LF NETWORKING
Virtual Developer & Testing Forum
June 22 - 25, 2020

Reference CNF development journey and outcomes

Victor Morales

v.morales@samsung.com

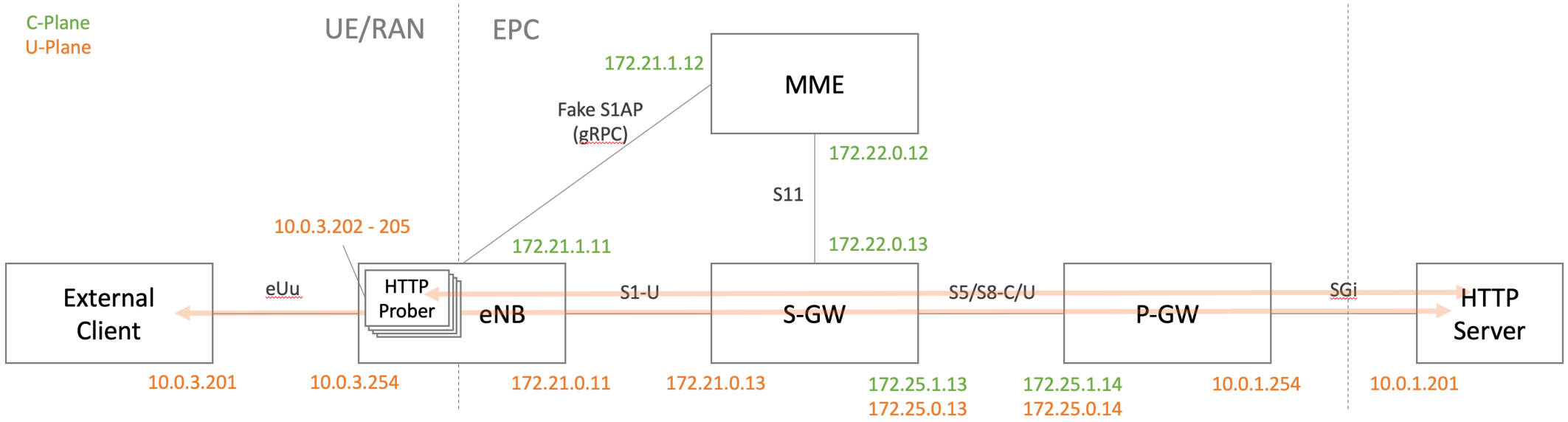
LF NETWORKING

Anti-Trust Policy Notice

› Linux Foundation meetings involve participation by industry competitors, and it is the intention of the Linux Foundation to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.

› Examples of types of actions that are prohibited at Linux Foundation meetings and in connection with Linux Foundation activities are described in the Linux Foundation Antitrust Policy available at http://www.linuxfoundation.org/antitrustpolicy. If you have questions about these matters, please contact your company counsel, or if you are a member of the Linux Foundation, feel free to contact Andrew Updegrove of the firm of Gesmer Updegrove LLP, which provides legal counsel to the Linux Foundation.

GW Tester



Features

- Multiple networks
- Isolated network
- Microservices
- Boot order / Dependency management
- Overlay networking

https://github.com/wmnsk/go-gtp/tree/master/examples/gw-tester
https://github.com/cncf/cnf-testbed/tree/master/examples/use_case/gogtp-k8s

Phase 1 - Containerization

Dockerfile – Minimal size

```
FROM golang:1.14-alpine3.11 as builder

RUN apk --no-cache add git
RUN go get -v github.com/wmnsk/go-gtp/examples/gw-tester/enb

FROM alpine:3.11

COPY --from=builder /go/bin/enb /usr/local/bin/
COPY ./enb_default.yml /etc/enb.yml

ENTRYPOINT ["/usr/local/bin/enb", "-config", "/etc/enb.yml"]
```

Multi-stage builds
(~560MB per image)

https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
https://12factor.net/

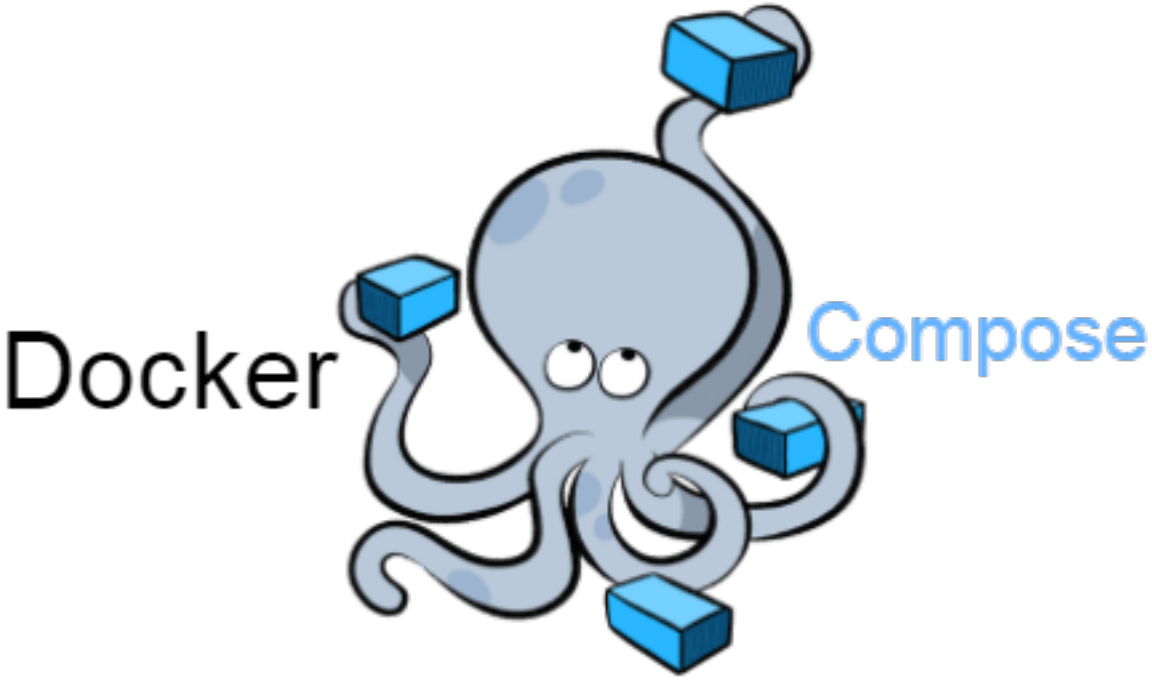
Dockerfile – Minimal Permissions

```
ENV APP_ROOT=/opt/gw-tester
COPY --from=builder /go/bin/mme ${APP_ROOT}/bin/
COPY ./mme_default.yml /etc/gw-tester/mme.yml

RUN chmod -R u+x ${APP_ROOT} && \
    chmod -R g=u ${APP_ROOT} /etc/gw-tester && \
    chgrp -R 0 ${APP_ROOT}

USER 10001
WORKDIR ${APP_ROOT}

EXPOSE 36412/udp
EXPOSE 2123/udp
```

Rootless

https://github.com/GoogleContainerTools/distroless

Phase 2 - Aggregation



Docker Compose – Compute

```yaml
services:
  enb:
    image: electrocucaracha/enb
    build:
      context: ../enb
    volumes:
      - ../enb/enb_default.yml:/etc/gw-tester/enb.yml
    networks:
      lte-s1u:
        ipv4_address: 172.21.0.11
      lte-euu:
        ipv4_address: 10.0.3.254
      lte-s1c:
        ipv4_address: 172.21.1.11
    depends_on:
      - sgw
      - mme
      - pgw
    cap_add:
      - NET_ADMIN
```

Multiple networks

Boot order / Dependency management

Minimal permissions

Docker Compose – Network

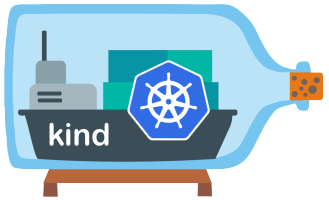
```yaml
version: '2.4'

networks:
  lte-euu:
    driver: overlay
    driver_opts:
      com.docker.network.driver.overlay.vxlanid_list: 2
    ipam:
      driver: default
      config:
      - subnet: 10.0.3.0/24
        ip_range: 10.0.3.128/24
  lte-sgi:
    driver: overlay
    driver_opts:
      com.docker.network.driver.overlay.vxlanid_list: 3
    internal: true # This network must be internal
    ipam:
      driver: default
      config:
      - subnet: 10.0.1.0/24
        ip_range: 10.0.1.128/24
```

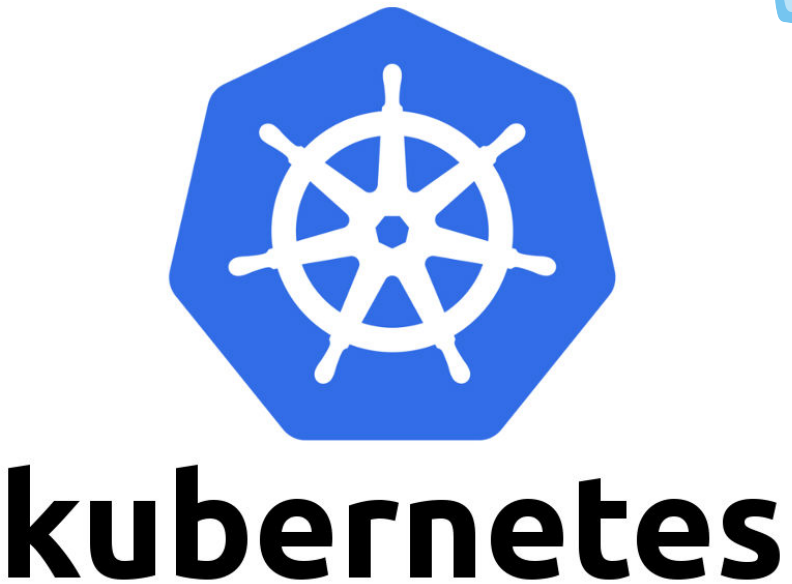
VNI selection

Isolated network

Internal networks creates an externally isolated overlay network

https://docs.docker.com/compose/compose-file/compose-file-v2/#internal





Phase 3 - Deployment

Kubernetes – Init Containers

```
services:
  http_server:
    image: python:3.8.2-alpine3.11
    networks:
      lte-sgi:
        ipv4_address: 10.0.1.201
    depends_on:
      - pgw
    command: [sh, -c, "ip route add 10.0.3.0/24 via 10.0.1.254 && python -m http.server 80"]
    cap_add:
      - NET_ADMIN
```

```
initContainers:
  - name: configure
    image: busybox
    securityContext:
      capabilities:
        add: ["NET_ADMIN"]
    command: ["ip", "route", "add", "10.0.3.0/24", "via", "$(pgw_sgi_ip)"]
    env:
      - name: pgw_sgi_ip
        value: $PGW_SGI_IP
containers:
  - image: python:3.8.2-alpine3.11
    name: http-server
    command: ["python", "-m", "http.server", "80"]
```

https://github.com/docker/compose/issues/6855

https://kubernetes.io/docs/concepts/workloads/pods/init-containers/

Kubernetes - Boot order / Dependency management

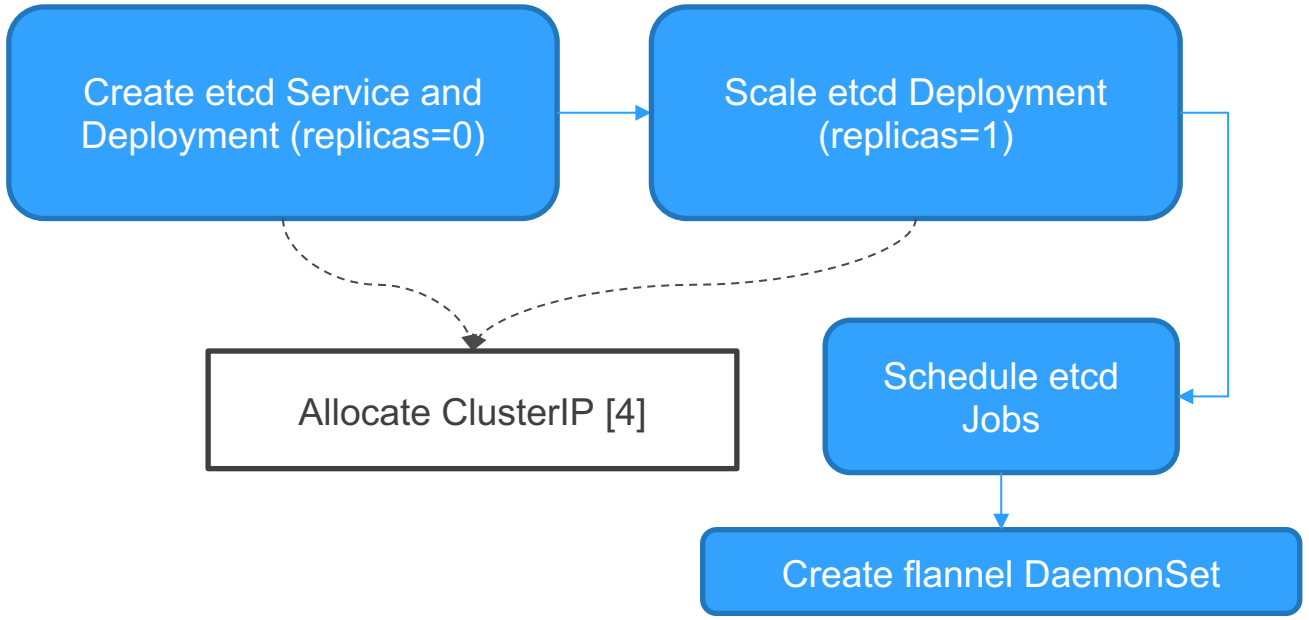
```
# Deploy SAE-GW helm charts
if [ -n "${PKG_MGR:-}" ] && [ "${PKG_MGR:-}" == "helm" ]; then
    helm install saegw "./${multi_cni}/charts/saegw/"
    for chart in pgw sgw; do
        kubectl rollout status "deployment/saegw-$chart"
    done
else
    for pod in pgw sgw; do
        kubectl apply -f "./${multi_cni}/${pod}.yml"
        kubectl wait --for=condition=ready pod "$pod" --timeout=120s
    done
fi
```

flannel

Phase 3.a – Overlay Network

Flannel – Deployment known issues

1. A single daemon **does not support** running multiple networks[1].
2. Multi-network deployment in Kubernetes[2] requires **ETCD v2** [3].




```
containers:
- name: etcdctl-lte-euu
  image: quay.io/coreos/etcd:v3.3.20
  env:
  - name: ETCDCTL_API
    value: "2"
  command: ["etcdctl"]
  args:
  - --no-sync
  - '--endpoint=http://$(FLANNEL_ETCD_SERVICE_HOST):12379'
  - set
  - '/euu/network/config'
  - '{ "Network": "10.0.3.0/24", "Backend": {"Type": "vxlan", "VNI": 2}}'
```

[1] https://github.com/coreos/flannel/blob/master/Documentation/running.md#multiple-networks
[2] https://dzone.com/articles/how-to-understand-and-setup-kubernetes-networking
[3] https://github.com/kubernetes/kubernetes/issues/57354#issuecomment-451679688
[4] https://kubernetes.io/docs/concepts/services-networking/connect-applications-service/#accessing-the-service

Flannel – Deployment known issues

SubnetLen (integer): The size of the subnet allocated to each host. Defaults to 24 (i.e. /24) unless Network was configured to be smaller than a /24 in which case it is one less than the network.

```
vagrant@ubuntu1804:/vagrant$ kubectl logs -n kube-system lte-euu-flannel-ds-zgpdg
I0604 19:44:28.249368       1 main.go:518] Determining IP address of default interface
I0604 19:44:28.249732       1 main.go:531] Using interface with name eth0 and address 172.80.1.3
I0604 19:44:28.249745       1 main.go:548] Defaulting external address to interface address (172.80.1.3)
I0604 19:44:28.249813       1 main.go:246] Created subnet manager: Etcd Local Manager with Previous Subnet: None
I0604 19:44:28.249819       1 main.go:249] Installing signal handlers
I0604 19:44:28.257125       1 main.go:390] Found network config - Backend type: vxlan
I0604 19:44:28.257231       1 vxlan.go:121] VXLAN config: VNI=2 Port=0 GBP=false Learning=false DirectRouting=false
I0604 19:44:28.291194       1 local_manager.go:234] Picking subnet in range 10.0.3.64 ... 10.0.3.192
E0604 19:44:28.291292       1 main.go:291] Error registering network: failed to acquire lease: out of subnets
I0604 19:44:28.291329       1 main.go:370] Stopping shutdownHandler...
```

```
vagrant@ubuntu1804:/vagrant$ docker exec -ti k8s-worker cat /run/flannel/lte-euu.env
FLANNEL_NETWORK=10.0.3.0/24
FLANNEL_SUBNET=10.0.3.193/26
FLANNEL_MTU=1450
FLANNEL_IPMASQ=true
vagrant@ubuntu1804:/vagrant$ docker exec -ti k8s-worker2 cat /run/flannel/lte-euu.env
FLANNEL_NETWORK=10.0.3.0/24
FLANNEL_SUBNET=10.0.3.129/26
FLANNEL_MTU=1450
FLANNEL_IPMASQ=true
vagrant@ubuntu1804:/vagrant$ docker exec -ti k8s-worker3 cat /run/flannel/lte-euu.env
FLANNEL_NETWORK=10.0.3.0/24
FLANNEL_SUBNET=10.0.3.65/26
FLANNEL_MTU=1450
FLANNEL_IPMASQ=true
```

Race condition during the Flannel deployment

K8s Setup:

- 1 Control plane node
- +2 Worker nodes

https://github.com/coreos/flannel/blob/v0.12.0/Documentation/configuration.md

Flannel – Runtime known issues

kube-scheduler doesn't have scheduling policies for Flannel's subnets.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: http-server
  annotations:
    k8s.v1.cni.cncf.io/networks: '[
      { "name": "lte-sgi", "ips": [ "10.0.1.201" ] }
    ]'
spec:
```

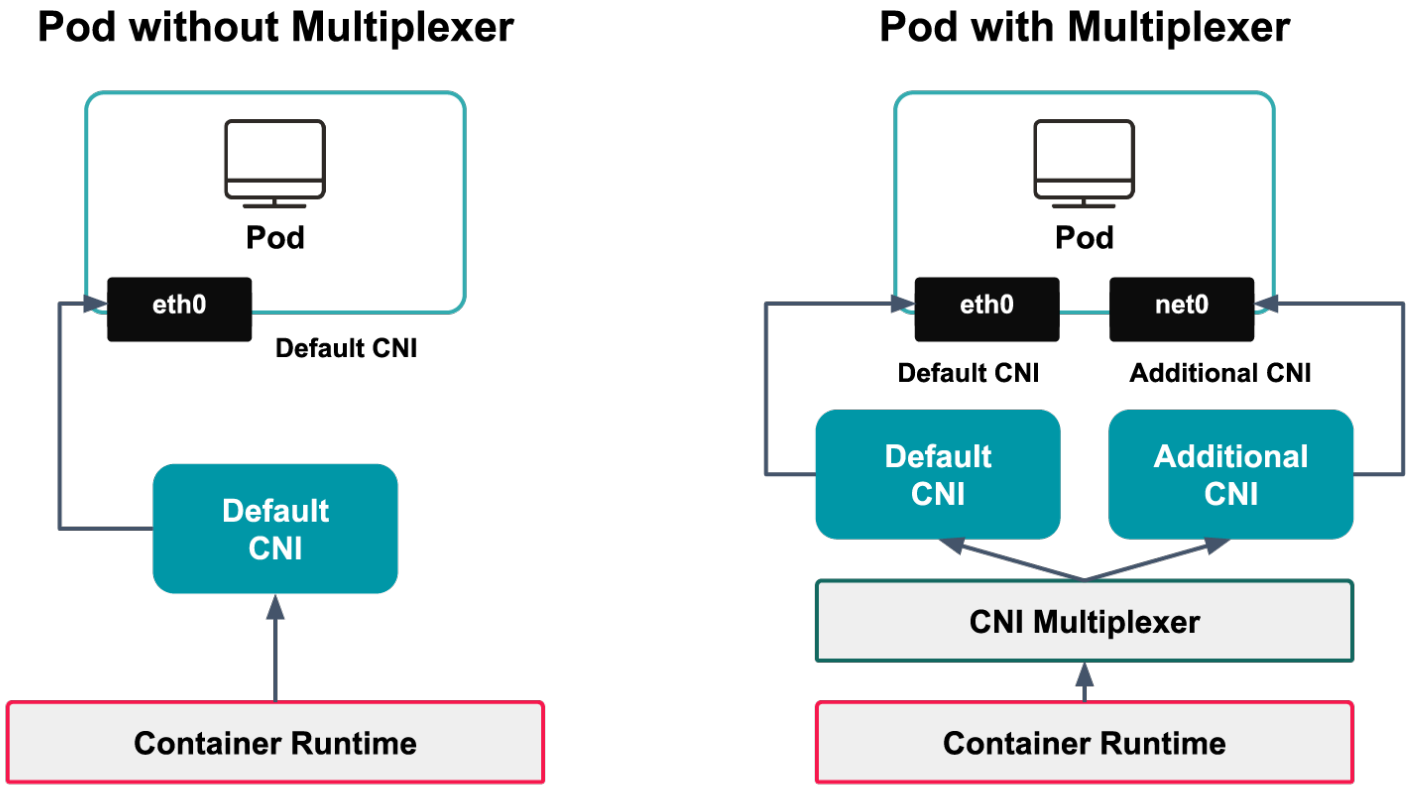
Don't use IP Static Address in a Multi-node Setup

```
root@k8s-worker:/# cat /run/flannel/lte-sgi.env
FLANNEL_NETWORK=10.0.1.0/24
FLANNEL_SUBNET=10.0.1.129/26
FLANNEL_MTU=1450
FLANNEL_IPMASQ=true
```

```
root@k8s-worker2:/# cat /run/flannel/lte-sgi.env
FLANNEL_NETWORK=10.0.1.0/24
FLANNEL_SUBNET=10.0.1.193/26
FLANNEL_MTU=1450
FLANNEL_IPMASQ=true
```

https://kubernetes.io/docs/reference/scheduling/policies/

Kubernetes – Multiple Networks



https://codilime.com/kubernetes-workloads-using-multiple-networks/

MULTUS

DANM

https://github.com/intel/multus-cni

https://github.com/nokia/danm

Multus – Multiple networks

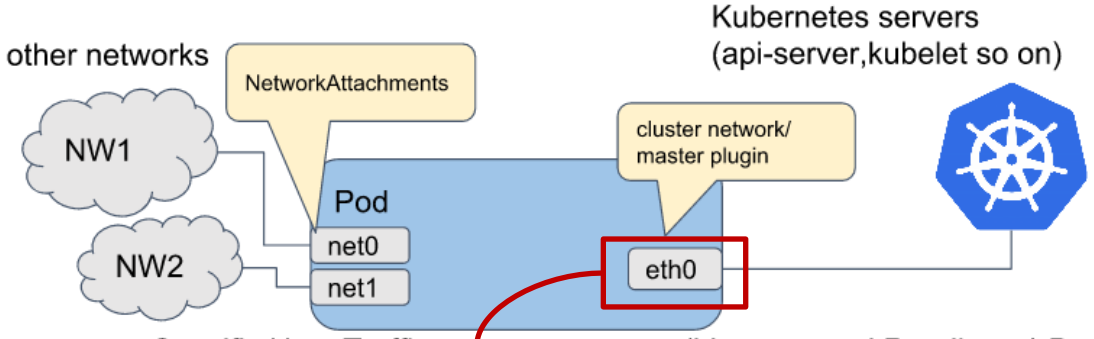
```yaml
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: lte-euu
spec:
  config: '{
      "cniVersion": "0.3.1",
      "type": "flannel",
      "subnetFile": "/run/flannel/lte-euu.env",
      "dataDir": "/var/lib/cni/flannel_euu",
      "delegate": {
        "bridge": "kbr1"
      }
    }'
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: enb
  annotations:
    k8s.v1.cni.cncf.io/networks: |
      [
        {"name": "lte-euu", "interface": "euu1"},
        {"name": "lte-s1c", "interface": "s1c2"},
        {"name": "lte-s1u", "interface": "s1u3"}
      ]
```

NIC definition

```yaml
annotations:
  v1.multus-cni.io/default-network: lte-sgi
```

Implicit default network



- Specific User Traffic
- (Liveness and Readiness) Probes
- Communication between API and Pod

DANM – Multiple networks

```yaml
apiVersion: danm.k8s.io/v1
kind: ClusterNetwork
metadata:
  name: lte-euu
spec:
  NetworkID: euu
  NetworkType: flannel
  Options:
    container_prefix: euu
```

NIC definition

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: enb
  annotations:
    danm.k8s.io/interfaces: |
      [
        {"clusterNetwork":"default"},
        {"clusterNetwork":"lte-euu"},
        {"clusterNetwork":"lte-s1c"},
        {"clusterNetwork":"lte-s1u"}
      ]
  labels:
    app.kubernetes.io/name: enb
    network: e-utran
```

Explicit default network

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: enb
  annotations:
    k8s.v1.cni.cncf.io/networks: |
      [
        {"name": "lte-euu", "interface": "euu1"},
        {"name": "lte-s1c", "interface": "s1c2"},
        {"name": "lte-s1u", "interface": "s1u3"}
      ]
    danm.k8s.io/interfaces: |
      [
        {"clusterNetwork":"default"},
        {"clusterNetwork":"lte-euu"},
        {"clusterNetwork":"lte-s1c"},
        {"clusterNetwork":"lte-s1u"}
      ]
  labels:
    app.kubernetes.io/name: enb
    network: e-utran
spec:
```

Whenever is possible use both

https://github.com/electrocucaracha/gw-tester/blob/master/k8s/enb.yml

Dependency management

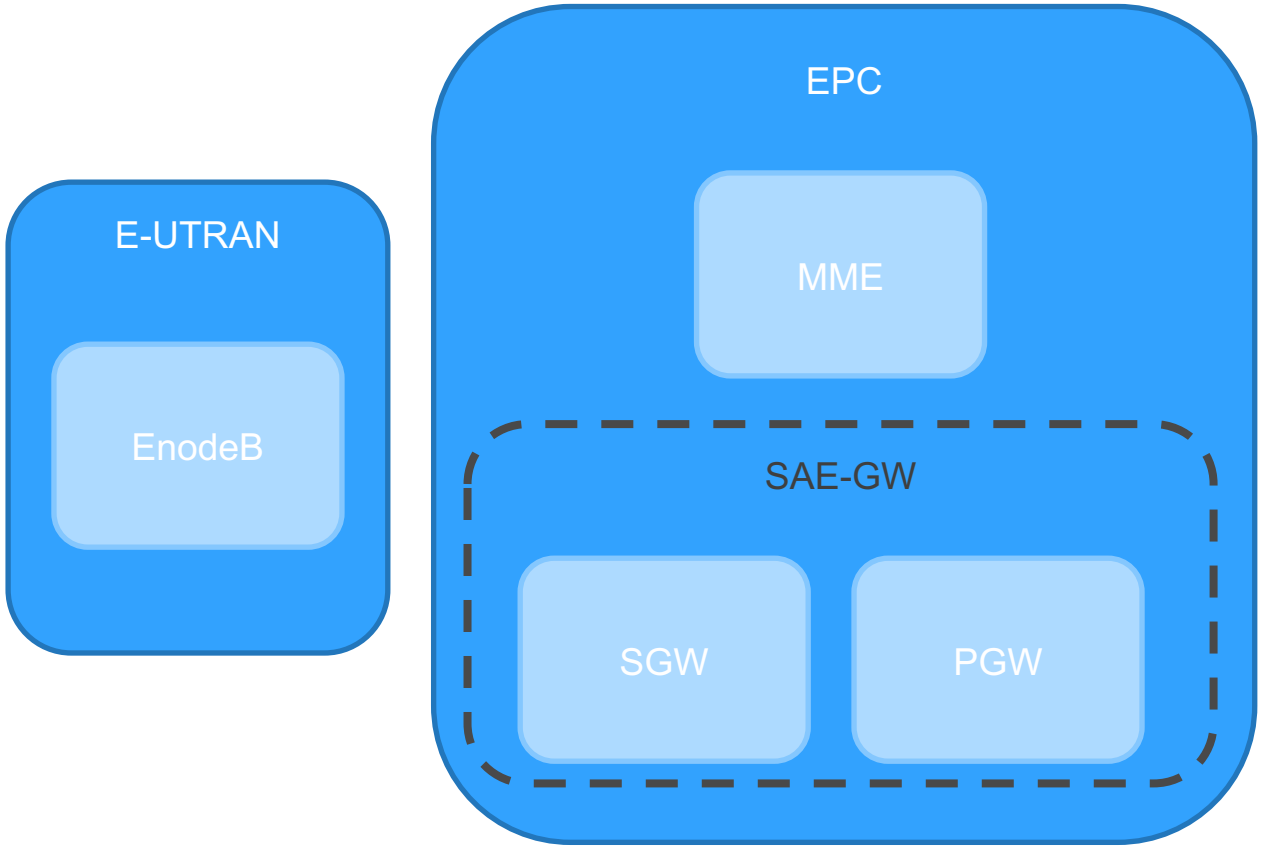
https://github.com/k8snetworkplumbingwg/network-attachment-definition-client/blob/release-1.16/pkg/utils/net-attach-def.go#L63

```
if [ "$multi_cni" == "multus" ]; then
    MME_S1C_IP=$(kubectl get pods -l=app.kubernetes.io/name=mme \
    -o jsonpath='{.items[0].metadata.annotations.k8s\.v1\.cni\.cncf\.io/networks-status}' \
    | jq -r '.[] | select(.name=="lte-s1c").ips[0]')
else
    MME_S1C_IP=$(kubectl get pods -l=app.kubernetes.io/name=mme \
    -o jsonpath='{range .items[0].status.podIPs[*]}{.ip}{"\n"}' \
    | grep "172.21.1")
fi
```

https://github.com/nokia/danm/blob/v4.2.0/pkg/danmep/danmep.go#L355

Phase 4 - Packaging

Helm – Manage dependencies via charts/




```
.
├── enb
│   ├── Chart.yaml
│   ├── charts
│   ├── templates
│   │   ├── NOTES.txt
│   │   ├── _helpers.tpl
│   │   └── deployment.yaml
│   └── values.yaml
├── mme
│   ├── Chart.yaml
│   ├── charts
│   ├── templates
│   │   ├── NOTES.txt
│   │   ├── _helpers.tpl
│   │   └── deployment.yaml
│   └── values.yaml
└── saegw
    ├── Chart.yaml
    ├── charts
    │   ├── pgw
    │   │   ├── Chart.yaml
    │   │   ├── charts
    │   │   ├── templates
    │   │   └── values.yaml
    │   └── sgw
    │       ├── Chart.yaml
    │       ├── charts
    │       ├── templates
    │       └── values.yaml
    └── values.yaml
```

https://helm.sh/docs/topics/charts/#managing-dependencies-manually-via-the-charts-directory

Helm – Chart installation

```
# Deploy MME service
SGW_S11_IP=$(kubectl get pods -l=app.kubernetes.io/name=sgw \
    -o jsonpath='{.items[0].metadata.annotations.k8s\.v1\.cni\.cncf\.io/networks-status}' \
    | jq -r '.[] | select(.name=="lte-s11").ips[0]')
PGW_S5C_IP=$(kubectl get pods -l=app.kubernetes.io/name=pgw \
    -o jsonpath='{.items[0].metadata.annotations.k8s\.v1\.cni\.cncf\.io/networks-status}' \
    | jq -r '.[] | select(.name=="lte-s5c").ips[0]')
if [ -n "${PKG_MGR:-}" ] && [ "${PKG_MGR:-}" == "helm" ]; then
    helm install mme ./charts/mme --set sgw.s11.ip="$SGW_S11_IP" \
    --set pgw.s5c.ip="$PGW_S5C_IP"
    kubectl rollout status deployment/mme
else
    export SGW_S11_IP PGW_S5C_IP
    envsubst \$PGW_S5C_IP,\$SGW_S11_IP < mme.yml | kubectl apply -f -
    kubectl wait --for=condition=ready pod mme
fi
```

```
vagrant@ubuntu1804:/vagrant$ helm ls
NAME    NAMESPACE    REVISION    UPDATED                                   STATUS      CHART        APP VERSION
enb     default      1           2020-05-21 19:16:46.765907666 +0000 UTC   deployed    enb-0.1.0    0.7.5
mme     default      1           2020-05-21 19:16:34.044353509 +0000 UTC   deployed    mme-0.1.0    0.7.5
saegw   default      1           2020-05-21 19:16:17.525839369 +0000 UTC   deployed    saegw-0.1.0  0.7.5
```

Q&A