RESILIENCE

REALIZED

KubeCon | CloudNativeCon

North America 2021

k8snetlook – Root-Causing k8s Network Problems in an Automated Way

Arun Sriraman

vmware

ex - PLATFORM9

@arun_sriraman

sarun87

https://www.github.com/sarun87/k8snetlook

What could go wrong?


```
root@cp1:/vagrant# kc get no
NAME      STATUS   ROLES                  AGE    VERSION
cp1       Ready    control-plane,master   2d2h   v1.22.1
worker1   Ready    <none>                 2d2h   v1.22.1
worker2   Ready    <none>                 2d2h   v1.22.1
root@cp1:/vagrant# kc get po -o wide --all-namespaces
NAMESPACE              NAME                                        READY   STATUS    RESTARTS      AGE    IP               NODE      NOMINATED NODE   READINESS GATES
default                bbox-6fcbbb566f-jknjn                       1/1     Running   5 (42m ago)   47h    10.244.189.68    worker2   <none>           <none>
default                hostnames-ccdf74459-85p5j                   1/1     Running   0             46h    10.244.235.132   worker1   <none>           <none>
default                hostnames-ccdf74459-mzl2w                   1/1     Running   0             46h    10.244.235.131   worker1   <none>           <none>
default                hostnames-ccdf74459-pdcj7                   1/1     Running   0             47h    10.244.189.67    worker2   <none>           <none>
default                traefik-7499c455c-56ltg                     1/1     Running   0             5m31s  10.244.189.72    worker2   <none>           <none>
kube-system            calico-kube-controllers-58497c65d5-fcmp2    1/1     Running   0             2d     10.244.2.4       worker1   <none>           <none>
kube-system            calico-node-dlp2d                           1/1     Running   0             2d     10.0.124.10      cp1       <none>           <none>
kube-system            calico-node-h9w6j                           1/1     Running   0             2d     10.0.124.11      worker1   <none>           <none>
kube-system            calico-node-smhnz                           1/1     Running   0             2d     10.0.124.12      worker2   <none>           <none>
kube-system            coredns-78fcd69978-xtd8z                    1/1     Running   0             46h    10.244.189.69    worker2   <none>           <none>
kube-system            coredns-78fcd69978-zs7jk                    1/1     Running   0             46h    10.244.189.70    worker2   <none>           <none>
kube-system            etcd-cp1                                    1/1     Running   1             2d2h   10.0.124.10      cp1       <none>           <none>
kube-system            kube-apiserver-cp1                          1/1     Running   1             2d2h   10.0.124.10      cp1       <none>           <none>
kube-system            kube-controller-manager-cp1                 1/1     Running   1             2d2h   10.0.124.10      cp1       <none>           <none>
kube-system            kube-proxy-254j7                            1/1     Running   0             2d2h   10.0.124.11      worker1   <none>           <none>
kube-system            kube-proxy-9lc5h                            1/1     Running   0             2d2h   10.0.124.10      cp1       <none>           <none>
kube-system            kube-proxy-t7f9d                            1/1     Running   0             2d2h   10.0.124.12      worker2   <none>           <none>
kube-system            kube-scheduler-cp1                          1/1     Running   1             2d2h   10.0.124.10      cp1       <none>           <none>
kubernetes-dashboard   dashboard-metrics-scraper-856586f554-5cvh7  1/1     Running   0             13m    10.244.235.133   worker1   <none>           <none>
kubernetes-dashboard   kubernetes-dashboard-67484c44f6-bzxgk       1/1     Running   0             13m    10.244.189.71    worker2   <none>           <none>
metallb-system         controller-6b78bff7d9-m84kq                 1/1     Running   0             2m16s  10.244.235.134   worker1   <none>           <none>
metallb-system         speaker-4t5lk                               1/1     Running   0             2m16s  10.0.124.12      worker2   <none>           <none>
metallb-system         speaker-d26xm                               1/1     Running   0             2m16s  10.0.124.11      worker1   <none>           <none>
metallb-system         speaker-klnmd                               1/1     Running   0             2m16s  10.0.124.10      cp1       <none>           <none>
root@cp1:/vagrant#
```

```
root@cp1:/vagrant# kubectl exec -it bbox-6fcbbb566f-jknjn -- wget -qO-  $(kubectl get svc hostnames -o go-template={{.spec.clusterIP}}):80
hostnames-ccdf74459-85p5j
root@cp1:/vagrant# kubectl exec -it bbox-6fcbbb566f-jknjn -- wget -qO-  $(kubectl get svc hostnames -o go-template={{.spec.clusterIP}}):80
hostnames-ccdf74459-mzl2w
root@cp1:/vagrant# kubectl exec -it bbox-6fcbbb566f-jknjn -- wget -qO-  $(kubectl get svc hostnames -o go-template={{.spec.clusterIP}}):80
^Ccommand terminated with exit code 130
root@cp1:/vagrant# kubectl exec -it bbox-6fcbbb566f-jknjn -- wget -qO-  $(kubectl get svc hostnames -o go-template={{.spec.clusterIP}}):80
hostnames-ccdf74459-pdcj7
root@cp1:/vagrant#
```

Agenda

- K8s networking primer
  - Communication primitives (pod, service)
  - Network components
- Common problems and issue classes
- Network issue resolution strategies
- Open source troubleshooting tools - synthetic probes & live pod level debugging
- K8snetlook
  - Introduction
  - Demo

K8s networking primer

The three golden rules

All containers can communicate with all other containers without NAT

All nodes can communicate with all containers (and vice-versa) without NAT

The IP that a container sees itself as, is the same IP that others see it as

K8s networking primer

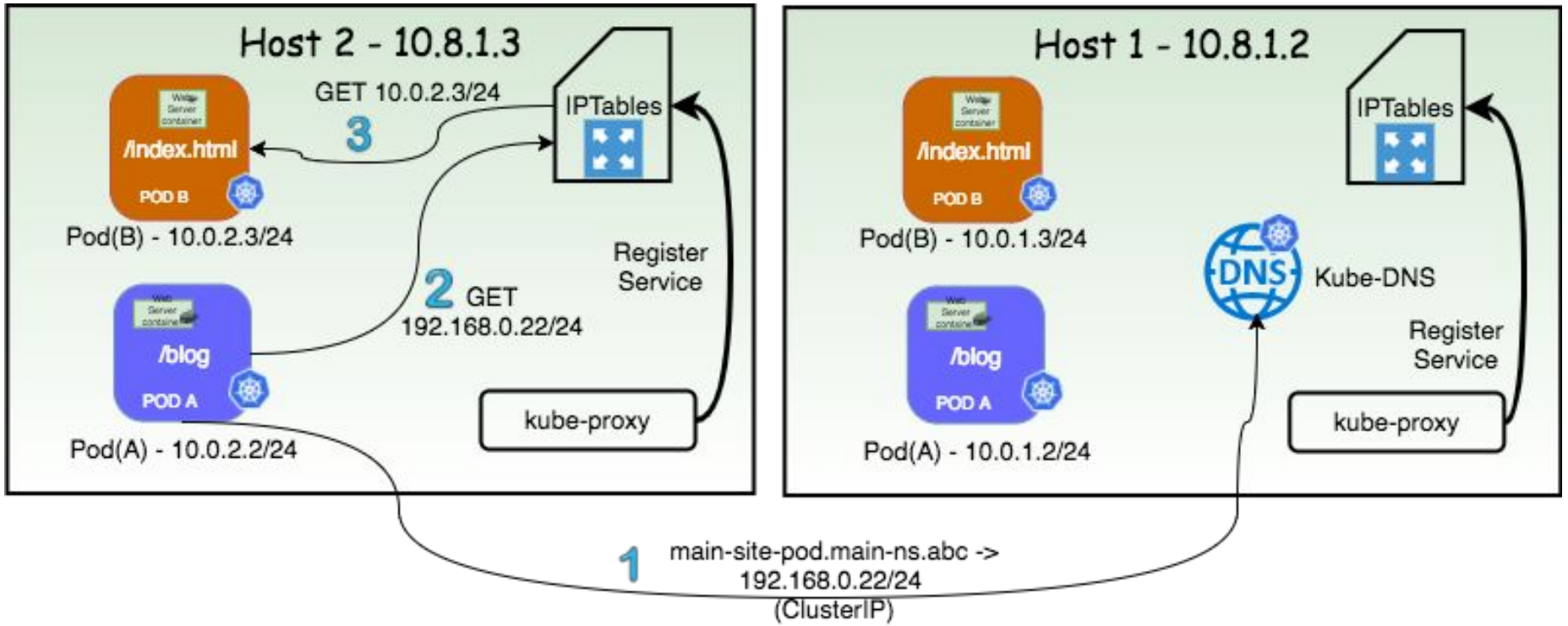
Network Components

- CNI backend implementation (Eg: Calico, Flannel, Cilium, Antrea...)
- Kube-proxy
- Loadbalancer implementation (metallb, aws elb, f5...)
- Ingress implementation (nginx, haproxy, traefik…)
- Service mesh implementation (Istio, linkerd, consul connect…)
- Iptables, ipvs, linux network stack, sr-iov, dpdk...

K8s networking primer

KubeCon CloudNativeCon North America 2021

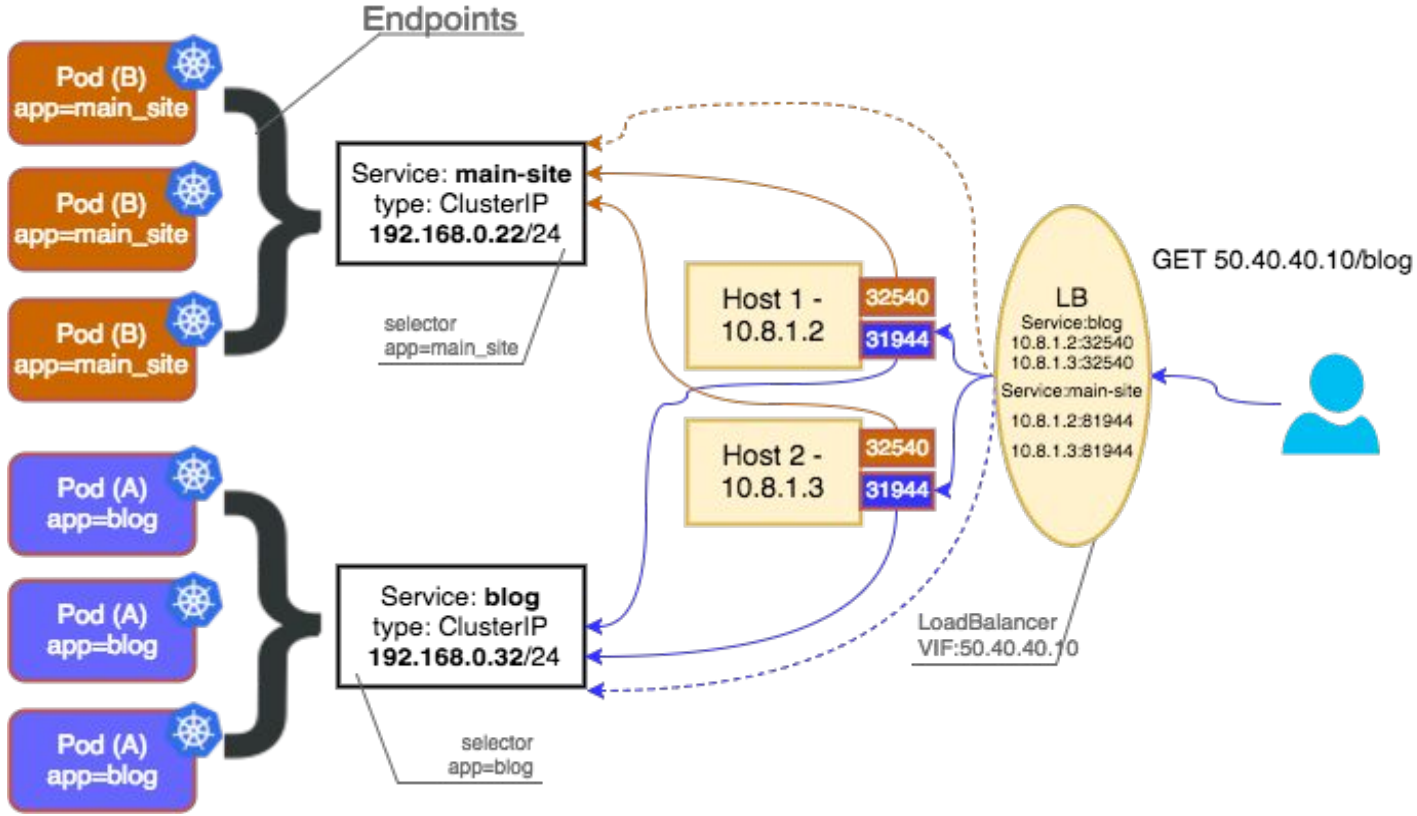
Communication Primitives - Pods & Services



K8s networking primer

KubeCon CloudNativeCon North America 2021

Communication Primitives - Endpoints, Nodeports and Loadbalancers



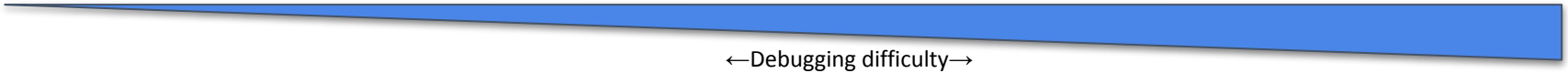
Common Problems

Problem class/type

Complete service outage (no connectivity)

Partial outage (#n pods not responding)

Degraded performance (dropped packets, buffer overflows, network loops)



Common problems

Application problems

- App errors
- File descriptors (i/o)
- memory/cpu limits

Platform components

- Dns
- Load-balancer
- Kube-proxy (service)

Configuration problems

- mtu
- routing (routes)
- subnet (ip) overflow

Resolution Strategies

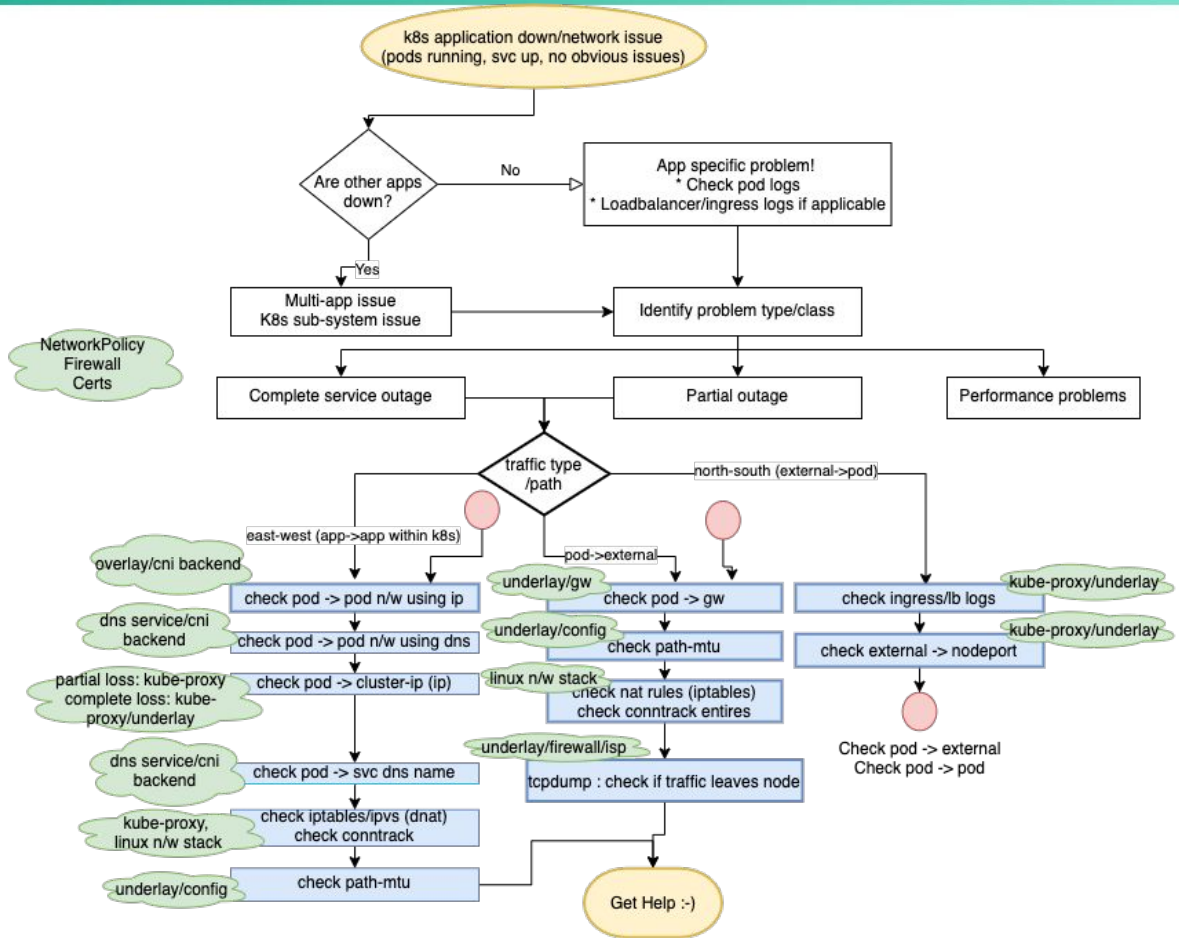
- Big-hammer approach
  - Bounce/delete k8s objects - pod, deployment, service, ingress, et.al
    - May not be feasible for stateful apps
  - delete node/s
    - Not simple for non cloud k8s deployments (well may be fine with minikube)

- Ask for help - network experts!!

- App not a good candidate for cloud-native architecture & k8s :-((“Grapes are sour”)

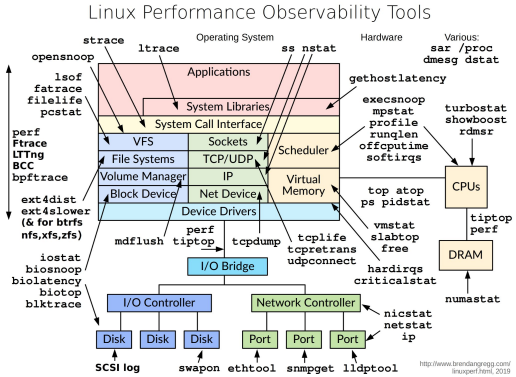
- Self-Troubleshoot live setups & remediate selectively (manually?)

Troubleshooting flowchart

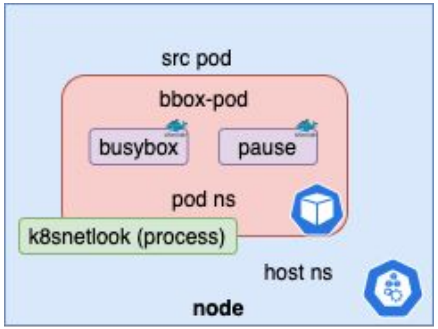


Open source tools

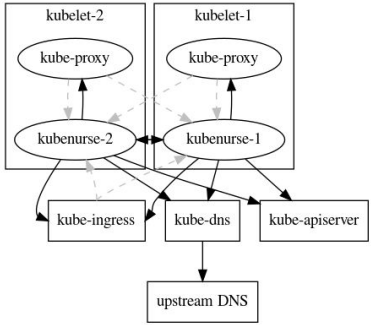
- Netshoot image



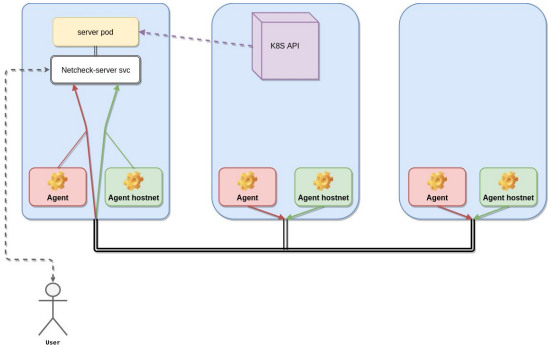
- K8snetlook :)



- Kubenurse



- K8s-netchecker*not updated >2yrs



What to use when

KubeCon CloudNativeCon North America 2021

Problem type/classification

Application logging (ELK…), **k8snetlook**, Service meshes

Iperf3 (manually), K8s standard perf utility - netperf

Complete service outage (no connectivity)

Partial outage (#n pods not responding)

Degraded performance (dropped packets, buffer overflows, network loops)

Continuous monitoring (prometheus, node-exporter), probes (kubernurse)

←Debugging difficulty→

Common problems

Manual n/w tools (ping, tcpdump, …), **k8snetlook**

Application problems

- App errors
- File descriptors (i/o)
- memory/cpu limits

App logs (ELK...), node-exporter, k8s deployment specs

Platform components

- Dns
- Load-balancer
- Kube-proxy (service)

Probes, **k8snetlook,** k8s metrics, (tcpdump, ping...)

Configuration problems

- mtu
- routing (routes)
- subnet (ip) overflow


KubeCon | CloudNativeCon

North America 2021

RESILIENCE

REALIZED

k8snetlook

k8snetlook

What

- External-to-k8s binary/docker image
- Src side debugging* needs to be run on the node that runs the src pod
- Imitates the pod by running within the pod network namespace

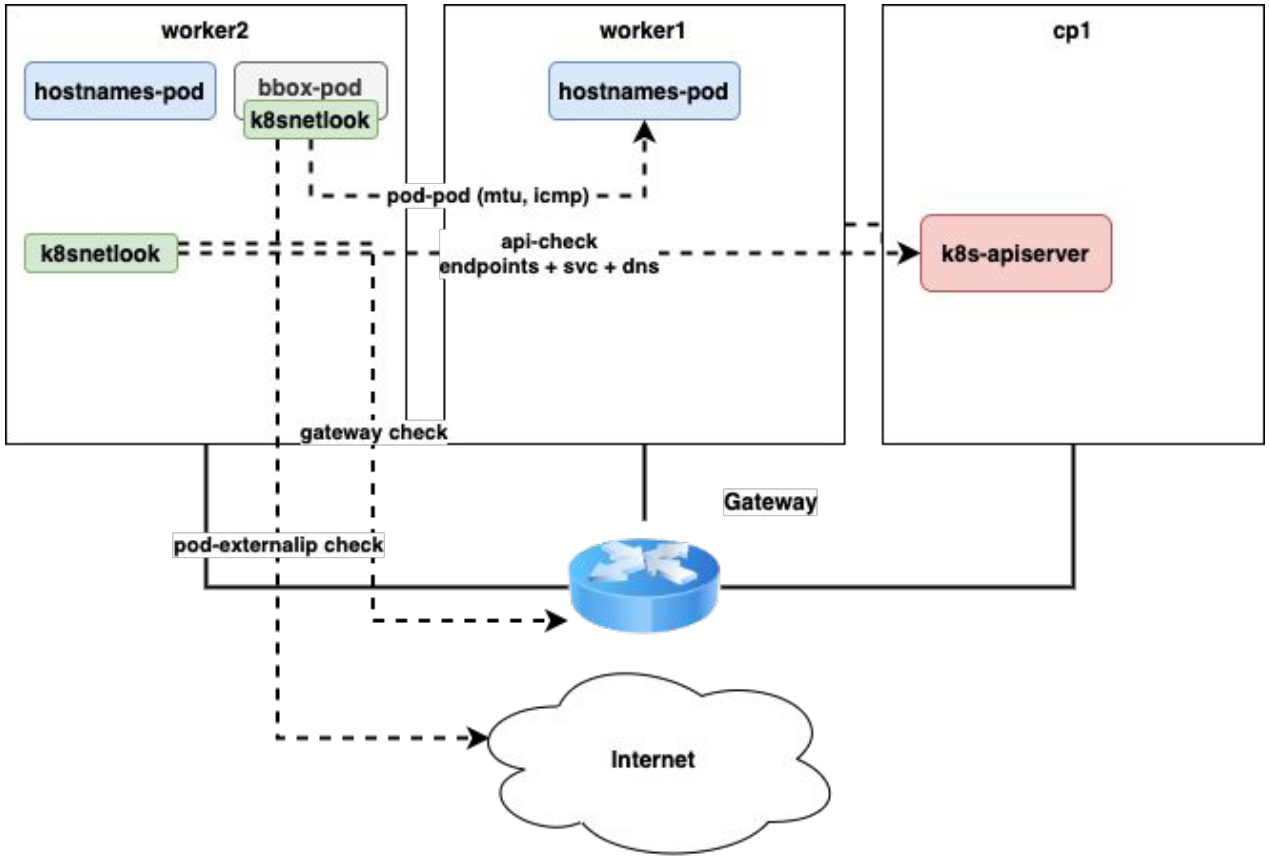
Why

- Network is fundamentally unreliable and debugging issues are hard. Most issues need to be debugged in live environments
- Make network debugging easy!
- Time is of essence during debugging
- Automate mundane debugging steps when possible
- Self-service debugging : Users can run the tool as a first pass reducing need for additional support
- Inspired by “https://github.com/kubernetes/node-problem-detector”

Capabilities

Host Checks	Pod Checks
Default gateway connectivity (icmp)	Default gateway connectivity (icmp)
K8s-apiserver ClusterIP check (https)	K8s-apiserver ClusterIP check (https)
K8s-apiserver individual endpoints check (https)	K8s-apiserver individual endpoints check (https)
K8s-apiserver health-check api (livez)	Destination Pod IP connectivity (icmp)
	External IP connectivity (icmp)
	K8s DNS name lookup check (kubernetes.local)
	K8s DNS name lookup for specific service check
	Path MTU discovery between Src & Dst Pod (icmp)
	Path MTU discovery between Src Pod & External IP (icmp)
	All K8s service endpoints IP connectivity check (icmp)

Demo


worker2
worker1
cp1
hostnames-pod
bbox-pod
k8snetlook
hostnames-pod
pod-pod (mtu, icmp)
k8snetlook
api-check
endpoints + svc + dns
k8s-apiserver
gateway check
Gateway
pod-externalip check
Internet

Q & A

KubeCon | CloudNativeCon
North America 2021

RESILIENCE
REALIZED

Feature asks

- external -> pod debug automation (ingress, lb, …)
- CNI aware debug automation (eg: calico-bgp config…)
- …