Emulating Network Topologies in k8s

Marcus Hines (hines@google.com)
Rob Shakir (robjs@google.com)
on behalf of Google and OpenConfig
Network Topologies? k8s? WTF?

- **Why?** Emulating networks for fun and profit.
- **What?** Introducing KNE.
- **How?** What makes up an emulated topology?
- **Huh?** A real-world use case.
Disclaimer!

We’re presenting on behalf of a **tonne** of talented engineers.

**Thanks to all of them** for their awesome work and open source contributions.

- Prototyping for features that do not depend on hardware.
- ~Infinite numbers of topologies, at least one per developer!

• Virtualised topologies ⇒ more reliable.
• Faster turn up.
• Easy lifecycle management for hermetic builds.
• Ability to emulate hard to create physical scenarios.

- Moving compliance away from human interpretation to code.
- Reproduction of scenarios in a packaged way.
- Ability to plug in different vendors.
Why? Affordable testing scale.

- Many production scenarios ⇒ high lab infrastructure cost.
- Ability to flexibly produce many topologies.
- Production scale (and beyond) verification possible.
What? Introducing KNE.

- **Kubernetes Network Emulation.**
- **Goals:**
  - Lightweight environment for functional, integration and solution testing.
  - Single developer (1-10) ⇒ Large Scale (1000s+) nodes.
  - Common container lifecycle provided by k8s owned by the node vendors.
- Simple user-facing commands tailored to network developers.

- Lower-resource consumption.
- Fast turn up/down times.
- Clear standardised interface.
- Security.
- Still compatible with VMs if needed (VM-in-container).
What? Leveraging K8S.

- Steal whatever we can!
- Reduction in orchestration effort - focus on network problem.
- CNI used to build network mesh.
- Controllers used to do versioning, upgrades, licensing.
- CRD model allows vendors to encapsulate their specifics.

**Services** - ports exposed by containers.
- sasl: tcp/22
- gnmi: tcp/9339
- gribi: tcp/9340

**Nodes** - containers running in topology.

**Implicit management network to host.**

**Links** - connections between nodes.

- ceos:latest
- srl:latest
- vxr:latest
How? KNE Workflow.

Acquire device images → Define Nodes → Define Services → Define Links → Deploy!
How? Defining Nodes.

```json
nodes: {
  name: "r1",
  type: "ARISTA_CEOS",
  model: "ceos",
  os: "eos",
  config: {
    image: "ceos:latest",
    config_path: "/mnt/flash",
    config_file: "startup-config",
    file: "r1.ceos.cfg"
  }
}
```

- **Node type** - allowing vendor specific handling.
- **Specification of parameters** - allows different personalities.
- **Container image name within cluster.**
- **External files available to container.**
- **Additional per-node parameters.**

nodes: {
  name: "r2"
  type: CISCO_XRD
  ...
  config: {
    file: "r2.iosxr.cfg"
    init_image: "networkop/init-wait:latest"
    image: "xrd:latest"
  }
  interfaces: {
    key: "eth1"
    value: {
      name: "GigabitEthernet0/0/0/0"
    }
  }
}

Additional parameters such as helper containers.

Specific handling for Linux interfaces to emulated interfaces.

```
nodes: {
    name: "r4"
    type: JUNIPER_CEVO
    vendor: JUNIPER
    model: "cptx"
    os: "evo"
    services:
        key: 50051
        value:
            name: "gnmi"
            inside: 50051
    }
}
```

Service map exposes container services to external endpoints.

Name key allowing mapping to service endpoints used by test frameworks.

Multiple external ports can be mapped a single internal container port.

```json
links: {
    a_node: "r1"
    a_int: "eth1"
    z_node: "r2"
    z_int: "eth1"
}
```

---

Magic!
veth → gRPC!

Interfaces exposed by the node.

Connections - A:Z pairs.

Topology visualisation - generated from topology.
The KNE Ecosystem.
The KNE Ecosystem.

...plus any container!
Huh? What are we using KNE for?

Programmatically, repeatably validating network deployments.

Ondatra zibethicus
**ONDATRA**

Open Network Device Automated Test Runner & API

- **Go Test**
  - Network test logic - owned by testing team.

- **Ondatra API**
  - Testbed
  - Config
  - Traffic
  - Operations
  - Telemetry
  - Framework abstracts core capabilities.

- **Binding API**
  - APIs exposed by a specific test environment.

- **Lab Infra**
  - Reservation
  - DUT
  - XConnects
  - ATE
  - Physical or virtual resources that tests are run on.
Functional / Standalone Testing

- Simple tests that can be used to validate functionalities of devices.
Enter... KNE!

KNE

ixia-c DUT

Developer Machine

GitHub Actions

donopenconfig/featureprofiles

ixia-c DUT

KNE

VM
Demo!
Ondatra+KNE Demo
TDD for Network Devices

API Design

Develop
Test

Implement production device

Qualify and deploy

openconfig/public
openconfig/gnoi
openconfig/gribi
etc.

Reference
Implementation

openconfig/featureprofiles

openconfig/lemming
openconfig/magna

NANOG
Thank you!

hines@google.com  //  robjs@google.com
www.openconfig.net  //  github.com/openconfig