Data Structures

**LabelTrie** = Simple Trie structure on concat labels

label = color: green  
concat\_label = color:green

* Quick by looking at first label type, then value

Stores list of Objects at nodes

**IpTrie** = Trie structure on protocol + Ip ranges

First level in trie = protocol  
Each “ip segment” is a level in the trie

Tcp on 192.168.1.10 = tcp 192 168 1 10

Udp on 192.168.1.15-192.168.1.20 = udp 192 168 1 15:20

Udp on 192.168.0.28-192.168.1.30 =udp 192 168 0:1 28:30

Tcp on 192.168.1.0/29 =tcp 192 168 1 1:6

Max depth of tree = 5

**Store =** store objects in lists given 2 keys

e.g. store.save(id1, id2, obj)

=> key = (id1, id2)

Store[key] = [obj]

# SGIC(Security Group Information Cluster)

sg\_to\_nodes: set Security Group ID -> Set of Nodes  
node\_to\_sgs: set Node ID -> Set of Security Groups

IngressTrie = IpTrie  
EgressTrie = IpTrie

* We have an egress rule, now let us look for a corresponding ingress rule by protocol and IP

check\_sg\_connectivity(nodeName1, nodeName2, connection\_wanted)

Gets SGs, gets their rules, verifies if they can communicate in the right direction.

Reports according to connection\_wanted boolean

# KIC (Kubernetes Information Cluster)

ContainerTrie = labelTrie

* Given a NP we can find all pods that have the label applied to them.

Reachabilitymatrix:

* Kanomatrix
* Dict containers
* Dict policies
* Responsible policies = Store()

e.g. get the policies that allow connection from container1 to container2:

store.get(container1, container2) returns [(pol1, pol2), (pol3, pol4)]

pol1 and pol3 are ingress, pol2 and pol4 are egress.

# The overview:

Watcher = Watches the events

Parser = Creates Container, Policy, SG objects

Analyzer = Analyses the events

KIC = information from Kubernetes level (Uses LabelTrie)

SGIC = information from Security group VM level (Uses IpTrie)

**StartupCheck**:

Check for every existing connection in Kano matrix if SGs comply with them

**AnalyseEvent**:

POLICY:

CREATION:

- Does it create new pod connections?

Yes: Check if SGs are compliant

DELETION:

- Does it remove any pod connections?

No: Report which other policies are responsible for upholding the connection

Yes: Check if SGs are compliant

POD:

CREATION:

- print which node it is deployed on and which pods it can communicate with and according to which NPs

- check if SGs are compliant with new connections

DELETION:

- Print which pod connections are now removed, and which NPs are redundant (We check each NP that uses this label wether another pod still uses it)

- Report any redundant NP as well

- Check if SGs are compliant