

k8s-node-external-ip-watcher

A lightweight Kubernetes node event watcher

Fredrik Steen



\$ finger stone

```
Login: stone                               Name: Fredrik Steen <stone@varnish-software.com>
Directory: /home/stone                     Shell: /bin/zsh
Logged in.
A lot of mail.
Plan:

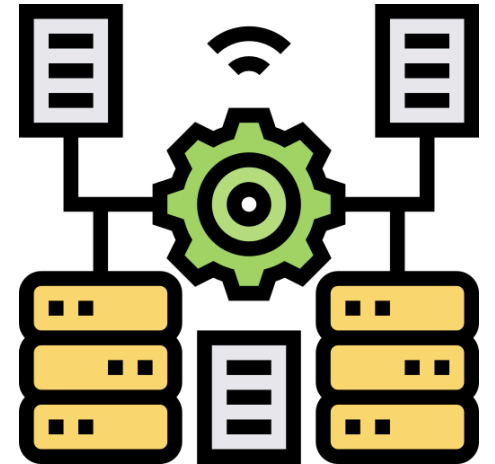
Director of Software Engineering @ Varnish Software https://www.varnish-software.com/

LinkedIn: fredriksteen
GitHub: stone
Home: https://tty.se/

Linux and Open Source DevOps/Platform Engineer/developer since the mid 90s,
PsyTrance-DJ, Beekeeper, Electronics, Microbiology and Beer enthusiast.
```

The Problem

- Some Cloud providers lack UDP load balancing
- External systems need to know the cluster topology
- Manual updates are error-prone and slow
- Need a dynamic way to inform external services about node staticIP
- Need safeguards to avoid misconfigurations
- Must be lightweight and easy to deploy



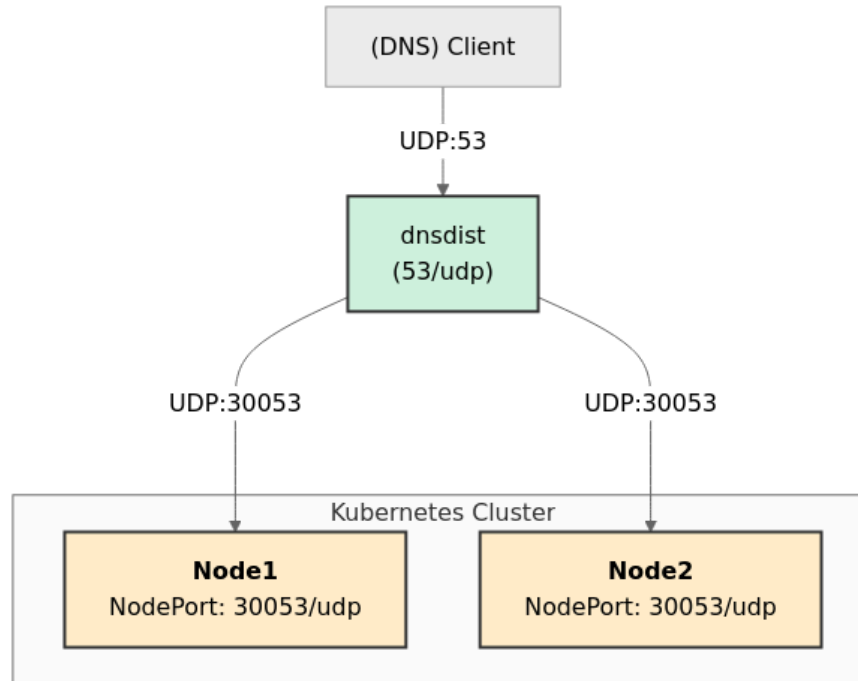
Real-World Example: DNS

- Running an API backed DNS service in kubernetes.
- Using a DNS load balancer (dnssdist)
- Running outside the Kubernetes cluster
- UDP traffic on NodePorts
- A need to update dnssdist backend server list when Kubernetes nodes change

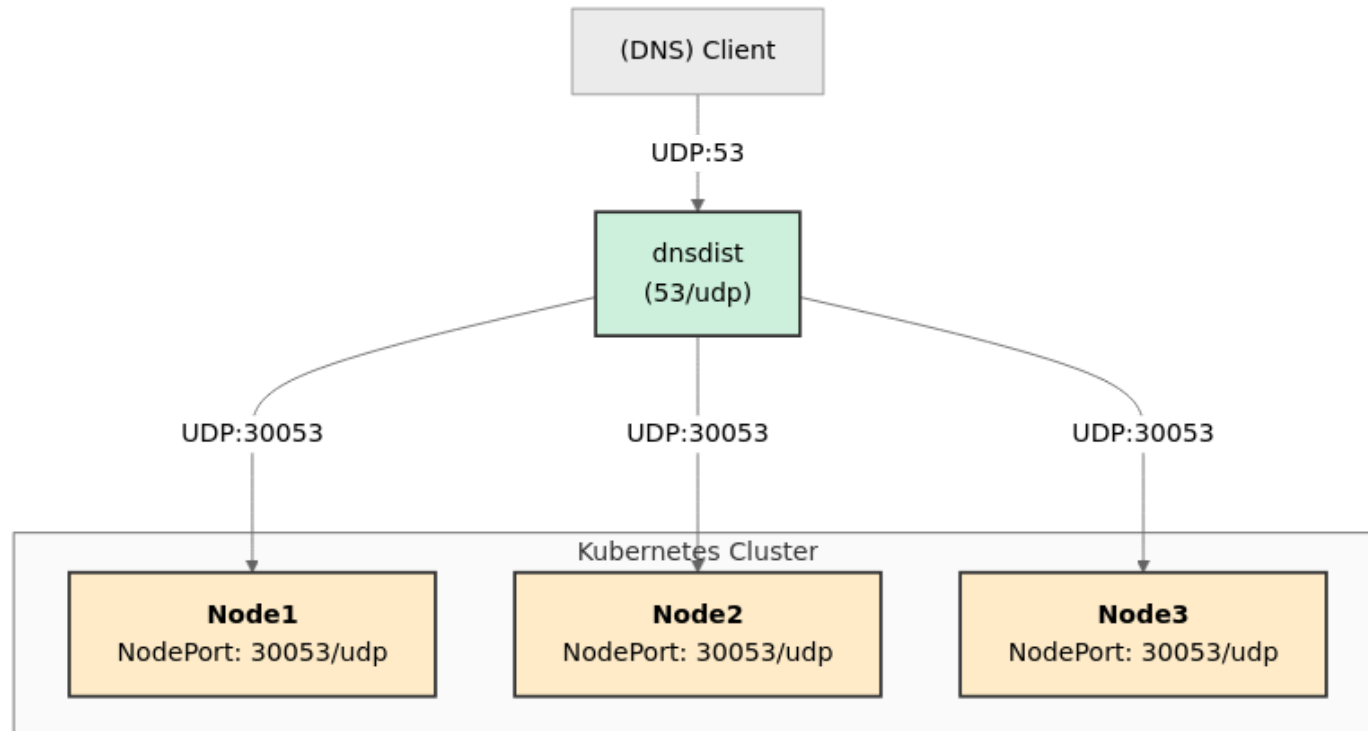
Naive approach:

```
newServer({address="10.0.0.2:53", name="server1"})  
newServer({address="10.0.0.3:53", name="server2"})
```

Real-World Example: DNS



Real-World Example: DNS



The result: k8s-node-external-ip-watcher

A simple Go program, built with Kubernetes Informers.

(with a "pinch" of "not invented here syndrome" going on)

- Name is descriptive, and stuck/suck :)
- Monitors node additions, updates, and deletions
- Renders Go templates with node external IPs
- Executes custom commands after rendering templates
- Support for static IP addresses
- **Safeguard against removal of all nodes**



NOT INVENTED HERE



Automatic update of dnsmasq backends

Config:

```
templatePath: dnsmasq-backend.tpl
outputPath: /etc/dnsmasq/backends.conf
command: /usr/local/bin/reload-dnsmasq.sh
minNodeCount: 3
```

Template:

```
-- Generated: {{ .Timestamp.Format "2006-01-02 15:04:05" }}
{{- range .Nodes }}
newServer({address="{{ .ExternalIP }}:30053", name="{{ .Name }}"})
{{- end }}
```

Implemented using Kubernetes Informers

```
// Attach to kubernetes informer factory
factory := informers.NewSharedInformerFactory(...)
nodeInformer := factory.Core().V1().Nodes().Informer()

// Add event handlers
nodeInformer.AddEventHandler(cache.ResourceEventHandlerFuncs{
    AddFunc: func(obj any) { node := obj.(*corev1.Node) handleNodeEvent("ADD", node) },
    UpdateFunc: func(oldObj, newObj any) {node := newObj.(*corev1.Node) handleNodeEvent("UPDATE", node) },
    DeleteFunc: func(obj any) {node := obj.(*corev1.Node) handleNodeEvent("DELETE", node) },
})
```

Why Informers?

Efficient

- Built-in caching reduces API server load
- Automatic reconnection and resync
- Watch mechanism for "real-time" updates
- Reduced API server load
- Production-ready pattern used throughout Kubernetes ecosystem
- Using github.com/kubernetes/client-go



Cache Sync and Initial State

```
// Start informer
factory.Start(ctx.Done())

// Wait for cache to fully synchronize
w.logger.Info("Waiting for cache sync")
if !cache.WaitForCacheSync(ctx.Done(), nodeInformer.HasSynced) {}

// Get current state of all nodes
if err := w.initialSync(nodeInformer); err != nil {}
```

WaitForCacheSync guarantees state is synchronized (blocking).

Hash-Based Change Detection

```
// Calculate hash of node IPs
currentHash := w.calculateHash()

// Only render if something actually changed
if currentHash == w.lastHash {return nil}

w.lastHash = currentHash
w.logger.Info("Changes detected, rendering template")
```

Prevents redundant template renders and command executions.

Configuration

```
logLevel: info
kubeConfig: /path/to/kubeconfig
templatePath: /etc/template.tpl
outputPath: /etc/backends.conf
command: /usr/local/bin/reload-service.sh
# Safety net
minNodeCount: 2
# Static IPs always included
staticIPs:
  - "192.168.1.100"
  - "192.168.1.101"
# Resync every 5 minutes
resyncInterval: 300
```

/metrics - We all like metrics

```
// Removed Go Built-in metrics for clarity
- k8s_node_watcher_events_total{event_type}
- k8s_node_watcher_renders_total{result}
- k8s_node_watcher_command_executions_total{result}
- k8s_node_watcher_nodes_current
- k8s_node_watcher_start_time_seconds
```

Endpoints:

- `/metrics` - Prometheus metrics endpoint
- `/healthz` - Health check endpoint

Kubernetes setup, RBAC Minimal Permissions

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: node-ip-watcher
rules:
- apiGroups: [""]
  resources: ["nodes"]
  verbs: ["get", "list", "watch"]
```

(Create a ServiceAccount with a RoleBinding to the ClusterRole)

Getting Started

Available on GitHub:

- github.com/stone/k8s-node-external-ip-watcher
- `.deb` , `.rpm` , `.apk` , `.tgz` releases (amd64, arm64, darwin)
- Systemd service units included
- Containers: ghcr.io/stone/k8s-node-external-ip-watcher

```
$ docker run --rm ghcr.io/stone/k8s-node-external-ip-watcher:v0.1.2 -h
Usage of /app/k8s-node-external-ip-watcher:
  -config string
    Path to configuration file (default "config.yaml")
  -kubeconfig string
    Path to kubeconfig file
  -log-level string
    Log level (debug, info, warn, error)
  -metrics-addr string
    Address for metrics and health (default: localhost:8089)
  -output string
    Path to output file
  -template string
    Path to template file
  -version
    Show version and exit
$
```

Demo



Thank you for your attention!

- Questions?
- Contributions welcome!

Contact:

- GitHub: [@stone](#)
- Email: fredrik@tty.se
- Web: <https://tty.se/>