Constructing Heterogeneous K8s control plane with Konnectivity /K8s apiserver network proxy

Tamil Vanan
Tech Lead, Arcesium

kubernetes

Overview

- What?

  What you meant by “Heterogeneous/Remote control Plane”

- Why?

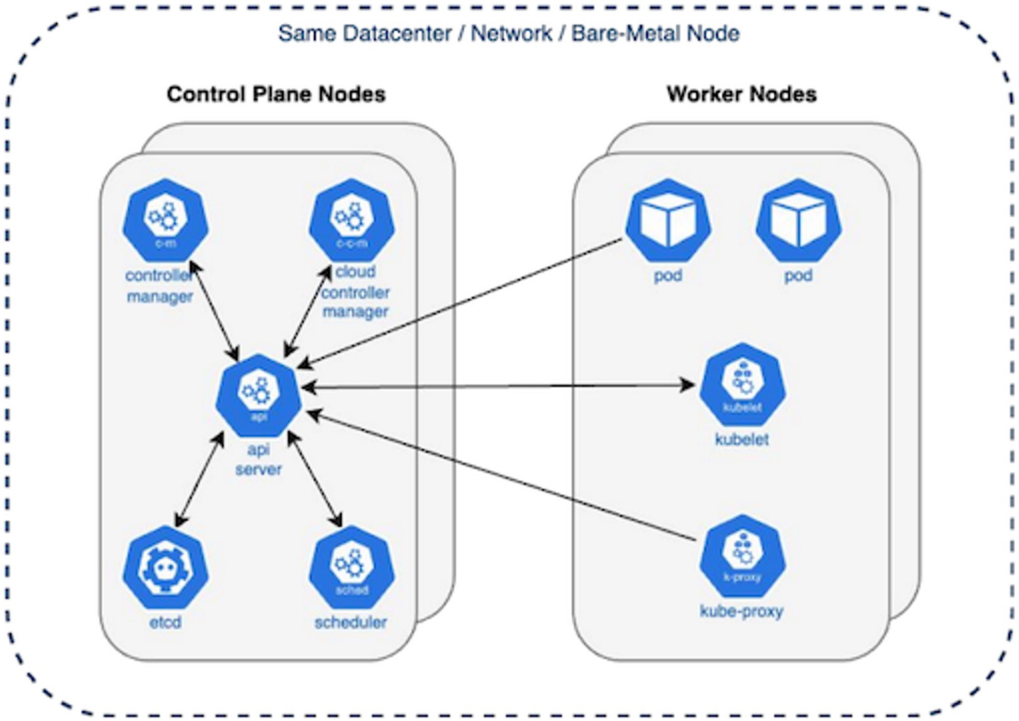
  Example use cases for Heterogeneous control plane

- How ?

  Building blocks and concepts to make this happen

kubernetes

Kubernetes control plane



Communication between the controller < - > worker node

- Bidirectional
- Same datacenter/ Same L2/L3 network domain

Use cases

- Kubernetes at Edge - Worker nodes (e.g. resource-constrained) at edge, control plane in a cloud/datacenter

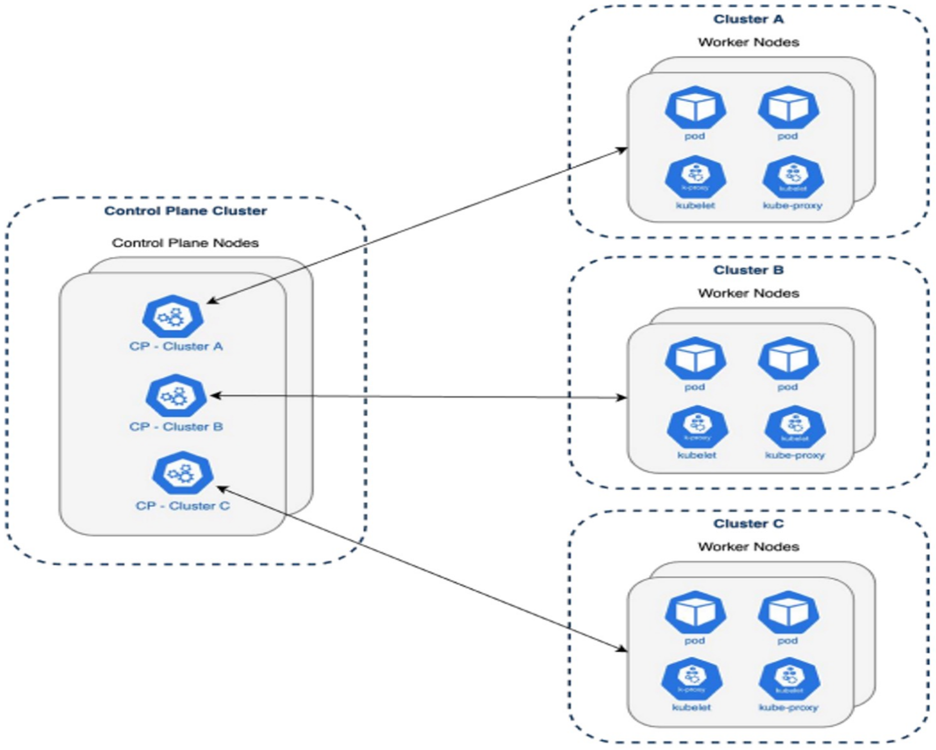
    - Internet of Things (IoT) and Edge Devices
    - Telecommunications and 5G Networks
    - Autonomous Vehicles

- Hybrid Cloud
    - Worker nodes on different platform / cloud than the control plane
    - Easy migration of worker-nodes between platforms

- Co-located control-plane for multiple clusters:
    - Easy operation for 100s / 1000s clusters

kubernetes

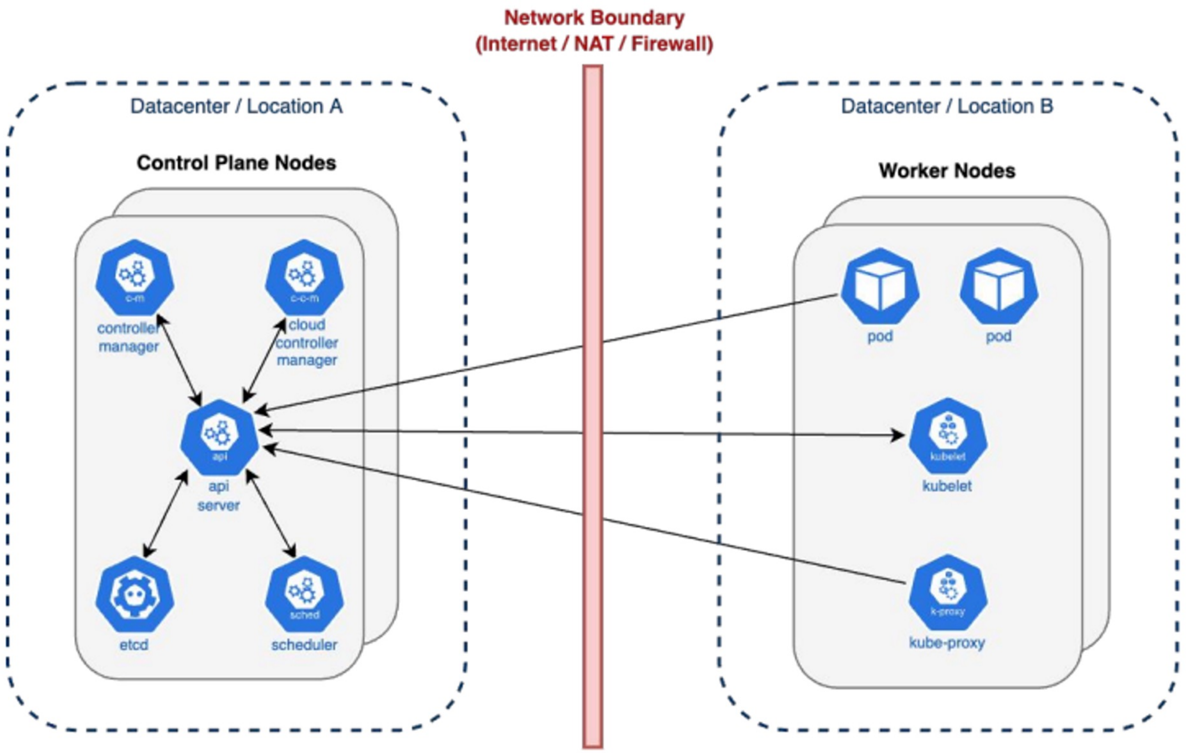
Co-located control-plane for multiple clusters



- Easy operation for 100s / 1000s clusters
- Same control-plane experience across different (hybrid) cloud platforms
- Fast cluster spin-up time - good for temporary / short-lived clusters
- Build your own Kubernetes as a service

kubernetes

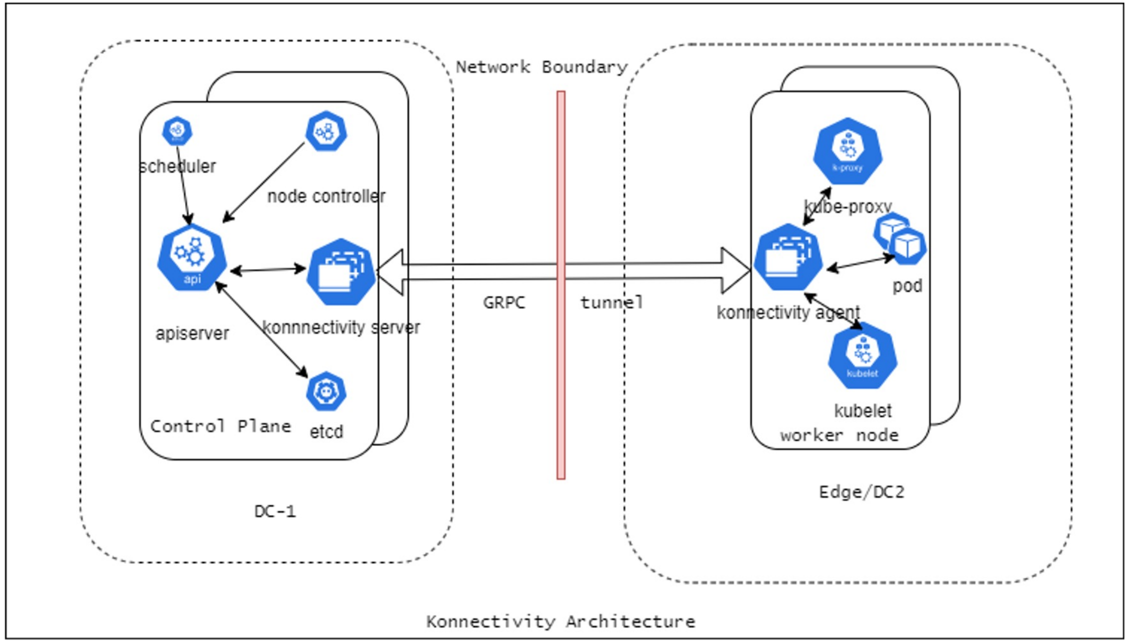
Kubernetes Remote Control plane



- Kubernetes supported SSH tunnels in the past, deprecated at v1.9
- VPN tunnels
- Drawbacks
  - Security implications
  - Vendor lock-in

kubernetes

SIG apiserver-network-proxy



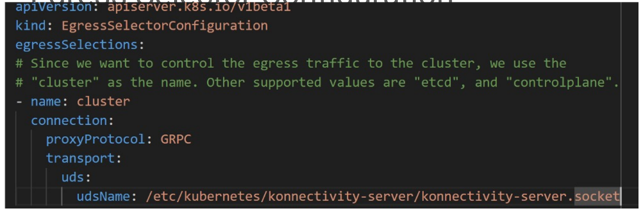
- Agent opens the bi-directional connection to server
- Much like SSH reverse tunnels

kubernetes

Konnectivity components

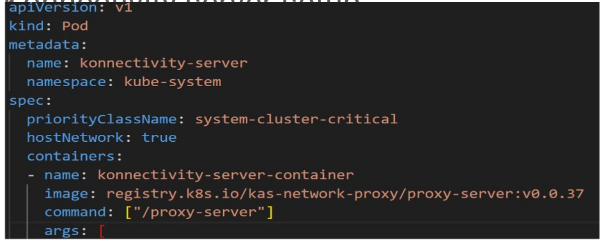
Controller configuration

- Egress Selector Configuration



```yaml
apiVersion: apiserver.k8s.io/v1beta1
kind: EgressSelectorConfiguration
egressSelections:
# Since we want to control the egress traffic to the cluster, we use the
# "cluster" as the name. Other supported values are "etcd", and "controlplane".
- name: cluster
  connection:
    proxyProtocol: GRPC
    transport:
      uds:
        udsName: /etc/kubernetes/konnectivity-server/konnectivity-server.socket
```

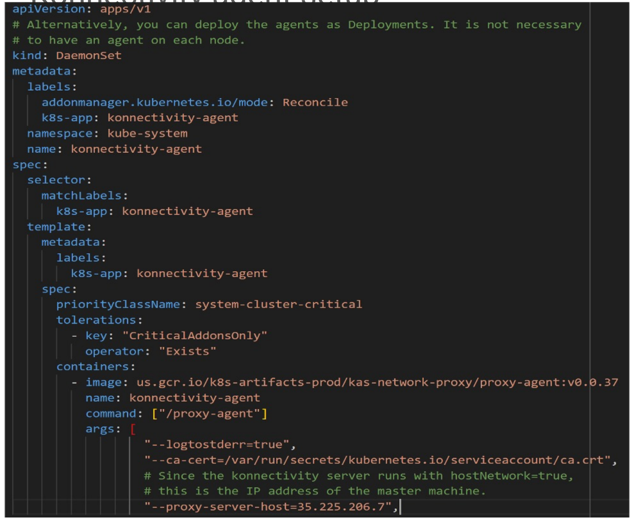
- Konnectivity server setup



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: konnectivity-server
  namespace: kube-system
spec:
  priorityClassName: system-cluster-critical
  hostNetwork: true
  containers:
  - name: konnectivity-server-container
    image: registry.k8s.io/kas-network-proxy/proxy-server:v0.0.37
    command: ["/proxy-server"]
    args: [
```

Worker Node configuration

- Konnectivity agent setup



```yaml
apiVersion: apps/v1
# Alternatively, you can deploy the agents as Deployments. It is not necessary
# to have an agent on each node.
kind: DaemonSet
metadata:
  labels:
    addonmanager.kubernetes.io/mode: Reconcile
    k8s-app: konnectivity-agent
  namespace: kube-system
  name: konnectivity-agent
spec:
  selector:
    matchLabels:
      k8s-app: konnectivity-agent
  template:
    metadata:
      labels:
        k8s-app: konnectivity-agent
    spec:
      priorityClassName: system-cluster-critical
      tolerations:
        - key: "CriticalAddonsOnly"
          operator: "Exists"
      containers:
        - image: us.gcr.io/k8s-artifacts-prod/kas-network-proxy/proxy-agent:v0.0.37
          name: konnectivity-agent
          command: ["/proxy-agent"]
          args: [
                  "--logtostderr=true",
                  "--ca-cert=/var/run/secrets/kubernetes.io/serviceaccount/ca.crt",
                  # Since the konnectivity server runs with hostNetwork=true,
                  # this is the IP address of the master machine.
                  "--proxy-server-host=35.225.206.7",
```

Thank you

- Slack #api-server-network-proxy
- References
  - https://github.com/kubernetes-sigs/apiserver-network-proxy
  - https://kubernetes.io/docs/tasks/extend-kubernetes/setup-konnectivity/

Contact: @tamilhce

Scan to download the slides

kubernetes