CRI

CNI

The journey starts here!

When it comes to Windows networking & the CNI, that happens somewhere in runhcs??

kubectl apply/delete -f pod.yaml

kube-api

Linux (Windows probably has a different flow)

kubelet
syncPod->KuePodSandbox
KillPod->KuePodSandbox

By Mike Zappa

container runtime Status

Kindly mentioned in discussion of the interesting market for Windows containers (slide from Michael)

I think the dockershim did something like docker run -d --net=container:{ContainerId} image...

RunPodSandbox: Creates Network Namespace & Creates Sandbox Container on Infrastructure container (Pause) and Other containers

container runtime RunPodSandbox

oci.cri.v1 GetPodNetworkStatusWithCtxt rpcs2

libcni CheckNetworkList

oci.cri.v1 SetUpPodWithContext

go-cni (containerd) Setup

libcni AddNetworkList

Hashicorps Nomad uses go-cni

libcni addnetwork

libcni ExecPluginWithResult

libcni ExecPlugin

plugin cniulnid

exit 0
Pod Setup Succeeds!

This is where the "type" key comes in

bridge

host-local

non exit 0
Pod Setup Fails!

If anything fails along the way the pod does not move (cleanup (same for deletion))

```
{
  "cniVersion": "0.4.0",
  "name": "statelessnet",
  "plugins": [
    {
      "type": "bridge",
      "bridge": "cni0",
      "isGateway": true,
      "ipMasq": true,
      "promiscMode": true,
      "ipam": {
        "type": "host-local",
        "ranges": [
          [
            {
              "subnet": "10.240.0.0/24"
            }
          ]
        ],
        "routes": [
          { "dst": "0.0.0.0/0" }
        ]
      }
    }
  ]
}
```

```
{
  "cniVersion": "1.0.0",
  "code": 7,
  "msg": "Invalid Configuration",
  "details": "Network 192.168.0.0/30 too small to allocate from."
}
```

If the json does not conform to the cni spec this will fail

Cached Locally:
/var/lib/cni/results/containerd-net-bc7174f5aede0f8a3d6cfacbc0a136f85cb26a83256de139aef8d95b04e08ea3-eth0

Format:
/var/lib/cni/results/{NetworkName}-{ContainerId}-{InterfaceName}

```
{
  "cniVersion": "0.4.0",
  "interfaces": [
    {
      "name": "cni0",
      "mac": "56:a6:2ac:e453:bd"
    },
    {
      "name": "veth3148e617",
      "mac": "7e:84:ef:54:fd:be"
    },
    {
      "name": "eth0",
      "mac": "7e:7c:81:64:89:15",
      "sandbox": "/var/run/netns/cni-1234"
    }
  ],
  "ips": [
    {
      "version": "4",
      "interface": 2,
      "address": "10.240.0.2/24",
      "gateway": "10.240.0.1"
    }
  ],
  "routes": [
    {
      "dst": "0.0.0.0/0"
    }
  ],
  "dns": {}
}
```

Host-Local Caches At:
/var/lib/cni/networks/containerd-net/

Format:
/var/lib/cni/networks/{NetworkName}

Details are visible in kubectl describe pod or crictl run ...

container runtime StopPodSandbox

libcni delnetwork

libcni ExecPluginWithResult

libcni ExecPlugin

Since the container runtime deletes the network namespace, we really just need to make sure the host-local reclaims the ip address

exit 0
Pod Successfully Deleted

go-cni (containerd) Remove

libcni DelNetworkList

plugin cniulnit

bridge

host-local

Garbage Collection Pics from Kubernetes Chi Slow

container

container runtime RemovePodSandbox

CRI-O

oci.cri.v1 TearDownPodWithCtxtrpc

non exit 0
Pod Deletion Fails

```
{
  "cniVersion": "1.0.0",
  "code": 7,
  "msg": "Invalid Configuration",
  "details": "cni plugin not initialized"
}
```

By Mike Zappa

Not implemented (yet)

When it comes to Check we will need to define the failure behavior

oci.cri.v1 GetPodNetworkStatusWithCtxtrpc

go-cni (containerd) Check

libcni CheckNetworkList

libcni checknetwork

libcni ExecPluginWithResult

libcni ExecPlugin

plugin cniulCheck

Exit 0

Non Exit 0

In cri-o CHECK is called after ADD. As a user cannot call CHECK via the cli or define its behavior anywhere