1. What are options to connecting to database instance without overhead of Proxies or Load Balancers

If Database are deployed within K8s cluster (typically as Statefulset), but DON'T need external connectivity access - the typical scenarios would be:

- Internal Clients (e.g., frontend apps / database clients) would connect to Internal ClusterIP Service for scenarios where it's ok to be Load Balanced between different DB replica instances
  - Internal DNS Lookup for Service will return ClusterIP Address that frontends all the database pod IPs. The path taken would leverage kube-proxy/iptables pathway.
- Alternatively, Internal Clients could leverage `Headless` Services to connect to specific Stateful Pod / DB Replica Instance (i.e., secondary readonly instance) - https://kubernetes.io/docs/concepts/services-networking/service/#headless-services
  - With `Headless` Service, the ClusterIP attribute would be set to `None`, therefore no `ClusterIP` is allocated, `kube-proxy` won't be leveraged to handle services and there is no load balancing or proxying would be done by Kubernetes Platform. A DNS Service Lookup will simply return the Pod IP Address.
- In both cases, DNS is configured automatically based on whether or not the `Service` has `Selectors` defined
  - When `Selector` is defined, `Endpoints` are configured with Respective A Host Records in DNS (i.e., coredns) for each service / statefulset pod.

Alternatively, if database are external/outside the kubernetes clusters, then you'd want to leverage Service of type `ExternalName` and DNS CNAME lookups

- Without `Selector`, A `CNAME` records could be leveraged with `ExternalName` type as means of connecting to external service from within Kubernetes
  - https://kubernetes.io/docs/concepts/services-networking/service/#externalname
  - https://www.googblogs.com/kubernetes-best-practices-mapping-external-services/

Lastly, If Database are deployed within K8s cluster but need external connectivity access - the typical scenarios would be:

- Leverage Port Forwarding for temporary access
  - https://kubernetes.io/docs/tasks/access-application-cluster/port-forward-access-application-cluster/
- OR Leverage External DNS Operator and Headless Service (with type=NodePort) to Create DNS Records for External IP of Host(s) (vs PodIPs)
  - https://github.com/kubernetes-sigs/external-dns
  - https://github.com/kubernetes-sigs/external-dns/blob/master/docs/tutorials/hostport.md

Additional References if needed:

- https://cloud.google.com/blog/products/databases/to-run-or-not-to-run-a-database-on-kubernetes-what-to-consider
- https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/
- https://kubernetes.io/docs/tasks/run-application/run-replicated-stateful-application/

2. Is there any step-by-step documentation on configuring Calico BGP (Border Gateway Protocol) Peering

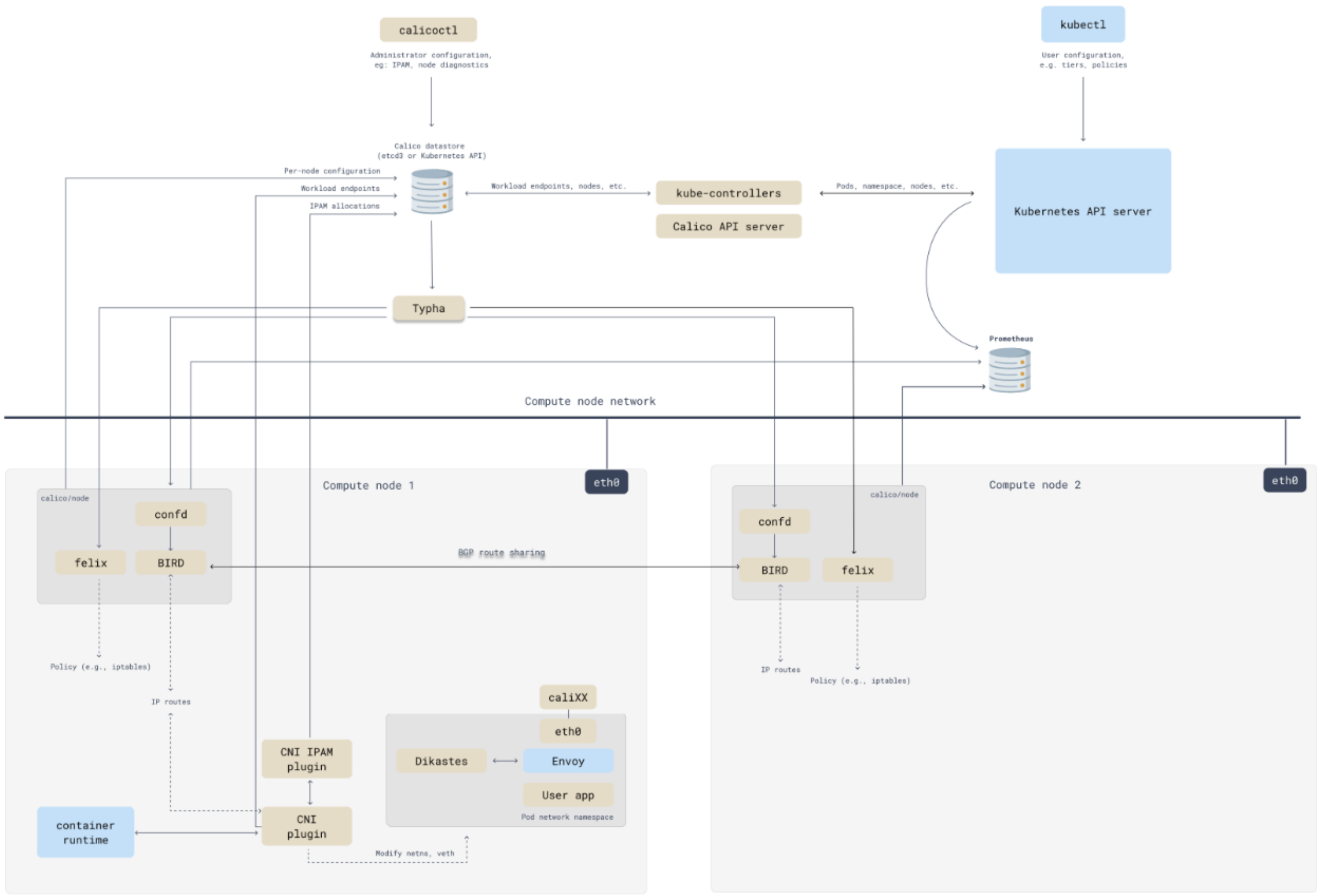
At this time, there is no official step-by-step Nutanix documentation on how to configure BGP Peering within Karbon using Calico. However, it is worth noting that when Karbon installs and configures Calico, BGP is already enabled with the default behavior set to create a full-mesh of internal BGP (iBGP) connections where each node peers with each other.

This can be verified by running `sudo calicoctl node status` post Karbon deployment from one of the worker nodes

```
$ sudo calicoctl node status

IPv4 BGP status
+--------------+-------------------+-------+----------+-------------+
| PEER ADDRESS |     PEER TYPE     | STATE |  SINCE   |    INFO     |
+--------------+-------------------+-------+----------+-------------+
| 10.38.11.32  | node-to-node mesh | up    | 19:13:45 | Established |
| 10.38.11.47  | node-to-node mesh | up    | 19:15:19 | Established |
| 10.38.11.44  | node-to-node mesh | up    | 19:15:46 | Established |
| 10.38.11.41  | node-to-node mesh | up    | 19:16:17 | Established |
| 10.38.11.36  | node-to-node mesh | up    | 19:13:43 | Established |
| 10.38.11.52  | node-to-node mesh | up    | 19:13:43 | Established |
| 10.38.11.61  | node-to-node mesh | up    | 19:14:46 | Established |
| 10.38.11.56  | node-to-node mesh | up    | 19:14:16 | Established |
+--------------+-------------------+-------+----------+-------------+
```

- Calico Component Architecture: https://projectcalico.docs.tigera.io/archive/v3.21/reference/architecture/overview



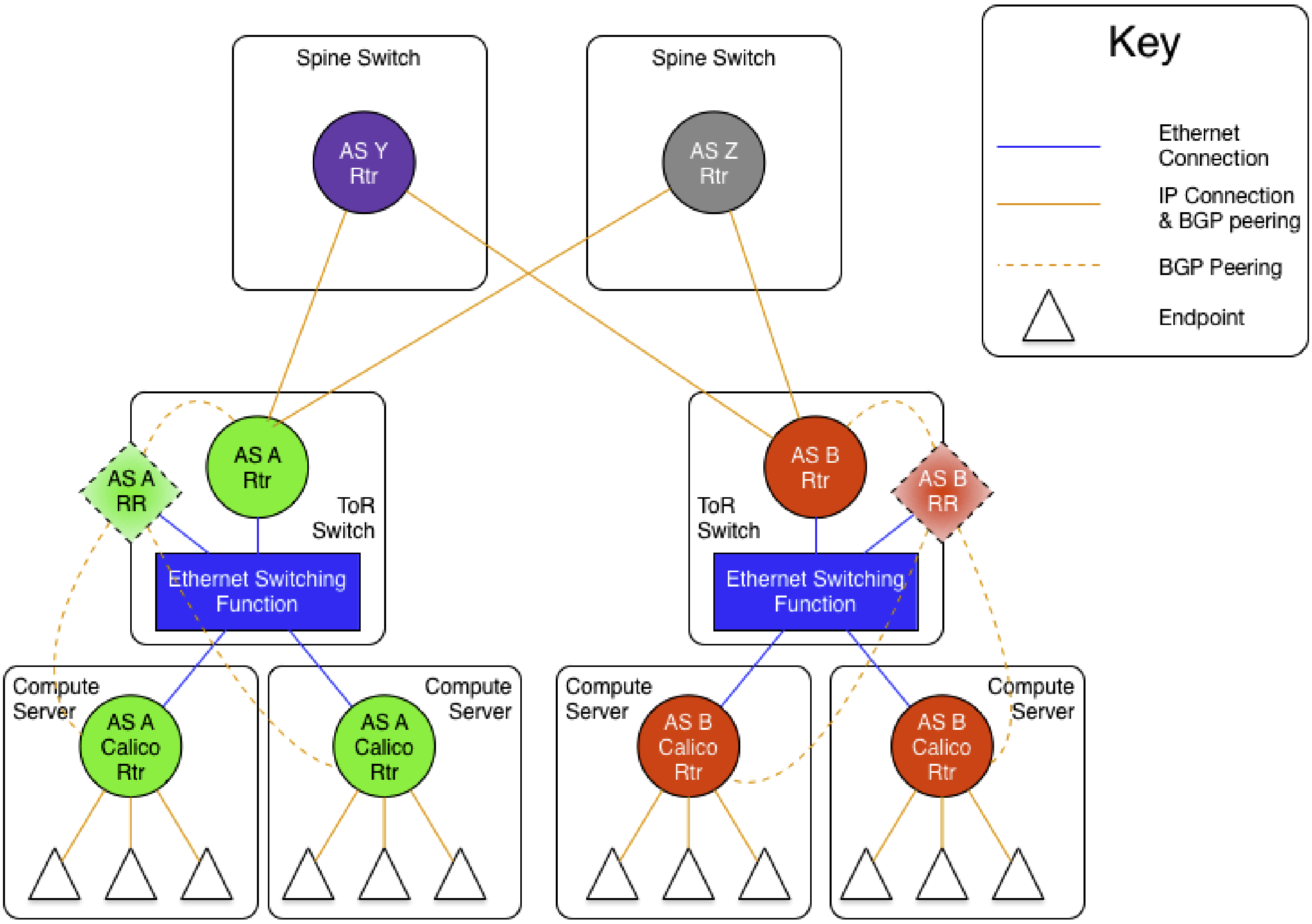
- Calico over IP fabrics: https://projectcalico.docs.tigera.io/reference/architecture/design/l3-interconnect-fabric

> "There are two approaches to building an IP fabric for a scale-out infrastructure. However, all of them, to date, have assumed that the edge router in the infrastructure is the top of rack (TOR) switch. In the Calico model, that function is pushed to the compute server itself."

There are multiple methods to build a BGP-only interconnect fabric - with the two most common methods being:

- Method Example 1: AS (Autonomous System) per Rack:

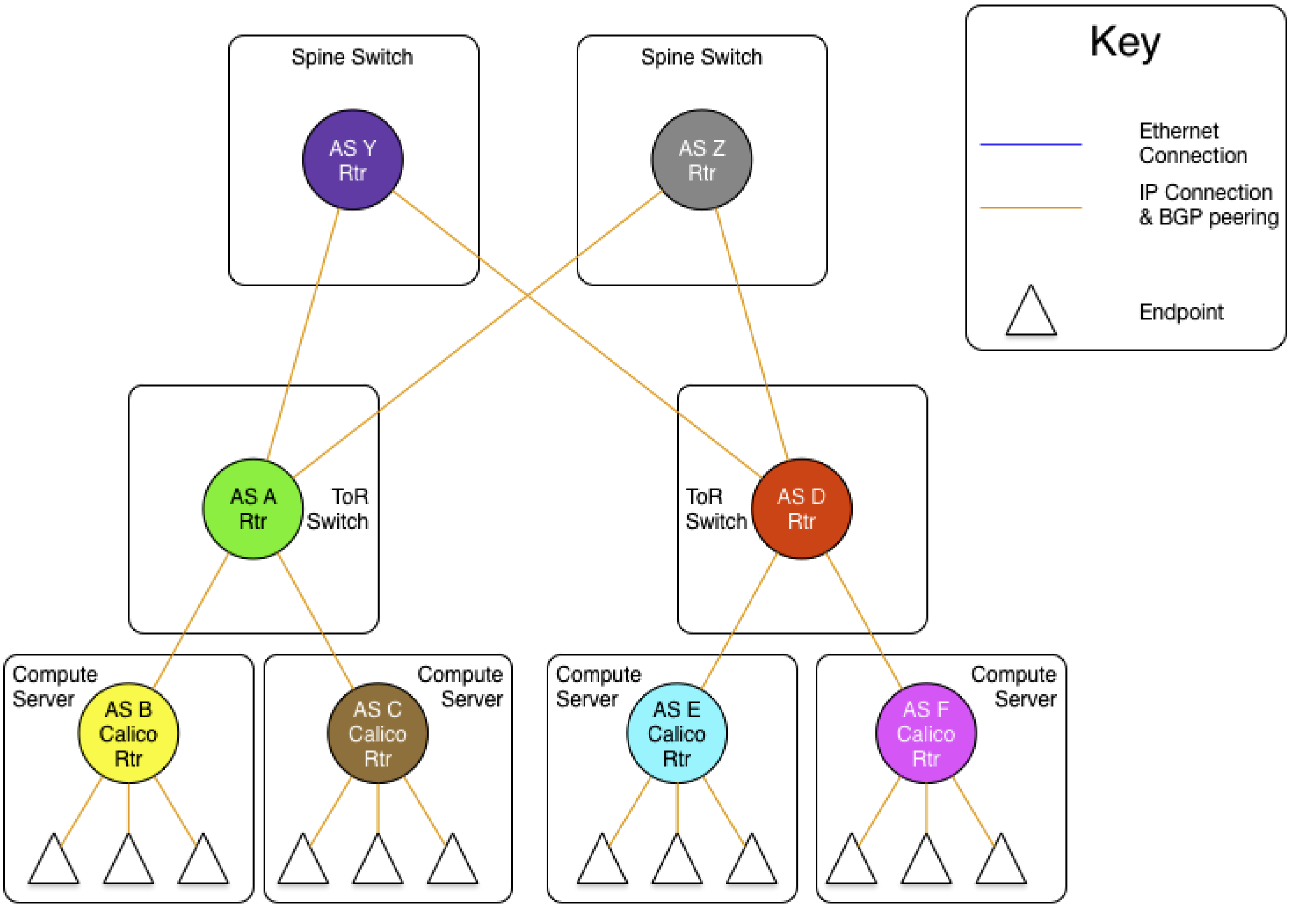
> "A BGP fabric where each of the TOR switches (and their subsidiary compute servers) are a unique Autonomous System (AS) and they are interconnected via either an Ethernet switching plane provided by the spine switches in a leaf/spine architecture, or via a set of spine switches, each of which is also a unique AS..."



The diagram above shows the AS per rack model where the ToR switches are physically meshed via a set of discrete BGP spine routers, each in their own AS.

- Method Example 2: AS (Autonomous System) per Server:

> "A BGP fabric where each of the compute servers is a unique AS, and the TOR switches make up a transit AS. We'll refer to this as the AS per server model..."



> "The Project Calico team recommends the use of the `AS per rack` model if the resultant routing table size can be accommodated by the ToR and spine switches, remembering to account for projected growth."

...in any case, refer to tigera documentation for additional details on variations and options that would explicitly meet your needs.

To configure Calico nodes to peer with route reflectors, or with top-of-rack (ToR) routers.

- https://projectcalico.docs.tigera.io/networking/bgp

Below are examples of a configuration I've seen leveraging Calico BGP peering directly with their physical network infrastructure using a combination of top-of-rack (ToR) leaf-spine switches/routers (as per-node bgp peers/ASNs) and custom nodeSelectors to control IP pool distribution between application and system pods.

Example Disabling Node to Node ServiceMesh

Default config for Karbon is Node to Node Service Mesh, so that would need to be disabled and ASN provided:

01_networking -

```yaml
---
apiVersion: projectcalico.org/v3
kind: BGPConfiguration
metadata:
  name: default
spec:
  logSeverityScreen: Info
  nodeToNodeMeshEnabled: false
  asNumber: 63400

```

Example Per-Node BGP Peer Configurations

01_networking -

```yaml
---
apiVersion: projectcalico.org/v3
kind: BGPPeer
metadata:
  name: bgppeer-lv-1-cr-i18-leaf01
spec:
  peerIP: 10.255.250.30
  nodeSelector: rack=='lv-1-cr-i18'
  asNumber: 420001012

---
apiVersion: projectcalico.org/v3
kind: BGPPeer
metadata:
  name: bgppeer-lv-1-cr-i18-leaf02
spec:
  peerIP: 10.255.250.31
  nodeSelector: rack=='lv-1-cr-i18'
  asNumber: 420001012

---
```

```yaml
apiVersion: projectcalico.org/v3
kind: BGPPeer
metadata:
  name: bgppeer-lv-1-cr-i19-leaf01
spec:
  peerIP: 10.255.250.32
  nodeSelector: rack=='lv-1-cr-i19'
  asNumber: 420001013

---
apiVersion: projectcalico.org/v3
kind: BGPPeer
metadata:
  name: bgppeer-lv-1-cr-i19-leaf02
spec:
  peerIP: 10.255.250.33
  nodeSelector: rack=='lv-1-cr-i19'
  asNumber: 420001013
```

Example Multi-IPAM Pool with NodeSelectors

Below is example of Multiple IPAM Pool for handling master vs. worker node scenarios.

01_networking -

```yaml
---
apiVersion: projectcalico.org/v3
kind: IPPool
metadata:
  name: inside-ipv4-ippool
spec:
  blockSize: 23
  cidr: 10.143.32.0/20
  ipipMode: Never
  natOutgoing: false
  nodeSelector: has(inside-node)
  vxlanMode: Never
---
apiVersion: projectcalico.org/v3
kind: IPPool
metadata:
  name: master-ipv4-ippool
spec:
  blockSize: 27
  cidr: 10.143.62.0/23
  ipipMode: Never
  natOutgoing: true
  nodeSelector: has(node-role.kubernetes.io/master)
  vxlanMode: Never

```

Example Master BGP Peers config with NodeSelectors

01_networking -

```yaml
---
apiVersion: projectcalico.org/v3
kind: BGPPeer
metadata:
  name: bgppeer-master-ibgp-mesh
spec:
  nodeSelector: has(node-role.kubernetes.io/master)
  peerSelector: has(node-role.kubernetes.io/master)
```

Simulation / Testing

If you're looking to do some level of testing without actual physical routers/top of rack switches, you could leverage gobgp (https://github.com/osrg/gobgp) and the following documentation below to simulate various BGP configuration scenarios that (i.e., Multi-AS Configs & BGP route reflectors) possibly validate how you wish to proceed towards your respective initiatives.

- https://projectcalico.docs.tigera.io/archive/v3.21/getting-started/kubernetes/hardway/configure-bgp-peering
- https://www.tkng.io/cni/calico/

Additional References if needed:

- https://projectcalico.docs.tigera.io/archive/v3.21/getting-started/kubernetes/hardway/
- https://learnk8s.io/kubernetes-network-packets

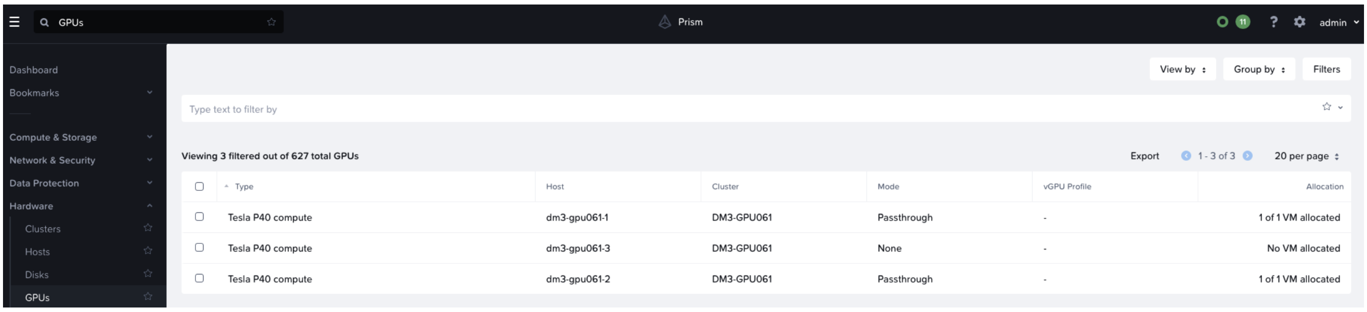
3. Is there a way to leverage Nutanix Karbon for AI/ML solutions, e.g., KubeFlow explicitly on BareMetal or Compute-Heavy Nodes

As of March 2022, the Karbon 2.4 release on added GPU pass-through support for Karbon node pools. This enables containerized use of GPU for AI/ML use cases with NVIDIA GPU.

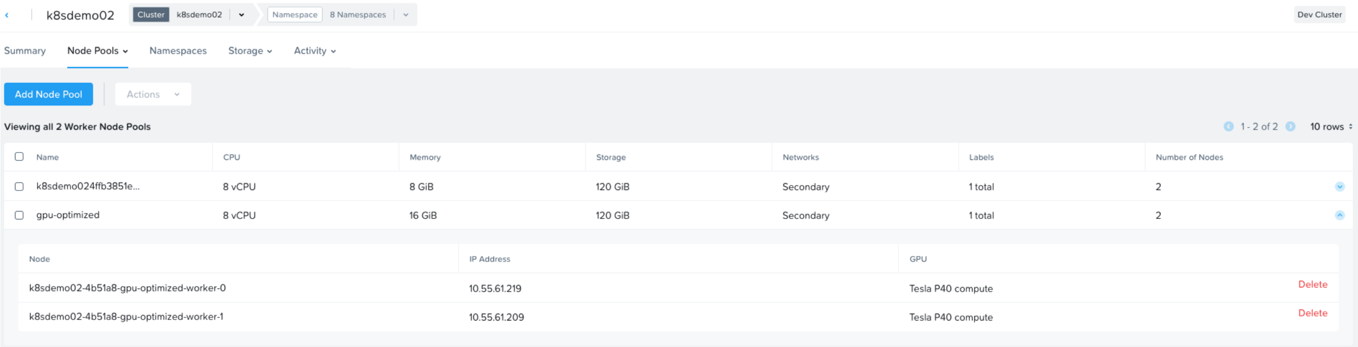
Adding GPUs to a Karbon cluster is accomplished through the creation of a new worker node pool with the GPU feature enabled.

The screenshots below were captured after walking through the following blog `Configuring GPU Operators on Karbon`: https://portal.nutanix.com/page/documents/details?targetId=Karbon-v2_4:kar-karbon-gpu-configure-t.html

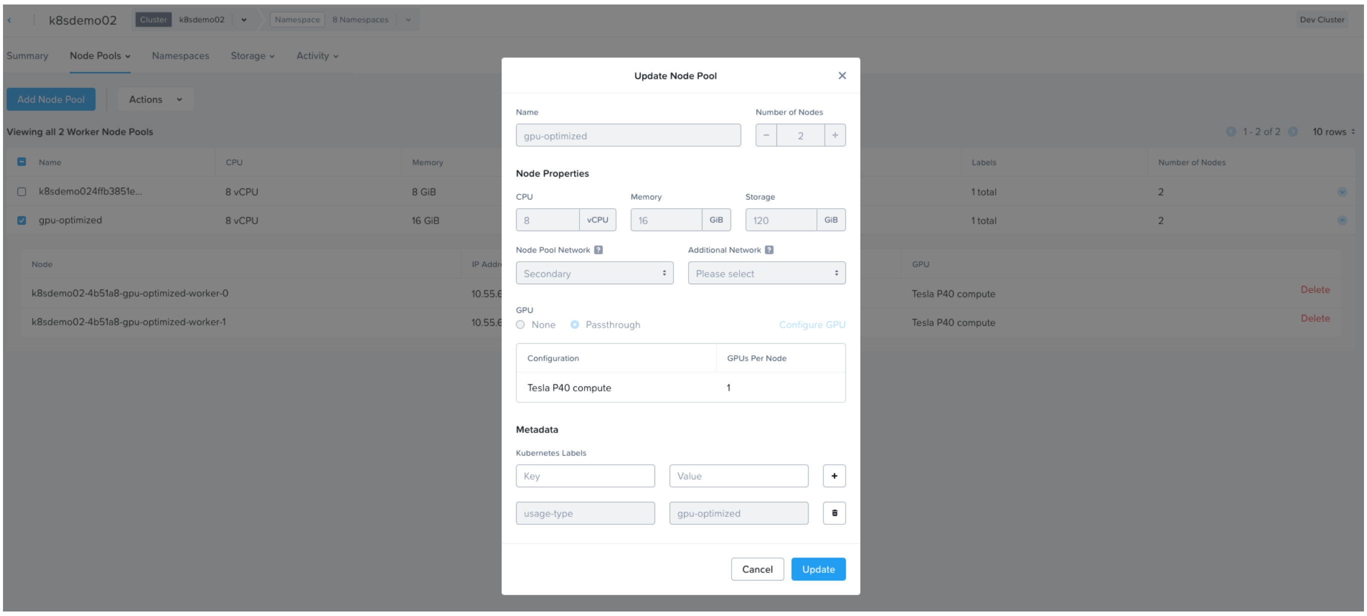
Below is example screenshot of Prism Central View of Hardware with GPU Devices Configured



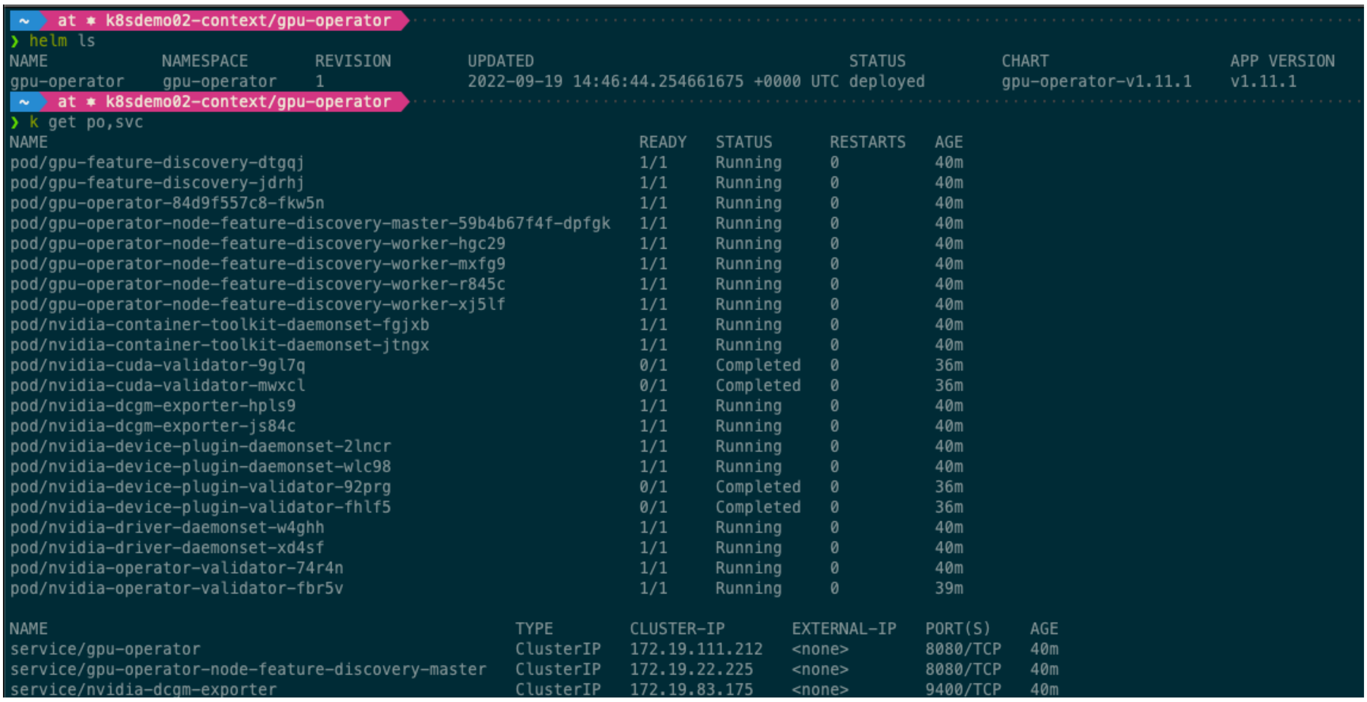
Below is example screenshot of Karbon Worker Node Pool with GPU enabled nodes



Below is example screenshot of Karbon Worker Node with GPU Passthrough Configured



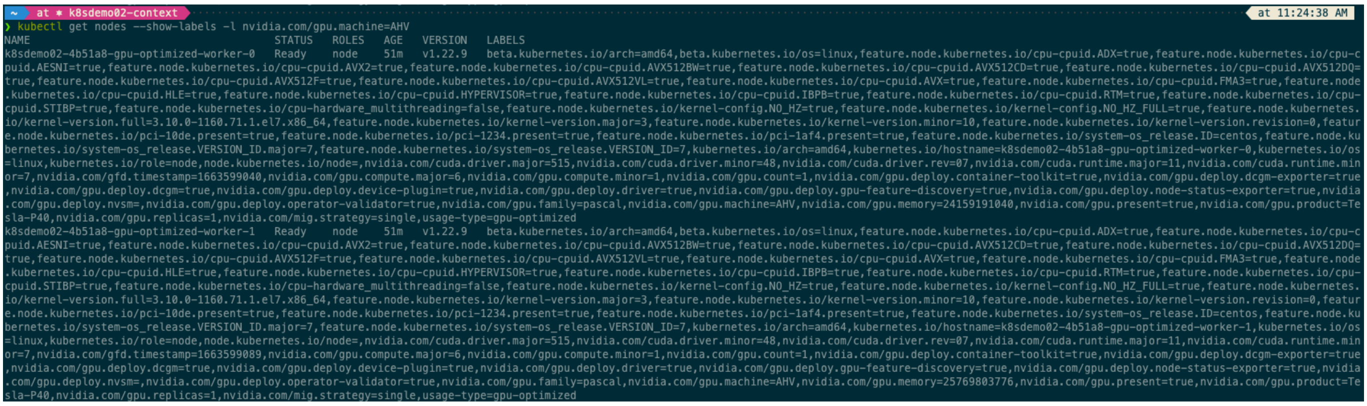
Below is kubeconfig view of nvidia gpu device operator running successfully after being installed via helm



```
~  at ✱ k8sdemo02-context/gpu-operator
❯ helm ls
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART                   APP VERSION
gpu-operator    gpu-operator    1               2022-09-19 14:46:44.254661675 +0000 UTC deployed        gpu-operator-v1.11.1    v1.11.1
~  at ✱ k8sdemo02-context/gpu-operator
❯ k get po,svc
NAME                                                              READY   STATUS      RESTARTS   AGE
pod/gpu-feature-discovery-dtgqj                                   1/1     Running     0          40m
pod/gpu-feature-discovery-jdrhj                                   1/1     Running     0          40m
pod/gpu-operator-84d9f557c8-fkw5n                                 1/1     Running     0          40m
pod/gpu-operator-node-feature-discovery-master-59b4b67f4f-dpfgk   1/1     Running     0          40m
pod/gpu-operator-node-feature-discovery-worker-hgc29              1/1     Running     0          40m
pod/gpu-operator-node-feature-discovery-worker-mxfg9              1/1     Running     0          40m
pod/gpu-operator-node-feature-discovery-worker-r845c              1/1     Running     0          40m
pod/gpu-operator-node-feature-discovery-worker-xj5lf              1/1     Running     0          40m
pod/nvidia-container-toolkit-daemonset-fgjxb                      1/1     Running     0          40m
pod/nvidia-container-toolkit-daemonset-jtngx                      1/1     Running     0          40m
pod/nvidia-cuda-validator-9gl7q                                   0/1     Completed   0          36m
pod/nvidia-cuda-validator-mwxcl                                   0/1     Completed   0          36m
pod/nvidia-dcgm-exporter-hpls9                                    1/1     Running     0          40m
pod/nvidia-dcgm-exporter-js84c                                    1/1     Running     0          40m
pod/nvidia-device-plugin-daemonset-2lncr                          1/1     Running     0          40m
pod/nvidia-device-plugin-daemonset-wlc98                          1/1     Running     0          40m
pod/nvidia-device-plugin-validator-92prg                          0/1     Completed   0          36m
pod/nvidia-device-plugin-validator-fhlf5                          0/1     Completed   0          36m
pod/nvidia-driver-daemonset-w4ghh                                 1/1     Running     0          40m
pod/nvidia-driver-daemonset-xd4sf                                 1/1     Running     0          40m
pod/nvidia-operator-validator-74r4n                               1/1     Running     0          40m
pod/nvidia-operator-validator-fbr5v                               1/1     Running     0          39m

NAME                                                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/gpu-operator                                 ClusterIP   172.19.111.212  <none>        8080/TCP   40m
service/gpu-operator-node-feature-discovery-master   ClusterIP   172.19.22.225   <none>        8080/TCP   40m
service/nvidia-dcgm-exporter                         ClusterIP   172.19.83.175   <none>        9400/TCP   40m
```

Below is kubeconfig view of automatic node labeling that occurs post install of GPU operators. GPU device (much like other devices like FPGA) operators leverage the kubernetes device plugin framework as a means of scheduling workings on special purpose hardware - https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/#node-labeller



```
~  at ✱ k8sdemo02-context                                                                                     at 11:24:38 AM
❯ kubectl get nodes --show-labels -l nvidia.com/gpu.machine=AHV
NAME                                    STATUS   ROLES   AGE   VERSION   LABELS
k8sdemo02-4b51a8-gpu-optimized-worker-0   Ready    node    51m   v1.22.9   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,feature.node.kubernetes.io/cpu-cpuid.ADX=true,feature.node.kubernetes.io/cpu-c
puid.AESNI=true,feature.node.kubernetes.io/cpu-cpuid.AVX2=true,feature.node.kubernetes.io/cpu-cpuid.AVX512BW=true,feature.node.kubernetes.io/cpu-cpuid.AVX512CD=true,feature.node.kubernetes.io/cpu-cpuid.AVX512DQ=
true,feature.node.kubernetes.io/cpu-cpuid.AVX512F=true,feature.node.kubernetes.io/cpu-cpuid.AVX512VL=true,feature.node.kubernetes.io/cpu-cpuid.AVX=true,feature.node.kubernetes.io/cpu-cpuid.FMA3=true,feature.node
.kubernetes.io/cpu-cpuid.HLE=true,feature.node.kubernetes.io/cpu-cpuid.HYPERVISOR=true,feature.node.kubernetes.io/cpu-cpuid.IBPB=true,feature.node.kubernetes.io/cpu-cpuid.RTM=true,feature.node.kubernetes.io/cpu-
cpuid.STIBP=true,feature.node.kubernetes.io/cpu-hardware_multithreading=false,feature.node.kubernetes.io/kernel-config.NO_HZ=true,feature.node.kubernetes.io/kernel-config.NO_HZ_FULL=true,feature.node.kubernetes.
io/kernel-version.full=3.10.0-1160.71.1.el7.x86_64,feature.node.kubernetes.io/kernel-version.major=3,feature.node.kubernetes.io/kernel-version.minor=10,feature.node.kubernetes.io/kernel-version.revision=0,featur
e.node.kubernetes.io/pci-10de.present=true,feature.node.kubernetes.io/pci-1234.present=true,feature.node.kubernetes.io/pci-1af4.present=true,feature.node.kubernetes.io/system-os_release.ID=centos,feature.node.ku
bernetes.io/system-os_release.VERSION_ID.major=7,feature.node.kubernetes.io/system-os_release.VERSION_ID=7,kubernetes.io/arch=amd64,kubernetes.io/hostname=k8sdemo02-4b51a8-gpu-optimized-worker-0,kubernetes.io/os
=linux,kubernetes.io/role=node,node.kubernetes.io/node=,nvidia.com/cuda.driver.major=515,nvidia.com/cuda.driver.minor=48,nvidia.com/cuda.driver.rev=07,nvidia.com/cuda.runtime.major=11,nvidia.com/cuda.runtime.min
or=7,nvidia.com/gfd.timestamp=1663599040,nvidia.com/gpu.compute.major=6,nvidia.com/gpu.compute.minor=1,nvidia.com/gpu.count=1,nvidia.com/gpu.deploy.container-toolkit=true,nvidia.com/gpu.deploy.dcgm-exporter=true
,nvidia.com/gpu.deploy.dcgm=true,nvidia.com/gpu.deploy.device-plugin=true,nvidia.com/gpu.deploy.driver=true,nvidia.com/gpu.deploy.gpu-feature-discovery=true,nvidia.com/gpu.deploy.node-status-exporter=true,nvidia
.com/gpu.deploy.nvsm=,nvidia.com/gpu.deploy.operator-validator=true,nvidia.com/gpu.family=pascal,nvidia.com/gpu.machine=AHV,nvidia.com/gpu.memory=24159191040,nvidia.com/gpu.present=true,nvidia.com/gpu.product=Te
sla-P40,nvidia.com/gpu.replicas=1,nvidia.com/mig.strategy=single,usage-type=gpu-optimized
k8sdemo02-4b51a8-gpu-optimized-worker-1   Ready    node    51m   v1.22.9   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,feature.node.kubernetes.io/cpu-cpuid.ADX=true,feature.node.kubernetes.io/cpu-c
puid.AESNI=true,feature.node.kubernetes.io/cpu-cpuid.AVX2=true,feature.node.kubernetes.io/cpu-cpuid.AVX512BW=true,feature.node.kubernetes.io/cpu-cpuid.AVX512CD=true,feature.node.kubernetes.io/cpu-cpuid.AVX512DQ=
true,feature.node.kubernetes.io/cpu-cpuid.AVX512F=true,feature.node.kubernetes.io/cpu-cpuid.AVX512VL=true,feature.node.kubernetes.io/cpu-cpuid.AVX=true,feature.node.kubernetes.io/cpu-cpuid.FMA3=true,feature.node
.kubernetes.io/cpu-cpuid.HLE=true,feature.node.kubernetes.io/cpu-cpuid.HYPERVISOR=true,feature.node.kubernetes.io/cpu-cpuid.IBPB=true,feature.node.kubernetes.io/cpu-cpuid.RTM=true,feature.node.kubernetes.io/cpu-
cpuid.STIBP=true,feature.node.kubernetes.io/cpu-hardware_multithreading=false,feature.node.kubernetes.io/kernel-config.NO_HZ=true,feature.node.kubernetes.io/kernel-config.NO_HZ_FULL=true,feature.node.kubernetes.
io/kernel-version.full=3.10.0-1160.71.1.el7.x86_64,feature.node.kubernetes.io/kernel-version.major=3,feature.node.kubernetes.io/kernel-version.minor=10,feature.node.kubernetes.io/kernel-version.revision=0,featur
e.node.kubernetes.io/pci-10de.present=true,feature.node.kubernetes.io/pci-1234.present=true,feature.node.kubernetes.io/pci-1af4.present=true,feature.node.kubernetes.io/system-os_release.ID=centos,feature.node.ku
bernetes.io/system-os_release.VERSION_ID.major=7,feature.node.kubernetes.io/system-os_release.VERSION_ID=7,kubernetes.io/arch=amd64,kubernetes.io/hostname=k8sdemo02-4b51a8-gpu-optimized-worker-1,kubernetes.io/os
=linux,kubernetes.io/role=node,node.kubernetes.io/node=,nvidia.com/cuda.driver.major=515,nvidia.com/cuda.driver.minor=48,nvidia.com/cuda.driver.rev=07,nvidia.com/cuda.runtime.major=11,nvidia.com/cuda.runtime.min
or=7,nvidia.com/gfd.timestamp=1663599089,nvidia.com/gpu.compute.major=6,nvidia.com/gpu.compute.minor=1,nvidia.com/gpu.count=1,nvidia.com/gpu.deploy.container-toolkit=true,nvidia.com/gpu.deploy.dcgm-exporter=true
,nvidia.com/gpu.deploy.dcgm=true,nvidia.com/gpu.deploy.device-plugin=true,nvidia.com/gpu.deploy.driver=true,nvidia.com/gpu.deploy.gpu-feature-discovery=true,nvidia.com/gpu.deploy.node-status-exporter=true,nvidia
.com/gpu.deploy.nvsm=,nvidia.com/gpu.deploy.operator-validator=true,nvidia.com/gpu.family=pascal,nvidia.com/gpu.machine=AHV,nvidia.com/gpu.memory=25769803776,nvidia.com/gpu.present=true,nvidia.com/gpu.product=Te
sla-P40,nvidia.com/gpu.replicas=1,nvidia.com/mig.strategy=single,usage-type=gpu-optimized
```

There are many resources that could be referenced for understanding AHV GPU Requirements, Karbon Capabilities/Limitations and step-by-step on how to enable below:

- Kubernetes Device Plugin Framework: https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/device-plugins/
    - Device Plugin Examples: https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/device-plugins/#examples
    - Sheduling GPUs: https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/
- Supported GPUs: https://portal.nutanix.com/page/documents/details?targetId=AHV-Admin-Guide-v6_5:ahv-gpu-support-on-ahv-c.html
- GPU Pass-Through for Guest VMs: https://portal.nutanix.com/page/documents/details?targetId=AHV-Admin-Guide-v6_5:ahv-gpu-passthrough-for-guest-vms-intro-c.html

4. Leveraging Application Consistent Snapshots on Nutanix Volumes

It was also determined that there are efforts to enable Native Application Consistent Snapshots within Nutanix Data Services (e.g., Volumes and/or Files) and is currently on Roadmap. In addition, Volume Cloning via CSI Driver for Nutanix Files is also currently on Roadmap

As an interim solution, third-party partner solutions such as Kasten K10 and S3 via Nutanix Objects could be leveraged to achieve the same objective.

Below are examples:

- How to Protect Cloud Native Application Data with Kasten K10 and Nutanix Karbon: https://www.kasten.io/kubernetes/resources/blog/protect-cloud-native-appdata-kasten-k10-nutanix-karbon
- Application Consistent Backups with Kasten Kanister:
  - MongoDB Backup: https://docs.kasten.io/latest/kanister/mongodb/install_app_cons.html
  - PostGreSQL Backup: https://docs.kasten.io/latest/kanister/postgresql/install_app_cons.html

5. How can we automate the ETCD Backups (without manually logging into etcd node)

- Prerequisites:

  - Nutanix Karbon kubectl/krew plugin - https://github.com/nutanix/kubectl-karbon/blob/main/README.md
  - kubectl

- Procedures below are in alignment with documented ETCD Backup procedure: https://portal.nutanix.com/page/documents/details?targetId=Karbon-v2_4:kar-karbon-backing-up-etcd.html

```
## Set Prism Central and Target Karbon Cluster Details
PC_IP_ADDRESS=10.38.11.9
KARBON_CLUSTER=kalm-main-11-1
PC_USER=admin
PC_PASSWORD=<password>

## Get Karbon Kubeconfig File and SSH File
KARBON_PASSWORD=${PC_PASSWORD} kubectl-karbon login -k --server
${PC_IP_ADDRESS} --cluster ${KARBON_CLUSTER} --user ${PC_USER} --
kubeconfig ~/.kube/${KARBON_CLUSTER}.cfg --force --ssh-file
export KUBECONFIG=~/.kube/${KARBON_CLUSTER}.cfg

## Set ETCD Variables needed for etcdctl
export ETCD_IP_0=$(kubectl get ep -n kube-system etcd -o
jsonpath='{.subsets[].addresses[0].ip}')
export ETCD_IP_1=$(kubectl get ep -n kube-system etcd -o
jsonpath='{.subsets[].addresses[1].ip}')
export ETCD_IP_2=$(kubectl get ep -n kube-system etcd -o
jsonpath='{.subsets[].addresses[2].ip}')
export ETCDCTL_CACERT=/var/nutanix/etc/etcd/ssl/ca.pem
export ETCDCTL_CERT=/var/nutanix/etc/etcd/ssl/peer.pem
export ETCDCTL_KEY=/var/nutanix/etc/etcd/ssl/peer-key.pem
```

```
export
ETCDCTL_ENDPOINTS="https://$ETCD_IP_0:2379,https://$ETCD_IP_1:2379,https:/
/$ETCD_IP_2:2379"

## Set Alias for handling subsequent calls made directly from etcdctl
alias etcdctl='ssh -i ~/.ssh/${KARBON_CLUSTER} nutanix@${ETCD_IP_0} -C
"sudo ETCDCTL_API=3 ETCDCTL_CACERT=${ETCDCTL_CACERT}
ETCDCTL_CERT=${ETCDCTL_CERT} ETCDCTL_KEY=${ETCDCTL_KEY} etcdctl"'

## Validate Alias and ETCDCTL is setup correctly
etcdctl --endpoints=${ETCDCTL_ENDPOINTS} endpoint status --write-out=table

❯ etcdctl --endpoints=${ETCDCTL_ENDPOINTS} endpoint status --write-
out=table
+---------------------------+------------------+---------+---------+-------
----+------------+-----------+------------+--------------------+--------+
|         ENDPOINT          |        ID        | VERSION | DB SIZE | IS
LEADER | IS LEARNER | RAFT TERM | RAFT INDEX | RAFT APPLIED INDEX | ERRORS
|
+---------------------------+------------------+---------+---------+-------
----+------------+-----------+------------+--------------------+--------+
| https://10.38.11.54:2379 | 11e2c57d5c8f22f8 |  3.3.17 |   25 MB |
false |      false |         2 |    1952796 |                  0 |
|
| https://10.38.11.31:2379 |  ee1d483e0a639cf |  3.3.17 |   25 MB |
true |      false |         2 |    1952796 |                  0 |        |
| https://10.38.11.35:2379 | 228ae1995af35bad |  3.3.17 |   24 MB |
false |      false |         2 |    1952796 |                  0 |
|
+---------------------------+------------------+---------+---------+-------
----+------------+-----------+------------+--------------------+--------+

## Snapshot DB on initial instance

etcdctl --endpoints https://$ETCD_IP_0:2379 snapshot save
/root/snapshot.db

## Write-out status to validate that snapshot db looks good

etcdctl --endpoints https://$ETCD_IP_0:2379 snapshot status
/root/snapshot.db --write-out=table

❯ etcdctl --endpoints https://$ETCD_IP_0:2379 snapshot status
/root/snapshot.db --write-out=table
+----------+----------+------------+------------+
|   HASH   | REVISION | TOTAL KEYS | TOTAL SIZE |
+----------+----------+------------+------------+
| 55403e8d |  1744319 |       2494 |      25 MB |
+----------+----------+------------+------------+
```

6. How can you configure Splunk Log Forwarding on Karbon?

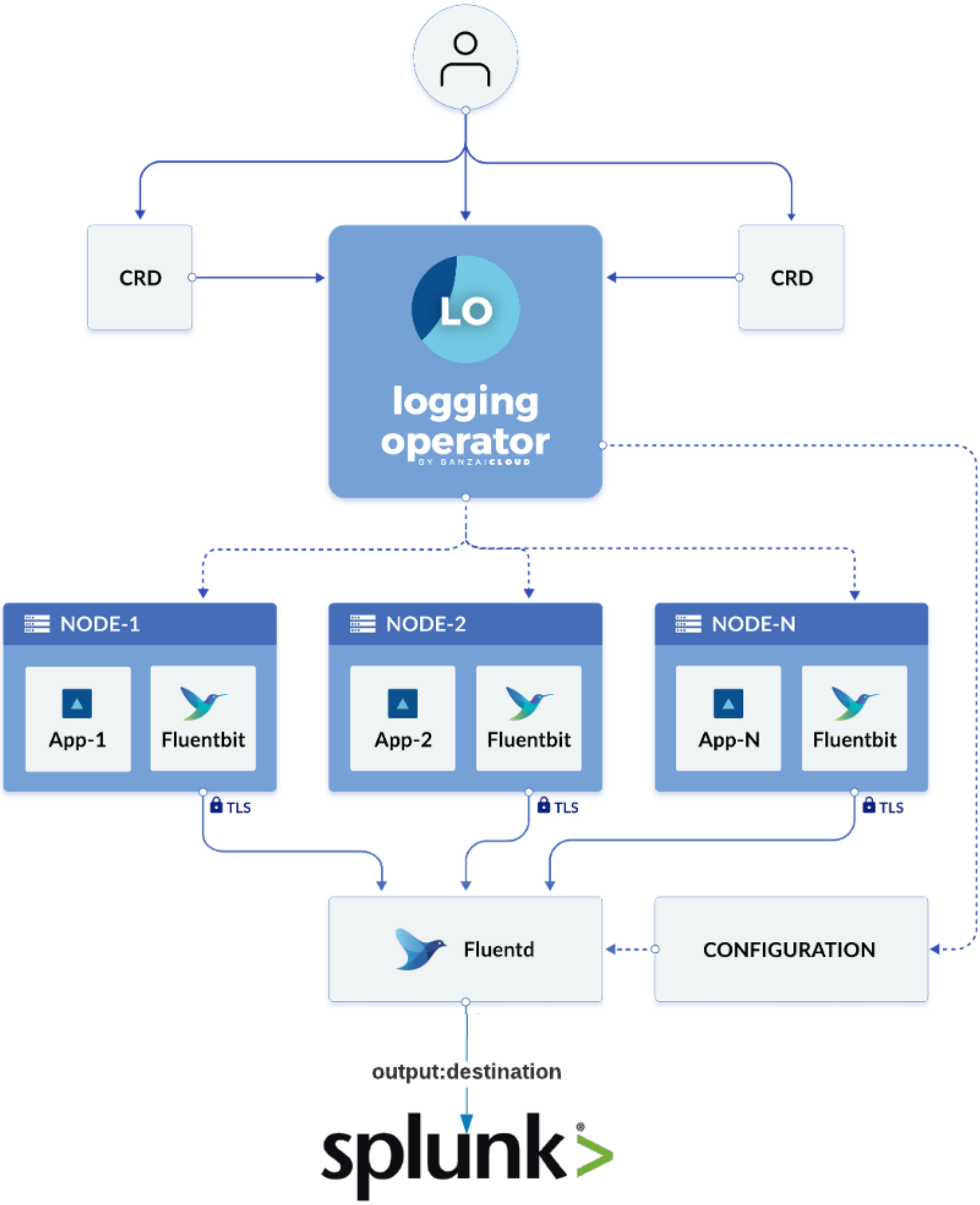
> NOTE: `This is NOT intended for production use cases but rather for information purposes / functional integration testing. This is not a certified solution`

Banzai Cloud Logging Operator - v2.5+ 20k Foot Overview

The logging solution discussed below leverages the `Banzai Cloud Logging operator` to automate the deployment and configuration of a `Kubernetes` logging pipeline.

Similar to that of `NKE`, a `Fluent Bit DaemonSet` is deployed and configured on every node to collect container and application logs from the node file system.

`Fluent Bit` queries the Kubernetes API and enriches the logs with metadata about the pods, and transfers both the logs and the metadata to `Fluentd`. `Fluentd` receives, filters, and transfers logs to multiple `Outputs` (i.e., `Splunk`).



There are four key concepts to understand the Banzai Cloud logging Architecture:

1. `Outputs` are a configuration resource that determine a `destination` for collected logs. This is where settings for aggregators such as `Splunk`, `ElasticSearch`, `Kafka`, etc. are stored. `Outputs` are `namespaced` resources.
2. `Flows` are a configuration resource that determine `collection`, `filtering`, and `destination` rules for logs. It is within a flow that one will configure what logs to collect, how to mutate or filter them, and which Outputs to send the logs to. `Flows` are `namespaced` resources, and can connect either to an `Output` in the `same namespace`, or a `ClusterOutput`.
3. `ClusterOutputs` serve the same functionality as `Outputs`, except they are a `cluster-scoped` resource. `ClusterOutputs` are necessary when collecting logs `cluster-wide`, or if you wish to provide an `Output` to `all namespaces` in your `cluster`.

4. `ClusterFlows` serve the same function as `Flows`, but at the `cluster level`. They are used to configure `log collection` for an `entire cluster`, instead of on a per-namespace level. `ClusterFlows` are also where `mutations` and `filters` are defined, same as `Flows` (in functionality).

> NOTE: `This is NOT intended for production use cases but rather for functional integration testing.`

For additional details around logging architecture, see:

- https://banzaicloud.com/docs/one-eye/logging-operator/#architecture
- https://docs.fluentbit.io/manual/concepts/key-concepts
- https://banzaicloud.com/docs/one-eye/logging-operator/quickstarts/splunk/
- https://docs.fluentbit.io/manual/pipeline/outputs/splunk

Install the Logging Operator and Splunk Operator and Configure

1. Install the Banzai Cloud Logging operator via Helm - https://banzaicloud.com/docs/one-eye/logging-operator/
2. Install the Splunk Instance via Operator and Configure Logging - https://banzaicloud.com/docs/one-eye/logging-operator/quickstarts/splunk/

> Small workaround: update secret name to be `splunk-single-standalone-secret-v1` as doc was incorrect

```
HEC_TOKEN=$(kubectl get secret -n logging splunk-single-standalone-secret-v1 -o jsonpath='{.data.hec_token}' | base64 --decode)
kubectl -n logging get secret splunk-single-standalone-secret-v1 -o jsonpath='{.data.password}' | base64 --decode
```

1. Use the following command to retrieve the password of the admin user:

`kubectl -n logging get secret splunk-single-standalone-secret-v1 -o jsonpath='{.data.password}' | base64 --decode && echo`

Validate that Application Generated Logs are forwarding to Splunk

1. Open the Splunk dashboard in your browser: http://localhost:8000

- Use the following command to retrieve the password of the admin user:

```
kubectl -n logging get secret splunk-single-standalone-secret-v1 -o jsonpath='{.data.password}' | base64 --decode && echo
```

1. Navigate to `Search & Reporting` and enter `"kubernetes.namespace_name"="logging-demo"` as search query. You should the sample log messages from the logging demo application.

Troubleshooting

1. Validate the `Status` and `Problems` fields of target `Outputs` and `Flows` are healthy

Check the status of your resources. All custom resources have a `Status` and a `Problems` field. In a healthy system, the `Problems` field of the resources is empty, for example:

```
$ kubectl get output,flow -n logging
NAME                                              ACTIVE   PROBLEMS
output.logging.banzaicloud.io/splunk-output    true

NAME                                          ACTIVE   PROBLEMS
flow.logging.banzaicloud.io/splunk-flow    true
```

1. Navigate to Splunk HTTP Event Collector (http://localhost:8000/en-US/manager/search/http-eventcollector) and Verify HEC Token is correct.

2. Check Fluent Bit Configuration

```
❯ kubectl get ds -n logging
NAME                                 DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR   AGE
default-logging-simple-fluentbit   7         7         7       7
7           <none>          34m

❯ kubectl get secrets default-logging-simple-fluentbit -o jsonpath="
{.data['fluent-bit\.conf']}" -n logging  | base64 -d

[SERVICE]
    Flush        1
    Grace        5
    Daemon       Off
    Log_Level    info
    Parsers_File parsers.conf
    Coro_Stack_Size    24576
    storage.path  /buffers

[INPUT]
    Name         tail
    DB  /tail-db/tail-containers-state.db
    DB.locking  true
    Mem_Buf_Limit  5MB
    Parser  docker
    Path  /var/log/containers/*.log
    Refresh_Interval  5
    Skip_Long_Lines  On
    Tag  kubernetes.*
[FILTER]
    Name        kubernetes
    Buffer_Size  0
```

```
    Kube_CA_File  /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
    Kube_Tag_Prefix  kubernetes.var.log.containers
    Kube_Token_File  /var/run/secrets/kubernetes.io/serviceaccount/token
    Kube_URL  https://kubernetes.default.svc:443
    Match  kubernetes.*
    Merge_Log  On
    Use_Kubelet  Off

[OUTPUT]
    Name          forward
    Match         *
    Host          default-logging-simple-fluentd.logging.svc.cluster.local
    Port          24240

    Retry_Limit  False
```

1. Check Fluentd Configuration

```
❯ kubectl get sts,secrets -n logging -l
app.kubernetes.io/component=fluentd
NAME                                             READY   AGE
statefulset.apps/default-logging-simple-fluentd   1/1     37m

NAME                                        TYPE     DATA   AGE
secret/default-logging-simple-fluentd       Opaque   4      37m
secret/default-logging-simple-fluentd-app   Opaque   1      37m

❯ kubectl get secrets default-logging-simple-fluentd-app -o jsonpath="
{.data['fluentd\.conf']}" -n logging | base64 -d
<source>
  @type forward
  @id main_forward
  bind 0.0.0.0
  port 24240
</source>
<match **>
  @type label_router
  @id main
  metrics false
  <route>
    @label @6bc936470c7f215d44044b63adc490b6
    metrics_labels {"id":"flow:logging:splunk-flow"}
    <match>
      labels app.kubernetes.io/name:log-generator
      namespaces logging
      negate false
    </match>
  </route>
</match>
<label @6bc936470c7f215d44044b63adc490b6>
  <match kubernetes.**>
    @type tag_normaliser
```

```
    @id flow:logging:splunk-flow:0
    format ${namespace_name}.${pod_name}.${container_name}
  </match>
  <filter **>
    @type parser
    @id flow:logging:splunk-flow:1
    key_name log
    remove_key_name_field true
    reserve_data true
    <parse>
      @type nginx
    </parse>
  </filter>
  <match **>
    @type splunk_hec
    @id flow:logging:splunk-flow:output:logging:splunk-output
    hec_host splunk-single-standalone-headless
    hec_port 8088
    hec_token 28F0E8F3-8E5D-A146-FAB5-9F667810FF46
    index main
    insecure_ssl true
    <buffer tag,time>
      @type file
      chunk_limit_size 8MB
      path /buffers/flow:logging:splunk-flow:output:logging:splunk-
output.*.buffer
      retry_forever true
      timekey 10m
      timekey_wait 1m
    </buffer>
    <format>
      @type json
    </format>
  </match>
</label>
<label @ERROR>
  <match **>
    @type null
    @id main-fluentd-error
  </match>
</label>
```

- The `Logging` operator has a builtin mechanism that validates the generated `fluentd` configuration before applying it to fluentd. You should be able to see the `configcheck` pod and it's log output.

```
❯ kubectl get po -l app.kubernetes.io/component=fluentd-configcheck -n
logging
NAME                                                   READY   STATUS
RESTARTS   AGE
default-logging-simple-fluentd-configcheck-e1c61a0f   0/1     Completed
0          17m
```

```
❯ kubectl logs default-logging-simple-fluentd-configcheck-e1c61a0f -n
logging
fluentd -c /fluentd/etc/fluent.conf --dry-run
2022-09-19 16:35:02 +0000 [info]: parsing config file is succeeded
path="/fluentd/etc/fluent.conf"
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-mixin-config-placeholders'
version '0.4.0'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-aws-elasticsearch-
service' version '2.4.1'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-azure-storage-append-
blob' version '0.2.1'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-cloudwatch-logs'
version '0.14.2'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-concat' version
'2.5.0'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-datadog' version
'0.14.1'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-dedot_filter' version
'1.0.0'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-detect-exceptions'
version '0.0.14'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-elasticsearch'
version '5.2.2'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-enhance-k8s-metadata'
version '2.0.0'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-gcs' version '0.4.0'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-gelf-hs' version
'1.0.8'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-geoip' version
'1.3.2'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-grafana-loki' version
'1.2.18'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-grok-parser' version
'2.6.2'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-kafka' version
'0.17.5'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-kinesis' version
'3.4.2'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-kube-events-
timestamp' version '0.1.3'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-kubernetes-metadata-
filter' version '2.5.3'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-kubernetes-sumologic'
version '2.0.0'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-label-router' version
'0.2.10'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-logdna' version
'0.4.0'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-logzio' version
'0.0.21'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-multi-format-parser'
version '1.0.0'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-newrelic' version
```

```
'1.2.1'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-opensearch' version
'1.0.5'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-oss' version '0.0.2'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-parser-logfmt'
version '0.0.0'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-prometheus' version
'2.0.3'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-record-modifier'
version '2.1.0'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-redis' version
'0.3.5'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-remote-syslog'
version '1.1'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-rewrite-tag-filter'
version '2.4.0'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-s3' version '1.6.1'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-splunk-hec' version
'1.2.13'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-sqs' version '3.0.0'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-sumologic_output'
version '1.8.0'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-syslog_rfc5424'
version '0.9.0.rc.8'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-tag-normaliser'
version '0.1.1'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-throttle' version
'0.0.5'
2022-09-19 16:35:02 +0000 [info]: gem 'fluent-plugin-webhdfs' version
'1.5.0'
2022-09-19 16:35:02 +0000 [info]: gem 'fluentd' version '1.14.6'
2022-09-19 16:35:02 +0000 [info]: starting fluentd-1.14.6 as dry run mode
ruby="2.7.6"
2022-09-19 16:35:03 +0000 [warn]: define <match fluent.**> to capture
fluentd logs in top level is deprecated. Use <label @FLUENT_LOG> instead
2022-09-19 16:35:03 +0000 [info]: using configuration file: <ROOT>
  <system>
    rpc_endpoint "127.0.0.1:24444"
    log_level info
    workers 1
  </system>
  <source>
    @type forward
    @id main_forward
    bind "0.0.0.0"
    port 24240
  </source>
  <match **>
    @type label_router
    @id main
    metrics false
    <route>
      @label "@6bc936470c7f215d44044b63adc490b6"
      metrics_labels {"id":"flow:logging:splunk-flow"}
```

```
        <match>
          labels app.kubernetes.io/name:log-generator
          namespaces logging
          negate false
        </match>
      </route>
    </match>
    <label @6bc936470c7f215d44044b63adc490b6>
      <match kubernetes.**>
        @type tag_normaliser
        @id flow:logging:splunk-flow:0
        format "${namespace_name}.${pod_name}.${container_name}"
      </match>
      <filter **>
        @type parser
        @id flow:logging:splunk-flow:1
        key_name "log"
        remove_key_name_field true
        reserve_data true
        <parse>
          @type "nginx"
        </parse>
      </filter>
      <match **>
        @type splunk_hec
        @id flow:logging:splunk-flow:output:logging:splunk-output
        hec_host "splunk-single-standalone-headless"
        hec_port 8088
        hec_token xxxxxx
        index "main"
        insecure_ssl true
        <buffer tag,time>
          @type "file"
          chunk_limit_size 8MB
          path "/buffers/flow:logging:splunk-flow:output:logging:splunk-
output.*.buffer"
          retry_forever true
          timekey 10m
          timekey_wait 1m
        </buffer>
        <format>
          @type "json"
        </format>
      </match>
    </label>
    <label @ERROR>
      <match **>
        @type null
        @id main-fluentd-error
      </match>
    </label>
    <match **>
      @type null
      @id main-no-output
```

```
  </match>
</ROOT>
2022-09-19 16:35:03 +0000 [info]: finished dry run mode
```

- Review Fluentd logs...below is example with invalid splunk service connection details

```
$ kubectl exec -it default-logging-simple-fluentd-0 cat /fluentd/log/out
...
2022-09-19 16:13:45 +0000 [info]: starting fluentd-1.14.6 pid=7
ruby="2.7.6"
2022-09-19 16:13:45 +0000 [info]: spawn command to main:  cmdline=
["/usr/bin/ruby", "-Eascii-8bit:ascii-8bit", "/usr/bin/fluentd", "-o",
"/fluentd/log/out", "--log-rotate-age", "10", "--log-rotate-size",
"10485760", "-c", "/fluentd/etc/fluent.conf", "-p", "/fluentd/plugins", "-
-under-supervisor"]
2022-09-19 16:13:46 +0000 [info]: adding match in @FLUENT_LOG
pattern="fluent.*" type="null"
2022-09-19 16:13:47 +0000 [info]: adding match in @ERROR pattern="**"
type="null"
2022-09-19 16:13:47 +0000 [info]: adding match pattern="**"
type="label_router"
2022-09-19 16:13:47 +0000 [info]: adding match pattern="**" type="null"
2022-09-19 16:13:47 +0000 [info]: adding source type="forward"
2022-09-19 16:13:47 +0000 [info]: #0 starting fluentd worker pid=16 ppid=7
worker=0
2022-09-19 16:13:47 +0000 [info]: #0 [main_forward] listening port
port=24240 bind="0.0.0.0"
2022-09-19 16:13:47 +0000 [info]: #0 fluentd worker is now running
worker=0
2022-09-19 16:36:05 +0000 [info]: #0 fluentd worker is now stopping
worker=0
2022-09-19 16:36:05 +0000 [info]: #0 shutting down fluentd worker worker=0
2022-09-19 16:36:05 +0000 [info]: #0 shutting down input plugin
type=:forward plugin_id="main_forward"
2022-09-19 16:36:05 +0000 [info]: #0 shutting down output plugin
type=:label_router plugin_id="main"
2022-09-19 16:36:05 +0000 [info]: #0 shutting down output plugin
type=:null plugin_id="main-fluentd-log"
2022-09-19 16:36:05 +0000 [info]: #0 shutting down output plugin
type=:null plugin_id="main-fluentd-error"
2022-09-19 16:36:05 +0000 [info]: #0 shutting down output plugin
type=:null plugin_id="main-no-output"
2022-09-19 16:36:06 +0000 [error]: Worker 0 finished unexpectedly with
status 0
2022-09-19 16:36:07 +0000 [info]: adding match in
@6bc936470c7f215d44044b63adc490b6 pattern="kubernetes.**"
type="tag_normaliser"
2022-09-19 16:36:07 +0000 [info]: adding filter in
@6bc936470c7f215d44044b63adc490b6 pattern="**" type="parser"
2022-09-19 16:36:07 +0000 [info]: adding match in
@6bc936470c7f215d44044b63adc490b6 pattern="**" type="splunk_hec"
2022-09-19 16:36:07 +0000 [info]: adding match in @FLUENT_LOG
```

```
pattern="fluent.*" type="null"
2022-09-19 16:36:07 +0000 [info]: adding match in @ERROR pattern="**"
type="null"
2022-09-19 16:36:07 +0000 [info]: adding match pattern="**"
type="label_router"
2022-09-19 16:36:07 +0000 [info]: adding match pattern="**" type="null"
2022-09-19 16:36:07 +0000 [info]: adding source type="forward"
2022-09-19 16:36:07 +0000 [info]: #0 starting fluentd worker pid=21 ppid=7
worker=0
2022-09-19 16:36:07 +0000 [info]: #0 [main_forward] listening port
port=24240 bind="0.0.0.0"
2022-09-19 16:36:07 +0000 [info]: #0 fluentd worker is now running
worker=0
...
```

[Optional] Disable NKE Infra Logging

If you wish to avoid the redundancy/overhead of having the default logging infrastructure deployed with NKE (more specifically the `Elasticsearch & Kibana`), below are following instructions:

1. Disable Elasticsearch and Kibana - Logging Infrastructure

Cheatsheet:

`/home/nutanix/karbon/karbonctl cluster infra-logging disable --cluster-name=cluster-name`

> validate kubernetes cluster before disabling infra logging via kubectl

```
## BEFORE disabling infra logging
❯ kubectl get ds,deploy,sts,pvc -l 'k8s-app in (kibana-
logging,elasticsearch-logging)' -n ntnx-system

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/kibana-logging    1/1     1            1           23d

NAME                                      READY   AGE
statefulset.apps/elasticsearch-logging    1/1     23d

NAME
STATUS   VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS           AGE
persistentvolumeclaim/elasticsearch-logging-data-elasticsearch-logging-0
Bound    pvc-fdf48bf1-1342-4279-92d0-d7dac5b12cd1   112Gi      RWO
default-storageclass   23
```

> disable infra logging via karbonctl and validate

```
## disabling of infra-logging from prism central via karbonctl
$ /home/nutanix/karbon/karbonctl cluster infra-logging disable --cluster-
name karbon-cluster-01

Successfully disabled infra logging: [POST /karbon/v1-
alpha.1/k8s/clusters/{name}/disable-infra-logging][200]
postDisableInfraLoggingOK

## AFTER disabling infra logging. PVC/PV remains in case you wish to re-
enable and have previous data available.
$ kubectl get ds,deploy,sts,pvc -l 'k8s-app in (kibana-
logging,elasticsearch-logging)' -n ntnx-system

NAME
STATUS   VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS           AGE
persistentvolumeclaim/elasticsearch-logging-data-elasticsearch-logging-0
Bound    pvc-fdf48bf1-1342-4279-92d0-d7dac5b12cd1   112Gi      RWO
default-storageclass   23
```