# User Space CNI

Billy McFall <bmcfall@RedHat.com>
July 17, 2018

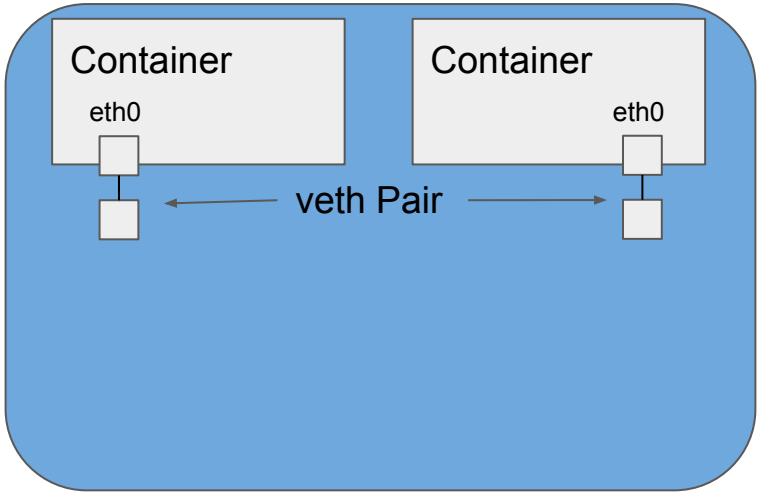
# Warning and Disclaimer:

- **New to GO, so there may be rookie mistakes in code. If you don’t like the way something is done, let me know and I will rework**
- **Also new to K8s and CNIs. During the first round of reviews I was off the mark on some of the json layout, so it could happen again. Open to suggestions.**

# Vhost-User-Net-Plugin:

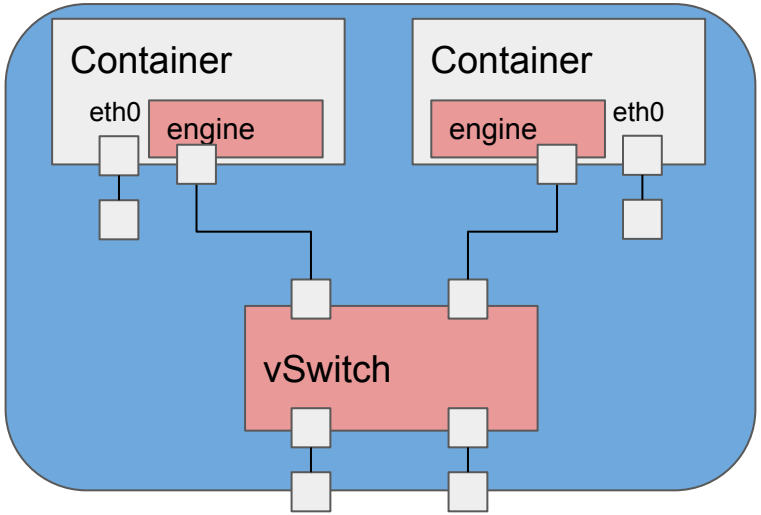
- **Based on Vhost-User-Net-Plugin from Intel**
  - **https://github.com/intel/vhost-user-net-plugin**
- **Reasons for changes:**
  - **The current Vhost-User CNI plugin is limited to vhost-user.**
    - **Want to be able to use other types of implementations (vhost-user|memif|tap|...).**
  - **The current Vhost-User CNI plugin is written in GO and is currently calling a python script passed in from the input JSON. The python script then builds up a CLI command (either VPP or OVS) and then executes the command in a shell command.**
    - **Would like the implementation to be more API driven.**
    - **VPP has a GO API and would like to take advantage of that.**
    - **Even if the python script is used, shouldn't be in the input JSON.**
  - **The current Vhost-User CNI plugin creates the Vhost interface and is done.**
    - **Need to add host side of interface into local network.**
    - **Need to consume the interface in Container.**
    - **Need to add container side of interface into container network.**

# Traditional CNI:




- veth pair create
- IP applied by host through namespace
- eth0 shows up in container and ready to use
- Nothing needed in container
- Only one interface in container, so no question as to which interface to use for what

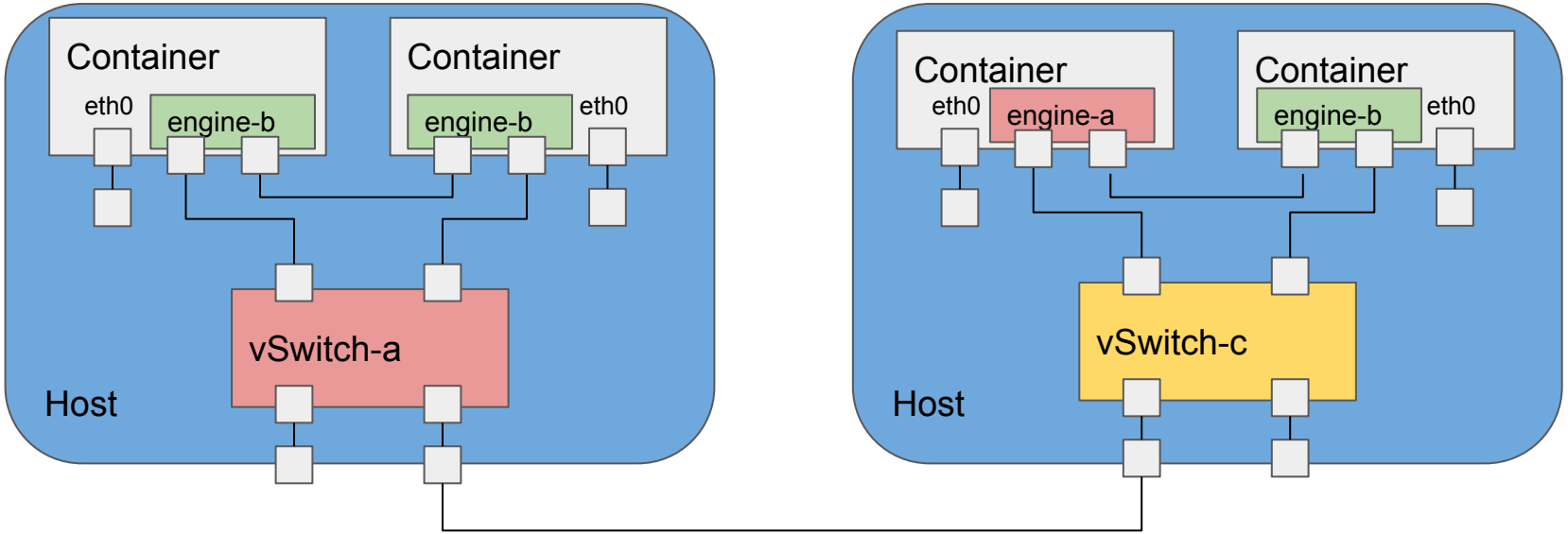
# UserSpace CNI:




- Interfaces (vhost-user/memif) don’t show up in container as straight ethernet interface.
- Something may be needed in container to consume interface.
- Additional processing needed in container to consume interface.
- On host, not strictly IP, so may need or want provisioning on local vSwitch

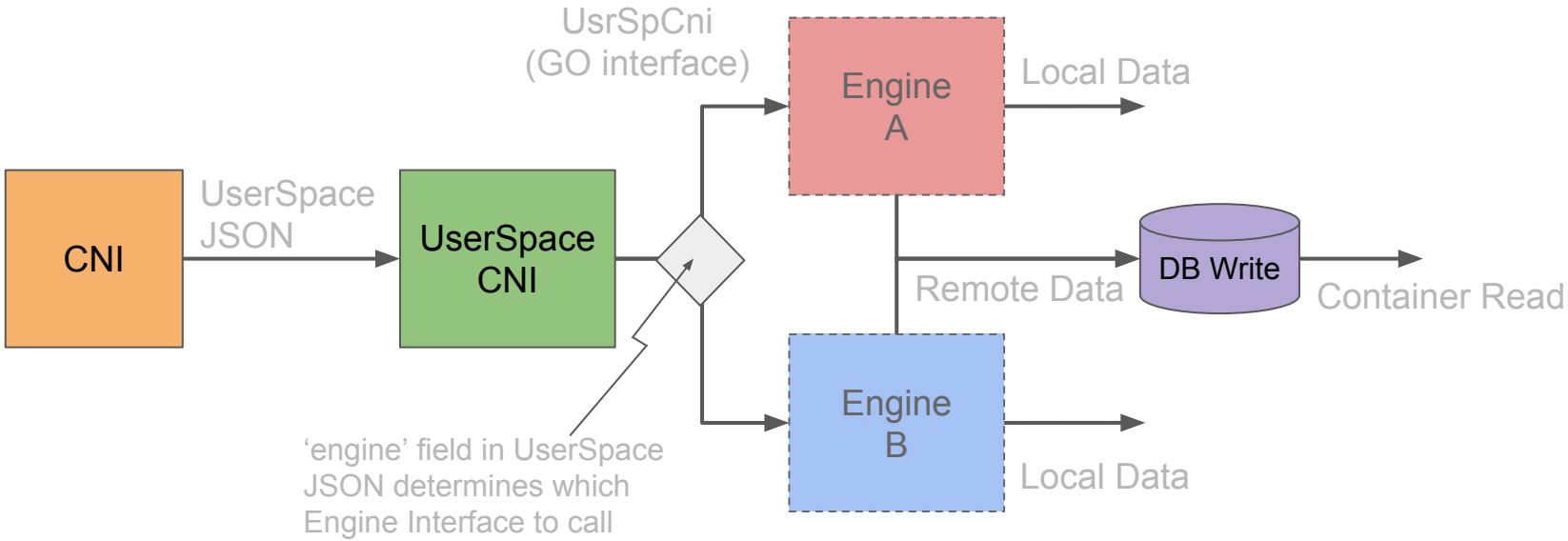
# UserSpace CNI:

Same UserSpace CNI should support multiple ‘engines’ (vSwitch technologies) simultaneously.

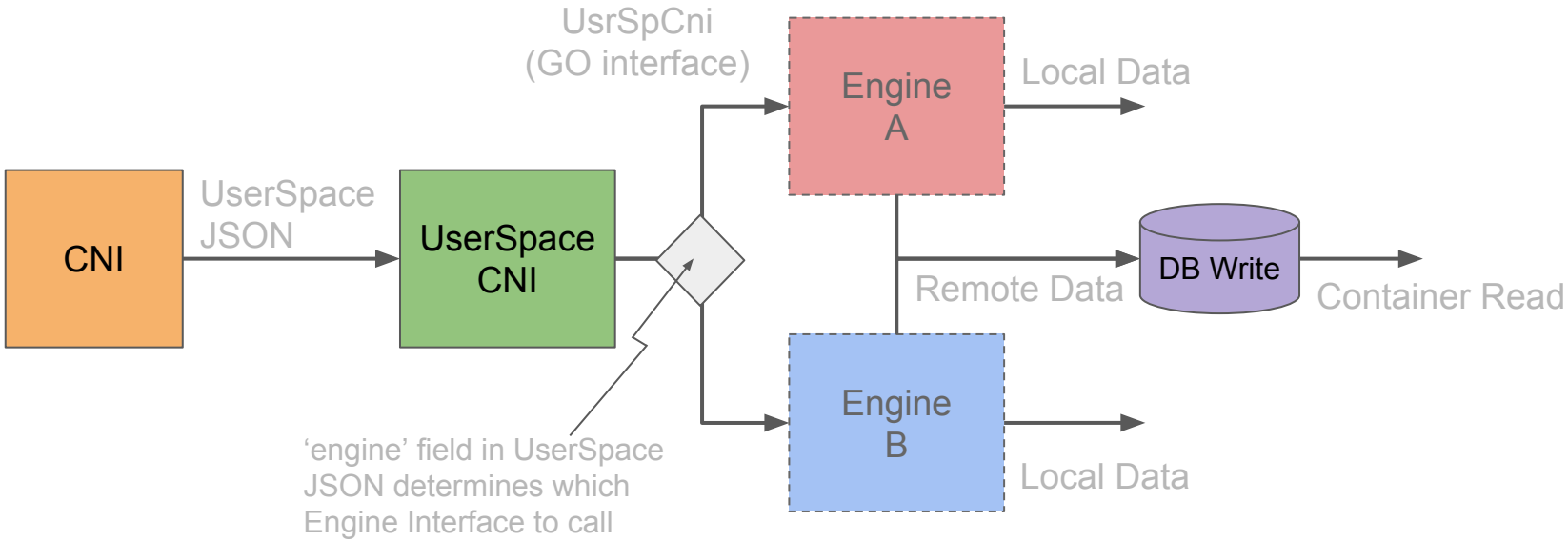



**Note:** This was original goal, but not sure if it what is drawn is doable. Want to at least support different ‘engines’ between host and container (left side). For example: VPP on host, OVS in Container.

# UserSpace CNI: High-Level

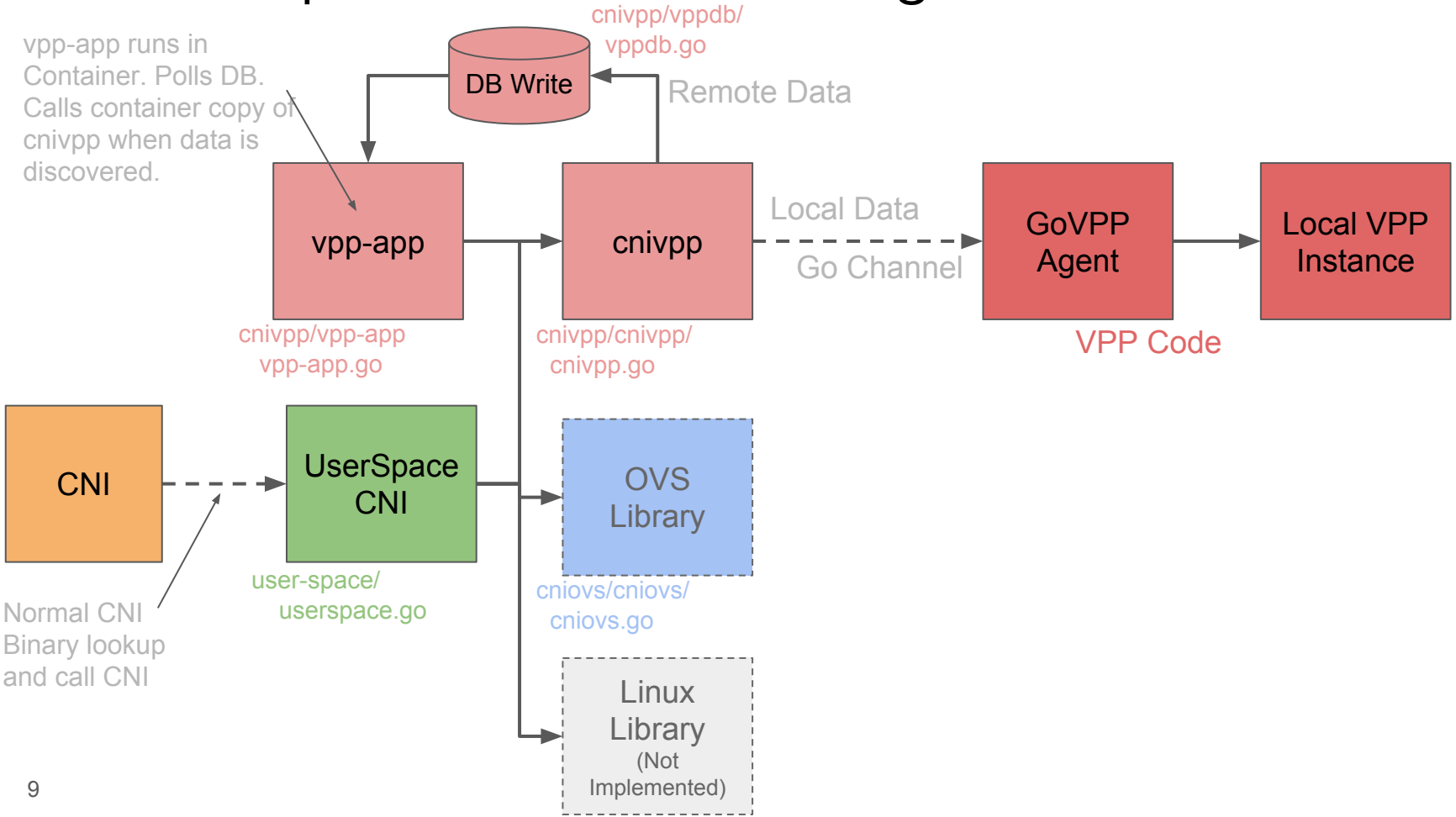


# UserSpace CNI: Design Discussion

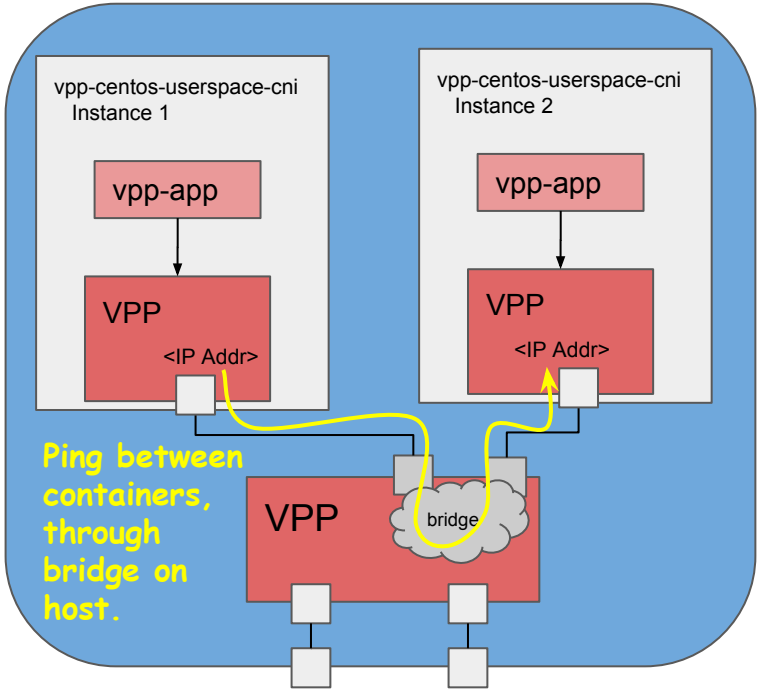



- Which of the interfaces should be common?
  - UserSpace JSON - YES
  - UsrSpCni (GO interface) - YES
  - Local Data - NO
  - Remote Data - NO?
  - Container Read - NO?

# UserSpace CNI: CNI-VPP High Level



# UserSpace CNI: VPP Demo




- User-space-net-plugin (based on input config) creates local memif interface and adds to bridge via cnivpp
- User-space-net-plugin (based on input config) writes remote memif interface and IP to DB
- On container boot, vpp-app reads DB and creates local memif interface and adds IP to interface via cnivpp

# UserSpace CNI: Current Design Limitations

**This code is a work in progress and has its own set of deficiencies:**

- **The input structures defined in usrsptypes may not be the typical CNI layout and probably need some adjustments.**
- **There is spot in the code to branch to OVS or Linux or some other implementation. OvS is currently still using the python script and will add interface to local network (hardcoded) if that network already exists.**
- **Currently, all implementations are compiled in. Not a way to currently only link in the implementations that are desired.**
- **Have only tested with the scripts provided with the Container Network Interface (CNI) project. Have not tested with Multus or Kubernetes.**
- **Moved from a build script to a simple makefile. Long term probably need to go back to the build script, or at least add install functionality. Only had one file to compile so went with simplicity for now. Make/Build are not my strong suit.**
- **Created my own github repo. If any of the code is desired, need to move it back to the Intel repo once it is rebranded.**

## Let go look at code …

- https://github.com/containernetworking/
- https://github.com/Billy99/user-space-net-plugin
- https://hub.docker.com/r/bmcfall/vpp-centos-userspace-cni/

# UserSpace CNI Layout

```
$ tree
.
├── cniovs ...
├── cnivpp ...
├── glide.lock
├── glide.yaml
├── LICENSE
├── Makefile
├── README.md
├── scripts
│   └── vpp-docker-run.sh   <--- Test script (from CNI, add volume mounts)
├── userspace
│   ├── userspace           <--- Binary to be copied into $CNI_PATH
│   └── userspace.go        <--- Main code
├── usrsptypes
│   └── usrsptypes.go       <--- Netconf Definition
└── vendor ...
```

# CNI-OVS Layout

```
$ tree
.

├── cniovs
│   ├── cniovs
│   │   └── cniovs.go          <--- Primary CNI logic. Parse input and call APIs.
│   ├── ovsdb
│   │   └── ovsdb.go           <--- Save data for delete (swIndex)
│   └── scripts
│       └── ovs-config.py      <--- Python script to build up and execute OVS CLI calls.
:
```

# CNI-VPP Layout

```
$ tree
.
├── cnivpp
│   ├── api                              <--- Access VPP GO-Agent
│   │   ├── bridge
│   │   │   └── bridge.go
│   │   ├── infra
│   │   │   └── infra.go
│   │   ├── interface
│   │   │   └── interface.go
│   │   ├── memif
│   │   │   └── memif.go
│   │   └── vhostuser
│   │       └── vhostuser.go
│   ├── cnivpp
│   │   └── cnivpp.go                    <--- Primary CNI logic. Parse input and call APIs.
│   ├── docker
│   │   └── vpp-centos-userspace-cni     <--- Files to create Docker image with VPP and vpp-app
│   │       ├── 80-vpp.conf
│   │       ├── Dockerfile
│   │       ├── fdio-release.repo
│   │       ├── README.md
│   │       ├── startup.conf
│   │       ├── vpp-app
│   │       └── vppcni.sh
:   :
```

# CNI-VPP Layout (cont)

```
$ tree
.
├── cnivpp
:   :
│   ├── docs
│   │   └── UserSpace\ CNI.pdf
│   ├── README.md
│   ├── test
│   │   ├── ipAddDel
│   │   │   └── ipAddDel.go
│   │   ├── memifAddDel
│   │   │   └── memifAddDel.go
│   │   └── vhostUserAddDel
│   │       └── vhostUserAddDel.go
│   ├── vpp-app
│   │   ├── vpp-app            <--- Binary to be added to Container running VPP
│   │   └── vpp-app.go         <--- main(), thin shim, loops looking for DB, calls into local cnivpp
│   └── vppdb
│       └── vppdb.go           <--- Save data for delete (swIndex) and write Remote Data for Container.
:
```