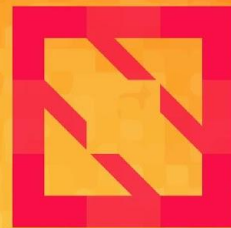




KubeCon



CloudNativeCon

North America 2019





KubeCon



CloudNativeCon

North America 2019

BYO Private 5G Network on Kubernetes

Frank Zdarsky, Red Hat

Raymond Knopp, Eurecom



\$> whoami



KubeCon



CloudNativeCon

North America 2019



Frank Zdarsky

- Sen. Principal Software Engineer, Office of the CTO, Red Hat
- Responsible for Edge Computing
- Contributor to OpenShift, OpenStack, OAI, ONAP, Akraino



Raymond Knopp

- Professor in Communication Systems @ EURECOM
- President of the OpenAirInterface Software Alliance
- Expert in Radio-Access Networks

What's The Fuzz about 5G?



KubeCon



CloudNativeCon

North America 2019

- **Ultra flexible radio access = 5G *New Radio* (NR)**
 - Higher bandwidth and spectral efficiency (bits/s/Hz/m²)
 - Bandwidth parts (tailor bandwidth to UE class)
 - New abstractions for service classification down to Layer 1 (slicing)
- **Radio Access Network compatibility with 4G and 5G cores**
 - 5G dual-connectivity (non-standalone operation)
 - Interconnection of evolved 4G eNodeB (ng-eNB) with 5G core
- **5G core cloud-native architecture, and evolulutory path to cloud-native radio-access, too**

New Verticals



KubeCon



CloudNativeCon

North America 2019

- **5G is 3GPP's answer for enabling new use cases**
 - beyond smartphones and traditional IoT applications
 - industrial AR, remote control of drones/farming vehicles, support for new vehicular services (NR-V2X), etc.
- **Address requirements not satisfied by today's WiFi or 4G-based solutions**
 - network density/scalability, resilience for critical communications
- **Private 5G**
 - evolution of private LTE exploiting the new 5G features (i.e. low-latency and ultra-reliable transmission, service classification through slicing)
 - licensed bands for non-public applications (e.g. Industry 4.0)
 - Customized IT + Radio solution (OS+Servers+Radio+Edge Computing)

How Does a 5G Network Look Like?



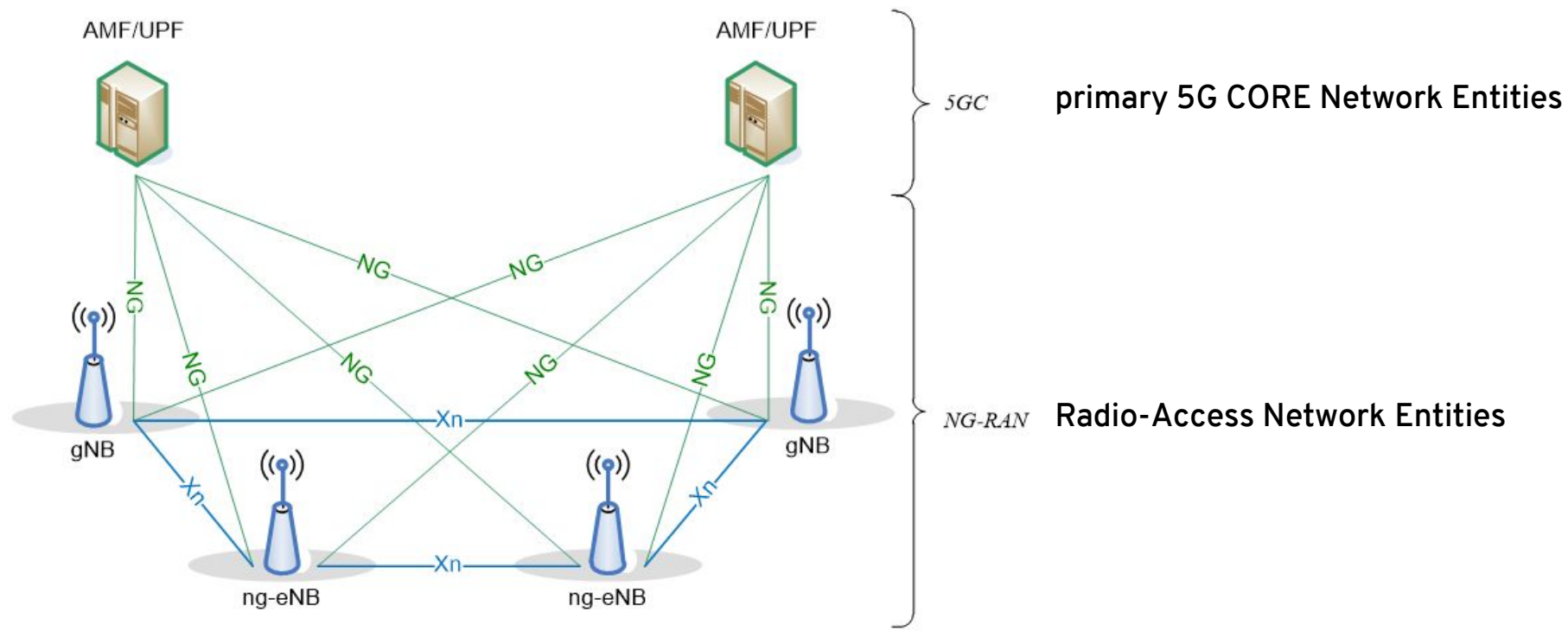
KubeCon



CloudNativeCon

North America 2019

Elements of “pure” 5G network



NSA vs SA... We're Doing NSA Here



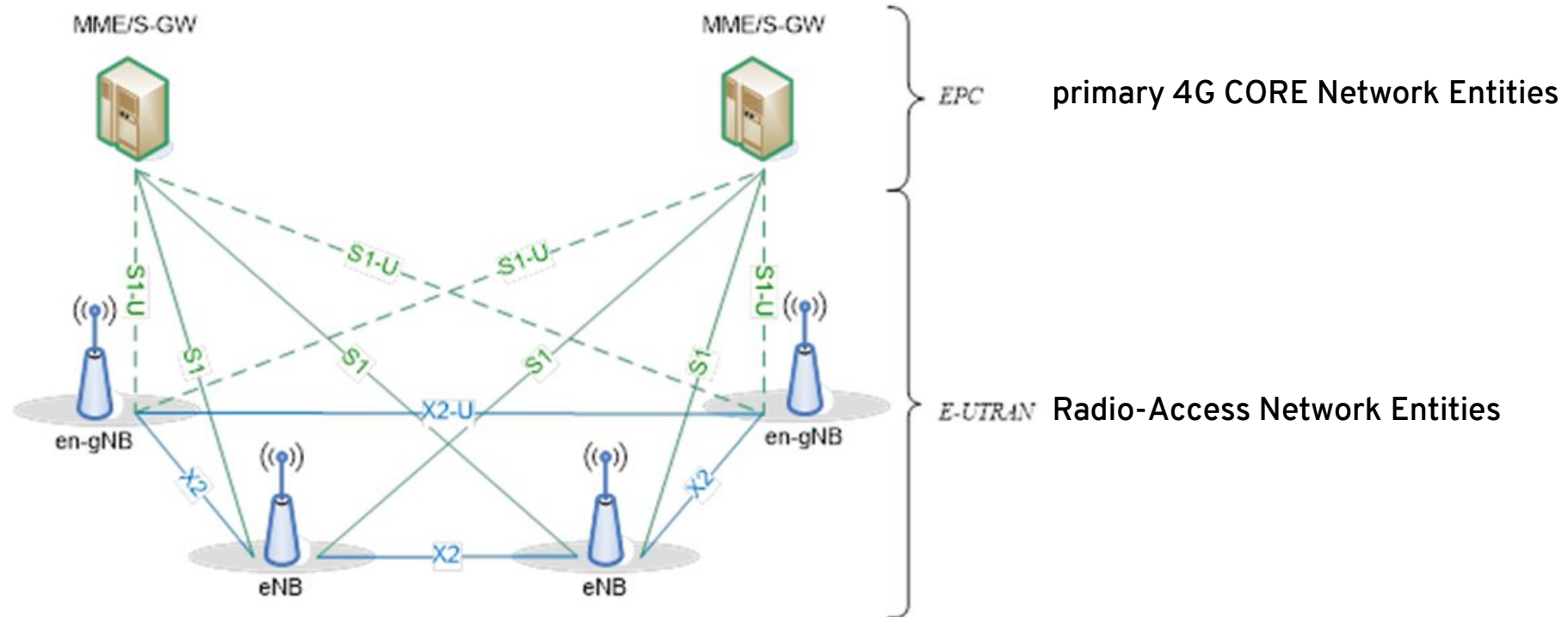
KubeCon



CloudNativeCon

North America 2019

Today's 5G network == Non-StandAlone, i.e. 5G has a 4G anchor (EN-DC - EUTRA-NR Dual-Connectivity) ⇒ essentially a 4G network with 5G hotspots (user-plane only)



OAI and Friends



KubeCon



CloudNativeCon

North America 2019

It's become feasible to put a fully compliant 4G/5G eNodeB/gNodeB and EPC/5GC in a commodity x86 box

- Emergence of “radio-hackers” and development/user communities experimenting with 3GPP software implementations
- Types of software
 - Amarisoft (closed, commercial)
 - OAI (open-source, 3GPP-friendly), ORAN (partially open-source, 3GPP-friendly)
 - srsLTE/openLTE (open-source, 3GPP-unfriendly)

About OAI Alliance



KubeCon



CloudNativeCon

North America 2019

- Founded in 2014 as a “Fond de Dotation” = Endowment Fund
- 3GPP strategic members in 2015-2019 (Orange Labs, TCL 5G Lab, Nokia Bell Labs, Fujitsu, Technicolor, Interdigital, Qualcomm, PAWR)
- Many associate members (Red Hat, Cisco, Samsung USA, Renault, Kyocera, Viettel, Blackned, B-COM, INRIA, IMT, Fraunhofer, Rutgers WINLAB, U. Utah, BUPT, etc.)
- Non-member donors: Facebook
- Donations are to maintain an engineering support team
 - CI/CD
 - Community management/building
 - Industry relations



KubeCon



CloudNativeCon

North America 2019

Let's Build This!

Low-End Prototyping Hardware



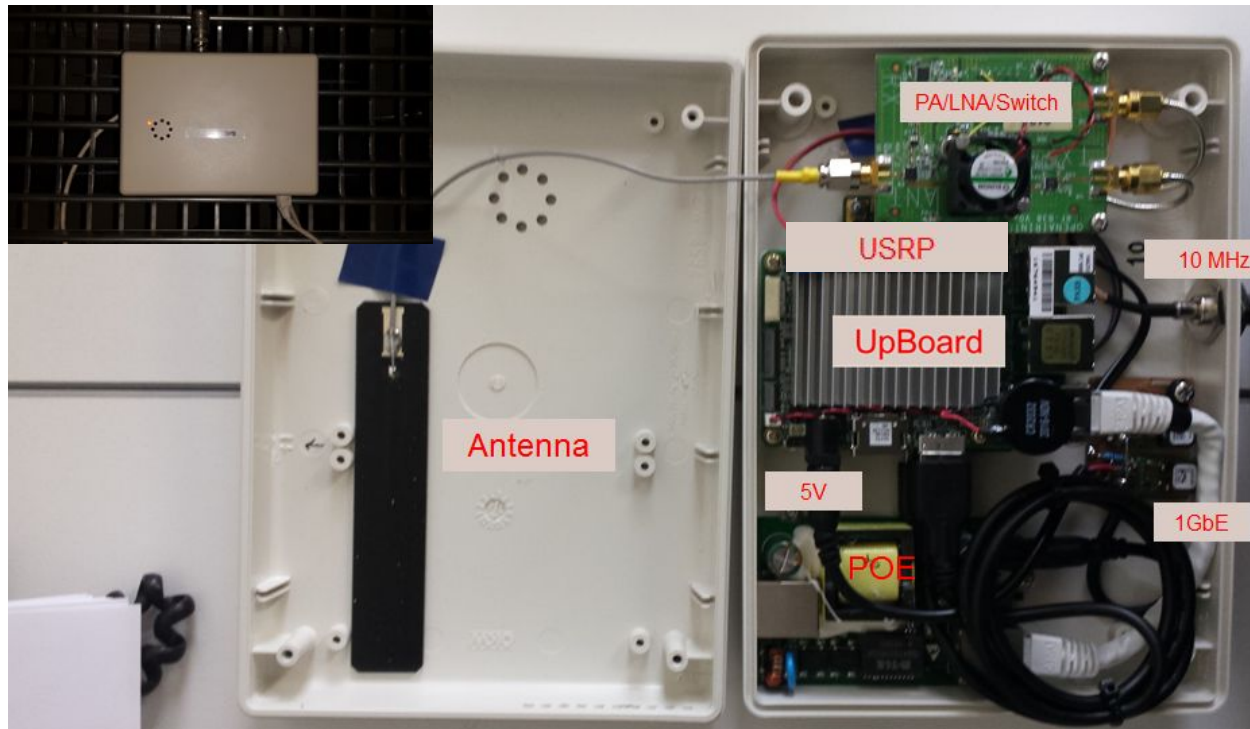
KubeCon



CloudNativeCon

North America 2019

< 50 MHz BW



Shopping List:

- USRP B200-mini (\$500)
 - up to 50 MHz BW
- custom 20 dBm PA/LNA/Switch (\$300) - band 38, 42/43, n38/n77-78
- Upboard/Upboard2 (low-end \$90 PC)
- GbE fronthaul POE+
- Antenna
- optional GPSDO

High-End Prototyping Hardware



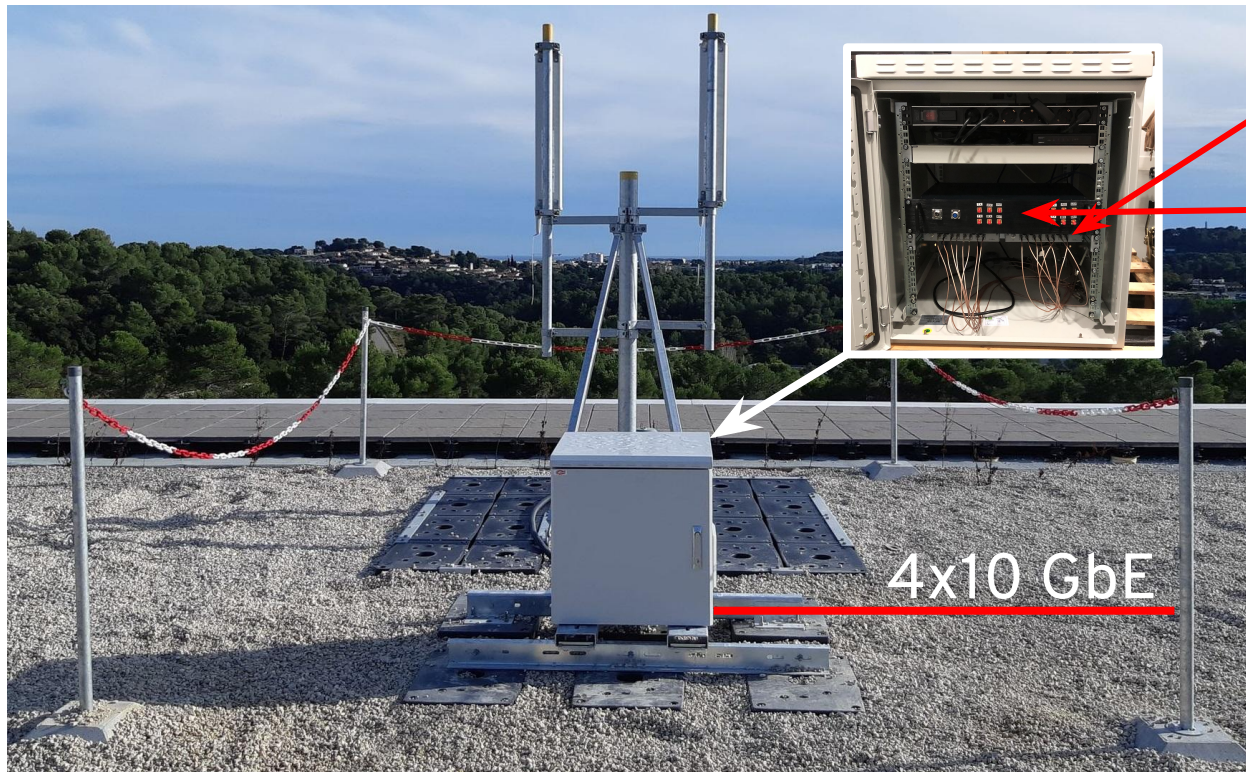
KubeCon



CloudNativeCon

North America 2019

8 antenna, 100 MHz (FR1)



4x10 GbE

Shopping List:

- two USRP N310 (~\$20000)
- up to 100 MHz BW, 8 antennas in total
- eight 2W PA/LNA/Switch (~\$2500) - 2.6 or 3.5 GHz bands, e.g.
www.zhixun-wireless.top
- 10 GbE optical fronthaul
- two 4-port Kathrein Antennas
- GPS antenna for N310s

Production-Level Hardware



KubeCon



CloudNativeCon

North America 2019

Commercial radio units

- eCPRI/ORAN Ethernet-based fronthaul solutions can be “commodity” and “whitebox” for standard interconnection with switching fabric

<http://aw2s.com/RRU.html>

<https://benetel.com/product/ran-remote-radio-unit-rru/>

- High-power (43 dBm); can cost less than high-end prototype described earlier.
- Less generic/flexible but complete product ready for deployment in specific bands.



BIOS Configuration



KubeCon



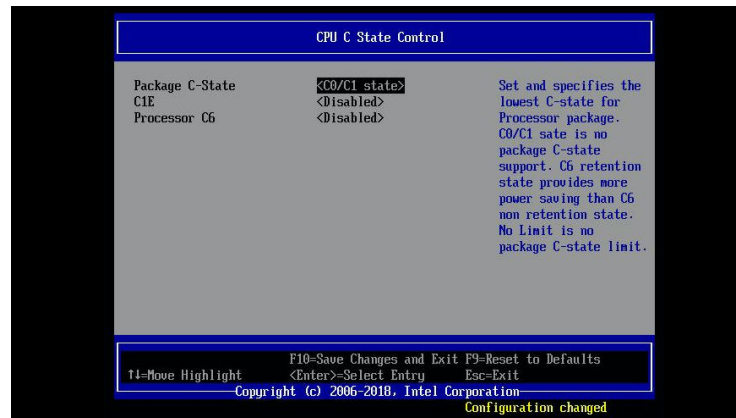
CloudNativeCon

North America 2019

Eliminate HW- and firmware-level sources of non-determinism

- disable C-states (CPU power save)
- disable P-states (CPU freq. scaling)
- disable EDAC (ECC memory scans)
- don't touch SMIs!

Run `hwlatdetect` for 24h to detect HW/firmware-induced latency spikes; no OS-level tuning can fix these!



```
[core@babylon ~]$ sudo hwlatdetect --duration=1d
hwlatdetect: test duration 86400 seconds
detector: tracer
parameters:
  Latency threshold: 10us
  Sample window:    1000000us
  Sample width:     500000us
  Non-sampling period: 500000us
  Output File:      None

Starting test
test finished
Max Latency: Below threshold
Samples recorded: 0
Samples exceeding threshold: 0
```


Host OS Configuration



KubeCon



CloudNativeCon

North America 2019

Use real-time pre-empt kernel.

Specify huge pages and isolcpus cores in `/etc/tuned/realtime-variables.conf`, use `'tuned-adm profile realtime'` to auto-configure system parameters.

Verify OS indeed shows p- and c-states disabled.

```
[core@babylon ~]# sudo echo "isolated_cores=4-23" >>
/etc/tuned/realtime-variables.conf
[core@babylon ~]# sudo tuned-adm profile realtime && sudo reboot

[core@babylon ~]$ uname -a
Linux babylon 4.18.0-80.11.2.rt9.157.el8_0.x86_64 #1 SMP PREEMPT RT
Mon Sep 16 15:45:17 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux

[core@babylon ~]$ cat /proc/cmdline
BOOT_IMAGE=[...] skew_tick=1 isolcpus=4-23 intel_pstate=disable
nosoftlockup nmi_watchdog=0 audit=0 mce=off kthread_cpus=0
irqaffinity=0 skew_tick=1 processor.max_cstate=1 idle=poll
intel_idle.max_cstate=0 intel_pstate=disable intel_iommu=on
iommu=pt default_hugepagesz=1G hugepagesz=1G hugepages=10 nohz=on
nohz_full=4-24 rcu_nocbs=4-24

[core@babylon ~]$ sudo cpupower monitor
          | Nehalem
PKG | CORE | CPU | C3 | C6 | PC3 | PC6 | Mperf | C0 | Cx | Freq
  0 |   0 |   0 | 0.00 | 0.00 | 0.00 | 0.00 | 99.97 | 0.03 | 3000
  0 |   1 |   4 | 0.00 | 0.00 | 0.00 | 0.00 | 99.97 | 0.03 | 3000
  0 |   2 |   8 | 0.00 | 0.00 | 0.00 | 0.00 | 99.97 | 0.03 | 3000
  0 |   3 |   6 | 0.00 | 0.00 | 0.00 | 0.00 | 99.97 | 0.03 | 3000
[...]
```

K8s Config: Machines



KubeCon



CloudNativeCon

North America 2019

OpenShift enables declarative management of machines and host OS via the Machine API.

To apply the RT-config automatically,

- define a MachineConfig for RT-tuning via a one-shot systemd unit

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker-rt
  name: machine-config-worker-rt
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
        - contents:
            source: data:text/plain;base64,[...]
            filesystem: root
            mode: 0777
            path: /opt/setup_rt.sh
    systemd:
      units:
        - contents: |
            [Unit]
            After=network-online.target
            ConditionPathExists=!/opt/rt_executed
            [Service]
            Type=oneshot
            ExecStart=/opt/setup_rt.sh
            [Install]
            WantedBy=multi-user.target
            enabled: true
            name: install_realtime.service
```

K8s Config: Machines



KubeCon



CloudNativeCon

North America 2019

OpenShift enables declarative management of machines and host OS via the Machine API.

To apply the RT-config automatically,

- define a MachineConfig for RT-tuning via a one-shot systemd unit
- select it to the MachineConfigPool applied to all nodes labeled with the node-role “worker-rt”

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker-rt
  name: machine-config-worker-rt
spec:
  config:
    ignition:
      apiVersion: machineconfiguration.openshift.io/v1
      kind: MachineConfigPool
      metadata:
        name: worker-rt
      spec:
        machineConfigSelector:
          matchExpressions:
            - {key: machineconfiguration.openshift.io/role,
              operator: In, values: [worker,worker-rt]}
        maxUnavailable: null
        nodeSelector:
          matchLabels:
            node-role.kubernetes.io/worker-rt: ""
        paused: false

[Service]
Type=oneshot
ExecStart=/opt/setup_rt.sh
[Install]
WantedBy=multi-user.target
enabled: true
name: install_realtime.service
```

K8s Config: CPU Resource Mgmt.



KubeCon



CloudNativeCon

North America 2019

To place a real-time workload (e.g. `cyclictest`) on isolated cores on the worker-rt node:

- configure the “static” `cpuManagerPolicy` on the Kubelet
- set resource requests and limits for *both CPU and memory* resources

```
apiVersion: v1
kind: Pod
metadata:
  name: cyclictest
spec:
  containers:
  - name: cyclictest
    image: docker.io/cscojianzhan/cyclictest
    resources:
      limits:
        cpu: 4
        memory: "400Mi"
      requests:
        cpu: 4
        memory: "400Mi"
    securityContext:
      capabilities:
        add:
        - SYS_NICE
        - SYS_RAWIO
        - IPC_LOCK
    volumeMounts:
    - mountPath: /dev/cpu_dma_latency
      name: cstate
  nodeSelector:
    node-role.kubernetes.io/worker-rt: ""
  volumes:
  - name: cstate
    hostPath:
      path: /dev/cpu_dma_latency
```

```
apiVersion: machineconfiguration.openshift.io/v1
kind: KubeletConfig
metadata:
  name: cpumanager-enabled
spec:
  machineConfigPoolSelector:
    matchLabels:
      custom-kubelet: cpumanager-enabled
  kubeletConfig:
    cpuManagerPolicy: static
    cpuManagerReconcilePeriod: 5s
    kubeReserved:
      cpu: "1"
```

K8s Config: Networking



KubeCon

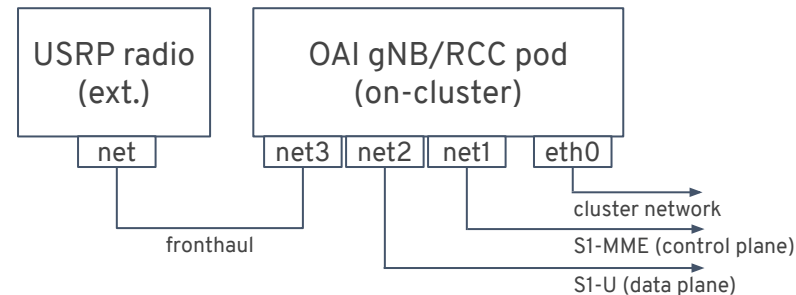


CloudNativeCon

North America 2019

We use Multus CNI to segregate the 3GPP control and data plane networks from the cluster network (used for mgmt.).

eNB and gNB pods are connected to the USRP software-defined radios via dedicated interfaces.



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: oai-gnb
  labels:
    app: oai-gnb
spec:
  selector:
    matchLabels:
      app: oai-gnb
  template:
    metadata:
      labels:
        app: oai-gnb
      annotations:
        k8s.v1.cni.cncf.io/networks:
          control, data, fronthaul
    spec:
      [...]
```

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: fronthaul
spec:
  config: '{
    "cniVersion": "0.3.0",
    "type": "macvlan",
    "master": "bond0",
    "mode": "bridge",
    "ipam": {
      "type": "static",
      "addresses": [
        {"address": "192.168.18.206/24",
          "gateway": "192.168.18.1"}
      ]
    }
  }'
```

Deploying OAI



KubeCon



CloudNativeCon

North America 2019

Clone openair-k8s repo* from Github.

On a RHEL host, build OAI images and
push to local cluster registry:

```
hack/build_images
```

```
hack/push_images $your_cluster_registry
```

Adapt config to your deployment.

Deploy:

```
kubectl apply -k manifests
```

*) <https://github.com/openairinterface/openair-k8s>



KubeCon



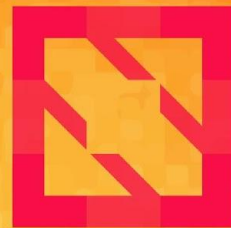
CloudNativeCon

North America 2019

Let's Demo This!



KubeCon



CloudNativeCon

North America 2019

