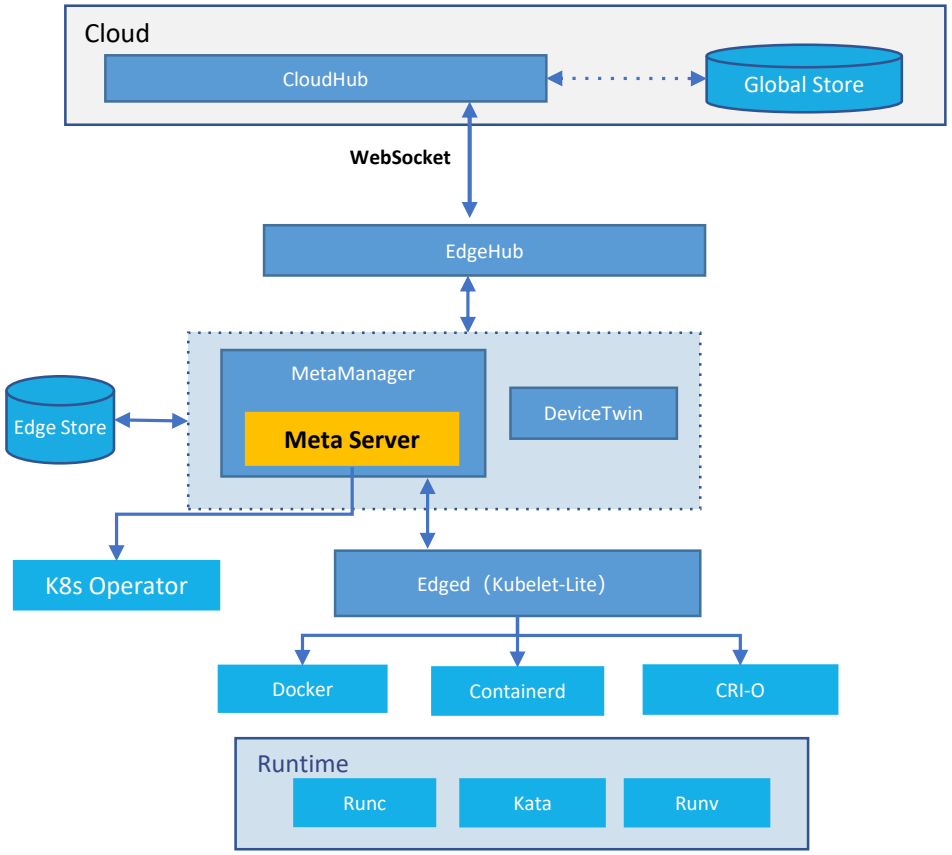
➢ Application/data edge autonomy

- **Edge offline autonomy**, ensuring business continuity and fault recovery capabilities
- Edge data stream processing, defining **edge data cleansing, data analysis** and other processing tasks.

➢ Edge-cloud integrated resource scheduling and traffic coordination

- **Hybrid management of edge and cloud nodes.**
- Provision of **edge-cloud data communication** and **edge-to-edge data communication.**

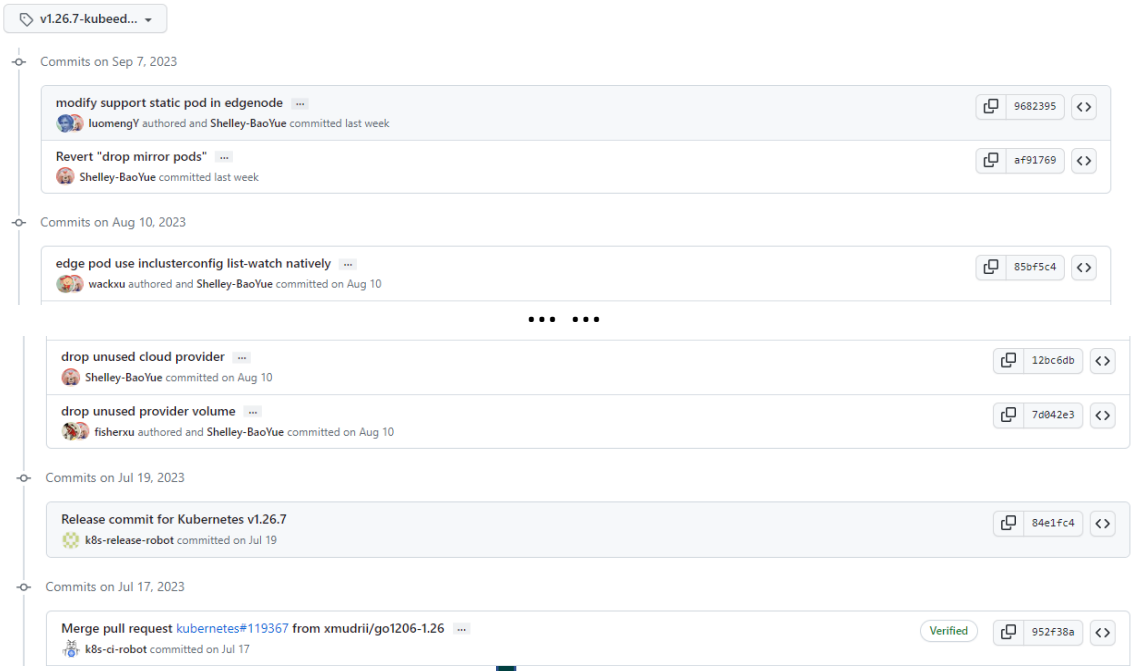
Edge Architecture



Integrate a custom-tailored Kubelet in Edged

kubeedge/kubernetes

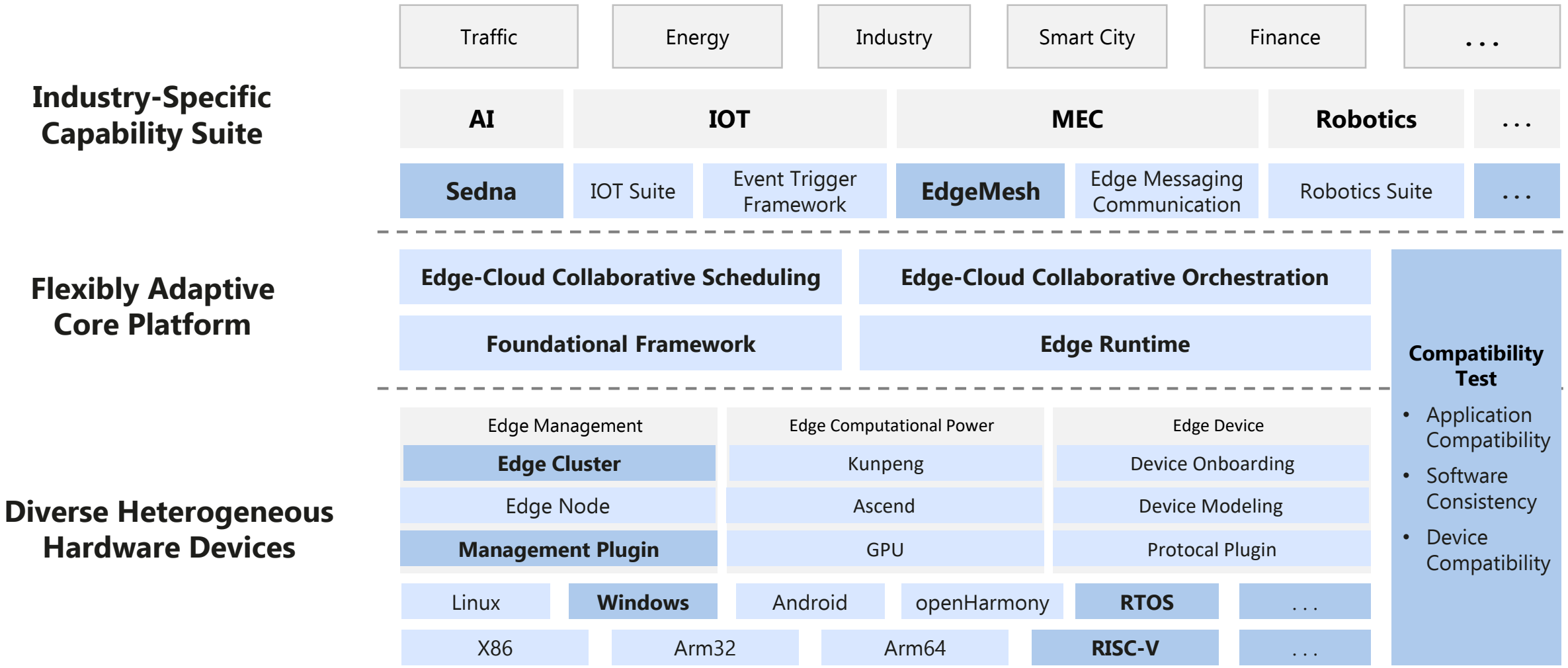
Commits



kubeedge/kubeedge go.mod

```
k8s.io/kubelet => github.com/kubeedge/kubernetes/staging/src/k8s.io/kubelet v1.26.7-kubeedge1
k8s.io/kubernetes => github.com/kubeedge/kubernetes v1.26.7-kubeedge1
k8s.io/legacy-cloud-providers => github.com/kubeedge/kubernetes/staging/src/k8s.io/legacy-cloud-providers v1.26.7-kubeedge1
k8s.io/metrics => github.com/kubeedge/kubernetes/staging/src/k8s.io/metrics v1.26.7-kubeedge1
k8s.io/mount-utils => github.com/kubeedge/kubernetes/staging/src/k8s.io/mount-utils v1.26.7-kubeedge1
k8s.io/node-api => github.com/kubeedge/kubernetes/staging/src/k8s.io/node-api v1.26.7-kubeedge1
```

Edge Computing platform



Runing KubeEdge on AndroidOS: Taking *RK3568* As An Example

Technical Key Points

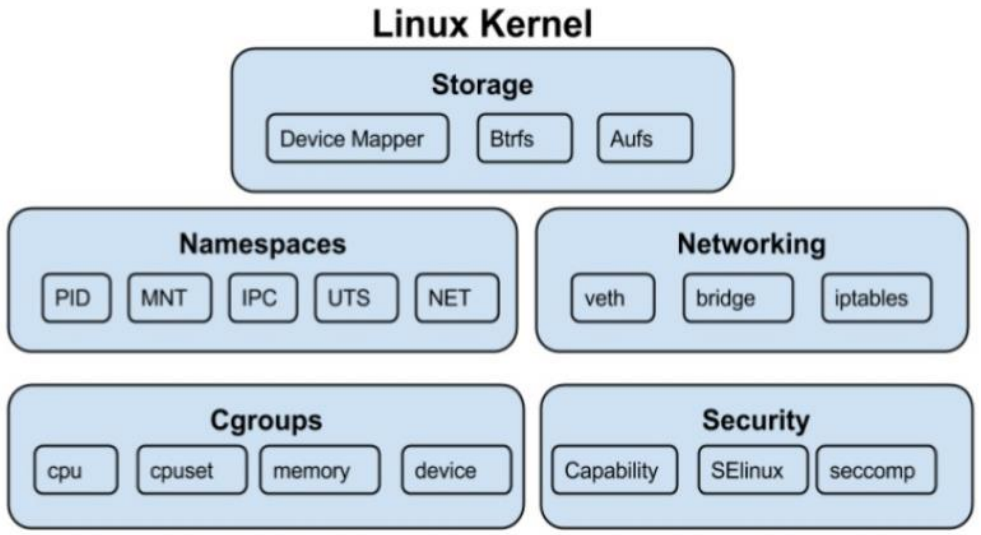
- **Android Linux Kernel:** Modifying the kernel to support containers, similar to Linux OS
  - (UTS, Pid, IPC etc.) Namespaces
  - (Mem etc.) Control group
  - Network
  - Overlay filesystem and etc.
  - CONFIG_SCHED_WALT -> CONFIG_CFS_BANDWIDTH

- **Android Network**
  - Impact of the Paranoid Network
  - trace/workaround is required because the specific nature of iptables/routing causes Android devices to lack support for traffic forwarding

- **Android Storage**
  - Lack of support for overlay
  - Encryption file systems cannot be used as backing filesystems. Special attention is required when dealing with Android devices.

- **A thorough understanding of Cloud-Native**: such as Android host Mali GPU/camera/speaker resource

Technical Key Points



- Mount directories for Docker

```
mount -t cgroup -o none,name=systemd cgroup /sys/fs/cgroup/systemd
mount -t cgroup -o blkio,nodev,noexec,nosuid cgroup /sys/fs/cgroup/blkio
mount -t cgroup -o cpu,nodev,noexec,nosuid cgroup /sys/fs/cgroup/cpu
mount -t cgroup -o cpuacct,nodev,noexec,nosuid cgroup /sys/fs/cgroup/cpuacct
mount -t cgroup -o cpuset,nodev,noexec,nosuid cgroup /sys/fs/cgroup/cpuset
mount -t cgroup -o devices,nodev,noexec,nosuid cgroup /sys/fs/cgroup/devices
mount -t cgroup -o freezer,nodev,noexec,nosuid cgroup /sys/fs/cgroup/freezer
mount -t cgroup -o hugetlb,nodev,noexec,nosuid cgroup /sys/fs/cgroup/hugetlb
mount -t cgroup -o memory,nodev,noexec,nosuid cgroup /sys/fs/cgroup/memory
mount -t cgroup -o net_cls,nodev,noexec,nosuid cgroup /sys/fs/cgroup/net_cls
mount -t cgroup -o net_prio,nodev,noexec,nosuid cgroup /sys/fs/cgroup/net_prio
mount -t cgroup -o perf_event,nodev,noexec,nosuid cgroup /sys/fs/cgroup/perf_event
mount -t cgroup -o pids,nodev,noexec,nosuid cgroup /sys/fs/cgroup/pids
mount -t cgroup -o rdma,nodev,noexec,nosuid cgroup /sys/fs/cgroup/rdma
mount -t cgroup -o schedtune,nodev,noexec,nosuid cgroup /sys/fs/cgroup/schedtune
```

- Add routing rules

```
ip rule add pref 1 from all lookup main
ip rule add pref 2 from all lookup default
```

- Disabling SELinux

```
setenforce 0
```

```
+CONFIG_UTS_NS=y
+CONFIG_PID_NS=y
+CONFIG_OVERLAY_FS=y
+CONFIG_CGROUP_PIDS=y
+CONFIG_CGROUP_DEVICE=y
+CONFIG_MEMCG=y
+CONFIG_BLK_CGROUP=y
+CONFIG_CFS_BANDWIDTH=y
+CONFIG_IPC_NS=y
+CONFIG_USER_NS=y
+CONFIG_NETFILTER_XT_MATCH_ADDRTYPE=y
+CONFIG_NETFILTER_XTABLES=y
+CONFIG_IP_VS=y
+CONFIG_NETFILTER_ADVANCED=y
+CONFIG_NETFILTER_XT_MATCH_IPVS=y
+CONFIG_NETFILTER_XT_TARGET_CHECKSUM=y
+CONFIG_POSIX_MQUEUE=y
+CONFIG_EXT4_FS_POSIX_ACL=y
+CONFIG_VXLAN=y
+CONFIG_AUFS_FS=y
+CONFIG_IP_SET=y
+CONFIG_IP_SET_HASH_IP=y
+CONFIG_IP_SET_HASH_NET=y
+CONFIG_NETFILTER_XT_SET=y
+CONFIG_SYSVIPC=y
+CONFIG_BLK_DEV_THROTTLING=y
+CONFIG_BINFMT_MISC=y
+CONFIG_NETFILTER_XT_MATCH_CGROUP=y
+CONFIG_IP_VS_PROTO_TCP=y
+CONFIG_IP_VS_PROTO_UDP=y
+CONFIG_IP_VS_RR=y
+CONFIG_IP_VS_NFCT=y
+CONFIG_NET_CLS_CGROUP=y
+CONFIG_CGROUP_NET_PRIO=y
+CONFIG_MACVLAN=y
+CONFIG_IP_VS_WRR=y
+CONFIG_IP_VS_SH=y
+CONFIG_CGROUP_PERF=y
+CONFIG_ANDROID_PARANOID_NETWORK=n
```

Implementation Process

- SDK code download: https://www.t-firefly.com/doc/download/107.html

- Modify **kernel/arch/arm64/configs/firefly_defconfig and check config**
  - https://github.com/moby/moby/blob/master/contrib/check-config.sh
  - ./check-config.sh ./config.gz

- Modify **Docker overlay backing filesystem**
  - Expand the storage of RK3568 by connecting an SSD hard drive of micro SD card via PCIs, format the SSD card as f2fs.

- Resolve **Android network**
  ip route add default via 10.0.20.1 dev eth1; ip rule add from all table eth1 prio 3; iptables -D tetherctrl_FORWARD 1

- Download docker static binaries: https://download.docker.com/linux/static/stable/aarch64/
  If Android system is ARM 32-bit, ensure to select the ARMHF version. golang release static binary config /etc/resolv.conf

- Service auto-start：/system/etc/init/hw/init.rc

Implementation Process


```
130|rk3568:/ # docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

```
u202:/ # docker run -itd nginx
ad06c310d50ffa71db3e293e3d43d54b0eb6a2e9654e1341960d4b71732a197c
u202:/ #
u202:/ # docker ps
CONTAINER ID   IMAGE                                                    COMMAND                  CREATED          STATUS          PORTS                                       NAMES
ad06c310d50f   nginx                                                    "/docker-entrypoint.…"   4 seconds ago    Up 2 seconds    80/tcp                                      amazing_
cerf
e98228e5a811   harbor.thundercomm.com/kirnu/redis-arm:6.2.6-alpine      "docker-entrypoint.s…"   22 hours ago     Up 22 hours     0.0.0.0:6379→6379/tcp, :::6379→6379/tcp     edgex-re
dis
d8afc1a846e9   harbor.thundercomm.com/kirnu/tutk-arm:latest             "./P2PTunnelServer G…"   22 hours ago     Up 22 hours                                                 tutk
630e13180fa7   harbor.thundercomm.com/kirnu/osware/apigateway/gateway-setup-arm:2.0.0-18   "/go-one"   2 weeks ago   Up 13 days   0.0.0.0:9400→9000/tcp, :::9400→9000/tcp     gateway-
setup
741bbacc4b9f   harbor.thundercomm.com/kirnu/traefik-arm:v2.6-config     "/entrypoint.sh --ap…"   2 weeks ago      Up 13 days                                                  api-gate
way
b0e1fe46b047   harbor.thundercomm.com/kirnu/osware/storage-manager-arm:latest   "/bin/sh -c /storage…"   2 weeks ago   Up 12 days   127.0.0.1:3434→3434/tcp                     storage-
manager
ac3d99f5c47a   harbor.thundercomm.com/kirnu/osware/appengine/light-engine-go-arm:latest   "/lite_app_engine"   2 weeks ago   Up 13 days   0.0.0.0:7005→7005/tcp, :::7005→7005/tcp     light-en
gine
```

Implementation Process

Enable cloud-edge network service communication between **KubeEdge CloudCore** and **Android edge**

1. Use KubeEdge v1.10.0 release, compile static edgecore v1.10.0 used by Android;
2. Use kubeedge/edgemesh-{agent, server}:v1.9.0 image;
3. Pay attention to controlling the num of edgemesh-{agent,server} cpu requests, if the node resources are limited.

Compile KubeEdge static EdgeCore on Arm server

```
docker build -t kubeedge/edgecore:v1.10.0 -f build/edge/Dockerfile .
docker cp $(docker create --rm kubeedge/edgecore:v1.10.0):/usr/local/bin/edgecore ./edgecore.1.10.0
```

Centos : no rule EDGE-MESH-TCP DNAT to:169.254.96.16:40001 → Ubuntu 20.04 cloud master

Implementation Process


```
[root@iZ8vbg0k6lxlscxr1lphgmZ ~]# kubectl get node
NAME                      STATUS   ROLES                  AGE     VERSION
iz8vbg0k6lxlscxr1lphgmz   Ready    control-plane,master   123d    v1.21.5+k3s1
linaro-alip               Ready    agent,edge             16d     v1.22.6-kubeedge-v1.10.0
gandroid                  Ready    agent,edge             3d15h   v1.22.6-kubeedge-v0.0.0-master+$Format:%h$
1542131233004580876       Ready    agent,edge             9d      v1.22.6-kubeedge-v1.10.0
```

```
[root@iZ8vbg0k6lxlscxr1lphgmZ helm]# kubectl get pod -nkubeedge
NAME                                READY   STATUS    RESTARTS   AGE
edgemesh-server-57448c7f4-p5wvl     1/1     Running   0          18m
edgemesh-agent-52kdh                1/1     Running   0          18m
edgemesh-agent-24jnm                1/1     Running   1          18m
edgemesh-agent-mv4dv                1/1     Running   0          18m
```

```
ubuntu@VM-16-2-ubuntu:~/wenfc/kubeedge/edgemesh$ kubectl get pod -owide
I0711 19:24:25.194458 2361818 request.go:668] Waited for 1.105020581s due to client-side throttling, not priority and fairness, request: GET:https://12
7.0.0.1:6443/apis/k3s.cattle.io/v1?timeout=32s
NAME                        READY   STATUS    RESTARTS   AGE     IP           NODE            NOMINATED NODE   READINESS GATES
alpine-test                 1/1     Running   0          6m43s   10.42.0.94   vm-16-2-ubuntu  <none>           <none>
websocket-test              1/1     Running   0          6m43s   10.42.0.93   vm-16-2-ubuntu  <none>           <none>
hostname-edge-84cb45ccf4-n7g4f   1/1     Running   0          6m6s    172.17.0.2   gandroid        <none>           <none>
ubuntu@VM-16-2-ubuntu:~/wenfc/kubeedge/edgemesh$
```

```
ubuntu@VM-16-2-ubuntu:~/wenfc/kubeedge/edgemesh$ kubectl exec -it alpine-test -- sh
I0711 19:22:04.563303 2359749 request.go:668] Waited for 1.054851529s due to client-side throttling, not priority and fairness, request: GET:https://12
7.0.0.1:6443/apis/certificates.k8s.io/v1beta1?timeout=32s
/ #
/ # curl hostname-svc:12345
hostname-edge-84cb45ccf4-n7g4f
/ #
```

Summary and Outlook

For more details, see: https://github.com/ThunderSoft001/kubeedgeOnAndroid

Future Expansion Directions:

- RISC-V Architecture SBC (e.g. Allwinner) & Loongson SBC
- Further explore practical applications in a wider range of mobile edge scenarios. (Android, OpenHarmony and etc.)
- The third wave of Cloud-Native Technologies, such as WebAssembly (WasmEdge, Fermyon)
- etc.

Kernel modifications can be complex. Are there any alternative solutions that don't require kernel modifications?


KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023
Thank You!